DCLab
Dynamics and Control Laboratory

Autonomous Navigation System for Mobile Robot by using Twin-Delayed Deep Deterministic Policy Gradient (TD3)

Students : SOEURN Sothearrith & THORNG Rithy

Advisor : Asst. Prof. Dr. SRANG Sarot

Department of Industrial and Mechanical Engineering

Dynamics and Control Laboratory

1. Introduction
2. Methodology
3. Results and Discussions
4. Conclusions and Future Works
5. References

1 Introduction

- Background
- Problem statement
- Objectives
- Scope and Limitation

2 Methodology

3 Results and Discussions

4 Conclusions

5 References

1 Introduction

- Background
- Problem statement
- Objectives
- Scope and Limitation

2 Methodology

3 Results and Discussions

4 Conclusions

5 References

Background

Reinforcement Learning (RL) is the science of learning to make decisions.

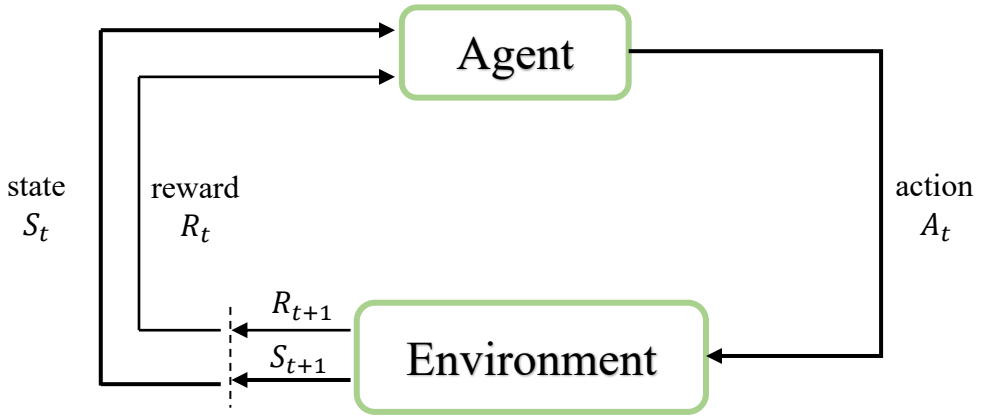


Fig.1: The agent-environment interaction in a **Markov Decision Process** (Sutton & Barto, 2018).

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$$

$$\mathrm{p}(s'|s,a) \doteq \Pr\{S_{t+1} = s' |\ S_t = s, A_t = a\}$$

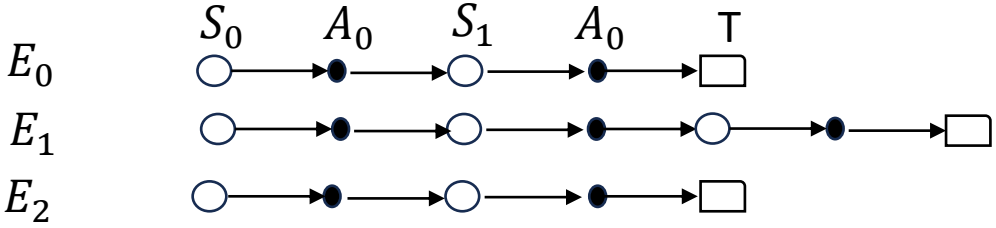
Background

- **Agent:** The learner or decision maker.
- **State:** A situation that the agent perceives in a given time.
- **Action:** A move that the agent can make in the environment.
- **Reward:** Value judgment from the agent's action.
- **Sum of discounted rewards (Return):**

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ \ldots \ = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Background

- **Episode :** Complete RL run.



- **Policy :** Agent's behavior.
  - In the deterministic case, the policy is described by

    $\mu(s): S \rightarrow A$
  - In the stochastic case, the policy is described by

    $\pi(a|s): \mathrm{A} \times S \rightarrow [0,1]$

The agent takes action based on its Policy.

Background

- **State Value Function:**

$$v_\pi(s) \doteq \mathrm{E}_\pi[\, G_t | S_t = s] = \mathrm{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,|\, S_t = s\,] \qquad \text{for all s} \in S$$

- **State-Action Value Function:**

$$q_\pi(s,a) \doteq \mathrm{E}_\pi[\, G_t \;|\, S_t = s, A_t = a] = \mathrm{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,|S_t = s, A_t = a]$$

$$\text{for all s} \in S, a \in A$$

Background

Deep Reinforcement Learning (DRL) = Reinforcement Learning + Neural Networks.

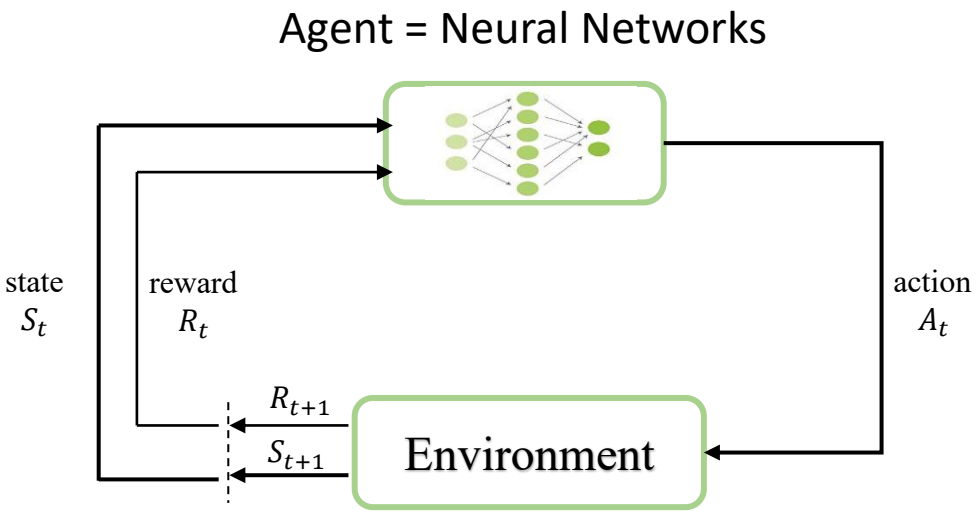


Fig.2: Interaction between agent and environment in DRL.

1 Introduction

- Background
- Problem statement
- Objectives
- Scope and Limitation

2 Methodology

3 Results and Discussions

4 Conclusions

5 References

Problem Statement

- Autonomous navigation is a key challenge for mobile robots.
- Traditional navigation algorithms often struggle with **dynamic environments** and **unpredictable obstacles**.
- **Deep Reinforcement Learning (DRL)** enables robots to learn navigation strategies directly from experience (Mnih et al., 2015).
- **Deep Q-Network (DQN)** is limited to discrete action spaces and cannot handle continuous control tasks effectively (Silver et al., 2014).
- **Deep Deterministic Policy Gradient (DDPG) addresses** continuous actions but suffers from overestimation bias (Lillicrap et al., 2016).
- **TD3 improves stability and learning efficiency**, making it well-suited for continuous and complex navigation tasks (Fujimoto et al., 2018).

1 Introduction

- Background
- Problem statement
- Objectives
- Scope and Limitation

2 Methodology

3 Results and Discussions

4 Conclusions

5 References

Objectives

- To investigate the effectiveness of the TD3 algorithm.
- The robot should navigate to each goal sequentially while avoiding collision.

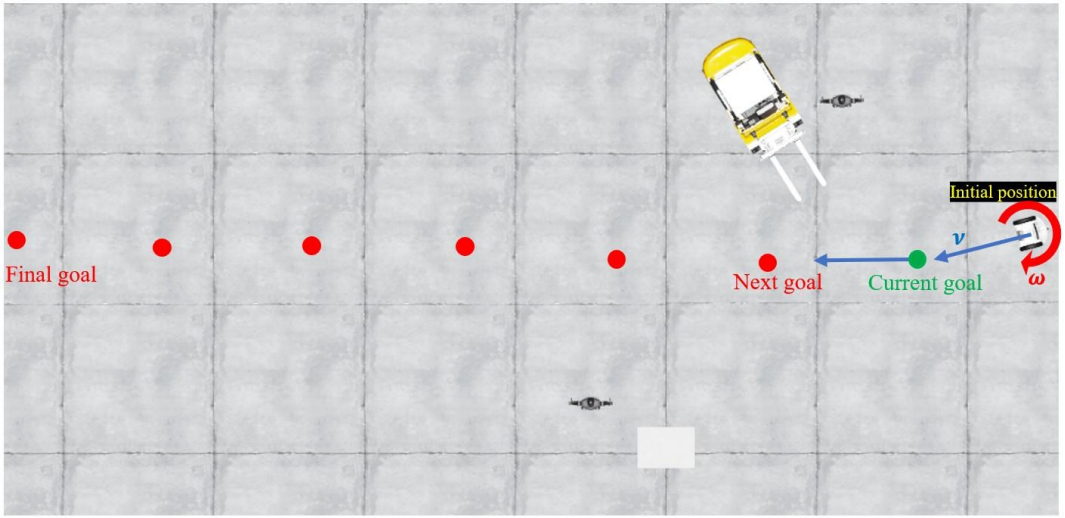


Fig. 3: The seven sequential waypoints.

1 Introduction

- Background
- Problem statement
- Objectives
- Scope and Limitation

2 Methodology

3 Results and Discussions

4 Conclusions

5 References

Scope and Limitation

Scope

- This project uses Isaac Sim to simulate robots in a virtual 3D environment.
- In this study, perfect robot data was used for training and evaluation.

Limitation

- A differential drive robot is used for this study.
- Robot is equipped with laser range scanner (LiDAR).

1 Introduction

2 Methodology

- Environment Setup
- TD3 Implementation
- Agent and Environment Integration
- Evaluation and Metrics

3 Results and Discussions

4 Conclusions

5 References

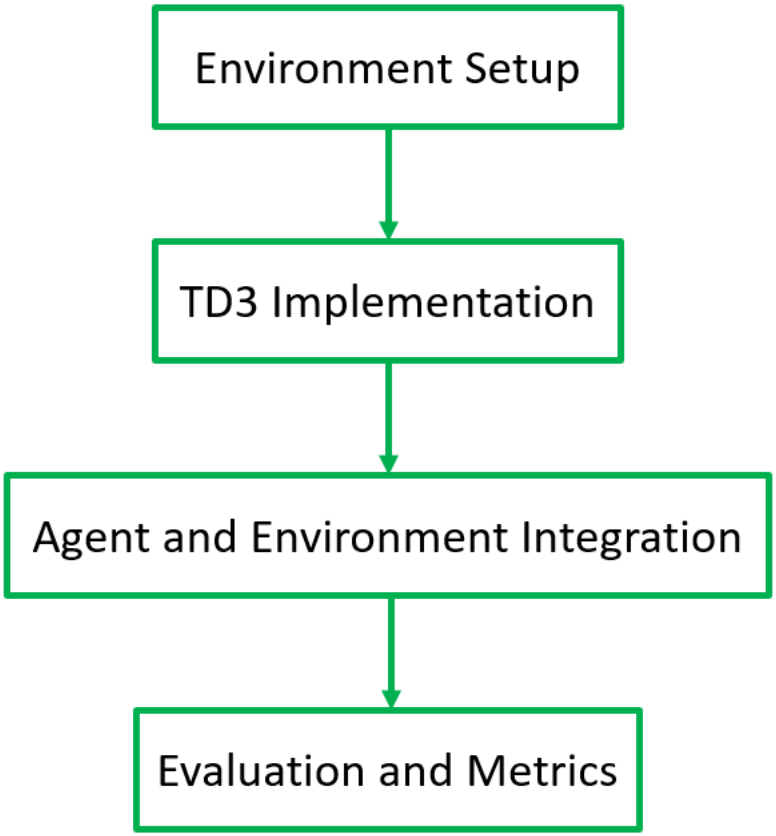


Fig. 4: Workflow of the Project.

1 Introduction

2 Methodology

- Environment Setup
- TD3 Implementation
- Agent and Environment Integration
- Evaluation and Metrics

3 Results and Discussions

4 Conclusions

5 References

Building the world environment in Isaac Sim

- 30m × 19m – realistic warehouse environment built in **Isaac Sim** .
- **Shelving unit**, **tables**, and **crates** to mimic real warehouse.
- **Static humanoid models** and **forklifts** were added to represent potential **dynamics obstacles**.
- **Friction properties** to ensure accurate robot turning.

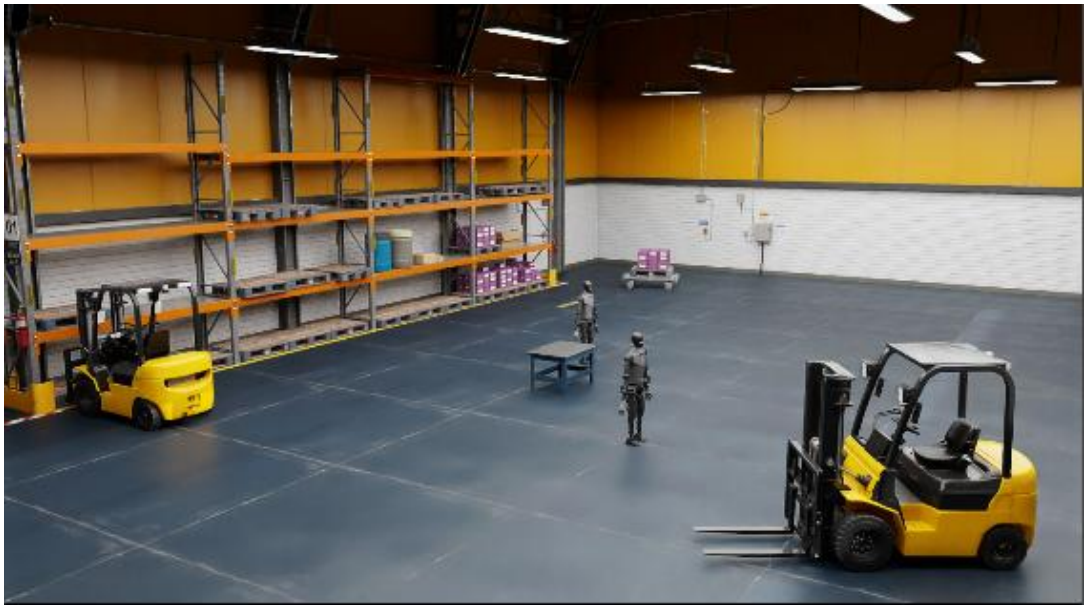


Fig. 5: Simulated warehouse environment.

Utilizing the Pre-built Carter Robot

Only the *left_wheel* and *right_wheel* joints were accessed for motor control in the simulation.

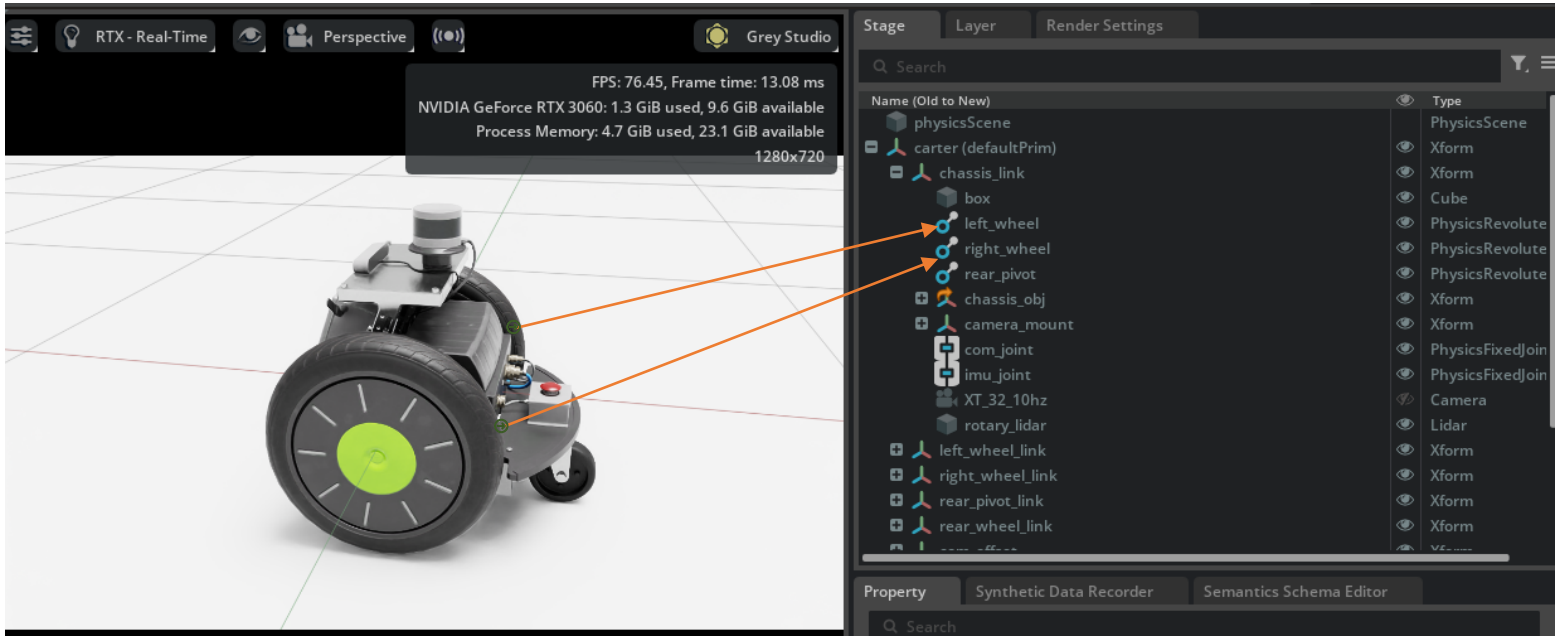


Fig. 6: Carter robot loaded into simulation.

Simulated LiDAR

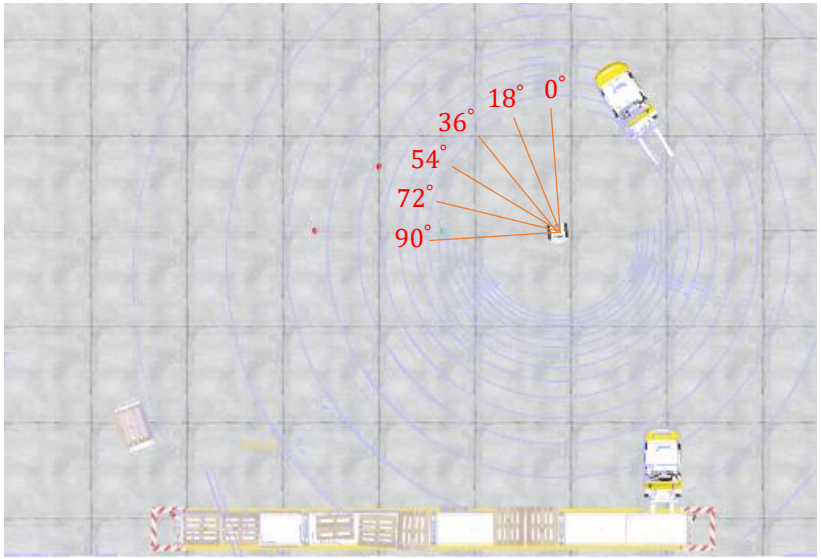


Fig. 7: Simulated laser scan.

- 30m range LiDAR with 360° coverage.
- Only 20 beams were used with 18° intervals.

State Space and Action Space

Policy Output

- Linear and angular velocity.

Policy Input

- LiDAR scan data.
- Distance to the target.
- Heading information.
- Action space.

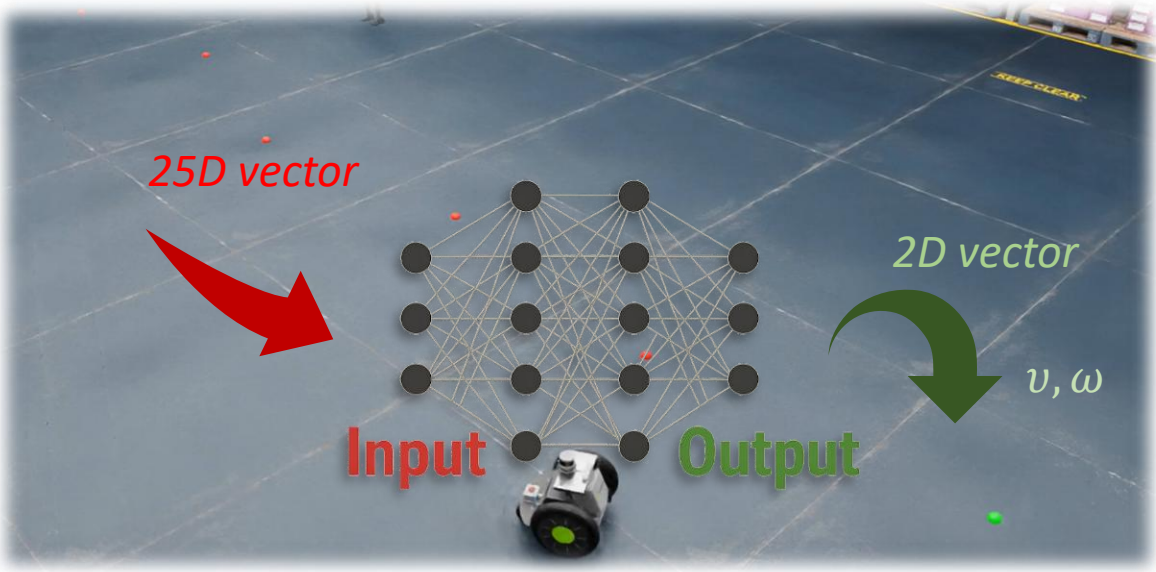


Fig. 8: Neural Network Policy Mapping from State Inputs to Velocity Commands.

Scoring System

$$R = R_{pos} + R_{head} + R_{goal} + R_{lidar} + R_{smooth} + R_{angular}$$,where

- $R_{pos} = (d_{prev} - d_{current})$
- $R_{head} = e^{(-\frac{|\theta_e|}{K})}$
- $R_{goal} = \begin{cases} r_g, & if\ d_{current} < \epsilon \\ 0, & otherwise \end{cases}$.
- $R_{lidar} = \begin{cases} -\alpha, & if\ d_{min} < d_{safe} \\ 0, & otherwise \end{cases}$.
- $R_{smooth} = -\beta \|\Delta u\|^2$.
- $R_{angular} = -\gamma |\omega_z|$.

1 Introduction

2 Methodology

- Environment Setup
- TD3 Implementation
- Agent and Environment Integration
- Evaluation and Metrics

3 Results and Discussions

4 Conclusions

5 References

TD3 Implementation

TD3 contains 2 actor (policy) networks: actor-evaluate and actor-target. They have same architecture and different random weights.

Notation: $a = \mu(s)$

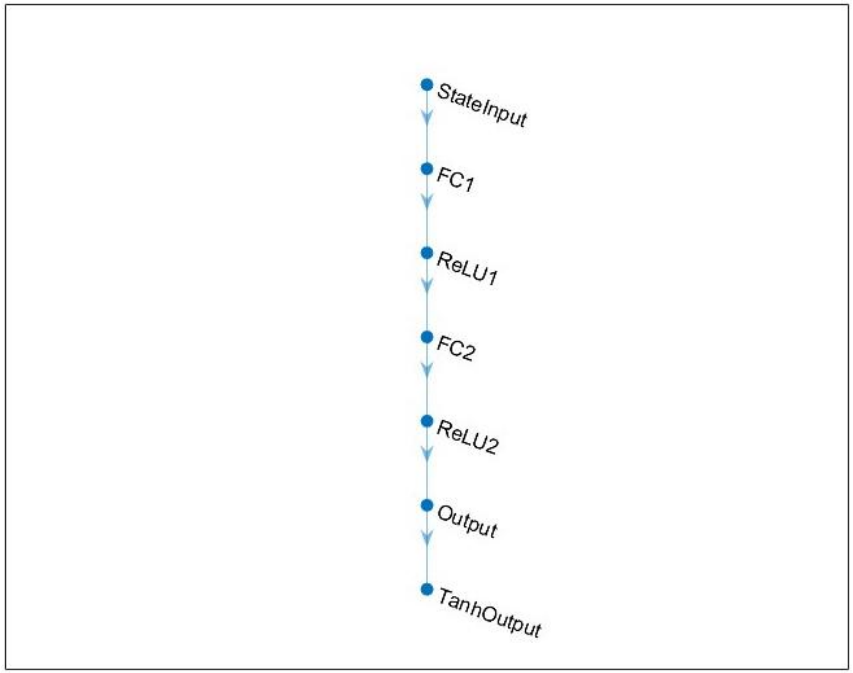


Fig. 9: Actor Network Architecture.

TD3 Implementation

TD3 contains 4 critic networks: 2 evaluate critics and 2 target critics. They have same architecture and different random weights.

Notation: Q-value $= q(s, a)$

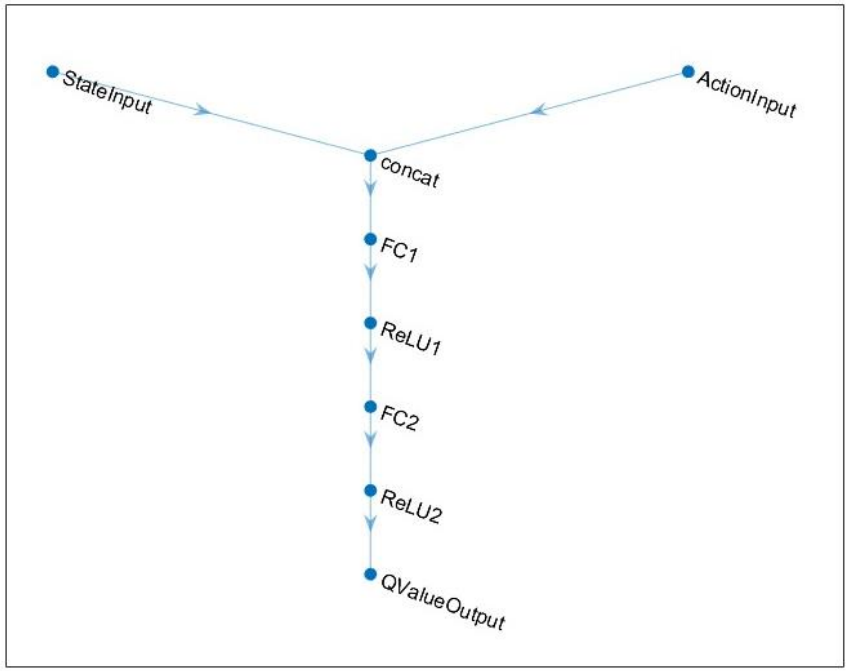


Fig. 10: Critic Network Architecture.

TD3 Implementation

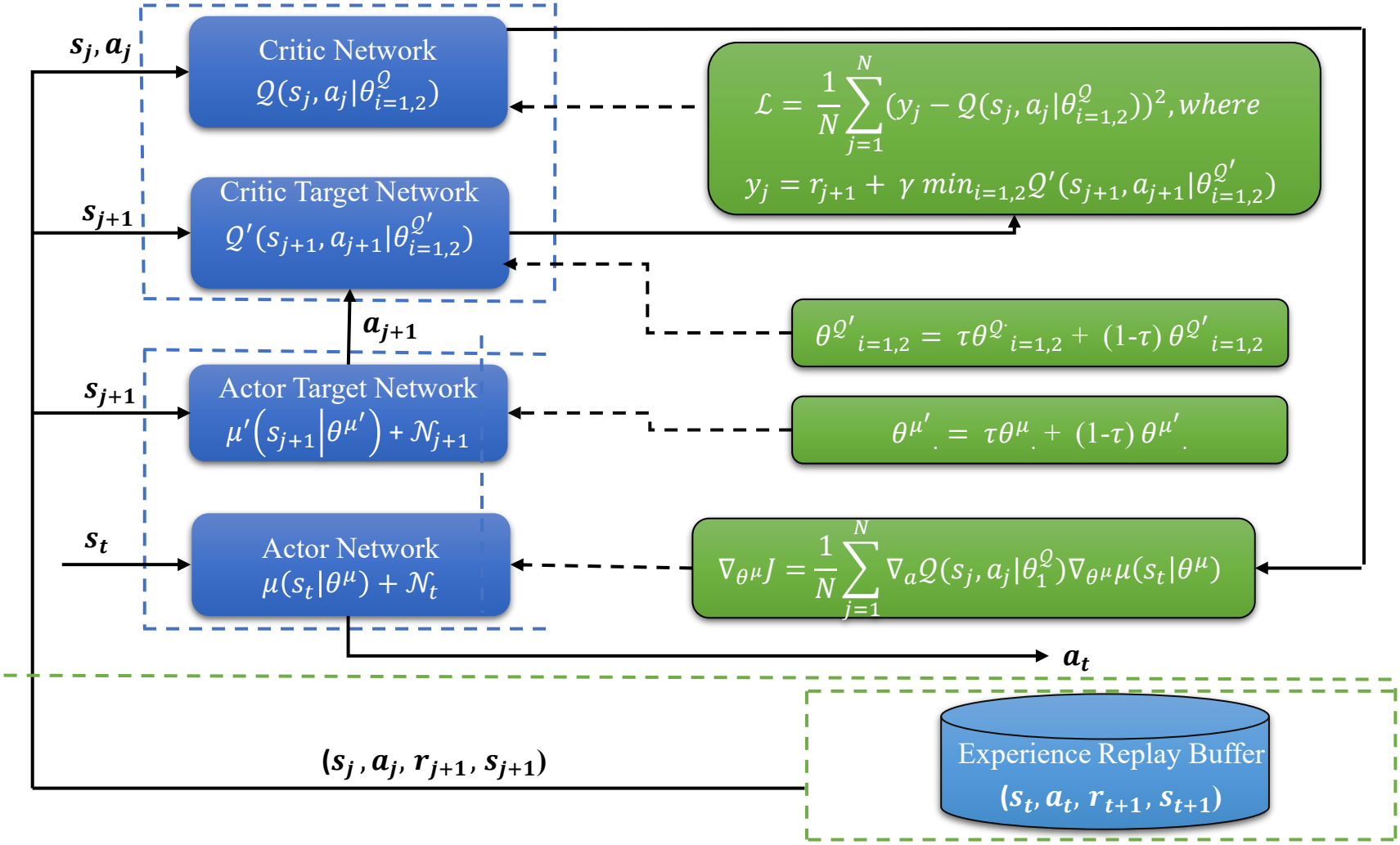


Fig. 11: Architecture of TD3 algorithm.

1 Introduction

2 Methodology

- Environment Setup
- TD3 Implementation
- Agent and Environment Integration
- Evaluation and Metrics

3 Results and Discussions

4 Conclusions

5 References

Agent and Environment Integration

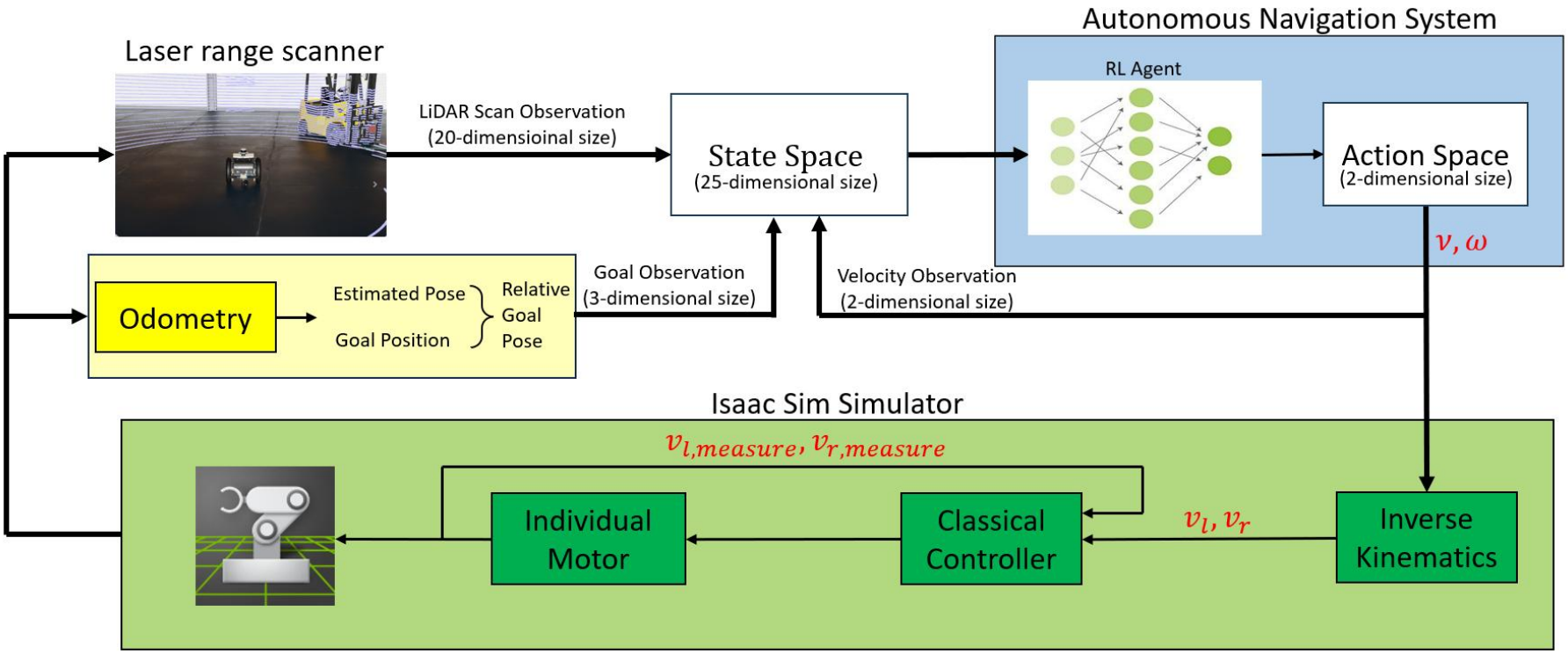


Fig. 12: Overall system framework.

1 Introduction

2 Methodology

- Environment Setup
- TD3 Implementation
- Agent and Environment Integration
- Evaluation and Metrics

3 Results and Discussions

4 Conclusions

5 References

Evaluation and Metrics

- Total Reward per Episode $= \sum_{t=1}^{T} r_t$.
- Policy Loss $= -\frac{1}{N}\sum_{j=1}^{N} Q(s_j, \mu(s_j|\theta^{\mu})|\theta_1^Q)$.
- Critic Loss $= \frac{1}{2N}\sum_{j=1}^{N}[(y_j - Q(s_j, a_j|\theta_1^Q))^2 + (y_j - Q(s_j, a_j|\theta_2^Q))^2]$.

1 Introduction

2 Methodology

3 Results and Discussions

- Training Results
- Testing Results
- Discussions

4 Conclusions

5 References

1 Introduction

2 Methodology

3 Results and Discussions

- Training Results
- Testing Results
- Discussions

4 Conclusions

5 References

Average Reward

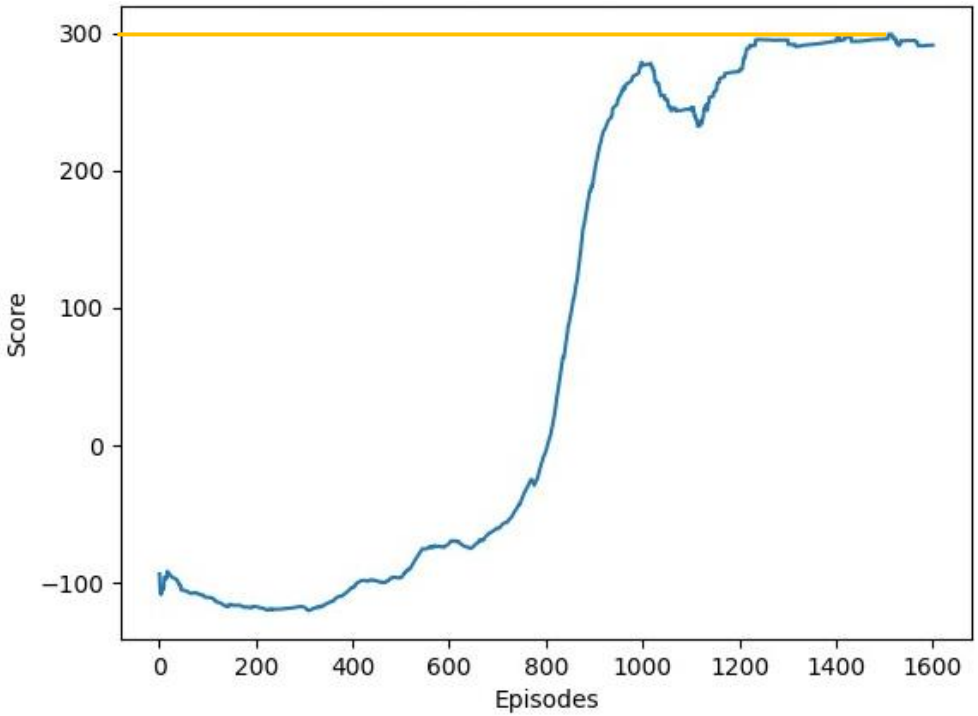


Fig. 13: Learning Curve.

- The score converged to the value of 315.

Critic Loss

$$\text{Critic Loss} = \frac{1}{2N}\sum_{j=1}^{N}[(y_j - Q(s_j, a_j|\theta_1^Q))^2 + (y_j - Q(s_j, a_j|\theta_2^Q))^2]$$

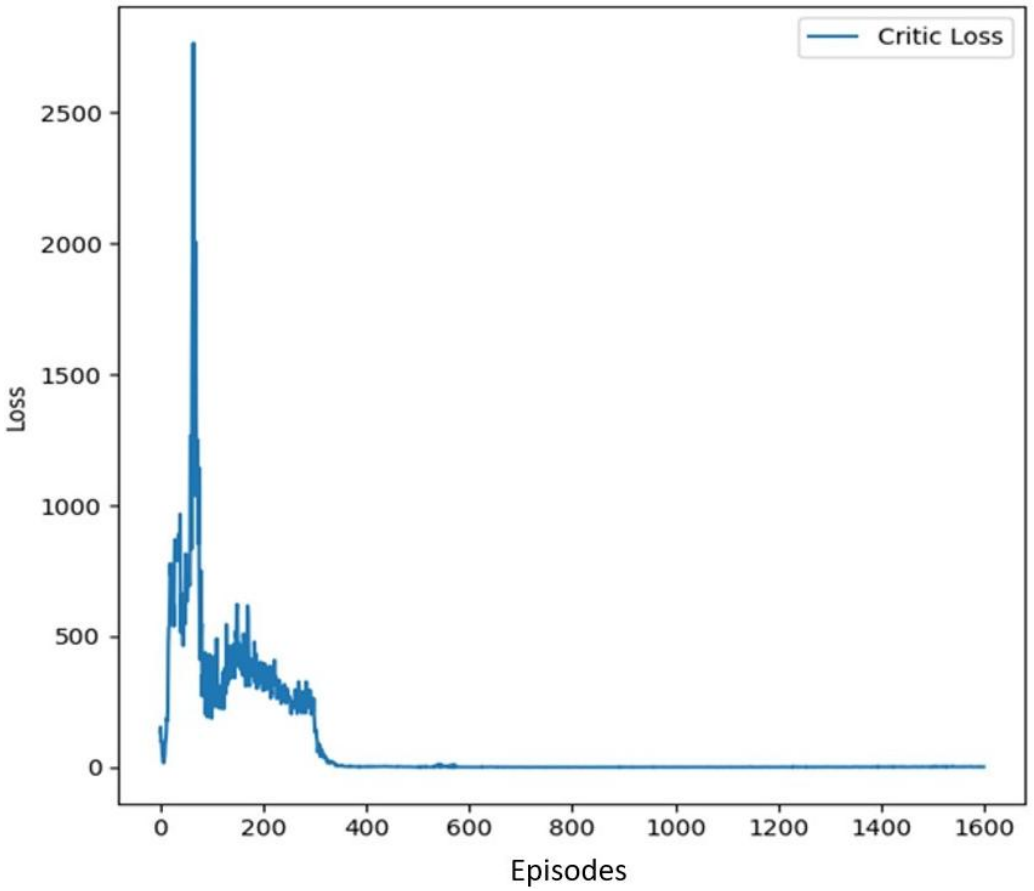


Fig. 14: Critic Loss Graph.

Actor Loss

$$\text{Policy Loss} = -\frac{1}{N}\sum_{j=1}^{N} Q(s_j, \mu(s_j|\theta^{\mu})|\theta_1^{Q})$$

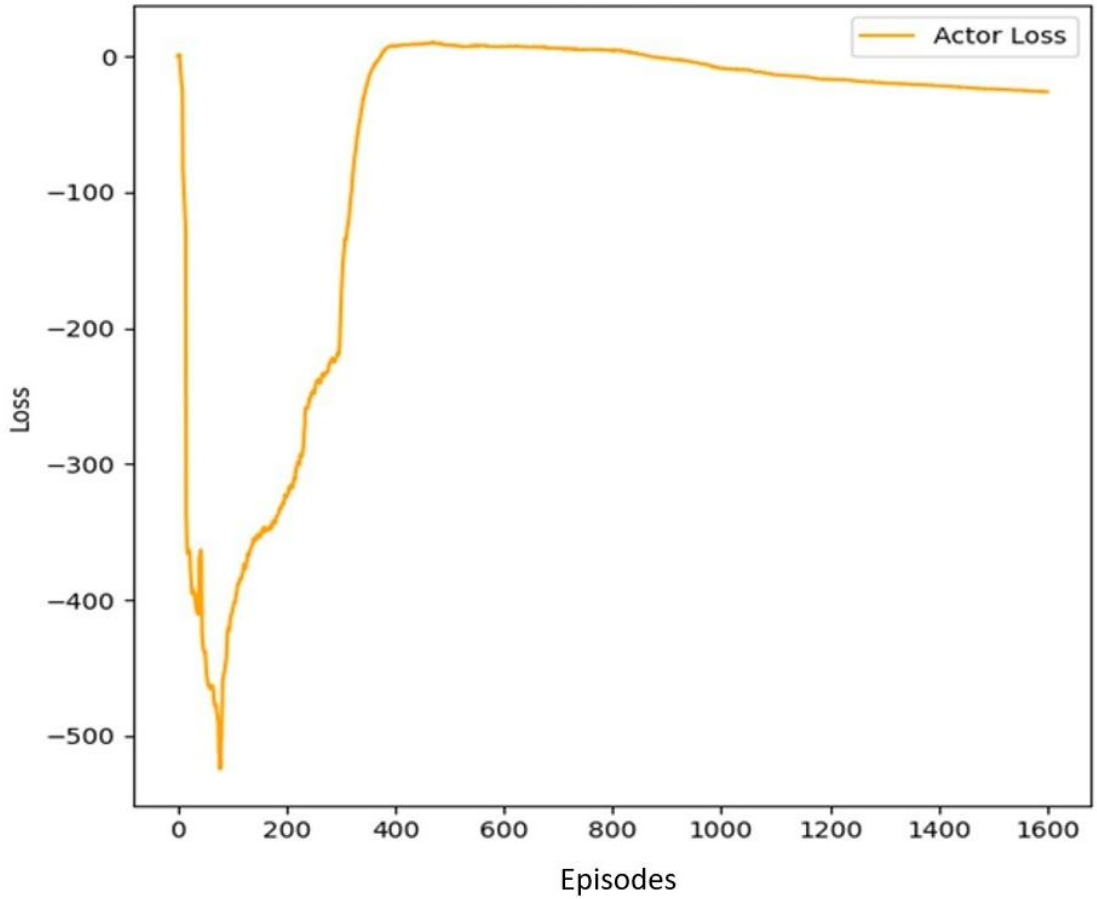


Fig. 15: Actor Loss Graph.

Performance of the robot at the initial training

1 Introduction

2 Methodology

3 Results and Discussions

- Training Results
- Testing Results
- Discussions

4 Conclusions

5 References

Evaluation episode rewards using trained Agent

- **Episode 0**: Score = 315.68.
- **Episode 1**: Score = 314.37.
- **Episode 2**: Score = 314.96.
- **Episode 3**: Score = 314.54.

- The agent successfully generalized its behavior.
- Navigating the environment reliably.

Robot Velocities while Testing

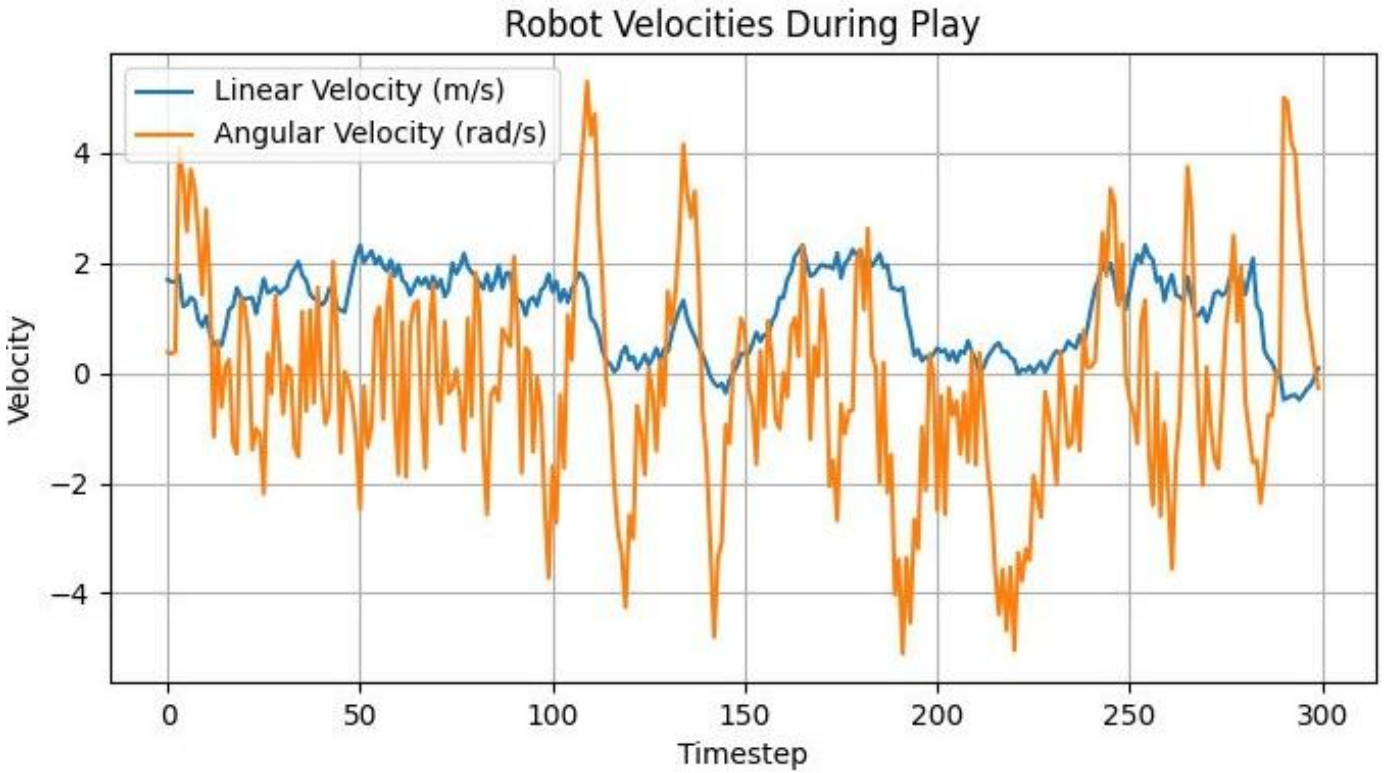


Fig.16 : Robot Velocities.

Left and Right Wheel Speeds while Testing

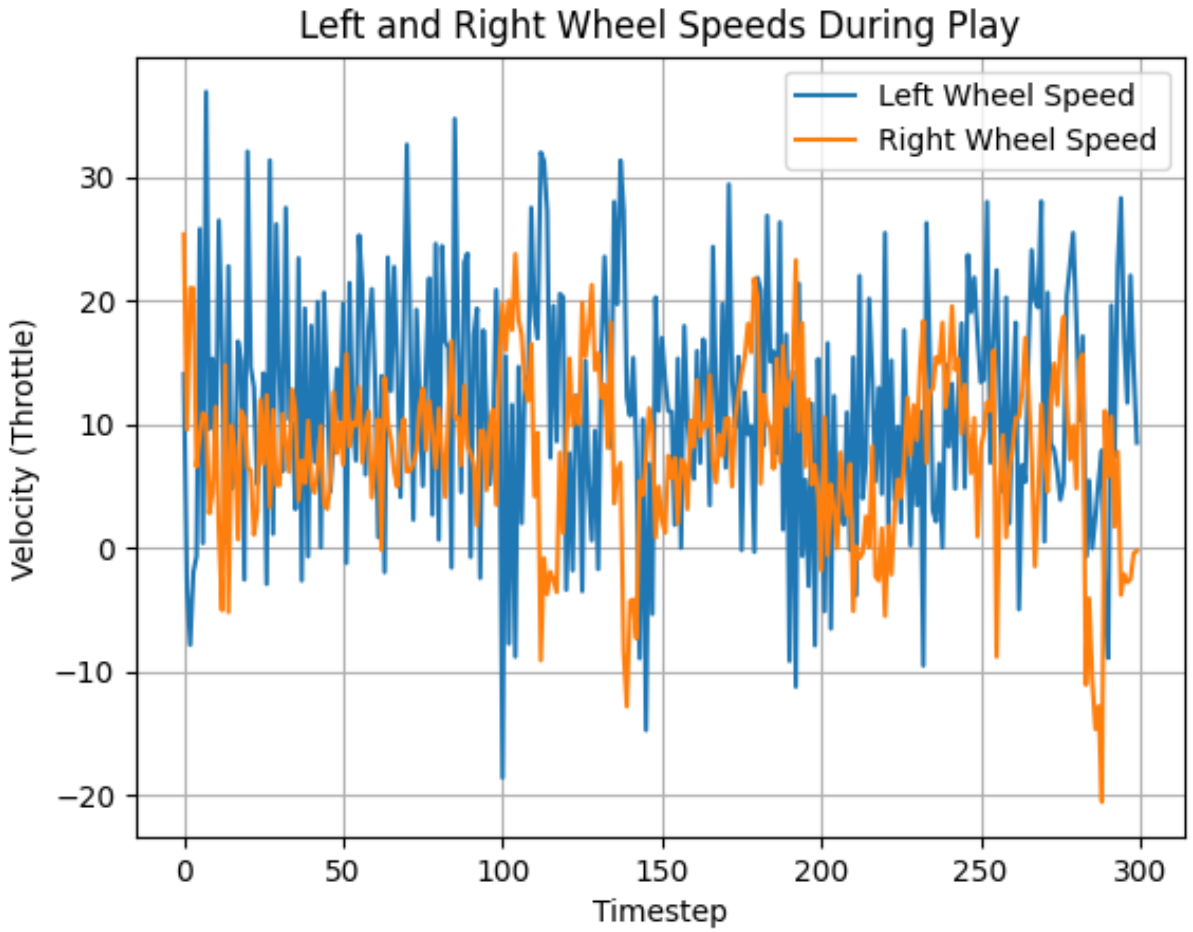
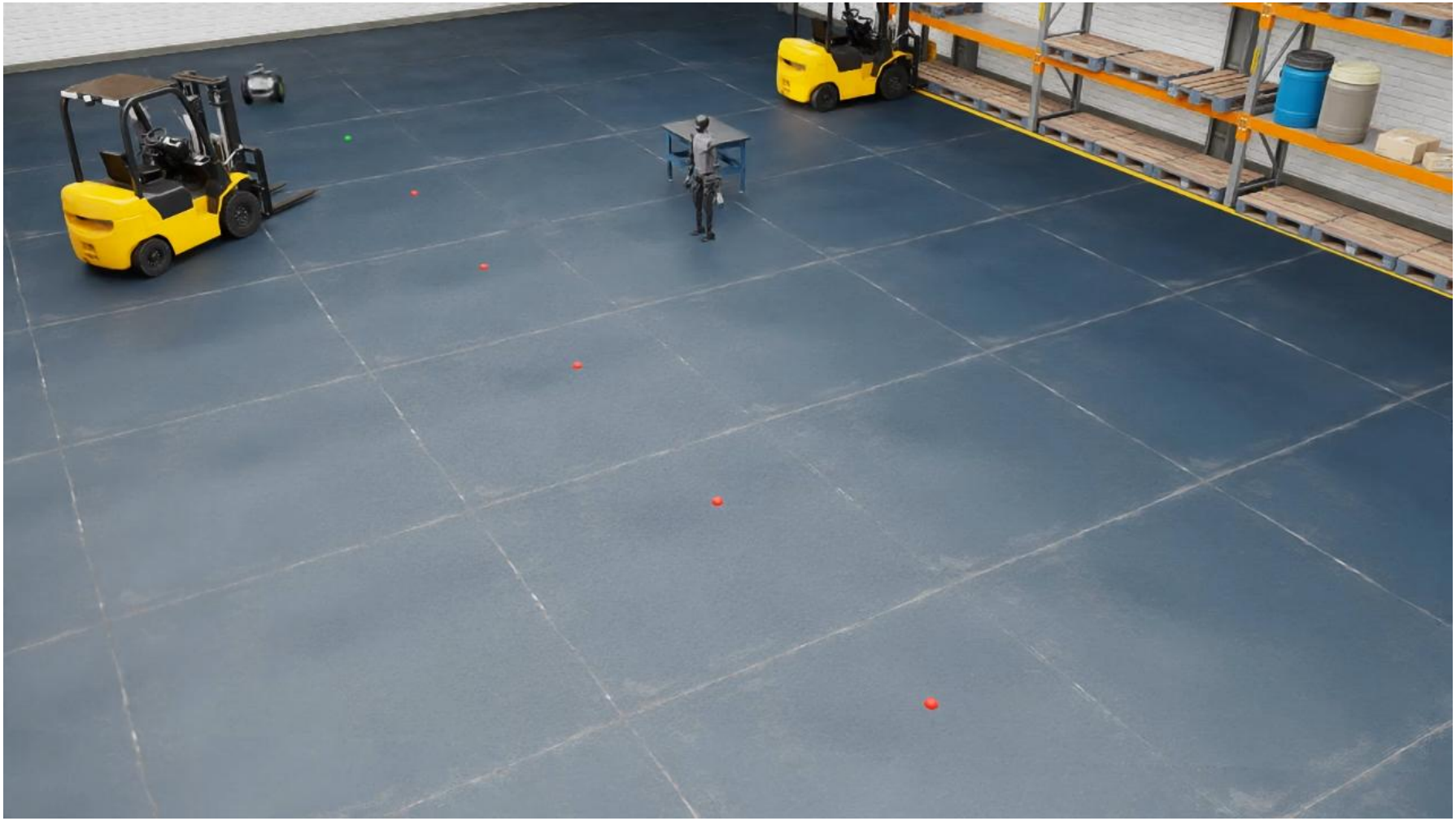


Fig. 17: Speed of Wheels.

Testing the Robot with trained Agent



1 Introduction

2 Methodology

3 Results and Discussions

- Training Results
- Testing Results
- Discussions

4 Conclusions

5 References

Discussions

- Fluctuations in the graph of the velocity possibly caused by imperfect state representations.
- Partially observation (low-quality perception).
- These could be addressed by tuning hyperparameters or improving sensory inputs.

Conclusions

- Both critic and actor networks converged as expected.
- The agent learned an effective policy for navigating toward waypoints.
- Successfully implemented the TD3 algorithm for local path planning in a simulated Carter robot.

Future Work

- Incorporating with richer sensory data.
- Incorporating with additional sensor (e.g. Depth camera).
- Testing sim-to-real transfer.
- Compare TD3 performance with alternative algorithms (SAC or DDPG).

References

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez,Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning.," in ICLR (Y. Bengio and Y. LeCun, eds.), 2016.

S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods.," CoRR,vol. abs/1802.09477, 2018.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. A. (2014). Deterministic policy gradient algorithms. In Icml (pp. 387–395, Vol. 32). JMLR.org. http ://dblp.unitrier.de/db/conf/icml/icml2014.html#SilverLHDWR14.

Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction (Second Edition). The MIT Press. Retrieved from http://incompleteideas.net/book/the-book-2nd.html.


THANK
YOU

Appendix A: Robot Specifications

Parameter	Value
Overall Length	0.52 m
Width	0.38 m
Height	0.48 m
Mass	11.5 kg
Wheel Radius	0.09 m
Wheel Base	0.28 m

Appendix B: TD3 Algorithm pseudocode

Algorithm 3: TD3 algorithm

Initialize critic networks $Q(s,a|\theta_1^Q)$, $Q(s,a|\theta_2^Q)$, and actor network $\mu(s|\theta^\mu)$ with random parameters θ_1^Q, θ_2^Q, θ^μ

Initialize target network $Q'(s,a|\theta_1^{Q'})$, $Q'(s,a|\theta_2^{Q'})$ and μ' with weights $\theta_1^{Q'} \leftarrow \theta_1^Q$, $\theta_2^{Q'} \leftarrow \theta_2^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$

Initialize replay buffer $\mathcal{D}$

for $t = 1$ **to** T **do**

Select action $a = \mu(s|\theta^\mu) + \epsilon$ according to the current policy and exploration noise

$\epsilon \sim \mathcal{N}(0,\sigma)$ and observe reward r and new state s

Store transition tuple (s,a,r,s') in $\mathcal{D}$

Sample mini-batch of N transition (s,a,r,s') from $\mathcal{D}$

$\tilde{a} \leftarrow \mu'(s'|\theta^{\mu'}) + \epsilon$, $\epsilon \sim \text{clip}(\mathcal{N}(0,\tilde{\sigma}), -c, c)$

$y \leftarrow r + \gamma \min_{i=1,2} Q'(s', \tilde{a}|\theta_i^{Q'})$

Update critics $\theta_i^Q \leftarrow \text{argmin}_{\theta^{Q_i}} N^{-1} \sum \left(y - \left(Q(s,a|\theta_i^Q)\right)\right)^2$

if t mod d **then**

Update θ^μ by the deterministic policy gradient:

$\nabla_{\theta^\mu} J(\theta^\mu) = N^{-1} \sum \nabla_a Q(s,a|\theta_1^Q)|_{a=\mu(s|\theta^\mu)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)$

Update target networks:

$\theta_i^{Q'} \leftarrow \tau\theta_i^Q + (1-\tau)\theta_i^{Q'}$

$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$

end if

end if

Appendix C: Hyperparameters

Parameters	Values
Maximum Training Steps Per Episode	40
Replay Buffer Size	1000000
Batch Size	100
Learning Rate of Actor Network	0.0001
Learning Rate of Critic Network	0.001
Discount Factor	0.99
Exploration Noise	$\mathcal{N}(0,0.1)$
Target Policy Smoothing Noise	$\epsilon \sim clip(\mathcal{N}(0,0.2), -0.5, 0.5)$

Appendix D: Python Code

https://683fe272147a81e4abf50e18--lustrous-pastelito-c51e85.netlify.app/