Environmental Change Adaptation for Mobile Robot Navigation

Takashi Minato and Minoru Asada
Dept. of Adaptive Machine Systems
Graduate School of Engineering
Osaka University
Suita, Osaka 565-0871, Japan
e-mail:minato@er.ams.eng.osaka-u.ac.jp,asada@ams.eng.osaka-u.ac.jp

Abstract

Most of existing robot learning methods have considered the environment where their robots work unchanged, therefore, the robots have to learn from scratch if they encounter new environments. This paper proposes a method which adapts robots to environmental changes by efficiently transferring a learned policy in the previous environments into a new one and effectively modifying it to cope with these changes. The resultant policy (a part of state transition map) does not seem optimal in each individual environment, but may absorb the differences between multiple environments. We apply the method to a mobile robot navigation problem of which task is to reach the target avoiding obstacles based on uninterpreted sonar and visual information. Experimental results show the validity of the method and discussion is given.

1 Introduction

In order for not only biological systems but also artificial systems, typical example of which is a robot system, to adapt themselves to changes in different environments, learning and development are essential processes. Conventional robot learning methods to cope with such changes employ multiple modules each of which corresponds to one or a class of environments so as to achieve the goal in each environment. However, such methods immediately suffer from two serious problems:

- It takes enormous learning time because the robot has to learn from scratch if it encounters new environments due to the lack of knowledge on them.
- It needs large capacity of memory and processing modules since the robot must have a variety of policies in terms of each individual environment to accomplish a given task in various kinds of environments.

To cope with the first problem, Thrun et al. [4, 3, 5] proposed "lifelong learning" which accelerates the learning by storing invariant knowledge for a class of environments in advance and reuses it as *a priori* knowledge in new environments categorized into the same class. Their methods need a clear definition for the class of environments to obtain the invariant knowledge, and therefore, no learning seems necessary in new environments but just implementation of the planned navigation. Tanaka and Yamamura [2] applied the similar idea to a simple navigation task on a grid world using a reinforcement learning method, in which the knowledge invariant in previous $(n-1)$ environments are extracted in advance and checked in the $n-th$ environment if it is effective or not. These methods might accelerate their learning, but the robot must have many policies corresponding to the individual environments, respectively, and therefore the second problem has not been addressed.

In this paper, we propose a method which adapt robots to environmental changes by transferring a Q-learned policy in the previous environments into a new one and modifying it to cope with these changes. The resultant policy (a part of state transition map) does not seem optimal in each individual environment, but may absorb the differences between multiple environments. We apply the method to a mobile robot navigation problem of which task is to achieve the target avoiding obstacles based on uninterpreted sonar and visual information. The difficulty of this task is twofold: 1) roles of multi-modal sensory information have not been assigned in advance, therefore the robot has to determine how to use the multi sensory information, that is, the sensory information is uninterpreted, and 2) both target reaching and obstacle avoidance tasks have to be achieved simultaneously. Nakamura et al. [1] considered the above two issues, but their robot has to learn from scratch if it encounters different environments.

Another important issue in applying Q-learning to real robot tasks is how to construct the state space on which the learning seriously depends. Too large state space causes enormous learning time while too compact one causes serious aliasings. Therefore, adequate state space is necessary to be found. In [4, 3, 5, 2, 1], the state space is given in *a priori* by human designer. Here, we attempt at finding a compact but seemingly adequate state space by searching for combinations of sensory features.

The reminders of this article is structured as follows. First, a brief summary of reinforcement learning, especially Q-learning is given. Next, task and assumption are given and the process of finding a state space is shown. Then, the method for environmental change adaptation is explained. Finally, experimental results are given to show the validity of the method and discussion is given.

2 Basics of Q-Learning

Q-learning is a form of model-free reinforcement learning based on stochastic dynamic programming. It provides robots with the capability of learning to act optimally in a Markovian environment. We assume that the robot can discriminate the set $\boldsymbol{S}$ of distinct world states, and can take the set $\boldsymbol{A}$ of actions on the world. A simple version of a Q-learning algorithm used here is shown as follows.

1. Initialize $Q(s,a)$ to 0 for all state s and action a.

2. Perceives current state s.

3. Choose an action a according to action value function.

4. Carry out action a in the environment. Let the next state be s' and immediate reward be r.

5. Update action value function from s, a, s', and r,

$$\begin{aligned} Q_{t+1}(s,a) &= (1-\alpha_t)Q_t(s,a) \\ &\quad +\alpha_t(r+\gamma \max_{a'\in\boldsymbol{A}} Q_t(s',a')) \end{aligned}$$

   where α_t is a learning rate and γ is a fixed discounting factor between 0 and 1.

6. Return to 2.

3 Task, Robot, and Assumptions

3.1 Our Robot

Our robot has a Power Wheeled Steering (PWS) system driven by two motors into each of which we can send a motor command, independently. In our experiment, we quantized each motor command into three levels which correspond to forward, stop, and backward, respectively. Totally, the robot has 9 actions.

The robot is equipped with a ring of 12 ultrasonic ranging sensors (ranging from 0.0 to 300 *cm*) which has a field view of roughly 30° (see Figure 1(a)). The robot is also equipped with a color CCD camera. Image processing procedure provides the position and the size of the target area in the image as visual information(see Figure 1(b)). However, it cannot detect obstacles, because it is not given how obstacles can be seen.

3.2 Task

The task of the robot is to reach the target while avoiding obstacles as shown in Figure 2. Nakamura et al. [1] has demonstrated a limited ability coping with the above problems in a single isolated environment, but they suffered from the curse of dimension problem, a huge state space. In addition to this, the environment may change in a few ways here:

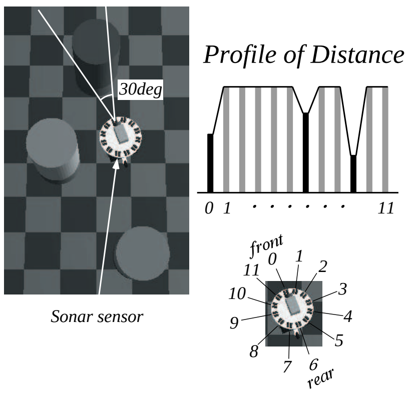
1. configurations of the target and obstacles may change, and

2. the number of obstacles also changes.

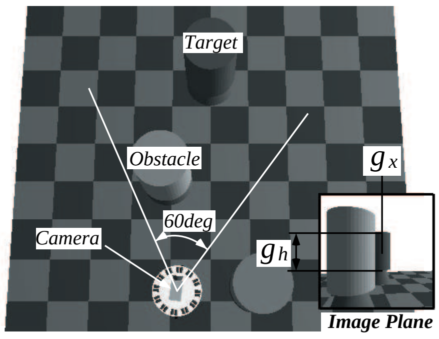
Therefore, a learned policy obtained in a single environment may not be applicable to different environments, and usually it takes enormous learning time if the robot learns from scratch.

3.3 State Vector Selection

As mentioned above, the state space construction problem is one of the most serious issues in reinforcement learning even in an isolated single environment. Since our robot has a considerably large sensor space, we have to build a reduced-size state space from the original sensor space. As primitive features, we have selected the center position g_x and the height g_h of the target image from vision as image features, and followings (Table 1) from sonar profile as sonar features (see Figure 3).



(a) sonar



(b) vision

Figure 1: Sensory information

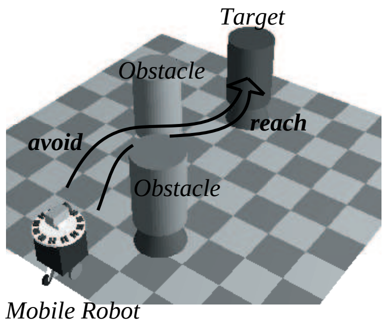


Figure 2: Task and environment

Table 1: sonar features

range feature	
d_{min}	minimum range value
d_{max}	maximum range value observed
d_{mean}	mean range value observed
d_{diff}	$d_{max} - d_{min}$
direction feature	
θ_{min}	direction of d_{min}
θ_{max}	direction of d_{max}
θ_{mean}	mean between θ_{min} and θ_{max}
θ_{diff}	width between θ_{min} and θ_{max}

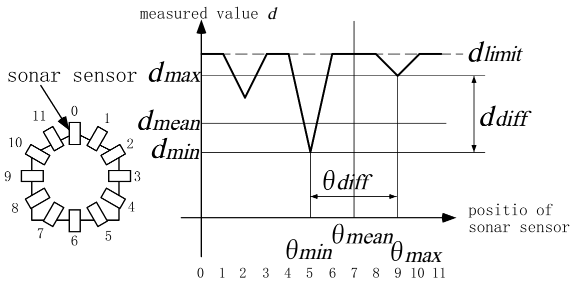
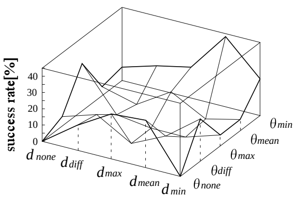


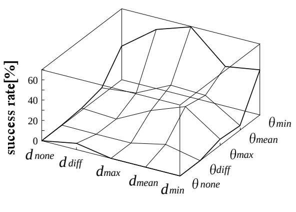
Figure 3: Primitive features from sonar profile

Since these features still consists a large feature space, we have constructed several spaces as the combinations of one or two image features and one or two sonar features, and applied Q-learning to the robots having each state space in a single environment (see Figure 2). Here, we have appropriately quantized each state space under a constraint of the maximum number of substates to realize physical memory capacity constraint. Figure 4 shows the success rates by applying each obtained policy to the environment, in which d_{none} and θ_{none} means no features. Hereafter, we define the success rate as (number of successful target reaching)/(trials). In Figure 4, feature vectors including θ_{min} show high success rates regardless of the range features, which means that the robot pays its much more attention to the direction of the minimum range value than to the value itself. According to the result of these analysis, we have selected the following state vector $\boldsymbol{x}$ for the task.

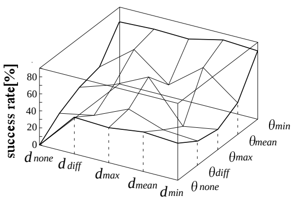
$$\boldsymbol{x} = \begin{pmatrix} g_x & g_h & \theta_{min} & d_{min} \end{pmatrix}^T \qquad (1)$$



(a) g_x, θ_*, d_*



(b) g_h, θ_*, d_*



(c) g_x, g_h, θ_*, d_*

Figure 4: Performance of each state space

4 Environmental change adaptation

The basic idea to accelerate the learning and to save memory is to adapt a robot to a new environment by transferring the Q-learned policy in the previous environments and modifying it to cope with environmental changes. The robot realizes the environmental changes by a drop of the success rate. Unless the success rate drops, the robot does not realize the environmental change even if the environment actually changes.

If the robot realizes the environmental change, it re-learns the policy of the states where it fails to achieve the goal. Figure 5 shows an example of two environments which have different state transitions. Using the policy in the environment 2 which has been obtained in the environment 1 (optimal path), the robot fails the task (e.g. makes a collision with the obstacles) at the state A. Then the robot re-learns the policy only around state A so that it can achieve the goal and as a result, it finds the policy $A \rightarrow a_2$, which is not optimal but feasible in the both environment. It is expected that the robot adapts the environmental changes by modifying the policy around states where it fails. However, there is in practice a trade-off to what extent the current policy can be modified. Then, we introduce the adaptability rate β by which the robot controls to what extent the previous policy can be destroyed.

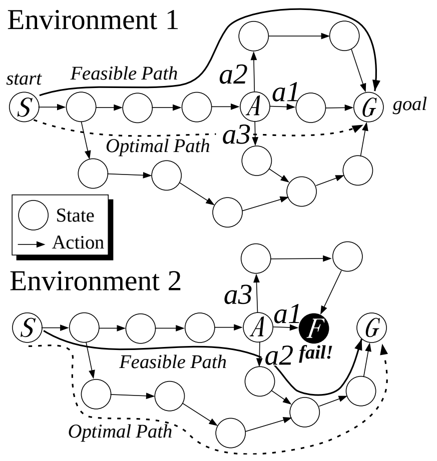


Figure 5: Not an optimal path but a feasible one can be found.

The algorithm is as follows:

1. Quantize the state space as S.
2. Apply Q-learning to the initial environment, and obtain the policy $P : \boldsymbol{S} \rightarrow \boldsymbol{A}(\boldsymbol{A}$:action set) until the success rate exceeds the pre-specified success rate RS.
3. Apply P to any environment unless RS decreases.
4. If RS decreases, then find states $\boldsymbol{S}_r \subset \boldsymbol{S}$ where P fails to achieve the goal, and modify P for such states by applying Q-learning as follows until RS recovers to pre-specified adaptability rate β.
5. Apply Q-learning to $\boldsymbol{S}_r \subset \boldsymbol{S}$. Action selection during the learning is as follows:

   if $s \in \boldsymbol{S}_r \cup \boldsymbol{S}_u$ follow the normal action selection in Q-learning, where $\boldsymbol{S}_u \subset \boldsymbol{S}$ denotes inexperienced states.

   otherwise follow P
6. Go to 3 with the obtained policy P.

The adaptability rate β determines the extent to which re-learning occurs, that is:

$$RS_d = RS_c + \beta(RS_p - RS_c), \tag{2}$$

where RS_p, RS_c and RS_d denote the success rates in the previous environment, in the current one, and the desired one, respectively.

Based on the selected state vector, we apply the algorithm to the given task with the following specifications:

- $\alpha_t = 0.25$, $\gamma = 0.9$, $\beta = 0.5$. The number of all states is 3060.
- If the robot reaches the target, the positive reward 1 is given. Otherwise 0.
- One trial terminates if the robot reaches the target, makes a collision with any obstacles, or the given time interval expires.
- The states $\boldsymbol{S}_r$ are around states where the robot makes a collision with any obstacles.

5 Experimental Results

We tested five different environments called $E1, \ldots, E5$ shown in Figure 6 where a solid black circle, gray circles, and empty pentagons indicate the target, obstacles, and robot trajectories, respectively. The sequence of the environments the robot encounters is $E1, E2, \ldots, E5$, where $E1$ is the initial environment. Table 5 shows the success rates in terms of Pi which is obtained policy in Ei by the method. The second column shows the number of states found to be re-learned when Pi applied to $E(i+1)$. Here, we set $\beta = 0.5$ but iteration of modification is performed in terms of discreet state space. Therefore, some of policies were improved much more than β.

When the robot encounters an inexperienced environment, the success rate starts to decrease and robot realizes that the environment has been changed. After policy modification, the robot obtained the policy which may cope with the previous environments, too. The successful trajectories in Figure 6 are the result of application of the final policy $P5$ to each environment. The final policy does not seem optimal in each individual environment, but can absorb the differences between multiple environments. For example, in Figure 6(e), the most left trajectory includes curves turning to the right although its optimal policy was just a straight path to the target.

Next we tested how transferring a Q-learned policy accelerated the learning. Two methods in $E2$ are compared.

- case 1:the normal learning method from scratch in $E2$.
- case 2:our method with $P1$.

5400 (case 1) trials and 1900 (case 2) trials were needed for the two methods to converge. By transferring the policy the convergence was accelerated up to 35%.

Finally, we apply the learned policy to a real situation as a new environment. Figure 7 shows the result of real robot experiment. The system used is almost same as in [1]. We used a mobile robot platform "Yamabico" which is modeled in section 3.1. There are a target (trash can) and two obstacles (trash cans) in the environment, and Yamabico uses the policy $P5$. Then, Yamabico succeeded in reaching the target avoiding two obstacles.

Table 2: Success rates of each policy [%]

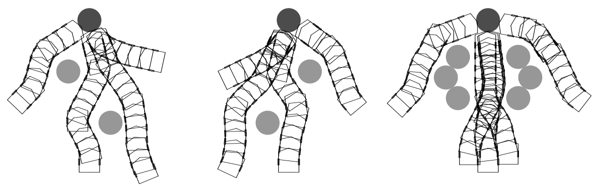
policy	# of $s \in S_r$	$E1$	$E2$	$E3$	$E4$	$E5$
$P1$	-	90.9	61.0	(48.2)	(47.3)	(61.5)
$P2$	61	92.4	90.8	45.2	(50.2)	(69.5)
$P3$	75	88.2	88.8	93.3	82.7	(68.9)
$P4$	44	82.2	87.0	93.2	87.5	67.4
$P5$	69	89.4	89.9	89.8	76.4	84.7

6 Discussion and Future work

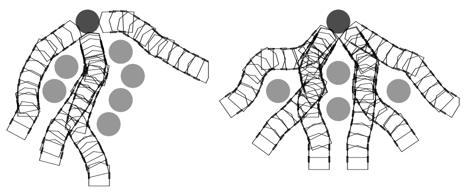
We have proposed a method which adapts robots to environmental changes by transferring a Q-learned policy in the previous environments into a new one and modifying it to cope with them. The resultant policy could absorb the differences between multiple environments. The resultant policy includes:

1. invariant state-action pairs common to all environments,
2. state-action pairs common to some environment but may not be applicable in other environments,
3. state-action pairs effective in the last environments,

but excludes state-action pairs that conflicts from each other (class 4) in multiple environments. Our method attempts at minimizing the number of class 4 by adopting not optimal but feasible solutions. However, to make the obtained policy reasonable, we need to be careful about the sequence of environments which the robot encounters. Next, a method that defines the task class in such a way that the robot can gradually skill up the learned policy should be developed. In addition to that, there is another problem of state space construction: state vector selection is currently off-line process, but ideally the issue should be included in the on-line learning process. These are under the investigation.



(a) Env.$E1$ (b) Env.$E2$ (c) Env.$E3$



(d) Env.$E4$ (e) Env.$E5$

Figure 6: Environments and successful trajectories

References

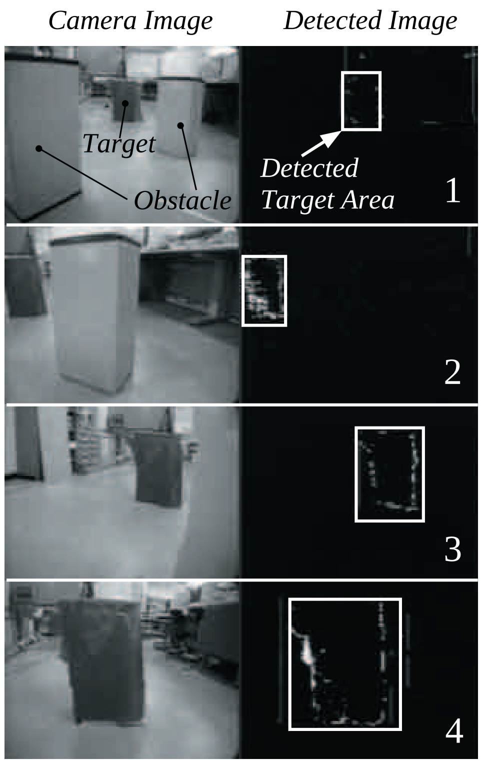
[1] T. Nakamura, J. Morimoto, and M. Asada. Direct coupling of multisensor information and actions for mobile robot behavior acquisition. In *Proc. of 1996 IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration*, pages 139–144, 1996.

[2] F. Tanaka and M. Yamamura. An approach to lifelong reinforcement learning through multiple environments. In *6th European Workshop on Learning Robots*, pages 93–99, 1997.

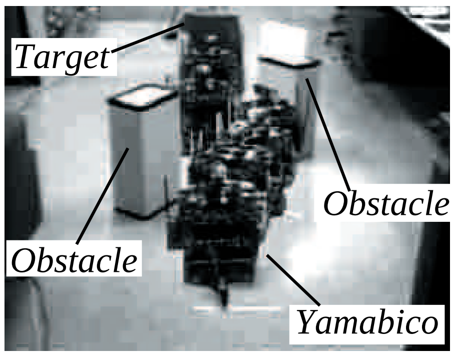
[3] S. Thrun. A lifelong learning perspective for mobile robot navigation. In *Proc. of the IEEE/RSJ/GI Conference on Intelligent Robots and Systems*, pages 23–30, 1994.

[4] S. Thrun and T. Mitchell. Lifelong robot learning. Technical Report IAI-TR-93-7, University of Bonn, Dept. of CS III, 1993.

[5] S. Thrun and J. O'Sullivan. Discovering structure in multiple learning tasks: The tc algorithm. In *Proc. of the thirteenth International Conference on Machine Learning*, 1996.



(a) Real input images



(b) Success trajectory

Figure 7: Experimental result