

Trajectory Generation for a Mobile Robot by Reinforcement Learning

Masaki Shimizu¹, Makoto Fujita², and Hiroyuki Miyamoto³

¹ Kyushu Institute of Technology, Kitakyushu, Japan
shimizu-masaki@edu.brain.kyutech.ac.jp

² makoto-fujita@edu.brain.kyutech.ac.jp

³ miyamo@brain.kyutech.ac.jp

Summary. Q-learning in the Reinforcement Learning (RL) field is the powerful and attractive tool to make robots generate autonomous behavior. But it needs large amount of computational cost because of its discrete state and action. To generate smooth trajectory with less computational cost, we propose two ingredients for Q-learning. We applied Q-learning to the simulated two wheeled robot to generate trajectory for Ball-To-Goal task in robot soccer. ...

1.1 Introduction

Q-learning shows good performance and easy to use. However it is difficult to generate smooth behavior because the state and action is not continuous. To generate smooth behavior we can use finely divided state and action spaces, but it needs large amount of computational cost. Recently, various work in this field have been proposed (Q-learning in continuous space or action [3][4], hierarchical RL in huge state-action space [5][6].)

In this paper we combine Q-learning and the following simple ingredient. (1) new motor control method (Pulltoy approach) to generate smooth and quick behavior. (2) small number of unevenly divided action space to generate smooth behaviour.

The validity of our method was confirmed by simulated two wheeled robot.

1.2 A Pulltoy Approach

In this section, we first describe our conventional motor control method "Thruster", then we propose a new motor control method "Pulltoy" for smooth and quick behavior.

Our conventional motor control method "Thruster" is consist of "Rotation" and "Forward" commands (ω and V_c in Eq. 1.1).

$$\begin{cases} V_r = V_c + A \times \omega \times L \\ V_l = V_c - A \times \omega \times L \end{cases} \quad (1.1)$$

where $V_{r,l}$ is the velocity of the right and left wheel respectively. L is the distance between each wheel. V_c is the forward velocity of the robot. A is constant. When ω is angle from the robot to the ball, the robot approaches the ball. When A is large the robot prioritizes "Rotate" over "Forward", but the trajectory becomes jerky. The robot can generate smooth trajectory when A is small, but can not respond to the ball quickly if the ball is behind the robot.

Human turns towards the ball and then approach it when the ball is behind him. However, it is difficult and takes a lot of time to correctly analyze and design the exact human approach. We propose a new approach "Pulltoy"⁴ that prioritizes "Rotate" over "Forward" and simplifies the human approach. The robot can quickly rotate by pulling and pushing in the direction of an arrow (Fig.1.1(a)). When the ball is in front of the robot, Pulltoy can approach the ball smoothly. ϕ_b of Fig.1.1(a) can be omitted by transformation.

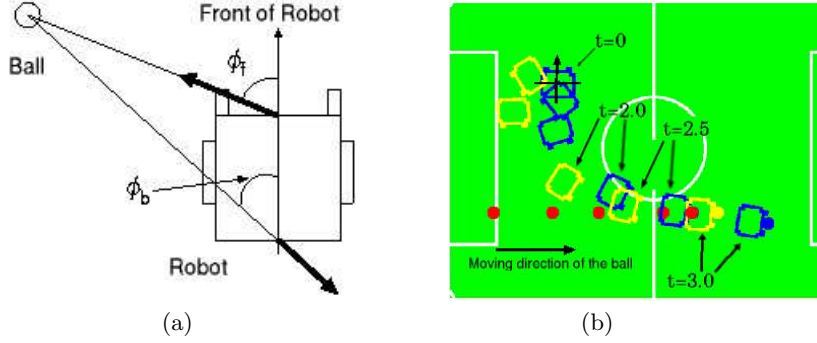


Fig. 1.1. (a) "Pulltoy", (b) Comparison of Pulltoy and Thruster trajectory. Blue and Yellow shows the Pulltoy and Thruster, respectively. Ball moves from left below to right. Pulltoy and Thruster catch the ball at $t=2.5$ and $t=3.0$, respectively.

The Pulltoy approach is defined as

$$\begin{cases} V_r = A \times V_c \\ V_l = A \times V_c (\cos(\phi_f) - \sin(\phi_f)) \end{cases} \quad (0 \leq \phi_f < \frac{\pi}{2})$$

$$\begin{cases} V_r = -A \times V_c (\cos(\phi_f) - \sin(\phi_f)) \\ V_l = -A \times V_c \end{cases} \quad (\frac{\pi}{2} \leq \phi_f \leq \pi)$$

(if ϕ_f is negative, r and l transpose to l and r respectively)

⁴ "Pulltoy" is a toy where a cord ties to the front edge of a small hobbyhorse or car such that a child pull it.

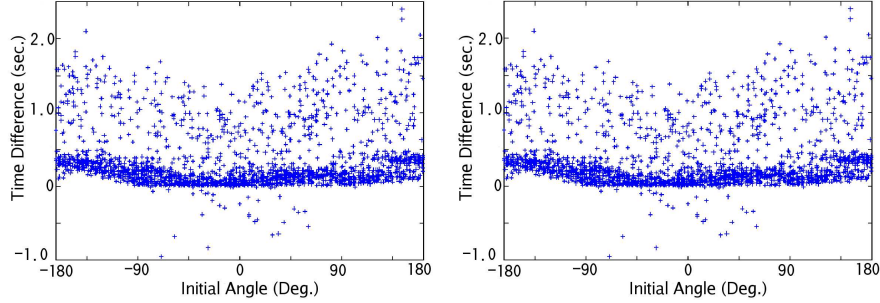


Fig. 1.2. Graph of varying initial angle and distance. Upside in each graph represents that the Pulltoy approach could get the ball before the Thruster approach. Ordinate axis is subtraction of time for getting the ball by using the Thruster and the Pulltoy approach (Thruster - Pulltoy = Time Difference). Abscissa axis is each initial angle and distance from the robot to the ball.

where ϕ_f is the angle from the front edge of the robot to the ball. V_c is the forward velocity (not over maximum velocity of the motor.)

We compare the performance of Pulltoy and Thruster (Fig.1.1(b)). We use the various initial angle and distance from the robot to the ball (Fig.1.2). From these results, we can say that the Pulltoy approach is more effective than the Thruster in almost all condition. Moreover, Pulltoy is able to apply to the different hardware without any change of parameters.

1.3 Q-learning

The action-value-function $Q(s, a)$ calculates the value of taking action a in the state s for policy π . This function is updated as

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (1.3)$$

where r is the reward. γ is the discount rate. s_{t+1} is the state at $t + 1$. α is a constant step-size parameter. Agent chooses action a so as to maximizes the value in the state S .

The raw motor command is often selected as action in mobile robot. In this case, the trajectory becomes jerky when evenly divided the action space. Thus we use the Pulltoy approach as action (angle for input to the Pulltoy.) When we use the Pulltoy, the policy of the Q-learning need not change even if the robot modified the hardware parameters.

The number of state dimension are 12 for angle, 4 for distance from the robot to the ball, 12 for angle, 4 for distance from the robot to the goal, 5 for the velocity of ball. Generally in the mobile robot, action space is evenly divided around robot(Fig.1.3(a)). New action space as shown in Fig1.3(b).

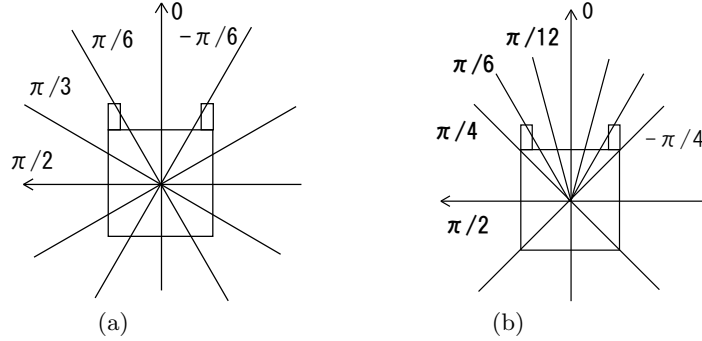


Fig. 1.3. (a)evenly divided action space,(b)unevenly divided action space.

We thickly divide the action space around the target and roughly divide the other action space.

Agent starts learning by choosing exploratory moves because the correct action-value-function is unknown. Getting rewards is difficult at the initial learning phase and takes a lot of time. To solve this problem, we use the Pulltoy trajectory for generating initial value for the policy for choosing action. Learning effectively progresses by starting the exploration from this initial value. Humans arguably progress in the same way.

1.4 Trajectory in different situation

We want to make the robot generate the different trajectory in different situation. The situations are depend on the interaction between a robot, ball and goal. For example, if the situation assumes that the robot has to defend because an opponent robot is approaching with the ball to our goal. In this situation, it is more efficient that the agent puts itself in the line between our goal and the ball than heading straight to the ball.

1.5 Experiment

Reward for Ball-To-Goal task is calculated as

$$r = \begin{cases} 2 & \text{if}(d_g < 20 \text{ and } d_b < 20) \\ 1 & \text{if}(d_b < 20) \\ 0 & \text{otherwise} \end{cases} \quad (1.4)$$

r is reward, d_g is distance from the robot to the goal, d_b is the distance for catching ball. Fig.1.4(a) shows the trajectory after learning. Black and Yellow show the learned trajectory by using the evenly and unevenly divided action space, respectively. Black shows smoother trajectory than yellow.

Motor control of Pulltoy version had to change target as To-Ball or To-Goal task. But in our research we achieved that generates trajectories from the initial position to Goal only RL setting the initial value. We confirmed that the robot could choose appropriate action according to velocity of ball because the state includes the velocity of the ball. Fig.1.4(b) shows convergence of learning.

In the case of relatively simple task, generated trajectory after learning is almost the same with initial trajectory. Even if the robot do not use learning for task, Pulltoy approach for initial value is effective.

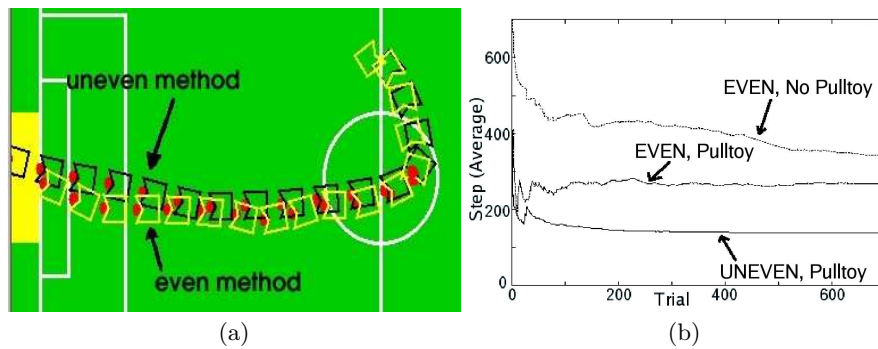


Fig. 1.4. (a)Trajectory of the robot for Ball-To-Goal task that generated by Q-learning,(b)convergence graph of learning

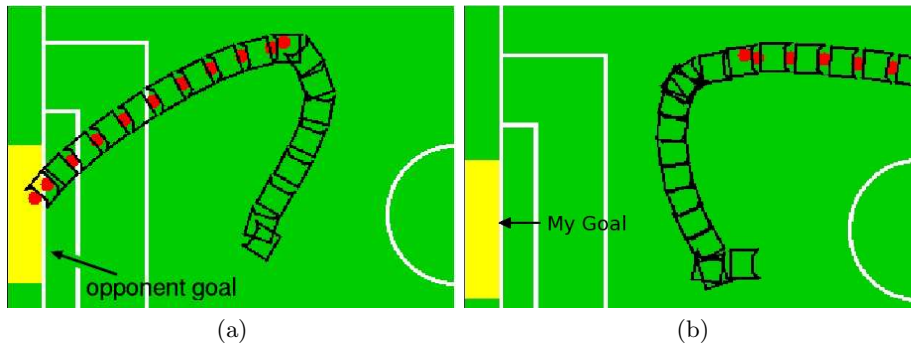


Fig. 1.5. (a)Generated trajectory for offence task,(b)defence task

Fig.1.5(a),(b) show that the robot obtained not only a simple approaching behavior to the ball but also more complex behavior based on strategy according to situation. "Offense" action as shown in Fig.1.5(a) is generated by the robot when the ball is loosed and the robot is able to offense. At the

point where robot finally touches the ball, the robot already has turned to the opponent goal by the learned policy. In the situation that an opponent robot moves to our goal with the ball, Reward of offence strategy is given as

$$r = \begin{cases} 2 & \text{if}(d_g < 20 \text{ and } d_b < 20) \\ 1 & \text{if}(d_b < 20 \text{ and } \phi < |30|) \\ 0 & \text{otherwise} \end{cases} \quad (1.5)$$

ϕ is direction of robot as 0 at opponent goal. the robot has to select a "Defense" strategy. This strategy involves the robot moving to a path between the own goal and the ball and then approach the ball. Even in such cases, the robot as shown in Fig.1.5(b) generates appropriate trajectories. Reward of defense strategy is given as

$$r = \begin{cases} 2 & \text{if}(d_g < 20 \text{ and } d_b < 20) \\ 1 & \text{if}(d_b < 20 \text{ and } \theta < |30|) \\ 0 & \text{otherwise} \end{cases} \quad (1.6)$$

θ is angle from robot to line that connect the ball and the goal. When the robot has strategy or role, designer do not satisfy trajectory by using Pulltoy or the other. We generated trajectory for complex task only by modifying reward function.

1.6 Conclusion

We showed that our method is useful by using the following simple ingredient. (1) Pulltoy approach to generate smooth trajectory. (2) small number of uneven action space to reduce computational cost and generate smooth trajectory.

References

1. R.S.Sutton, A.Barto, "Reinforcement Learning:An Introduction", MIT Press,1998.
2. J.Peng, R.J.Williams, "Incremental Multi-Step Q-Learning", Machine Learning, Vol.22,pp283-290,1996.
3. A.Sherstov, P.Stone, "On Continuous-Action Q-Learning via Tile Coding Function Approximation", In Under Review., June 2004.
4. S.Hagen, B.Kröse, "Neural Q-learning", In Neural Computing & Applications, 12(2), pages 81-88, November 2003.
5. A.Barto, S.Mahadevan, "Recent Advances in Hierarchical Reinforcement Learning", Discrete Event Dynamic Systems:Theory and Applications, 13,41-77,2003.
6. Y.Takahashi, M.Asada, "Multi-Layered Learning Systems for Vision-Based Behavior Acquisition of A Real Mobile Robot", Proceedings of SICE Annual Conference 2003 in Fukui, Vol.CD-ROM, pp.2937-2942, 2003.