Learning-based local navigation in dynamic environments

Matej Dobrevski, Danijel Skočaj

Fakulteta za računalništvo in informatiko
Univerza v Ljubljani, Večna pot 113, 1000 Ljubljana
E-pošta: matej.dobrevski@fri.uni-lj.com

Abstract

Safe navigation is a crucial capability of mobile robots. Unstructured environments with moving obstacles are especially difficult to navigate in. In this paper we train a neural-network-based reinforcement-learning model to perform local navigation and planning in environments containing moving pedestrians. We evaluate our approach against the standard Dynamic Window Approach (DWA) and show that we can outperform it in presence of dynamic obstacles.

1 Introduction

Mobile robot navigation is an enabling technology for the development of machines that can operate autonomously in unstructured environments. As such, it has received a substantial amount of interest in the research community [6, 4, 18, 12].

Systems for planning the movement of robots in complex environments are usually organized in two stages: a global planner and a local planner. The global planner has access to a map of the whole environment and plans efficient paths to the goal location. The local planner has access to the on-board sensors and insures that the robot follows the global path while reacting to any changes in the environment, such as new obstacles. The local planner can also serve as a control algorithm, directly generating motor movement commands.

Local planning is especially significant since for some environments maintaining an accurate map can be impractical or impossible, because of frequent changes or lack of identifiable features. In such cases we can obtain the position of the robot and the goal in the environment with the use of an external localization system, such as GPS, indoor localization beacons, marker detection or another method. In those kinds of environments it is possible to navigate using a local planner.

For local planners one of the most difficult tasks is to plan movement in the presence of dynamic obstacles. While there has been a significant amount of research on this topic, most approaches assume the knowledge of the position and velocity of all the dynamic obstacles in the environment [5, 18, 2], which requires an additional step of perception in the navigation method.

In this research we model the problem of navigation as a Markov Decision Process (MDP) and train a neural network to perform local navigation/planning in environments densely populated with humans, directly from sensor inputs, joining the steps of perception and navigation in a single network. The network is trained in a reinforcement learning (RL) setting using the state of the art Proximal Policy Optimization (PPO) [16] algorithm. The learning is performed in a simulator with different stages which contain realistically simulated human crowds. We evaluate our method against the established Dynamic Window Approach (DWA) to local planning [6] and show that in the presence of dynamic obstacles it achieves superior results.

2 Related work

Local navigation has been a long studied field in robotics. Among the most significant approaches to local navigation are the DWA [6], the potential field methods [4] and Velocity obstacles [5] which have spawned numerous extensions and modifications such as the Global Dynamic Window Approach [3], Convergent Dynamic Window Approach [12], Harmonic Potential Functions [8], Reciprocal Velocity Obstacles [18] and many more.

The approaches which consider dynamic obstacles assume the knowledge of the position and velocity of all the moving obstacles in the scene, while we assume only the access to the range scanner of the robot and its global position relative to the goal.

In recent years learning based approaches have become popular. In [13] the authors trained a neural network to map laser range measurements and the relative goal position to control commands. However, they use expert demonstrations for training the network in a supervised manner, which can be impractical. Furthermore, they do not address dynamic obstacles.

Reinforcement learning has been extensively applied to the problem of navigation. Until recently these approaches [1, 9, 19] have mainly relied on severe discretizations of the environment which can severely limit the accuracy and precision of the navigation. Recently, following the improvement of the techniques for training neural networks in a reinforcement learning setting [11, 10, 15, 16], several approaches have been proposed, which perform robot navigation directly from sensor readings. In [14] a variant of Q-learning was used to generate safe commands on the basis of a single image. An obsta-

cle avoidance network was trained using the dueling and double-Q learning techniques in [20]. In [21] the authors developed a visual navigation method for known scenes. The work done in [17] is most closely related to our approach, however in their work they do not consider dynamic obstacles in the environment.

3 Learning a navigation policy

3.1 Navigation as a RL problem

We model the problem of navigation in dynamic environments as an MDP defined with (S, A, R, T, γ), where:

- S defines the set of all possible observations. In our case an observation $s \in S$ consists of three consecutive laser-scans of the environment (64 measurements in a 240° angle) together with the distance and angle to the goal location as well as the current velocity of the robot.
- A defines the set of all possible actions. For out differential drive robot $A = \{(v, w)|v \in [0, 0.5], w \in [-2, 2]\}$, where v is the transnational and w is the angular speed of the robot.
- The reward function $R : S \rightarrow R$ is constructed such that the robot receives a reward of -0.1 for each time-step, a reward of -120 for a collision with an agent or the environment, and a reward of 100 for reaching the goal.
- The transition function $T : S \times A \rightarrow S$ defines how the states evolve, and is implicitly defined by the simulator we use for learning, the robot dynamics and the sensor readings.
- The discount factor γ is set to $\gamma = 0.99$.

For optimization we use the state-of-the-art PPO algorithm which is a more efficient variant of the Trust Region Policy Optimization (TRPO) [15] algorithm. Both algorithms are policy gradient algorithms which directly optimize the policy network. The main difference from vanilla policy gradient methods is that both PPO and TRPO limit the gradient updates to the network coefficients so that the KL divergence of the old and new policies does not surpass a predetermined limit. As a policy gradient method PPO is also an on-policy learning method, which means that in updating we can only use transitions that have been generated by the policy we are updating.

3.2 Network

The policy network is represented in Figure 1. The three last laser observations are fed through three convolutional layers and two fully connected (dense) layers. The output of the dense layers is joined with the distance and angle to the goal, as well as the current velocity of the robot. Further 4 dense layers process the joined representation before the last layer, where we generate the translational and rotational speeds which have a tanh activation.

The value network, which is used during the optimization for calculating the action advantage according to the Generelized Advantage Estimation (GAE) algorithm shares the same design, with the difference being in the output layer, where there is only one linear output.

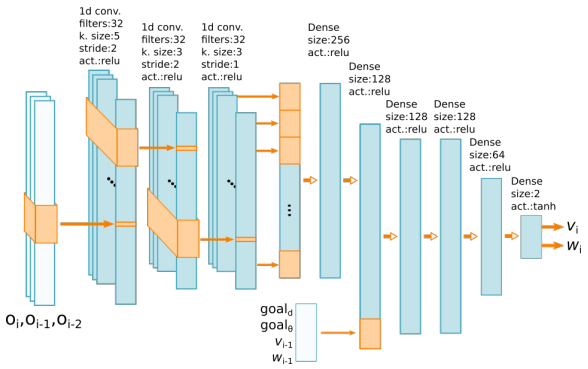

Figure 1: The design of the policy network. Three convolutional layers followed by 5 fully connected ones. The input are three consecutive laser scans, the angle and distance to the goal as well as the current translational and rotational speeds, which are joined with the 2nd fully connected layer.

3.3 Training

In order to learn the navigation policy we perform a curriculum learning procedure in two phases. In the first phase the robot is put in a polygon without any obstacles besides the walls of the room (Figure 2 left). The goal location is generated randomly $[0, 3]m$ away from its location. We train the policy for $100K$ steps in this environment. In the second phase (represented in Figure 2 right) we add static obstacles as well as dynamic obstacles which are simulated humans with their movement modeled by the Social Forces Model [7]. In the second phase the goal location is generated randomly $[2, 11]m$ away from the robot. We train the robot in this environment for further $250K$ steps.

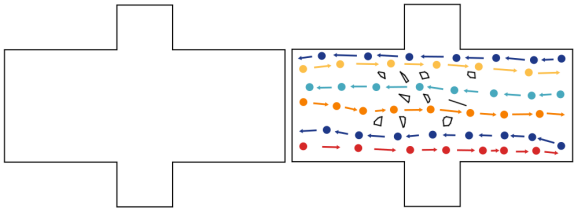

Figure 2: The two environments representing phase 1 and phase 2 during learning.

We find that this type of training is necessary because in the beginning the robot acts randomly and if the goal is far away, and the environment is cluttered with obstacles, it fails to reach the goal and experience a reward, thus failing to learn to progress in learning. Another possibility would be to augment the reward so that the robot receives a small reward for getting closer to the goal, however this type of fine tuning the reward function can lead to unexpected behaviours and interfere with the main objective of learning.

The robot and the environment are simulated in our extension of the pedsim_ros [1] library and controlled through the exposed ROS [2] topics. The learning algorithm is implemented in Python and Tensorflow.

4 Evaluation

4.1 Evaluation procedure

We evaluate our learned policy vs. the performance of the DWA on the four stages represented in Figure 3. We evaluate the DWA approach because it is a very popular approach to local navigation and because it is the default navigation method implemented in the navigation stack of our Turtlebot robot. In each stage we generate 100 starting locations and goals, and then evaluate the performance of both approaches. The performance is evaluated in terms of how many times the goal is reached successfully, with the length of the successful trajectories and with the duration of the successful episodes. The four stages are:

- Stage 1: An empty room with size $10m \times 5m$. The starting and goal points are generated randomly.
- Stage 2: A room with size $10m \times 5m$ with humans entering and exiting in different directions. The starting and goal locations are the same as in Stage 1.
- Stage 3: A large hall with size $20 \times 10m$ with randomly generated convex obstacles scattered. The starting location is generated randomly (so that it is not in collision) and the goal location is generated at a distance $\in [3, 6]m$ from the robot.
- Stage 4: The same hall as in Stage 3 but with humans entering and exiting in different directions. The starting and goal locations are at a distance of $\in [1, 4]m$ from the robot.

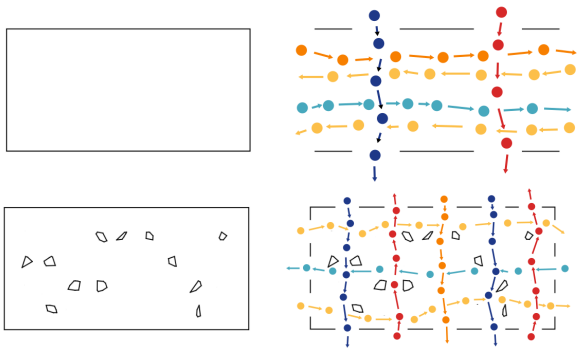

Figure 3: Four stages used for evaluation. Stages 1 and 3 contain only static obstacles, while stages 2 and 4 contain both static and dynamic obstacles (humans).

[1] https://github.com/srl-freiburg/pedsim_ros
[2] https://www.ros.org/

4.2 Results

As can be seen in Table 1 in the first stage both algorithms navigated successfully to all of 100 given goals. The distance traveled as well as the time to reach the goal are longer for our policy. In the second stage, when the moving humans are added to the scene, the number of goals that are reached drops dramatically for both navigation methods, however the proposed method reaches 6 more goals than the baseline. It should be noted that perfect score in the stages containing humans is virtually impossible, since the humans move with a greater speed than the robot and do not care if they collide with it.

Table 1: Quantitative results of the navigation evaluation. Shown are the number of goals that were reached, the average distance per goal and the average time per goal.

	goals		avg. dist.		avg. time	
Stage	**DWA**	**Ours**	**DWA**	**Ours**	**DWA**	**Ours**
1	100	100	2.70	2.84	5.86	7.51
2	71	77	2.55	2.68	5.59	7.77
3	82	94	6.48	7.60	14.24	21.15
4	65	75	3.37	3.56	7.87	10.26

In the third stage when the static convex obstacles are introduced again the number of goals that are reached drops compared to Stage 1. In evaluating the trajectories we found that DWA fails when the robot tries to pass through tight openings, in avoiding the first obstacle it makes a sharp turn and can not stop before it hits the second obstacle which can be out of the sight of the 240 degree laser. For our policy the collisions occur when the robot is approaching straight on a sharp obstacle. We suspect that because of the reduction of the laser to 64 readings a sharp obstacle can interchangeably appear in the two frontal readings of the laser and an oscillatory behaviour occurs.

The fourth stage is the most demanding of all the stages. Perfect performance in this stage is not possible, because of the moving agents. As can be seen in Table 1 our policy manages to reach 10 more goals than the baseline. The length of the trajectories is slightly longer for the RL-Policy and the time taken is somewhat longer.

In Figure 4 we visualize three executed trajectories for the same starting and goal locations. We can see that the trajectories generated by the baseline are generally smoother, while our RL-Policy sometimes generates more choppy trajectories.

In evaluating our policy among moving obstacles we can see that it learned a few different evasion maneuvers in the presence of moving obstacles: in the presence of a fast moving human, it generally stops until the human has passed, or turns and reduces its speed until the obstacle is gone. In the presence of a slow moving obstacle, it will try to increase its speed and overtake the human. Some of these maneuvers are depicted in Figure 5 where we drive the robot to the same goal location in a $3m$ wide corridor with humans moving in different directions.

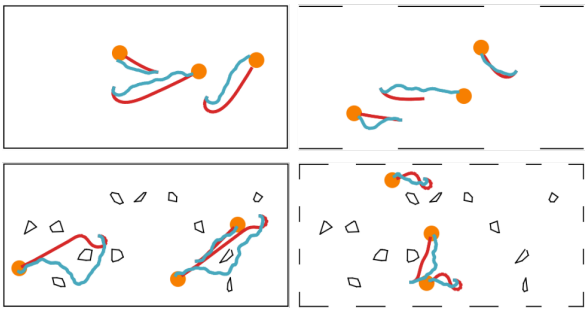

Figure 4: Visualizations of some of the performed trajectories during evaluation. Blue is RL-Policy and red is DWA policy.

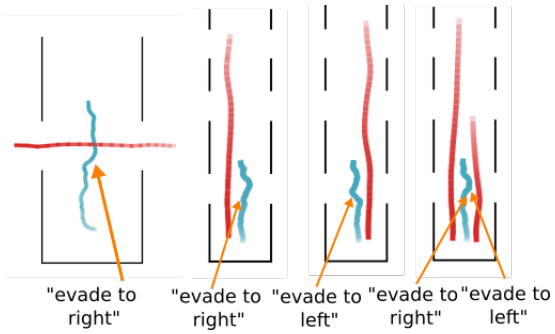

Figure 5: Visualizations of some of the learned collision avoidance techniques. Blue is RL-Policy and red are movements of humans. Saturation representes the passage of time.

5 Conclusion

In this paper we introduced a new way of performing local navigation in dynamic environments. Our approach is to train a neural network to perform this navigation on the basis of only the reading of a range scanner, current velocity of the robot and the distance and angle to the goal. Our approach does not depend on the knowledge of the position and velocity of the moving objects in the scene. To train this policy we use a reinforcement learning method, and construct our simulator, develop a reward function and a two-stage training procedure.

Our evaluation shows that our navigation method can outperform the established DWA approach in the presence of dynamic humans.

References

[1] Abdel, M., Jaradat, K., Al-rousan, M., Quadan, L.: Reinforcement based mobile robot navigation in dynamic environment. Robotics and Computer Integrated Manufacturing (1) (2011)

[2] van den Berg, J., Guy, S.J., Lin, M., Manocha, D.: Reciprocal n-body collision avoidance. In: Pradalier, C., Siegwart, R., Hirzinger, G. (eds.) Robotics Research. pp. 3–19. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)

[3] Brock, O., Khatib, O.: High-speed navigation using the global dynamic window approach. In: Proceedings 1999 IEEE ICRA) (1999)

[4] Dudek, G., Jenkin, M.: Computational principles of mobile robotics, Computational principles of mobile robotics, Cambridge University Press, Cambridge (2000)

[5] Fiorini, P., Shiller, Z.: Motion Planning in Dynamic Environments Using Velocity Obstacles. IJRR (1998)

[6] Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. IEEE Robotics & Automation Magazine **4**(1), 23–33 (1997)

[7] Helbing, D., Molnár, P.: Social force model for pedestrian dynamics. Phys. Rev. E **51**, 4282–4286 (1995)

[8] Kim, J.., Khosla, P.K.: Real-time obstacle avoidance using harmonic potential functions. IEEE Transactions on Robotics and Automation **8**(3), 338–349 (1992)

[9] Konar, A., Chakraborty, I.G., Singh, S.J., Jain, L.C., Nagar, A.K.: A Deterministic Improved Q-Learning for Path Planning of a Mobile Robot. IEEE Transactions on Systems, Man, and Cybernetics: Systems **43**(5), 1141–1153 (2013)

[10] Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T.P., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous Methods for Deep Reinforcement Learning. arXiv **48**, 1–28 (2016), http://arxiv.org/abs/1602.01783

[11] Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature (2015)

[12] Ogren, P., Leonard, N.E.: A convergent dynamic window approach to obstacle avoidance. IEEE Transactions on Robotics (2005)

[13] Pfeiffer, M., Schaeuble, M., Nieto, J., Siegwart, R., Cadena, C.: From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In: IEEE ICRA (2017)

[14] Sadeghi, F., Levine, S.: $(CAD)^2$RL: Real Single-Image Flight without a Single Real Image. arXiv:1611.04201 (2016), http://arxiv.org/abs/1611.04201

[15] Schulman, J., Levine, S., Jordan, M., Abbeel, P.: Trust Region Policy Optimization. ICML (2015)

[16] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. CoRR **abs/1707.06347** (2017)

[17] Tai, L., Paolo, G., Liu, M.: Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. IROS (2017)

[18] van den Berg, J., Ming Lin, Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: IEEE ICRA. pp. 1928–1935 (2008)

[19] Wen, S., Chen, X., Ma, C., Lam, H.K., Hua, S.: The Q-learning obstacle avoidance algorithm based on EKF-SLAM for NAO autonomous walking under unknown environments. RAS (2015)

[20] Xie, L., Wang, S., Markham, A., Trigoni, N.: Towards monocular vision based obstacle avoidance through deep reinforcement learning. In: RSS 2017 workshop on New Frontiers for Deep Learning in Robotics (2017)

[21] Zhu, Y., Mottaghi, R., Kolve, E., Lim, J.J., Gupta, A., Fei-Fei, L., Farhadi, A.: Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: IEEE ICRA) (2017)