Vision-based Reinforcement Learning for Robot Navigation

Weiyu Zhu, Stephen Levinson

University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering, Beckman Institute
405 N. Mathews, Urbana, IL61801, USA
{weiyuzhu, sel}@ifp.uiuc.edu

Abstract

In this paper, we present a novel vision-based learning approach for autonomous robot navigation. A hybrid state-mapping model, which combines the merits of both static and dynamic state assigning strategies, is proposed to solve the problem of state organization in navigation-learning tasks. Specifically, the continuous feature space, which could be very large in general, is firstly mapped to a small-sized conceptual state space for learning in static. Then, ambiguities among the aliasing states, i.e., the same conceptual state is accidentally mapped to several physical states that require different action policies in reality, are efficiently eliminated in learning with a recursive state-splitting process. The proposed method has been applied to simulate the navigation learning by a simulated robot with very encouraging results.

1 Introduction

Over the years, reinforcement learning methodology has been extensively studied by researchers for autonomous robot skill acquisitions, such as the goal-oriented navigation [1][2], the hand-eye corporation [3] and the playing for a soccer game [4]. Figure 1 depicts a typical reinforcement learning system. The *learner* receives descriptions of the *environment*, which are called *states*, from peripheral sensors and chooses *actions* to perform. The effect of an action to the environment is evaluated by a *teacher*, which could be a person or some special sensors, and fed back to the learner in the form of positive or negative *rewards*. The mission of the learner is to find the action rules to optimally achieve a certain goal through its interaction with the environment.

A simple and straightforward implementation idea of reinforcement learning is assigning each action performed an instant reward and adjusting the corresponding action policies right away. Salatian *et al* applied such a scheme in their biped robot-walking project [5] and Gullapalli *et al* realized the control of "peg-to-hole" inserting by a robot arm in [3]. Instant rewarding methods are simple and easy to implement. Nevertheless, since the learning highly relies on the direct instructions from the teacher, it performs poor when the feedback is wrong or the evaluation is unavailable sometimes, which is quite common in real-time robot controls, where the effect of each action can hardly be analyzed individually.

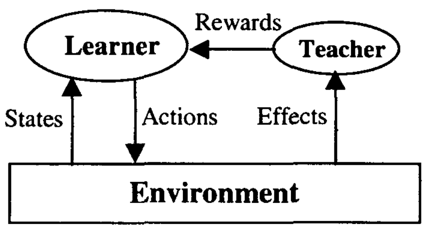


Figure 1. A typical structure of reinforcement learning

To overcome the drawbacks in the instant-rewarding method, the idea of *Delayed rewarding* is introduced [6]. Instead of evaluating actions separately, delayed-rewarding schemes analyze the effect of action sequences as a whole and the action policies are adjusted for maximizing the expected future discounted rewards. Provided that the system model, which includes the state transition matrix and rewarding policy, is known, delayed-rewarding problems can be easily solved with traditional Dynamic Programming (DP) approaches. However, obtaining a precise model for each learning task is actually not easy, especially when the number of states is large.

An elegant solution to model-free learning problems is offered by the so-called Temporal Difference (TD) method [6], such as Sarsa(λ) and $Q(\lambda)$. Action policies are incrementally updated under the TD learning paradigm by consecutively exploring the outside world. In [7], Randlov *et al* presented a Sarsa(λ)-based learning system for autonomous bicycle-riding; and in [1] and [2], Thrun and Millan *et al* , respectively, applied the Q-learning algorithm in navigation learning by an autonomous robot. Compared with the model-based approaches, TD methods learn more robustly, especially for real-time robot control tasks, since it does not require an accurate model of the world.

An essential implementation issue in reinforcement learning is how to map the feature space, which is called the physical state space, in reality into the conceptual state space for learning. Static mapping is a straightforward choice, where the number of conceptual states is constant and their mapping to the physical state space is fixed. The static mapping scheme is easy to implement but it has trouble to handle the learning with large feature spaces, such as the case in Thrun's work [1], where a 46-dimensional real-valued feature space was defined for learning. In order to unambiguously cover the entire feature space, the conceptual state space must be defined extremely large, with which prohibitive costs of storage and learning time are inevitably introduced. Otherwise, the optimal action policy cannot be learnt due to the aliasing existed, i.e., the same conceptual state is mapped to several physical states that require different action policies in reality.

Unlike the static methods, dynamic state mapping schemes, as used by Kros *et al* and Mullin *et al* in the robot collision-free navigation learning in [8] and [9], respectively, map conceptual states dynamically to where they are most needed. Since each conceptual state is able to "serve" for a varying size of physical states in dynamic schemes, the number of conceptual states required is dramatically decreased. However, at the expense of the state-size reducing, an additional cost for state adjustment in each learning step is imposed, which is nontrivial too in practice.

In this paper, we proposed a hybrid state-mapping method, which combines the static and dynamic strategies together, to realize the navigation learning by an autonomous robot. The robot senses its environment with a pair of stereo cameras mounted on its head and learns to reach its target in a reinforcement manner. Negative rewards are received whenever the robot loses the sight of the target in either view; and positive rewards are assigned when the robot reaches his goal, which could be sensed by the sensors on its gripper. Watkins' $Q(\lambda)$ algorithm [10] is used for learning with a hybrid state mapping strategy exploited. Initially, a small set of conceptual states is defined in static to represent the continuous feature space. Then, whenever confusion occurs in learning, i.e., the same state-action pair happens to lead to both positive and negative rewards, that state would split automatically to eliminate the ambiguities. The proposed hybrid state-organizing method acquires the merits of both static and dynamic mapping strategies, with which fast learning is achieved, since the number of conceptual states to be learnt is small, while ambiguities among the aliasing states are dynamically eliminated in learning.

The remaining sections in this paper are organized as follows: In section 2, a detailed description to our learning approach is given, including some implementation issues of the $Q(\lambda)$ algorithm. Section 3 presents the experiments carried out in our simulated learning system. Finally, a brief conclusion, including the future work, is given in section 4.

2 Proposed Approach

In this section, we first give a brief introduction to Watkins' $Q(\lambda)$ learning algorithm. The issues of feature extraction and the definition of learning states are discussed next, followed by the description of the proposed hybrid state-mapping strategy.

2.1 $Q(\lambda)$ algorithm

$Q(\lambda)$ is an on-line multi-step learning algorithm developed by Watkins in 1989 [10]. Its simplest form, $Q(0)$, the one-step-learning, is defined as

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)\right]$$

where $Q(s_t,a_t)$ is the expected discounted return by taking action a_t at state s_t. The action with the largest $Q(s_t, a_t)$ is said the optimal action, denoted as a^*, at state s_t. r_{t+1} is the instant reward received after action a_t is performed. α is a constant step-size parameter and γ is called the *discount rate*, which indicates the influence of future rewards on the current state.

```
Initialize $Q(s, a)$ and set eligibility trace $e(s, a)=0$
Repeat (for each learning episode):
  Set initial state $s$ and pick initial action $a$
  Repeat (for each step of episode):
    Take action $a$, observe reward $r$ and the next state $s'$
    Choose $a'$ for $s'$ with a certain policy
    $a^* \leftarrow \arg\max_a Q(s', a)$
    $\delta \leftarrow r + \gamma Q(s', a^*) - Q(s, a)$
    $e(s, a) \leftarrow e(s, a) + 1$
    For all $s$, $a$:
      $Q(s, a) \leftarrow Q(s, a) + \alpha\delta e(s, a)$
      If $a' = a^*$, then $e(s, a) \leftarrow \gamma\lambda\, e(s, a)$
              else $e(s, a) \leftarrow 0$
    $s \leftarrow s'$; $a \leftarrow a'$
  until $s$ is the terminal
```

Figure 2. Outline of $Q(\lambda)$ learning algorithm

$Q(0)$ algorithm learns quite slow because only one time-step is traced for each action. To boost the convergence of the learning, a multi-step tracing mechanism, which is called the *eligibility trace,* is introduced, with which the Q values of a sequence of actions can be updated simultaneously according to the respective lengths of the eligibility traces. An outline of the multi-step $Q(\lambda)$ learning algorithm, which is based on the tableau version in [6], is shown in Figure 2.

2.2 Definition of Learning States

In our navigation system, the robot senses its environment solely with on a pair of stereo cameras. A build-in object detection and tracking algorithm is used to detect the object of interest and keep track of it thereafter. The floor for navigation is assumed plain and parallel to the cameras so that the only spatial freedoms related to the navigation are the distance and horizontal position of the target to the robot. Given the distances can be uniquely determined by the disparities of the stereo images, a straightforward definition of states can be stated as

$$\text{State} = \{h, d\} \qquad (1)$$

where h is the horizontal position of the target in the view and d is the corresponding disparity.

Determining a quantitative representation for h in (1) is trivial. Heuristically we defined it as the horizontal coordinate of the left-most point of the target in the left image. Computing disparities, however, is a tricky issue in practice because the traditional stereo matching technique is very likely to fail when the robot gets too close to the target, as illustrated in Figure 3, due to the lack of feature points for stereo matching.

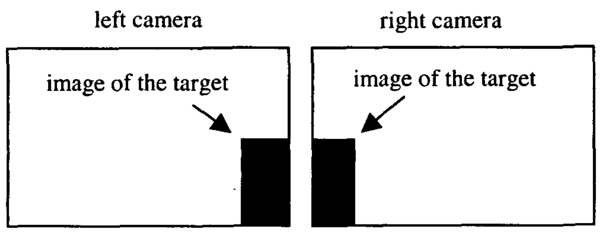


Figure 3. Difficulties in stereo matching when the target is too close to the robot

To cope with the difficulties in stereo matching, instead of using real disparities of image pairs, we introduced a new concept called the *pseudo-disparity*:

$$pseudo-disparity = x_{left} - x_{right} \qquad (2)$$

where x_{left} and x_{right} are the horizontal coordinates of the left/right-most points of the target in the left and right images, respectively. The computation for *pseudo-disparity* only requires the horizontal locations of both target images so that it is easy and fast to implement. However, before admitting *pseudo-disparity* for state definition, we need to prove two issues:

1. Is *pseudo-disparity* able to uniquely represent distance?
2. Is the action policy learnt with a certain object still applicable to other objects with different widths? For the case with real disparities, the answer is *yes* because the width of objects plays no roles in the state definition at all. However, how about the case with pseudo disparities?

Proof:

Given an object with width=W, denote its imaging width as v and the real and pseudo disparity between image pairs as d and d', respectively, we have

$$d = \frac{A}{D}, \quad v = \frac{WB}{D} \qquad (3)$$

where A and B are hardware-related constants and D is the distance from the object to the robot. Combining (3) with the definition of pseudo disparity in (2) yields

$$d' = d - v = \frac{A - WB}{D} \qquad (4)$$

That is, given a certain object, pseudo disparities do represent distances uniquely.

End of the proof for the first statement.

For proving the second statement, we first introduce a concept called the *physically equivalent states.*

Definition:

Two conceptual states, defined with different objects, are said *physically equivalent* if and only if they represent the same spatial relationship between the object and the robot. Referring to the state definition described above, the concept of "same spatial relationship" could be formulated as the same distance plus the same horizontal position, which is evaluated at the left-most side of the object, from the object to the robot.

Then, given a conceptual state S_2 for object 2 and a learnt system with object 1, the proving of the second statement is equivalent to checking whether the physically equivalent state of S_2 can be found in the learnt conceptual state space with object 1. If the answer is *yes*, then the optimal action for the state S_2 could be simply determined as the optimal action of the physically equivalent state of S_2 in the learnt system. That is, the action policy learnt with one object is still applicable to others with different widths.

Since the horizontal position of the target in the view, which is defined as the value of x_{left} in (2), is invariant with the changes of object widths, what we need to show is how to find the physically equivalent d' for different objects.

Denote the widths of two objects as W_1 and W_2 and assume the action policy for object 1 has already been learnt. Given an observed conceptual state $S_2 = (x_{left}, d_2')$, its physically equivalent pseudo disparity d_1' in the leant system can be computed as

$$d_1' = \frac{(A - W_1 B)d_2'}{A - W_2 B} \qquad (5)$$

where the width of any object is obtained by

$$W_i = \frac{A(d_i - d_i')}{Bd_i} \qquad (6)$$

where d_i and d_i' could be an arbitrary pair of real and pseudo disparities of object i at any distance.

End of the proof for the second statement.

Replacing the notation h and d in (1) with x_{left} and d', ultimately we have the state definition of

$$\text{State} = \{x_{left}, d'\} \qquad (7)$$

2.3 Hybrid State Structure

According to the state definition in last subsection, each conceptual state consists of two integer values x_{left} and d'. Assuming the width of images is 320 pixels and the pseudo disparities may go from 0 to 319 (small negative values are actually possible), the largest state space would consist of over 90,000 states. Learning with such a large state space is extremely time-consuming and not necessary at all because many neighboring states may share the same action policy in reality so that, intuitively, we can just treat them as a whole in learning. The remaining problem is we don't know which set of states would share the same action policies in advance.

The hybrid state-mapping model proposed is aimed at solving the above dilemma. The basic idea is to work with a small set of conceptual states defined in static first, in which aliasing might exist. Then, eliminate the aliasing in learning by automatic state splitting.

States Defined in Static. The initial small set of conceptual states is defined in static: x_{left} is uniformly quantized with a certain step and d' is quantized in the logarithm manner. Figure 6 shows an example of state mapping, where the initial states are delineated with solid lines. 70 conceptual states are defined in the example, where x_{left} is uniformly cut into 10 segments and d' is mapped to 7 regions.

Action Policy. Actions in learning are primarily chosen according to the ε-greedy scheme, where the current optimal action would be taken with the probability of ε and others share the remaining opportunities of 1-ε. In addition to the ε-greedy scheme, an extra restriction on actions with negative Q values is imposed. Negative Q values indicate the potential failure of the corresponding action-state pair. Therefore, the robot should avoid trying it unless no meaningful choice is left for the current state, when that state is said *ambiguous*. The robot would pick actions randomly in ambiguous states and perform state splitting thereafter.

State Splitting. For each pair of conceptual state s and action a, two lists, denoted as "fail" and "succeed", are maintained to record the physical states that have ever been experienced. A physical state is inserted into the "fail" list if an instant negative reward is received by taking the action a at that state, and to the "succeed" list otherwise. For simplicity and notational convenience, the physical state space is represented as the largest possibly defined conceptual state space. For example, the physical state space for the system described at the beginning of this subsection would consist of 320×320 quantized states. Given a fixed conceptual state, an action is said "ambiguous" if its both physical state lists are non-empty. A conceptual state is said "ambiguous" if all its Q values, each of which associates to an action, are negative. Whenever a state becomes "ambiguous" or the robot is stuck into some action loops, e.g., keeps switching between turning left and turning right, a state splitting procedure, as illustrated in Figure 4, is applied.

```
Function State_splitting ($s$)

    Choose a direction, $x_{left}$ or $d'$, to split; the one with the
    broader spanning range is preferred

    Compute new boundaries for new states
    Relocate the records of physical states to new states
    For each new state $s_{new}$
        if ambiguous action exists, call State_splitting ($s_{new}$)
```

Figure 4. Outline of the state splitting process

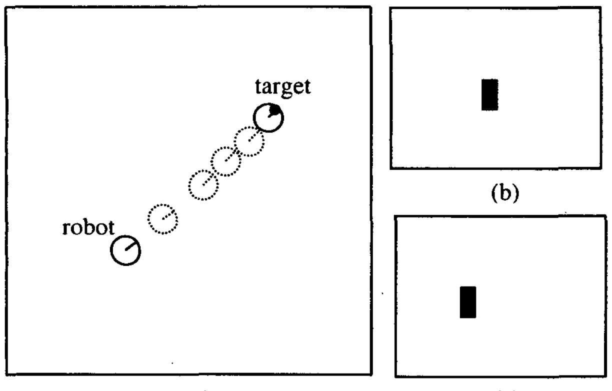
The state splitting process eliminates any ambiguous actions in new states by verifying the experiences of "fail" or "succeed" of the associated records of physical states previously experienced. However, not all states with ambiguous actions should split, as far as the robot still has meaningful action to choose. With a proper choosing of the initial mapping resolution, the number of conceptual states defined in static could be small and only a few may split in learning.

3 Experiments

Experiments were carried out in our simulation environment, where the coordinates of object images were computed according to the perspective camera model:

$$x = f\frac{X}{Z} \qquad y = f\frac{Y}{Z} \tag{8}$$

where f is the focal length, X/Y and x/y are the spatial coordinates and imaging coordinates on the horizontal/vertical directions of the object. The simulated robot worked in an environment of 4×4 m^2 and had 5 actions to choose: turning left/right for 3° and going forward for 6/12/24 inches. The target object was defined as a rectangular shape of 6×15cm in width and height. We assume that the stereo cameras were mounted on the front-top of the robot with a fixed baseline distance equal to 8cm. The resolution of both simulated images was 320×240 pixels. A typical scenario in the simulation is shown in Figure 5, where (b) and (c) are the simulated left and right images obtained by the robot in (a).



(a) (c)

Figure 5. Simulated learning system. (a) Robot navigation environment with the target object represented by a black dot. (b)-(c) Simulated left and right images perceived by the robot.

Each learning episode started from a random position and ended when the robot "catches" the object, as illustrated in Figure 5 (a), when the robot received a reward of 1000. A penalty of −1000 was assigned when the target disappeared from either image, when the robot would "redo" the action just performed and continue thereafter. Ambiguous states were detected automatically in learning and the recursive state-splitting procedure was applied to remove the aliasing whenever the robot ran out of meaningful actions to choose or was stuck into some action loops, which was detected by inspecting the action history that was recorded in learning.

Figure 6 shows the ultimate state distribution map after 400 learning episodes, where 70 conceptual states were initially defined and the learning parameter α, γ, ε and λ, as discussed in section 3, were heuristically set as 0.6, 0.9, 0.8 and 0.8, respectively. The learning converged after about 400 learning episodes and only 15 new states were generated in state splitting. It means that most initially defined states could be learnt properly without introducing ambiguities.

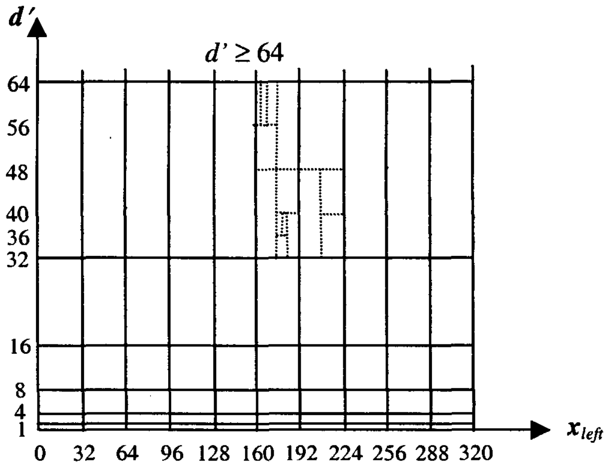


Figure 6. State map after 400 learning episodes. 10×7 states are initially defined in static. States delineated with dash lines are generated by splitting.

Figure 7 depicts the learning performance with varying size of the initial state spaces, where d' was still quantized in logarithm and x_{left} was uniformly quantized in three steps: 1, 32 and 106. The parameter set was set the same as that in the last experiment. The system was firstly learnt with a varying size of learning episodes. Then, the learnt systems were tested and the averaged number of required steps in testing was computed over 100 independent tests. The parameter ε was set to be 1 in the testing stages so that no exploration was performed any more and the system completely performed under the optimal action policies learnt.

The depicted curves in Figure 7 shows that the learning with 70 initially defined states performed the best, where the "robot" learned to behavior properly after only about 10 episodes. Large state space, which corresponds to curve 3, significantly limits the learning speed while sparse state space, which corresponds to curve 1, performs not well either because the same conceptual state is very likely to be visited consecutively in learning. According to the value updating mechanism in the $Q(\lambda)$ algorithm, consecutive visiting of the same state may lead to the oscillating updating of the action values of that state. Consequently, the converged optimal action policy cannot be learnt due to the endless updating of the Q values in learning. Therefore, as a basic implementation requirement, the initial state space defined in static should be dense enough to avoid the frequent "self-state-transitions" in learning.

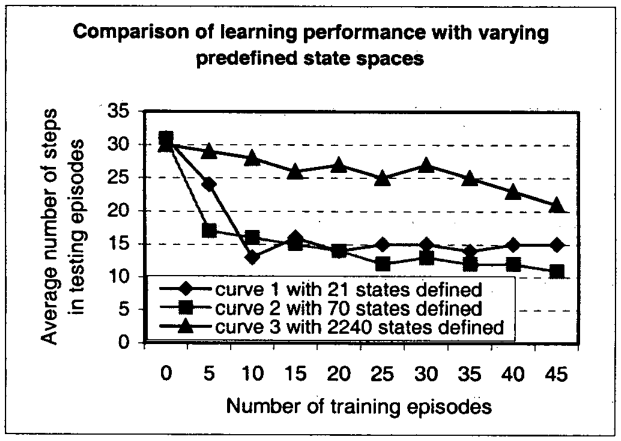


Figure 7 Comparison of learning performance with varying size of initial state spaces.

4 Conclusion

This paper presents a novel hybrid state mapping method which aims at solving the problem of conceptual state definition in reinforcement learning. The difficulty of learning with large feature spaces is relieved by mapping the physical state space into a small-sized conceptual state space in static. Then, the ambiguities among aliasing states are detected and eliminated via a recursive state splitting process in learning. We have used the proposed method to learn action policies by a simulated robot in our simulated navigation environment with very encouraging results. The implementation on a real robot is undertaking.

Acknowledgement

We would like to acknowledge the funding from the National Science Foundation Grant with award number IIS-00-85980.

References

[1] S. Thrun, "An Approach to learning Mobile Robot Navigation", *Robotics and Autonomous Systems*, 1995, pp.301-319

[2] J. Millan, R. del, "Reinforcement learning of goal-directed obstacle-avoiding reaction strategies in an autonomous mobile robot", *Robotics & Autonomous Systems*, 1995, pp.275-99.

[3] V. Gullapalli, J. Franklin, H. Benbrahim, "Acquiring robot skills via reinforcement learning", *IEEE Control Systems Magazine*, 1994, pp.13-24.

[4] H. Asada, S. Noda, S. Tawaratsumida, K. Hosoda, "Purposive behavior acquisition for a real robot by vision-based reinforcement learning", *Machine Learning*, 1996, pp.279-303.

[5] Salatian, Y. Keon, Y. Zheng, "Reinforcement learning for a biped robot to climb sloping surfaces", *Journal of Robotic Systems*, 1997, pp.283-96.

[6] R. Sutton, A. Barto, *Reinforcement learning*, MIT Press, 1998

[7] J. Randlov, P. Alstrom, "Learning to Drive a Bicycle Using Reinforcement learning and Shaping", *International Conference on Machine Learning*, 1998, pp.463-471

[8] Krose BJA, van Dam JWM, "Learning to avoid collisions: a reinforcement learning paradigm for mobile robot navigation", *Artificial Intelligence in Real-Time Control 1992. Selected Papers from the IFAC/IFIP/IMACS Symposium. Pergamon.* 1993, pp.317-21.

[9] J. Millan, R. del, "Reinforcement learning of goal-directed obstacle-avoiding reaction strategies in an autonomous mobile robot", *Robotics & Autonomous Systems*, 1995, pp.275-99.

[10] CJCH. Watkins, *learning from delayed reward*, Ph.D. thesis, 1989, Cambridge University