Integration of Coordination Architecture and Behavior Fuzzy Learning in Quadruped Walking Robots

Dongbing Gu, *Senior Member, IEEE*, and Huosheng Hu, *Senior Member, IEEE*

***Abstract*—This paper presents the design and implementation of a coordination architecture for quadruped walking robots to learn and execute soccer-playing behaviors. A typical hybrid architecture combing reactive behaviors with deliberative reasoning is developed. The reactive behaviors directly map spatial information extracted from sensors into actions. The deliberative reasoning represents temporal constraints of a robot's strategy in terms of finite state machines. In order to achieve real-time and robust control performance in reactive behaviors, fuzzy logic controllers (FLCs) are used to encode the behaviors, and a two-stage learning scheme is adopted to make these FLCs adaptive to complex situations. The experimental results are provided to show the suitability of the architecture and effectiveness of the proposed learning scheme.**

***Index Terms*—Behavior-based control, fuzzy logic controller (FLC), reinforcement learning (RL).**

I. Introduction

A. Architecture

COMPLEX tasks in the real world require autonomous robots to behave not only in real time and in a reflex way but also in causal reasoning and a temporal order. Three categories for robot control architectures have been developed so far to achieve complex goals, namely deliberative, reactive, and hybrid.

The deliberative architecture works in the sense–model–plan–act manner based on a single execution pipeline given the condition that a predictable world model between sensory information and action is given [1]. It is obviously difficult to accommodate the sensory uncertainty and the environmental dynamics. The reactive or behavior-based architectures are able to handle the problems that appear in the deliberative architecture by a bottom-up approach or horizontal task decomposition. The collection of different behaviors and the way of coordination form the foundation of the software architecture. There are many representative paradigms of behavior coordination, for example, subsumption [2] and schema-based architectures [3]. However, these types of architecture have suffered from difficulties in high-level reasoning, symbolic knowledge representation, temporal constraints, and scalability, when the number of behaviors increases and the degree of their interaction grows.

The way to build control architectures for both low-level control and high-level reasoning has emerged by hybridizing both aspects to gain the strength of the two architectures. Many architectures belong to this category, such as autonomous robot architecture (AuRA) [4], task control architecture (TCA) [5], reactive action package (RAP) [6], procedural reasoning system (PRS) [7], robot schemas (RS) [8], and open robot controller computer-aided design (ORCCAD) [9]. Most of these architectures have been built in three levels. The lowest level is committed to control robustness, real-time response, control minimalism, and concurrent processing. The highest level performs deliberative computation that is time-consuming. Normally, a human–machine interface is attached with this level to receive tasks. The tasks in this level are divided into subtasks such as cognition reasoning and decision making according to task requirements and world representations. The middle level, often called a sequencer or an interpreter, associates the subtasks with the behaviors by means of selection [4], [6], advising [10], or adaptation [8]. In this level, robots reconfigure their control strategies on the fly according to sensory information so that dynamic changes in the real world can be handled effectively.

B. Learning Fuzzy Logic Controller (FLC)

Based on the coordination architecture, different behaviors are designed individually. The aim is to enable robots to deal with local uncertain sensor reading and imperfect motor driving. There is an increasing tendency to build robot behaviors by using FLC since human experience can be embedded into the FLC to reduce the search space in the FLC learning process.

Many learning approaches have been proposed to model an FLC [11], including neural network (NN) based [12]–[14], reinforcement learning (RL) based [15]–[19], and genetic algorithm (GA) based [20]–[26]. The NN-based FLC can automatically determine or modify its structure and parameters with unsupervised or supervised learning by representing it in a connectionist way. However, the problem is that a large number of data pairs have to be provided to train the networks. In many applications, it is impossible to provide such input and output data pairs since these data pairs need to be learned. What can normally be obtained for evaluating a behavior is a delayed reward, i.e., the final outcomes after the behavior is executed.

Both GA-based and RL-based learning are two equivalent learning schemes, which need a scalar response from the real world by interacting with it to show their performance [27], [28]. Such a response can be a scalar value that is easier to collect than the desired input and output data pairs in robot applications. Also, it can be in any form without differentiable limitation.

Manuscript received March 18, 2005, revised August 18, 2005. This paper was recommended by Associate Editor G. Feng.

The authors are with the University of Essex, Colchester CO4 3SQ, U.K. (e-mail: dgu@essex.ac.uk; hhu@essex.ac.uk).

Digital Object Identifier 10.1109/TSMCC.2007.897491

1094-6977/$25.00 © 2007 IEEE

Further, it can be a delayed reward that is important for robot control in most robotic applications. The difference between GA-based and RL-based learning lies in the manner of state–action space searching.

GA-based learning is a population-based approach that encodes the structure and/or parameters of each FLC into chromosomes to form an individual and evolves individuals across generations with genetic operators to find the best one. Karr and Gentry [23] first used a GA to evolve the membership functions of an FLC. In [29], Homaifar and McCormick simultaneously evolved membership functions and rule sets. In these early research works, the input space is partitioned into a number of grids, with each cell representing a fuzzy rule. An FLC is encoded as an individual. FLCs in one generation compete with each other for survival. A trial in a learning process consists of several runs, each of which starts from a grid in order to make learning cover as many regions as possible for an FLC. In [24], Leitch and Probert used the same approach, but devised a novel coding technique, called context-dependent coding (CDC), to evolve the parameters and structure of an FLC. The credit assignment for an individual is made implicitly in the light of the fact that poor individuals will have few offspring in future generations. Bonarini [20] and Juang *et al.* [21] adopted different evolution strategies where an individual represents a rule, not an FLC. Rules in one generation compete with each other in order to be selected. The credit to individuals is assigned according to their contribution and eligibility traces. In order to reduce the search space, rules with the same antecedents are grouped into a subpopulation where competence occurs [20]. The FLC is constructed by selecting a rule from each subpopulation.

RL-based learning uses statistical techniques and dynamic programming methods to evaluate the value of policies in states of the real world in which each FLC is interpreted as a policy. It includes two similar learning schemes: fuzzy actor-critic learning (FACL) [15], [16], [18], [19] and fuzzy Q-learning (FQL) [17]. FACL is used for parameter learning, and its difficulty is that it needs both an actor network and a critic network to converge simultaneously. FQL is a kind of Q-learning that uses fuzzy logic to generalize the mapping between Q values and sensory-action pairs. In [17], it was successfully used to learn the actions of an FLC. Actually, GA and Q-learning described in [20] and [17], respectively, are both employed to select the consequences of an FLC. Their difference lies in the learning strategies employed to explore the output space.

C. Work in This Paper

In a hybrid system with three-layer architecture, the responsibilities of each level are not strictly defined. In most cases, the researches in each domain expand the capabilities and dominance of the layer within which they are working. The middle layer separates the planning from information on system's reactive abilities [30]. Since different hybrid systems are applied to different takes, they are far from being compatible [31]. The behaviors coordination as task-oriented planning in [32] makes the robot system become a two-layer architecture. In this paper, a coordination architecture is proposed for legged robots to learn soccer-playing behaviors in the Robo Cup domain [33]–[35]. In this architecture, reasoning or decision making for soccer playing is located in a deliberative level, and reactive behaviors are collected in a reactive level. The temporal order represented by a finite state machine (FSM) is used to model the deliberative reasoning. We use fuzzy logic inference to determine the inputs to the FSM in this paper, which is similar to the discrete event system in [36].

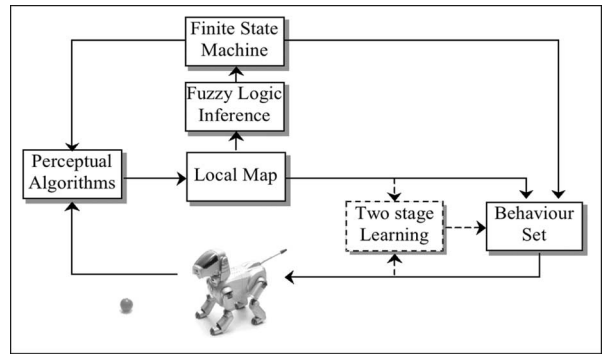


Fig. 1. Coordination architecture.

Evolving FLC for a robot's behavior has been explored by many researchers, for example, [37]–[40], etc. The learning described in this paper is used for legged robots and consists of two stages in order to reduce the search space. The first stage is the structure learning in which the rule base of an FLC is learned by a GA with a framework of the Q-learning scheme. The second stage is the parameter learning in which further fine-tuning of the learned FLC will be carried out.

The rest of the paper is organized as follows. Section II describes the proposed coordination architecture in which a local map is adopted to provide spatial features for temporal reasoning and behavior control. Section III illustrates the two-stage learning for fuzzy behaviors. Simulation and real experiment results are provided in Section IV to show the learning performance. Finally, conclusions and future work are briefly given in Section V.

II. Map-Based Temporal Coordination Architecture

A map-based temporal coordination architecture is proposed in Fig. 1 for Sony legged robots to learn soccer-playing behaviors, including perception, reasoning, a local map, reactive behaviors, and behavior-learning mechanism. The perception component reads data from sensors, analyzes them, and recognizes the objects that appear in the structured environment. Its outputs are the local spatial features with reference to the robot itself, and they are maintained in a local map. Though there is no direct connection between perception and action, a local map is more useful for behaviors as it serves as a short-term memory. The robots can behave more reasonably based on the short-term memory than based on immediate perceptual data with the limited view of their sensors. This means the robots have more information about their surroundings in the memory than in their instant view. The perception is a goal-oriented process in which the deliberative reasoning guides the perception by

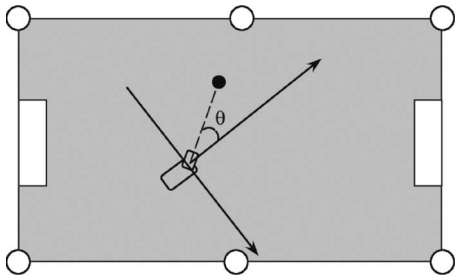


Fig. 2. Playing field.

multiplexing different object-recognition processes in different states. It leads to decreasing processing time in the perception component and increasing the reactivity in behavior control.

The deliberative reasoning part is composed of two components: FSM and fuzzy logic inference in Fig. 1. State definition and transitions are based on human heuristic experience. The triggers to state transition are inferred from spatial information in the local map, and fuzzy logic is employed to implement this inference with the consideration of ambiguous local information. Each state in the FSM corresponds to a behavior in the behavior set component. Once a state is active, the corresponding behavior will produce motion commands from spatial information in the local map.

The learning component expressed in dashed lines in Fig. 1 is an off-line learning mechanism. During the off-line learning process, the robot explores its state and action spaces, samples current states and rewards from the local map, and evaluates the actions just being executed. We initially encode behaviors by hand-crafted FLCs. Then, these FLCs are learned individually in local regions of the state space. This approach decouples the interaction between the behaviors.

In our team, two attackers and one goalie are used [41]. The two attackers are positioned on the left and right part of the pitch, and the goalie stays near its goal and faces the opponent's goal. The two attackers have the same contents in their individual components except for left and right position information. The goalie has the same contents in its perception component and local map as the attackers. However, the behavior set and reasoning components of the goalie are different from that of the attackers since they have totally different strategies in order to play their roles. The following section will describe the local map, and the attacker's deliberative reasoning and behavior sets. The goalie's deliberative reasoning is modeled in the same way, but not repeated in this paper. The learning will be presented in the next section.

A. Local Map

The pitch for the Sony legged robots is 3 m in length and 2 m in width. The goals are centered on both ends of the pitch, with a size of 60 cm $\times$ 30 cm. Six unique colored landmarks are placed around the edges of the pitch, as shown in Fig. 2. The ball, walls, goals, landmarks, and robot uniforms are painted with eight different colors distributed in the color space so that a robot can easily distinguish them.

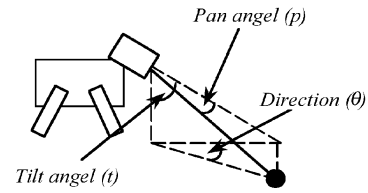


Fig. 3. Object direction.

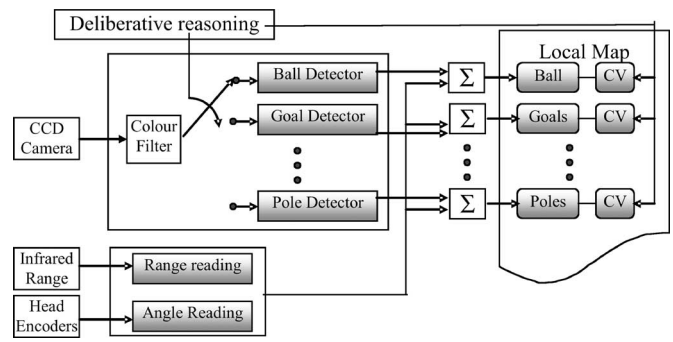


Fig. 4. Map building.

The local map is two-dimensional and centered at the robot head, with its X-axis aligned with the robot body (see Fig. 2). The objects that are recorded in the local map could be ball, goals, beacons, ground, teammates, opponents, and white lines. Spatial features of an object B in the local map are defined as

$$B(\theta, h, s, \mathrm{CV})$$

where θ is the angle relative to robot's orientation. It is calculated by pan angle (p) and tilt angle (t) reading from the head's encoders (see Fig. 3)

$$\theta = a\ \tan\left[\frac{\tan(p)}{\cos(t)}\right]$$

where h is the distance relative to the robot's position. It is read from an infrared range sensor. The variable s is the object's pixel size in the image. It is an outcome of a morphology filter and a run length encoding (RLC) [42]. CV indicates the certainty values in the robot's memory for what is seen previously by the robot and will decay with time exponentially.

Object recognition is a color-based processing that filters the image captured by the onboard color CCD camera, based on a color detection table (CDT) [42]. The map updating is goal oriented through switching the perceptual algorithms by a signal from the deliberative reasoning (see Fig. 4). It leads the robot to saving time in perceptual processing and enhancing the capability of the real-time response. Having chosen one object for processing, the perceptual algorithm proceeds with image recognition. The recognized results are combined with other sensor readings, and their CV values are put into the local map. A maximum CV value is assigned when the corresponding object is recognized. The CV values decay with time.

B. Temporal Coordination

Selection and design of robot behaviors depends upon the robot's task and its environment. Arkin [3] concluded that most

recently used methods fall into three categories: the ethnologically guided design, the situated activity-based design, and the experimentally driven design. For a given environment such as a football pitch described previously, the situated activity-based design is more reasonable since the robot's action can be predicted based on the situation where the robot locates itself. For example, the robot needs to find the ball when the ball disappears from its view. If it finds the ball, the robot needs to approach the ball to manipulate it. This design method actually constrains the behaviors with the temporal order and spatial features. While the spatial features can be represented in the local map from which a robot extracts stimuli for its behaviors, the order for the robot's activities forms behavioral temporal constraints.

To play a role as an attacker in the team, the robot is provided with a set of behaviors defined in a temporal order according to what it perceives from the following situations.

1) *Standing up* (SU): The robot should try to stand up from the beginning or when it falls down.
2) *Looking for ball* (LB): This behavior will be triggered when the ball is lost from the robot's view for a certain time and remain active until the ball appears in its view again.
3) *Approaching ball* (AB): The robot can move toward the ball when the ball is within its view. This behavior will transit to the last behavior LB when it loses the ball, or to next behavior *looking for goal* when the ball is large enough and its position is right in front of the robot.
4) *Looking for goal* (LG): This behavior will end when the robot finds the goal or it knows the goal's location.
5) *Aligning with goal* (AG): The robot tries to walk behind the ball in order to let the ball's position lie in a line with the robot and goal. The robot will either achieve its purpose with a certain angle tolerance and go to *kicking ball* behavior, or lose the ball due to touching the ball by itself or other robots and go to *backward looking* behavior.
6) *Kicking ball* (KB): The robot uses fast walking command to kick the ball and pays more attention to the goal's direction rather than the ball's position. It will shift into *LB* behavior if it gets a score or *backward looking* behavior if it loses the ball.
7) *Backward looking* (BL): This behavior enables the robot to move back for a few steps only while its head looks for the ball since the robot believes the ball is somewhere nearby when it just loses the ball. If the robot finds the ball again, it returns to last behaviors. Otherwise, it goes to *LB* behavior.

Each of the behaviors described before is regarded as a state in a time series process. This attacker's deliberative reasoning for scoring a goal is modeled as a discrete event system. A discrete event system can be represented by an FSM by which the implementation and verification of architectures are easy to achieve. Brooks employed an augmented FSM for each behavior in his subsumption architecture [2]. Arkin used FSMs to represent the temporal coordination of perceptual algorithms for navigation [43]. Moreover, FSMs are also used to design stable behaviors for a mobile robot in [44], and vision-based path control in [45]. There are four elements in designing an FSM.

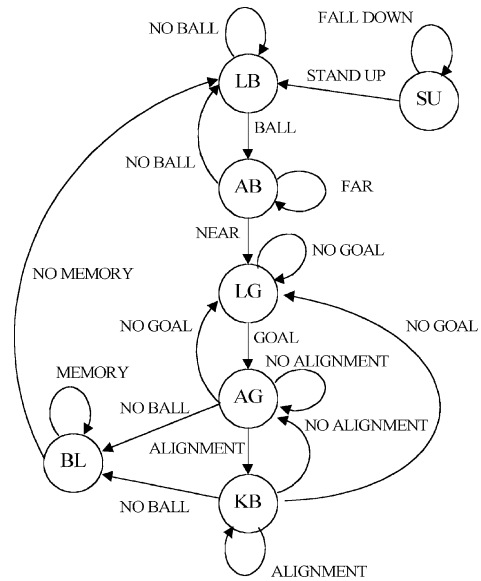


Fig. 5. FSMs for an attacker.

1) *Allowable states A:* Seven allowable states for an attacker are SU, LB, AB, LG, AG, KB, and BL.
2) *State outputs O:* They indicate the corresponding behavioral execution.
3) *State inputs Y:* They are related to spatial features and explanation is given in the next section.
4) *State transition:* It includes the circles that represent the states, the arcs with arrow that represent the direction of state transition, and the symbols near the arcs that represent triggered conditions.

C. Fuzzy Logic Inference for the State Inputs

As described in the previous section, the state inputs in an FSM are inferred from some heuristic rules, based on the object features maintained in the local map. A set of symbols of the state inputs for an attacker shown in Fig. 5 is described as follows.

1) BALL or NO BALL: It depends on the CV value, size, and direction θ of the ball.
2) GOAL or NO GOAL: It depends on the CV value, size, and direction θ of the goal.
3) FAR or NEAR: It depends on the CV value, size, and direction θ of the ball.
4) ALIGNMENT or NO ALIGNMENT: It depends on CV values of both the ball and goal, and direction of both the ball and goal.
5) MEMORY or NO MEMORY: It depends on the CV value of the ball.
6) FALL DOWN or STAND UP: It depends on the falling-down indicator.

Some of them are easy to determine, for example, MEMORY or NO MEMORY and FALL DOWN or STAND UP. Others are difficult to determine since the spatial features are vague and can be better described by fuzzy sets. An inference mechanism of

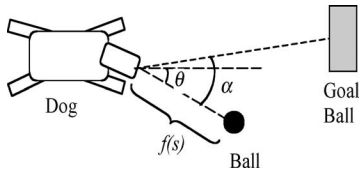


Fig. 6. AB behavior.

fuzzy logic is employed to produce the symbols of state inputs that are modeled as fuzzy output singleton sets.

For different states in an FSM, there exist different fuzzy logic inferences since the spatial features used in the local map and heuristic rules are different. Let $X^k = [x_1^k, x_2^k, \ldots, x_{p^k}^k]^T$ be p^k spatial features selected from the local map for the kth state and Y^k be its inference output. For example, for the third state AB, $X^3 = [\text{CV}_{\text{ball}}, s_{\text{ball}}, \theta_{\text{ball}}]^T$ and Y^3 is one of the state input symbols {NO BALL, FAR, NEAR}. The heuristic rules are expressed as fuzzy rules

$$R_i^k : \text{ if } x_1^k \text{ is } L_{i1}^k \ldots \text{ and } x_{pk}^k \text{ is } L_{ip^k}^k \text{ then } Y^k \text{ is } S_i^k$$

where i denotes the ith rule, $L_{i1}^k, \ldots, L_{ip^k}^k$ the fuzzy antecedent sets, and S_i^k one of the state input symbols for the kth state in the FSM. If $\mathbf{x}^k = \left[x_1^k, \ldots, x_{p^k}^k\right]$ is read from the local map, the activation degree of this rule is

$$\alpha_i^k(\mathbf{x}^k) = \prod_{j=1}^{p^k} L_{\text{ij}}^k(x_j^k), \qquad i = 1, 2, \ldots, M \tag{1}$$

where M is the number of rules. The output of the fuzzy logic inference is

$$Y^k = S_{i^*}^k, \qquad i^* = \arg\max_i \alpha_i^k. \tag{2}$$

Then, Y^k is used to trigger the current state k to transit into next state according to the FSM in Fig. 5.

III. Fuzzy Behavior Learning

A. Fuzzy Reactive Behavior

The AB behavior makes the robot move as close as possible to the ball and head toward the goal at the end. There are three variables associated with this behavior: the head direction to the ball represented by θ, the distance to the ball by h, and the goal's angle by α (see Fig. 6). A visual tracking algorithm is implemented for the head to track the ball based on color detection. It leads θ to be expressed by the head's pan angle that can be easily read from the encoder mounted on the pan motor when the ball is tracked.

There is an infrared range finder in the robot head for measuring the distance to the ball. It is difficult for a tracking algorithm to point to the exact position on the ball due to the direction sensitivity of infrared ray. Instead, we use the number of image pixels of the ball (s) captured by the CCD camera of the robot to estimate distance ($f(s)$). The tracking algorithm will also yield the goal's angle during the tracking process. Therefore, the input vector of this behavior is $S = [s_1, s_2, s_3]^T = [\theta, s, \alpha]^T$.

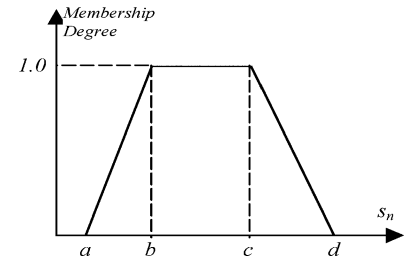


Fig. 7. Membership function.

The output set of the behavior is a group of one-step commands: *MOVE FORWARD*, *LFFT FORWARD*, *RIGHT FORWARD*, *LEFT TURN*, *RIGHT TURN*, *BACKWARD*, or *STOP* provided by low-level walking software.

Both sensory information and actions are imprecise, incomplete, and vague. For example, the ball size measured in pixels from the CCD camera does not accurately correspond to the distance from the robot to the ball, but just has a fuzzy concept such as *small* (S), *middle* (M), or *large* (L). The pan angle measured in degrees from the head motor encoder is accurate to some extent, but is not a precise direction from the robot to the ball since the ball may not stay in the image center when the robot moves. We have a set of fuzzy concepts *left* (L), *zero* (Z), or *right* (R) for the pan angle. The angle to the goal is also defined as *left* (L), *zero* (Z), or *right* (R). The output of this behavior is one of seven one-step commands that cannot be precisely described by odometry due to nonperfect actuators or slippage on the ground.

We define F_n^j as the jth fuzzy set $(j = 1, \ldots, 3)$ for the nth input variable s_n. A quadruple of (a, b, c, d) is used to represent a triangle or trapezoid membership function of a fuzzy set as shown in Fig. 7. The output o of the FLC is a crisp value expressed as a fuzzy singleton c^m $(m = 1, \ldots, 7)$. The membership degree for the fuzzy set F_n^j is expressed as

$$\mu_{F_n^j}(s_n) = \begin{cases} \max\left[0, \frac{d-s_n}{d-c}\right] & s_n > c \\ \max\left[0, \frac{s_n-a}{b-a}\right] & s_n < b \\ 1, & \text{otherwise.} \end{cases} \tag{3}$$

A linguistic fuzzy rule is expressed as

$$R_i : \text{IF } s_1 \text{ is} F_1^{j_i} \text{ AND } s_2 \text{ is } F_2^{j_i} \text{ AND } s_3 \, is \; F_3^{j_3}, \text{THEN } o \text{ is} c^{m_i}. \tag{4}$$

There are $N \quad (N = 3 \times 3 \times 3)$ rules in total. The true value for the ith rule activated by an input vector S is calculated by Mamdani's minimum fuzzy implication

$$\alpha_{R_i}(S) = \min(\mu_{F_1^{j_i}}(s_1), \mu_{F_2^{j_i}}(s_2), \mu_{F_3^{j_3}}(s_3)). \tag{5}$$

The crisp output a stimulated by the input vector S is calculated by the center of area method (COA)

$$o(S) = \frac{\sum_{i=1}^N \alpha_{R_i}(S) \cdot c^{m_i}}{\sum_{i=1}^N \alpha_{R_i}(S)}. \tag{6}$$

The output o will change if the robot visits different vectors S. The change is also determined by the quadruples (a, b, c, d). The approach of how the robot learns these quadruples to achieve

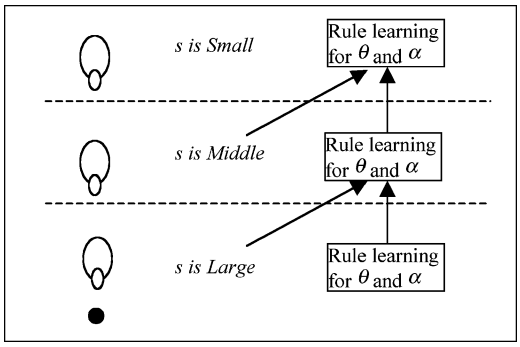


Fig. 8. Rule learning.

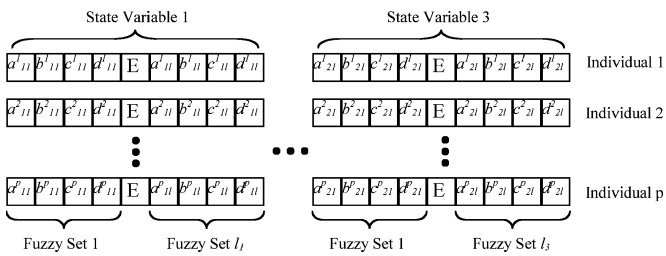


Fig. 9. Real value encoding of FLCs for one generation with p individuals.

its behavior goal is presented in Section III-C and the rule base generation is described in the next section.

B. Rule Base Generation

The aim of AB behavior is to move toward the ball with the orientation to the goal. Payoffs should be designed to guide the learning of FLCs to achieve this goal. We define payoffs as

$$r = w_1 \frac{\sum s}{l} + w_2(A-l) + w_3(B-\theta) + w_4(C-\alpha) + w_5 s \quad (7)$$

where w_i $(i = 1, \ldots, 5)$ is the weight, A, B, and C are the constants used to define a maximum optimal problem, and l is the number of steps used to achieve the goal. Other notations have been defined earlier. The first term indicates payoffs received during the whole movement. The second term rewards those FLCs that have fewer steps. The last three terms, associated with θ, d, and α in the last step, reflect the final position of the robot.

We use the idea of FQL proposed in [17] to redefine the rule base of an FLC in (4) as

$$\begin{aligned} R_i: \quad & \text{IF } s_1 \text{ is } F_1^{j_i} \text{ AND } s_2 \text{ is } F_2^{j_i} \text{ AND } s_3 \text{ is } F_3^{j_3}, \\ & \text{THEN } o \text{ is } c^1 \text{ with } q(i,1) \text{ or } o \text{ is } c^2 \text{ with } q(i,2) \\ & \text{or } o \text{ is } c^7 \text{ with } q(i,7) \end{aligned} \quad (8)$$

where $q(i, j)$ is a q-value employed to evaluate the conclusion parts. The learning robot does not calculate $Q(s, a)$ values during the learning process. It is only interested in finding the best conclusion for each rule, i.e., the action with the best q-value. This can be done by incrementally learning the rules through backup updating.

The parameters in membership functions are initially assigned from our experience. Then, we choose these grids as initial positions of the learning robot where the membership degree of a fuzzy set is equal to 1 and those of the other two fuzzy sets are equal to 0. The learning starts from the grid that is the closest to the ball, as shown in Fig. 8. The robot moves toward the ball under different moving commands and receives the individual payoffs at the end of each test. The command with the highest q-value is selected as the conclusion for the corresponding rule. The same procedure is executed in the next grid and the learning will exploit the rules learned in the last grid. The learning will be terminated when all of the grids are tested. Fig. 8 shows how the robot learns incrementally and the rule base of an FLC is updated from the grid (*s is large*) to the grid (*s is small*).

The learning makes use of the stochastic exploration strategy of GA, which explores the rule conclusion by using the crossover and mutation operators. The regions where the fuzzy sets overlap are implicitly generalized by the rules in the regions where there is no overlap among fuzzy sets. The fine-tuning of an FLC within the overlapped regions will be presented in the next section by the parameter learning.

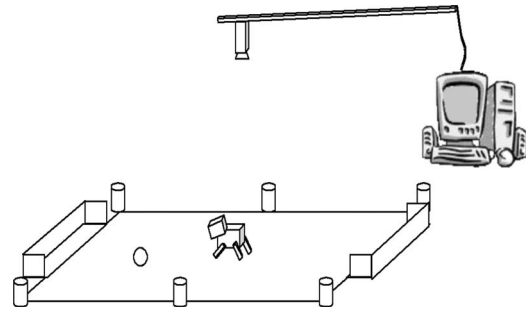


Fig. 10. Test environment.

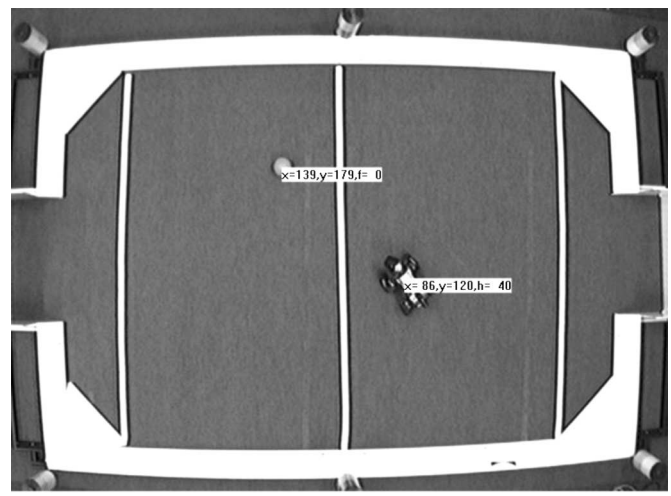


Fig. 11 Top view of the playing field.

C. GA Learning for the Parameters of an FLC

GA used here is to optimize the parameters in membership functions in order to improve the reactive behavioral performance. One FLC is encoded as one individual. In each generation, a population of individuals is maintained to compete with each other for survival. By evolving through genetic operators,

the best one will be selected as the optimal FLC for behavior control.

The parameters of input membership functions for an FLC are real values. It is better to manipulate them directly in a real value space rather than in a discrete space. Therefore, we adopt a real value GA learning to optimize membership parameters. The real value encoding of FLCs for one generation with p individuals is shown in Fig. 9, where one individual represents one FLC.

The payoffs in (7) are used as fitness functions. The learning procedure starts from a random formation of the initial generation. After completing the trials of one generation, GA evolves into a new generation and the learning repeats again until the GA terminal condition is met.

There are three genetic operators employed: reproduction, crossover, and mutation. The real value mutation is implemented by adding a certain amount of noise to new individuals to produce offspring with a mutation probability p_m. For the ith gene in the jth individual g_i^j, it will be evolved to $\hat{g}_i^j$ according to the following rules:

$$\hat{g}_i^j = \begin{cases} g_i^j + \Delta(t, \mathrm{UB} - g_i^j) \\ g_i^j - \Delta(t, g_i^j - \mathrm{LB}) \end{cases} \tag{9}$$

where UB *and* LB are lower and upper bounds of variable g_i^j. The function $\Delta(t, y)$ returns a value in the range [0, y] such that the probability of $\Delta(t, y)$ being close to 0 increases as the generation t increases. This causes the operator to search the space uniformly at early stage and converge locally at later stages

$$\Delta(t, y) = yr\left(1 - \frac{t}{T}\right)^b \tag{10}$$

where r is a random number in [0, 1], T is the maximal generation number, and b is a parameter determining the degree of nonuniformity [46].

IV. Simulation and Experimental Results

Several simulations and experiments have been conducted to verify the proposed architecture, rule learning, and fuzzy membership function parameter learning. First, the coordination architecture for an attacker is tested on both simulator and a real robot. The seven behaviors are implemented by handcrafted rules and FLCs. The simulator is built up by using the data collected from testing on real Sony legged robot walking. Second, the FLC learning is tested on both the simulator and a real robot. The learning simulation includes both the structure and parameter learning. The real robot is only employed to evaluate the structure learning. Since too many runs would be needed to run the parameter learning in a real robot, it is only possible to learn the parameters off-line.

Although the onboard sensors can provide the results of the interaction between a robot and its environment, the perception aliasing is severe for the large perception space and high noised sensors. To evaluate the performance of robot behaviors, the environment should provide fitness to robots with a certain accuracy to improve the learning efficiency. A global monitor is set up in our test (see Fig. 10), which includes an overhead camera, a desktop computer, and visual tracking software, to provide the external judgment for the interaction between the robot and its environment [47]. The function of the monitor is to feed the position information of the robot and the ball to the robot. Then, the robot can autonomously test its control strategies and evaluate the results.

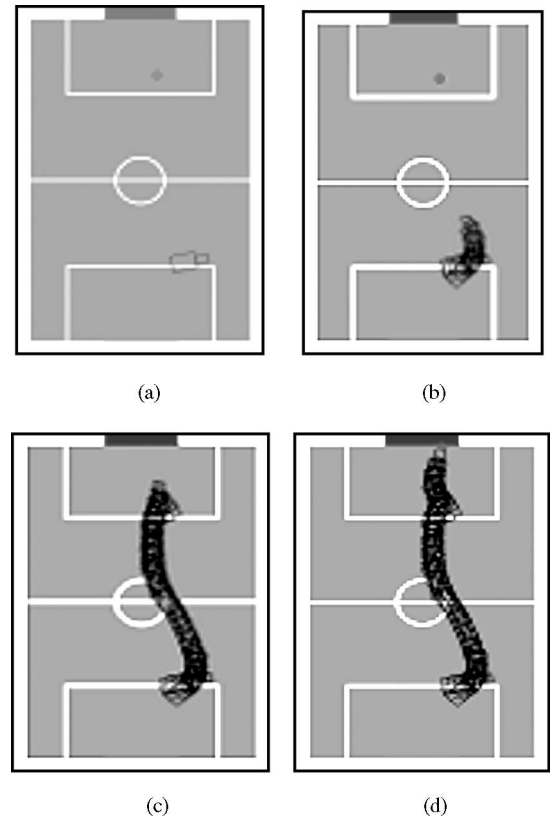


Fig. 12. Simulation results of an attacker.

The monitor recognizes the robot and the ball according to their color (see Fig. 11). Through the image processing, the monitor updates their position continuously. The robot can ask for the information at any time during its learning process. The communication is achieved through Internet where the monitor system acts as a server and the robot acts as a client. The server provides the global information when the client has a request. Since the robot only evaluates its performance at the end of one run, there is no significant effect of the communication on the learning process.

A. Coordination Architecture

A simulation environment was built based on the exact size of a real pitch. The motion of a Sony legged robot is identified by measuring the consequences of the actual execution of individual walking commands. In order to reflect real-world uncertainty, Gaussian noise is added in both motion equations and sensor readings during the simulation. One-step delay is used in command execution.

The simulation results for an attacker are shown in Fig. 12. The attacker is positioned near its goal and the ball is placed at the other side of the field [Fig. 12(a)]. The robot first goes

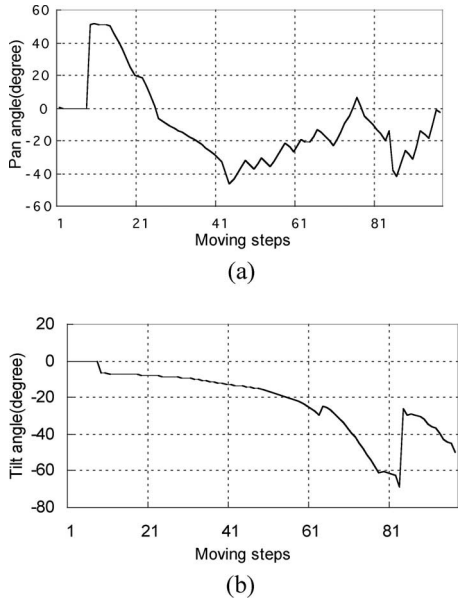


Fig. 13. Simulation results of pan and tilt.

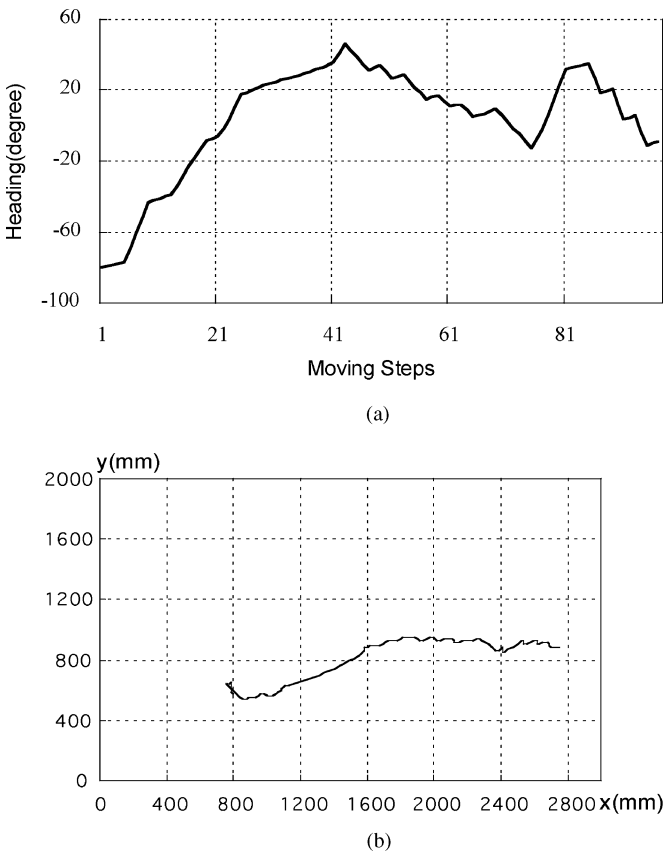


Fig. 14. Simulation results of the position and heading.

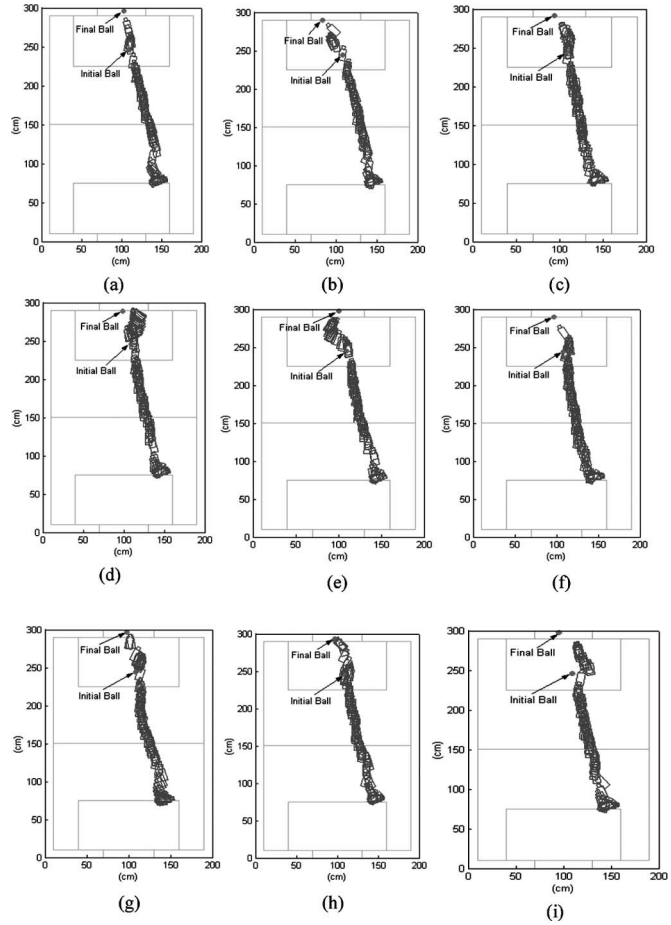


Fig. 15. Real robot runs.

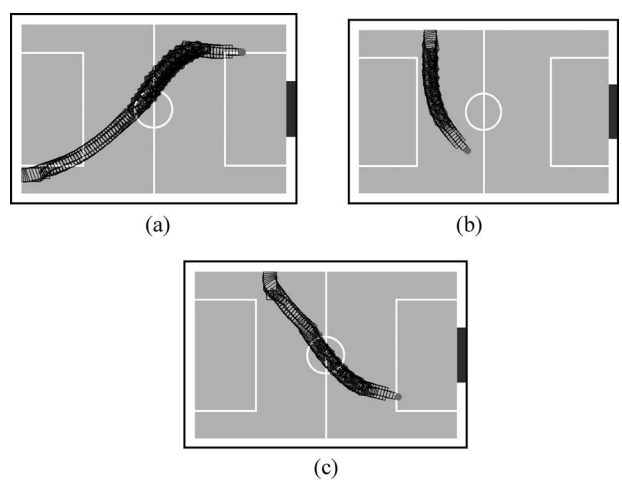


Fig. 16. Simulation results of the structure learning.

into the LB state where *LB* behavior is executed. Upon finding the ball, the robot transits into the AB state from the LB state. [Fig. 12(b)] shows how the robot tries to approach the ball in the AB state. Then, the robot starts to align itself with the goal until the directions of the ball and the goal fall in the given range [Fig. 12(c)]. Finally, the robot kicks the ball into the goal by a fast-walking command [Fig. 12(d)]. The robot takes about 90 steps to shoot the ball into the goal from a given initial position. The transition from LB to AB appears around the 10th step after the robot turns left a few steps. The pan angle is turned left about 50° in order to track the ball [see Fig. 13(a)]. When the robot gets close to the ball, it raises its head to look for the goal. This happens at the 83rd step where the tilt angle increases

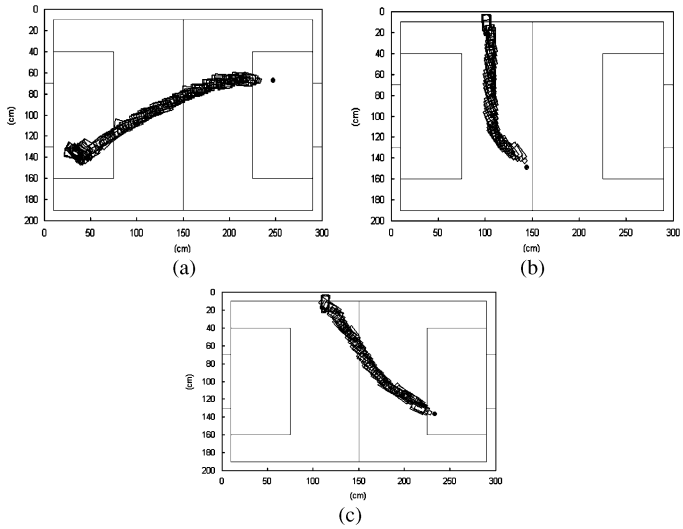


Fig. 17. Real robot results of the structure learning.

abruptly [Fig. 13(b)]. As soon as the goal is found, the robot goes to the AG state to try to align with the goal. At the same time, it moves its head to track the ball according to the previous position stored in the local map. The tilt angle decreases again after the 83rd step. The corresponding pose of the robot is shown in Fig. 14. The initial pose is (780 mm, 620 mm, $-80°$). The robot moves along an arc rather than a straight line after it turns to the ball. The reason is due to the dynamic characteristics of legged robots. Finally, the robot has to align with the goal before it shoots the ball [see the changes around the 81st step in Fig. 14(a)].

The real robot is used to valid the proposed architecture. The robot and the ball are initially placed in the same position as in the simulation in order to evaluate the simulation model. Nine runs are carried out and all trajectory results are shown in Fig. 15. Since the robot is initially placed to face the left side of the field, all of the runs are started from the LB state. When the robot finds the ball, the robot runs the AB behavior until it moves to the ball position. Then, the robot looks for the goal, aligns with the goal, and kicks the ball. All ten runs are successfully terminated at scoring a goal. Due to the randomness of the ball movement after the robot touches it, the robot aligns with the goal in different angles. These angles can be seen from the final parts of the trajectories (after the ball initial positions). Fig. 15(e) shows that the robot maneuvers the ball from the left side rather than from the right side as in case of all other nine runs. Comparing with the simulation trajectory in Fig. 12(f), they share the similarity though they are not exactly same due to the stochastic nature.

The number of steps to score a goal is recorded to further evaluate the simulation model. For the real robot, the mean and variance of the numbers of steps in ten runs are $\mu_r = 85.2$ and $\sigma_r = 8.1$. For the simulation, the mean and variance of the numbers of steps in ten runs are $\mu_s = 89.3$ and $\sigma_s = 3.1$. For the error level $\alpha = 5\%$, the T-test value is $T = 1.5$, which is smaller than t-distribution value $t_\alpha = 2$. It means there is not a significant difference between two mean values.

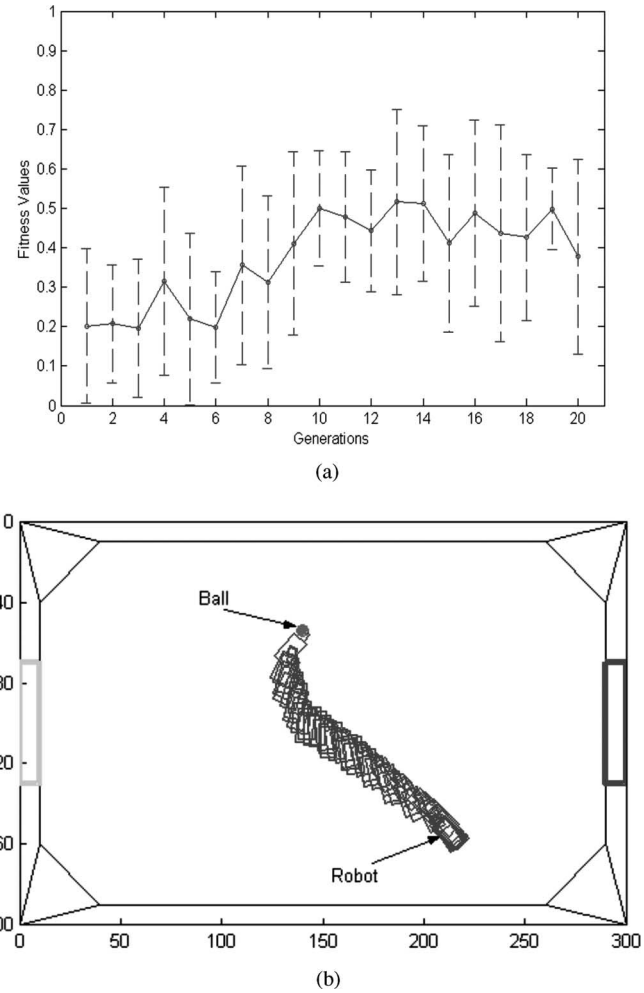


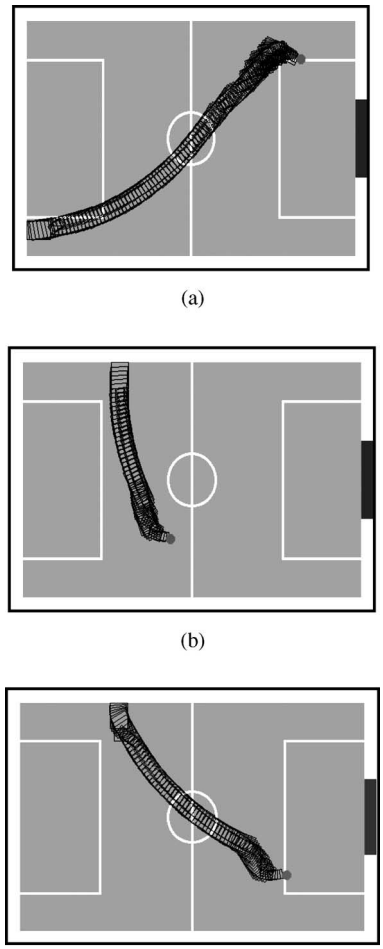
Fig. 18. Online learning results. (a) Fitness values. (b) Trajectory after rule base generation.

B. FLC Learning Experiments

The rule base is incrementally built up by a backup updating approach described in Section IV. Fig. 16 shows the results for the AB behavior controlled by the FLC with the learned rule base. It can be seen that the robot can move to the ball although the robot does not exactly face the goal at its final position, i.e., the robot faces the end of the pitch at the left side in Fig. 16(a), it faces the side of the pitch in Fig. 16(b), and it faces the end of the pitch at the right side.

The learned FLC is also employed in the real robot to test its performance. Fig. 17 shows the trajectory results of the real robot. It can be seen that they have behaviors similar to simulated behaviors, and the robot does not exactly face the goal at its final position.

An online version of the rule base generation is implemented on a real robot to verify the proposed approach. Each individual represents an FLC in which a rule conclusion is coded as a gene. The fitness function in (7) is used, but final results are normalized to 1. The robot is placed on different positions to explore the state space. The population size is 10, the crossover probability is 0.2, and the mutation probability is 0.1. The average fitness values and the standard deviations are calculated for



(a)

(b)

(c)

Fig. 19. Simulation results of the parameter learning.

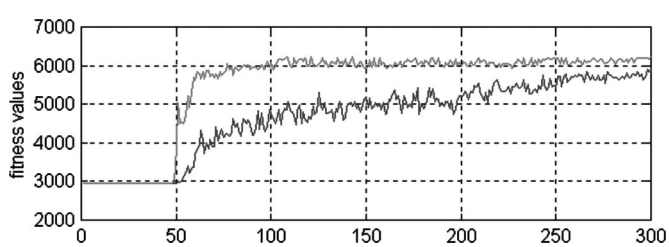
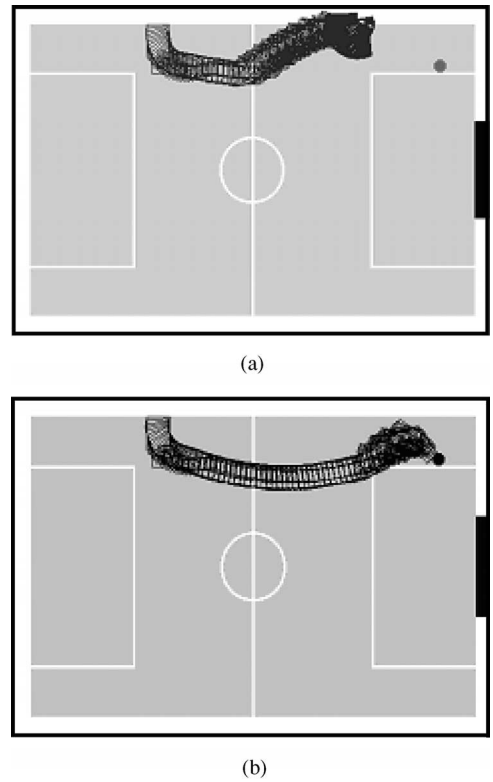


Fig. 20. Evolution process of the parameter learning.

each generation. After 30 generations, the calculated results are shown in Fig. 18(a). The fitness values are gradually increased. The dropdowns in the middle of the curve indicate the solution exploration procedures by the mutation and crossover operators. The standard deviations do not converge to zero, but they are not diffuse. This is caused by many factors in the real situation. For example, the visual-tracking algorithm could fail to track the ball, the robot could slip on the pitch, the monitor could provide an inaccurate position, etc. The best FLC is picked up from the last generation according to their q-values to do the test. Fig. 18(b) shows the behavior is successfully evolved. The robot can move to a ball (in the middle of the playing field).



(a)

(b)

Fig. 21. Comparison between the structure learning and the parameter learning.

In the parameter learning, the probability of both crossover and mutation are chosen as 0.2. The size of the population in one generation is 50. In the mutation function (9), we assume $r = 0.3$ and $v = 5$. The same situations as in Fig. 16 are tested for the robot with the structure learned FLC. The results are shown in Fig. 19 where the improvement in final positions can be seen. The GA learning process is shown in Fig. 20 after evolving 300 generations. The upper curve is the maximal fitness values in each generation. The low curve is the average fitness values in each generation. It is shown that the average fitness values converge toward the maximum as the generation increases.

The quadruple (a, b, c, d) for each input fuzzy label should be constrained by their geometric shapes $(a \leq b \leq c \leq d)$ during GA learning. A validation process is employed to check the constraints for each FLC before it is used. Invalid FLCs are given up. The robot can see the ball even if it does not move and receive rewards. Therefore, most of the FLCs are not executed due to their invalidation at early stages and the fitness values are unchanged before the 50th generation.

Fig. 21 shows a comparison made before and after the parameter learning. In structure learning, only the regions where input fuzzy membership degrees are equal to 1 are learned. The overlapped regions are implicitly generalized by adjacent regions. Fig. 21(a) is the case that cannot be generalized, and Fig. 21(b) shows compensation made by parameter learning. After the parameter learning, the FLC is further fine-tuned so that fuzzy become more reasonable and the robot achieves its goal with this FLC. Fig. 22 shows other successful situations where the best FLC is applied after its evolving.

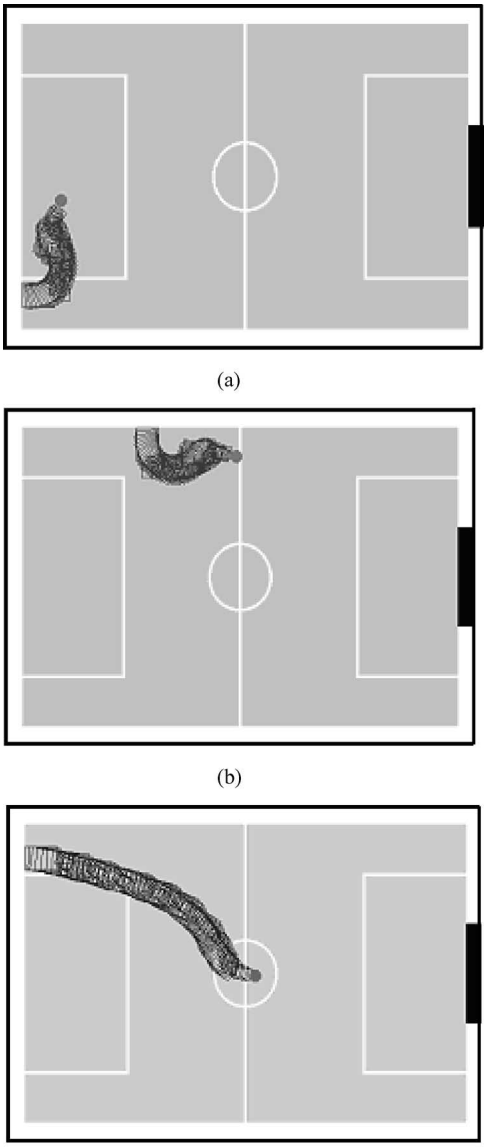


Fig. 22. Application of the best FLC after the structure and parameter learning.

V. Conclusion and Future Work

The coordination architecture proposed in this paper emphasizes feasibility, efficiency, and simplicity. It has three distinctive features such as building a goal-oriented local map, modeling the temporal constraints, and encoding a collection of reactive behaviors. The temporal constraints play a key role to coordinate multiple reactive behaviors and are represented as FSMs to simplify implementation and verification. Each state in an FSM is a reactive behavior that acts as a controller to suppress local perturbation. The proposed coordination architecture also leads to a partitioning of the underlying physical space into a set of stable regions and decreasing the dimension of the input space for behaviors.

In our approach, learning is integrated into an FLC-encoded behavior, which is addressed through both structure- and parameter-learning stages. The strategy is to make use of heuristic experience and autonomous exploration of the robot's environment to yield a good controller. The experimental results on both simulation and real robots show that the coordination architecture is feasible, and the performance of robot behaviors is, indeed, improved through the proposed learning schemes.

Our ongoing research focuses on the following two aspects. First, it is necessary to integrate a learning mechanism into the system to implement the behavior coordination learning without the need for the FSM state inputs to be manually designed. Second, the evolution of real-value parameters of FLC and the fast learning process of coevolution will be implemented on real robots rather than the simulated robots.

Acknowledgment

The authors wish to thank D. Golubovic and B. Li, who made their contribution on the construction of the software environment. They also thank the anonymous reviewers for their constructive comments and suggestions that improved the quality of this paper.

References

[1] J. Albus, "Outline for a theory of intelligence," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 3, pp. 473–509, May/Jun. 1991.
[2] R. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robot. Autom.*, vol. 2, no. 1, pp. 14–23, Mar. 1986.
[3] R. C. Arkin, *Behavior-Based Robotics*. Cambridge, MA: MIT Press, 1998.
[4] R. C. Arkin and T. Balch, "AuRA: Principles and practice in review," *Exp. Theor. Artif. Intell.*, vol. 9, no. 2, pp. 175–189, 1997.
[5] R. G. Simmons, "Structured control for autonomous robots," *IEEE Trans. Robot. Autom.*, vol. 10, no. 1, pp. 34–43, Feb. 1994.
[6] R. J. Firby, "An investigation into reactive planning in complex domains," in *Proc. Natl. Conf. Artif. Intell.*: Seattle, WA, Jul. 1987, pp. 202–206.
[7] M. Georgeff and A. Lansky, "Reactive reasoning and planning," in *Proc. Natl. Conf. Artif. Intell.*, Seattle, WA, Jul. 1987, pp. 972–978.
[8] D. M. Lyons and A. J. Hendriks, "Planning for reactive robot behaviour," in *Proc. Int. Conf. Robot. Autom.* Nice: France, May 1992, pp. 2675–2680.
[9] J. J. Borrelly *et al.*, "The ORCCAD architecture," *Int. J. Robot. Res.*, vol. 17, no. 4, pp. 338–359, Apr. 1998.
[10] D. W. Payton, J. K. Rosenblatt, and D. M. Keirsey, "Plan guided reaction," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, no. 6, pp. 1370–1382, Nov. 1990.
[11] S. Mitra and Y. Hayashi, "Neuro-fuzzy rule generation: Survey in soft computing framework," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 748–768, May 2000.
[12] J. R. Jang, "ANFIS: Adaptive-neural-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 3, pp. 665–685, May/Jun. 1993.
[13] C. T. Lin and C. S. G. Lee, "Neural-network-based fuzzy logic control and decision system," *IEEE Trans. Comput.*, vol. 40, no. 12, pp. 1320–1336, Dec. 1991.
[14] D. Nauck, F. Klawonn, and R. Kruse, *Foundations of Neuro-Fuzzy Systems*, Hoboken, NJ: Wiley, 1997.
[15] H. R. Beom and H. S. Cho, "A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, no. 3, pp. 464–477, Mar. 1995.
[16] H. R. Berenji and P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 724–740, Sep. 1992.
[17] P. Y. Glorennec and L. Jouffe, "Fuzzy Q-learning," in *Proc. 6th IEEE Int. Conf. Fuzzy Syst.*, 1997, vol. 2, pp. 659–662.
[18] L. Jouffe, "Fuzzy inference system learning by reinforcement methods," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 28, no. 3, pp. 338–355, Aug. 1998.
[19] C. T. Lin and C. P. Jou, "GA-based fuzzy reinforcement learning for control of a magnetic bearing system," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 30, no. 2, pp. 276–289, Apr. 2000.

[20] A. Bonarini, "Evolutionary learning of fuzzy rules: Competition and cooperation," in *Fuzzy Modelling: Paradigms and Practice*, W. Pedrycz, Ed. Norwell, MA: Kluwer, pp. 265–284, 1996.
[21] C. F. Juang, J. Y. Lin, and C. T. Lin, "Genetic reinforcement learning through symbiotic evolution for fuzzy controller design," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 30, no. 2, pp. 290–301, Apr. 2000.
[22] S. J. Kang, C. H. Woo, H. S. Hwang, and K. B. Woo, "Evolutionary design of fuzzy rule base for nonlinear system modelling and control," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 1, pp. 37–45, Feb. 2000.
[23] C. L. Karr and E. J. Gentry, "Fuzzy control of pH using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 1, no. 1, p. 46, Feb. 1993.
[24] D. Leitch and P. Probert, "New techniques for genetic development of a class of fuzzy controllers," *IEEE Trans. Syst., Man., Cybern. C, Appl. Rev.*, vol. 28, no. 1, pp. 112–123, Feb. 1998.
[25] V. Matellan, C. Fernandez, and J. M. Molina, "Genetic learning of fuzzy reactive controllers," *Robot. Auton. Syst.*, vol. 25, pp. 33–41, 1998.
[26] Y. Shi, R. Eberhart, and Y. Chen, "Implementation of evolutionary fuzzy systems," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 2, pp. 109–119, Apr. 1999.
[27] L. P. Kaelbling and A. W. Moor, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, 1996.
[28] D. E. Moriarty, A. C. Schultz, and J. J. Grefenstette, "Evolutionary algorithms for reinforcement learning," *J. Artif. Intell. Res.*, vol. 11, pp. 241–276, 1999.
[29] A. Homaifar and E. McCormick, "Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 2, pp. 129–139, Apr. 1995.
[30] R. Volpe, I. Nesnas, T. Estlin, D. Mutz, R. Petras, and H. Das, "The CLARAty architecture for robotic autonomy," in *Proc. 2001 IEEE Aerosp. Conf.*, Big Sky, MT, pp. 10–17.
[31] E. Coste-Maniere and R. Simmons, "Architecture, the backbone of robotic systems," in *Proc. 2000 IEEE Int. Conf. Robot. Autom.* San Francisco, CA, Apr., pp. 67–72.
[32] P. Pirjanian, "Behaviour co-ordination mechanism—State-of-the-art," Inst. Robot. Intell. Syst., Sch. Eng., Univ. Southern California, Los Angeles, Tech. Rep. IRIS-99-375, Oct. 1999.
[33] M. Veloso, E. Winner, S. Lenser, J. Bruce, and T. Balch, "Vision-servoed localization and behaviour-based planning for an autonomous quadruped-legged robot," presented at the AIPS 2000, Breckenridge, CO.
[34] M. Asada, H. Kitano, I. Noda, and M. Veloso, "RoboCup: Today and tomorrow—What we have learned," *Artif. Intell.*, vol. 110, no. 2, pp. 193–214, Jun. 1999.
[35] H. Kitano, M. Fujita, S. Zrehen, and K. Kageyama, "Sony legged robot for RoboCup challenge," in *Proc. 1998 IEEE Int. Conf. Robot. Autom.*, Leuven, Belgium, pp. 2605–2612.
[36] J. Kosecka and R. Bajcsy, "Discrete event systems for autonomous mobile agents," *Robot. Auton. Syst.*, vol. 12, pp. 187–198, 1994.
[37] A. Bonarini, "Comparing reinforcement learning algorithms applied to crisp and fuzzy learning classifier systems," in *Proc. Int. Conf. Learn. Classif. Syst.*, San Francisco, CA: Morgan Kaufmann, 1999, pp. 228–235.
[38] J. Grefenstette and A. Schultz, "An evolutionary approach to learning in robots," presented at the Mach. Learn. Workshop Robot Learn., New Brunswick, NJ, 1994.
[39] A. Ram, R. Arkin, G. Boone, and M. Pearce, "Using genetic algorithms to learn reactive control parameters for autonomous robotic navigation," *J. Adapt. Behav.*, vol. 2, no. 3, pp. 277–303, 1994.
[40] L. Steels, "Emergent functionality in robotic agents through on-line evolution," in *Proc. 4th Int. Workshop Synth. Simul. Living Syst. Artif. Life IV*, R. Brooks and P. Maes, Eds. Cambridge, MA: MIT Press, 1994, pp. 8–14.
[41] H. Hu and D. Gu, "Reactive behaviours and agent architecture for Sony legged robots to play football," *Int. J. Ind. Robot*, vol. 28, no. 1, pp. 45–53, Jan. 2001.
[42] D. Gu and H. Hu, "Learning and evolving of Sony legged robots," in *Proc. Int. Workshop—Recent Adv. Mob. Robots*: Leicester, U.K., Jun. 2000, pp. 52–59.
[43] R. C. Arkin and D. MacKenzie, "Temporal coordination of perceptual algorithms for mobile robot navigation," *IEEE Trans. Robot. Autom.*, vol. 10, no. 3, pp. 276–286, Jun. 1994.
[44] J. T. Feddema, R. D. Robinett, and B. J. Driessen, "Designing stable finite state machine behaviours using phase plane analysis and variable structure control," in *Proc. 1998 IEEE Int. Conf. Robot. Autom.*, Leuven, Belgium, pp. 1134–1141.
[45] K.-Y. Fok and M. R. Kabuka, "An automatic navigation system for vision guided vehicles using a double heuristic and a finite state machine," *IEEE Trans. Robot. Autom.*, vol. 7, no. 1, pp. 181–189, Feb. 1991.
[46] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. Berlin, Germany: Springer-Verlag, 1996.
[47] D. Golubovic and H. Hu, "An interactive software environment for gait generation and control design of Sony legged robots," in *Proc. 6th Int. Symp. RoboCup,* Fukuoku, Japan, 2002, pp. 24–25.

Dongbing Gu (M'01–SM'07) received the B.Sc. and M.Sc. degrees in control theory from the Beijing Institute of Technique, Beijing, China, in 1985 and 1988, respectively, and the Ph.D. degree in computer science from the University of Essex, Colchester, U.K., in 2004.

In 1988, he joined the Department of Electronic Engineering, Changchun Institute of Optics and Fine Mechanics, China, as a Lecturer, where he became an Associate Professor in 1993 and a Professor in 1998. From October 1996 to September 1997, he was a Visiting Scholar at the Department of Engineering Science, University of Oxford, U.K. He is currently a Lecturer in the Department of Computer Science, University of Essex. His current research interests include real-time systems, microprocessors, optical instruments, automatic control, machine learning, and autonomous robots. He is the author or coauthor of more than 50 papers published in journals and conferences.

Dr. Gu has received several research awards in China.

Huosheng Hu (M'94–SM'01) received the M.Sc. degree in industrial automation from the Central South University, Changsha, China, in 1982, and the Ph.D. degree in robotics from the University of Oxford, Oxford, U.K., in 1993.

He is currently a Professor in the Department of Computer Science, University of Essex, Colchester, U.K., where he leads the Human Centred Robotics Group. Since 2000, he has been a Visiting Professor at six universities in China, namely, Central South University, Shanghai University, Wuhan University of Science and Engineering, Kunming University of Science and Technology, Northeast Normal University, and Chongqing University of Post and Telecommunications. His current research interests include autonomous mobile robots, human–robot interaction, evolutionary robotics, multirobot collaboration, embedded systems, pervasive computing, sensor integration, RoboCup, intelligent control, and networked robotics. He is the author or coauthor of more than 180 papers published in journals, books, and conferences. He has served as a member of the Editorial Advisory Board of the *International Journal of Industrial Robots* during 1997–2000, and is currently the Editor-in-Chief of the *International Journal of Automation and Computing*. He is a Reviewer for a number of international journals such as the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, *Automatic Control*, *Neural Networks*, and the *International Journal of Robotics Research*.

Dr. Hu has been a founding member of the IEEE Society of Robotics and Automation Technical Committee of Internet and Online Robots since 2001. During 2001–2004, he was a member of the IASTED Technical Committee on "Robotics." He was the Conference Chairman for the 1st European Embedded Systems Conference, Paris, in 1996, and has been a member of the Program Committees for a number of international conferences such as IASTED Robotics and Applications Conferences (since 2000) and RoboCup Symposiums (2000–2004). He is a Chartered Engineer, and a member of the Institution of Engineering and Technology (IEE), the Association for the Advancement of Artificial Intelligence, the Association for Computing Machinery, the International Association of Science and Technology for Development, and the Intelligent Autonomous Systems Society. He is the recipient of two best paper awards.