Hybrid Learning Architecture for Fuzzy Control of Quadruped Walking Robots

Huosheng Hu,* Dongbing Gu†
Department of Computer Science, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, United Kingdom

This article presents a hybrid learning architecture for fuzzy control of quadruped walking robots in the RoboCup domain. It combines reactive behaviors with deliberative reasoning to achieve complex goals in uncertain and dynamic environments. To achieve real-time and robust control performance, fuzzy logic controllers (FLCs) are used to encode the behaviors and a two-stage learning scheme is adopted to make these FLCs be adaptive to complex situations. The first stage is called *structure learning*, in which the rule base of an FLC is generated by a Q-learning scheme. The second stage is called *parameter learning*, in which the parameters of membership functions in input fuzzy sets are learned by using a real value genetic algorithm. The experimental results are provided to show the suitability of the architecture and effectiveness of the proposed learning scheme. © 2005 Wiley Periodicals, Inc.

1. INTRODUCTION

The complex tasks in the real world require autonomous robots to behave both in an intelligent way and in real time. 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 a sense-model-plan-act manner based on a single execution pipeline given the condition that a predictable world model is available.1 It is obviously difficult to accommodate the sensory uncertainty and the environmental dynamics. In contrast, the behavior-based architectures are able to handle the problems that appear in the deliberative architecture by a bottom-up approach, namely, the coordination of different behaviors. There are a number of representative paradigms, including subsumption2 and schema-based architectures.3 However, these architectures have suffered from difficulties in high-level reasoning, knowledge base, temporal constraints, and system scalability when the number of behaviors and the degree of their interaction increased.

*Author to whom all correspondence should be addressed: e-mail: hhu@essex.ac.uk.
†e-mail: dgu@essex.ac.uk.
Recently, hybrid architecture has emerged as a superior way to build software architecture for both low-level control and high-level reasoning. It integrates behavior-based architecture with deliberative ones to gain the strength of both. Many architectures belong to this category, such as AuRA, TCA, RAP, PRS, RS, and ORCCAD. Most of these architectures have been built in three levels. The lowest level is committed to control robustness, real-time response, and concurrent processing. The highest level performs deliberative computation to divide a task 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, advising, or adaptation. At this level, robots reconfigure their control strategies on the fly according to the sensory information so that dynamic changes in the real world can be handled effectively.

In this article, a hybrid learning architecture is proposed for Sony walking robots to play football in the RoboCup domain. In this architecture, deliberative reasoning or decision making is located in a deliberative level and behaviors are in a reactive level that directly maps sensory information into actions. Each state corresponds to a behavior that is encoded by a fuzzy logic controller (FLC). A two-stage learning scheme is proposed to evolve the FLC. The first stage is the structure learning in which the rule base of a FLC is learned by Q-learning. The second stage is parameter learning in which further fine-tuning of the learned FLC will be carried out. A real value genetic algorithm (GA) is employed for parameter learning. Based on the hybrid architecture, different behaviors could be designed individually to enable walking robots to deal with uncertain sensor readings and imperfect motor actions.

The rest of the article is organized as follows. Section 2 outlines related research work of FLC learning. Section 3 describes the proposed hybrid architecture in which both low-level basic behaviors and high-level cooperative behaviors are adopted for multiple Sony walking robots to operate in the RoboCup competition. Section 4 illustrates two-stage learning of fuzzy behaviors including structure learning and parameter learning. Both simulated and real experimental results are provided in Section 5 to show the learning performance. Finally, conclusions and future work are briefly discussed in Section 6.

2. RELATED WORK

Recently, there has been an increasing tendency to build robot behaviors by using FLCs because human experience can be embedded into the FLC to reduce the search space in the learning process. In a FLC, uncertainty is represented by fuzzy sets and an action is generated cooperatively by several rules, each one triggering to some degree to produce a smooth, reasonable, and robust control effect. The problems in the FLC design include the parameter setting of membership functions and the composition of the rules. They are classified into two categories: structure identification and parameter identification. The structure identification includes partitioning its input space, selecting antecedent and consequent variables,
determining the number of IF-THEN rules, and initializing membership functions. In contrast, the parameter identification determines the parameters of membership functions.

Many learning approaches have been proposed to model a FLC, including neural network (NN) based, reinforcement learning (RL) based, and GA based. 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. The problem for NN-based learning in a FLC 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 because these data pairs need to be learned. What can normally be obtained for evaluating a behavior is a delayed reward, that is, the final outcomes after the behavior is executed. Both GA-based and RL-based learning are two equivalent learning schemes that need a scalar response from the real world by interacting with it to show their performance. 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. 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 a best one. Karr first used a GA to evolve the membership functions of a FLC. Homaifar and McCormick simultaneously evolved the 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. A 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 a FLC. 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 a FLC. The credit assignment for an individual is made implicitly in light of the fact that poor individuals will have few offspring in future generations. Obviously these methods suffer from a large search space so it is more difficult to find optimal or suboptimal solutions. In Refs. 21 and 24, the authors adopted different evolution strategies where an individual represents a rule, not a FLC. Rules in one generation compete with each other to be selected. The credit to individuals is assigned according to their contribution and eligibility traces. To reduce the search space, rules with the same antecedents are grouped into a subpopulation where competence occurs. 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 the 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) and fuzzy Q-learning (FQL). 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 Ref. 19, it was successfully used to learn the actions of a FLC. Actually GA and Q-learning described in Refs. 24 and 19, respectively, were both employed to select the consequences of a FLC. Their difference lies in the learning strategies employed to explore the output space.

3. HYBRID LEARNING ARCHITECTURE

The main objective for our research is to build a firm research platform for future research work on multi-agent learning systems. Currently, a hybrid learning approach is adopted here to achieve real-time performance.26,34,35

3.1. Modular Design

We have adopted a modular design in overall implementation of our hybrid learning architecture,25,36 as shown in Figure 1. More specifically:

- Perception: This module includes a multisensor system and a local map. The sensors being used are a color vision system, a position-sensitive device, five touch sensors, 20 optical encoders, two microphones, and three gyros. Incoming visual, proximity, ranging, and auditory information is processed by an on-board computer. A neural networks-based color detection table is used to handle uncertain and changing lighting conditions.37 A local map is then built and updated dynamically as long as new sensory data are available.
- Cognition: This module consists of both high-level behaviors for learning and collaboration and low-level behaviors for safeguard and game playing. This is a typical hybrid architecture25,35 that is to merge the advantages of the traditional planning-based approach and the behavior-based approach. Based on information from the perception module, it

![Figure 1. Agent architecture adopted at Essex.](image-url)
selects an action to perform and sends the result to the actuators module for execution. This is the most complex module because it does the “thinking” and “reasoning” for each robot agent. More details are given in the next section.

- Action: This module includes one speaker and 20 micro servomotors. Each leg has three joints driven by three servomotors. The synchronization of quadruped legs for each robot is extremely important for robot actions such as kicking the ball and moving toward the goal. A fuzzy logic controller is used to deal with uncertainty in noisy sensory data and imperfect actuators. The speaker is used to communicate with teammates for team formation and cooperation.

3.2. Robot Behaviors

An important issue in the design of the cognition module is to synthesis low-level basic behaviors and high-level cooperative behaviors of multiple Sony walking robots. Low-level behaviors enable individual robots to play a role in a specified task or game. High-level behaviors enable a team of robots to accomplish missions that cannot easily be achieved with an individual mobile robot. Figure 2 shows a hybrid learning architecture for multiple quadruped walking robots deployed at Essex. More specifically, we describe each of these behaviors as follows:

![Figure 2: Hybrid architecture for robot behavior learning.](image)
Walking behavior. A number of motion behaviors such as ForwardWalking, BackwardWalking, SideWalking, and Rolling were designed to make the robot fast and flexible. These behaviors rely on motion gaits that should be generated and optimized in advance. A hybrid GA-based approach has been adopted here to optimize the parameters in walking behaviors.26

Game-playing behaviors, to enable each robot to play a role in the competition, including approaching the ball, kicking the ball, dribbling the ball, passing the ball, intercepting, and shooting the goal. Fuzzy logic controllers are deployed here for all the behaviors to handle motion inaccuracy and sensor uncertainty, which is the main focus in this article.

Communication behavior, to realize interrobot communication by either an explicit way using loud speakers and microphones when possible or an implicit way by observing the motion of other robots.2,38

Localization selection, to find out where each robot is and where the goal is in order to position itself at the optimal position at any moment. This is a reactive planner to generate locally optimal strategies to direct the low-level behaviors of each robot. It is very important for each robot to play an effective role in the competition.

Coordination behavior, to enable multiple robots to form a team where each robot makes its own contribution toward a common goal.38 Note that the role of each robot should be able to be swapped dynamically to achieve optimal performance.

Learning behavior, to learn from its own experience and from other mobile robots. This is a key factor for each robot to improve its performance under an uncertain and dynamic environment.19 Learning here includes both evolving fuzzy rules for low-level behaviors and searching optimal parameters for high-level reactive planning.34,35

Additional cooperative behaviors can be synthesized during the next stage of our research, for instance, homing behavior and role-switching behavior. This should be easy to implement in our modular design.

3.3. Gait Generation and Motion Control

One of the most important advantages of walking robots is their superior mobility and terrain adaptability to wheeled/tracked mobile robots. Walking robots only require a few discrete footholds to travel around for off-road locomotion where the surfaces may be inaccessible to wheeled/tracked robots. To make such attractive characteristics more practical, a motion control algorithm should be developed to search and plan an optimum path and the foothold points, and to keep dynamic stability on a rough terrain.33 Although serious hardware limitations exist, teams with efficient coordination of quadruped leg motion can have major advantages in the RoboCup Sony Legged Robot competition.39

In our design, each leg tip of the robot is designed to move in a rectangle trajectory, as shown in Figure 3. Front and rear legs from the opposite sides are in the same phase and the other two legs are opposite. Projection of the mass of the robot to the ground is on the line that connects to legs, which are in touch with the ground. The center of paw rotation is initially at the same inverse kinematic coordinate for both front legs and both rear legs. This prevents the robot from falling to the side and stabilizes the camera. Gravity sensors are used to obtain information on body position.
The control system structure consists of both kinematics and dynamics levels, as shown in Figure 4. The kinematics level involves two sublevels: a pattern generator and a leg trajectory generator. Each leg has its own trajectory generator that determines the course of the leg endpoint. When a timing signal has been received, the leg must begin its swing/stance cycles. To emulate the accurate foot placement the trajectory generator plans a trajectory in foot position coordinates and then converts them to joint positions using inverse kinematics. The pattern generator provides repetitive motion of a leg and synchronization of movements with the other three legs. Gait planning depends on the velocity and heading of the robot. The time and space coordination of the motion involves a decision regarding which leg should be lifted or placed. It must be made in terms of the condition of terrain, stability requirements, speed requirements, mobility requirements, and power consumption. More details can be found in Ref. 26.

4. FUZZY BEHAVIOR LEARNING

In this section, we present our research on learning fuzzy behaviors for a robot to improve its approaching-ball (AB) behavior, which is one of most important behaviors in a football game. The results from this can be easily extended to other behaviors. In general, the AB behavior makes the robot move as close as possible to the ball and head toward the goal at the end.
4.1. A Fuzzy Reactive Behavior

There are three variables associated with this behavior: the head direction to the ball relative to its body, represented by $\theta$, the distance to the ball, represented by $h$, and the goal’s angle relative to the head direction, represented by $\alpha$ (see Figure 5). A visual tracking algorithm is implemented for the head to track the ball based on color detection. It leads $\theta$ to be expressed by the head’s pan angle, which can be easily read from the encoder mounted on the pan motor when the ball has been 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 the infrared ray. Instead, we use the number of image pixels of the ball \( s \) captured by the CCD camera mounted inside the head 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 \( S = [s_1, s_2, s_3]^T = [\theta, s, \alpha]^T \). Outputs of the behavior are a set of one-step commands such as \( MF(\text{move forward}), LF(\text{left forward}), RF(\text{right forward}), LT(\text{left turn}), RT(\text{right turn}), \text{Backwards}, \text{or Stop} \) 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 \( \text{Small} (S) \), \( \text{Middle} (M) \), or \( \text{Large} (L) \). 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 because the ball may not stay in the image center when the robot moves. We have a set of fuzzy concepts \( \text{Left} (L) \), \( \text{Zero} (Z) \), or \( \text{Right} (R) \) for the pan angle. The angle to the goal is also defined as \( \text{Left} (L) \), \( \text{Zero} (Z) \), or \( \text{Right} (R) \). 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 \( F_{nj} \) as the \( j \)th fuzzy set \( (j = 1 \ldots 3) \) for the \( n \)th 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 Figure 6. 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 \( F_{nj} \) is expressed as

\[
\mu_{F_{nj}}(s_n) = \begin{cases} 
\max \left[ 0, \frac{d - s_n}{d - c} \right] & \text{if } s_n > c \\
\max \left[ 0, \frac{s_n - a}{b - a} \right] & \text{if } s_n < b \\
1 & \text{otherwise}
\end{cases}
\]

Figure 6. A membership function for a fuzzy set.
A linguistic fuzzy rule is expressed as

\[ Ri: \text{If } s_1 \text{ is } F_{1ji} \text{ And } s_2 \text{ is } F_{2ji} \text{ And } s_3 \text{ is } F_{3ji}, \text{ Then } o \text{ is } c^{m_i} \]  

(2)

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

\[ \alpha_{R_i}(S) = \min(\mu_{F_{1ji}}(s_1), \mu_{F_{2ji}}(s_2), \mu_{F_{3ji}}(s_3)) \]  

(3)

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

\[ o(S) = \frac{\sum_{i=1}^{N} \alpha_{R_i}(S) \cdot c^{m_i}}{\sum_{i=1}^{N} \alpha_{R_i}(S)} \]  

(4)

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 its behavior goal are presented in subsection 4.3 and the rule base generation is described in the next section.

### 4.2. Rule Base Generation

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

\[ r = w_1 \cdot \frac{\sum s}{l} + w_2 \cdot (A - l) + w_3 \cdot (B - \theta) + w_4 \cdot (C - \alpha) + w_5 \cdot s \]  

(5)

where \( w_i \) (\( i = 1, \ldots, 5 \)) is the weight, \( A, B, \) and \( C \) are 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 \( \theta, d, \) and \( \alpha \) in the last step, reflect the final position of the robot.

We use the idea of fuzzy Q-learning proposed in Ref. 19 to redefine the rule base of a FLC in Ref. 38 as

\[ Ri: \text{If } s_1 \text{ is } F_{1ji} \text{ And } s_2 \text{ is } F_{2ji} \text{ And } s_3 \text{ is } F_{3ji} \]

Then \( o \) is \( c^1 \) with \( q(i,1) \) or \( o \) is \( c^2 \) with \( q(i,2) \) ... or \( o \) is \( c^7 \) with \( q(i,7) \)  

(6)

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, that is, 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 one of the 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 Figure 7. 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. Figure 8 shows how the robot learns incrementally and the rule base of a FLC is updated from the grid (s is Large) to the grid (s is Small).

It should be noticed that the number of the test grids is finite due to finite fuzzy sets. Therefore, learning does not make use of the stochastic exploration
strategy as normal Q-learning. The regions where the fuzzy sets overlap are not covered in the learning process. But they are implicitly generalized by the rules in the regions where there is no overlap among fuzzy sets. The fine-tuning of a FLC within the overlapped regions will be presented in the next subsection by the parameter learning.

4.3. GA Learning for the Parameters of an FLC

GA used here is to optimize the parameters in membership functions 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.

Conventional GA encodes an individual in form of a discrete or binary string. This kind of encoding generates very long strings that slow down the evolution process. On the other hand, it is difficult to find an optimal individual if the string length is not long enough. 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. 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 Figure 8, where one individual represents one FLC.

The payoffs in Formula 5 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.

- **Reproduction**: Individuals are copied according to their fitness values. The individuals with higher fitness values have more offspring than those with lower fitness values.
- **Crossover**: The crossover will happen for two of the parents that have high fitness values with a crossover probability \( p_c \). One point crossover is used to exchange the genes.
- **Mutation**: The real value mutation is preceded by adding a certain amount of noise to new individuals to produce offspring with a mutation probability \( p_m \). For the \( i \)th gene in the \( j \)th individual \( g_j^i \), it will be evolved to \( \tilde{g}_j^i \) according to the following rules:

\[
\tilde{g}_j^i = \begin{cases} 
  g_j^i + \Delta(t, UB - g_j^i) \\
  g_j^i - \Delta(t, g_j^i - LB)
\end{cases}
\]  

(7)

where \( UB \) and \( LB \) are lower and upper bounds of variable \( g_j^i \). 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. More specifically, it causes the operator to search the space uniformly at an early stage and converge locally at later stages, that is,

\[
\Delta(t, y) = y \cdot r \cdot \left(1 - \frac{t}{T}\right)^b
\]  

(8)
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.

5. EXPERIMENTAL RESULTS

We have conducted several experiments to verify the proposed architecture, rule learning, and fuzzy membership function parameter learning. A simulator was adopted by using the data collected from testing on real Sony robot walking, and then real robots were used in experiments to verify outcomes of the learning. More specifically, the hybrid architecture for an attacker was then tested on both simulator and real robots. The seven behaviors were implemented by handcrafted rules and FLCs. Finally, the FLC learning experiments were implemented by using the GA-based approach.

5.1. The Hybrid Architecture Experiment

In this experiment, the structure learning is implemented, in which the rule base of an FLC was generated by a Q-learning scheme described in Section 4.2. A simulation environment that has the exact size of the pitch and ball is adopted first. The motion of a Sony robot is identified by the consequences of the actual execution of individual walking commands. To reflect real-world uncertainty, Gaussian noise is added in motion equations and sensor readings during the simulation. One-step delay is used in command execution. After simulation, the real robot experiment is then used to validate the proposed hybrid architecture.

The simulation results for an attacker are shown in Figure 9. The attacker is positioned near its own goal and the ball is placed at the other side of the field (Figure 9a). The robot first goes into the looking for the ball (LB) state where looking for ball behavior is executed (Figure 9b). Upon finding the ball, the robot transits into the AB state from the LB state. Figure 9c,d 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 (Figure 9e). Finally, the robot kicks the ball into the goal by a fast-walking command (Figure 9f). The robot took about 96 steps to shoot the ball into the goal from a given initial position.

The pan angle is turned left about 50° to track the ball, as shown in Figure 10a. 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 abruptly, as shown in Figure 10b. As soon as the goal is found, the robot aligns its heading toward the goal. At the same time, it moves its head to track the ball according to the previous position stored in the local map. It should be noticed that the tilt angle decreases again after the 83rd step as shown in Figure 10b.

Figure 11a shows the simulation result for the corresponding heading of the robot, and Figure 11b shows the result for the robot position. The initial position is (750 mm, 680 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 heading changes around the 81st step in Figure 11a.

The real robot experiment is tested to validate the proposed architecture. The LB state is shown in Figure 12a. The endeavor of approaching the ball is shown from Figure 12b to Figure 12d after it was found. Then, the robot looked for and aligned with the goal (Figure 12e), and kicked the ball (Figure 12f).

5.2. The FLC Learning Experiments

In this section, the results from the FLC learning experiments are presented to show its feasibility. The rule base is incrementally built up by a backup updating
approach described in Section 4.2. Figure 13 shows the results for the Approaching-ball 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 did not exactly face the goal at its final position.

After the GA-based parameter learning that is described in Section 4.3, we carried out the same experiments shown in Figure 12 for the robot with the learned FLC. The results are shown in Figure 14, where the improvement in final positions can be seen. In 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 mutation function 8, we assume $r = 0.3$ and $b = 5$. The GA learning process is shown in Figure 15, 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 $(a < b < c < d)$ during GA learning. A validation
process is employed to check the constraints for each FLC before it is used. Invalid FLCs are given up. But the robot can still 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.

Figure 16 shows a comparison made before and after parameter learning. Figure 16a is the case that cannot be generalized, and Figure 16b shows compensation made by parameter learning. After parameter learning, the FLC is further fine-tuned so that fuzzy become more reasonable and the robot achieves its goal with this FLC. Figure 17 shows two successful situations where the best FLC is applied after its evolving. More specifically, Figure 17a shows the robot has approached the ball and aligned its heading toward the goal. In Figure 17b, we can see that the robot has also reached the ball and faced to the goal although the ball is far from the goal.

Figure 11. Simulation results of the position and heading: (a) The heading angle; (b) The position change.
6. CONCLUSIONS AND FUTURE WORK

This article presents a hybrid learning system for quadruped walking robots in the robot football domain in order to investigate both behavior coordination and embodied intelligence. Each robot requires sensor-motor-skills to shoot the ball, to dribble and pass the ball, to avoid serious crashes with other players, and so on. This is a very challenging task, as the strategies and behaviors of the opponents are uncertain. Because Sony legged robots have very limited on-board computing power, many useful image processing algorithms cannot be adopted directly for real-time implementation. It remains a big challenge to produce an effective vision algorithm for Sony robot to play a fast football game. In general, hybrid architectures integrate deliberative reasoning with reactive behaviors to provide a robot with the ability to achieve more complex goals. Many variants have been developed based on different tasks, environments, and robots. The hybrid software architecture proposed in this article emphasizes feasibility, efficiency, and simplicity through building a goal-oriented local map, modeling the temporal constraint, and
encoding a collection of reactive behaviors. The temporal constraints play a key role to coordinate multiple reactive behaviors and are represented as finite state machines to simplify implementation and verification. Each state in a FSM is a reactive behavior that acts as a controller to suppress local perturbation.

The proposed 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. This hierarchical fuzzy logic control is different from the one in Ref. 41, where their coordination between behaviors is called context-dependent blending. Learning is integrated into a behavior that is encoded by an FLC. Learning an FLC 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 hybrid architecture is feasible and the performance of robot behaviors is indeed improved through the proposed learning schemes.

We are currently investigating (i) how low-level behaviors such as kicking, passing, and intercepting are effectively integrated with high-level behaviors, that is, position selection and team formation; (ii) what data fusion algorithms are required to capture environment features effectively and deal with uncertainties; and (iii) how adaptation and learning algorithms should be adopted to make robotic
systems more flexible in a dynamic environment. These tasks are not only useful in the RoboCup domain, but also significantly important for many real-world applications. The advantages of cooperative mobile robots over a single complex mobile robot include the potential for increased fault tolerance, simpler robot design, and wide application domain. Further research will focus on learning in the deliberative reasoning where currently the division of sensory space only depends on our ambiguous experience. We believe that learning can fine-tune this division to achieve better performance. Another research focus in the future will be on the

![Figure 14. Parameter learning.](image)

![Figure 15. Evolution process of FLC parameter learning.](image)
coevolution of multiple robots, which will speed up the learning processes. The robots will not only meet more complex situations during the learning process, but also learn more cooperation behaviors to replace straightforward position cooperation currently being addressed.

Acknowledgments

We wish to thank both Dragos Golubovic and Bo Li who have made their contributions on adaptive gait generation and visual perception algorithms.

References