JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 24, 1397-1407 (2008)

A Self-adaptive Predictive Policy for Pursuit-evasion Game

ZHEN LUO, QI-XIN CAO AND YAN-ZHENG ZHAO
Research Institute of Robotics
Shanghai Jiaotong University
Shanghai 200240, P.R. China

The proposed self-adaptive predictive pursuing policy consists of an action decision-making procedure and a procedure of adjusting the estimation of evader's action preference. Since correct estimation of opponent's intention would do good to win adversarial games, it introduces the conception of action preference to model opponent's decision-making. Because evader often has different action preference in different situation, to model evader's decision-making, pursuer has to divide the situation space into many categories and provide a set of estimation of evader's action preference for each kind of situation. Pursuer adjusts the estimation of evader's action preference in certain situation by observing evader's action. Action decision-making procedure consists of situation sorting, possible future states computation, payoff evaluation and action selection. Action decision-making is based on the decision tree constructed by expected payoffs. Expected payoffs are integrated from single payoffs. Single payoffs are evaluated by gains of features reflecting adversarial situation. A simulation of middle size soccer robots has been carried out and illustrated that the proposed policy is effective.

Keywords: action preference, payoff function, predictive, pursuit-evasion games, self-adaptive

1. INTRODUCTION

Pursuit-evasion games are important issues. There are many types of pursuit-evasion games, for examples visibility-based pursuit-evasion game [1], the game of multi-pursuers capturing one evader [2], pursuit-evasion game with safety-zone [3, 4], *etc.* It can also be considered that the RoboCup Soccer Robot game is constituted with some pursuit-evasion scenarios. The scenario of a robot team intercepting an opponent dribbling is a typical pursuit-evasion game with safety-zone. Today, RoboCup is a popular game to foster intelligent robotics research by providing a standard problem [5].

Correct speculation of opponent's intention is a key to win an adversarial game. To estimate opponents' intention, some researchers assumed that game players are rational [6, 7]. In such works, the assumption "I KNOW THAT HE KNOWS THAT I KNOW" is granted by the game players. But in many adversarial games, especially today's adversarial games of robot, players are often short of information of their opponents. For example, in the typical pursuit-evasion game of missile interception problem [3, 4], evader is often thought as blind, *i.e.* it is impossible that players of this kind game are rational. As the state-of-the-art of RoboCup soccer robot, "rational" assumption is not always in accord with the fact of soccer robots. For examples, robots of EIGEN MSL team 2006 use Fuzzy Potential Method (FPM) to generate action [8], robots of Philips MSL Team 2006 are reactive [9] and they could not be rational. In fact, even when professional hu-

Received November 24, 2006; revised March 1, 2007; accepted June 1, 2007.
Communicated by Takeshi Tokuyama.

man athletes play adversarial games requiring quick decision-making, such as basketball game, they are often not "rational" and trained to follow "if *X*, then execute action *Y*" rules.

To antagonize with "irrational" opponents, a short-term prediction based pursuing policy for pursuit-evasion games is proposed in [10]. If it is possible, it is useful to predict other's action [11]. The policy proposed in [10] supposes that player assumes that the probability of the opponent executing each action is equal. The opponent's preference for certain action in certain situation is ignored. In real world, agent often follows some given action decision-making rules, *i.e.*, there is a known or potential mapping relationship between situations and actions. For example, robots with reactive intelligent system architecture [12-14] have action preference for each situation.

Obviously, to beat the opponent with action preference, it is helpful to take the action preference of opponent into account. Thus, a self-adaptive predictive pursuing policy is presented. The self-adaptation of it is realized by modifying the estimation of opponents' action preferences in different situations.

The presented self-adaptive method is not same with reinforcement learning. The conception of reinforcement learning is first suggested by Minsky [15]. The basic idea of reinforcement learning is as follows: if an action leads to positive reward, then the possibility of the agent executing the action will be increased; otherwise, if an action leads to negative reward, then the possibility of executing the action would be decreased. As the paper [16] said, a drawback of reinforcement learning is that the learning procedure would be long. On the contrary, the estimation of opponent's action preference in certain situation can be adjusted quickly. For many adversarial games, such as soccer game, it is not a good idea to use the "try and adjust" procedure of reinforcement learning (although reinforcement learning is useful in training soccer robot). Furthermore, in many adversarial games, situation is not so simple as the event of shooting/defending in most time. It is a problem to design a perfect reward evaluation. On the other hand, if situations can be evaluated perfectly, why not use the method that decision is made by evaluating the result of actions directly? With the presented method, designer can focus on how to improve the reward evaluation.

For the great maneuverability of intelligent players in many games, their possible state spaces of long-term running would be extremely huge. So in such games, it is often impractical to do long-term prediction before selecting an action to execute. Furthermore, it is difficult to work out a non-greedy GLOBAL-MAX solution in such pursuit-evasion games. Thus, the presented method only uses short-term information for decision-making.

The basic action decision-making procedure is described in section 2. Section 3 expatiates on the method of action preference estimation. A simulation is presented in section 4. Section 5 concludes.

2. ACTION DECISION-MAKING PROCEDURE

In practice, it is difficult to work out an optimal action solution in pursuit-evasion games with intelligent and highly maneuverable opponents. Player often has to execute an action that seems to be reasonable. Thus, it is reasonable to assume that an agent **P** has

a discrete and finite action space and it justly selects a "rational" action within the limited action space to execute. In fact, for many real systems, their actions or controls are restricted to be a limited number of classes. For example, in several versions of middle size soccer robot intelligent system developed by us, there are only several levels of power in use. Thus, if W is the action space of **P**, $W = \{w_i \mid i = 1, \ldots, N\}$, where w_i denotes action i and N is the number of actions, then the result of **P**'s decision-making is that it selects one action w_j (or action sequence) to execute.

For the proposed predictive policy, the selected action w_j should lead to a "maximal" reward or "minimal" payoff for certain future. Payoff (or reward) is used to indicate the vantage grade of players in an adversarial situation. Since an agent usually does not know whether its opponent is rational or not, it is reasonable to use the aggregate values of payoffs at some future time to make action decision. Here, the aggregate value of payoff is called as expected payoff. Expected payoff is integrated with single payoff. Single payoff can be defined as "in current situation (player **P** is in state s_0 and its opponent **E** is in state s'_0,), the payoff $J(s_0, s'_0, w_i, u_j, t)$ of a player **P** executing action w_i with time t against its opponent **E** executing action u_j". Thus, after time t, expected payoff $E[J(s_0, s'_0, w_i, u, t)]$ of a player **P** executing action w_i in current situation satisfies the following equation:

$$E[J(s_0, s'_0, w_i, u, t)] = \int_U p(u) J(s_0, s'_0, w_i, u, t) du, \tag{1}$$

where U is the action space of player **E**, $p(u)$ is the probability of **E** executing action u in the situation (constituted by **P** state s_0 and **E** state s'_0). It satisfies the equation:

$$\int_U p(u) du = 1. \tag{2}$$

Definition 1 The probability or probability density $p(u)$ of an agent executing an action u in certain situation is the action preference of the agent executing u in that situation. In this paper, the action preference $p(u)$ is **P**'s subjective estimation of the probability of **E** executing action u.

The payoff is evaluated with the result of executing certain actions, *i.e.* it is evaluated by evaluated the situation as result of executing certain actions. If in current situation, **P** will be in state S_{it} by executing action w_i with time t and **E** will be in sate S'_{ut} by executing action u with time t, then

$$J(s_0, s'_0, w_i, u, t) = \psi(s_{it}, s'_{ut}). \tag{3}$$

Thus, Eq. (1) can be rewritten as follows:

$$E[J(s_0, s'_0, w_i, u, t)] = \int_U p(u) \psi(s_{it}, s'_{ut}) du. \tag{4}$$

According to Eq. (4), to evaluate its expected payoff of executing each kind action with time t in current situation, a player needs to know the action preference of its oppo-

nent and the future states of players. Because a player usually has different action preference in different situations, it needs estimate the type ofr the current situation firstly. Thus, the proposed action decision-making procedure consists of current situation sorting, possible short-term future states computation for all players, payoff evaluation and action selection, as shown in Fig. 1.

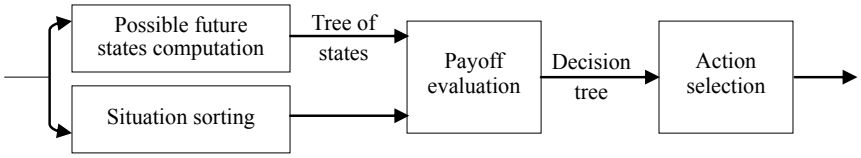


Fig. 1. Basic decision-making procedure.

2.1 Payoff Evaluation and Decision-making

Payoff evaluation procedure is divided into two steps: at first, evaluates single payoff; then integrates single payoffs into expected payoffs. Action decision is made based on a decision tree constructed from expected payoffs.

2.1.1 Single payoff evaluation

In pursuit-evasion games, single payoff is often scaled with the distance among game players. For many adversarial pursuit-evasion games, the vantage of situation is not only reflected by the parameters of distance. With the conception of feature that reflects profile of the situation vantage, a single payoff evaluation method is proposed as follows:

(1) At first, works out values of features those reflect the situation constituted by **P** state s_{it} against **E** state s'_{ut};
(2) Evaluates each feature respectively;
(3) Integrates gains of all features into a single payoff.

For pursuit-evasion games with safety-zone, such as typical scenario of intercepting a robot **E** dribbling in middle size soccer robots game, several features could be extracted: the distance between **P** and **E** at that moment, the distance between **P** and the possible shooting line of **E**, the related approaching speed between **P** and **E**, *etc.*

Payoff function used to evaluate certain feature can be designed based on experience knowledge and adjusted by experiments. For example, in soccer robots game, it is shown with experience knowledge that the shorter the distance between **P** and the possible shooting line of **E** is, the safer **P** is. So the payoff function of feature "the distance between **P** and the possible shooting line of **E**" can be in following form:

$$f = e^{-d/k} \tag{5}$$

where d is the distance and k is a positive tunable constant.

Assuming that feature payoff functions are $f_1, f_2, \ldots, f_n$ respectively, the single payoff function of **P** state s_{it} against **E** state s'_{ut} can be a linear weighted sum:

$$J(s_0, s_0', w_i, u, t) = \psi(s_{it}, s_{ut}') = \sum_{j=1}^{n} \alpha_j f_j \tag{6}$$

where α_j is the weight coefficient.

2.1.2 Expected payoff evaluation

As Eq. (4) shown in the above, the expected payoff $E[J(s_0, s_0', w_i, u, t)]$ of a player **P** executing action w_i in current situation can be evaluated. Since in adversarial game, players usually cannot know their opponents' actions in detail and they usually can only gain an estimation of their opponents' action space bound values. Thus it is not bad for a player **P** to assume its opponent has a continuous action space. It means that the payoff evaluation method as Eq. (4) is reasonable.

But it is difficult to compute expected payoff $E[J(s_0, s_0', w_i, u, t)]$ with Eq. (4) directly. Thus, approximately evaluation method is needed. According to the principle of Quasi-Monte Carlo, the whole characteristic of something can be reflected approximately by integrating characteristics of evenly distributed sample points. Since the expected payoff $E[J(s_0, s_0', w_i, u, t)]$ of **P** executing action w_i in some situation is a function of u as Eq. (4), **P** can select evenly distributed "virtual" actions u of **E** to evaluate $E[J(s_0, s_0', w_i, u, t)]$ approximately. For example, select angular velocities $\pm j\omega_{emax}/m$ as values of **E**'s $2m + 1$ actions u_j, where $j = 0, 1, \ldots, m$. Thus, the expected payoff of **P** executing action w_i in the situation can be evaluated approximately as follows:

$$E[J(s_0, s_0', w_i, u, t)] \cong \sum_{j=1}^{2m+1} J(s_0, s_0', w_i, u_j, t) p(u_j) \tag{7}$$

where $p(u_j)$ is the estimated action preference of **E** executing action u_j in the situation.

After all expected payoffs have been evaluated, **P** establishes a decision tree shown as Fig. 2, supposing that **P** has n choices of actions. In Fig. 2, 1, …, k_0 represent the indexes of predicting step.

Based on the decision tree, **P** determines which action to execute in next time.

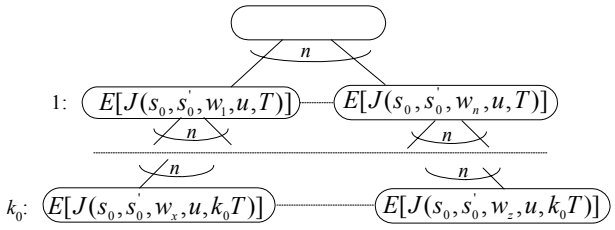


Fig. 2. Decision tree of **P**.

2.2 States Predicting

As mentioned in the above, payoff evaluation is based on the predicted situations; the policy executer **P** should work out possible future states of all players.

P can do multiple steps predicting. Here, a single predicting step is corresponding to time T: for **P** calculating future states of a game player, it is assumed that the game player executes certain action continuously with time T. If **P** has worked out all possible states of all players at time T, **P** finishes the first step prediction; if **P** has worked out all possible states of all players at time $2T$, **P** finishes the second step predicting; …. In general terms, if **P** has worked out all possible states of all players at time T, $2T$, …, nT, **P** finishes n steps prediction.

P predicts its own future states as follows: with current state s_0, it calculates the future state s_{iT} as the result of executing action w_i with time T. After it obtaining all possible states at time T, it finishes the first step prediction of itself. Similarly, at the second prediction step, it works out all possible states at time $2T$ with the states predicted at the first step. In the same way, **P** can finish n steps prediction.

The procedure of **P** predicting **E**'s future states is similar with predicting its own future states. The difference is that: when **P** predicts its own states, it uses real executed actions; when **P** predicts **E**'s future states, it is based on the "virtual" selected actions that are evenly distributed in **E**'s "virtual" continuous action space.

The time of each prediction step should not be too long. Obviously, if it is too long, it will lead to insufferable error for decision-making. For example, there may exist two actions A and B, at time T, executing A seems to be better than executing B; but at some time in (0, T), executing A means lose and executing B means win.

On the other hand, the time T of each prediction step should not be too short either. As the work shown in [10], it is better to let the total prediction time length reach some value to gain a reasonable decision. Thus, to achieve a reasonable decision, the less the time of each prediction step is, the bigger the prediction step number will be. With a big prediction steps number, the computation complexity will be enormous. The computing complexity may affect the capability to react in real-time. The computing complexity can be estimated with the times of computing single payoff.

Suppose that **P** has n kinds of actions and **P** assumes that **E** has m kinds of actions, then after k prediction steps, the predicted states number of **P** and **E** are $\sum_{i=1}^{k} n^i$ and $\sum_{i=1}^{k} m^i$ respectively, the number of calculating single payoffs is $\sum_{i=1}^{k} n^i m^i$.

Thus, the growth of prediction step number leads to great increasing of computing complexity. For example, if $n = m = 10$, increasing one prediction step would lead to 100 times augment of computing complexity; if the prediction step number $k = 5$, it takes more than 5s to calculating single payoffs using a 2GHz computer; on the contrary, if $k = 3$, the time used is about 1 ms quantity.

The time of each prediction step can be determined according to the maneuverability of players.

3. ACTION PREFERENCE ESTIMATION

A game player usually executes different actions in different situations, *i.e.* there are different types of action preference of the player for different situations. Thus, situation should be divided into many categories according to some standards. **P** has to estimate action preferences of **E** for all kinds of situations. In essence, the procedure of estimating

player **E**'s action preferences is a procedure of modeling **E**'s decision-making.

Obviously, it is ideal for player **P** that **P** divides the situation space in the same way as **E** does. But it is impractical since **P** usually doesn't know how **E** divides its situation space. If **P** wants to capture **E**'s intention correctly, size of situations in **P** should be less than that in **E**. On the other hand, **P** is usually unable to know the size of situations in **E**. In order to achieve the best estimation, what **P** can do is to divide the situation space into as many classes as possible with the limitation of computing complexity and the requirements for real-time decision-making.

The estimation of a player's action preference in certain situation is self-adaptive. In each decision-making cycle, the estimation of **E**'s action preference is adjusted. As pointed in Definition 1, the estimation of **E**'s action preference in certain situation is defined as the estimation of the probability of **E** executing each kind of action. According to the principles of probability, the probability of event u_i can be approximately estimated as follows:

$$p(u_i) \approx x_i \Big/ \sum_{j=1}^{n} x_j \ , \tag{8}$$

where x_j is the occurrence count of event u_j, n is the total event number of the sample space. Thus, **E**'s action preference in certain situation can be estimated by counting executed times of each action in the situation.

Initially, **P** may supposes that the probability of **E** executing each action is equal, *i.e.* the initial executed times of all actions are set to be equal. If x_{mi} is the executed times of **E** executing action u_i in certain situation m, then initially, all x_{mi} are set to be c_0, where c_0 is a non-negative constant.

In each decision-making cycle, **P** estimates the parameters of action that **E** executed and selects the "virtual" action u_i, that is the least difference with the estimated parameters, as the observed action. Then, **P** adjusts the executed times x_{mi} of the "virtual" action u_i for the last situation m as follows:

$$x_{mi} + c \rightarrow x_{mi}, \tag{9}$$

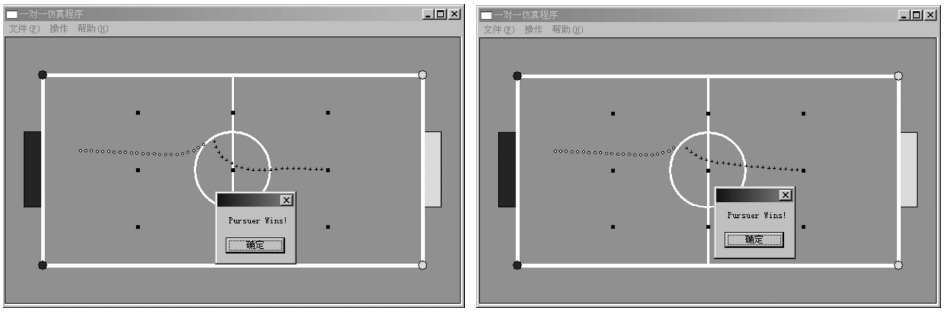
where c is a constant, such as 1. Finally, recalculate action preference of **E** in the situation m:

$$p_m(u_i) = x_{mi} \Big/ \sum_{j=1}^{n} x_{mj} \ . \tag{10}$$

4. SIMULATION

A simulation is carried out based on the practice of the autonomous soccer robots game. In order to illustrate the policy more explicitly, it uses One VS One intercepting scenario: agent **P** is responsible for intercepting the football dribbled or shot by **E**. For convenience, "**E**" is used to denote the robot dribbling the football or the shot football: if the football is shot, **E** is the football; otherwise, **E** is the robot dribbling the football.

The victory criterion of **E** is that the ball is sent into the goal. The victory criterion



(a) The first time simulation result. (b) The fifth time simulation result.

Fig. 3. Simulation results.

of **P** is that one of following events happens: (a) **P** enter into a zone with size 0.5m × 0.6m in front of **E**; (b) The football is out of the field.

In the simulation, the field size is 10m × 5m. The goal (safety-zone of **E**) is located at (10m, 1.5m-3.5m). **P**'s linear velocity is 1.5m/s and maximum angular velocity is 2 rad/s. **E**'s linear velocity is 1.5m/s and maximum angular velocity is 1 rad/s. The initial velocity of football shot by robot is 5m/s, and its acceleration is -2m/s^2. The system instruction cycles of **P** and **E** are both 0.1s. The horizontal orientation to goal side is 0rad as shown in Fig. 3.

E's policy is as follows: if the shooting condition is satisfied, then it shoots the football; otherwise, it tries to dribble the football to goal; if **P** appears at certain position in front of it, it turns around to avoid **P**; **E** does not predict.

In the simulation of **P** using the proposed self-adaptive predictive pursuing policy, it divides the adversarial situation into 108 categories:

(1) According to the X coordination of **E**, it is divided into 4 types: $x \le 5\text{m}$, $5\text{m} < x \le 6.5\text{m}$, $6.5\text{m} < x \le 8\text{m}$ and $x > 8\text{m}$;
(2) According to the Y coordination of **E**, it is divided into 3 types: $y < 1.5\text{m}$, $1.5\text{m} \le y < 3.5\text{m}$ and $y \ge 3.5\text{m}$;
(3) According to the relative position between **P** and **E**, it is divided into 9 types:
   (a) The distance between **P** and **E** is greater than 4m;
   (b) The distance between **P** and **E** is greater than 2m and not greater than 4m. According to the angle α between the orientation of **E** and the line between **P** and **E**, it is divided into 4 types furthermore: $\pi/6 \le \alpha < \pi/2$, $\pi/2 \le \alpha \le 3\pi/2$, $3\pi/2 \le \alpha \le 11\pi/6$ and otherwise;
   (c) The distance between **P** and **E** is not greater than 2m. According to the angle α between the orientation of **E** and the line between **P** and **E,** it is divided into 4 types furthermore: $\pi/6 \le \alpha < \pi/2$, $\pi/2 \le \alpha \le 3\pi/2$, $3\pi/2 \le \alpha \le 11\pi/6$ and otherwise.

In the simulation, **P** just uses 2 features to evaluate single payoff. The single payoff function of **P** executing action w_i against **E** executing certain action u_j is as follows:

$$J(s_0, s_0', w_i, u_j, t) = \psi(s_{it}, s_{jt}') = 0.5e^{-d_1/10} + 0.5e^{-d_2/10} \tag{11}$$

where d_1 is the distance from **P** to the intercepting point; d_2 is the distance from **P** to possible shooting line of **E**. Note: The single payoff function of **P** executing action w_i has not been optimized. The unit of distance in above equations is "m".

In the simulation, the initial coordination of **P** is (7.5m, 2.5m, 3.142 rad). The time of each prediction step is 1s. **P** does one-step prediction. **P** has 11 actions and assumes that **E** has 11 actions. The initial position of **E** is (5m, 2.5m) and its start angle varies.

For the proposed self-adaptive predictive pursuing policy, it carries on 50 times simulation in each initial condition (including players' initial positions, orientations):

(1) Initially, **P** assumes that the action preference of **E** is equal and adjusts the estimation of **E**'s action preference during simulation;
(2) With the same initial condition as the first time simulation and **P**'s adjusted estimation of **E**'s action preference as the result of last time simulation, the second time simulation is carried out and **P** adjusts the **E**'s action preference estimation furthermore;
(3) In the same way, fifty times of simulations are carried out.

As a contrast, in the same initial conditions, with the assumption that **E**'s action preference is equal; a simulation of short-term predictive pursuing policy is carried out. The simulation result is shown in Table 1.

Table 1. Result of simulation.

E's start angle	Self-adaptive pursuing policy	Equal action preference assumption
0	Win all	Win
0.2	Win all	Win
0.4	Win all	Win
0.6	Win the first 2 times	Win
0.8	Win the times of 1^{st}, 2^{nd} and 4^{th}	Win
0.9	Win the first 6 times, and the times of 8^{th}, 9^{th} and 10^{th}	Win
1.0	Win all	Lose
1.05	Win all	Lose
1.15	Win the first 18 times	Lose
1.25	Win the times of from 2^{nd} to 7^{th} and 15^{th} to 50^{th}	Lose

Result of Simulation:

(1) Using the self-adaptive predictive pursuing policy, it is possible that **P** improve the effect of pursuing. A simulation result with **E**'s original position (1m, 3m) and start angle 0 rad is shown in Fig. 3.
(2) The proposed policy is effective as illustrated in Table 1, where "Win" or "Lose" is about **P**. Although in some cases, the effectiveness of the proposed self-adaptive policy isn't better than the method with equal action preference assumption. There may exist several reasons. For example, the situation space division manner of **P** does not match with which of **E**.

In Fig. 3, "+" represents the pursuer, "o" represents the evader.

5. CONCLUSION

A self-adaptive predictive policy for pursuit-evasion games is presented, which takes action preference of its opponent into account. The estimation of an opponent's action preference in some situation would be adjusted according to the observation of the opponent. Thus, the predictive policy is self-adaptive. In essence, the adjustment procedure of the estimation of an opponent's action preference is a procedure of modeling the opponent's decision-making. Agent usually has different action preferences in different situations. To model an opponent more precisely, the player should divide the situation space into many categories. The policy can be used in real-time adversarial games. In such games, it may be unknown whether players are irrational or not.

Based on the model of RoboCup middle size soccer robots, a simulation has been carried out. The proposed policy has been illustrated to be effective.

In future, we would try to do further research and apply it in real soccer robots. It requires precise world modeling, including localization, motion estimation, *etc.*

REFERENCES

1. S. M. Lavalle and J. E. Hinrichsen, "Visibility-based pursuit-evasion: the class of curved environments," *IEEE Transactions on Robotics and Automation*, Vol. 17, 2001, pp. 96-202.
2. R. Vidal, O. Shakernia, H. J. Kim, *et al.* "Probabilistic pursuit-evasion games: Theory, implementation, and experimental evaluation," *IEEE Transactions on Robotics and Automation*, Vol. 18, 2002, pp. 662-669.
3. J. Shinar, Y. Lipman, and M. Zarkh, "Mixed strategies in missile versus missile interception scenarios," in *Proceedings of the American Control Conference*, 1996, pp. 4116-4120.
4. V. Turetsky and J. Shinar, "Missile guidance laws based on pursuit-evasion game formulations," *Automatica*, Vol. 39, 2003, pp. 607-618.
5. H. Kitano, M. Asada, I. Noda, and H. Matsubara, "RoboCup: robot world cup," *IEEE Robotics & Automation Magazine*, 1998, pp. 30-36.
6. J. P. Hespanha, M. Prandini, and S. Sastry, "Probabilistic pursuit-evasion games: a one-step nash approach," in *Proceedings of the 39th IEEE conference on Decision and Control*, 2000, pp. 2272-2277.
7. P. C. Zhou and B. R. Hong, "Group robot pursuit-evasion problem based on game theory," *Journal of Harbin Institute of Technology*, Vol. 35, 2003, pp. 1056-1059.
8. H. Fujii, M. Kawashima, E. Sugiyama, *et al.*, EIGEN Keio University Team Description, in *RoboCup 2006*, Bremen, Germany.
9. B. Dirkx, Philips RoboCup Team Description, in *RoboCup 2006*, Bremen, Germany.
10. Z. Luo and Q. X. Cao, "Pursuing method based on short-term prediction," *Journal of Shanghai Jiaotong University*, Vol. 40, 2006, pp. 1069-1073, 1078.
11. P. Stone and M. Veloso, "Multiagent systems: a survey from a machine learning perspective," *Autonomous Robots*, Vol. 8, 2000, pp. 345-383.

12. R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, Vol. RA-2, 1986, pp. 14-23.
13. B. L. Zhong, Q. Zhang, and Y. M. Yang, "Real time reactive strategies based on potential fields for robot soccer," in *Proceedings of the IEEE International Conference on Robotics*, *Intelligent Systems and Signal Processing*, 2003, pp. 1052-1057.
14. D. Camacho, F. Fernández, and M. A. Rodelgo, "Roboskeleton: an architecture for coordinating robot soccer agents," *Engineering Applications of Artificial Intelligence*, Vol. 19, 2006, pp. 179-188.
15. M. L. Minsky, "Theory of neural analog reinforcement systems and its application to the brain model problem," Ph.D. Thesis, Princeton University, New Jersey, 1954.
16. X. C. Wang, R. B. Zhang, and G. C. Gu, "Research on multi-agent team formation based on reinforcement learning," *Computer Engineering*, Vol. 28, 2002, pp. 15-16.

Zhen Luo (罗真) received his B.S. degree from Northwestern Poly-technology University, China, in 1996, received his M.S. degree in Cybernetics and Automation Engineering from Shanghai University in 1999. He is pursuing his Ph.D. degree in Shanghai Jiaotong University. His research interest is on autonomous robot.

Qi-Xin Cao (曹其新) received the M.S. degree from University of Miyazaki, Japan, in 1994, the Ph.D. degree from University of Kagoshima, Japan, in 1997. He is a professor of Research Institute of Robotics, Shanghai Jiaotong University, China. His research interests include autonomous robot, machine vision, neural network, and pattern recognition.

Yan-Zheng Zhao (赵言正) received his M.S. and Ph.D. degrees in Mechanical Engineering from Harbin Institute of Technology, Harbin, China, in 1988 and 1999. He is a professor of Research Institute of Robotics, Shanghai Jiaotong University, China. His research interests include special type robots and intelligent control, bionics robot, *etc.*