Learning fine-grained control for mapless navigation

Fred de Villiers, Willie Brink

Division of Applied Mathematics
Department of Mathematical Sciences
Stellenbosch University

Problem definition

- Agent : two-wheel differential drive robot.
- Environment : flat surface in indoor enclosed space.
- Goal : find collision-free path to target.
- Agent receives continuous state information and learns continuous control policy at the actuator level.
- Agent has no access to, and does not maintain, an external map.
- An oracle provides location of target (angle and distance) relative to agent's position.

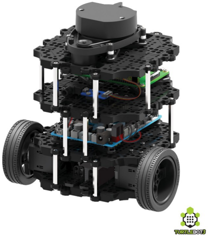

FIGURE 1 – TurtleBot 3 Burger

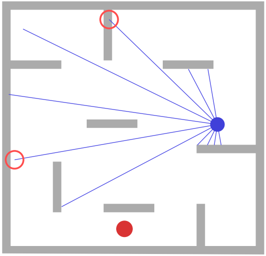

FIGURE 2 – The agent (blue circle) and target (red circle) in the simulated environment.

Markov Decision Process

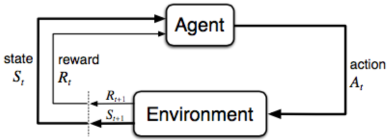

FIGURE 3 – The agent-environment interaction formulated as an MDP.

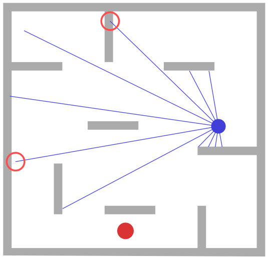

FIGURE 4 – The training environment.

- State : laser/ultrasonic rangefinder readings augmented with target angle and distance.
- Action : continuous angular velocity imparted on each wheel.
- Reward : scalar feedback balancing goals of safety and attaining target position.

Our aims :

- Design reward function to encourage local recovery and exploration.
- Eliminate dependence on platform-specific controllers to translate high-level movement commands.

Markov Decision Process

- Agent's goal : choose A_t to maximise expected discounted return (discount factor $\gamma \in [0, 1)$)

$$
\begin{aligned}
G_t &\triangleq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots \\
&= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \dots) \\
&= R_{t+1} + \gamma G_{t+1}
\end{aligned}
\tag{1}
$$

- Action selected according to policy π :

$$
\begin{gathered}
\pi(a|s) = p(A_t = a \mid S_t = s) \\
a = \pi(s)
\end{gathered}
\tag{2}
$$

- Value of taking action a in state s under policy π (action-value function) :

$$
\begin{aligned}
q_\pi(s, a) &\triangleq \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right] \\
&= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a \right]
\end{aligned}
\tag{3}
$$

Q-learning

- Discrete state space.
- Discrete action space.
- Construct two-dimensional matrix of state-action pairs $Q(s, a)$.
- Approximate true action-value function with learned action-value function.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[\underbrace{\overbrace{R_{t+1} + \gamma \max_a Q(S_{t+1}, a)}^{\text{improved estimate}} - Q(S_t, A_t)}_{\text{TD error}} \right] \tag{4}$$

Disadvantages of Q-table :

- Combinatorial explosion
- Unvisited states during exploration
- Maximising over continuous action space not computationally feasible

Deep deterministic policy gradients (DDPG)

- Continuous state and action space.
- The critic network $Q(s, a|\theta^Q)$ estimates the action-value function.
- The actor network $\mu(s|\theta^\mu)$ maps an action to a given state.
- The **target label** for training critic network :

$$y_i = r_i + \gamma Q(s_{i+1}, \mu(s_{i+1}|\theta^{\mu}_{\text{target}})|\theta^{Q}_{\text{target}}). \tag{5}$$

Sample mini-batch and optimise sequence of loss functions

$$L_i(\theta_i^Q) = \mathbb{E}_{s,a,r,s' \sim unif(\mathcal{M})}\left[\left(y_i - Q(s_i, a_i|\theta_i^Q)\right)^2\right], \tag{6}$$

with gradient update

$$\nabla_{\theta_i} L_i(\theta_i^Q) = \mathbb{E}_{s,a,r,s' \sim unif(\mathcal{M})}\left[y_i - Q(s_i, a_i|\theta_i^Q)\nabla_{\theta_i} Q(s_i, a_i|\theta_i^Q)\right], \tag{7}$$

using stochastic gradient descent.

Two key ideas for stable results :

- Separate target networks $Q(s, a|\theta^Q_{\text{target}})$ and $\mu(s|\theta^\mu_{\text{target}})$.
- Mini-batches sampled from replay memory $\mathcal{M}$.

DDPG continued

- The objective to update actor network :

$$J(\theta^{\mu}) = \mathbb{E}\left[Q(s,a)|_{s=s_t, a_t=\mu(s_t)}\right]. \tag{8}$$

Deterministic policy gradient computed by taking derivative of objective function w.r.t. policy parameter :

$$\nabla_{\theta^{\mu}} J(\theta^{\mu}) = \nabla_a Q(s,a) \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu}), \tag{9}$$

and mean of policy gradients in mini-batch calculated to perform stochastic gradient **ascent** :

$$\nabla_{\theta^{\mu}} J(\theta^{\mu}) \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^{Q})|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})|_{s=s_i}. \tag{10}$$

Actor-critic network architecture

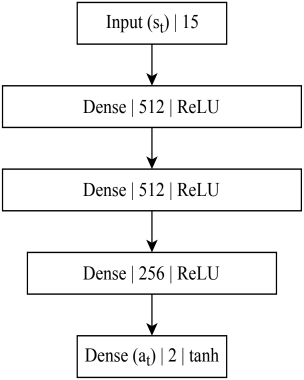

FIGURE 5 – Actor network

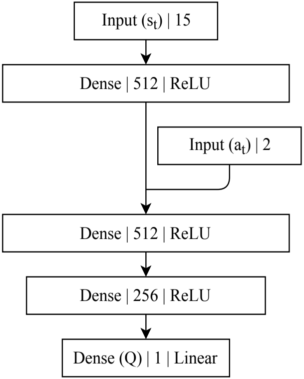

FIGURE 6 – Critic network

Reward function

- Typical distance-based reward (DB reward) :

$$r(\mathbf{s}_t, \mathbf{a}_t) = \begin{cases} r_{\text{found}} & \text{if agent reaches target,} \\ r_{\text{crash}} & \text{if agent collides,} \\ \beta_1(d_t - d_{t+1}) & \text{otherwise.} \end{cases} \tag{11}$$

- Our proposed distance-based reward with velocity term (DB-V reward) :

$$r(\mathbf{s}_t, \mathbf{a}_t) = \begin{cases} r_{\text{crash}} & \text{if agent collides,} \\ r_{\text{unsafe}} & \text{if min sensor} < d_{\text{safe}} \\ \beta_1 \max\{0, (d_t - d_{t+1})\} + \beta_2 v_t & \text{otherwise.} \end{cases} \tag{12}$$

Experimental setup

- Different agent trained for each reward function.
- Performance evaluated on four previously unseen test environments.
- Success rate recorded on 300 random start and target positions on each map.

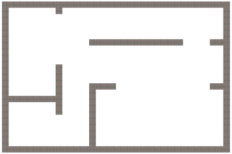

FIGURE 7 – Test map 1

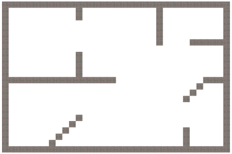

FIGURE 8 – Test map 2

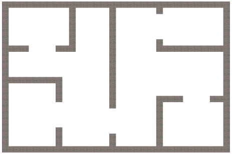

FIGURE 9 – Test map 3

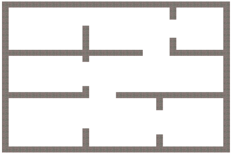

FIGURE 10 – Test map 4

Results

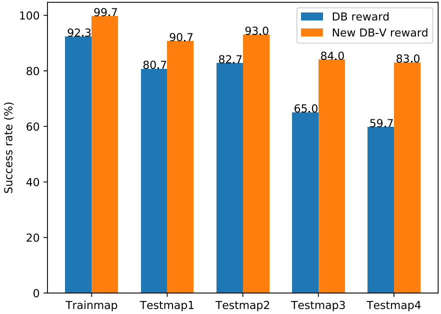

FIGURE 11 – Success rate of the agents in reaching random targets in training and previously unseen test environments.

Results

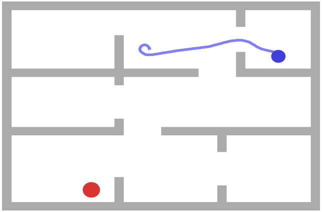

FIGURE 12 – Mapless agent : DB reward

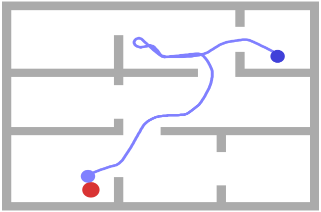

FIGURE 13 – Mapless agent : DB-V reward

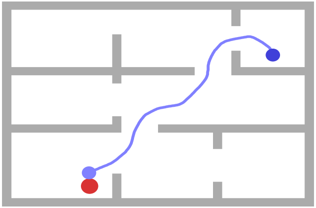

FIGURE 14 – Global planner with DB-V low-level control

Conclusion

- Continuous control policy trained to solve challenging navigation tasks.
- New reward function improves the agent's ability to solve difficult navigation problems.
- Learned policy may form effective low-level component when coupled with global path planner.
- Technique extends naturally to dynamic environments.