Transfer of learned exploration strategies of a mobile robot from a simulated to real environments

Artur Sagitov[1], Tetsuto Takano[2], Shohei Muto[3], Evgeni Magid[1]
[1]Laboratory of Intelligent Robotic Systems (LIRS), Intelligent Robotics Department, Higher Institute of Information Technology and Intelligent Systems, Kazan Federal University, 35 Kremlyovskaya street, Kazan, 420008, Russia
[2]Department of Electronic Information Sciences, Kanazawa University, Kanazawa, Japan
[3]Department of Mechanical Science and Engineering, Kanazawa University, Kanazawa, Japan
E-mail: E-mail: sagitov@it.kfu.ru, shohei0625@stu.kanazawa-u.ac.jp, magid@it.kfu.ru
http://kpfu.ru/robolab.html

Abstract

Reinforcement learning based approaches show promises in various robotic applications, but a significant amount of time and resources are required for a robot to learn optimal behavior. Using virtual environments, we could significantly speed up and improve performance of a target task. We implemented a reinforcement learning based exploration algorithm for a mobile robot, training in Gazebo environment and transferring learned strategy to a real robot. We show that it is convenient and appropriate to use simulation to train strategies for mobile robots.

Keywords: navigation, algorithm, mobile robots, reinforcement learning, exploration.

1. Introduction

Reinforcement learning is a field of machine learning where an agent learns a behavior by interactions within particular environment. All interactions are being graded and the goal of an agent is to achieve maximum possible cumulative grade. Method is inspired by how humans are taught in schools, i.e., taking tests and exams and receiving positive or negative feedback. In robotics, such method presents a way to design and implement complex behavior that are hard to conceptualize. Our initial goal was to build a simple mobile robot that autonomously explores an environment based upon reinforced learning algorithm, without any human interventions.

Major obstacle to be considered using reinforced learning is an amount of time that should be spent for learning optimal behavior, as real trials are slow and costly. One of the possible solution to this problem is the idea of incorporating simulations of a real environment. This process is self-correcting and can improve by obtaining information from real world trials to correct the simulated environment.

One of the major frameworks for reinforcement learning research is OpenAI Gym[1]. This framework provides an easy way to debug and benchmark an algorithm under a variety of different environments. Zamora et al. extended OpenAI Gym functionality with interfaces to Robot Operating System (ROS) and the Gazebo simulator, to simplify the integration with the robotic hardware to validate existing reinforcement learning algorithms in real environments[2].

Our custom mobile robot was integrated into ROS and Gazebo environment with control interfaces on both real and virtual robots unified. The robot was trained using Q-Learning algorithm. Reward increased if it had explored previously unsearched areas on the map and decreased if it had a collision with the walls. With the training completed in the simulation, resulting strategy is easily transferred to the real robot.

2. Related Work

Mathematical concepts of reinforcement learning began when optimal control framework for Markov decision process proposed by Bellman[3] was reformulated by Sutton[4] and Watkins[5]. In recent years, reinforcement learning has become an important method in robotics[6]. It was applied to locomotion[7,8], manipulation[9-12] and autonomous vehicle control[13]. Combining reinforcement learning with general-purpose neural networks shown significant potential, including real-time control of 7 degree-of-freedom manipulators[14-16]. Using large and deep neural networks have made it possible for robots to master complex manipulations with minimal manual engineering, though it is still unknown whether this can be easily applied to an arbitrary task[17-18].

In a study[19] deep neural network (combination of convolutional networks and a long short-term memory network) learns to self-calibrate from a history of previously set of actions and observations. Learning from multiple simulated samples that had consisted of trajectories and objectives, this network was able to learn controlling a robotic arm successfully, achieving goals being set from various frames of reference and using a non-calibrated camera.

3. System Setup

We used OpenAI gym-gazebo extension for a virtual robot and a virtual environment. Gym-gazebo is a combination of OpenAI Gym, ROS and Gazebo. OpenAI provides interface for implementation and testing of an algorithm that will controls real and virtual robot in the Gazebo simulation[20].

3.1. *Learning Robot*

The target for the experiments was a custom-build mobile robot being controlled through a mounted Arduino Uno microcontroller (Fig.1). This learning robot is a four-wheel mobile robot that was controlled by 4 DC motors (one per wheel, two motors on each side), with a spring amortization on each wheel. Velocity controller was implemented for each axis using *ros_control* package. Robot uses differential steering for taking turns and can make complete in-place (pivot) rotations by creating difference in velocities of the left or right side wheels. Sensory information was provided by Hokuyo LIDAR (UTM-30LX) mounted on the top of the frame.

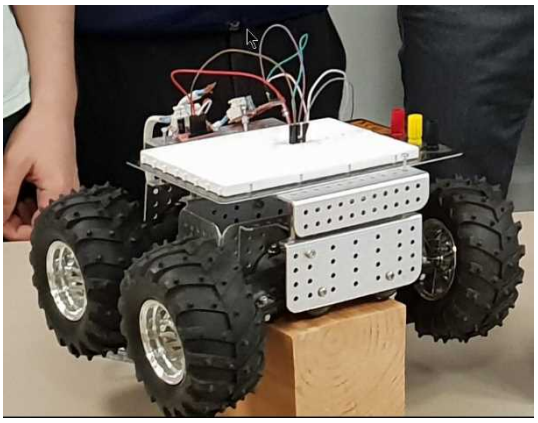

Fig. 1. Learning robot.

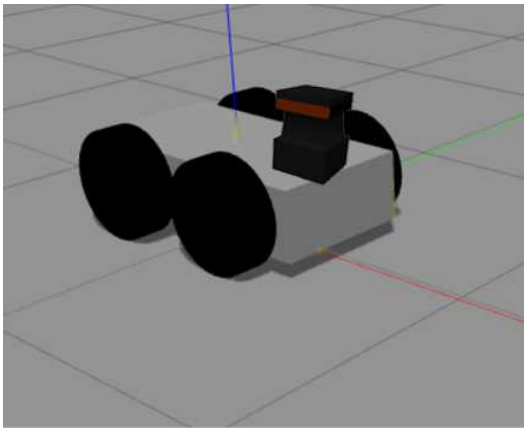

Fig. 2. Virtual learning robot in Gazebo.

ROS package *rosserial* provided a ROS communication protocol over Arduino's UART interface and allowed publishing and subscribing to ROS messages, publishing coordinate frame transforms and accessing ROS system time. The robot control and sensor processing was implemented using *rosserial* protocol by wrapping standard ROS serialized messages and multiplexing multiple topics over to the microcontroller. We implemented virtual representation of the learning robot for the Gazebo simulator, including controller interfaces, controller managers, transmissions and hardware interfaces to match exactly with the real robot (Fig.2), using the same approach that we had exploited in our previous research on crawler robot modelling[21].

3.2. *Learning algorithm*

We implemented multi-step Q-Learning method to facilitate learning of exploration strategy. Q-learning is a

widely used reinforcement learning method introduced by Watkins[5] that uses formulated policies, constructing robot strategy by defining action to take under what circumstances with probabilistic transitions and rewards.

We assumed exploration strategy as a finite Markov decision process (FMDP). Using Q-learning we searched for a strategy that is optimal in a sense that it maximizes an expected value of a total reward (area explored) over all successive steps. Starting from an initial point in decision space, Q-learning can find optimal action-selection strategy given formalized FMDP. Reward was used to provide the reinforcement and in our case it measured an added explored space after an action taken.

Learning robot have a set of states S, and a set of possible actions in this state A. When action $a \in A$ is selected, the robot makes transitions from one state to another state. With each action robot takes from a particular state a reinforcing reward will be changed. When a number of steps reaches a predefined limit or the robot gets stuck we calculate a final reward.

The robot tries to maximize its total reward by adding a maximum reward achievable from future states to the reward for achieving its current state, affecting current action selection given a possible future reward. A total potential reward, which is defined as a weighted sum of expected values of rewards of all future actions starting from the current state, is taken as an objecting function.

At each step, we update the quality function:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_{\alpha} Q(s_{t+1}, a_t) - Q(s_t, a_t) \right], \quad (1)$$

where parameters α (learning rate) and γ (discount factor) are hyperparameters that determine the influence of new information and possible future rewards.

We implemented four possible actions in each state: forward motion (both axes 0.2 m/s), left turn (left axis -0.1 m/s, right axis 0.1 m/s), right turn (left axis 0.1 m/s, right axis -0.1 m/s) and backward motion (both axes -0.2 m/s). The robot was severely penalized if it made a contact with a wall. As soon as a current action was achieved, a new goal was provided and a reward was issued based upon a change of explored territory area.

3.3. *Exploration environment*

As a test for our setup, we selected a standard gym-gazebo labyrinth *GazeboCircuit2TurtlebotLidar-v0*. This environment consists of a simple straight lined circuit with five right turns and one left turn (Fig.3). We used a LIDAR sensor to localize and map the environment, using no other positioning information.

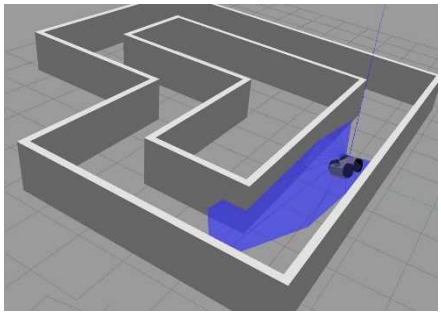

Fig. 3. Testing environment in ROS Gazebo.

4. Results

Our setup allowed us to get around 10 times faster than real time simulation; thus, our virtual robot could perform ten times more learning trials than the real robot. 5000 trials were completed within around 6 hours. Figure 4 demonstrates that the robot exploration strategy significantly improved over time, as the average total reward over time increased.

Even with training performed entirely in a simulator different from a real world, we have obtained exploration strategies, which perform well on the physical robot. We attribute successful transfer to randomizations of simulated environment and unified control interface between real and virtual robot.

We observed several factors that contributed to the difference between the simulation and the reality during executions: the real robot exhibits a lateral and longitude wheel slip, accumulates wheel velocity inaccuracies and is more prone to collisions when driving near the walls.

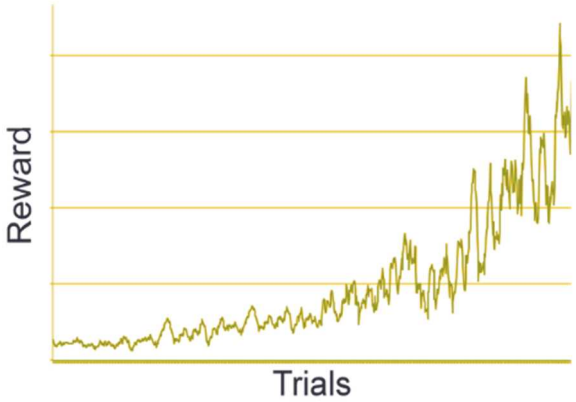

Fig. 4. Reward after trials.

5. Discussion and Future Work

There are several ways we are trying to improve the learning process. In the order of priority, we list:

1. Verify the approach for more complicated robots.
2. Speed up simulation for robots using multiple computers.
3. For complex environments, make simulation run at a faster speed.
4. Further diversify environments and add multiple robots.
5. Implement additional tools for calculating performance metrics for different algorithms.
6. Make automatic recommendations from the resultant strategy.

6. Conclusion

Reinforcement learning plays a significant role in the growing field of machine learning and, in order to overcome difficulties, robotics simulator like Gazebo is shown saving costs and speeding up the learning process. We have shown that reinforcement learning algorithms are capable of learning complex exploration skills from scratch and without purposefully designed trajectories, but our method has a number of limitations, such as simplified robot and test environment. We have plans to address these issues in the future.

Acknowledgements

This work was supported by the Russian Foundation for Basic Research (RFBR), project ID 18-58-45017. Part of the work was performed according to the Russian Government Program of Competitive Growth of Kazan Federal University.

References

1. G. Brockman, et.al., OpenAI gym (arXiv preprint, arXiv:1606.01540, 2016).
2. I. Zamora, et.al. Extending the OpenAI Gym for robotics: a toolkit for reinforcement learning using ROS and Gazebo (arXiv preprint arXiv:1608.05742, 2016).
3. R. Bellman, A Markovian decision process, in *Journal of Mathematics and Mechanics* (1957) pp. 679-684.
4. R.S. Sutton, Temporal credit assignment in reinforcement learning, (PhD Thesis, University of Massachusetts, Amherst, MA., 1984).
5. C.J.C.H. Watkins, Learning from Delayed Rewards (Ph.D. thesis, Cambridge University, 1989).
6. J.Kober, J.A. Bagnell, J. Peters, Reinforcement learning in robotics: A survey, in *The Int. J. of Robotics Research*, 32, (2013), pp. 1238-1274.
7. N. Kohl and P. Stone, Policy gradient reinforcement learning for fast quadrupedal locomotion, in *Int. Conf. on Robotics and Automation* 3 (2004), pp. 2619-2624.
8. G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng, Learning CPG-based biped locomotion with a policy gradient method: Application to a humanoid robot, in *Int. J. of Robotic Research*, **27**(2) (2008), pp. 213–228.
9. J. Peters and S. Schaal, Reinforcement learning of motor skills with policy gradients, in *Neural Networks*, **21**(4) (2008), pp. 682–697.
10. E. Theodorou, J. Buchli, and S. Schaal, Reinforcement learning of motor skills in high dimensions, in *Int. Conf. on Robotics and Automation* (2010) pp. 2397-2403.
11. J. Peters, K. Mulling, and Y. Altun, Relative entropy policy search, in *AAAI Conference on Artificial Intelligence* (2010), pp. 1607-1612.
12. M. Kalakrishnan, L. Righetti, P. Pastor, and S. Schaal, Learning force control policies for compliant manipulation, in *Int. Conf. on Intelligent Robots and Systems* (2011), pp. 4639-4644.
13. P. Abbeel, A. Coates, M. Quigley, and A. Ng, An application of reinforcement learning to aerobatic helicopter flight, in *Advances in Neural Information Processing Systems* (2006) pp. 1-8.
14. P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, Learning and generalization of motor skills by learning from demonstration, in *Int. Conf. on Robotics and Automation* (2009), pp. 763-768.
15. J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel, Trust region policy optimization, in *Int. Conf. on Machine Learning* (2015), pp. 1889-1897.
16. S. Levine, C. Finn, T. Darrell, and P. Abbeel, End-to-end training of deep visuomotor policies, *J. of Machine Learning Research*, **17**(1) (2016), pp. 1334-1373.
17. M. Deisenroth and C. Rasmussen, PILCO: a model-based and data efficient approach to policy search, in *Int. Conf. on Machine Learning* (2011), pp. 465-472.
18. T. Moldovan, S. Levine, M. Jordan, and S. Abbeel, Optimism-driven exploration for nonlinear systems, in *Int. Conf. on Robotics and Automation* (2015), pp. 3239-3246.
19. F. Sadeghi, A. Toshev, E. Jang and S. Levine, Sim2Real Viewpoint Invariant Visual Servoing by Recurrent Control, in *IEEE Conf. on Computer Vision and Pattern Recognition* (2018), pp. 4691-4699.
20. I., Afanasyev, A., Sagitov, E., Magid. ROS-based SLAM for a Gazebo-simulated mobile robot in image-based 3D model of indoor environment. In *Int. Conf. on Advanced Concepts for Intelligent Vision Systems*, (Springer, Cham, 2015), pp. 273-283.
21. M., Sokolov et al., Modelling a crawler-type UGV for urban search and rescue in Gazebo environment, *In IEEE Int. Conf. on Artificial Life and Robotics (2017).*