

# Teaching Robots to Coordinate their Behaviours

Dongbing Gu and Huosheng Hu

Department of Computer Science  
University of Essex, Colchester, UK  
Email: [gdu@essex.ac.uk](mailto:gdu@essex.ac.uk), [hhu@essex.ac.uk](mailto:hhu@essex.ac.uk)

**Abstract**—Behaviour co-ordination is one of the major problems in behaviour-based robotics. This paper presents a teaching method for mobile robots to learn the behaviour coordination. In this method, the sensory information is abstracted into a limited number of feature states that correspond to physical events in the interactive process between a robot and its environment. The continuous motor actions are abstracted into a limited number of behaviours. Then, the goal of the behaviour co-ordination is to map the feature states into the behaviours in the light of environment rewards. The teaching process consists of an imitation stage and an autonomous learning stage. Both stages employ Q-learning algorithms to implement the mapping. The imitation stage serves as a preliminary stage for the teaching method. The learning result will be used to bootstrap the autonomous learning stage. Experiments are conducted in the domain of soccer playing of Sony legged robots. Experiment results show that the robot can acquire the behaviour coordination ability.

**Keywords**—Robot learning, Reinforcement learning, Behaviour co-ordination

## I. INTRODUCTION

In the behaviour-based robotics, a robot is equipped with a group of basic behaviours or skills [1][3]. Each of these behaviours uses sensory information to produce immediate actions. The robot accomplishes its goal through coordinating the behaviours [3]. To coordinate behaviours, the robot needs to take into account the physical events occurred in the interaction between the robot and its environment. These events can be described by feature states extracting from sensory information. The dynamics of the behaviour coordination can be characterized by the behaviours and the feature states, which can be formulated as a MDP. Once the MDP is modeled, the robot can map the events into behaviours to achieve the behaviour coordination.

In a MDP, Q-learning [14][6] can be employed to acquire the behaviour co-ordination mechanism through the continuous interaction between the learning robot and its environment [8]. The learning objective is to find the optimal Q values that represent the predicted accumulative payoffs for the feature state-behaviour pairs.

During the Q-learning process, although the abstraction of states and actions into feature states and behaviours enables Q-learning algorithms to operate within a limited discrete space, the learning system knows nothing about the interaction between the robot and its environment at the early stage. The learning system has to choose arbitrary behaviours to try until a payoff is received. Long time could be lost before useful and important areas are explored. An efficient way to improve Q-learning algorithms is to incorporate prior experiences into the learning algorithms. The teaching method is one of the best ways to

incorporate prior experiences [2][12]. In the teacher method, a teacher shows several lessons to the learner. The learner converts the lessons into Q values and uses them to bootstrap next stage learning. A teacher could be a human operator or a control algorithm.

Using a neural network as the knowledge representation to implement the teaching method was reported in [11]. Its goal is to control a real vehicle moving on highway. A human driver is employed to provide the lessons for the learner. In [7], Lin used hand-coded sequences of experiences to bootstrap value functions of reinforcement learning algorithms in order to speed up the learning process. Other similar researches are carried out for learning individual behaviours [4][13].

In this paper, a teaching method is employed to incorporate prior experiences into Q values. A two-stage Q-learning is developed. In the first stage, a human operator chooses behaviours for the robot to run. During the running, the robot accumulates the human operator's experiences in the form of Q values. In the second stage, the robot treats these prior Q values as an initial setting. A standard Q-learning is then employed to autonomously update the prior Q values to find final results.

This paper is organised as follows. Section II presents how to model the behaviour coordination, including a brief introduction to the learning model, the learning task, and the environment setting. Section III describes the learning method for the behaviour coordination, which includes an imitation stage and an autonomous learning stage. Section IV reports the evaluation experiments, including the experimental set-up and results. The conclusions and future work are summarised in section V.

## II. Q-LEARNING SETTING

### A. Learning Model

A MDP is defined to formulate the behaviour coordination, which includes an action space (behaviour set) and a feature state space. The coordination purpose is to find optimal rules or functions to map the feature states into the behaviours. The robot observes the current feature state  $ps_t$ , takes a behaviour  $b_t$ , moves into next feature state  $ps_{t+1}$ , and gains a payoff  $r_t$  from the environment. The optimal policy can be found by maximising the accumulated rewards:

$$E \left( \sum_{t=0}^{\infty} \gamma^t r_t \right) \quad (1)$$

where  $\gamma$  is a discount factor.

When a robot interacts with its environment, it can collect a data vector  $(ps_t, b_t, r_t)$  at each time step  $t$ . After a learning trial is completed, a set of data vectors is collected. In the Q-learning,

the collected data is compressed into a table  $Q(ps_t, b_t)$ , in which each item is an estimated accumulative reward for the robot in the feature state  $ps_t$ , taking the behaviour  $b_t$ . The Q table can be updated during the learning when more trials are run. The Q values of all feature state-behaviour pairs are updated by following learning rule:

$$Q(ps_t, b_t) = Q(ps_t, b_t) + \beta(r_t + \gamma \cdot \max_{b_{t+1}} Q(ps_{t+1}, b_{t+1}) - Q(ps_t, b_t))e(ps_t, b_t) \quad (2)$$

The eligibility trace  $e(ps, b)$  is a record of the occurrence of visiting the feature state-behaviour pair  $(ps, b)$ , which can be used to solve the credit assignment problem[14]. It can be updated by following:

$$e(ps, b) = \begin{cases} 1 & ps_t = ps, b_t = b \\ e(ps, b)\gamma\lambda & otherwise \end{cases} \quad (3)$$

where  $\lambda$  is the recency factor, representing the exponential decay rate of the eligibility trace. After the learning, the final optimized behaviour  $b_t^*$  at the feature state  $ps_t$  can be acquired by:

$$b_t^*(ps_t) = \arg \max_{b_t} Q(ps_t, b_t) \quad (4)$$

## B. The Task

For a specific task, a group of primary behaviours can be developed according to domain knowledge about the task. These behaviours are the basic skills that the robot has to have in order to accomplish the task. The following five behaviours are pre-designed for Sony legged robot to play soccer:

- Find-ball behaviour ( $b_1=FB$ ): The robot starts to scan the field through its head motion. Simultaneously the robot rotates its body to increase its view and gradually moves towards the centre of the field.
- Chase-ball behaviour ( $b_2=CB$ ): The robot moves its head to track the ball. It moves towards the ball and tries to go behind the ball to face the goal in order to kick the ball.
- Align with goal behaviour ( $b_3=AG$ ): The robot slightly adjusts its position to align the ball, its orientation with the goal and in order to kick the ball into the goal.
- Kick-ball behaviour ( $b_4=KB$ ): some specialised kick skills are applied.
- Dribble-ball behaviour ( $b_5=DB$ ): The robot kicks the ball by using its feet and keeps its body moving.

Often, for robot tasks, some significant physical events are existed. These events represent critical points in sensory state space and can be described by binary values. The combination of the significant physical events forms a feature state space. Many behaviour-based robot systems adopt this concept to design learning algorithms. In [8], two physical events are used to define a conceptual state space for Q-learning. In [9], five physical events are used to define the behaviour conditions for Q-learning. In [5], a predicate state space is defined for the low level gait controllers.

Formally, each physical event can be represented by a binary state  $p$  ( $p=1$  for the event occurring). For a specific task, there are  $n$  physical events ( $p_1, \dots, p_n$ ) can be defined to characterise

the task. The combination of these events or binary values constitutes a feature state  $ps = p_1 \dots p_n$ . There are  $2^n$  feature states in total in the feature state space. Decision tree or other clustering techniques can be used to map the sensory physical states into the feature states. In this research, four physical events are defined for a robot to play soccer,  $p_1$ : the ball is found,  $p_2$ : the ball is near enough,  $p_3$ : the robot is behind the ball, and  $p_4$ : the robot is aligned with the goal. For example,  $ps=1010$  represents the robot find the ball and is behind the ball, but not near enough and not aligned with the goal.

## C. Environment rewards

In Q-learning, rewards can be singular values to indicate if goals or sub-goals are achieved and usually cannot be obtained immediately. These rewards are sparse in both temporal and spatial senses. For complex tasks, Q-learning would take long time to obtain sparse rewards. To avoid wasting time, the dense rewards [13] or progress estimators [9] are necessary for real robots.

The rewards are the payoffs returned from environment to indicate the effect of robot's actions taken so far. Due to noisy sensory data, the dense rewards may contain uncertainty as well or be wrongly interpreted by perceptual algorithms. Therefore, it is necessary to provide accurate information about the interaction between robots and environments to robots. The information should be produced from those sources that act as assessors to evaluate the robot behaviours.

In this research, the experiment environment includes a playing pitch, a Sony legged robot, a ball, and a global monitor system (see figure 1(a)). The monitor system is adopted to act as the assessor and consists of an overhead camera, a desktop computer, and visual tracking software. The monitor recognises the robot and the ball according to their colours and tracks them. The tracking results and the judgment about if a goal is scored are transferred to the robot. The monitor and the robot form a client-server architecture. The robot works as a client and continuously asks for information from the server (the monitor). Since the monitor system updates its image and tracking results in the rate of about 30 frames per second and the robot operates in the rate of about 10 frames per second, the monitor system has no significant effect on the learning process. Figure 1 (b) shows a top view of the playing pitch from the overhead camera. The goals are centred on both ends of the field. Six unique coloured beacons are placed around edges of the pitch, with one at each corner and one on each side of the halfway line.

The robot adopts both the sparse and dense rewards during its learning and these rewards come from both on-board sensors and the monitor system. They include:

- Task reward: this is the most significant reward for the learning robot. It is rewarded for their achievement toward the task goals. For the soccer playing robots, shooting a goal is designed as a task reward and will be rewarded by the monitor system.
- Sub-task rewards: these rewards are provided to evaluate the performance of individual behaviours. They can be sub-task goals denoted by sparse values or progress

estimation in a dense form. Two sub-task rewards are used in the soccer playing robots. One is the distance between the robot and the ball, which is provided by the monitor system. Another is the angle between the robot and the ball, which is taken from the on-board sensors. These rewards can continuously evaluate the robot performance.

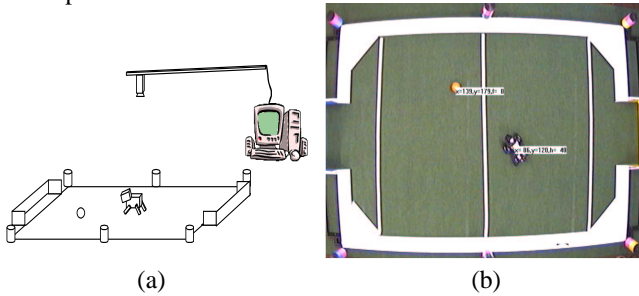


Figure 1 The learning environment

### III. LEARNING SCHEME

Teaching robots is different to programming robots. Most programming methods are designed for specific tasks. Changes of the tasks require re-programming. In contrast, the teaching methods provide a structure for domain knowledge representation. This structure can be a table in Q-learning, a weighted network in neural network learning, or other parameterised function approximators. The lessons or prior experiences received from teachers will be compressed into the structure representation.

Using reinforcement learning, and Q-learning in particular, to acquire control algorithms for robots enables human designers to concentrate on task description in the form of rewards. Generally, designing rewards is easier than designing the control algorithms, especially in noisy and unpredictable environments. The control algorithms are acquired automatically through learning guided by the rewards.

In the teaching method, the task specifications are first interpreted into rewards. It is followed by leading the robot to some interesting areas through the imitation stage. Based on the learned knowledge about the interesting areas, the robot autonomously explores the environment to update the utility values of state-action pairs in the autonomous learning stage. Finally, the robot simply searches the Q table to find those state-action pairs with maximum utility values.

#### A. Imitation Stage

This stage is to provide initial Q values to the Q-learning algorithms of the second stage in order to guide the learning to search interesting areas rather than blindly explore solution spaces at early learning stage. With these prior experiences, the robot could behave reasonably.

The teaching consists of several lessons, each of which is a trial starting from the robot's initial position and running until terminal conditions are met. The terminal conditions are either a goal is shot or the time for this trial is up. Figure 2 shows the flowchart of the imitation stage. There are two loops that are run

on both the robot (left loop) and the computer of the monitor system (right loop) asynchronously and communicate with each other through the Internet. In the left loop, when the robot is started, it localises itself first in order to extract the environment information. The feature states are then sent to the monitor system to help the teacher to select the behaviours. The robot executes one step of the behaviour selected by the teacher. Next it checks if the terminal conditions are met. If not, the robot goes to send the feature states and run the loop again. Finally the robot stops moving and starts to run an off-line Q-learning algorithm to update the Q table. The final Q table is sent back to the monitor system. The robot completes its current lesson and goes back to localization again for next lesson.

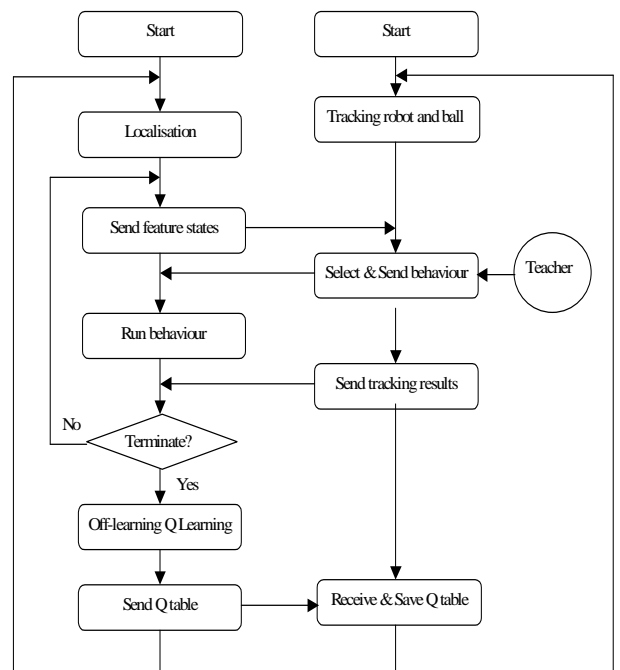


Figure 2 The flowchart of teaching stage

In the right loop, the monitor starts from the object-tracking algorithm that searches the ball and robot in the images and provides their positions to the robot. The teacher selects one of behaviours according to his view of the pitch and the feature states received from the robot. The selected behaviour and the tracking results are both sent to the robot for its learning process. The monitor system also records all of the learnt results, i.e. Q tables, when one lesson is completed.

During this stage, the Q table contains the knowledge learned from lessons. The robot completely follows the instructions and collects whole data set of the behaviours, feature states, and the rewards. After one lesson, it employs an off-line Q-learning algorithm to update the Q table. The teacher plays a key role in this stage, who decides which behaviour is to run next. Since a behaviour is a temporal extended action (not a single action), the teacher has time to make the decisions. The instructor may make mistakes (the wrong behaviour is selected). It means the lesson

may contain wrong data. However, the knowledge obtained in this stage is only used for initial values in the next Q-learning stage.

The off-line Q-learning algorithm is listed in figure 3. The updating of the Q values is only executed when a trial is finished (at step f).

- 1) *Initialisation. The initial behaviour is set as the find-ball behaviour.*
- 2) *If enough learning trials are completed, exit*
- 3) *Self-localisation.*
- 4) *Loop.*
  - a. *Perceive environment.*
  - b. *Extract the feature states.*
  - c. *Receive the behaviour instructed by teacher.*
  - d. *Run the behaviour.*
  - e. *Store the feature state, behaviour, and reward.*
  - f. *If the terminal conditions are met*
    - i. *Q value updating using (2) (3) for all collected data.*
    - ii. *Send Q table back to the monitor system for recording.*
    - iii. *Back to step 2) for next trial.*
  - g. *Otherwise, back to step 4) for next loop.*

Figure 3 The off-line Q-learning algorithm.

#### B. Autonomous Learning Stage

In this second stage learning, the robot starts to learn autonomously without the teacher's help. It is a standard Q-learning process, in which the robot needs to explore the solution space and exploit the learned experiences. The monitor system is still required to provide the rewards to the robot and save the Q table, but without the need to send the instructions. The learning system still follows the similar procedure as figure 3. Only three different steps are taken.

- The robot has to select its behaviour rather than follow the received one. The  $\epsilon$ -greedy method is employed to balance the exploration and exploitation. With the prior experiences learned in the imitation stage, the robot can act reasonably rather than randomly during the early stage though some behaviours are selected randomly with a small exploration probability.
- An on-line Q-learning version is implemented (see figure 4) to replace the off-line Q-learning. In the on-line version, the Q values are updated when the robot changes from one feature state to another. There is no need to wait for the terminal conditions.
- There is no teacher to intervene the learning process at the monitor system's side. The monitor system works as a server to supply the rewards to the robot client.

The on-line Q-learning algorithm is listed in the following. The updating of the Q value is not carried out at the end of each loop. It is executed when the feature states changes (at step c).

- 1) *Initialisation. The initial behaviour is set as the find-ball behaviour.*

- 2) *If enough learning trials are completed, exit*
- 3) *Self-localisation.*
- 4) *Loop.*
  - a. *Perceive environment.*
  - b. *Extract the feature states.*
  - c. *If the feature states have changed*
    - i. *Calculating the rewards.*
    - ii. *Q value updating using (2)(3).*
    - iii. *Select a behaviour using the  $\epsilon$ -greedy algorithm.*
  - d. *Run the behaviour.*
  - e. *If the terminal conditions are met*
    - i. *Send Q table back to the monitor system for recording*
    - ii. *Back to step 2) for next trial.*
  - f. *Otherwise, back to step 4) for next loop.*

Figure 4 The on-line Q-learning algorithm.

## IV. EXPERIMENT RESULTS

### A. Experiment Setting

The experiments are conducted on the real Sony legged robot, which is designed for RoboCup competition. The robot and the ball are placed on the pitch. The robot needs kick the ball into a goal. The behaviours and feature states defined in section II-B are used to achieve this task.

All experiments are conducted in five situations, each of which is called a play and has different setting for initial ball position and initial robot position. Figure 5 shows these five plays with the arrows pointing to the initial positions.

The initial Q table is acquired first through the imitation stage in which the learning algorithm (figure 3) is run based on the data set. After the first stage learning, the learning robot is run based on the Q-table to evaluate the performance. Then, the autonomous learning is conducted. Finally the learning robot is run again based on the final Q table to evaluate the performance. The experiment procedure is as following:

Phase 1: Five teaching lessons are taken during the imitation stage. For each play, one lesson is run. The off-line Q-learning is run at the end of each lesson.

Phase 2: Demonstration after the teaching: 30 trials are run after the imitation stage (the off-line Q-learning) by using those behaviours whose Q values are maximal. For each play, six trials are run.

Phase 3: Autonomous learning stage: 30 trials are run by using the on-line Q-learning algorithm shown in figure 4. For each play, six trials are run.

Phase 4: Demonstration after the autonomous learning: 30 trials are run after the on-line Q-learning by using those behaviours whose Q values are maximal. For each play, six trials are run.

In the Q-learning algorithms, the  $\epsilon$ -greedy algorithm uses  $\epsilon=1\%$  to explore the action space, i.e. 99% behaviours are selected by maximizing Q values and only 1% behaviours are selected randomly. The discount factor is selected as  $\gamma=0.9$ . The

larger the discount factor, the more the learning algorithm exploits the Q value predictions. The recency factor  $\lambda$  is selected as 0.9, which indicates the exponential decay rate of the eligibility trace. The learning rate of the Q values is selected as  $\beta=0.1$ . The Q-learning terminal conditions are either a goal is scored or 2000 steps are executed.

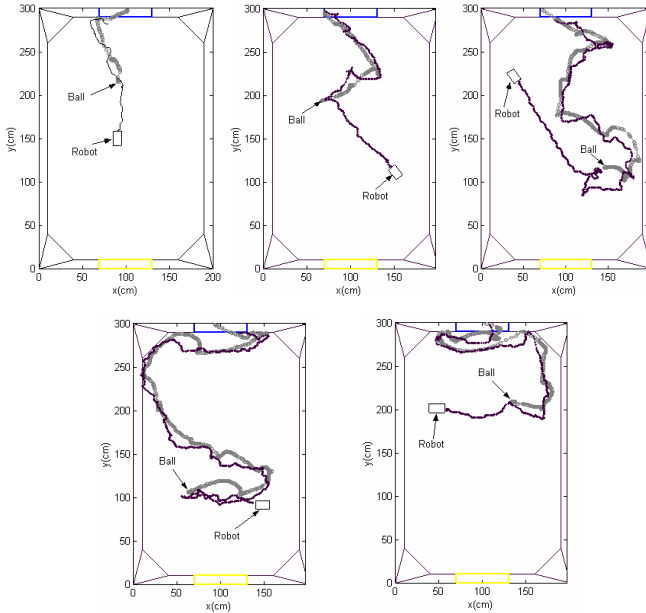


Figure 5 Five plays

## B. Results

The phase 1 is an off-policy learning process where the executed behaviours are selected by the teacher, which may not be necessarily same as the behaviours that the robot wants to learn. The phase 3 is an on-policy learning process where the behaviours executed are determined by the  $\epsilon$ -greedy method and only 1% behaviours are selected randomly.

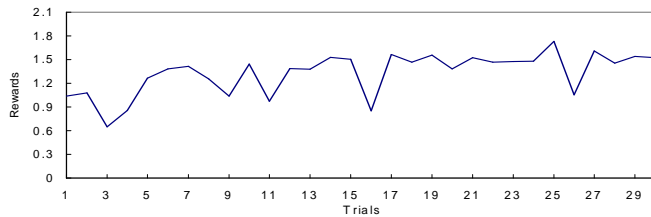


Figure 6 The average received rewards

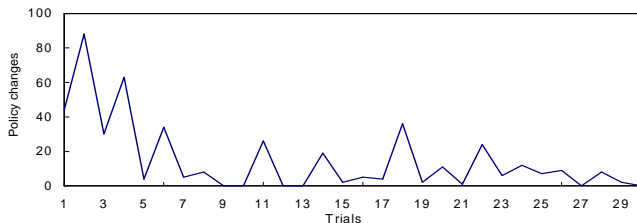


Figure 7 The number of the policy changes.

Figure 6 shows the increasing trend of the average rewards received during the phase 3. Several decreasing points in the curve reflect the exploration mechanism. The number of the policy changes is also decreased as shown in figure 7, indicating that the learning is going toward the steady results.

The estimated payoffs are distributed into the Q table. Figure 8 shows the Q value changes of a feature state during the learning process. The first five trials are run during the imitation stage (phase 1) and other 30 trials are run during the on-line Q-learning (phase 3). The CB behaviour wins at the end of the imitation stage. However, the situation changes during the on-line Q-learning where the FB behaviour outperforms the CB behaviour. The decreasing of the maximum Q value in figure 8 demonstrates that the knowledge acquired during the imitation stage can be corrected by the Q-learning at the autonomous learning stage.

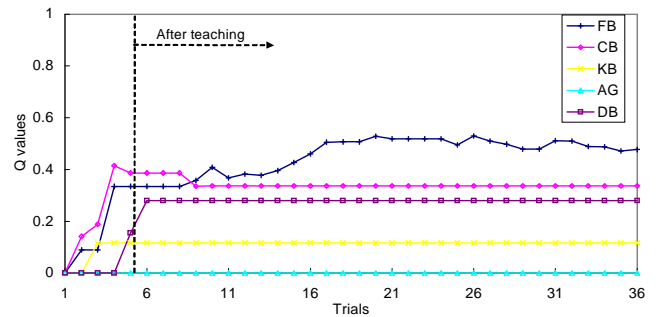


Figure 8 Q value changes in a feature state.

At the autonomous learning stage, the 30 learning trials are run in six rounds with the order play1, play2, play3, play4, and play5. The average steps in each round are calculated and displayed in figure 9. It indicates that the average steps in six rounds are gradually decreased and the standard deviations are also decreased. The robot gradually scores a goal in less steps or shorter time.

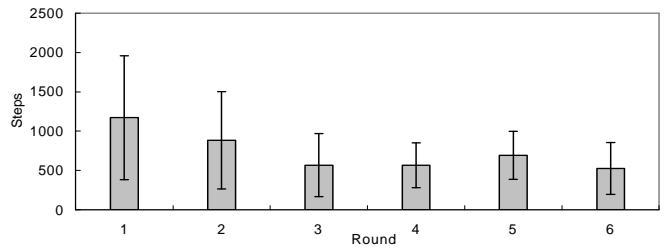


Figure 9 The average steps of a trial in six rounds.

The experiments are further evaluated in terms of the average steps of a trial, average rewards of a trial, and the number of failures for shooting a goal in 2000 steps. In figure 10, the average steps of a trial for both phase 2 and 4 are compared. The white bars represent the results after imitation stage (phase 1) and the grey bars represent the results after the autonomous learning stage (phase 3). Due to the different situations in five plays, they are separately compared. In all five plays, the average steps used to score a goal after the autonomous learning are less than those used after the imitation stage. The grey bars also show that the average steps used in five plays increase in the order of play1,

play2, play5, play4, and play3. This order coincides with the difficulties in the initial settings shown in figure 5. Figure 11 shows the average received rewards of a trial. Again in each play, the results after the autonomous learning stage are better than those after the teaching stage.

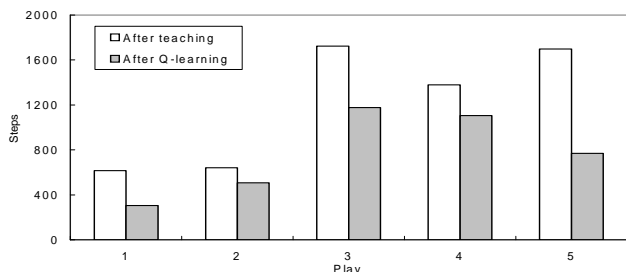


Figure 10 The average steps of a trial in a play.

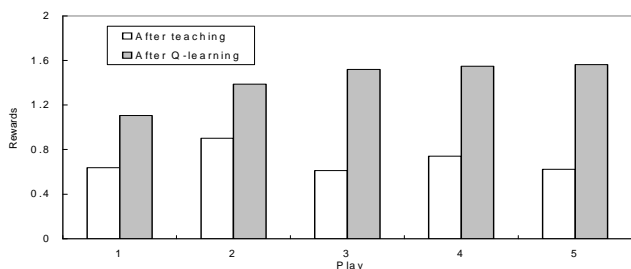


Figure 11 The average rewards of a trial in a play

The robot may fail to score a goal during a trial due to the uncertain nature of the experiments. Figure 12 shows the total number of the failures after 1500 steps for each round. The number in each round after the autonomous learning stage is less or equal to the number after the imitation stage.

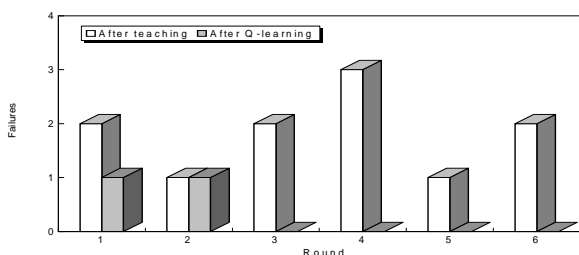


Figure 12 The total number of failures after 1500 steps for each round

The five successful plays are displayed in figure 5. The dark curves represent the robot's trajectories and the light curves represent the ball's trajectories. The robot can move to the ball and manoeuvre the ball into the goal. It demonstrates that the robot acquires the behaviour coordination strategy after the learning.

## V. CONCLUSION

This paper focuses on how to use Q-learning algorithms to learn the behaviour co-ordination. The continuous operation spaces are converted into discrete spaces. It is implemented by

the behaviours design for motor actions and the feature states extraction for sensory physical states. A teaching method is developed to speed up the learning process for real robots, which includes two learning stages. In the first stage, the robot gains prior experiences through teacher's instructions. These experiences can then lead the robot to explore interesting areas in the solution space rather than randomly searching without any experiences at the early stage of learning. In the second stage, the robot makes use of the learned Q values to effectively learn the behaviour coordination mechanism. The experiment results show the robot can achieve this goal.

Further investigation on this method for behaviour coordination should be carried out on larger or more complex systems to evaluate its effectiveness. Exploring if the imitation stage would lead to the learning converging to local minima could be an interesting topic for the future work.

## REFERENCES

- [1] Arkin, R. C., Behaviour-based Robotics, The MIT Press, 1998.
- [2] Breazeal, C. and Scassellati, B., Robots that imitate humans, Trends in Cognitive Science, vol. 6, 2002, pages 481-487.
- [3] Brooks, R., A Robust Layered Control System for a Mobile Robot, IEEE J. Robotics & Automation, Vol. 2, No.1, 1986, pages 14-23.
- [4] Dorigo, M. and Colombetti, M., The Role of the Trainer in Reinforcement Learning, Proceedings of MLC-COLT Workshop on Robot Learning, July 10th, New Brunswick, NJ, 1994.
- [5] Huber, M., A hybrid Architecture for Hierarchical Reinforcement Learning, Proceeding of 2000 IEEE Int. Conf. on Robotics & Automation, Detroit San Francisco, April 2000, pages 3290-3295.
- [6] Kaelbling, L. P. and Moor, A. W., Reinforcement Learning: A Survey, J. of AI Research, 4 (1996), pages 237-285.
- [7] Lin, Long-Ji, Self-improving Reactive Agents Based on Reinforcement learning, planning, and teaching, Machine Learning, Vol. 55, 1992, pages 293-321.
- [8] Martinson, E., Stoytchev, A., and Arkin, R., Robot Behavioural Selection Using Q-learning, Proceedings of 2002 Int. Conf. on Robotics and Automation, Washinton D.C., May 2002.
- [9] Mataric, M. j., Reinforcement Learning in the Multi-Robot Domain, Autonomous Robots, 4(1), March 1997, pages 73-83.
- [10] Pirjanian, P., Behaviour Co-ordination Mechanism - State-of-the-art, Tech-report IRIS-99-375, Institute for Robotics and Intelligent Systems, School of Engineering, University of Southern California, October, 1999.
- [11] Pomerleau, D., Neural Networks for Intelligent Vehicles, Proc. of the Intelligent Vehicles Conference, July 1993, pages 19 - 24.
- [12] Schaal, S., Is Imitation Learning the Route to Humanoid Robots? Trends in Cognitive Sciences, vol. 3, 1999, pages 233-242.
- [13] Smart, W. D. and Kaelbling, L. P., Effective Reinforcement Learning for Mobile Robots, Proceedings of the IEEE International Conference on Robotics and Automation, 2002.
- [14] Sutton, R. S. and Barto, A.G., Introduction to Reinforcement Learning, MIT Press, 1998.