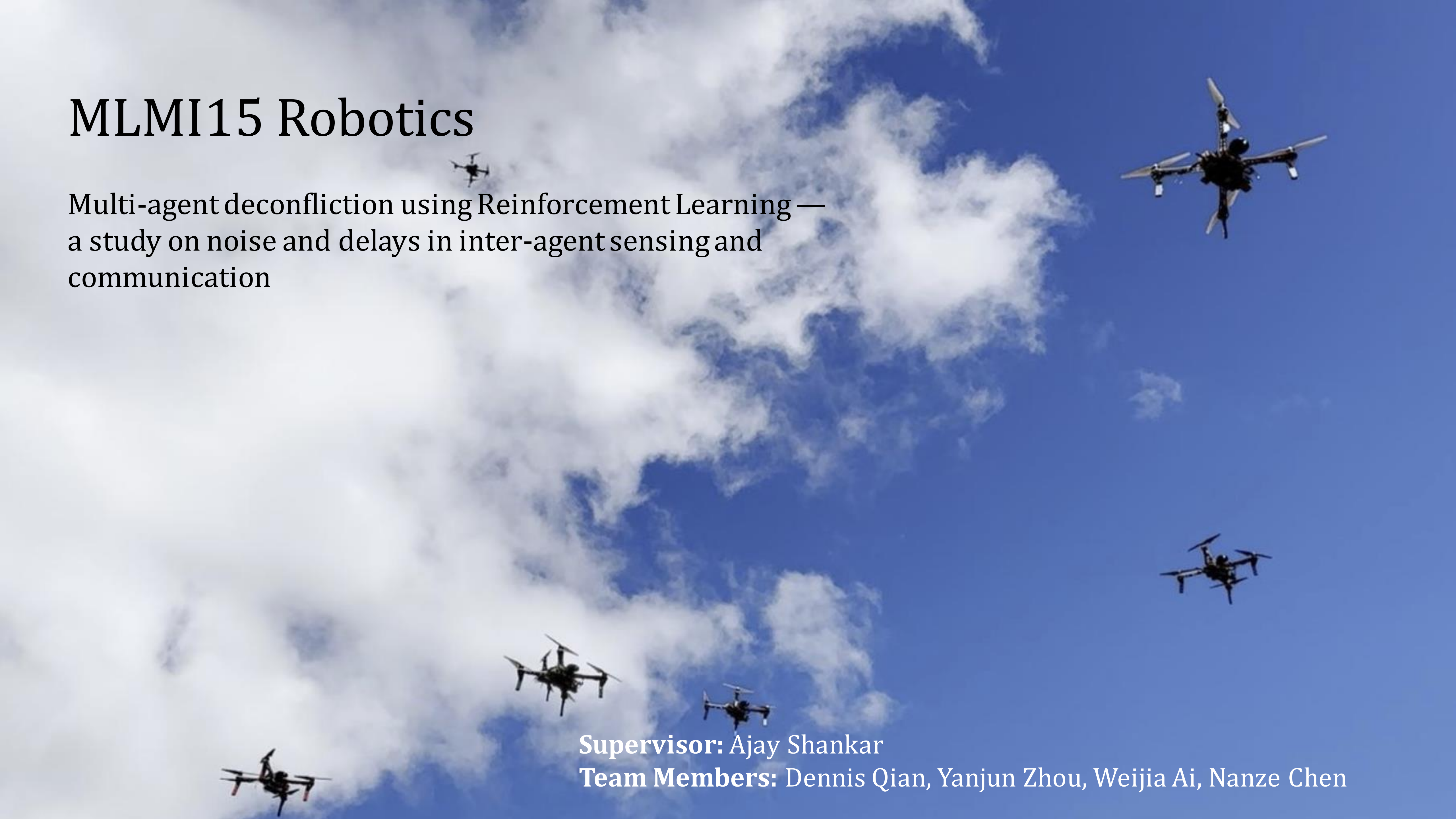


MLMI15 Robotics

Multi-agent deconfliction using Reinforcement Learning — a study on noise and delays in inter-agent sensing and communication

Supervisor: Ajay Shankar
Team Members: Dennis Qian, Yanjun Zhou, Weijia Ai, Nanze Chen

PROJECT INTRODUCTION

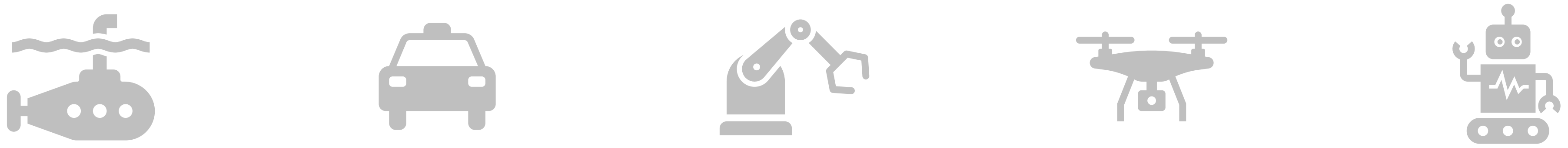
Multi-agent deconfliction using Reinforcement Learning — a study on noise and delays in inter-agent sensing and communication

CHALLENGES

Multi-agent systems face ubiquitous challenges of noise and communication delays in real-world environments. These imperfections, undermine the potential of autonomous agents, leading to **inefficiencies** and **safety hazards**.

OBJECTIVE

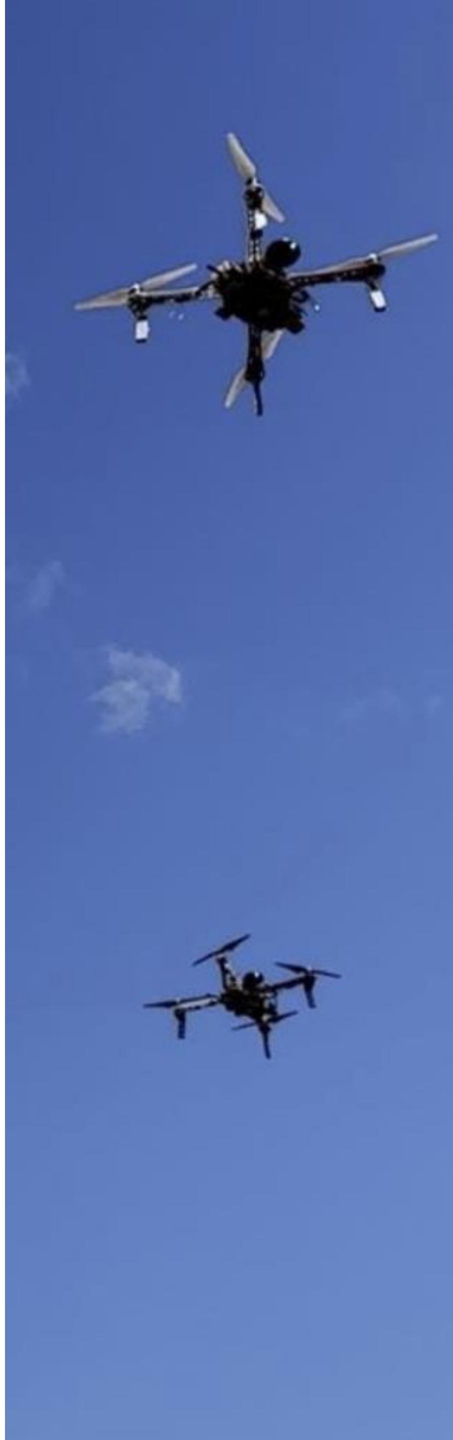
Examine, analyze, and improve the robustness and performance of **reinforcement learning multi-agent** navigation and deconfliction systems in environments with various types of sensory **noise** and **delay**.



PROJECT MILESTONES & CONTRIBUTIONS

Multi-agent deconfliction using Reinforcement Learning — a study on noise and delays in inter-agent sensing and communication

- ☑ Developed a MATLAB environment from scratch that simulations navigation and obstacle avoidance
- ☑ Trained DDPG agents in MATLAB and evaluated performance
- ☑ Trained PPO agents in PyTorch VMAS
- ☑ Rigorously analysed the model robustness against various types of noise and delay
- ☑ Proposed strategies to mitigate the effect of noise and delay



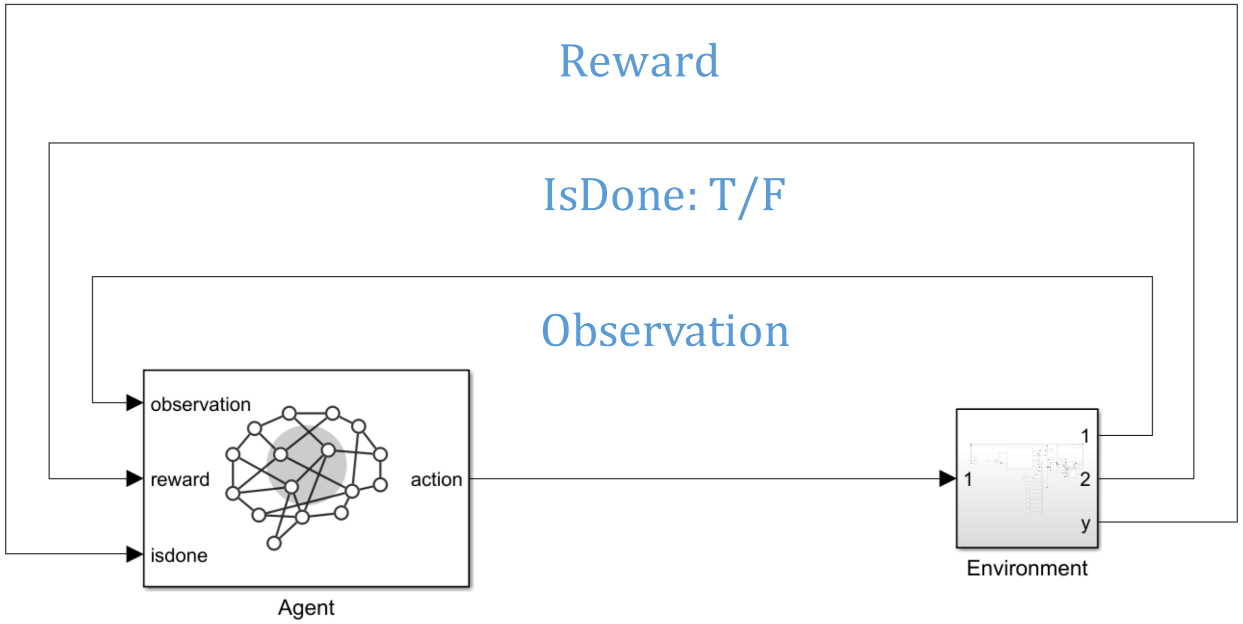
DDPG REINFORCEMENT LEARNING AGENT IN MATLAB

The Deep Deterministic Policy Gradient (DDPG) agent

- Combines the strengths of Deep Q-Learning and Policy Gradient methods
- Produce action (velocities) that maximises the reward

Vehicle environment output

- Model Observation: Vehicle system states and destination coordinates
- Isdone: If reached destination
- Reward: Given at the end of each episode



DDPG REINFORCEMENT LEARNING AGENT IN MATLAB

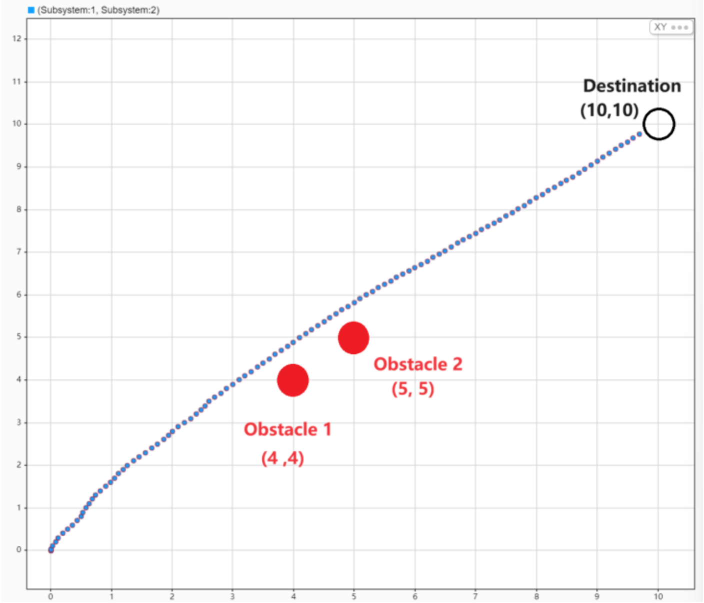
OBSTACLE AVOIDANCE

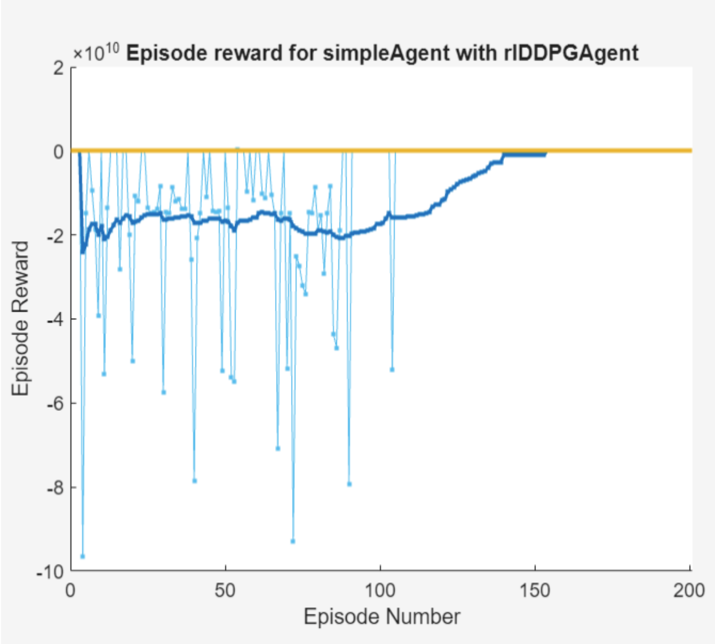
Observations

- x,y location
- x,y velocity
- Obstacle location
- Distance to destination
- Destination location

Reward Function

- -distance_to_dest^2
- Added penalty for collision & reward for reaching destination



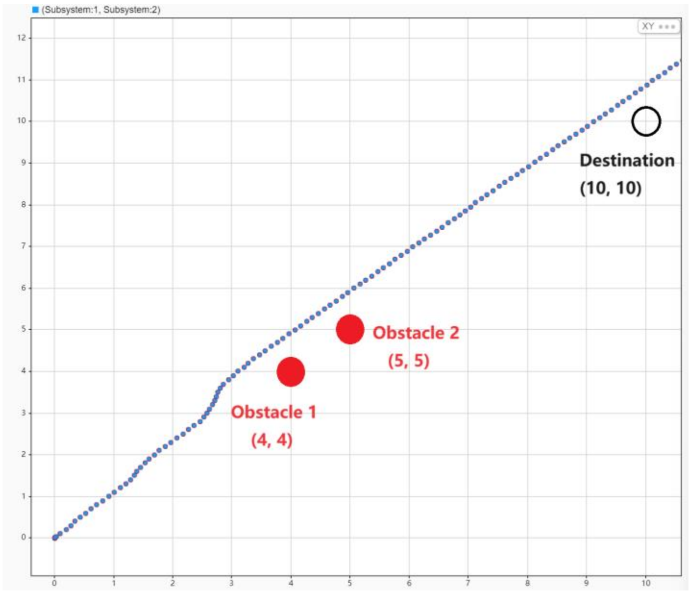


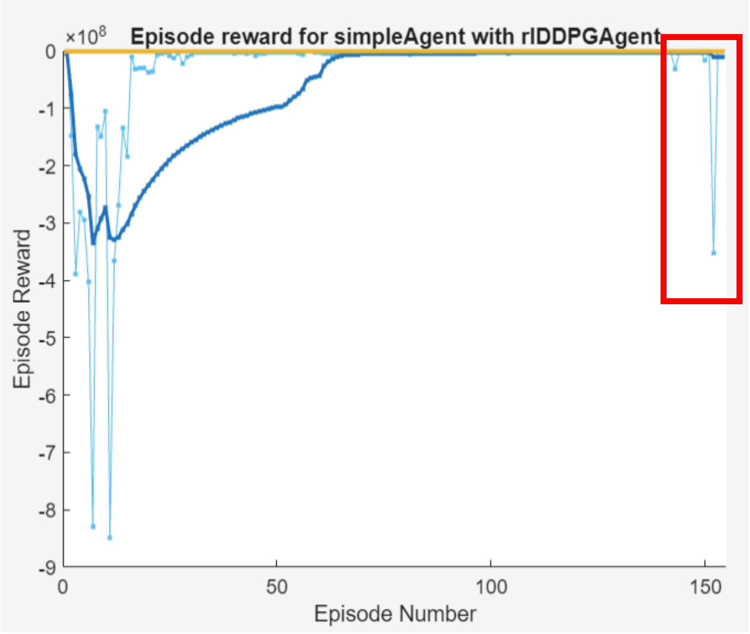
DDPG REINFORCEMENT LEARNING AGENT IN MATLAB

OBSTACLE AVOIDANCE

Drawbacks

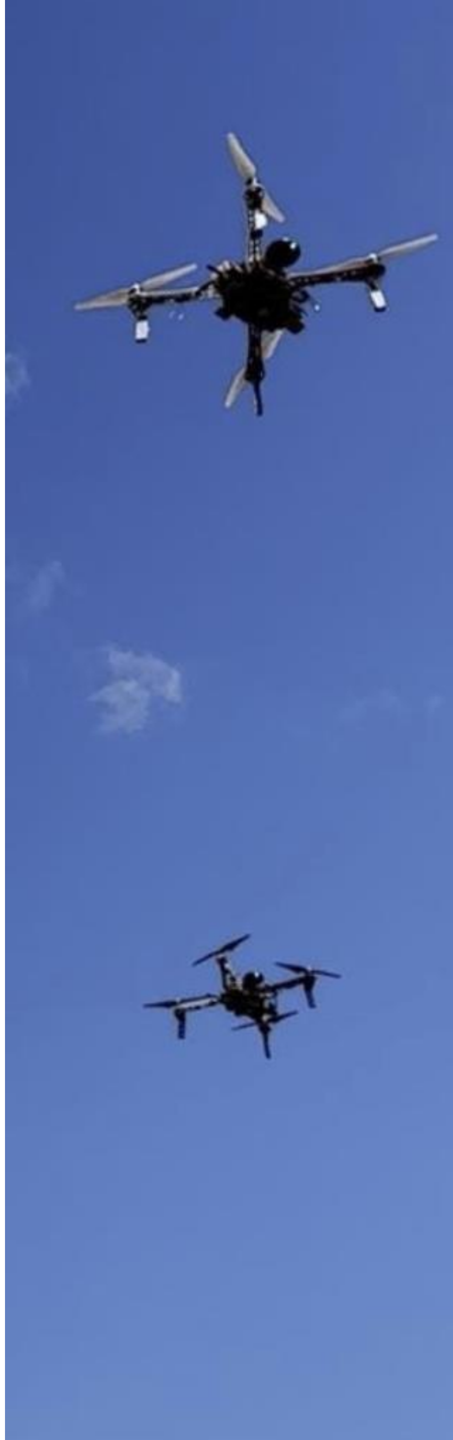
- Unstable (sometimes doesn't converge well)
- Extension to Multi-agent scenario fails due to this
- Takes a very long time to train





PROJECT MILESTONES & CONTRIBUTIONS

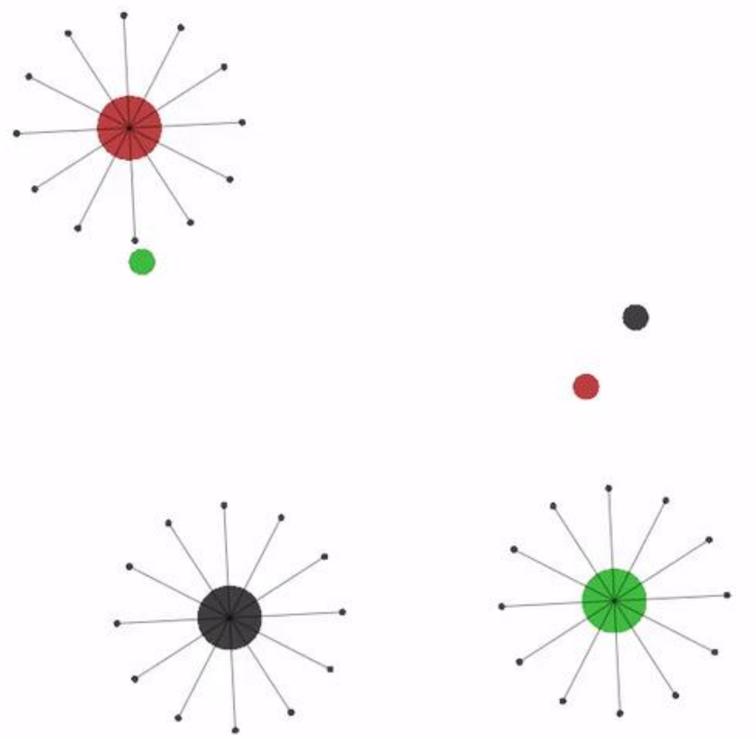
- [x] Developed a MATLAB environment from scratch that simulations navigation and obstacle avoidance
- [x] Trained DDPG agents in MATLAB and evaluated performance
- [x] Trained PPO agents in PyTorch VMAS
- [x] Rigorously analysed the model robustness against various types of noise and delay
- [x] Proposed strategies to mitigate the effect of noise and delay



PPO RL AGENT COMPARISON

	PPO (Proximal Policy Optimization)	DDPG (Deep Deterministic Policy Gradient)
Type	On-Policy	Off-Policy
Stability	Generally more stable and reliable	May be unstable at times

PPO Multi-agent RL Simulator IN PYTORCH



Goal

- Randomly spawned agents need to navigate to their goal.

Observations

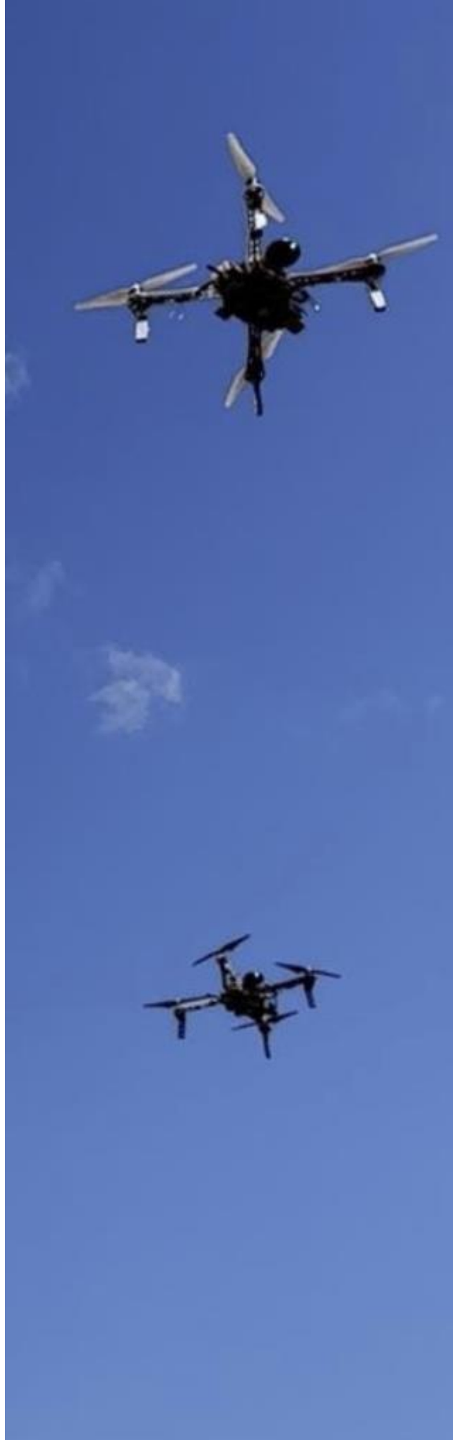
- Position
- Velocity
- Lidar Readings
- Relative position to the destination

Reward

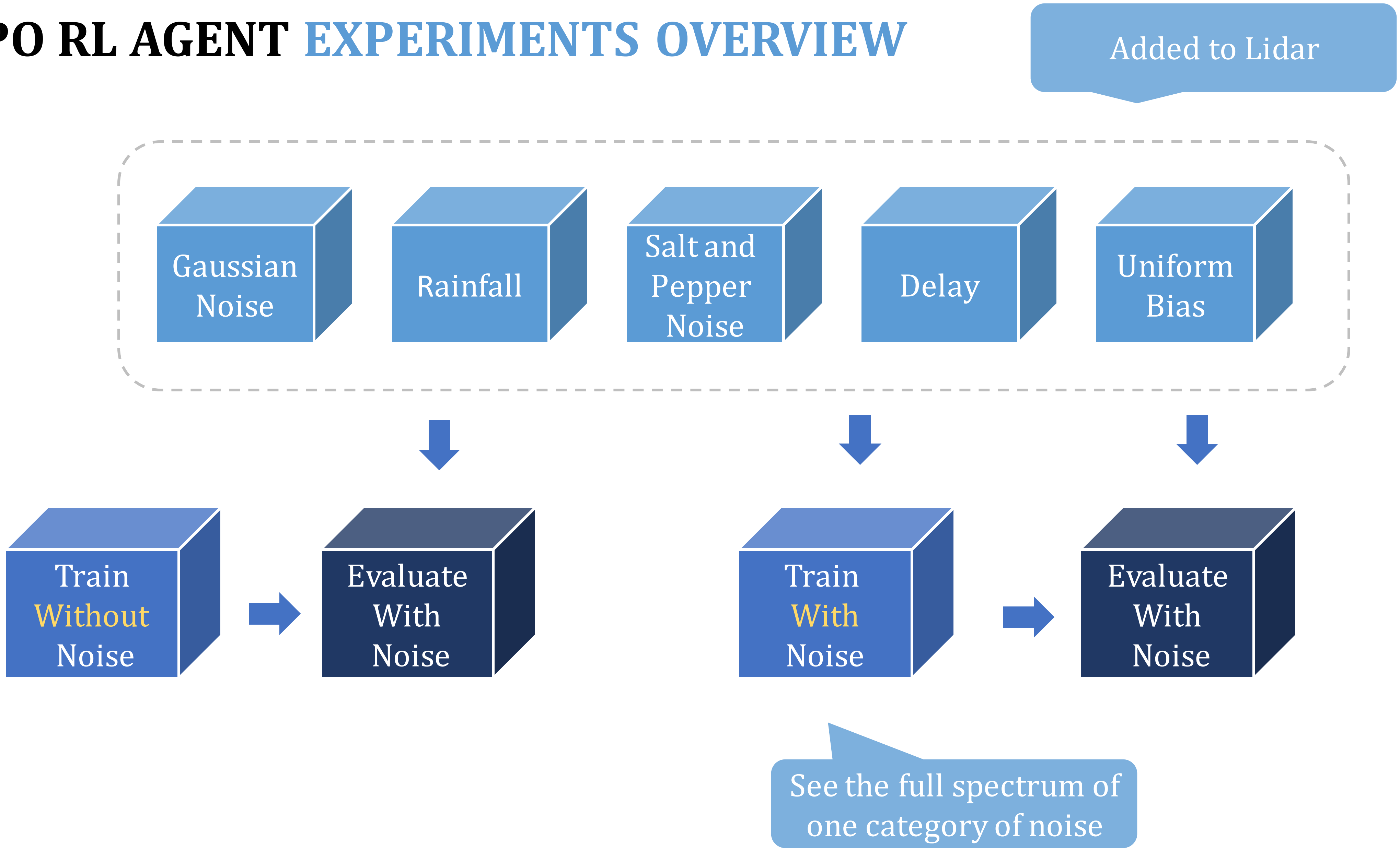
- Collision penalisation
- Relative distance to goal
- Shared reward when all agents reach their goal

PROJECT MILESTONES & CONTRIBUTIONS

- Developed a MATLAB environment from scratch that simulations navigation and obstacle avoidance
- Trained DDPG agents in MATLAB and evaluated performance
- Trained PPO agents in PyTorch VMAS
- Rigorously analysed the model robustness against various types of noise and delay
- Proposed strategies to mitigate the effect of noise and delay



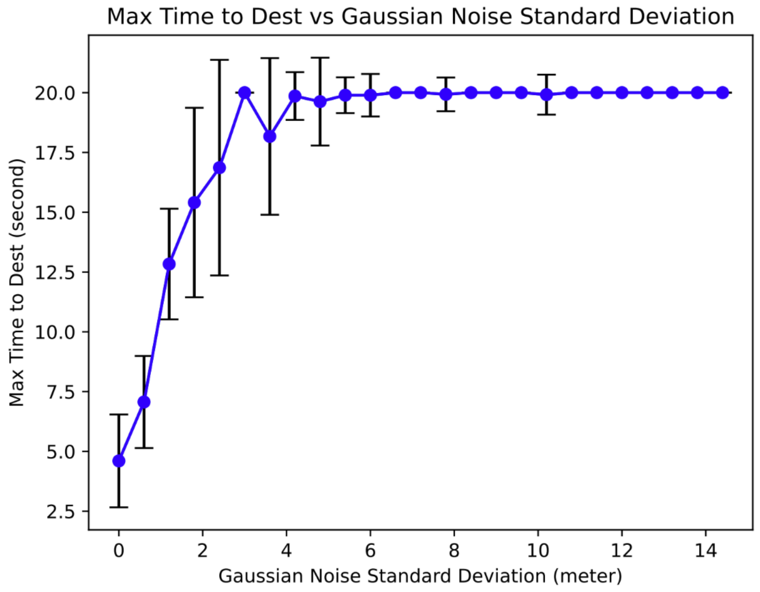
PPO RL AGENT EXPERIMENTS OVERVIEW

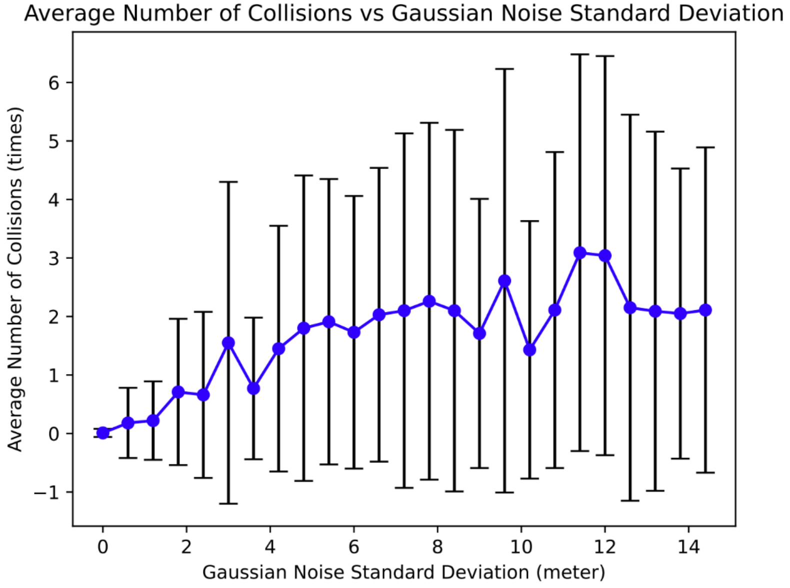


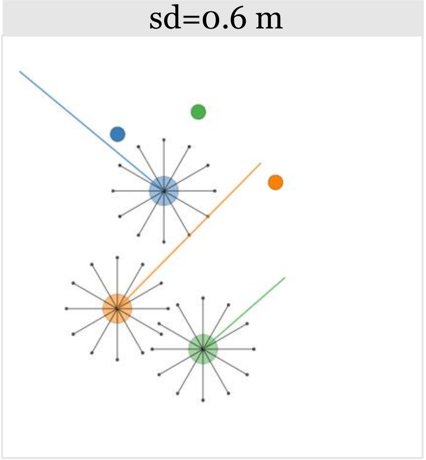
PPO AGENT

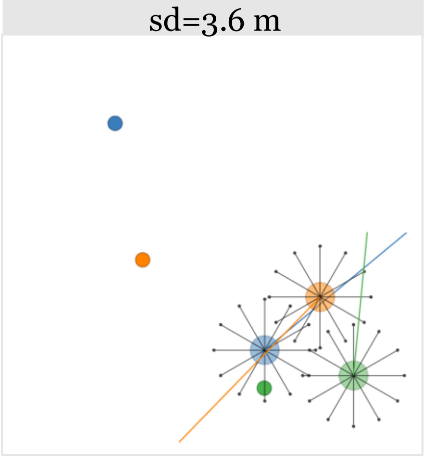
NOISE FREE TRAINING
GAUSSIAN NOISE IN TESTING

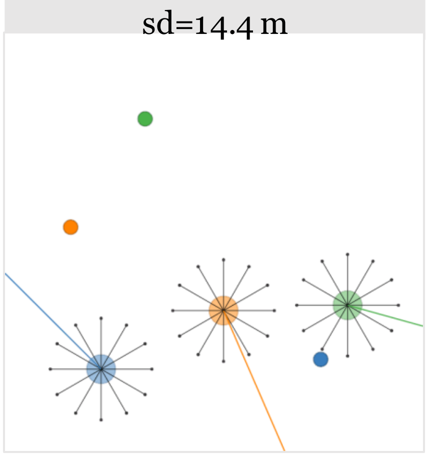
- Lidar Max range: 6m
- Not robust to Gaussian Noise







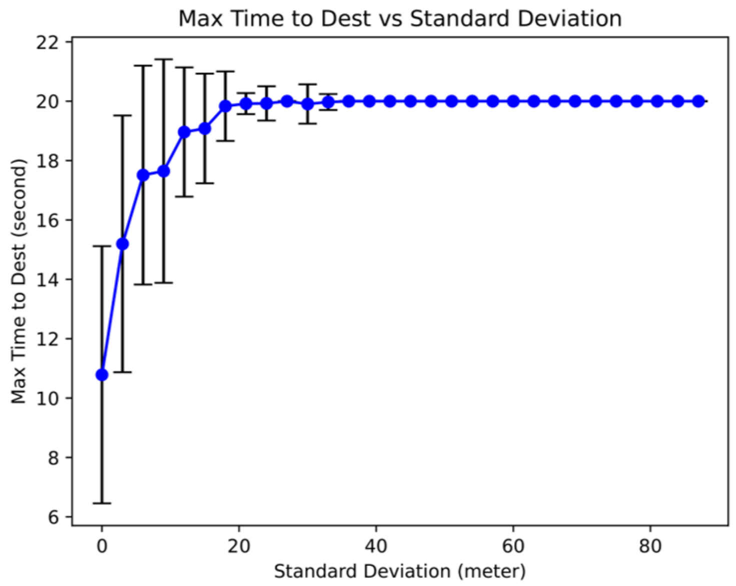


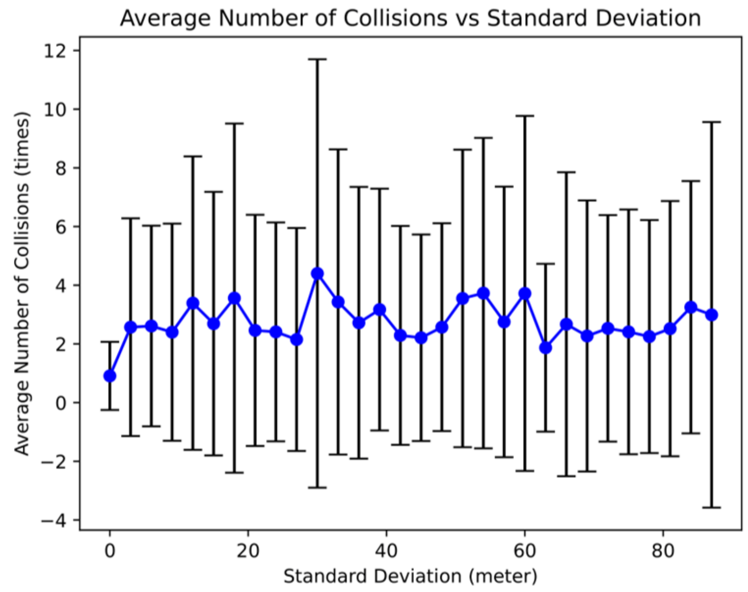


PPO AGENT

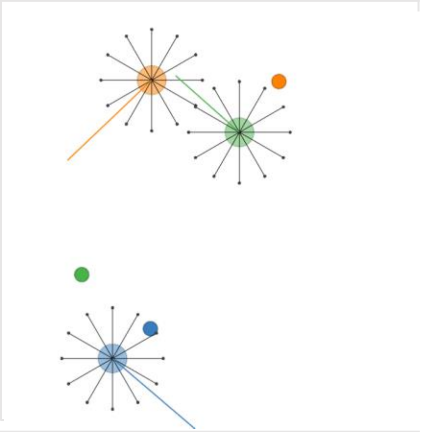
GAUSSIAN NOISE IN TRAINING AND TESTING

- The agent is more robust to Gaussian Noise

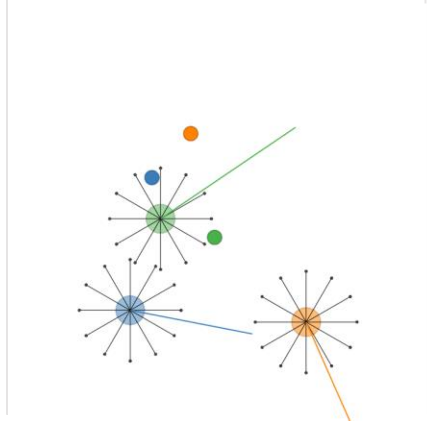




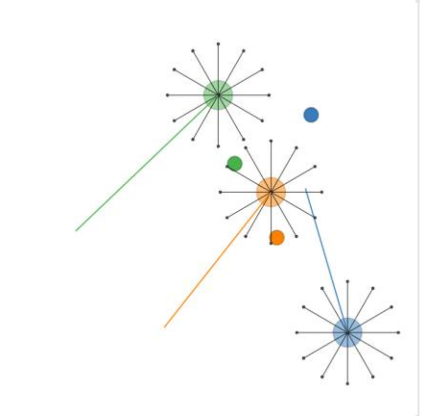
sd = 3 m



sd = 8 m



sd = 24 m



PPO AGENT

NOISE FREE TRAINING

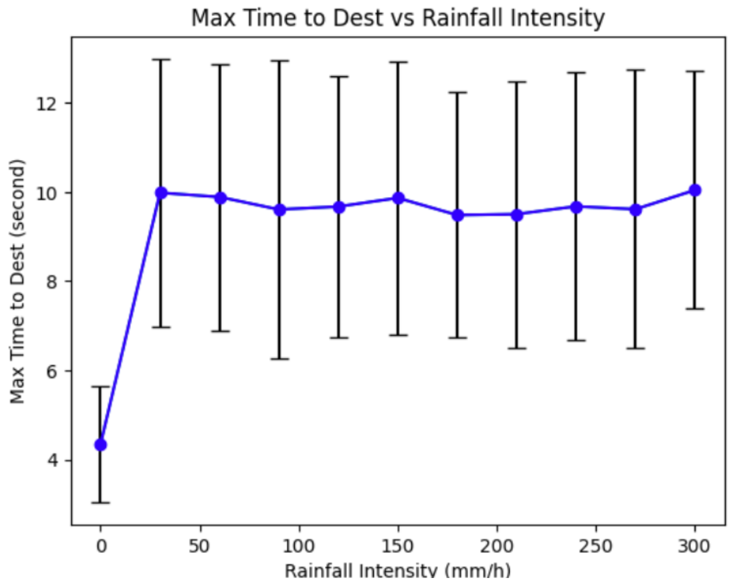
RAINFALL IN TESTING

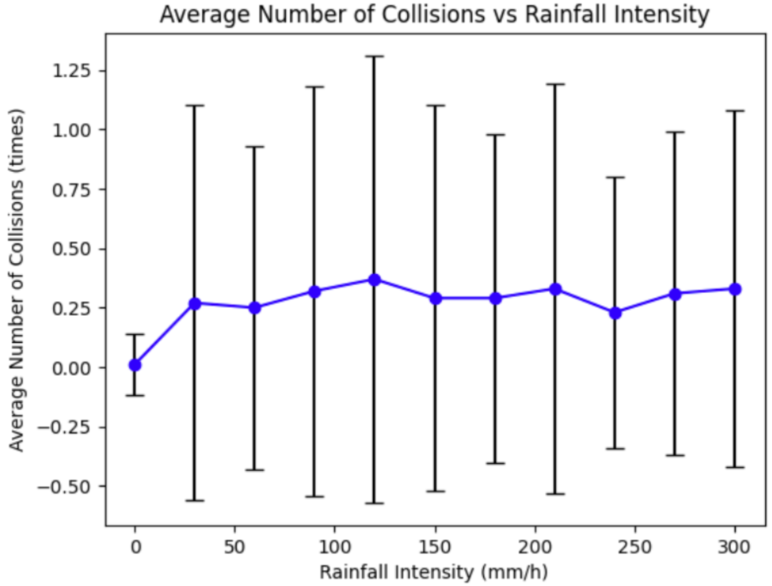
$$z' = z + N(0, 0.02z(1 - e^{-R})^2)\ ^*$$

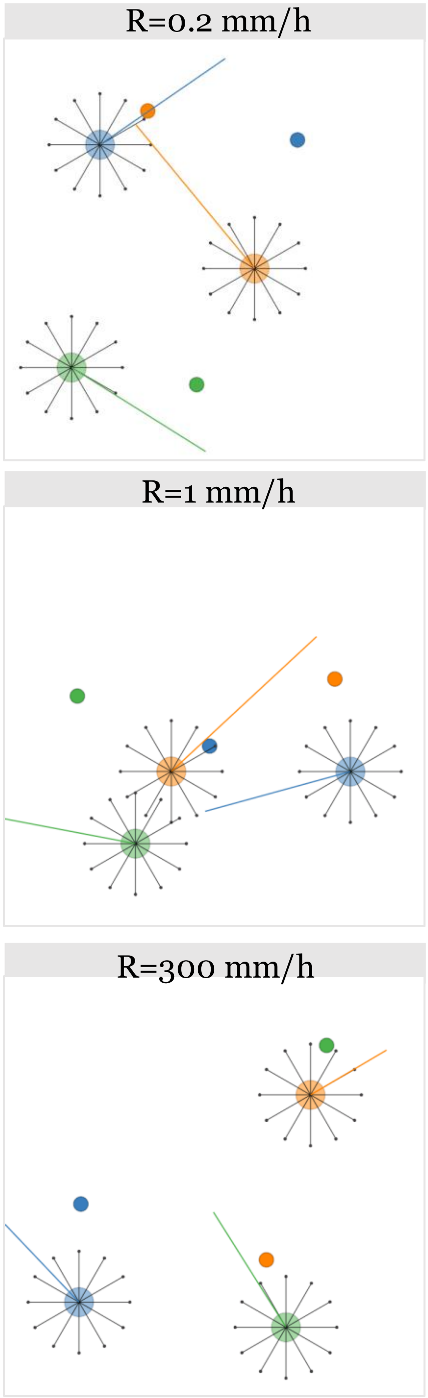
Where z is the Lidar observation
R is the rainfall in mm/h

Is robust to Rainfall

* Espineira, J.P. *et al.* (2021) 'Realistic lidar with noise model for real-time testing of automated vehicles in a virtual environment', *IEEE Sensors Journal*, 21(8), pp. 9919–9926. doi:10.1109/jsen.2021.3059310.



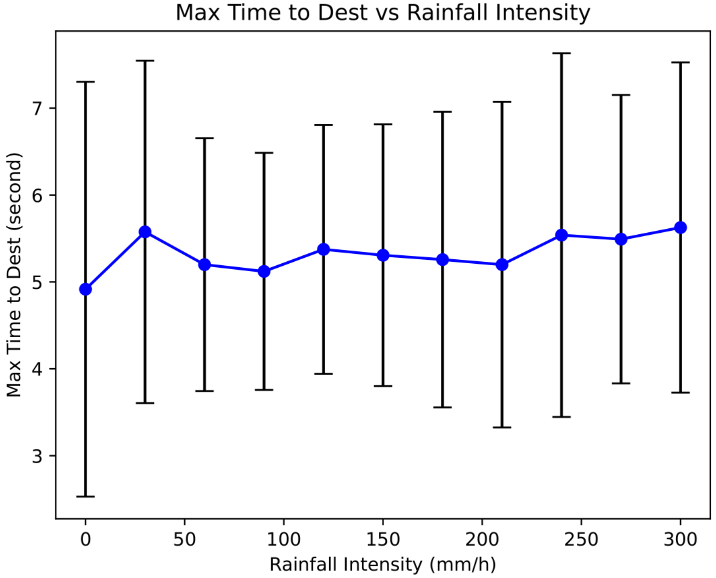


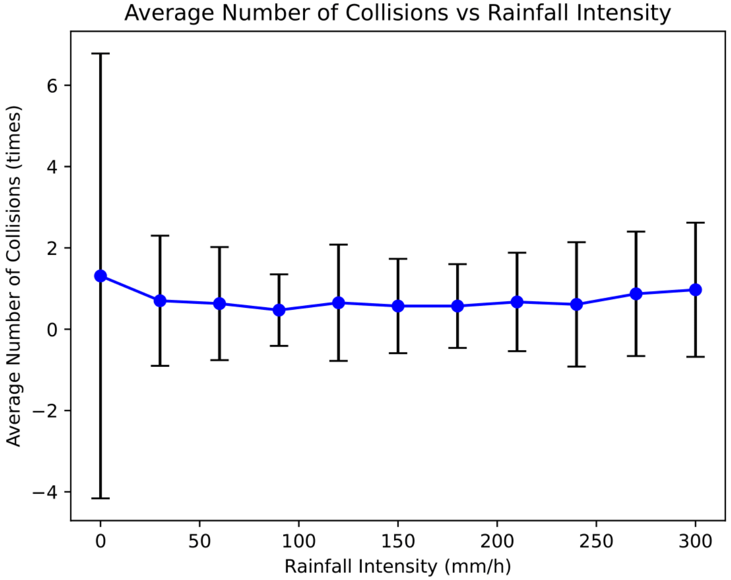


PPO AGENT

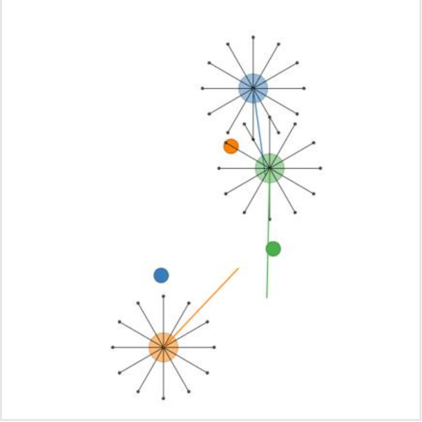
SIMULATED RAINFALL IN TRAINING AND TESTING

- Rainfall noise has a significant effect on long-range radar (automotive radar, range=200m).
- On our simulated robot car (range=2m), even with R = 305 mm/h (world record), there is no large effect.
- According to formula,
  - R = 1 mm/h, std=0.13
  - R = 2mm/h, std=0.17
  - R = 10mm/h, std=0.20
  - R = 300 mm/h, std=0.20

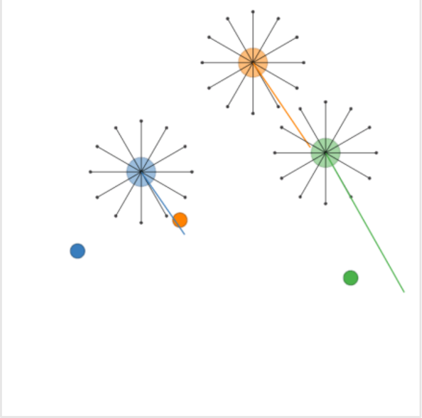




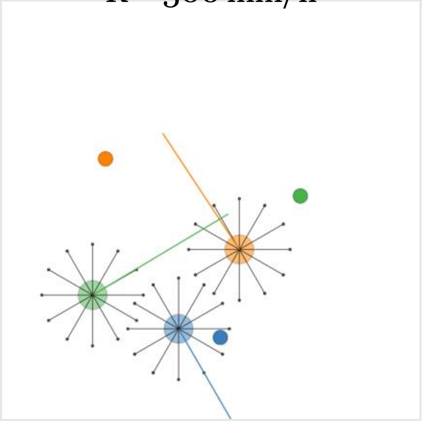
R = 0.2 mm/h



R = 1.0 mm/h



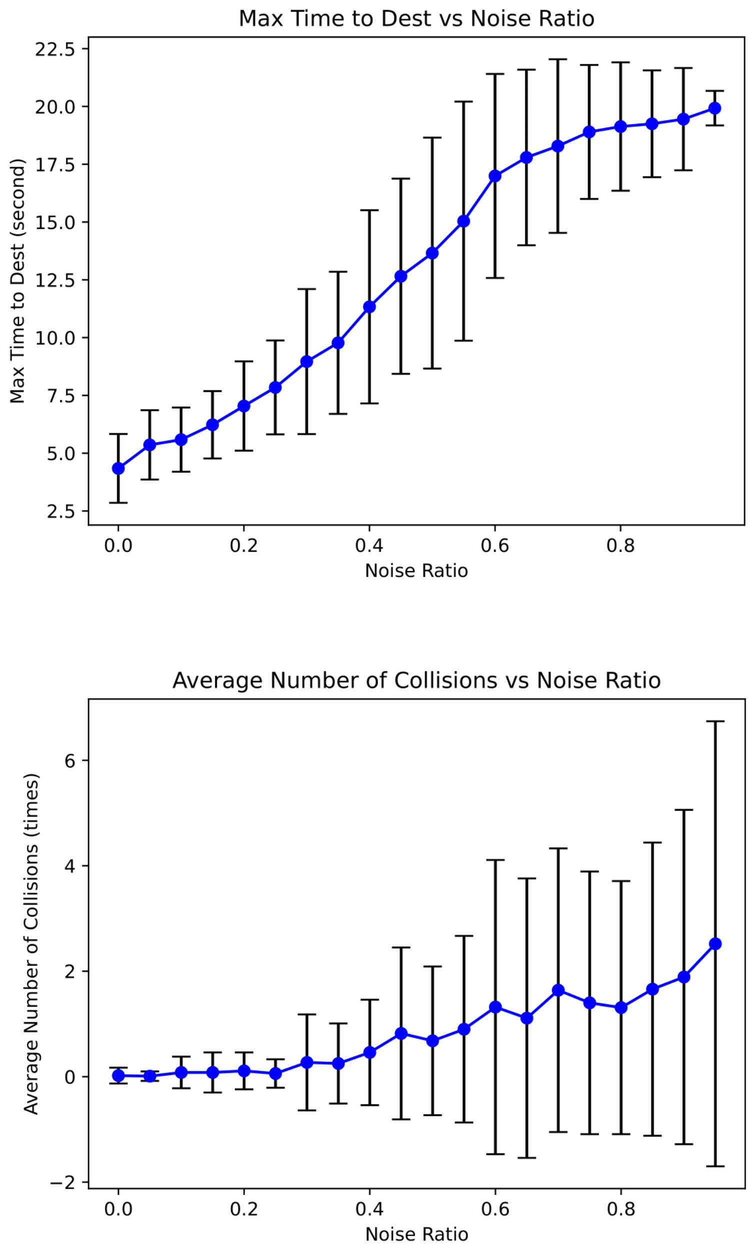
R = 300 mm/h

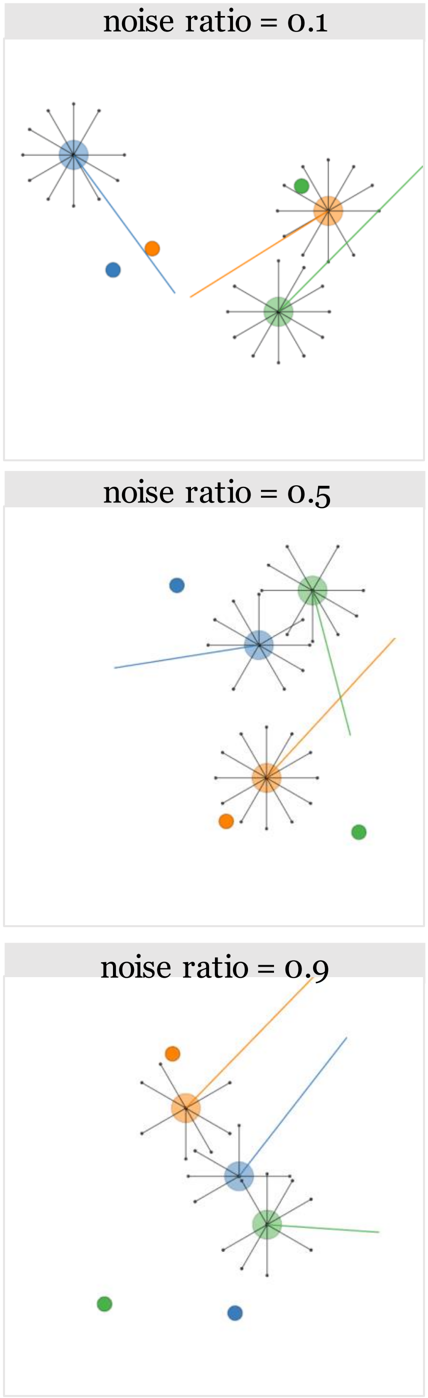


PPO AGENT

NO NOISE & DELAY IN TRAINING, SALT & PEPPER NOISE IN TESTING

- Agent trained without noise is not robust to Salt & Pepper noise.

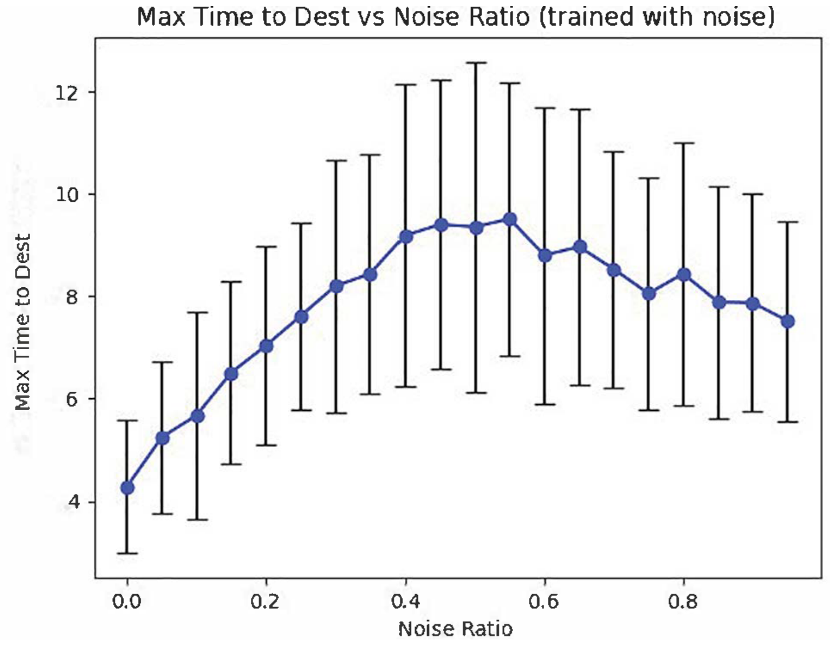


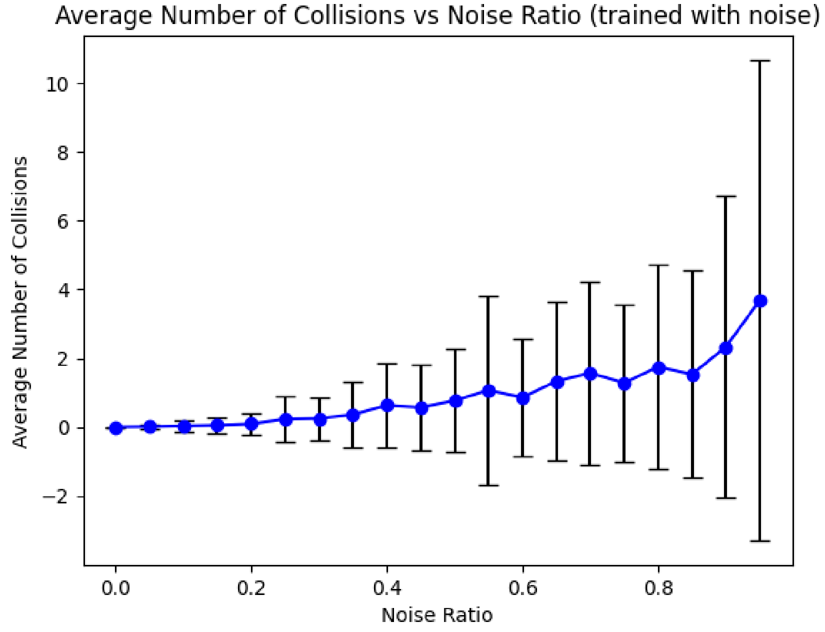


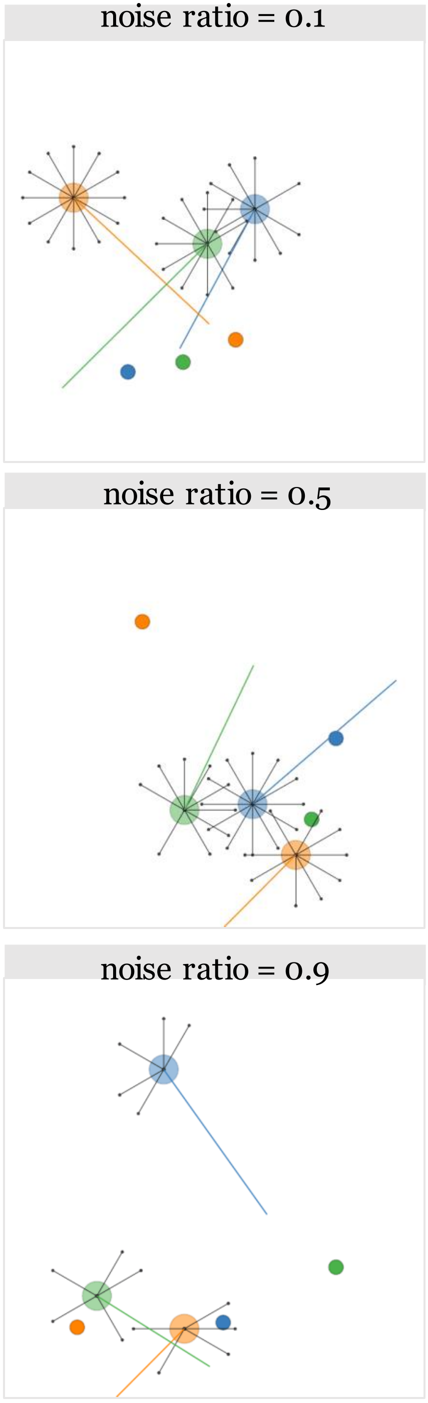
PPO AGENT

SALT AND PEPPER
NOISE IN TRAINING AND TESTING

- Agent is significantly more robust to salt and pepper noise after training with it.



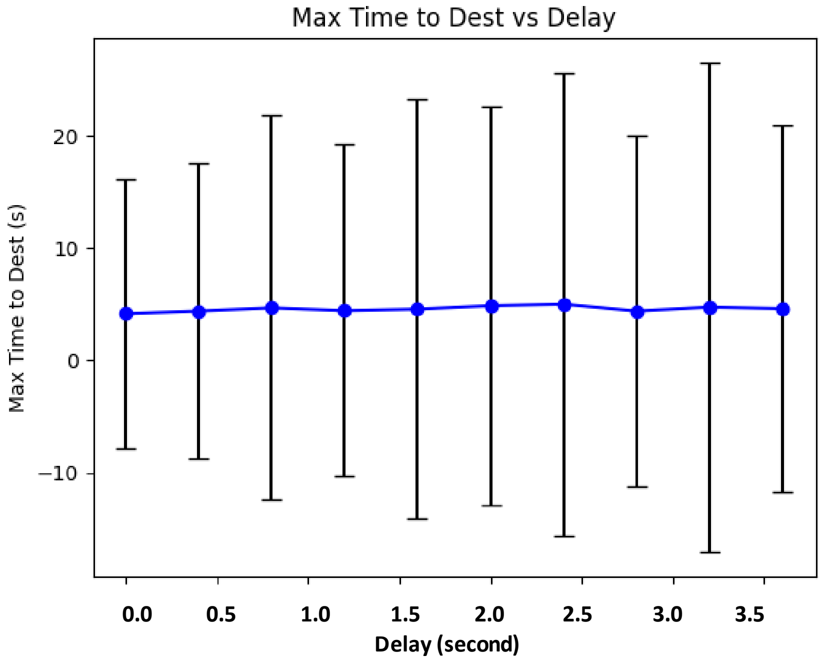


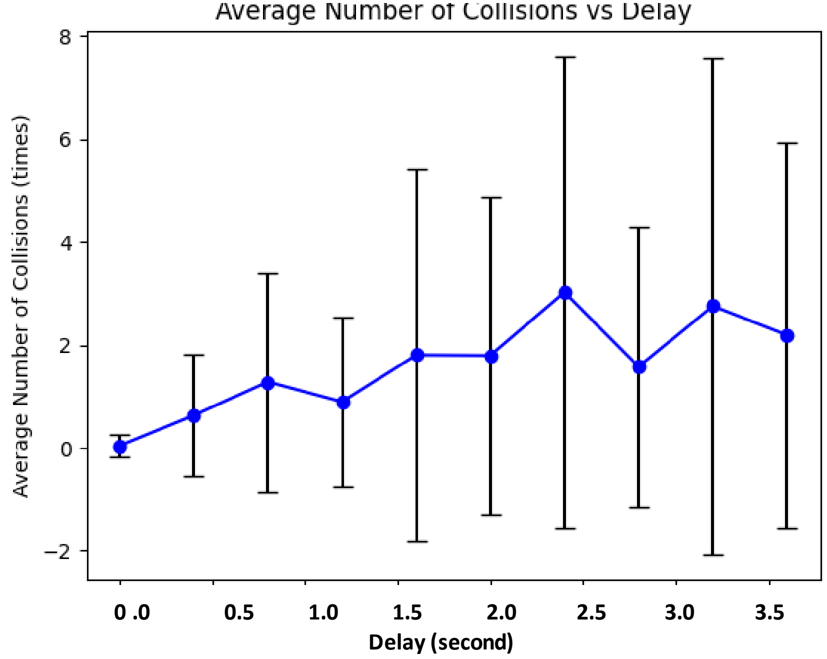


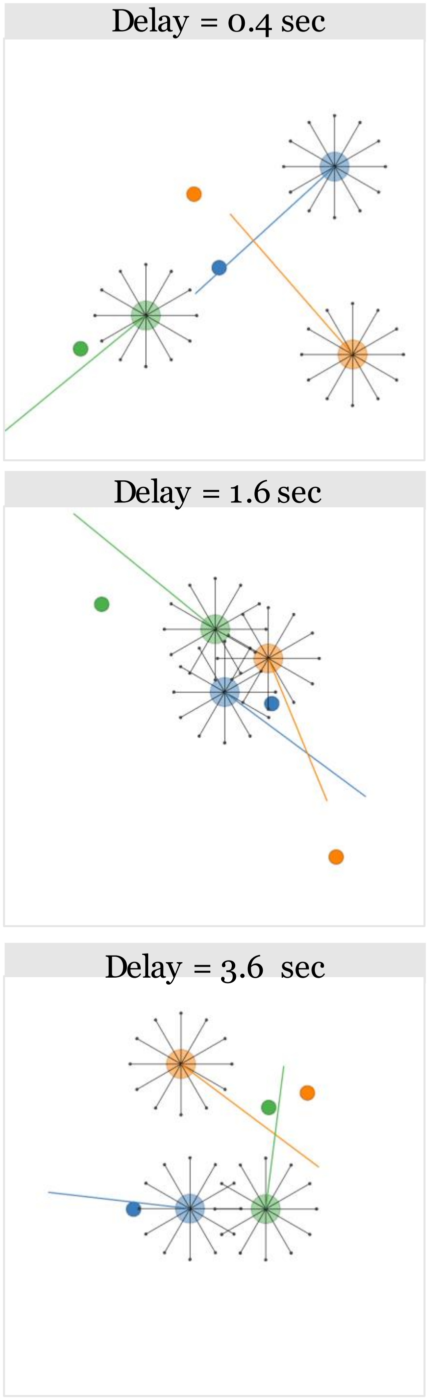
PPO AGENT

NO NOISE & DELAY IN TRAINING, DELAY IN TESTING

- Delay has nearly no effect on the robot's task of reaching the destination but will make it less capable of avoiding collision.



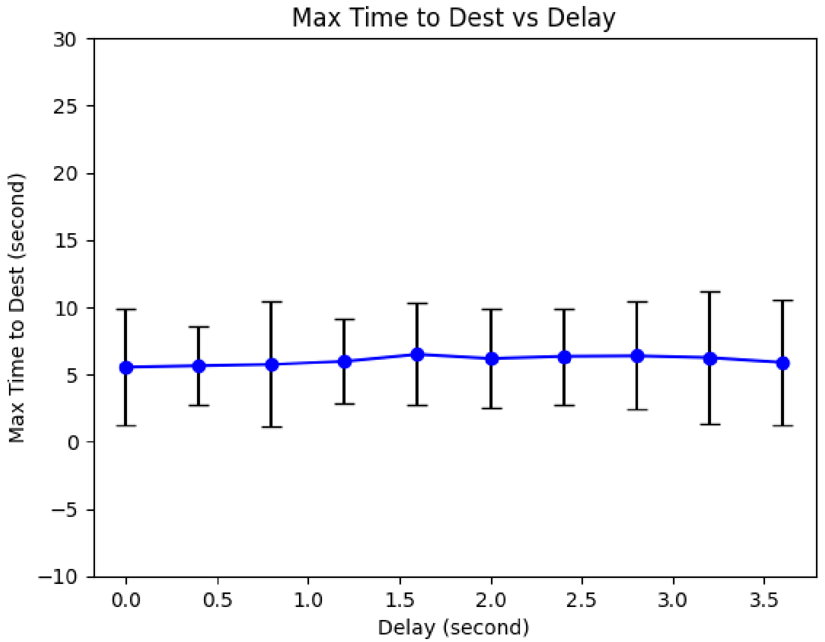


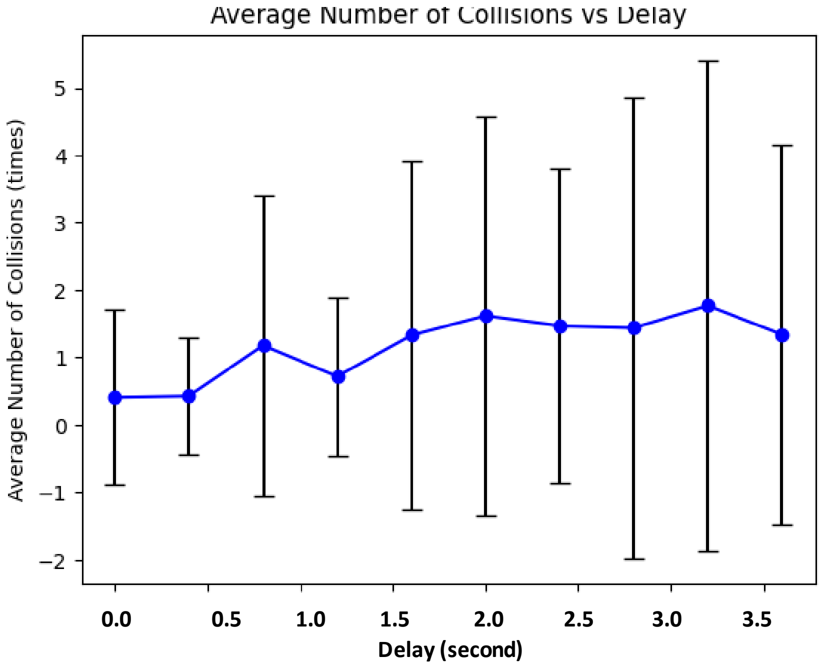


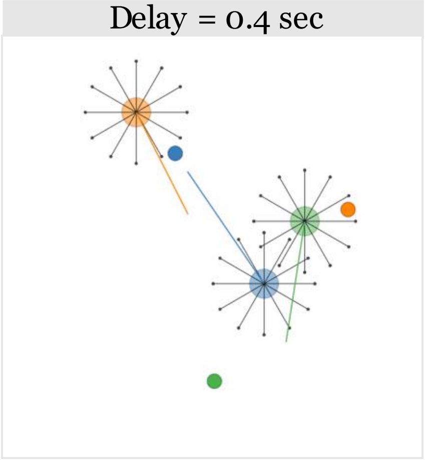
PPO AGENT

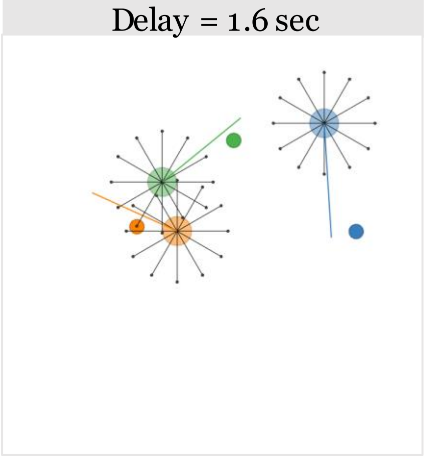
DELAY IN TRAINING AND TESTING

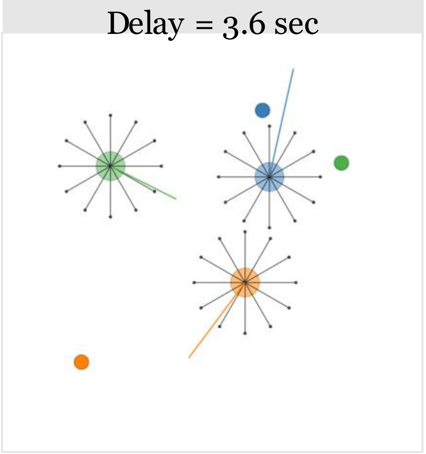
- Training with delay=3 seconds significantly reduce the variance of max time to destination.
- However, the situation of collision doesn't change much.
- Simply training with delay cannot resolve the effect of delay.











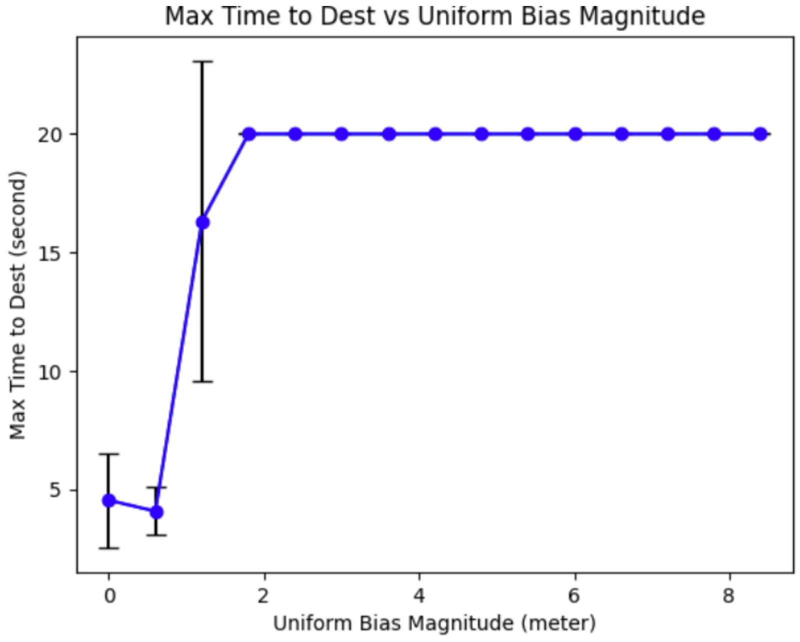
PPO AGENT

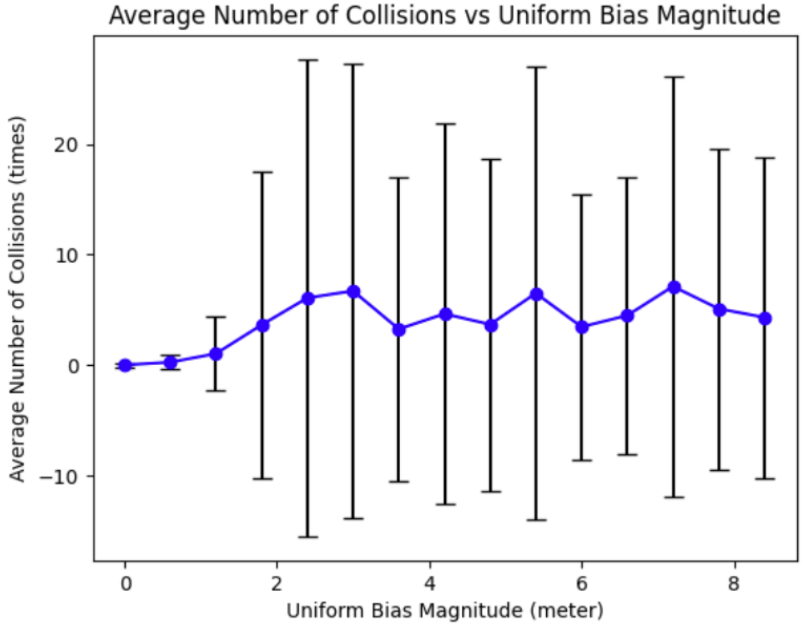
NO NOISE & DELAY IN TRAINING, BIAS IN TESTING

- [X , Y , Vx , Vy , Relative position to the destination, Lidar Readings]

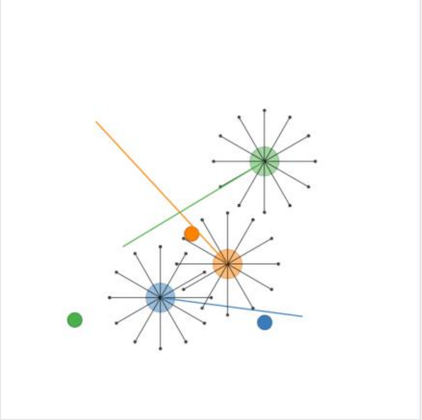
  1×18 vector
- Lidar Readings

  1×12 vector
- The shift of 12/18 observations moves the whole action space
- RL agents are susceptible to bias

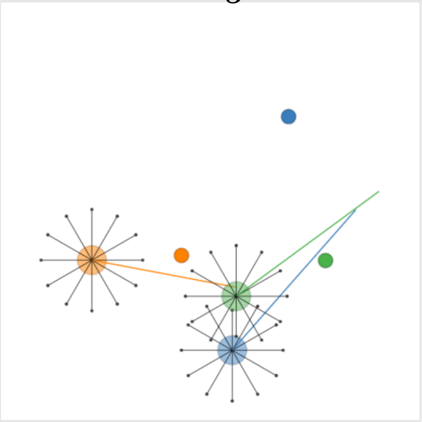




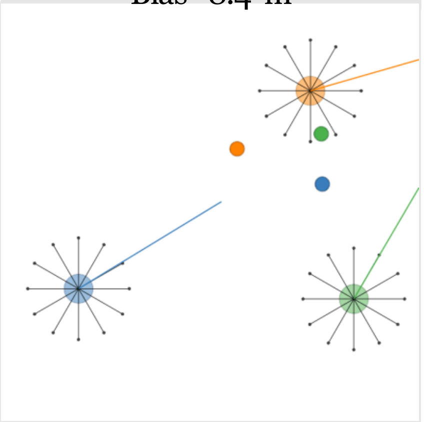
Bias=1.2 m



Bias=3 m



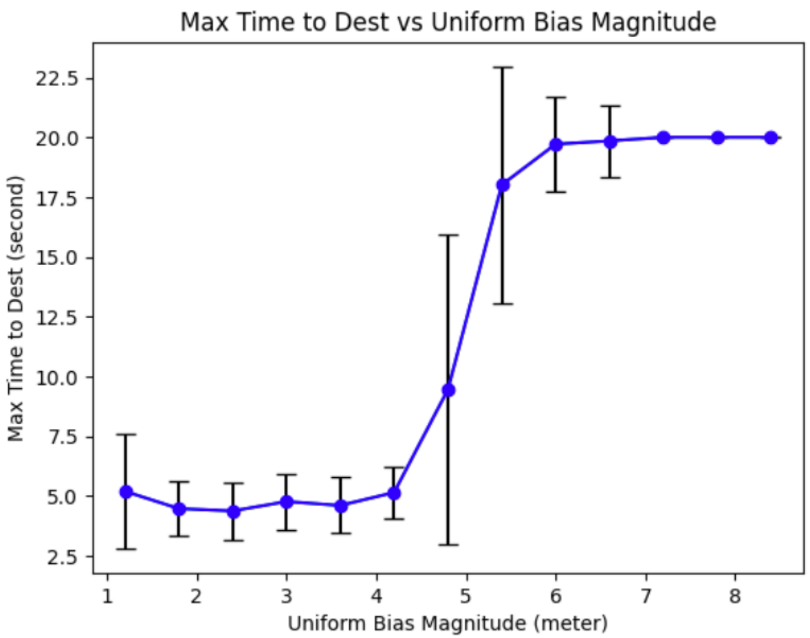
Bias=8.4 m

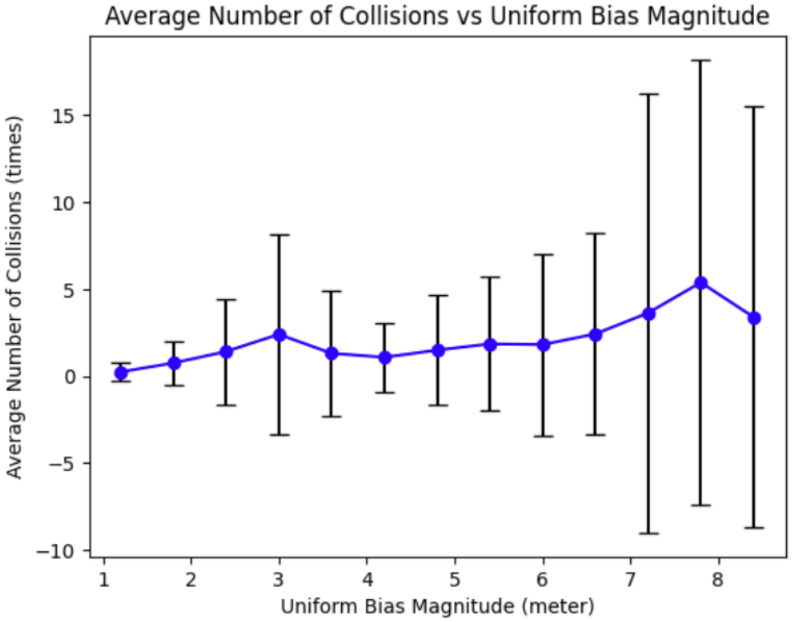


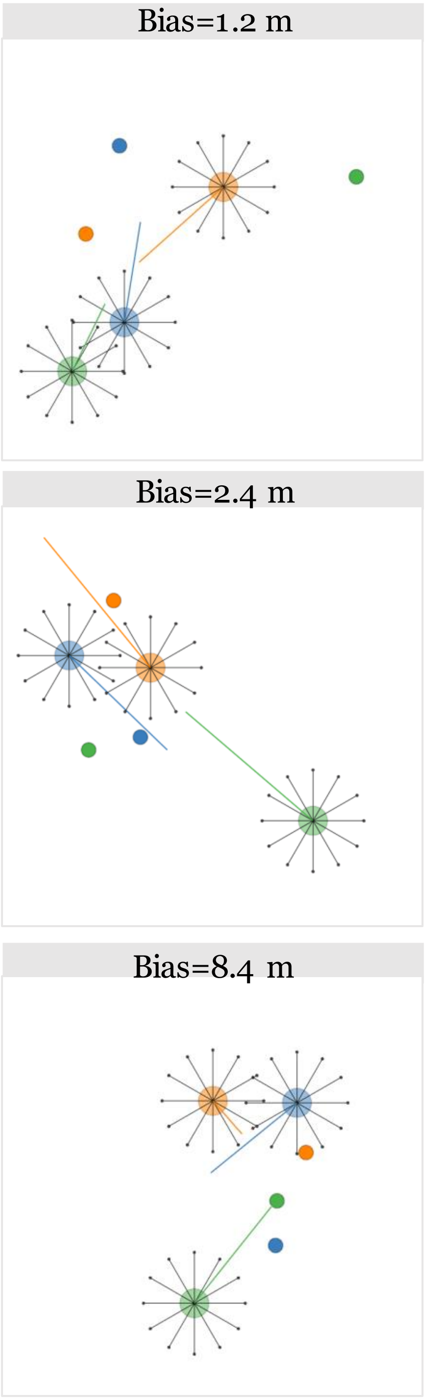
PPO AGENT

BIAS IN TRAINING AND TESTING

- Add bias in training make the system more robust







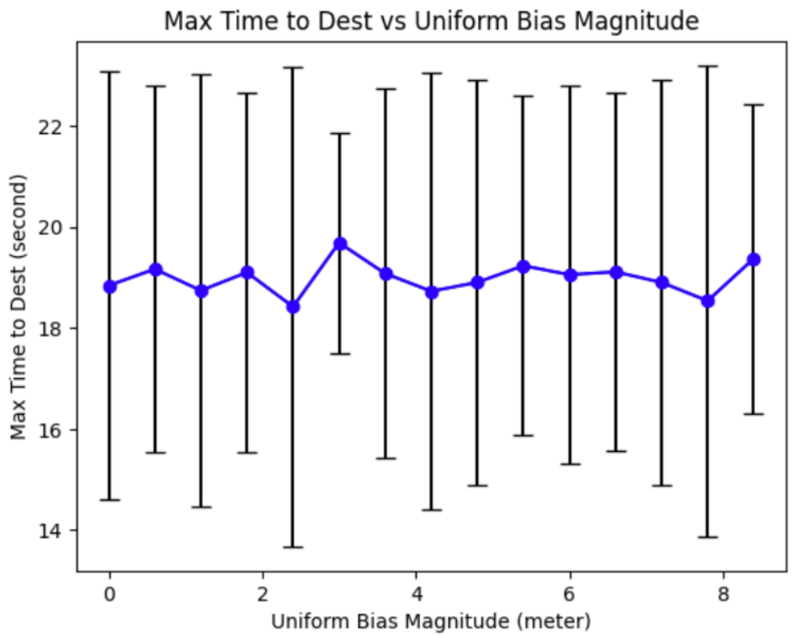
PPO AGENT

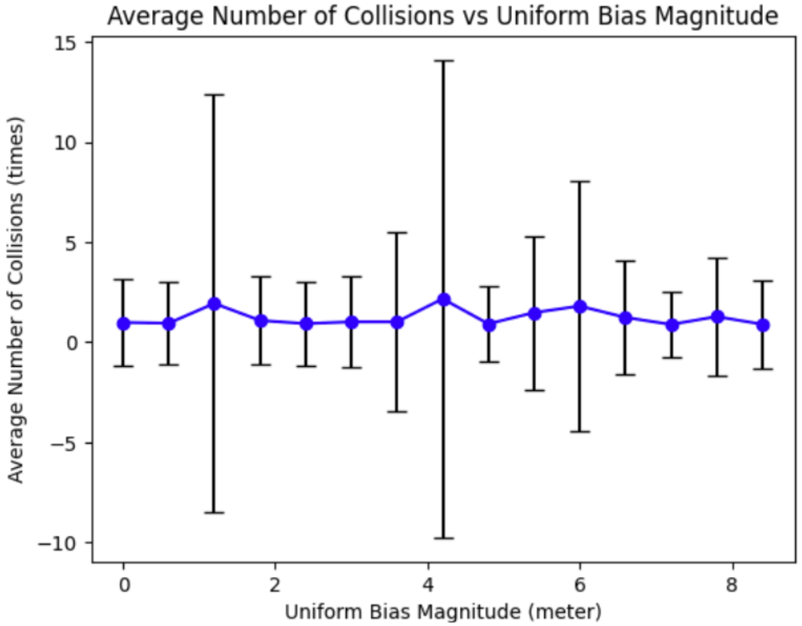
NO NOISE & DELAY IN TRAINING, UNIFORM BIAS MITIGATION

$$x_{\text{scaled}} = \left(\frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \cdot \frac{max\_range}{2} \right)$$

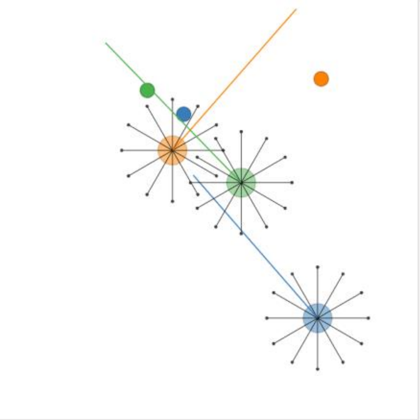
$$x_{\text{normalied}} = \left(\frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \cdot \frac{max\_range}{2} \right) + \max(0, -\min(x_{\text{scaled}}) + 0.01)$$

- Performance significantly improved

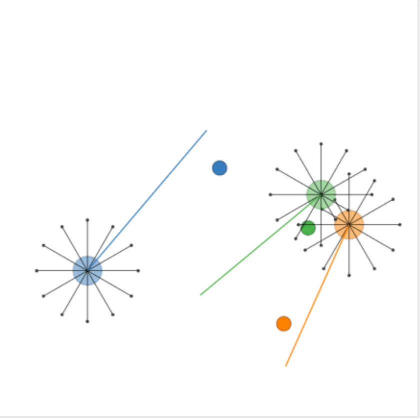




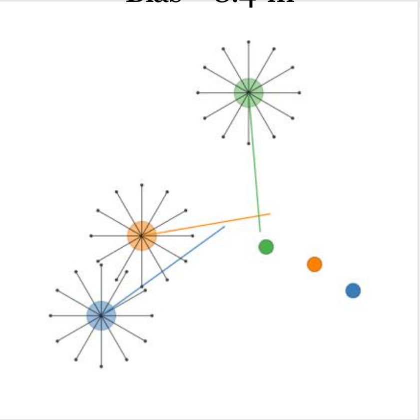
Bias=1.2 m



Bias =3 m



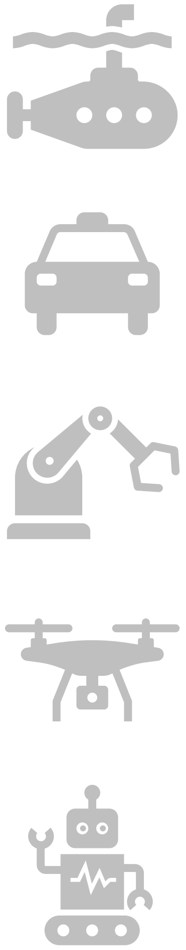
Bias =8.4 m



DISCUSSION AND PERSPECTIVE

	Trained without the noise	Trained with the noise
Gaussian Noise	Not robust to Gaussian Noise (Failure Threshold: std = 4m)	Become more robust (Failure Threshold: std = 20m)
Rainfall	The Gaussian Noise caused by rainfall is small	Complete navigation faster
Salt and Pepper Noise	Not robust to Salt and pepper Noise (Failure Threshold: Noise Ratio = 0.6)	More robust to Salt & Pepper Noise (Failure Threshold: Noise Ratio = 0.9) but does not reduce the number of collision
Delay	No significant effect on Max time to destination but increases the number of collision	Reduce the variance of Max time to destination but doesn't reduce the number of collision
Uniform Bias	RL agents are susceptible to bias (Failure Threshold: bias = 2m)	Failure Threshold increased to bias = 6m

POTENTIAL FUTURE WORK



- Extend the studies to other sensors (e.g. GPS, IMU Camera)
- Propose mitigation strategies for Gaussian noise, Salt and Pepper noise and delay
- Apply the study and strategies in other multi-agent systems (e.g. drones)

THANK YOU

We extend our heartfelt thanks to our supervisor, Dr. Ajay Shankar, for his exceptional guidance and support throughout the project.