Improving Mobile Robot Navigation with Deep Neural Robot Control

Max Ferguson
CS230 Deep Learning

Mentor: Ahmadreza Momeni

Introduction

We propose a new method which combines reinforcement learning with traditional search-based path-planning, to solve long navigation problems. This approach, referred to as Deep Neural Robot Control (DNRC), performs exceptionally well on several robot navigation tasks, learning a human-like policy for navigation and collision avoidance. We also introduce framework for zero-shot transfer of learned policies from simulation to the physical world.

Approach

A good navigation algorithm should ensure the robot moves to the target position quickly and safely. Specifically, a good mobile robot control system should:

- Avoid colliding with walls and stationary objects
- Anticipate the movement of people and other robots, and act to avoid collision wherever possible
- Minimize acceleration and rotation of the robot, so as to conserve battery resources
- Reach the target destination as quickly as possible, given the above conditions

The navigation problem is represented as a Partially Observed Markov Decision Process (POMDP). Observations of the state space include the velocity of the robot, as well as a multi-layer map of the surroundings:

$$o(s) = \begin{bmatrix} \dot{x}_{robot} + \epsilon_4 \\ \dot{y}_{robot} + \epsilon_5 \\ \dot{\theta}_{robot} + \epsilon_5 \\ x_{target} + \epsilon_6 \\ y_{target} + \epsilon_7 \\ \mathcal{I}_{walls} \\ \mathcal{I}_{objects} \\ \mathcal{I}_{robots} \\ \mathcal{I}_{checkpoints} \\ \mathcal{I}_{targets} \end{bmatrix}$$

Observations include automatically generated bitmaps describing the surroundings

$$R_{batt}(s,a) = -0.001 \left\| \max(|a| - 0.6, 0) \right\|^2$$

$$R_{spin}(s,a) = -0.001 \left\| \dot{\theta}_{robot} \right\|^2$$

$$R_{collision}(s,a) = \max(-1, -0.3e^{20(0.4-d)})$$

$$R(s,a) = R_{target}(s,a) + R_{collision}(s,a) + R_{batt}(s,a) + R_{spin}(s,a)$$

Navigation and Control with Deep Reinforcement Learning

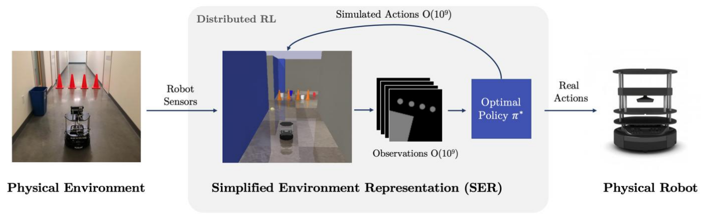


Figure 1: Learning a physical-world control policy through simulation and distributed reinforcement learning

Deep Neural Robot Control

In DNRC, we learn three functions by interacting with the environment: The actor $\pi(s)$, the critic $Q_\pi(s,a,t)$, and the value function $V_\pi(s,t)$. At the start of each episode, the A* search algorithm is used to find the shortest distance from the current position to the target, relying on $V_\pi(s,t)$ as a distance metric. This path is mapped back to a path in Euclidean space. Finally, we place checkpoints along the shortest path and execute $\pi(s)$, repeatedly to move between each checkpoint towards the goal.

The agent is trained using a variant of the DDPG algorithm [1], along with the APEX asynchronous optimizer with distributed prioritized replay buffers [2].

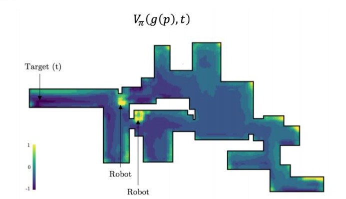


Figure 2: Value function projected into Euclidean space and overlaid on the house floorplan.

Simplified Environment Abstraction

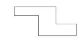


Figure 3: Playground Environment

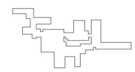


Figure 4: House Environment

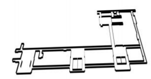


Figure 5: Building Environment

Each environment is generated by scanning a real building using sensors on the mobile robot. During training, observations from these environments are generated using a distributed computing cluster with 576 CPU cores and 2304 GB of RAM.

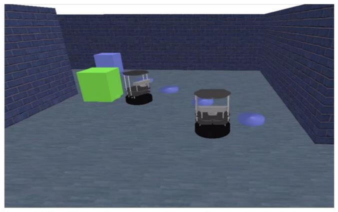
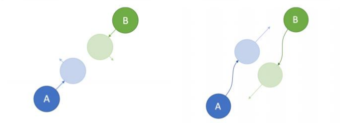


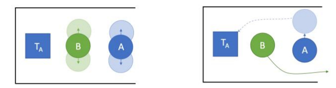
Figure 6: Simplified environment abstraction. The two black robots must reach their respective targets, depicted as green and purple cubes.

Results



(a) 100 million training steps, 2 hours (b) 800 million training steps, 16 hours

Figure 6: DNRC learns to anticipate the movement of other objects, and always pass other robots on the same side (for example on the right).



(a) 100 million training steps, 2 hours (b) 800 million training steps, 16 hours

Figure 7: Narrow hallways can lead to a deadlock situation when robots are travelling in opposing directions. DNRC learns to let the other robot passed in a narrow hallway, so as to avoid deadlock

Table 1: Average reward in different environments compared to vanilla RL.

Environment	PPO	TD3	DNRC (Ours)
Playground (3 Robots)	0.91	0.93	**0.96**
House (3 robots)	0.44	0.45	**0.91**
Building (5 robots)	0.03	0.05	**0.34**

Conclusion

We proposed a new method, Deep Neural Robot Control (DNRC), to improve robot control and navigation. This approach performs exceptionally well on several robot navigation tasks, learning a human-like policy for navigation and collision avoidance. We demonstrated that the DNRC policy can be transferred to a physical robot, using a simulated environment model as an abstraction layer over the physical world.

References

[1] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods", arXiv preprint arXiv:1802.09477, 2018.

[2] E. Liang, R. Liaw, P. Moritz, R. Nishihara, R. Fox, K. Goldberg, J. E. Gonzalez, M. I. Jordan, and I. Stoica, "Rllib: Abstractions for distributed reinforcement learning", arXiv preprint arXiv:1712.09381, 2017