Non-Physical Intervention in Robot Learning Based on LfE Method

Minoru Asada, Shoichi Noda, and Koh Hosoda*

**Dept. of Mech. Eng. for Computer-Controlled Machinery Osaka University, 2-1, Yamadaoka, Suita, Osaka 565, Japan asada@robotics.ccm.eng.osaka-u.ac.jp*

Abstract. *This paper proposes an efficient method of robot learning by which a set of pairs of a state and an action are constructed to achieve a goal. Basic ideas of our method are as follows: i) Since autonomous construction of state and action spaces is generally a very difficult problem, we construct a state space so that a group of situations in which an action command to achieve the goal is the same can be merged into one state even if these situations appear to be different from each other. An action is defined as a sequence of the same action command in such a state. ii) Following the LEM (Learning from Easy Missions) paradigm (Asada* et al.*, 1995), we first find a set of states (in terms of action) closest to the goal state, and then find a set of states closest to the set found previously. iii) In order to reduce an enormous number of trials to find such states, we place a robot so that it can observe objects which the state space consists of (in our case, a ball and a goal). iv) During the above process, the optimal action to achieve the goal is found in every state. This means that a robot can take an adequate action to achieve the goal from every state. We show the experimental results using a real robot system.*

Key Words. Robotics, Robot Learning, Non-Physical Interaction, State and Action Space Construction

1 Introduction

Building a robot that learns to perform a task has been acknowledged as one of the major challenges facing Robotics and AI. Reinforcement learning has recently been receiving increased attention as a method for robot learning with little or no *a priori* knowledge and higher capability of reactive and adaptive behaviors (Connel and Mahadevan, 1993*a*). In the reinforcement learning method, a robot and an environment are modeled by two synchronized finite state automatons interacting in a discrete time cyclical processes. The robot senses the current state of the environment and selects an action. Based on the state and the action, the environment makes a transition to a new state and generates a reward that is passed back to the robot. Through these interactions, the robot learns a purposive behavior to achieve a given goal.

To apply robot learning methods such as reinforcement learning to real robot tasks, we have to deal with two issues. First, we need an enormous number of trials (examples), which is a reason why PbD (Programming by Demonstration) researchers are disappointed with LfE (Learning from Examples) methods. Second, we need well-defined state and action spaces by which a robot correctly learns to perform the task at hand.

For the first problem, human interventions from a variety of aspects have been conducted so as to reduce the number of enormous trials. One of such interventions, the learning by watching method (hereafter, LBW), in real robot applications has been done by Kuniyoshi et al. (Kuniyoshi and Inoue, 1993). They directly showed performances of a human operator in front of a robot with high speed visual tracking routines so as to teach how to perform a given task in a qualitative manner, depending on the knowledge database on semantics of human actions and their meanings. Here, we call this kind of intervention "physical intervention" in robot learning. The physical intervention seems very efficient and useful for human operators. However, the robot system seems to need much amount of knowledge and sophisticated perception capabilities. Therefore, the number of followers is not so many (Kang and Ikeuchi, 1994).

For the various kinds of situations and tasks, it seems difficult to make the physical intervention always possible. On the other hand, non-physical intervention seems feasible for many kinds of situations although various kinds of non-physical interventions have their own advantages and disad-

vantages. Task decomposition (Connel and Mahadevan, 1993*b*) can be regarded as a typical non-physical intervention in robot learning to reduce the number of trials. Also, learning from external critic (hereafter LEC) (Whitehead, 1991) is another example of non-physical intervention technique in reinforcement learning to speed up the learning time. These techniques need accurate knowledge on the properties of the whole task. For example, in the case of task decomposition, decomposed tasks should be independent of each other and no interference should occur. In the case of LEC, the external critic should always correctly advise the agent to guarantee the convergence of the learning.

In our previous work (Asada *et al.*, 1994*b*),(Asada *et al.*, 1994*a*) in which a soccer robot learned to shoot a ball into a goal using only visual information, we proposed a Learning from Easy Missions paradigm to speed up the learning time with less knowledge of the task performance such as when, what, and how to do. In the LEM paradigm, non-physical intervention is realized as a learning schedule for the agent from where the trials should start given the state and action spaces. The learning time depends on the accuracy of the knowledge which situation is easier than other. Rather, the plausible point of the LEM is that we can roughly use less accurate knowledge. This is the difference from other non-physical interventions such as task decomposition and LEC which need accurate knowledge.

The second problem is called the "state-action deviation problem" (Asada et al., 1995) which occurs when we construct state and action spaces in a way that they reflect physical sensors and actuators. In our previous work (Asada *et al.*, 1994*b*; Asada *et al.*, 1994*a*), we had this problem due to the difference in resolution between a robot action in a 3-D space and its projection onto a 2-D image. We solved this problem by restructuring the action space so that one action may cause one state transition. That is, first we fix the state space, and then construct the action space so that the state and action spaces can be consistent with each other. While, one can construct a state space fixing the action space first (Chapman and Kaelbling, 1991; Dubrawski and Reingnier, 1994). Generally, the optimal state space design should be supported by the optimal action space design, and vice versa. This resembles the well-known "chicken and egg problem" (Figure 1). Again, the "physical intervention" such as LBW seems much more difficult to be applied in robot learning when the state and action space are not specified *a priori*.

To cope with these two problems, in this paper

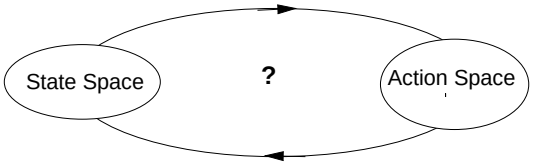


Figure 1 *The inter-dependence between state and action spaces*

we propose an efficient method of robot learning by which a set of pairs of a state and an action are constructed. Basic ideas of our method are as follows:

1. Since autonomous construction of state and action spaces is generally a very difficult problem, we construct a state space so that a group of situations in which an action command to achieve the goal is the same can be merged into one state even if these situations appear to be different from each other. An action is defined as a sequence of the same action command in such a state.

2. Following the LEM (Learning from Easy Missions) paradigm (Asada *et al.*, 1995), we first find a set of states (in terms of action) closest to the goal state, and then find a set of states closest to the set found previously.

3. In order to reduce an enormous number of trials to find such states, we place a robot so that it can observe objects which the state space consists of (in our case, a ball and a goal).

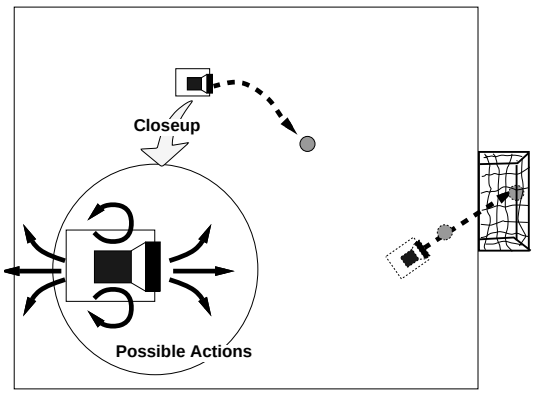
4. During the above process, the optimal action to achieve the goal is found in every state. This means that a robot can take an adequate action to achieve the goal from every state.

The remainder of this article is structured as follows: In the next section, we review our previous work, including explanations for the task and assumptions. Next, we show the method to automatically construct the state space, and finally, we show the experimental results and give a discussion with concluding remarks.

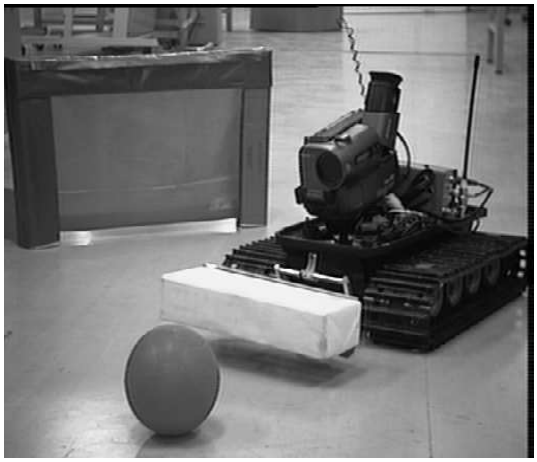
2 The Domain and Our Previous Work

2.1 Task and Assumptions

The task for a mobile robot is to shoot a ball into a goal as shown in Figure 2(a). The environment consists of a ball and a goal, and the mobile



(a) Task is to shoot a ball into the goal.



(b) A picture of a radio-controlled vehicle.

Figure 2 *Task and our real robot.*

robot has a single TV camera. The robot does not know the location and the size of the goal, the size and the weight of the ball, any camera parameters such as focal length and tilt angle, or kinematics/dynamics of itself. We applied Q-learning (Watkins, 1989), a most widely used reinforcement learning method, with the reward value to be 1 when the ball is kicked into the goal and 0 otherwise. Figure 2(b) shows a picture of the real robot with a TV camera (Sony handy-cam TR-3) used in the real experiments.

2.2 State and Action Spaces

(a) a state space S The image, supposed to capture the ball and/or the goal, is the only source of information the robot can obtain about the environment. The ball image is classified into 9 sub-states, combinations of three classifications of positions (left, center, or right) and three types of sizes (large (near), middle, or small (far)). The goal image has 27 sub-states, combinations of three properties each of which is classified into three categories (see Figure 3). Each sub-state corresponds to one posture of the robot towards the goal, that is, position and orientation of the robot in the field. In addition to these 243 (27×9) states, we add other states such as the cases in which only the ball or only the goal is captured in the image. In all, we have 319 states in the set $\boldsymbol{S}$.

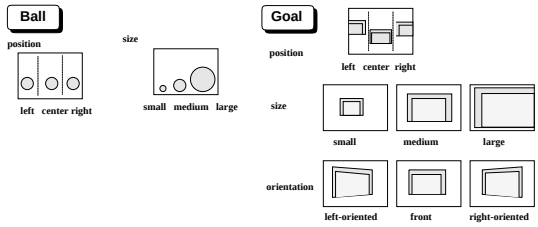


Figure 3 *The ball sub-states and the goal sub-states*

(b) an action space A The robot can select an action to be taken in the current state of the environment. The robot moves around using a P-WS (Power Wheeled Steering) system with two independent motors. Since we can send the motor control command to each of the two motors separately, we construct the action set in terms of two motor commands ω_l and ω_r, each of which has 3 sub-actions, forward, stop, and back. All together, we have 9 actions in the action set $\boldsymbol{A}$. An action is defined by a sequence of the same action command as explained later.

3 State and Action Space Construction

3.1 Problems with Our Previous Work

In our previous work, we divided a state space very coarsely in order to reduce the size of the state space on which the learning time depends (it is generally an exponential order in the size of the state space (Whitehead, 1991)). We have considered only the perceived data, that is, the visual information, and ignored the effects on the image caused by robot actions. Due to the peculiarity of the visual information, that is, the same motion in the 3-D space results in a large change in the image if it happens near the observer but a small change in the image if it happens far from the observer, one action does not always corresponds to one state transition. We called this **"state-action deviation problem."** Then, we reconstructed the action space by redefining an action

so that every action can cause one state transition. However, the state space construction was designed by the programmer. The agent should find a state space by itself through interactions with the environment. The following are the requirements for the problem:

1. The state and action spaces should reflect physical sensor(s) and actuator(s) of a robot. The deterministic state transition models (e.g. one action is forward, backward, left, or right, and the states are encoded by the locations of the agent) are useful only for simple toy problems in computer simulations.
2. Since everything changes asynchronously in real world (Mataric, 1994), the state and action spaces directly reflecting the physical sensors and actuators suffer from the state - action deviation problem. The state and action spaces should be restructured to cope with this problem.
3. The state space should be robust against the various disturbances such as sensor noise, delay, and uncertainty of action execution.

3.2 The Method

Basic ideas of our method are that input vectors are merged into one state unless the optimal action changes, that an action is defined as a sequence of one kind command actions (the length of the sequence has no meaning in our method, it depends on the state changes), and that such states are found in the order of closeness to the goal state (Figure 4). In order to efficiently realize the idea, a robot and a ball are initially placed so that the robot can see the ball and the goal regardless of the distance to the goal. In every trial, the same action command is repeatedly executed, and observed vectors during the robot motion are recorded in the state space if the robot reached the goal state.

Algorithm

1. Set the goal state as a target zone $\boldsymbol{Z}_T$.
2. Store an input state vector $\boldsymbol{x} \in \boldsymbol{R}^m$ (m: the size of the vector) with an index of the action $a \in \boldsymbol{A}$ the robot took when it could succeed in achieving $\boldsymbol{Z}_T$ from $\boldsymbol{x}$. Do not include the vectors that have been already categorized into the previously divided states.
3. Fit a multi-dimensional uniform distribution function (a concentration ellipsoid (Cramër, 1951)) to a cluster of stored vectors with the same action index $a \in \boldsymbol{A}$ obtained above, and construct a state s_a ($a \in \boldsymbol{A}$). The boundary surface of the ellipsoid is given by:

$$(\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) = m + 2, \qquad (1)$$

where μ and Σ denote the mean vector and the covariance matrix, respectively.

4. Update the target zone $\boldsymbol{Z}_T$ as a union of states $s_a (a \in \boldsymbol{A})$ obtained in the previous step. If a vector is categorized into plural states s_{a_j} ($j = 1, ..$) (clusters are overlapped), select one state so that the following distance normalized by its covariance can be the minimum:

$$\Delta_j = (\boldsymbol{x} - \boldsymbol{\mu}_j)^T \Sigma_j^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_j)$$

5. Stop if the state space is almost covered by the divided clusters. Else, go to 2.

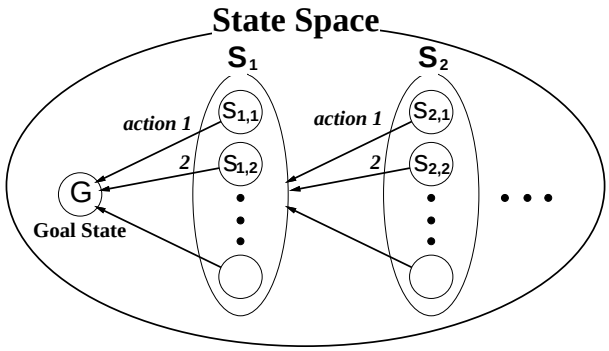


Figure 4 The goal-directed and action-based state space construction

We call a set of states $s_a (a \in \boldsymbol{A})$ the i-th closest to the goal state $\mathbf{S_i}$. By the above algorithm, we can obtain not only the state and action spaces but also the optimal path to the goal from everywhere.

4 Experimental Results

The experiment consists of three stages. First, we collect the data obtained by real robot motions. Next, we construct the state and action spaces based on the sampled data. Finally, we control the robot based on the acquired state and action map.

There are several sources of disturbances which make the method unstable. Two major sources are delay due to sensory information processing and uncertainty of action execution. The contents of the image processing are color filtering (a ball

and a goal are painted in red and blue, respectively), edge enhancement, localizing and counting edge points, and vector calculation (Asada *et al.*, 1994*a*). We have been using a pipeline image processor and it takes about 33ms to perform these processings, that is, a period of one action command. The latter is caused by the delay necessary to stabilize motor rotation after sending motor commands, and it is about 100ms. Therefore, the uncertainty of the action execution increases when motor commands often change.

The size of the observed image is 512 by 480 pixels, and the center of image is the origin of the image coordinate system (see Figure3). Each state vector $\boldsymbol{x}$ for a shooting task consists of:

- x_1: the size of the ball, the diameter that ranges from 0 to about 270 pixels,
- x_2: the position of the ball ranging from -270 to +270, considering the partial observation,
- x_3: the size of the goal, the height average of the left and right poles (or ends in image) that ranges from 0 to 480 pixels,
- x_4: the position of the goal, the average of the positions of the left and right poles (or ends in image) that ranges from -256 to +256, and
- x_5: the orientation of the goal, the ratio of the height difference between the left and right poles (or ends) to the size of the goal x_3. x_5 ranges from -1.00 to +1.00.

Figure 5 shows the process of state space division. The state space in terms of ball size, ball position, and goal size is indicated when the position and the orientation of the goal (x_4 and x_5) are both zeros (in front of the goal). In the first step, only one big ellipsoid ($\mathbf{S_1}$) is obtained that corresponds to the forward motion (Figure 5 (a)). In the second step, two ellipsoids ($\mathbf{S_2}$) corresponding to forward and backward motions are obtained (Figure 5 (b)). The state divided by programmer in the previous work (Asada *et al.*, 1994*a*) would be a rectangular parallelepiped parallel to the axes in Figure 5. The remainder of the state space in Figure 5 (b) corresponds to infeasible situations such as "the goal is near and the ball is far" although we did not recognize such a meaningless state in the previous work.

Figure 6 shows a real robot experiment using the state and action map obtained by the method. The robot succeeded in finding and shooting a ball into the goal.

Table 1 compares the method with our previous work (Asada *et al.*, 1994*a*). The search time in the previous work means the learning time in terms of the period of one action command (33ms) since the state space is given a priori. It takes about 500M (M=10^6) steps because the number of states is much larger. The proposed method performs better than the previous work. The size of the state space is about 1/8 of that of the previous work. The search time is about 1/12 of the previous work.

Table 1 *Comparison of the methods with our previous work*

	# of States	Search Time	Success Rate (%)
Previous work	243	500M*	77.4
Proposed method	33	41M	83.3

* indicates the Q-learning time.

Only the problem is that the size of one state is considerably larger, and therefore the possibility of the incorrect merging of input vectors into wrong states seems high. This might be partly a reason why the success rate is less than 90%.

5 Discussion and Concluding Remarks

We have proposed an efficient robot learning method by which the robot learned to achieve the goal by constructing the state and action spaces based on experiences. To speed up the learning (search), we assumed that the state space is smooth and uniform, by which we could place the robot at places where the possibility to achieve the goal is very high and could apply a multi-dimensional uniform distribution function (a concentration ellipsoid (Cramër, 1951)) to represent a region of one state.

There is no guarantee that states can be approximated by the concentration ellipsoids. We have tried another function, a Gaussian probability density one. The performance was not so different from the concentration ellipsoid, but the Gaussian function needs a threshold to define the boundary to what extent a state can be expanded. While, the concentration ellipsoid does not need such a threshold because we assume that the distribution of the input vector is uniform. The reason why the success rate is less than 90% is that some vectors do not hold this assumption.

Popular methods for clustering such as *kd-trees* (Samet, 1984), ID-3 (Quinlan, 1983), and *k nearest neighbor* do not seem suitable for the method because we intended to construct a state which

consists of various kinds of situations that might be quite different in their appearance. If we apply these methods, we would have a large number of states.

6 REFERENCES

Asada, M., S. Noda, S. Tawaratsumida and K. Hosoda (1994*a*). "purposive behavior acquisition on a real robot by vision-based reinforcement learning". In: *Proc. of MLC-COLT (Machine Learning Conference and Computer Learning Theory) Workshop on Robot Learning*. pp. 1–9.

Asada, M., S. Noda, S. Tawaratsumida and K. Hosoda (1994*b*). "Vision-based behavior acquisition for a shooting robot by using a reinforcement learning". In: *Proc. of IAPR / IEEE Workshop on Visual Behaviors-1994*. pp. 112–118.

Asada, M., S. Noda, S. Tawaratsumida and K. Hosoda (1995). Vision-based reinforcement learning for purposive behavior acquisition. In: *Proc. of IEEE Int. Conf. on Robotics and Automation*. pp. 146–153.

Chapman, D. and L. P. Kaelbling (1991). "Input generalization in delayed reinforcement learning: An alogorithm and performance comparisons". In: *Proc. of IJCAI-91*. pp. 726–731.

Connel, J. H. and Mahadevan, S., Eds.) (1993*a*). *Robot Learning*. Kluwer Academic Publishers.

Connel, J. H. and S. Mahadevan (1993*b*). "Rapid task learning for real robot". In: *Robot Learning* (J. H. Connel and S. Mahadevan, Eds.). Chap. 5. Kluwer Academic Publishers.

Cramër, H. (1951). *Mathematical Methods of Statistics*. Princeton University Press. Princeton, NJ.

Dubrawski, A. and P. Reingnier (1994). Learning to categorize perceptual space of a mobile robot using fuzzy-art neural network. In: *Proc. of IEEE/RSJ/GI International Conference on Intelligent Robots and Systems 1994 (IROS '94)*. pp. 1272–1277.

Kang, S. B. and K. Ikeuchi (1994). Determination of motion breakpoints in a task sequence from human hand motion. In: *Proc. of IEEE Int. Conf. on Robotics and Automation*. pp. 551–556.

Kuniyoshi, Y. and H. Inoue (1993). Qualitative recognition of ongoing human action sequence. In: *Proc. of IJCAI-93*. pp. 1600–1609.

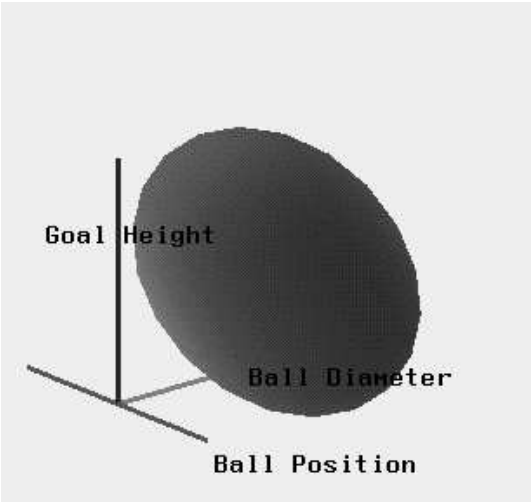
Mataric, M. (1994). "Reward functions for accelerated learning". In: *Proc. of Conf. on Machine Learning-1994*. pp. 181–189.

Quinlan, J. R. (1983). "Learning efficient classification procedures". In: *Machine Learning vol.1: An Artificial Intelligence Approach* (R. S. Michalski, J. G. Carbonell and T. M. Mitchell, Eds.). pp. –. Tioga Press.

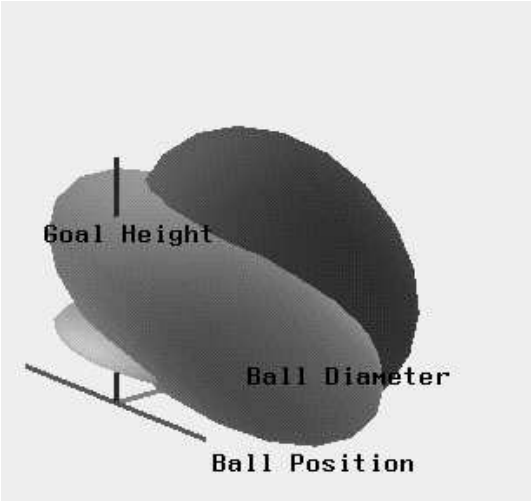
Samet, H. (1984). "The quadtree and related hierarchical data structures". *ACM Computing Surveys* **16-2**, 187–260.

Watkins, C. J. C. H. (1989). Learning from delayed rewards". PhD thesis. King's College, University of Cambridge.

Whitehead, S. D. (1991). "A complexity analysis of cooperative mechanisms in reinforcement learning". In: *Proc. AAAI-91*. pp. 607–613.



(a) First step



(b) Second step

Figure 5 *Process of state space division*

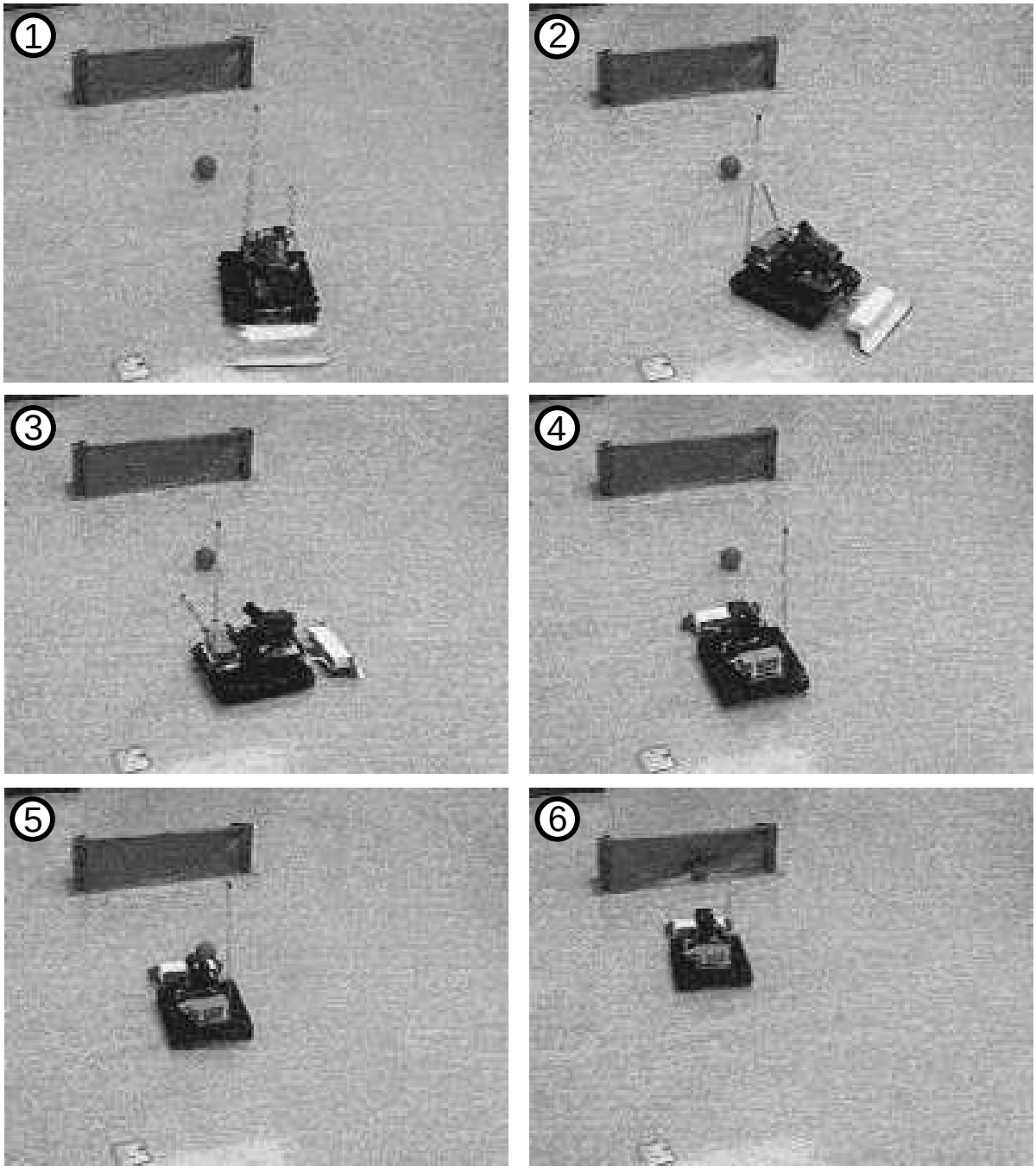


Figure 6 The robot succeeded in finding and shooting a ball into the goal