



# Benchmarking Deep Reinforcement Learning for Navigation in Denied Sensor Environments

Mariusz Wisniewski<sup>1</sup> · Paraskevas Chatzithanos<sup>1</sup> · Weisi Guo<sup>1</sup> · Antonios Tsourdos<sup>1</sup>

Received: 21 October 2024 / Accepted: 2 September 2025  
© The Author(s) 2025

## Abstract

Deep Reinforcement learning (DRL) is used to enable autonomous navigation in unknown environments. Most research assumes perfect sensor data, but real-world environments may contain natural and artificial sensor noise and denial. Here, we present a benchmark of both well-used and emerging DRL algorithms in two navigation tasks - Lidar + position, and vision end-to-end - with configurable sensor denial effects. In particular, we are interested in comparing how different DRL methods (e.g. model-free, on-policy PPO vs. model-free off-policy TD3, vs. model-based DreamerV3) are affected by imperfect sensor readings. We show that DreamerV3 outperforms other methods in the visual end-to-end navigation task with a dynamic goal. Furthermore, DreamerV3 generally outperforms other methods in sensor-denied environments. In order to improve robustness, we use adversarial training and demonstrate an improved performance in denied environments, although we show that this may lead to the agent learning to choose high-risk actions in case of uncertain sensor readings, which is not appropriate for safety-critical scenarios. We anticipate this benchmark of different DRL methods and the usage of adversarial training to be a starting point for the development of more elaborate navigation strategies that are capable of dealing with uncertain and denied sensor readings.

**Keywords** Autonomy · Navigation · Sensor fusion · Deep reinforcement learning · Machine learning · Complex environments · Sensor faults · Sensor failure · Adversarial attacks · Adversarial training · Adversarial environmental perturbations

## 1 Introduction

Autonomous navigation is a fundamental challenge in unmanned systems. Conventional autonomous systems typically use Simultaneous Localisation and Mapping (SLAM) alongside Kalman filters that fuse data from multiple sources to map the environment, localize the agent, and generate trajectories to the goal. In recent years, deep reinforcement learning (DRL) methods have been used to train end-to-end policies that can navigate using a camera. These improvements have been made aided by advances in digital image processing largely due to convolutional neural networks trained using backpropagation [1–3], and parametrization of RL using deep neural networks [4].

Recent advances in vision-based reinforcement learning have demonstrated impressive results in agile robotic control, particularly in constrained tasks such as drone racing [5]. These methods leverage techniques like asymmetric actor-critic architectures, domain randomization, and learned abstractions to enable robust performance directly from pixel inputs. However, they often rely on structured environments with clearly defined visual landmarks, such as racing gates, and assume full sensor availability. Limitations remain in handling sensor degradation or outright sensor denial, where models trained in simulation can falter when key inputs are absent or corrupted. In contrast, our work focuses explicitly on benchmarking DRL algorithms under progressive sensor denial conditions, shedding light on their robustness, adaptability, and sim-to-real transfer characteristics in partially observable scenarios.

Whilst autonomous visual navigation is achievable in simulated environments [6], most of the research assumes perfect sensor readings. But, in real-world scenarios, information about the environment may be uncertain, incomplete, or even

✉ Mariusz Wisniewski  
m.wisniewski@cranfield.ac.uk

<sup>1</sup> Centre for Autonomous and Cyberphysical Systems,  
Cranfield University, College Road, MK43 0AL Cranfield,  
Bedfordshire, UK

intentionally denied. This can limit the ability of the navigation system to adapt to changes in the environment and make optimal decisions. These failure modes and how they affect the systems need to be understood for safety-critical operations, such as unmanned aerial vehicles or autonomous road vehicles. In the case of DRL agents, which are notoriously hard to explain, it is important to study how the sensor artefacts or failures affect the training and evaluation.

Further, real systems typically use a suite of sensors. Whilst a lot of the cutting-edge work in DRL focuses on end-to-end visual navigation, in practice, it is important to understand how the models learn across different modalities, by taking in data from multiple sensors. In this work, we study how sensor perturbations affect the training and evaluation of policies that use a camera or Lidar as the observation space.

For the clarity of the reader, we use the term environment in the classical RL sense: a system that takes an action as an input and outputs state and reward to the agent. We also use the term sensor/state perturbation, which we define as a modification of the state an adversarial process prior to the agent receiving the state (i.e. state-action Markov Decision Process [7]).

To understand the effects of sensor noise and failures, we compare several popular DRL architectures: TD3, PPO, PPO-LSTM, and DreamerV3. TD3 and PPO are a commonly used for evaluating environments with continuous action spaces, PPO-LSTM is the recurrent version of PPO (recurrent models were shown to outperform non-recurrent counterparts by Mirowski et al. [8]), and DreamerV3 which uses autoencoders to create an internal world model of the environment, and was shown to outperform PPO on certain environments.

We generate a baseline of their performance in a ‘vanilla’ environment, with no perturbations - noise or failures - to the sensors. We then progressively add random areas to the environment in which the sensors are perturbed. The policies are then evaluated to understand how successful they are in navigation under varying degrees of noise. To train and evaluate the methods we use a modified version of the ROS-based gymnasium environment for training navigation policies, *DRL-Robot-Navigation* [9]. It contains a 3D maze in which a robot is tasked with navigating to a goal. This publication is an extension of the conference paper on autonomous navigation presented at ICUAS 2024 [10]. It expands on the original conference paper by including a comparison of different RL architectures, including PPO, PPO-LSTM, and DreamerV3 (the original paper only used TD3). It presents a much more detailed study which includes both quantitative and qualitative results to understand the performance of the agents under adversarial attacks for both lidar and camera observation spaces (the original study only presented quantitative results on a subset of the scenarios studied here).

## 1.1 Review of DRL Applications to Navigation

Several studies have achieved end-to-end navigation using DRL with the camera as the sole sensor in a simulation environment [11]. Popular environments include the DeepMind Lab [12], ViZDoom [13], and DRL-Robot-Navigation [9].

Mirowski et al. [8] present an application of the A3C [14] algorithm to the Deepmind Lab environment. They use the camera as the sole sensor and their best-performing architecture consists of the image encoder, 2 LSTM layers, depth and loop predictions. The agent is rewarded for reaching the goal, or subgoals in the form of a fruit. The agent is not penalized for wall collisions, allowing strategies such as ‘wall-following’ that do not require memory. DRL can also be used to race FPV unmanned aerial vehicles (UAVs) against expert human pilots [15] by using the image sensor combined with the IMU (which are combined to detect gates and generate VIO states input to the DRL algorithm) in static environments. Polvara et al. show how a hierarchy of DQNs can be used for the autonomous landing of UAVs [16]. Their method handles a large variety of simulated environments and outperforms human pilots in some conditions. Interpretable deep reinforcement learning methods for end-to-end autonomous driving can handle complex urban scenarios [17]. Their method outperforms many baselines, including Deep Q-Networks (DQNs), deep deterministic policy gradient (DDPG), Twin Delayed DDPG (TD3), and Soft Actor Critic (SAC) in urban scenarios with surrounding vehicles. The memory of FPV agents in 3D mazes is evaluated by Pasukonis et al. [18]. They find that models such as Dreamer and IMPALA can match humans on smaller mazes (9x9), but their performance significantly drops off on larger mazes (15x15). Lample and Chaplot [19] show that DRL can be used to train agents to play the first-person-shooter game DOOM, in which the agent is required to navigate a 3D maze, collect items, and shoot at enemies. They found that by using DQN and DQRN with an LSTM layer, the agent can converge, but required an addition of predicting game features (using a separate loss function) to be able to reliably shoot at enemies. Liquid neural networks [20] have been proposed in recent years to achieve robust navigation for changing environments [21]. In most of these cases, whilst the physical environment is unknown or stochastic, the papers did not consider compromised sensor readings (bar liquid neural networks that consider noise and other perturbations).

## 1.2 Review of Sensor Fusion under Attacks

### 1.2.1 Trajectory Estimation Approaches

The first approach deals with intermittent attacks or denial, for example, when GNSS coverage is lost and a Kalman filter (or more advanced recurrent neural network) is used to

hold over the trajectory estimation until coverage recovers [22, 23]. This is useful for brief outages. The same idea can preemptively discover other V2V networks to improve localisation [24]. However, if the environment requires constant sharp manoeuvring to avoid dynamic objects, this scenario rules out trajectory estimation approaches. Also, if the attacks are continuous and erode the holdover accuracy, this would be unsuitable.

### 1.2.2 Sensor Fusion and Generative AI Approaches

The alternative is suitable for high-end platforms that can afford a diverse sensor suite, performing sensor fusion to alleviate the impact of one sensor being compromised. This is usually unsuitable for low-end drones and does not guarantee success [25]. A cheaper alternative is to use generative AI to generate alternative sensor representations using a single cheap sensor. For example, infrared images can be generated using a camera sensor and then fused together to achieve more robust recognition of the environment [26]. However, these approaches require an onboard diverse sensor suite or a powerful GAN operating onboard, which does not guarantee success.

### 1.2.3 Adversarial Environmental Perturbations in DRL

First-person-view (FPV) navigation problems are typically considered partially observable [27] because only a portion of the entire state is visible to the agent at any given time. But, sensor faults also have to be considered. Robust partially observable MDP [28] considers the POMDP to have uncertain parameters. Other approaches [29] consider the interaction between the adversary and the agent to be a zero-sum minmax game and define it as a Probabilistic Action Robust MDP. Zhang et al. [7] consider the addition of adversarial perturbations as a state-adversarial MDP (SA-MDP). In this framing, the state returned by the environment may be perturbed by an adversary (an adversary may be another agent or an environmental process that perturbs the state observations), making the observation uncertain and resulting in the action returned by the agent being potentially sub-optimal. Because we consider that the adversary can be static (e.g. it is part of the environment), we prefer the SA-MDP framing for our problem.

When building robust RL models, some of these works focus on physical perturbations [30] that may affect the physics of the agent (e.g. by introducing uncertain forces), whilst others consider perturbations to the sensor state [31]. Korkmaz [32, 33] argues that vanilla training results in more robust policies compared with state-of-the-art adversarial and robust training techniques. Havens et al. [34] propose a hierarchical RL method of switching sub-policies in the

presence of adversaries. Applications of adversarial training [35] show how a zero-sum action-robust RL method can be used to reduce conflicts in air mobility traffic management scenarios. Detection of adversarial states to the sensor itself is a challenge, which can be done by detecting out-of-domain samples. Peng et al. [36] use adaptive iterative learning for spatiotemporal fault detection. Detecting and responding to adversarial in actuators is investigated by using fuzzy wavelet neural networks [37].

Detection of adversarial perturbations on the guidance and planning of RL policies for UAVs were studied by Hickling et al. [38], with a later work proposing an autoencoder solution for their detection [39]. Clark et al. [40] study the effects of adversarial attacks on Q-learning policy for the routing of a robot. Dasgupta et al. [41] show how GNSS spoofing attacks can be detected using deep RL. Wang and Aouf [42] show how adversarial attacks can be detected for GNC of autonomous planetary landings to improve sim-to-real transfer. Hosseini et al. [43] evaluate DRL policies under adversarial attacks on an autonomous driving navigation task. Zhang et al. [44] show the effects of adversarial attacks on the control policies for quadrupedal robots. Robust learning [45] aims to provide certified lower bounds of state-action values - effectively providing guaranteed safety in the case of adversarial perturbations.

Ibrahim et al. [46] provide a taxonomy of adversarial attacks for autonomous vehicles in safety scenarios, splitting it into known and unknown attacks and defences. Similarly, Deng et al. [47] survey adversarial attacks and defences for autonomous driving systems. Ilahi et al. [48] present a survey on adversarial attacks and countermeasures for deep RL. A more recent survey on adversarial RL by Standen et al. [49] presents taxonomies of different types of attacks.

## 1.3 Gap and Novelty

We acknowledge that research on DRL and sensor attacks exists, and attempts at detecting adversarial attacks on DRL policies for navigation have been tried [38, 39, 43]. Further, work on understanding what Standen et al. [49] called functional attacks was theoretically studied by Korkmaz [31, 33, 50, 51]. However, we have not found elaborate studies that combine the effect of sensor perturbations on DRL performance during learning of local navigation policies (e.g., akin to the DRL maze navigation studies by Mirowski et al. [8]) in the form of functional attacks (i.e., the sensor state is modified by a function). We argue that, in order to accurately prove the effectiveness of any method dealing with adversarial sensor attacks, a benchmark of standard methods is first required which to the best of our knowledge does not exist.

We aim to understand the effects of sensor perturbations - a potential failure mode in autonomous systems - on the

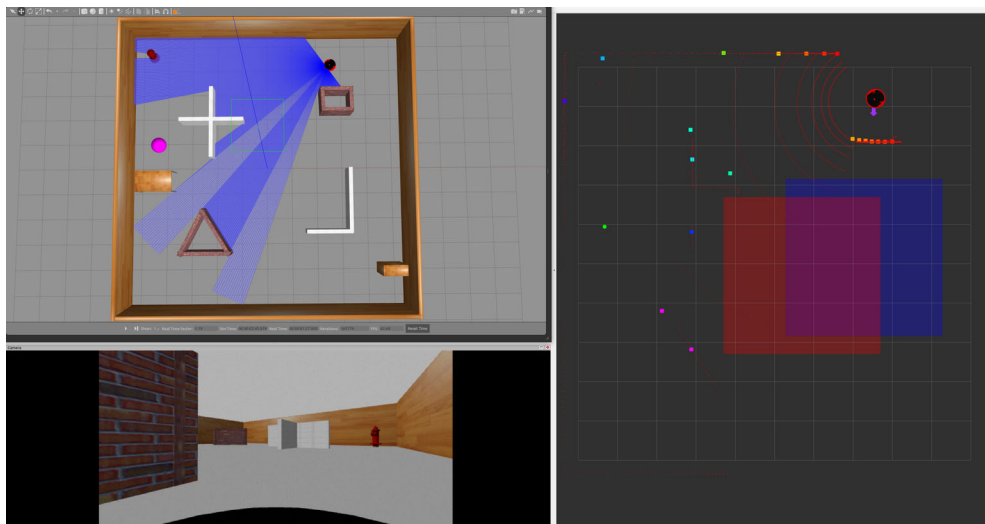


**Fig. 1** Example of the *DRL-Robot-Nav Maze Environment* [9]. Top-left: top-down view of the environment rendered in Gazebo. Bottom-left: the view from the RGB camera. Right: RViz visualization of the

discretized Lidar sensor readings, the robot, and the noisy sensor areas (blue: camera noise area, red: Lidar noise area). The pink sphere (top left) is added as a visual cue to mark the goal for vision-based policies

ability of DRL agents to learn navigation policies. To do this we:

- Benchmark different reinforcement learning algorithms on the *DRL-Robot-Navigation* environment.
- Study the effect of observation spaces - Lidar and camera - on the navigation capabilities.
- Perform quantitative and qualitative studies showing the effects of adversarial perturbations - Lidar noise and camera sensor failure - on the performance of DRL algorithms in the navigation task.

Further, to enable others to replicate our work, we open-source the modifications to the *DRL-Robot-Navigation* environment and update it to fit the gymnasium [52] interface.<sup>1</sup> This study aims to unify the research of state-adversarial perturbations [7], navigation [8], and robotics [9]. It achieves this by expanding on the research presented by Mirowski et al. [8] by using a real robot environment that can crash into walls and include state perturbations. It expands on the work of Cimurs et al. [9] by comparing different RL methods, different state spaces (camera) and studying state perturbations. It applies the concept of state-adversarial MDPs [7] to a robotics simulation. Finally, it is shown that the RL can be transferred to real-life environments.

Although first-person-view UAV SOTA methods [5, 15] use RL to learn to fly, they preprocess their image state observations to detect gates. Instead, we study whether DRL can use unfiltered observations to learn geometric features from

the states to navigate to the goal without any path planning or obstacle detection algorithms, or explicit sensor states (for the camera end-to-end state observation).

In this paper, we explain the methodology, i.e. the environment, changes to the environment, and the choice of reinforcement learning models, in Section 2. We present the results of the benchmark of different algorithms on the environment, the Lidar and camera noise studies, and discussion with respect to the literature in Section 3. Finally, we conclude and make suggestions for further work in Section 4.

## 2 Methodology

We build on top of the *DRL-Robot-Navigation* environment [9], which simulates a robot navigating around a 10 m x 10 m maze, populated with static obstacles. The robot has a suite of sensors - Lidar, camera, and odometry - but only a subset of these was used as the state observation to train the reinforcement learning model in the original paper. The original paper uses Lidar, distance to goal, and angle to goal, concatenated together as the observation space, and the camera is not used. An illustration of the environment, which shows the *DRL-Robot-Navigation* environment along with sensor readings, is presented in Fig. 1.

As the focus of this study is on studying adversarial sensor perturbations and testing different sensor suites, the changes to the environment are described in Section 2.1. The benchmarking procedure, of training the different RL algorithms - TD3, PPO, PPO LSTM, and DreamerV3, using different modalities and with differing sensor noise, is

<sup>1</sup> <https://github.com/mazqtpopx/cranfield-navigation-gym>

illustrated in Fig. 2. The reinforcement learning algorithms used in the benchmark are described in Section 2.2. Training experimental settings, including hyperparameters, are shown in Section 2.3. The evaluation process is described in Section 2.4.

## 2.1 Changes to the DRL-Robot-Navigation Environment

To enable our experiments, the *DRL-Robot-Navigation* environment required several changes. A primary change to the environment is the addition of a vision-based observation space. To do this, the image from the robot is outputted from the environment, and a visual marker is added to mark the position of the goal. A pink sphere of diameter 0.5 m is added where the goal resides and is shown in Fig. 1 (top left image).

The collision system is updated, as in the original implementation, which relied on the reading from the Lidar sensor. This is not a reliable way of detecting collisions, and adding noise to the Lidar sensor could trigger collisions. Instead, we generate a flag and detect any physical collisions between objects by triggering a ROS topic that can be subscribed to.

We added areas that contain sensor noise: separately for Lidar and camera. The areas are fixed-size rectangles that spawn at a random point of the map at the start of each episode. The way the sensor faults are modelled is described further in Section 2.1.1.

We updated the reward system. Originally, the agent received 100 reward for reaching the goal and -100 for a collision with a wall. Otherwise, a function of the actions (penalizing the robot for inaction) and the minimum distance from a wall (based on the Lidar observation, if the robot is closer than 1 m to a wall) was returned. First, we normalized the reward between +1 and -1. Next, we added a penalty for taking a step: of size -1 divided by the maximum number of steps. This was done to prevent a local minima where the

robot doesn't move and is in line with other gymnasium environments [52]. We also considered the function of penalizing inaction and the Lidar distance to the wall to be unnecessary. Instead, we consider that these items - inaction and obstacle avoidance - should be learned on the policy side from observation without reward shaping.

### 2.1.1 Modelling Adversarial Sensor Perturbations

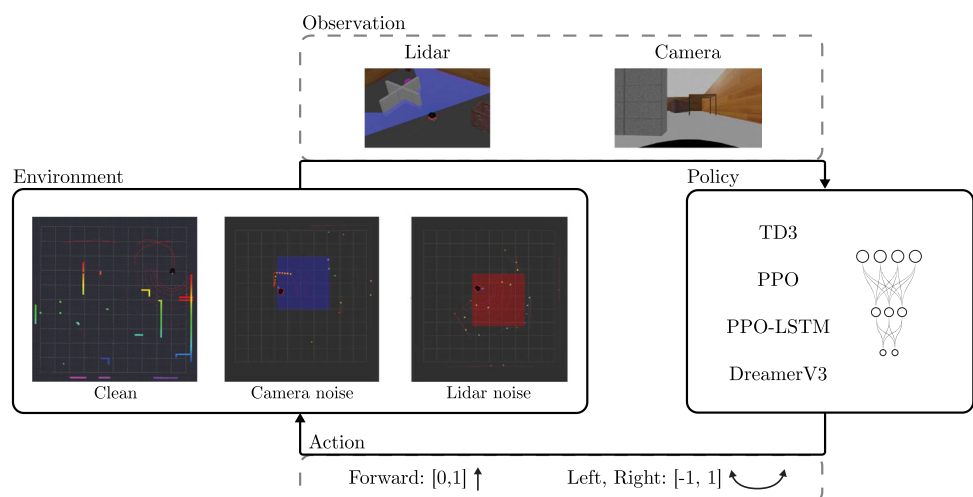
Autonomous vehicles often use a camera as a primary sensor for navigation, but physical sensor attacks are rarely considered. Kim et al. [53] perform a review of drone attacks, including laser attacks on the camera sensor. They find that when applied, the camera pixels are distorted. This shows that camera sensors are susceptible to physical attacks. Camera attacks are modelled as a complete failure, i.e. all the pixels are turned black (0) whilst inside the camera noise rectangle. Whilst this is not explicitly modelled after a physical attack, it is a possible temporary failure mode for a camera and is shown in Fig. 3(b). Lidar attacks are modelled by adding a strong Gaussian noise, ranging between 0-5 meters to each of the Lidar points. Sensor noise is applied evenly throughout the noisy area. A Lidar sensor perturbation is shown in Fig. 3(a). According to the taxonomy by Standen et al. [49], the camera attack could be classified as a functional attack.

## 2.2 Reinforcement Learning Benchmark

A benchmark of the following RL methods is performed on the *DRL-Robot-Navigation* environment:

- **Twin Delayed DDPG (TD3)** [54] is an off-policy, model-free algorithm designed to work on continuous action spaces. It employs two Q-value approximators (critics) and uses the smaller of the two Q-values to update the policy, which helps to mitigate the problem

**Fig. 2** Experimental diagram showing the process of the presented experiments. The environment consists of clean, camera noise, and Lidar noise variants. The environment outputs the observation in the form of two modalities, Lidar and camera. Multiple algorithms are then trained to navigate to the goal whilst avoiding the obstacles



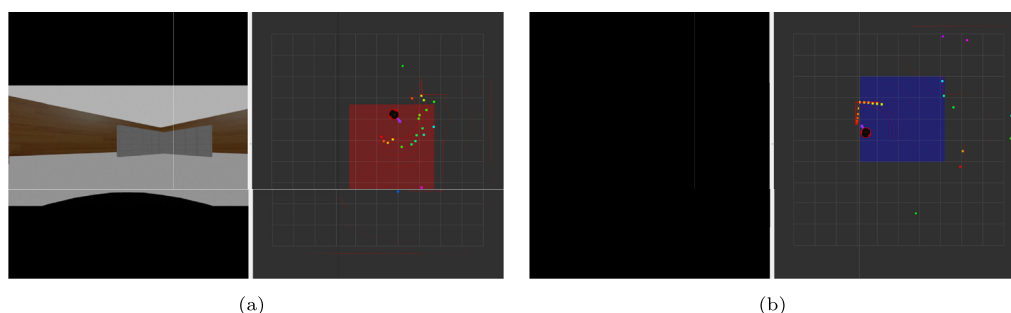


**Fig. 3** Example of a robot in the Lidar noise area, visualized by RViz. Once inside the red Lidar noise area, the discretized Lidar points have Gaussian noise added to them and no longer provide accurate representations of the environment. Example of a robot in the camera noise area,

visualized by RViz. Once inside the blue camera noise area, the image feed (shown on the left) has all pixels turned to black - this simulates a camera failure

of overestimation bias. TD3 also introduces a delay in updating the policy network to reduce the variance of policy updates and adds noise to the target action to encourage exploration. The hyperparameters used to train TD3 are shown in Table 1.

- **Proximal Policy Optimization (PPO)** [55] is an on-policy, model-free algorithm designed to work on continuous spaces. **Recurrent PPO**, a modification of PPO with the introduction of a long short-term memory (LSTM) layer [56], is also used. The hyperparameters used to train PPO and Recurrent PPO are shown in Table 2. The recurrent architecture is similar to the PPO, but contains an LSTM layer before the Actor-Critic Multilayer Perceptron (but after the convolutional layers for the camera observation).
- **DreamerV3** [57] is an off-policy, model-based algorithm that learns a world model from the experience by incorporating an autoencoder that encodes the input into a discrete representation. It uses recurrence and predicts the dynamics, reward, and continuity of the episode. The hyperparameters are fixed, hence tuning is not required. It has been shown to outperform PPO on several environments, including navigation on DeepMind Lab [12].

**Table 1** TD3 training parameters (final values are bolded)

| Parameter                      | Value                                                 |
|--------------------------------|-------------------------------------------------------|
| Learning Rate                  | 0.03 / 0.003 / <b>0.0003</b> / 0.00003 (see LR study) |
| Tau                            | 0.005                                                 |
| Learning Starts (after, steps) | 5000                                                  |
| Batch Size                     | 16384 (Lidar), 256 (Camera)                           |
| Discount Factor                | 0.99                                                  |
| Exploration Noise              | 0.3                                                   |

Based on this, it is expected that DreamerV3 should outperform the other algorithms on our navigation problems. The default model is used.

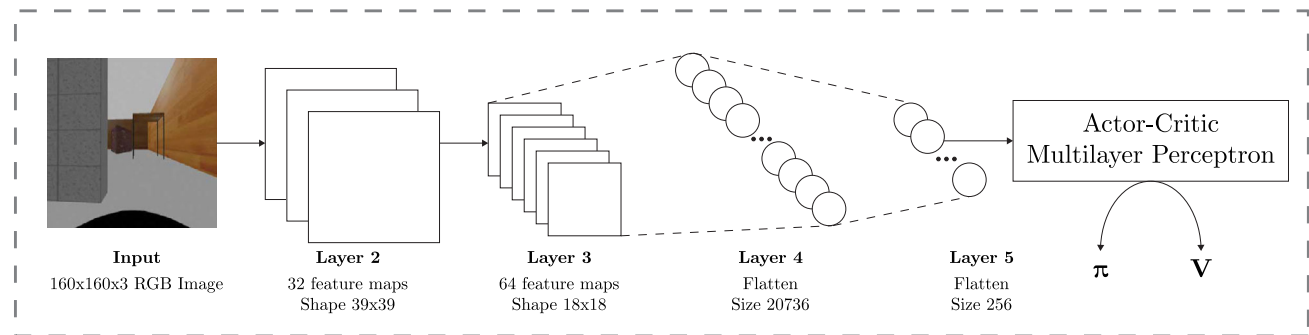
Stable-Baselines3 [58] implementations of TD3, PPO, and recurrent PPO were used for the experiments. The general setup for TD3 and PPO is shown in Fig. 4, showing the networks for both camera and Lidar observations. For the camera observation, the image is initially passed through a convolutional network to reduce the dimensionality to 256 features. Those features are then passed into the actor-critic MLP. In the case of the recurrent PPO, the features are first passed through an LSTM layer before going to the actor-critic. For the Lidar, because it only contains 24 data points, the features are passed straight to the actor-critic. We perform:

- A benchmark of the RL methods.
- A learning-rate discovery to find the optimal LR for each algorithm as it is the most sensitive hyperparameter (apart from Dreamerv3, as it does not require hyperparameter tuning). This is shown in the appendix.
- A study into the performance of the agents across different observation spaces in sensor-perturbed areas.

**Table 2** PPO and PPO LSTM training parameters (final values are bolded)

| Parameter              | Value                                                 |
|------------------------|-------------------------------------------------------|
| Learning Rate (Lidar)  | 0.03 / <b>0.003</b> / 0.0003 / 0.00003 (see LR study) |
| Learning Rate (Camera) | 0.03 / 0.003 / <b>0.0003</b> / 0.00003 (see LR study) |
| Batch Size             | 256                                                   |
| Discount Factor        | 0.99                                                  |

## Camera Observation



## Lidar Observation

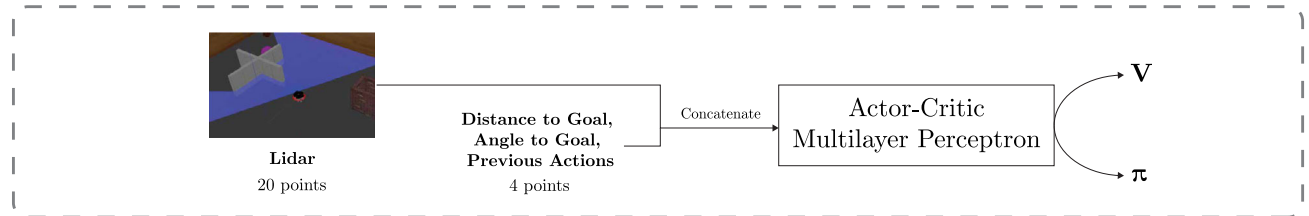


**Fig. 4** Camera and Lidar observation networks. In camera observation models (TD3, PPO, PPO-LSTM), the image is first passed through convolutional layers and flattened to 256 features to reduce the dimensionality of the input. These features are then input to the actor-critic

MLP. For the Lidar observation models (TD3, PPO), the input is already low-dimensional, and so the 20 Lidar readings are concatenated with the distance to goal, angle to goal, and previous actions and input to the MLP

## 2.3 Experimental Scenario Settings

The experimental settings are shown in Table 3. The Lidar observation space comprises 20 discrete points, taken at even intervals in a 180° arc around the robot. The camera image produces a 160x160 px RGB image and a 64x64 px RGB image for DreamerV3 (the input to the DreamerV3 is required to be a power of two<sup>2</sup>). Lidar noise is Gaussian, and camera noise results in a complete blackout. The agent position input (for Lidar modality only) is fully accurate. The goal position for Lidar is random, and for the camera, it is static. The time delta between steps is 0.05 seconds - this is the delay between which the agent makes actions. Note that the simulation runs at 4.0 times real-time speed, which results in this time step being 0.2 seconds, i.e. the agent runs at effectively 5 Hz in real time. The reward function produces a sparse reward when the agent reaches the goal, a penalty for colliding with a wall, and a small penalty each step to prevent inaction. The maximum steps per episode are 50.

There exist several random parameters within the simulation that affect the learning process. Randomness plays a key role, as shown by Mirowski et al. [8]. In their simulation, the RL agents outperformed humans on static small mazes but struggled on large complex mazes (with random parameters such as random agent spawning). Another difference

between *DRL-Robot-Navigation* and DeepMind Lab is that our environment penalizes collisions with the wall, preventing wall-following strategies.

Camera policies require longer training compared with Lidar (greater than 500k steps), which exceeds the computational budget for this study. To simplify it, the goal is spawned at the centre of the map, instead of being spawned randomly.

**Table 3** Simulation Parameters

|                     | Parameter details                                                                                                               |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------|
| Lidar Measurements  | 180°LiDAR data discretized into 20 intervals                                                                                    |
| Camera Measurements | 160px x 160px RGB image 64px x 64px RGB image (DreamerV3)                                                                       |
| Attack Type Lidar   | Gaussian noise added to each discrete point                                                                                     |
| Attack Type Camera  | Complete failure                                                                                                                |
| Agent Position      | Fully accurate (ground truth from Simulation)                                                                                   |
| Reward function     | $R(s, a) = \begin{cases} 1, & \text{goal reached} \\ -1, & \text{collision} \\ -1/50, & \text{otherwise} \end{cases} \quad (1)$ |
| Max. Episode Steps  | 50 for Lidar and camera (static goal)<br>500 for camera (random goal)                                                           |

<sup>2</sup> <https://github.com/danijar/dreamerv3/issues/12>

The robot is still spawned with a random position and orientation. This is akin to the ‘static maze’ from DeepMind Lab, as tested by Mirowski et al. Their ‘random goal maze’ is akin to our random goal position that we test during our Lidar experiments (albeit DeepMind Lab did not penalize wall collisions).

Generally, apart from DreamerV3, the camera policies likely take longer to converge to a solution due to increased observation space: even after convolutional layers, the state contains 256 features compared with the much simpler 24 points for Lidar (including previous actions, distance to goal, and angle to goal).

The original implementation of *DRL-Robot-Navigation* environment randomizes the starting position and orientation of the robot, as well as the position of the goal at the start of each episode. To improve the convergence of camera policies, in order to better understand the sensor denial effects, we simplify the environment for our camera experiments. We fix the position of the goal to always be in the centre of the maze. This allows for less stochasticity in the environment and enables quick convergence of the policy. For the Lidar experiments, we keep the spawn position of the goal and the robot random.

## 2.4 Training and Evaluation

The reinforcement learning (RL) models described in Section 2.2 are trained with the following configurations and environments:

- Lidar (this includes distance and angle to goal). The models are trained on a variety of 0x0 (no noise) maps, as well as maps with varying amounts of noise: 3x3, 5x5, and 7x7 metre noise zones.
- Camera only, static goal. The models are trained on a variety of 0x0 (no noise) maps, as well as maps with varying amounts of noise: 3x3, 5x5, and 7x7 metre noise zone.

Unless otherwise stated, camera configurations contain a static goal to enable better convergence, as described in Section 2.3. Each model is then evaluated on maps specified in the Section 2.4, but generally this consists of being tested on each of the environments of differing noise size: 0x0, 3x3, 5x5, 7x7. In some cases, the model is evaluated on a simplified version of the map, as described in Section 2.4.

Figure 5 shows the outline of the training and evaluation setup for Lidar and camera environments.

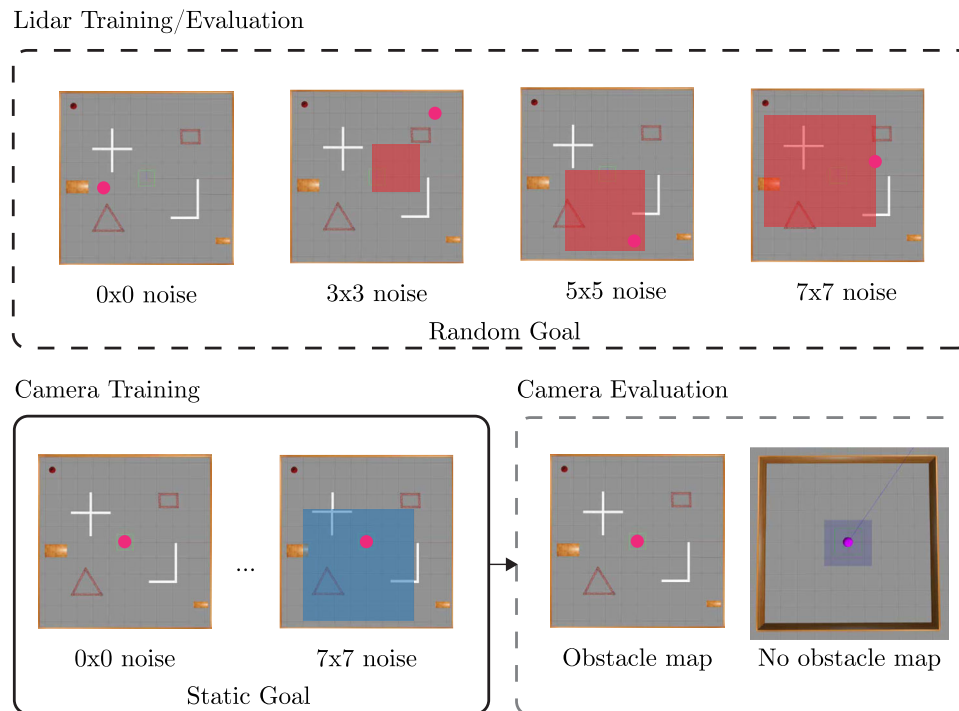


**Fig. 5** Training and evaluation scenarios for Lidar and camera sensors. For Lidar scenarios, the models are trained on a variety of 0x0 (vanilla) maps and maps with varying amounts of sensor perturbation: 3x3, 5x5, and 7x7 metre sensor noise zones. These zones contain Gaussian noise, which is added to the sensor readings. These trained policies are then evaluated on the same map. Likewise, for camera scenarios, the models are trained on a variety of 0x0 (vanilla) maps and maps with varying

amounts of sensor perturbation: 3x3, 5x5, and 7x7 metre sensor denial zones. Inside these zones, the camera sensor completely fails (all pixels turn to black). The models are trained on the default map, with a static goal in the middle, and the blue areas in which the camera completely fails. These policies are then evaluated on the default map with a varying degree of noise. A second evaluation is performed on a map with no obstacles to better understand the learned policies in sensor-denied areas

### 2.4.1 Camera Evaluation Maps

For the evaluation of camera policies, two different maps are used:

- *DRL-Robot-Navigation* map (same as training).
- A simplified *DRL-Robot-Navigation* map no obstacles, shown in Fig. 6. The goal is always static in the centre. The noise area is static in the centre of the map.

The purpose of using the simplified map is to better understand the behaviour of the policies. As the obstacles are taken out, the only task left is to get through the sensor-denied area to the goal. As the noise is static, it always obstructs the goal. In this evaluation task, we are ignoring the ability of the agent to avoid obstacles, but are focusing on whether the agent has learned to get through the noise area. The simplified map is only used for evaluation and is never used for training of the policies. By default, the *DRL-Robot-Navigation* is used for evaluation. The use of the simplified maps is explicitly specified in the results section.

## 3 Results

Several experiments are conducted to benchmark the different DRL algorithms on the environment and understand how different training regimes (vanilla vs adversarial) affect the performance of the models in environments with varying environmental perturbation size. While initially training of the Lidar and camera modalities is compared in Section 3.1, the rest of the results section is split into two parts, based on modalities. The first part, in Section 3.2, follows how PPO and TD3 perform on in terms of quantitative results, and then qualitatively in Section 3.3 by understanding the paths followed by the policies, and their differences under different training regimes.

The camera observation section begins with a direct training comparison of the different algorithms on the vanilla environment in Section 3.4 for the camera observation with both static and random goal, continues with a quantitative benchmark of the different models with sensor denial in Section 3.5, and the models are evaluated on the no obstacle map in Section 3.6. The camera modality with the random goal is evaluated in Section 3.7. Real world experiments are conducted Section 3.8 Finally, the results are discussed with respect to the literature in Section 3.9.

### 3.1 Modality Study

The modality study shown in Fig. 7 shows the differences in model performance for the PPO algorithm across Lidar (with a random goal), camera with a static goal, and camera with a random goal. Lidar and camera with a static goal are quick to converge to a solution, requiring only 200,000 steps. Camera with a random goal appears to be more challenging, and requires over 1,000,000 steps. The reason for the camera random goal scenario is so challenging is that the model has to learn to navigate to the pink goal without knowing the relative position between itself and the goal (which Lidar modality does contain). Instead, it needs to learn spatial information to be able to avoid the walls, and to learn to associate the pink goal with a reward.

### 3.2 Lidar Navigation with Noise

Although different algorithms are proposed in Section 2.2, we were unable to find an appropriate learning rate to get PPO-LSTM to converge, as shown in Appendix A3.

Our initial hypothesis was that, as this is a problem that requires navigating a dynamic robot, the learning would benefit from recurrent layers. We have found a similar effect in other studies that require complex dynamics [59]. Moreover, classic RL studies [4] use framestacking to contain

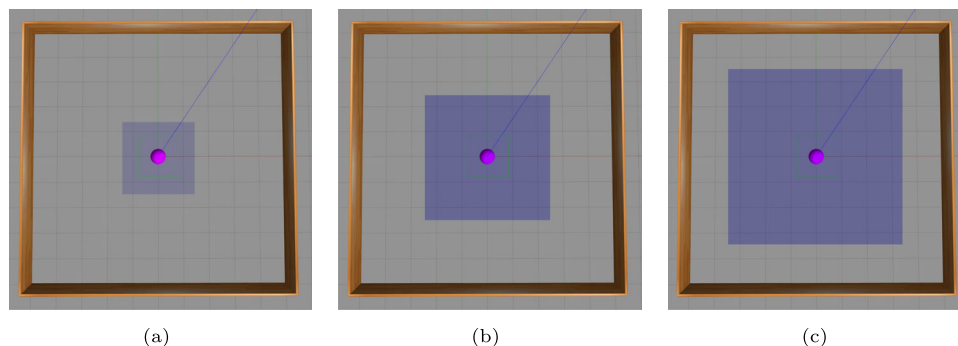


**Fig. 6** Simplified version of the *DRL-Robot-Navigation* map with no obstacles, with a static goal in the centre and a sensor perturbation area surrounding it. **(a)** shows the 3x3 sensor denied area, **(b)** shows the

5x5 sensor denied area, and **(c)** the 7x7 sensor denied area. The robot is spawned at a random position and orientation around the map. This map is only used to evaluate the camera policies, and not Lidar policies

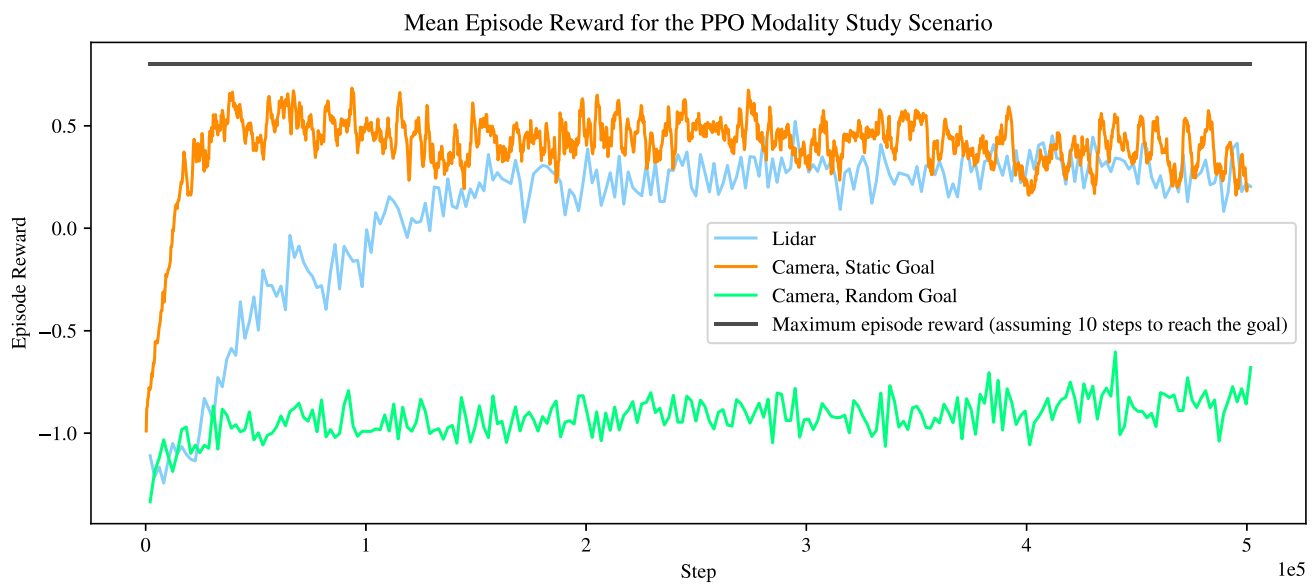


**Fig. 7** Modality study using PPO for Lidar and camera. The camera was trained on static and random goals. Each of the models was trained for 500,000 steps

dynamic information about the environment, and recurrence is an alternative that allows longer-term planning. However, the use of LSTMs also introduces new hyperparameters to the network that require tuning. It becomes sensitive to the number of steps in the episode, velocities/sample rate at which the actions are executed, and the relation to the batch size/number of sampling steps. This requires a thorough hyperparameter study alongside the learning rate. For sim-to-real transferability, using non-recurrent models is typically easier to achieve. This is because recurrent models keep memory in the form of hidden states, and this influences the actions based on previous states. Meanwhile, non-recurrent models simply learn the value and the action-value for each given state and hence

should be less sensitive to the dynamics transfers between the simulation and the real world. While we were able to achieve convergence for the non-recurrent models, more challenging versions of this environment might require recurrence to converge.

While it is possible to train DreamerV3 on the Lidar observation space, it is a large model for such a small observation space and we did not find that its performance increased over the other models. Hence, only TD3 and PPO are compared in this section.

Figure 8 shows the evaluation results of the TD3 and PPO using the Lidar observation space on the default map. The algorithms are trained on the 0x0 (vanilla) map, and evaluated

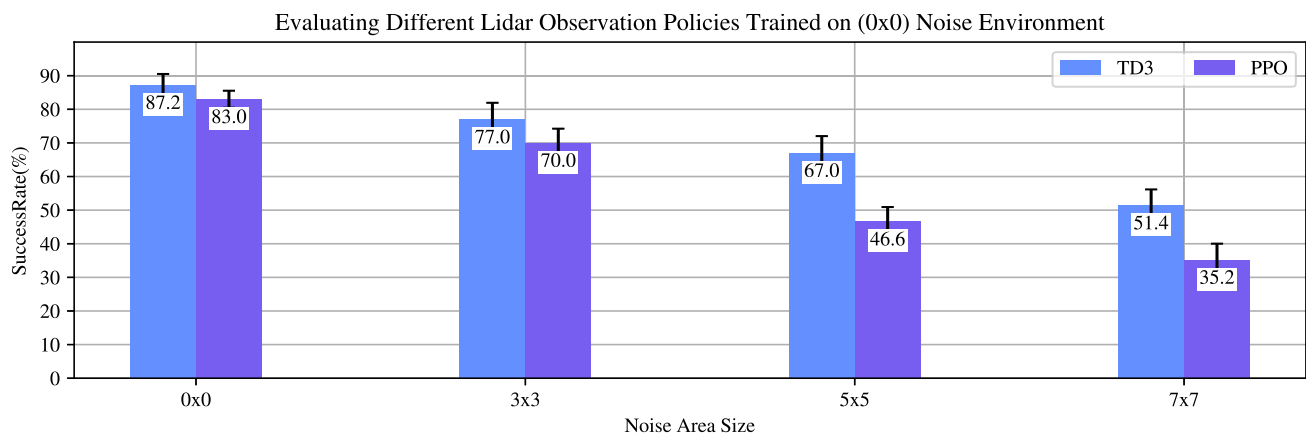


**Fig. 8** Lidar noise area study for algorithms trained on the default environment with no noise. The algorithms were then evaluated on the default environment (with a random goal) with changing noise area

sizes, as shown on the x-axis. The evaluation was performed for 100 episodes and repeated 5 times to establish a standard deviation

on environments of increasing Lidar noise area: 0x0 (vanilla), 3x3, 5x5, and 7x7 - referring to the size of the rectangle of the noise area in meters. TD3 outperforms PPO across the different evaluations. It achieves a mean success rate of 87.2% on the 0x0 evaluation scenario, compared with 83.0% for PPO. The performance differential between the two models generally increases with larger noise area sizes. When evaluated on the 7x7 noise area size, TD3 achieves 51.4%, while PPO achieves 35.2%.

In general, TD3 consistently outperforms PPO across the different noise area sizes. As the models were trained for the same number of timesteps, this is likely an effect of TD3 being more sample efficient due to its off-policy nature. This is consistent with the results from the vision-only modality later shown in Sections 3.4 and 3.7 where TD3 does not converge at all for random goal scenarios.

As the evaluation noise area size increases, the performance gap between TD3 and PPO increases. This shows that the vanilla TD3 is still robust to noise despite not being adversarially trained. Meanwhile, the PPO performance decreases, which is possibly an effect of poorer convergence. Because the noise samples can be considered out-of-domain, both of the models should, in principle, react similarly to being evaluated on noisy environments. The larger differences therefore suggest a better convergence of the TD3 model, which is likely due to the sample efficiency.

Because of the low dimensionality of the Lidar data, TD3 can be trained at a similar rate to PPO, and therefore presents as the more appropriate algorithm for this scenario. Whilst Cimurs et al. [9] did not perform a comparative study with other algorithms, they appear to have chosen the correct algorithm for their problem.

Figure 9 shows the same evaluation results of the TD3 and PPO, with the algorithms trained on the 3x3 noise map. In

these results, the PPO closely matches the performance of TD3 across the evaluations, outperforming it on larger noise area sizes. When evaluated on the vanilla 0x0 scenario, TD3 achieves a mean success rate of 69.6% and PPO achieves 68.0%. This is a performance drop from the models trained on the vanilla 0x0 environments. When evaluated on the 7x7 scenario, the PPO model achieves a mean success rate of 54.0%. This is an improvement from the PPO model trained on the vanilla 0x0 environment and slightly outperforms the TD3 model trained on the same regime. The TD3 model trained on the 3x3 noise environment shows performance drops across all evaluation scenarios compared with the TD3 model trained on the 0x0 environment.

### 3.3 Lidar Noise Training Regimes, Qualitative Results

As shown in Section 3.2, the training regimes affect the evaluation performance of the algorithms. To better understand the behaviour of Lidar-specific policies, we can more closely investigate the effects of training in noisy regimes for each algorithm and investigate the trajectories of the robot during evaluation.

Figure 10 shows the PPO policies trained on different Lidar noise regimes evaluated on the default map. It shows that the vanilla PPO (0x0) performs the best when evaluated on a map with no noise, with a mean success rate of 83%. The policies trained on noisy environments (PPO 3x3, 5x5, and 7x7) begin to outperform the vanilla policy when evaluated on the 5x5 noise map. While the noisy policies do not significantly degrade in performance as more noise is added to the environment, they show a consistent drop in performance compared with the vanilla model when evaluated on the 0x0 no noise environment. This is likely due

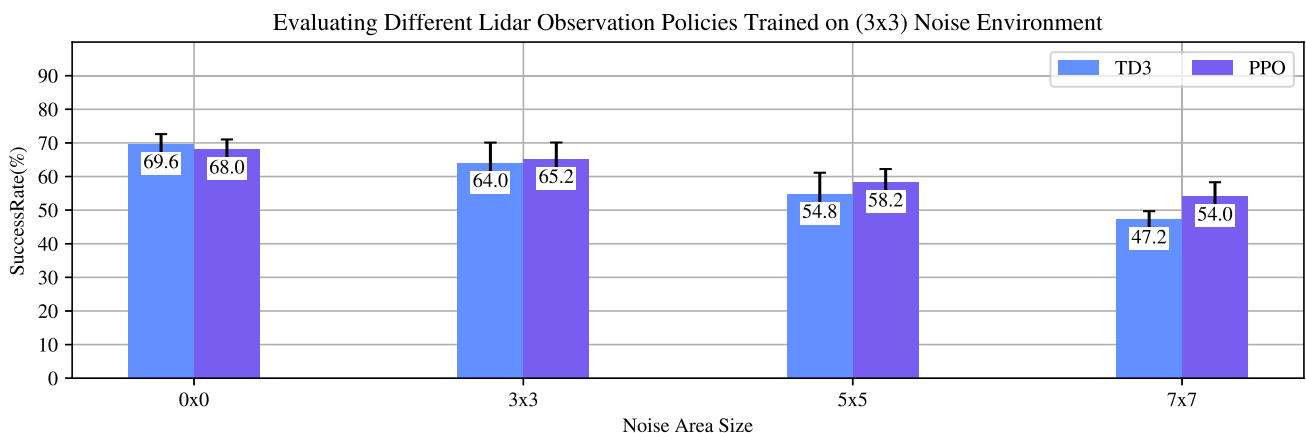


**Fig. 9** Lidar noise area study for algorithms trained on the default environment with random 3x3 noise areas and random goal. The algorithms were then evaluated on the same map with different noise area sizes, as

shown on the x-axis. The evaluation was performed for 100 episodes and repeated 5 times to establish a standard deviation

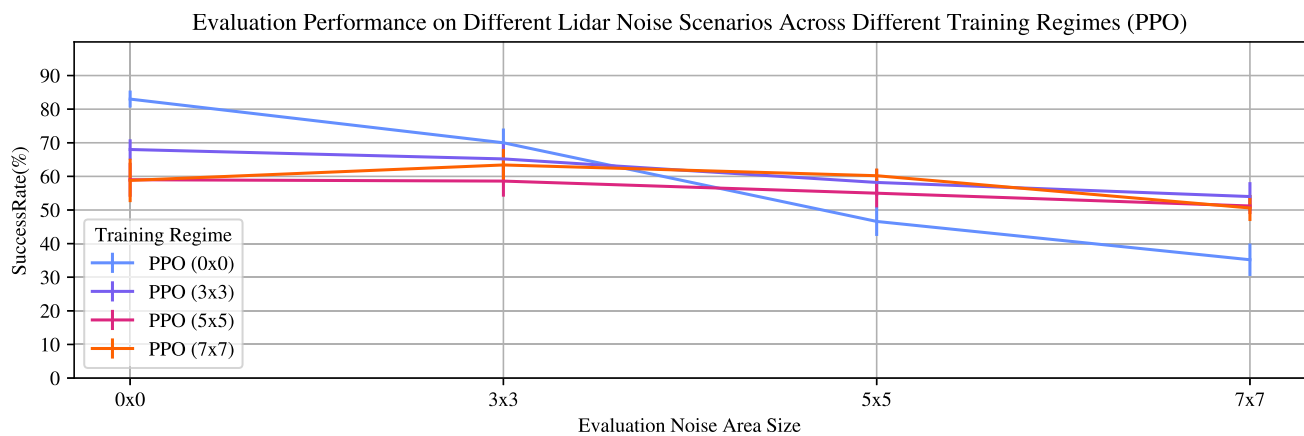


**Fig. 10** PPO policies trained on different Lidar noise regimes evaluated on the default map (with random goal) with varying noise area sizes. The policies were evaluated for 100 steps 5 times, and the plot shows the mean and the standard error across those 5 repeated runs

to the noisy policies following a riskier strategy, sacrificing collision avoidance capabilities to improve the navigation performance. To better understand this, we can qualitatively investigate the evaluation runs by plotting their paths, crashes and successful navigation.

Figure 11 shows the 100 path plots generated during the evaluation of 2 different policies (PPO 0x0 and PPO 3x3) on two different evaluation maps (0x0 and 7x7). Figure 11(a) shows the vanilla PPO evaluated on the no noise map. The same policy is evaluated on a map with random 7x7 noise areas in Fig. 11(b). The policy is unable to reach the targets and shows a significant decrease in performance (from 79 successes to 26 successes). It also shows that in half of the runs, the policy reached the maximum episode time step limit, and it neither crashed nor reached the target. Compared with the evaluation on the no noise map, the paths tend to be much shorter, showing that the policy prefers to follow a 'safe' strategy of not doing anything when inside the noisy areas. In contrast, when trained on a little noise (3x3 random areas) and evaluated on no noise as shown in Fig. 11(c), the overall performance drops from 79 successful episodes to 69, and the number of crashes increases from 21 to 31. But, when evaluated on the 7x7 noise area as shown in Fig. 11(d), the policy no longer follows a 'safe' strategy and instead opts for a 'riskier' approach. This increases the number of successful episodes from 26 to 49, but also increases the number of crashes from 24 to 51.

Overall, the safe policy did not result in better overall performance because the reward for reaching the max number of steps is -1, like with crashing, it is -1. But, because the risky policy results in more successful runs, resulting in the positive +1 reward, it is the more optimal policy for the noisy evaluation maps. It is an interesting observation that the riskier strategies consistently emerged across the PPO runs when trained on noisy maps, while when not trained on

any noise, the policy followed a safer strategy of letting the episodes run out when it was not certain of its sensor readings, as shown in Fig. 10. This is likely due to the noisy sensor readings falling outside of the previously observed domain for the policy trained on the non-noisy environment.

Figure 12 likewise shows the TD3 policies. Unlike in the PPO training schemes (Fig. 10), TD3 generally did not benefit from being trained on noisy scenarios, and the vanilla policy trained on the no-noise map performed the best across all evaluation scenarios. The difference in performance may have arisen from the differences in on-policy (PPO) and off-policy (TD3). As TD3 keeps a buffer of past experiences, any perturbed states will remain in the buffer and will affect the gradient updates. Meanwhile, because it is only trained on recent experience, PPO may learn from batches where the agent does not enter a noisy area (or the amount of perturbed states is insignificant). This allows for gradient updates to be performed separately, on both clean and perturbed experiences. Meanwhile, the gradient updates for TD3 will always be affected by perturbed experiences. Further, the clipped surrogate objective function of the PPO may be preventing large updates caused by adversarial perturbations - a mechanism not present in TD3.

Whilst ultimately, the effect of the training paradigm performance will depend on the environment and nature of the adversarial perturbation, because we have controlled for these in our environment, we conclude that there is a clear algorithmic difference on the model performance. Hence, we conclude that TD3 is more sample-efficient than PPO, resulting in better-performing policies from an equivalent number of training samples due to its off-policy nature. However, this mechanic also appears to negatively affect it when trained in adversarial environments. Meanwhile, the on-policy PPO ends up benefiting from being trained on adversarial environments, despite being less sample-efficient overall.

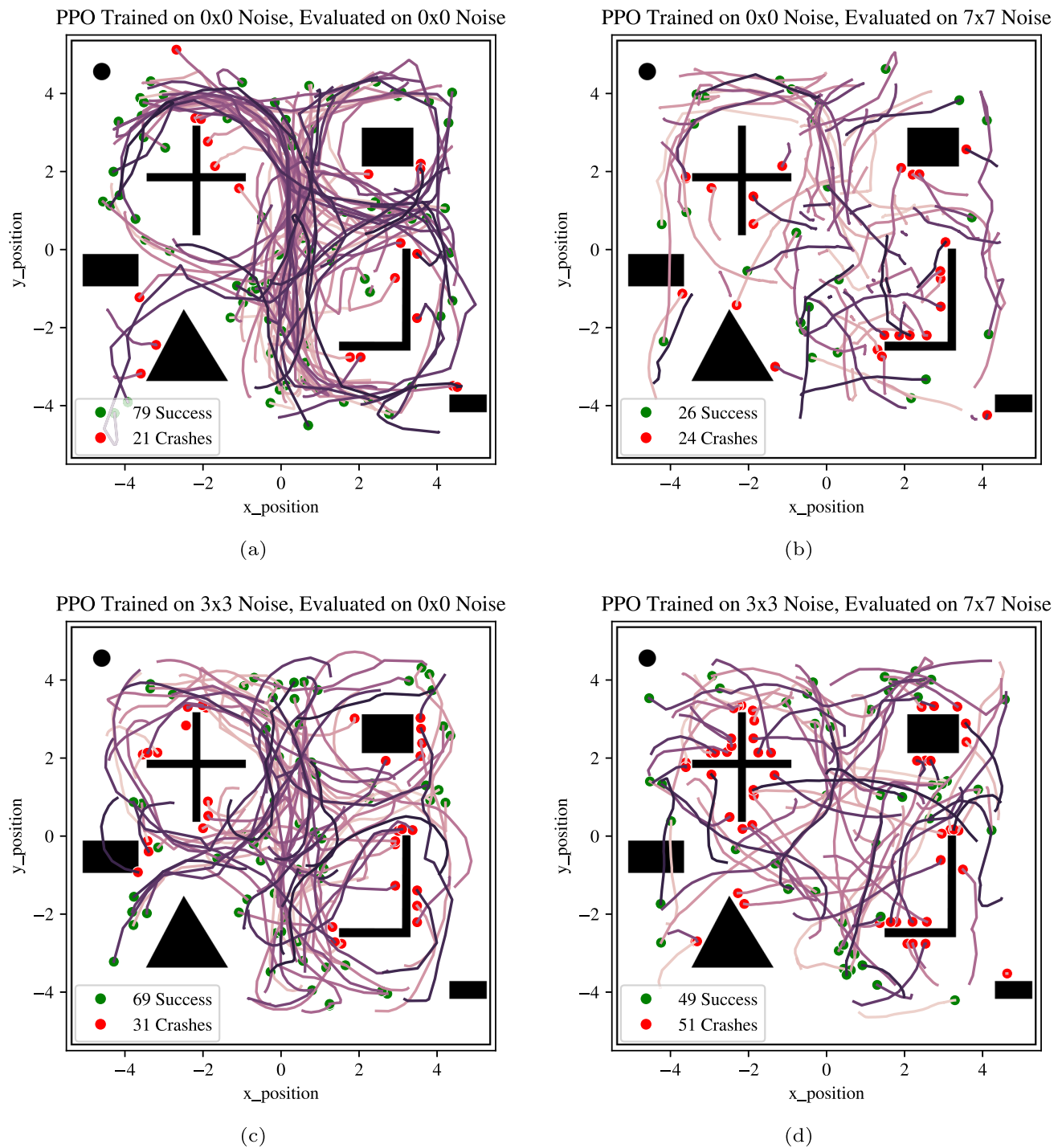


**Fig. 11** 100 path plots generated during evaluation by vanilla policy (PPO 0x0) evaluated on 0x0 noise map (a), vanilla policy (PPO 0x0) evaluated on 7x7 noise map (b), noise policy (PPO 3x3) evaluated on 0x0 noise map (c), and noise policy (PPO 3x3) evaluated on 7x7 noise

map (d). The red dot shows a collision and the green dot shows a successful navigation to the goal. Any paths not showing a success or a crash mean that the scenario was terminated due to reaching maximum steps

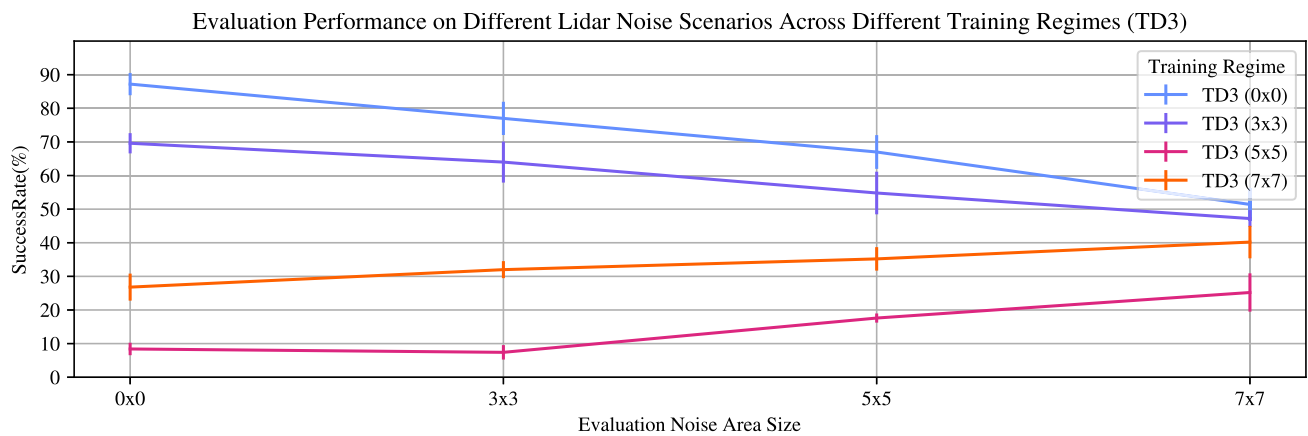


**Fig. 12** TD3 policies trained on different Lidar noise regimes evaluated on the default map (with random goal) with varying noise area sizes. The policies were evaluated for 100 steps 5 times, and the plot shows the mean and standard error across those 5 repeated runs

### 3.4 Algorithm Training Comparison (Camera Observation)

As described in Section 2.2, different RL methods are trained on the *DRL-Robot-Navigation* environment: TD3, PPO, Recurrent PPO (or PPO-LSTM), and DreamerV3.

Figure 13 shows the averaged episode reward during training for the different algorithms, across 500,000 steps of training. For the static goal scenario, TD3, PPO, and PPO-LSTM converge to a solution quickly, each one reaching 0.5 mean reward in less than 100,000 steps.

In general, all of the models converge within less than 100k steps for the static goal scenario. The difficulty of the scenario is now increased by randomizing the position of the goal around the map. This requires the models to learn better

obstacle avoidance and explore the map, as the goal may be hidden behind an obstacle. Each of the models was trained on the best hyperparameters as described in the Appendix A (apart from DreamerV3, which does not require hyperparameter tuning).

Figure 14 shows the averaged episode reward during training for the different algorithms, across 2,000,000 steps of training. DreamerV3 is the most sample-efficient in convergence, achieving it in 200k steps. TD3 is the second most sample efficient, converging in 800k steps. PPO converges somewhere between 1m-2m steps. PPO-LSTM does not converge.

However, the graph only considers the steps and not wall time of the training. The wall time taken to train each model differs due to the nature of each algorithm: PPO is an on-

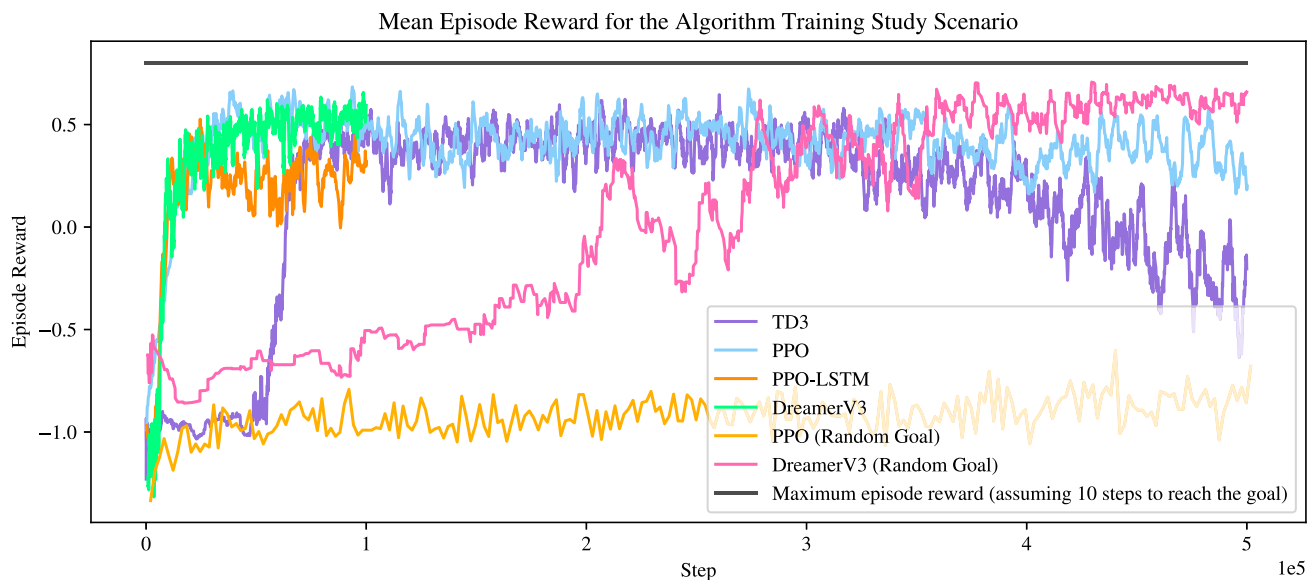


**Fig. 13** Convergence of different algorithms during training for the camera end-to-end, static goal scenario

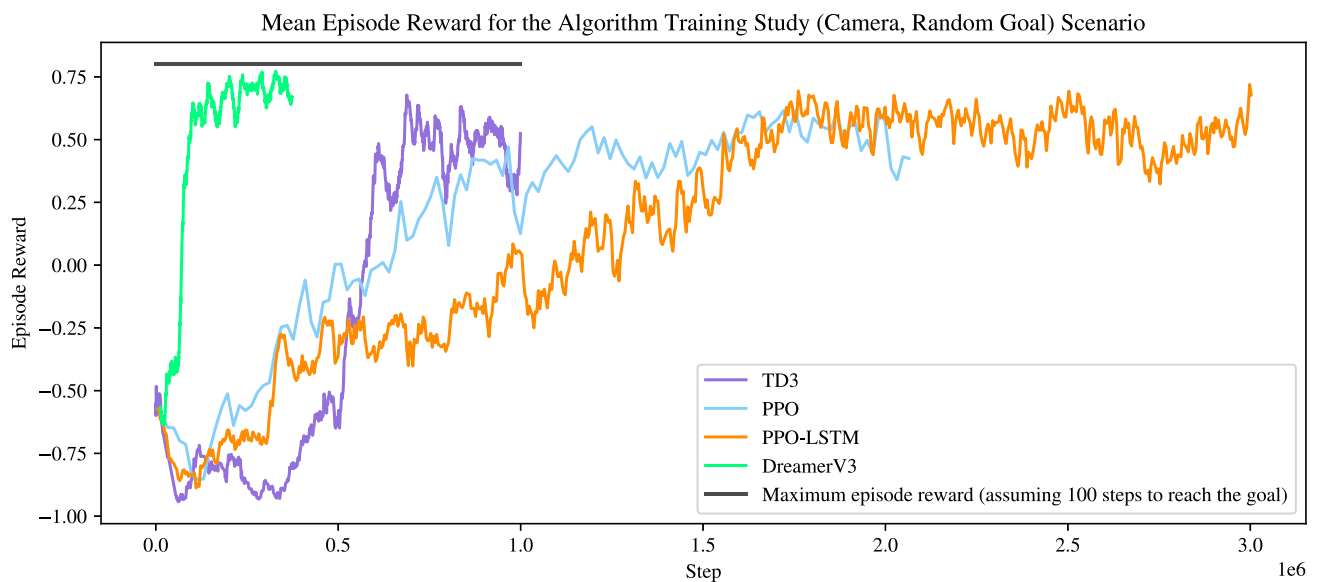


**Fig. 14** Convergence of different algorithms during training for the camera end-to-end, random goal scenario

policy algorithm that only uses the samples generated from the current policy for training. Because of that, combined with us speeding up the simulation (compared with the original Cimurs et al. [9] implementation), it was able to quickly sample steps with a policy before being updated. TD3 is off-policy and is updated every step (albeit, this can be adjusted). Because of this, it is bottlenecked by the gradient updates performed after each step. Hence, whilst it was more sample efficient compared to PPO, it was actually slower to converge in wall-time terms.

PPO-LSTM, like with the Lidar state observation, was hypothesized to perform well on this scenario because of the additional recurrent layers that should enable long-term planning. This was based on the results presented by Mirowski et al. [8], where they find that adding LSTMs to their model improved the performance of their camera navigation. Although the nature of the navigation environment is different, it is interesting that we do not see the benefits of recurrence in our experiments. A possible reason for not seeing PPO-LSTM converge for the camera observation is similar to those outlined for the Lidar observation in Section 3.2, in that it is sensitive to the batch size and the number of steps in the environment. Further, detailed studies of the batch size on the environment may lead to PPO-LSTM convergence.

DreamerV3 was even more sample efficient than TD3 and comparable to PPO in terms of the clock time. It did not require hyperparameter tuning, while PPO and TD3 did. The improved sample efficiency could be due to the learned world-model using an autoencoder structure.

Figure 15 shows the same training runs as Fig. 14, but with the wall time taken to train the model in seconds on the x-axis, instead of environmental steps. PPO is the fastest to converge, achieving it in less than 10,000 seconds. DreamerV3 is slower to converge but achieves convergence in a comparable time. PPO-LSTM and TD3 are much slower to converge, taking more than 30,000 seconds.

This is because the PPO-LSTM requires a smaller batch size and takes longer to train as it updates the LSTM layers. TD3 is slower because it updates every step. Whilst this is sample efficient (it converged quicker than PPO in step terms), it leads to slower wall time training, showing the trade-off between on-policy and off-policy methods. Whilst in the original Cimurs et al. [9] simulations, the TD3 algorithm was advantageous due to sample efficiency. As we were able to increase the speed of the simulation, this led to the on-policy PPO benefiting more than TD3. It should be noted that while PPO was quicker to converge compared to DreamerV3, PPO required hyperparameter tuning (which required multiple prior runs) whilst DreamerV3 did not.

Figure 16 shows the training of TD3, PPO, and DreamerV3 algorithms on vanilla (0x0) and adversarially perturbed (3x3) scenarios. For DreamerV3 and PPO, the convergence appears delayed in the adversarial scenarios, and the convergence reward appears to be lower (this could be due to requiring more steps to navigate, or a higher rate of collisions). Whilst TD3 is able to converge on the vanilla (0x0) scenario, it does not converge at all on the adversarial (3x3) scenario. This is consistent with the Lidar modality results from Sections 3.2 and 3.3, where TD3 training in adversarial

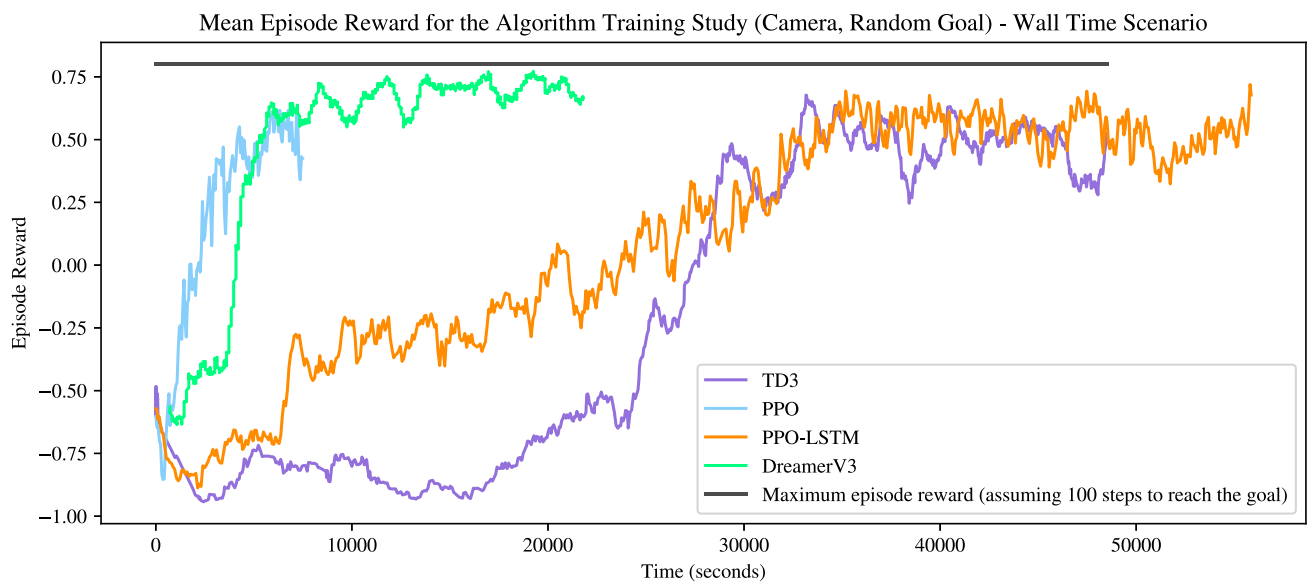


**Fig. 15** Convergence of different algorithms during training for the camera end-to-end, random goal scenario, with wall time to train each model on the x-axis

training environments led to less robust models compared with vanilla training.

Interestingly, this finding holds for different modalities. Again, this is likely due to the off-policy nature of TD3, where the adversarial states remain in the buffer and affect the training every update. Whilst this results in the algorithm being more sample efficient, it appears to also negatively impact it when training in adversarial environments, something that we did not see with PPO.

### 3.5 Camera Navigation with Sensor Denial, Static Goal

Figure 17 shows the evaluation results of the different algorithms, TD3, PPO, PPO-LSTM, and DreamerV3, using the camera observation space. The algorithms are trained on the vanilla 0x0 map, and evaluated on environments of increasing camera denial area: 0x0 (sensor denial), 3x3, 5x5, and 7x7 - referring to the size of the rectangle of the sensor denied

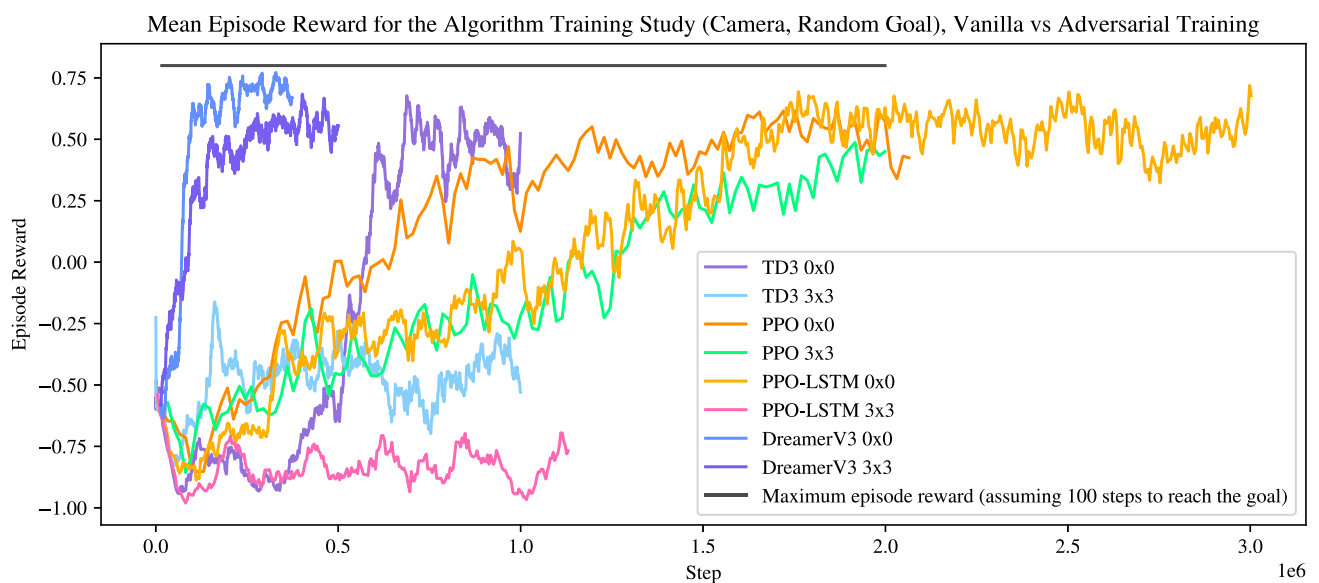


**Fig. 16** Convergence of different algorithms during training for the camera end-to-end, random goal scenario, trained on 0x0 and 3x3 perturbed environments

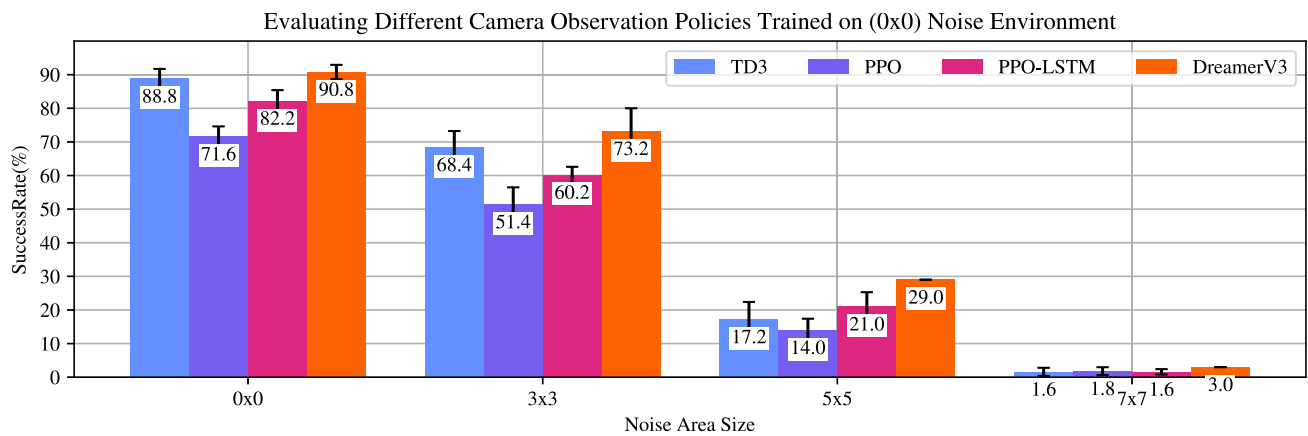


**Fig. 17** Camera noise area study for algorithms trained on the vanilla environment with no sensor denial. The algorithms were then evaluated on the vanilla environment with changing noise area sizes, as shown on

the x-axis. The evaluation was performed for 100 episodes and repeated 5 times to establish a standard deviation

area in meters. On the 0x0 evaluation scenario, DreamerV3 achieved the highest mean success rate of 90.8%, with TD3 achieving a similar performance of 88.8%. PPO-LSTM was third with a mean of 82.2% and PPO performed the worst with a mean success of 71.6%. On the 3x3 evaluation scenario, the pattern remains the same, with DreamerV3 achieving the highest mean of 73.2% (with a much larger standard error of 6.8), TD3 close behind with 68.4%, PPO-LSTM third and PPO last. The evaluation scenarios for 5x5 and 7x7 show severe degradation in performance across all the algorithms.

Similarly, Fig. 18 shows the evaluation results of the algorithms trained on a map with random 3x3 denied areas. When evaluated on the 0x0 scenario, a similar pattern emerges with DreamerV3 achieving the highest mean success rate of 86.4% (albeit, this is a performance drop compared with 90.8% when trained on the clean scenario). TD3 is second

with 82.4%, and PPO and PPO-LSTM show the worst performance, achieving 71.0% and 72.4%, respectively. All of the policies show performance drops compared to policies trained on the clean (i.e. no sensor denied areas) environment.

When evaluated on the 3x3 scenario, DreamerV3 performs the best, achieving 80.6% mean success rate. This is an improvement from the vanilla model (from 73.2%). TD3 mean success rate is 73.2%, PPO 69.2%, and PPO-LSTM is last with 58.6%. Likewise, TD3 and PPO also show improvements from the vanilla models. PPO-LSTM is the only model that did not report an improvement from the vanilla model.

All models report an improvement in evaluation performance on 5x5 and 7x7 noise area maps (compared with models trained on no noise in the environment), showing that training on some noise in the scenario generally improves the

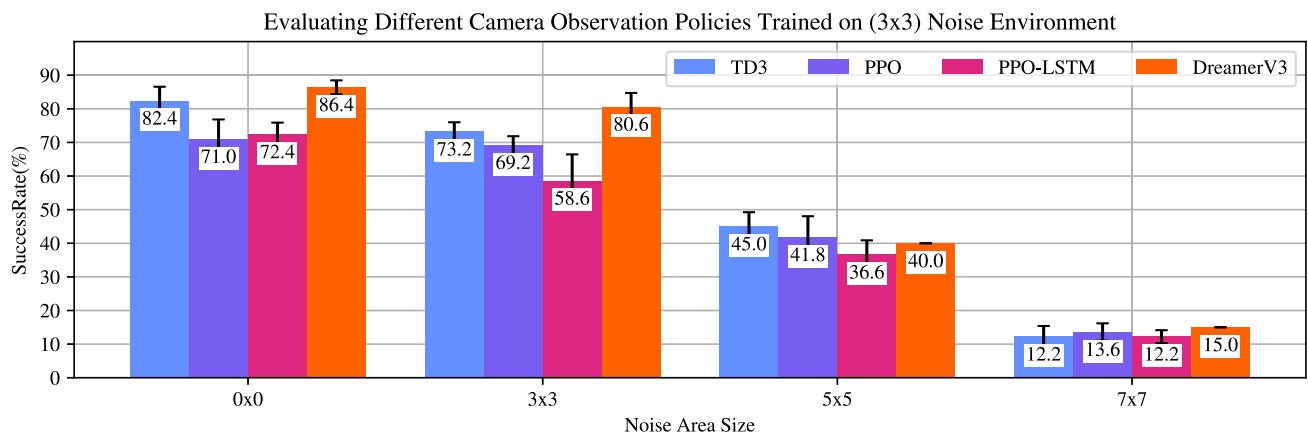


**Fig. 18** Camera noise is studied, for algorithms trained on the default environment with 3x3 sensor denial zones. The algorithms were then evaluated on the default environment with changing sensor denial area

sizes, as shown on the x-axis. The evaluation was performed for 100 episodes and repeated 5 times to present a mean and standard error

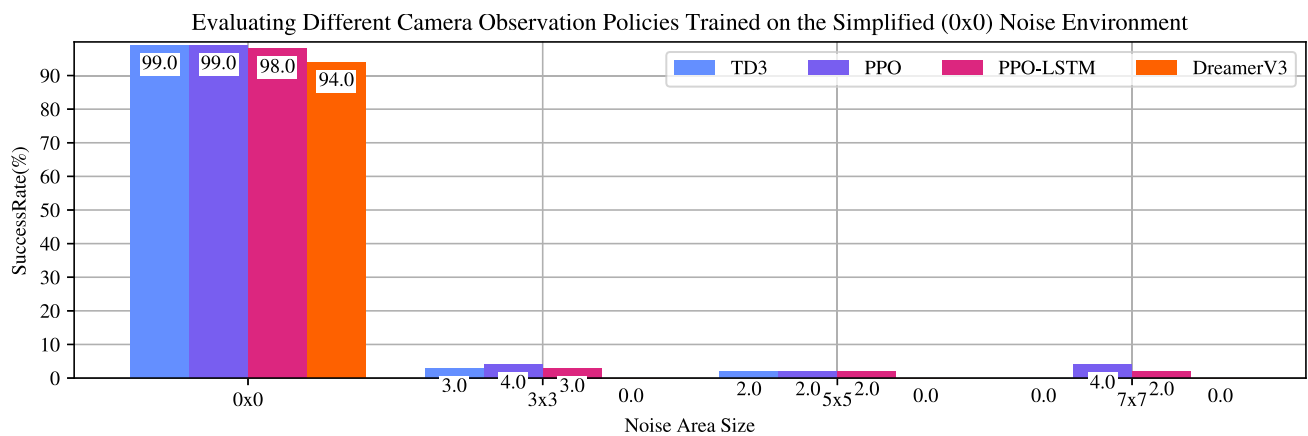


**Fig. 19** Camera noise area study for algorithms trained on the default environment with 0x0 noise. The algorithms were then evaluated on the simplified map with changing noise area sizes, as shown on the x-axis. The evaluation was performed for 100 episodes

evaluation performance on noisy evaluation scenarios. Still, these improvements are not so significant that the models are reliably achieving the goal. The downside of training the model on the 3x3 scenario with sensor-denied areas is that these models appear to suffer from performance degradation when evaluated on the 0x0 scenario.

### 3.6 Camera Navigation with Sensor Denial (No Obstacle Map, Static Goal)

Section 3.5 shows that the training regimes have an effect on policy behaviour in sensor-denied areas. To better understand this behaviour, they are evaluated on the simplified, no-obstacle map, as described in Section 2.4.

Figure 19 shows the evaluation results of the different algorithms, TD3, PPO, PPO-LSTM, and DreamerV3, using the camera observation space. The algorithms are trained on the vanilla 0x0 (no sensor denial) default map, and evaluated on the simplified, no obstacle map, with variations of

increasing, static camera denial area which is located in the centre of the map: 0x0 (no sensor denial), 3x3, 5x5, and 7x7 - referring to the size of the rectangle of the sensor denied area in meters. All four of the models exhibit similar behaviour of successfully navigating in nearly all of the episodes on the 0x0 evaluation, but then consistently failing when any noise is introduced (with a few successes likely caused by spawning near the goal).

When evaluated on a sensor-denied area, the perturbed state is out-of-sample for the trained policy as the policy has not seen these states during training. Hence, all of the models fail to perform when the sensor-denied area is introduced. Hence, by training the policies with the sensor-denied areas present, we are introducing these previously out-of-domain samples into the training.

Similarly, Fig. 20 shows the evaluation results of the different algorithms trained on the default map with random 3x3 noise areas. The 0x0 evaluation does not significantly change, with TD3, PPO, and DreamerV3 still successfully

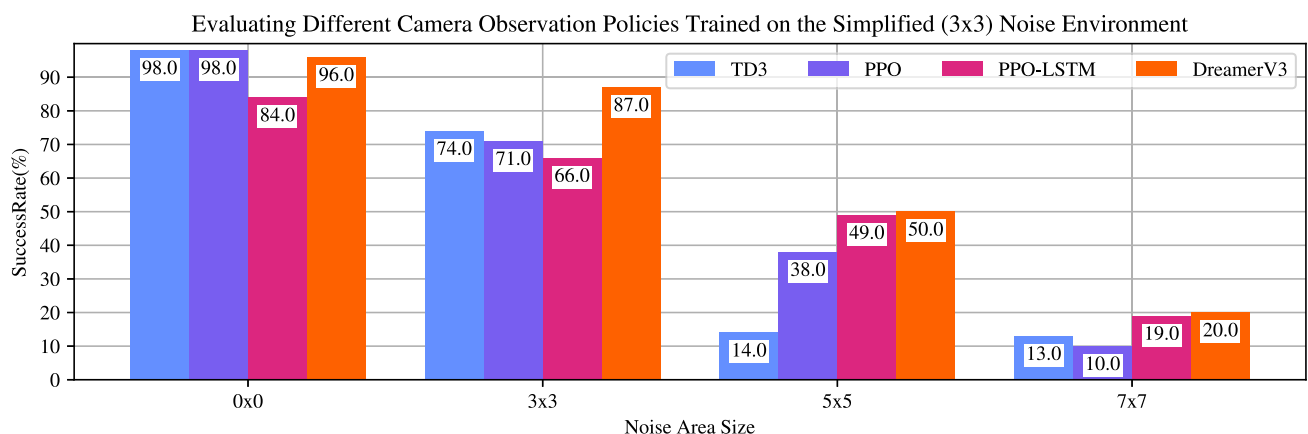


**Fig. 20** Camera noise is studied, for algorithms trained on the default environment with 3x3 noise. The algorithms were then evaluated on the simplified map with changing noise area sizes, as shown on the x-axis. The evaluation was performed for 100 episodes

navigating to the target 98, 98, and 96 times out of 100. For the 3x3 evaluation, TD3 reached the goal 74 times, PPO 71 times, PPO-LSTM 66 times, and DreamerV3 showed the best performance with 87 times. This is a significant improvement over the models trained on the 0x0 (no-noise) environment - a sign that the policies have learned to navigate to the goal with noise present. The 3x3-trained models also show performance improvements over the 0x0-trained models when evaluated on 5x5 and 7x7 scenarios. However, they do not achieve high (90+) success rates, which would be an indication of the policies truly solving the scenarios.

Compared with vanilla policies, the adversarially trained policies can perform despite sensor-denied areas being present in the evaluation because the policies do not consider the sensor-noise areas to be out-of-domain anymore. Hence, their performance increases across the board.

Figure 21 shows the PPO policies trained on different camera noise regimes evaluated on the no-obstacle map. It shows that the PPO (0x0) and PPO (3x3) perform the best when evaluated on the 0x0 scenario, achieving 99 and 98 successful episodes. The PPO (0x0) fails to navigate to the goal when evaluated on any noise maps. PPO (5x5) and PPO (7x7) both failed to navigate successfully in the 0x0 evaluation, a sign that too much noise hindered the learning of those models.

Figure 22 shows the 100 path plots generated during the evaluation of 2 different policies (PPO 0x0 and PPO 3x3) on two different evaluation maps (0x0 and 3x3 static sensor denial zones, static goal in the centre, and no obstacles present). Figure 22(a) shows the vanilla PPO evaluated on the 0x0 map. The same vanilla policy is evaluated on a map with a static 3x3 noise area around the goal in Fig. 22(b). The policy is unable to navigate to the goal and shows a significant decrease in performance (from 100 successes to 5 successes). When the robot enters the noise area, the camera fails, and the robot stops moving. This can be seen as many paths end inside the blue noise zone and do not continue.

In contrast, when adversarially trained with 3x3 random areas present in the environment and evaluated on the 3x3 map as shown in Fig. 22(d), the number of successful episodes increases to 75. When inside the blue sensor denied area, the agent simply moves in circles, as can be seen by path traces in the blue zone. This is not a reliable way of navigating with obstacles present in the environment, but it leads to success when no obstacles are present in the zone of sensor denial. The policy exploits the reward structure of the scenario, opting for a high-risk strategy of moving through the sensor-denied areas. Hence, like the Lidar examples shown in Fig. 11, the policies have learned a high-risk strategy of blindly navigating when sensor readings are uncertain.

Since we are evaluating on the simplified map, no obstacles are present. Clearly, once in a sensor-denied area, the policies have no way of avoiding obstacles, and this is deliberately not reflected in these results, in order to understand the policy behaviour in sensor-denied zones. Still, because the models were trained on the default map, they have learned to take risky actions when in sensor-denied zones with obstacles present. This likely means that due to the reward shaping of the environment, it was more rewarding for the policy to take the high-risk actions. However, this only holds up with small amounts of sensor-denied zones in the environment, as shown in Fig. 21, as larger amounts of noise result in failure to learn basic navigation behaviour.

### 3.7 Camera Navigation with Sensor Denial, Random Goal

As the vision models were shown to converge for the sensor observation on the random goal scenarios, they are evaluated on the random goal map with obstacles (as with the Lidar configuration).

Figure 23 shows the evaluation results of the different algorithms, TD3, PPO, PPO-LSTM, and DreamerV3, using

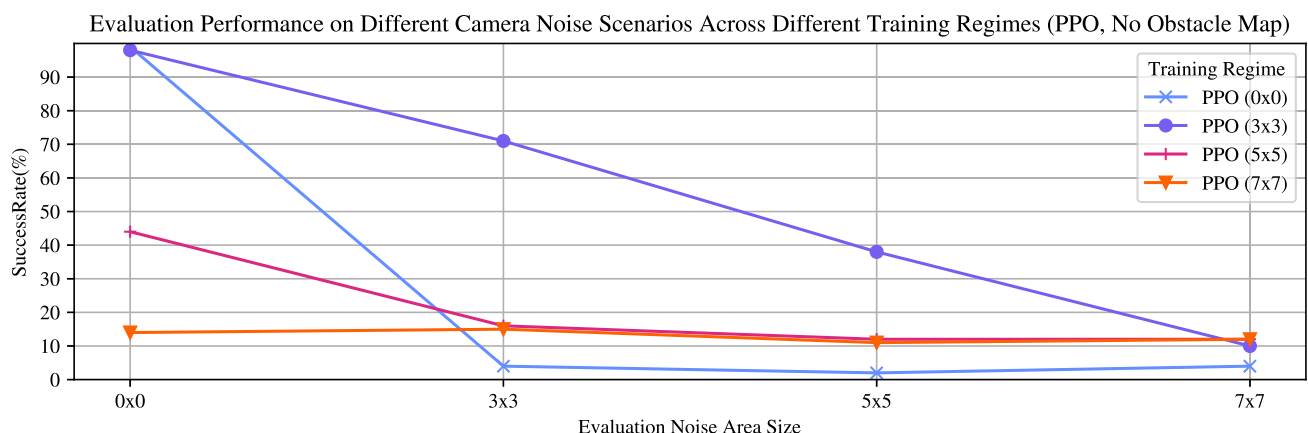


**Fig. 21** PPO policies trained on different noise regimes evaluated on the simplified map with varying noise area sizes

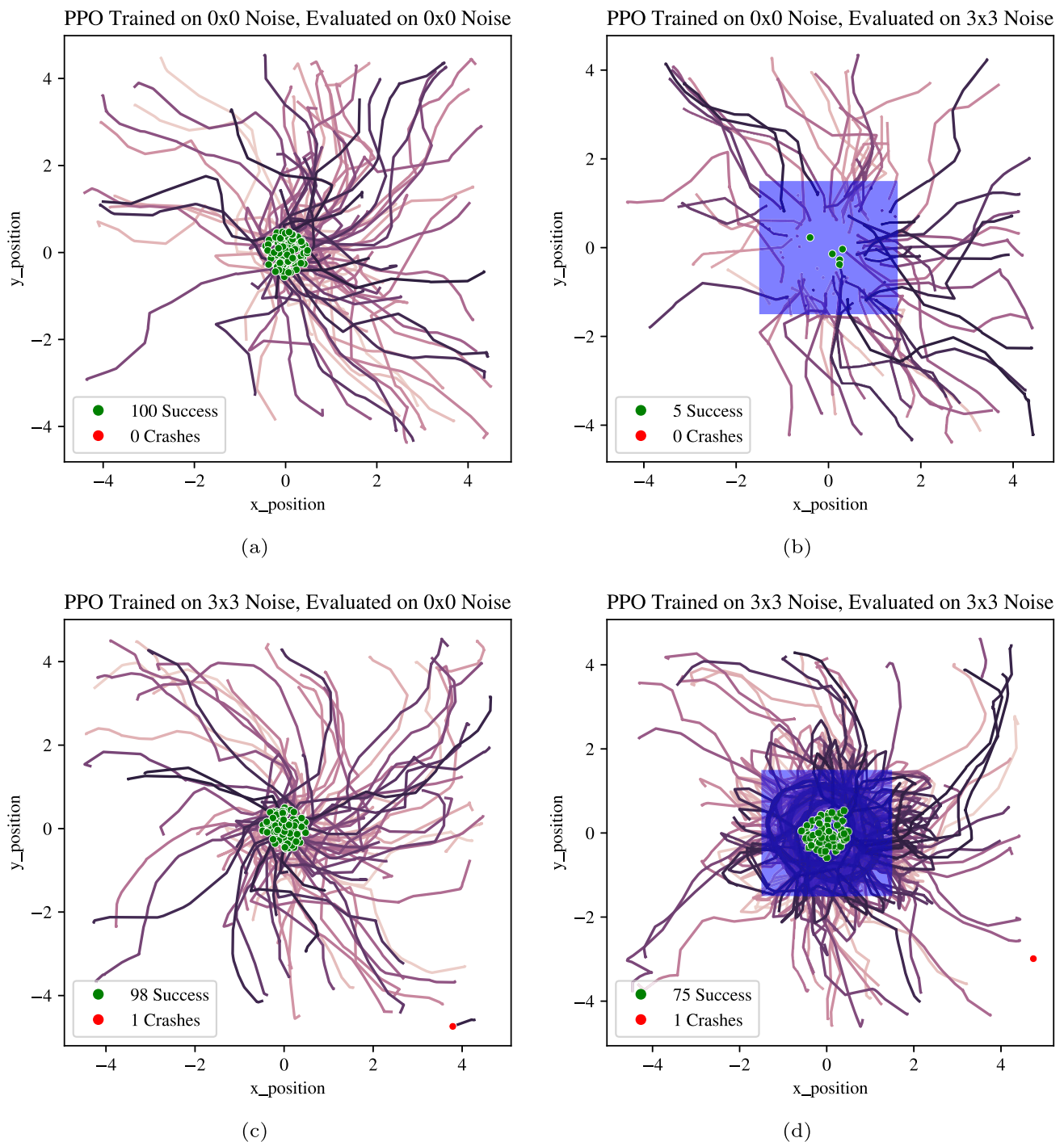


**Fig. 22** 100 path plots generated during evaluation by vanilla policy (PPO 0x0) evaluated on 0x0 map (a), vanilla policy (PPO 0x0) evaluated on 3x3 sensor denial map (b), adversarially trained (PPO 3x3) evaluated on 0x0 sensor denial map (c), and adversarially trained (PPO

3x3) evaluated on 3x3 sensor denial map (d). The red dot shows a collision, green dot shows a successful navigation to the goal. Blue zone shows the area of sensor denial

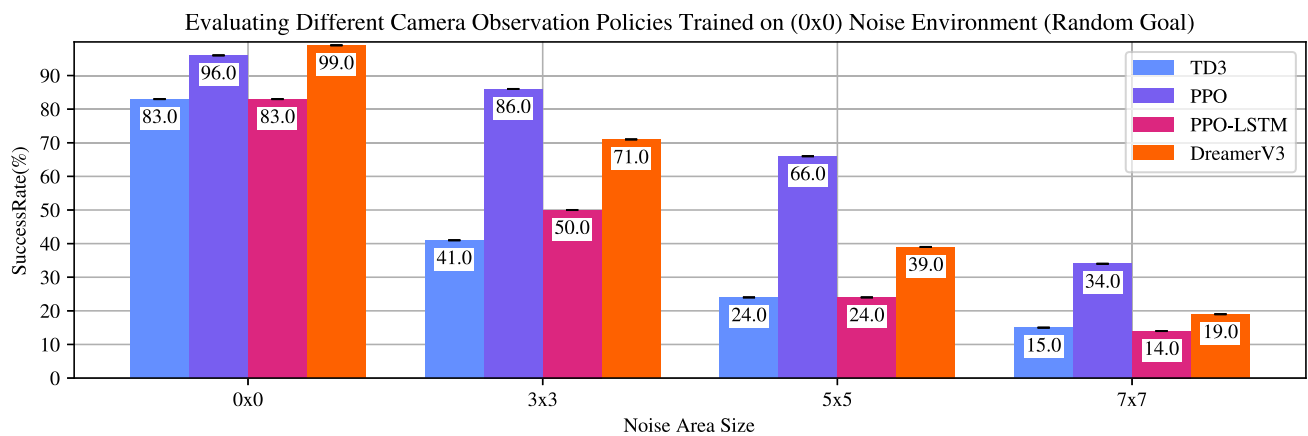


**Fig. 23** Camera noise area study for algorithms trained on the random goal vanilla environment with no sensor denial. The algorithms were then evaluated on the vanilla environment with changing noise area sizes, as shown on the x-axis

the camera observation space on the random goal scenario. The algorithms are trained on the vanilla 0x0 (no sensor denial) default map, and evaluated on the more complex random obstacle map with variations of increasing, static camera denial area which is located in the random positions of the map: 0x0 (no sensor denial), 3x3, 5x5, and 7x7 - referring to the size of the rectangle of the sensor denied area in meters.

All four of the models converged to a solution when no noise was present during training, as seen in Section 3.4. However, PPO and DreamerV3 consistently outperform TD3 and PPO-LSTM models, implying better convergence as they are closer to being fully successful on the vanilla map, with no collisions or scenario terminations. As the noise size increases, both PPO and DreamerV3 continue to outperform TD3 and PPO-LSTM. In general, the trend continues as the noise area increases. In previous results, we have seen that better convergence on the vanilla scenario generally leads to better results overall, as the agent is able to perform well on

scenarios where it gets lucky and does not come across any areas of sensor denial. Although it should be noted that there is some randomness in these results, as the noise areas are random during the evaluation.

Similarly, Fig. 24 shows the evaluation results of the different algorithms trained on the default map with random 3x3 noise areas. However, as the TD3 and PPO-LSTM models did not converge when trained with 3x3 noise areas, only the PPO and DreamerV3 models are compared. The 0x0 evaluation does not significantly change, with PPO, and DreamerV3 still successfully navigating to the target over 90 times out of 100. DreamerV3 and PPO remain robust on the no noise evaluation, successfully navigating 90+ times. Further, as the noise area size increases, the performance of both models is better compared to the vanilla equivalents. Even with a 5x5 noise area size, both models achieve 70+ navigation targets, which is comparable to TD3/PPO-LSTM without any sensor denial areas. This shows that including the noise areas during

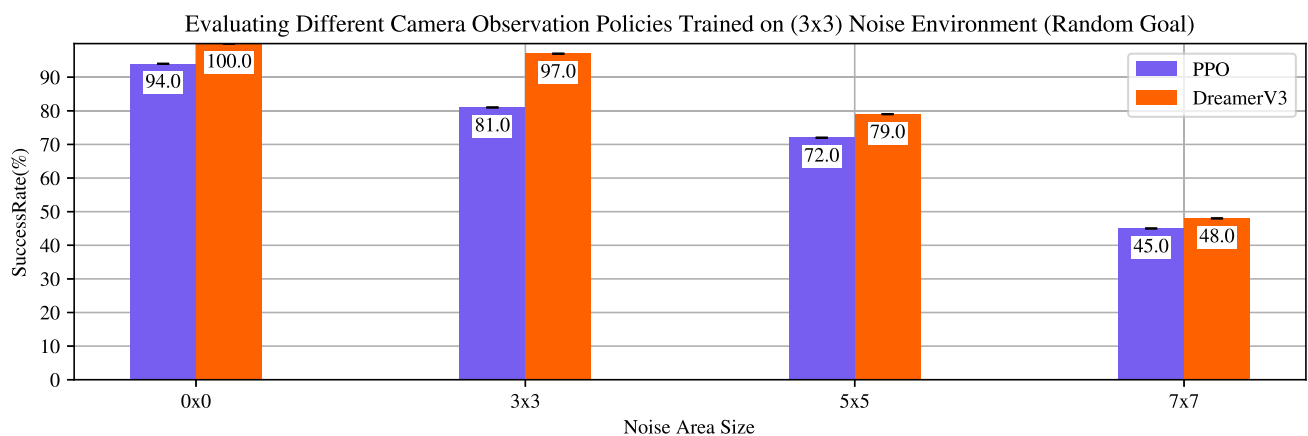


**Fig. 24** Camera noise area study for algorithms trained on the random goal 3x3 sensor denial environment. The algorithms were then evaluated on the vanilla environment with changing noise area sizes, as

shown on the x-axis. PPO-LSTM and TD3 did not converge on the 3x3 sensor denial scenario, so they were omitted

training improved the performance of the models, especially on other adversarial scenarios. However, this should be balanced with the fact that the TD3 and PPO-LSTM algorithms failed to converge because of the noise areas present during training. While we suggested that the reason for the failure of TD3 convergence may be due to its off-policy nature, PPO-LSTM remains on-policy, but fails to perform equivalently to PPO. This counters our hypothesis for including PPO-LSTM in the first place, as we expected it to outperform standard PPO due to the dynamic nature of this task. We also expected the recurrent models to potentially perform better in the presence of the adversarial perturbations, because they may be able to use recurrence to judge whether they should continue to move (if they, for example, saw the goal before entering the sensor denial zone).

DreamerV3 is recurrent and uses autoencoders to create a world model of the environment. It outperforms PPO on the 3x3 evaluation, which was used to train, achieving 97 out of 100. Combined with the fact that DreamerV3 required the least computing resources as it did not require hyperparameter tuning, we conclude that DreamerV3 is the most robust and usable model for adversarial navigation.

Figure 25 contains example observations as the agent successfully navigates around the maze using a camera as the only sensor with a random goal around the map. To better understand the learned behaviour of the agent, latent states are saved during the evaluation. As they are still highly dimensional (512 features), we use t-SNE, a non-linear dimensionality reduction method, to reduce the latent features to 2 dimensions and plot them with the aim of interpreting their meaning and differences between the vanilla and adversarial models. Similar methods were used by Zahavy et al. [60], although they used a principal component analysis to reduce the dimensionality from 512 to 50 before using t-SNE, and their network was a DQN, while we use PPO. We plot the reduced t-SNE dimensions for each state alongside the estimated value by the network.

Figure 26 shows the t-SNE plot of 5,000 of the vanilla PPO during the evaluation on the vanilla environment. The high-value states are very sparse, and for most states, the estimated value is 0.00 or less. This is the result of training the model on a sparse environment where only a few states (ones where the goal is in view) are deemed valuable. Other states are estimated to be around 0.0 or less (in most of the states, the agent is at risk of crashing into a wall). It shows the network has learned the underlying rules of the environment

- colliding with the wall is -0.5, and reaching the goal leads to a sparse reward of 1.0. Figure 26(b) shows observations used to generate each latent point (in the center of the image). The graph shows clusters of geometrically similar states together. For example, at the bottom left, the states show the red-brick cuboid wall from different angles. The top right shows the shadowy wooden wall. Top left shows a cluster around the table - a challenging object in the environment, as the small table legs lead to a collision and are quite hard to see on the camera. Whilst this is a qualitative description of the latent states, we can use it to compare latent states of the vanilla and adversarial models to see any differences in the learned states.

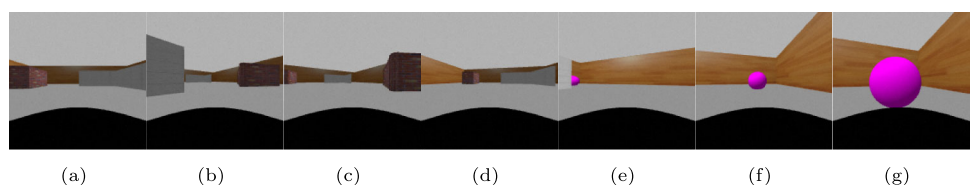
Figure 27 shows the vanilla model evaluated on the 3x3 sensor-perturbed environment. A clear cluster appears in the middle of the graph. When we overlay the images on top of the graph in Fig. 27(b), we can see that the clustered observations come from adversarial states. A high-value cluster appears on the top-right of the graph. When we overlay the observations on top, we can see that these come from states where the goal is clearly visible.

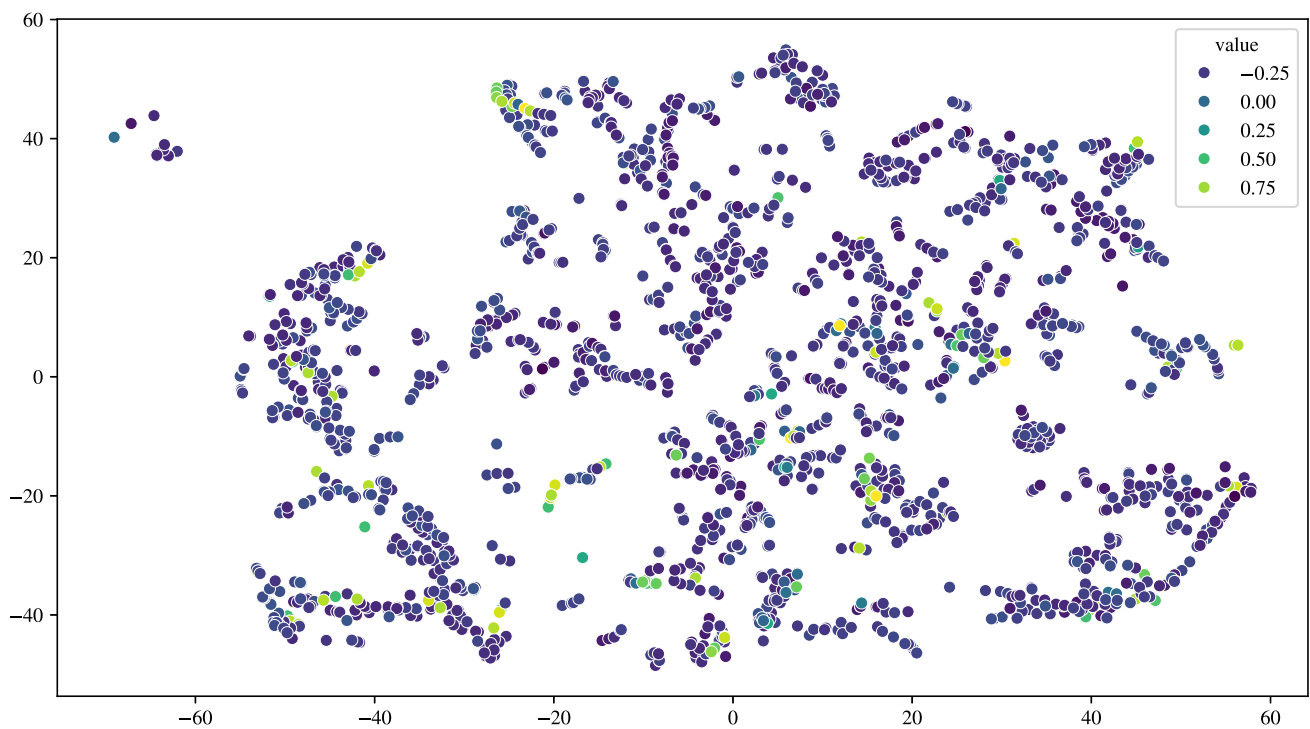
As t-SNE clusters points together by pushing similar points together and pulling different points apart, this visualization suggests that the perturbed states are treated as out-of-domain samples. This occurs because the model being evaluated has not seen any perturbed states during training and hence has not learned to correctly value these states. The hue plot shows the values to be estimated to be around 0.0, which might be an incorrect estimation of the state, given it has not seen these states during training.

Figure 28 shows the equivalent plot for the PPO model trained on the 3x3 sensor perturbation environment and evaluated on the same environment. There no longer appears to be a clear, distinctive cluster in the graph, as in Fig. 27. This is because the model was trained on the perturbed states, hence the model would have had a chance to learn their value during training. By overlaying the observations on top in Fig. 28(b) we can see that the perturbed states still appear as a small cluster at the bottom left, but they are less distinct.

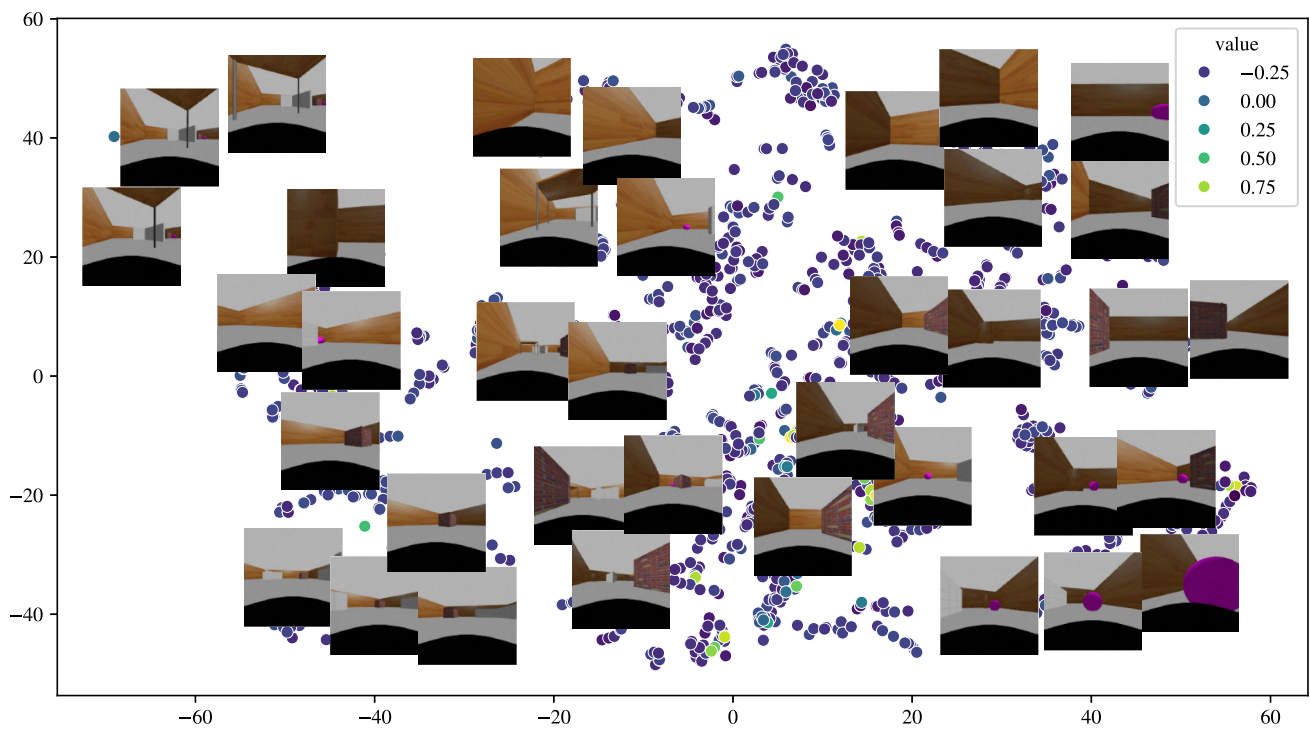
We conclude that by including sensor perturbations in the environment, the model can better estimate the value of the perturbed states. However, this poses a challenge, in that by including sensor perturbations during training, it makes it potentially harder to detect them as out-of-distribution states during evaluation. In safety-critical systems, it might be the case that sensor anomalies should be treated by

**Fig. 25** View from the robot camera as the PPO model navigates around the maze with no adversarial perturbations



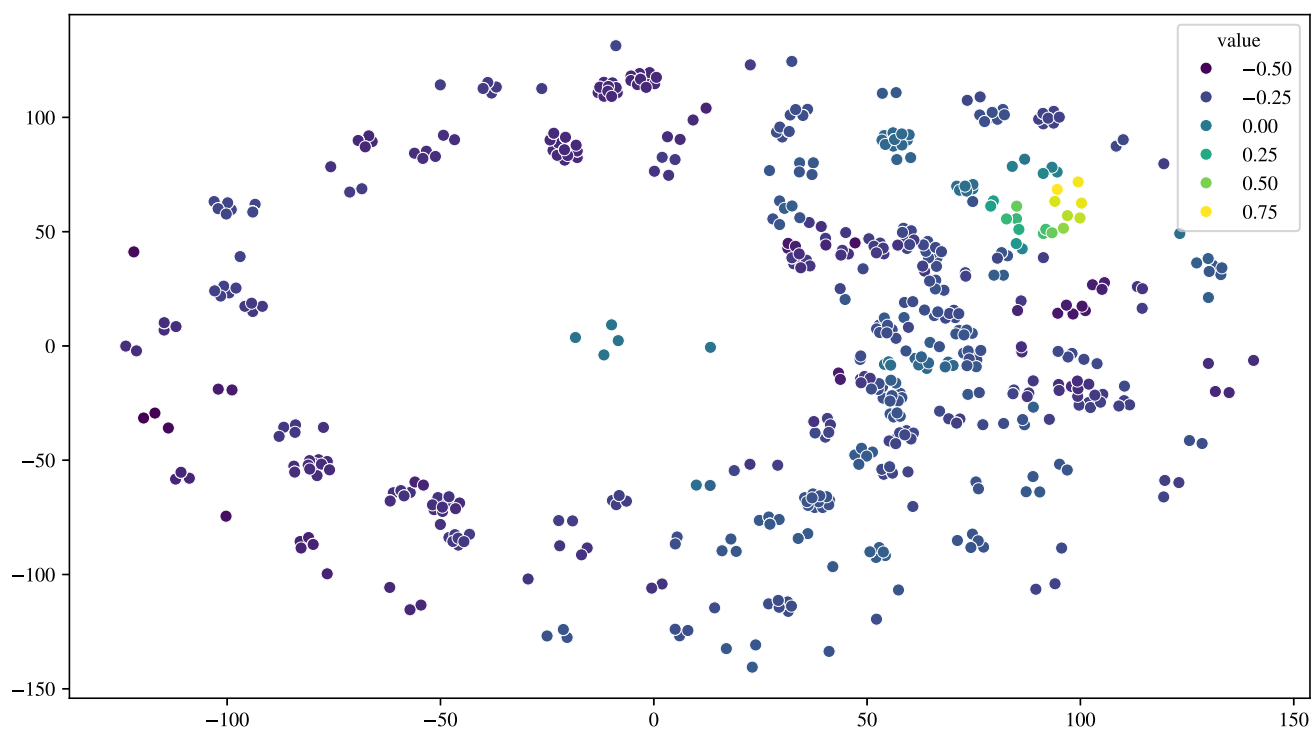


(a)

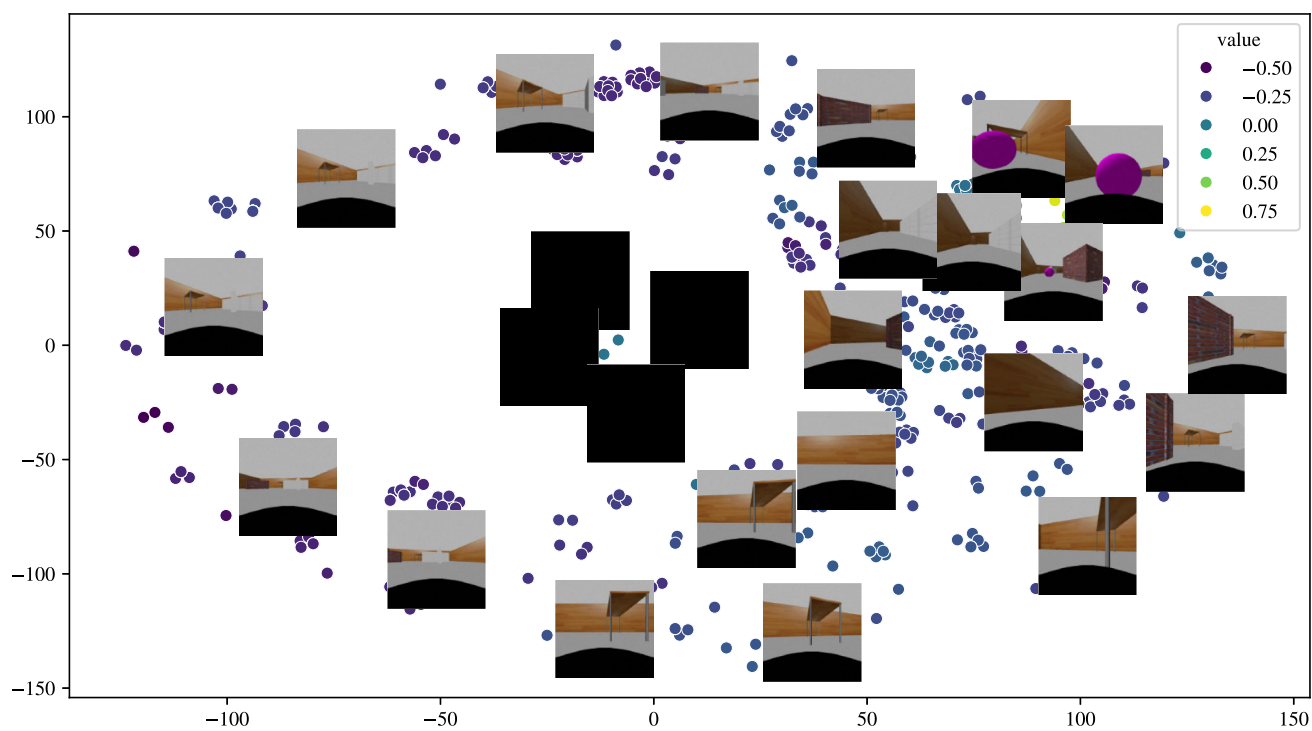


(b)

**Fig. 26** T-SNE plot of 5,000 states during evaluation **(a)** with the hue referring to the estimated value of the state, and the observation used to generate the latent space overlaid on top of the **(b)**



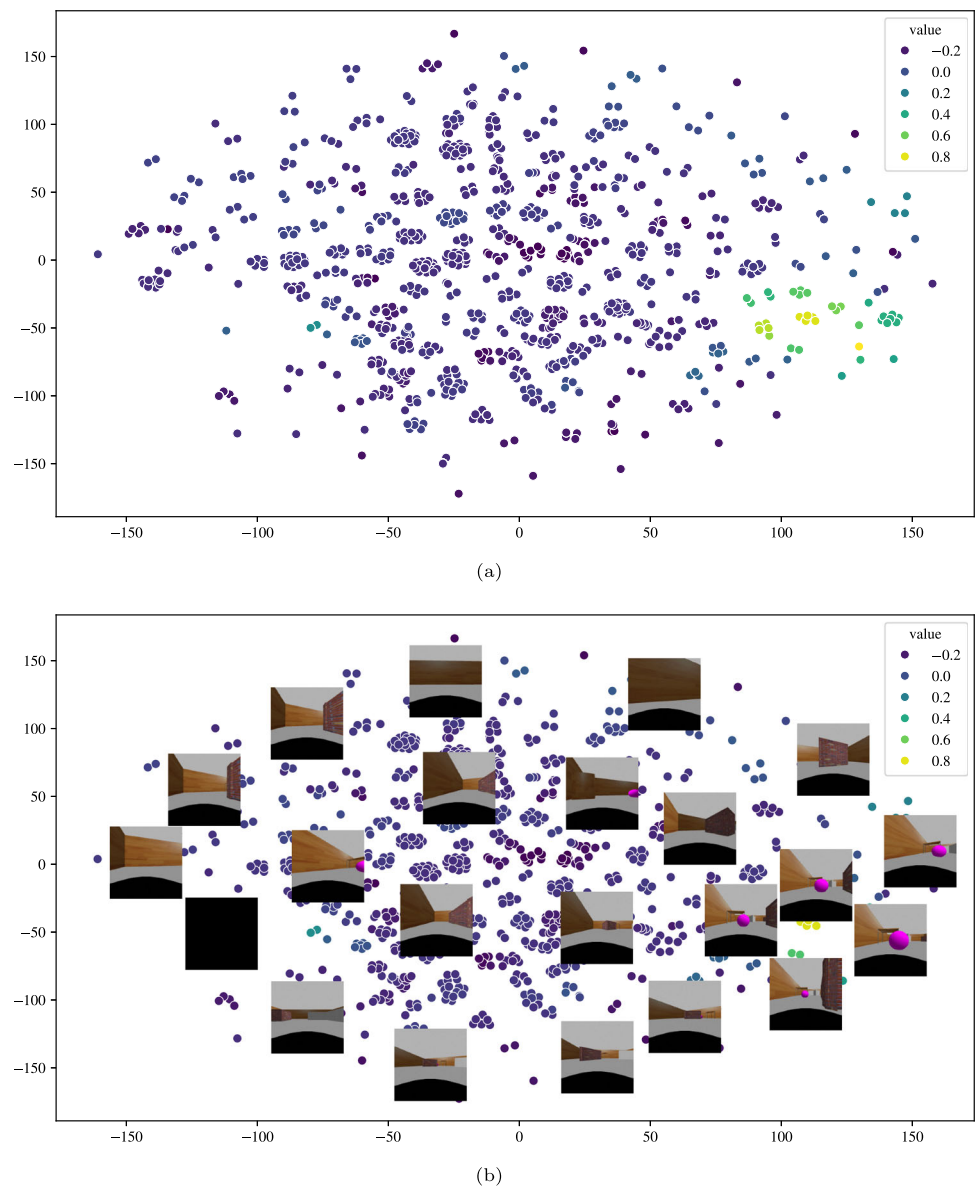
(a)



(b)

**Fig. 27** T-SNE plot of PPO trained on 0x0 sensor perturbed environment and evaluated on 3x3 sensor perturbed environment **(a)** with the hue referring to the estimated value of the state, and the observation used to generate the latent space overlaid on top of the **(b)**

**Fig. 28** T-SNE plot of PPO trained on 3x3 sensor perturbed environment and evaluated on 3x3 sensor perturbed environment (a) with the hue referring to the estimated value of the state, and the observation used to generate the latent space overlaid on top of the (b)



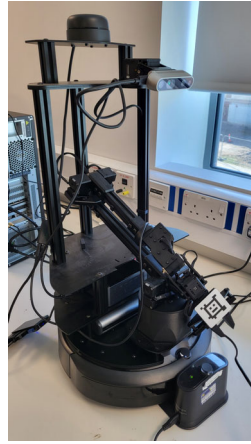
being detected as out-of-distribution samples. In our case, the model has learned to value the perturbed state at 0.0 - likely because a small portion of the experiences led to the robot achieving the goal (if the sensor area appeared close the goal). Again, for safety-critical systems, the high valuation of the sensor perturbed state might be problematic, as the agent might choose to take unsafe actions (which may lead to e.g. crashing into a wall in our environment). An alternative strategy could be to simply train the model on the vanilla, unperturbed environment, and treat any out-of-distribution samples as sensor failures, and use an alternative sensor arrangement (which in our case could be Lidar) for robust navigation. The multi-modal setup would have to be considered in such a case - although all the states could be

concatenated together, this might not lead to optimal performance in case of perturbations.

### 3.8 Real World Experiments

To test whether our trained deep reinforcement learning (DRL) models can be deployed on real robots, we conduct real-world experiments. We used a LoCoBot (Fig. 29) platform comprising a Create 3 mobile base (35 cm in diameter and 62 cm in height) equipped with an Intel NUC Mini PC (8th Gen Intel Dual-Core i3, 8GB DDR4 RAM, 240GB SSD) and an Intel RealSense D435 RGB-D (but only used RGB as state input) camera, mounted on a pan-tilt mechanism that for our experiments stayed static and with a slight

**Fig. 29** LoCoBot WidowX-250 6 DOF on its charging station



downward angle. The camera, mounted approximately 48 cm above the mobile base's top, provides high-resolution color ( $1920 \times 1080$ ) imagery for precise environmental perception. The images captured by the Intel RealSense D435 RGB-D camera were resized from their original resolution of  $1920 \times 1080$  pixels to  $160 \times 160 \times 3$  RGB images to match the input dimensions required by our models. A MAXOAK K2 50,000mAh Power Bank provided power for extended operation during experiments. Although our platform includes a WidowX 250 6-DoF robotic arm capable of complex manipulation tasks, we did not utilize the arm in the current set of experiments, as our focus remained on tasks related specifically to visual-based mobile navigation capabilities.

The experimental trials were carried out in a controlled indoor arena (Fig. 30), constructed from cardboard panels measuring approximately  $5 \times 5$  meters. This enclosed space was specifically designed to evaluate robot navigation and perception in a structured environment with defined boundaries. Our real-world experimental arena is designed to mirror the simulated environment, replicating key aspects of the simulated environment, such as dimensions, obstacle shapes, and spatial arrangement. The arena contained four distinct obstacles: a cross-shaped cardboard structure with each panel measuring 85 cm in width and 60 cm in height, a small box obstacle of dimensions  $30 \times 30$  cm base and 60 cm height,

a white rectangular obstacle sized  $60 \times 30$  cm base and 90 cm height, and a white triangular-shaped obstacle featuring a  $60 \times 60 \times 60$  cm triangular base and 90 cm height.

Robot control was implemented using ROS2 (Robot Operating System 2), a middleware that facilitates real-time communication and modularity in robotic applications, leveraging the existing packages, namely the “locobot\_control” ROS2 package provided by Interbotix, including ROS nodes like the “realsense2\_camera\_node”, which facilitates communication, perception, and actuation commands specific to the LoCoBot platform. Building upon these packages, we developed an additional ROS2 node written in Python to manage robot movement commands and sensor inputs. This node utilized the Stable-Baselines3 library [58], which provides implementations of reinforcement learning algorithms, to load already trained policy weights (trained using the PPO algorithm) for real-time action prediction. The trained weights used in the real experiments achieved more than 90% success rate in evaluation in the simulation environment (Fig. 23). Then the model weights were transferred onto the robot's onboard Intel NUC Mini PC. The ROS2 node subscribes to the robot's camera image topic, processes incoming images by resizing them to a  $160 \times 160$  RGB format consistent with the trained model's input dimensions, and then uses the processed images to predict linear and angular velocity commands. These commands were subsequently published at 10 Hz to the robot's velocity command topic (Fig. 31).

The experimental trials consisted of 20 individual runs, in which the robot was tasked with navigating autonomously toward the designated goal (specifically a pink-colored sphere, Fig. 30), randomly placed within the previously described  $5 \times 5$ -meter cardboard arena. For each trial, the robot was manually positioned at random initial locations within the arena to introduce diverse starting conditions. The robot relied exclusively on visual input from the onboard camera (without using the depth measurement) for perception and decision-making, without any additional sensor inputs such as LIDAR or ultrasonic sensors. Each trial was considered successful when the robot navigated and reached



(a)



(b)

**Fig. 30** The experiment arena. It is a controlled indoor arena, constructed from cardboard panels measuring approximately  $5 \times 5$  meters. The arena includes four cardboard obstacles and one navigational goal object, the soft pink sphere

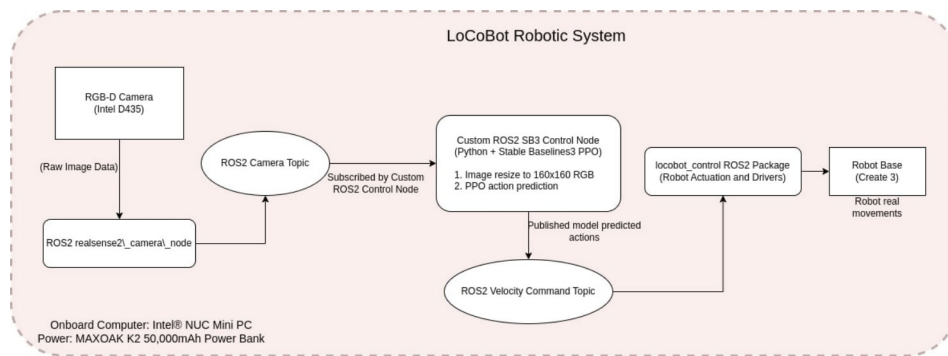


**Fig. 31** System diagram illustrating the information flow within the LoCoBot robotic system. Raw visual data from the Intel RealSense D435 RGB-D camera is processed through ROS2 nodes, including a custom-developed control node using Stable Baselines3 (PPO algorithm). The predicted velocity commands are communicated via ROS2

topics to the existing locobot\_control package, which translates them into physical actuation of the robot's Create 3 base. All computations occur onboard, powered by the Intel NUC Mini PC and the MAXOAK K2 50,000mAh Power Bank

the target goal, and unsuccessful when the robot collided with the environment.

In our real-world experiments (Figs. 32 and 33), the robot successfully navigated to the target in 11 out of 20 trials (55%) (Fig. 32). This represents a noticeable drop from the success rate observed in simulation (Fig. 23). Failures (Fig. 33) were consistently caused by obstacle collisions (9 out of 20 trials), unlike the simulation scenario, where failed episodes also involved the robot becoming stationary or stuck without direct collisions. Variations in environmental conditions, including inconsistent ambient lighting, shadows cast by obstacles, and textures from the cardboard surface, introduced unexpected perceptual challenges that led directly to these collisions. While statistical analysis is limited by our experiment sample size, qualitative observations consistently indicated that collisions were predominantly linked to discrepancies in sensor perception compared to the simulation environment, such as differences in camera intrinsics and environmental textures. Additionally, minor differences in environmental conditions, lighting variations could further impact the model's performance, highlighting challenges commonly encountered when transferring learned navigation policies from simulation to real-world robotic platforms.

This shows the inherent complexity of the sim-to-real transfer problem. Despite these limitations, the preliminary results validate the practical feasibility of deploying simulation-trained reinforcement learning models on a real robotic platform and highlight clear areas for improving simulation fidelity to narrow the sim-to-real performance gap.

Consequently, our real-world experiments were designed to serve as preliminary validations of our simulation. These initial real-world experiments showed that more elaborate sim-to-real training should be performed to reach successful sim-to-real transferability. These findings underscore that modest improvements in the simulation's realism and fidelity to actual robot sensor characteristics could considerably narrow the observed sim-to-real performance gap. Comprehensive real-world validations and extensive quantitative experiments are planned as future research to deepen and confirm these preliminary findings.

### 3.9 Discussion

The overall best performer for the camera navigation was DreamerV3. DreamerV3 consistently outperformed other models for the camera observations for both 0x0 and 3x3

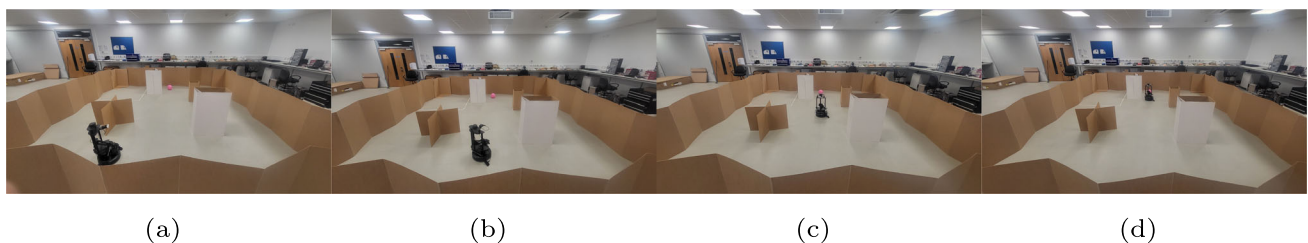


**Fig. 32** Sequence of four images captured during a successful experimental trial, illustrating the robot's autonomous navigation towards the target. The robot effectively identifies, approaches, and reaches the goal using solely visual input from the onboard Intel RealSense D435 RGB camera view



**Fig. 33** Sequence of four images from an unsuccessful experimental trial. Although the robot initially identifies and approaches the target, a collision happens, making the trial unsuccessful

noise area evaluations, when trained on both 0x0 and 3x3 environments (Figs. 17 and 18). This might result from the DreamerV3 having a different structure to the other models and building a world model due to the autoencoder structure. We hypothesized that it might be able to overcome the sensor-denied areas due to the autoencoder/recurrent structure. While it did outperform other models when trained and evaluated on the 3x3 environment (it reached a mean of 80.6 % in Fig. 17), that still does not match the baseline performance of 90.8% when trained and evaluated on the no noise environments. DreamerV3 was also the only model to have learned to navigate to a random goal using the camera-only observation space within 500,000 steps, as shown in Section 3.4. Other models either required the Lidar policy (which contained information about distance and angle to target), or required a static goal to be able to converge to a solution using the camera observation space. DreamerV3 was also convenient to use as it did not require any hyperparameter tuning. The best performer (between TD3 and PPO) for the Lidar modality was TD3. The 0x0 trained TD3 model generally outperformed most variations on PPO across different evaluations.

In Sections 3.2 and 3.6 we have learned that training policies with some noise present in the environment present an interesting result: these policies are more successful at navigating to the goal, but also take more high-risk actions when sensor readings are uncertain or completely denied, which may lead to colliding with obstacles more often - a terminating event in our scenario. This is a limitation of training RL policies for safety-critical scenarios. Training them to maximize the reward and reach the goal in uncertain scenarios may lead to the policies learning to select actions that are deemed rewarding and may lead to achieving a reward, even if these actions may lead to a catastrophic scenario.

Mirowski et al. [8] perform a similar study which includes static mazes and large random mazes. These are equivalent to the static goal (that we performed using the camera) and random goal (that we performed using Lidar). In their paper, algorithms that used recurrence outperformed non-recurrent algorithms. We did not see such a trend in our results, and PPO-LSTM generally performed poorly

(ignoring DreamerV3, which is also recurrent). This is surprising given that navigation is a partially observable Markov decision process (MDP), and hence, policies should benefit from memory.

Ultimately, the aim of the study was to understand the performance of standard DRL models under adversarial conditions during the navigation task. Our results show that indeed, adversarial perturbations applied via area-of-denial severely degrade the performance of the policy. Whilst this can be mitigated to some extent via adversarial training, it does not solve the problem (i.e., the policies never achieve the same performance as in vanilla scenarios).

Henceforth, we conclude that any model that may be able to solve the navigation task in the presence of an adversary must be able to: 1) detect the adversarial state and 2) deploy a mitigation strategy. Coincidentally, there already exists work on detection of adversarial states [38], e.g., via autoencoders [39], or by detecting high-sensitivity directions of the learned manifold [51]. A mitigation strategy might include using an alternate set of sensors (e.g., Lidar in a camera denied zone), or an adversarially trained policy. This is simple, given an accurate detection mechanism, and could be achieved using a state machine. Interesting alternatives could include hierarchical reinforcement learning [34, 61–63], or liquid neural networks [64–66] given their robustness to noise.

## 4 Conclusion and Further Work

This study has shown quantitative results of how common reinforcement learning algorithms perform with different modalities - Lidar + position, and pure vision - on the *DRL-Robot-Navigation* environment as well as the different effects of training and evaluating on environments containing noisy and sensor-denied areas. We observed that training DRL models on noisy environments creates policies that generally perform better on noisy environments, compared with models trained on vanilla, no-noise environments. However, we found that these policies opted for riskier actions when presented with a noisy or faulty observation: ones that might result in successful navigation to the goal, but also in collision

with another object. While these models result in better performance by quantitative metrics, it is insufficient for safety-critical scenarios and shows a requirement for more robust RL methods that are resilient to noise or sensor outages. We open-source the environments that we built on top of the *DRL-Robot-Navigation* to enable other researchers to evaluate their model performance on these scenarios. The results presented here should serve as a benchmark for any future studies on strategies of how to deal with sensor perturbations.

The study is limited in that only vanilla DRL methods were evaluated. Although real-world experiments were shown, generalizability questions (e.g., can the policies navigate in larger mazes [18]) remain. The sim-to-real transfer was performed by taking the policies trained in simulation and putting them on the real robot. To improve the transfer, domain randomization [67, 68] could be utilized. Although hyperparameter studies of DRL models were performed to find optimal values, they were limited by the computational resources and other unstudied parameters might also affect the training. While we present evaluation for Gaussian noise to Lidar and camera outages, different perturbations also should be integrated (e.g. [33] investigates transformations such as compression artefacts, brightness and contrast, and blur for camera observations).

The way that sensor perturbations are handled by the DRL should be a careful point of consideration for future research. We suggest that adversarial training might lead to worse policies as they may be influenced to perform unsafe actions and value the perturbed states highly as they sometimes may lead to a reward. While some work on safety-RL addresses this by

introducing a cost function, the solution should come from an architectural perspective in how the network is designed to deal with sensor perturbation. Therefore, the network should inherently aim to detect out-of-distribution samples or high-sensitivity directions (e.g., [39, 51]) and present mitigation strategies in such cases (e.g., hierarchical RL switching of policies [34] or liquid neural networks [64–66] given their robustness to noise). Hence, a successful solution should combine multi-modality, anomaly detection, and mitigation. Given this, we hope that this study becomes a starting point for the development of more elaborate detection and mitigation DRL architectures capable of detecting and mitigating adversarial sensor perturbations.

## Supplementary Information

The modified *DRL-Robot-Navigation* environment is available <https://github.com/mazqtpopx/cranfield-navigation-gym>.

## Appendix A: Learning Rate Sensitivity

As the algorithms (apart from DreamerV3) are sensitive to learning rate, we initially perform a learning rate study - we then continue using this learning rate throughout.

Figure 34 shows the learning rate discovery for the TD3 algorithm using the Lidar observation space. Both, 0.0003 and 0.00003 values converge to approximately a max of 0.5 episode reward, but 0.0003 appears to converge more quickly.

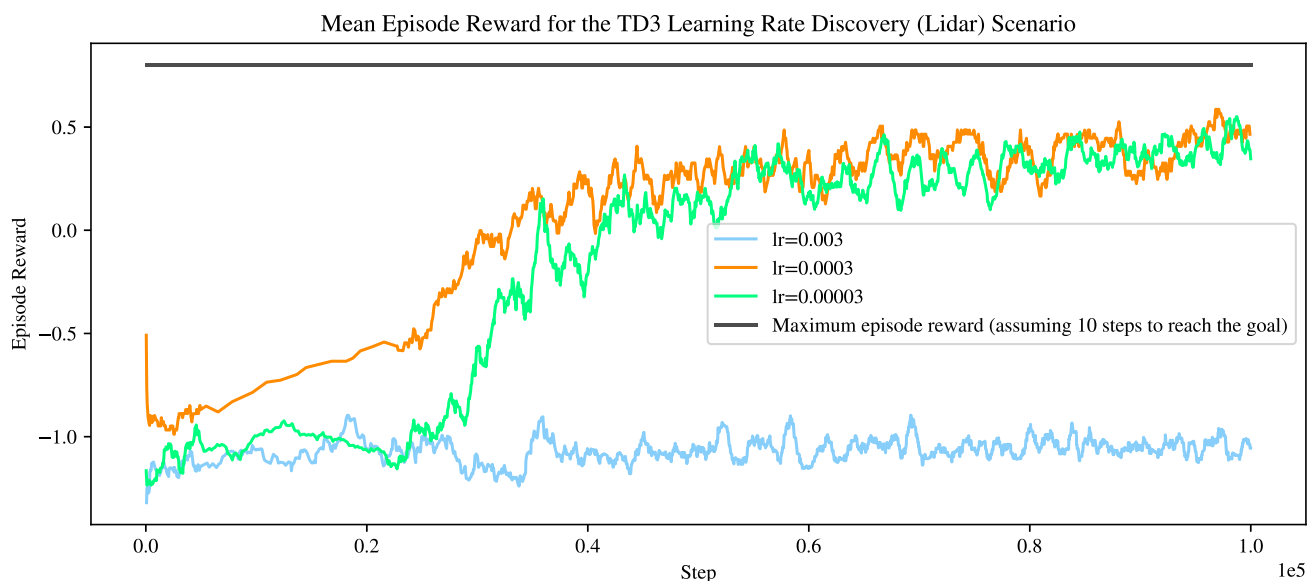


**Fig. 34** TD3 Learning Rate Discovery (Lidar)

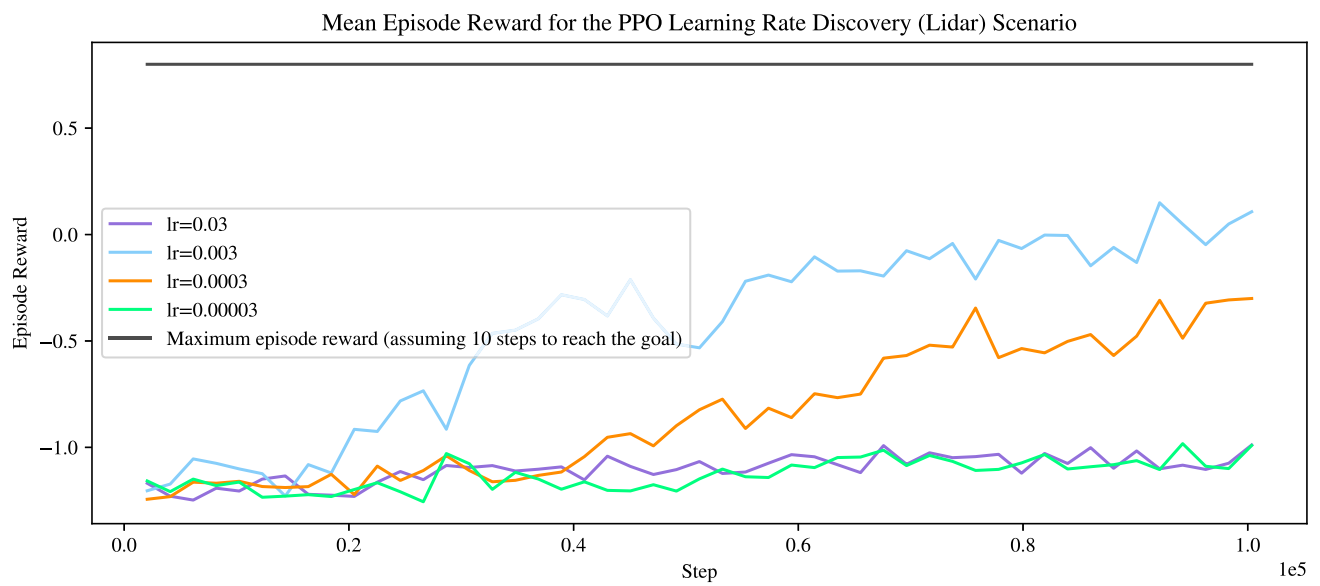


**Fig. 35** PPO Learning Rate Discovery (Lidar)

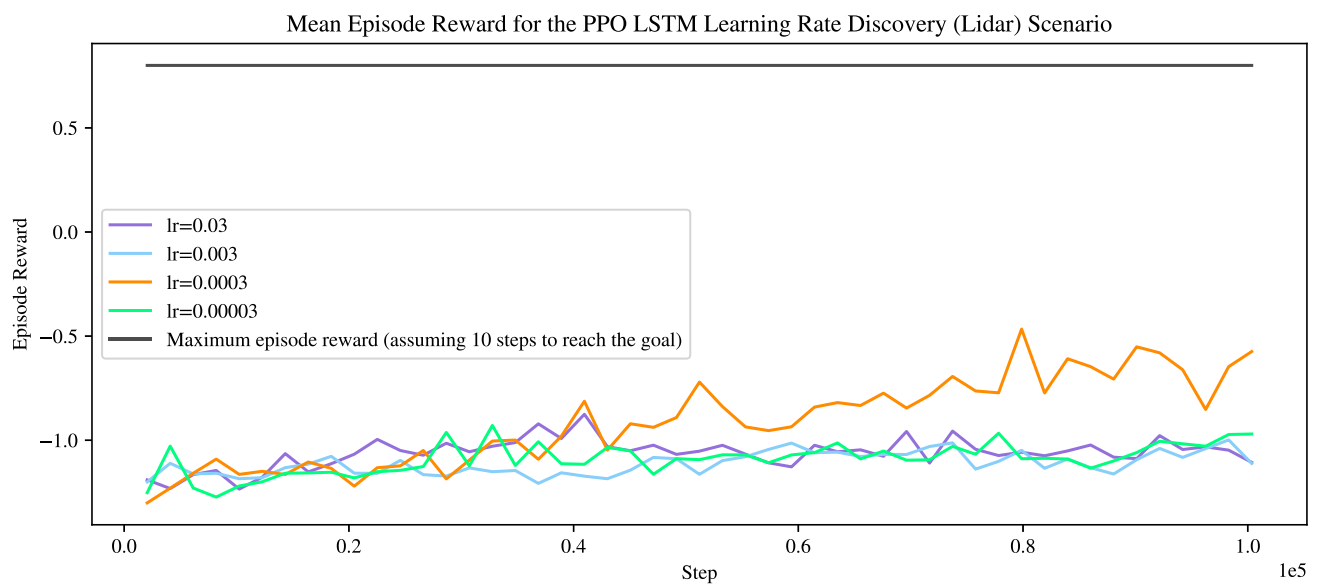


**Fig. 36** PPO LSTM Learning Rate Discovery (Lidar)

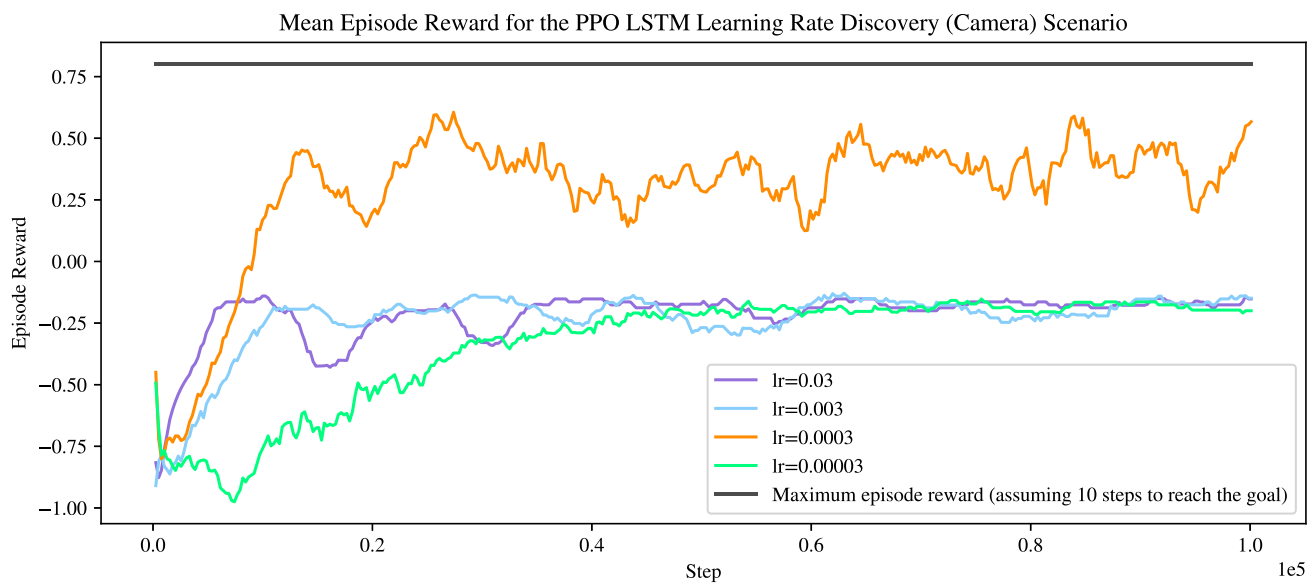


**Fig. 37** PPO LSTM Learning Rate Discovery (Camera, Static Goal)

Figure 35 shows the learning rate discovery for the PPO algorithm using the Lidar observation space. A value of 0.003 appears to be the optimal value converging quicker than 0.0003.

Figure 36 shows the learning rate discovery for the PPO LSTM algorithm using the Lidar observation space. No clear convergence was achieved - this is surprising because we expected recurrence networks to work on this problem. PPO without recurrence performs much better with the Lidar observation space as shown in Fig. 35.

Figure 37 shows the learning rate discovery for the PPO LSTM algorithm for the static goal scenario using the camera observation space. A value of 0.0003 is the only one to converge to a solution. We will use this value of 0.0003 for both PPO and PPO LSTM camera runs.

**Acknowledgements** This work with MW and PC is supported by Leonardo UK with Cranfield University, as well as WG and AT is supported by EPSRC TAS-S: Trustworthy Autonomous Systems: Security (EP/V026763/1).

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. *Commun. ACM* **60**(6), 84–90 (2017). <https://doi.org/10.1145/3065386>. Accessed 2021-07-21
- Deng, J., Dong, W., Socher, R., Li, L.-J., Kai Li, Li Fei-Fei: ImageNet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. IEEE, Miami, FL (2009). <https://doi.org/10.1109/CVPR.2009.5206848>. <https://ieeexplore.ieee.org/document/5206848/>. Accessed 2021-07-21
- He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. *arXiv:1512.03385* [cs] (2015). Accessed 2021-07-10
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. *Nature* **518**(7540), 529–533 (2015). <https://doi.org/10.1038/nature14236>. Accessed 2022-12-10
- Geles, I., Bauersfeld, L., Romero, A., Xing, J., Scaramuzza, D.: Demonstrating Agile Flight from Pixels without State Estimation. In: Robotics: Science and Systems XX. Robotics: Science and Systems Foundation, ??? (2024). <https://doi.org/10.15607/RSS.2024.XX.082>. <http://www.roboticsproceedings.org/rss20/p082.pdf>. Accessed 2025-03-31
- Walker, O., Vanegas, F., Gonzalez, F., Koenig, S.: A deep reinforcement learning framework for uav navigation in indoor environments. In: IEEE Aerospace Conference, pp. 1–14 (2019). <https://doi.org/10.1109/AERO.2019.8742226>
- Zhang, H., Chen, H., Xiao, C., Li, B., Liu, M., Boning, D., Hsieh, C.-J.: Robust Deep Reinforcement Learning against Adversarial Perturbations on State Observations. In: Advances in Neural Information Processing Systems, **33**, pp. 21024–21037. Curran Associates, Inc., ??? (2020). [https://proceedings.neurips.cc/paper\\_files/paper/2020/hash/f0eb6568ea114ba6e293f903c34d7488-Abstract.html](https://proceedings.neurips.cc/paper_files/paper/2020/hash/f0eb6568ea114ba6e293f903c34d7488-Abstract.html). Accessed 2024-10-09

8. Mirowski, P., Pascanu, R., Viola, F., Soyer, H., Ballard, A.J., Banino, A., Denil, M., Goroshin, R., Sifre, L., Kavukcuoglu, K., Kumaran, D., Hadsell, R.: Learning to Navigate in Complex Environments. *arXiv. arXiv:1611.03673* [cs] (2017). Accessed 2022-10-17
9. Cimurs, R., Suh, I.H., Lee, J.H.: Goal-Driven Autonomous Exploration Through Deep Reinforcement Learning. *arXiv. arXiv:2103.07119* [cs] (2021). Accessed 2023-10-19
10. Wisniewski, M., Ona, I.G., Chatzithanos, P., Guo, W., Tsourdos, A.: Autonomous Navigation in Dynamic Maze Environments Under Adversarial Sensor Attack. In: 2024 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 881–886 (2024). <https://doi.org/10.1109/ICUAS60882.2024.10557088>. ISSN: 2575-7296. <https://ieeexplore.ieee.org/document/10557088/?arnumber=10557088> Accessed 2024-08-23
11. Zhu, K., Zhang, T.: Deep reinforcement learning based mobile robot navigation: A review. *Tsinghua Science and Technology* **26**(5), 674–691 (2021) <https://doi.org/10.26599/TST.2021.9010012>. Conference Name: Tsinghua Science and Technology. Accessed 2024-08-27
12. Beattie, C., Leibo, J.Z., Teplyaev, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., Schrittwieser, J., Anderson, K., York, S., Cant, M., Cain, A., Bolton, A., Gaffney, S., King, H., Hassabis, D., Legg, S., Petersen, S.: DeepMind Lab. *arXiv. arXiv:1612.03801* [cs] (2016). Accessed 2023-05-19
13. Kempka, M., Wydmuch, M., Runc, G., Toczek, J., Jaśkowski, W.: ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning. *arXiv:1605.02097* [cs] (2016). Accessed 2022-12-10
14. 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:1602.01783* [cs] (2016). Accessed 2022-10-24
15. Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., Scaramuzza, D.: Champion-level drone racing using deep reinforcement learning. *Nature* **620**(7976), 982–987 (2023). <https://doi.org/10.1038/s41586-023-06419-4>. Number: 7976 Publisher: Nature Publishing Group. Accessed 2023-09-10
16. Polvara, R., Patacchiola, M., Sharma, S., Wan, J., Manning, A., Sutton, R., Cangelosi, A.: Toward end-to-end control for uav autonomous landing via deep reinforcement learning. In: International Conference on Unmanned Aircraft Systems (ICUAS), pp. 115–123 (2018). <https://doi.org/10.1109/ICUAS.2018.8453449>
17. Chen, J., Li, S.E., Tomizuka, M.: Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning. *IEEE Trans. Intell. Transp. Syst.* **23**(6), 5068–5078 (2022). <https://doi.org/10.1109/TITS.2020.3046646>
18. Pasukonis, J., Lillicrap, T., Hafner, D.: Evaluating Long-Term Memory in 3D Mazes. *arXiv:2210.13383* [cs] (2022). Accessed 2024-04-24
19. Lample, G., Chaplot, D.S.: Playing FPS Games with Deep Reinforcement Learning. *arXiv:1609.05521* [cs] (2018). Accessed 2022-10-14
20. Luo, C., Yang, S.X.: A bioinspired neural network for real-time concurrent map building and complete coverage robot navigation in unknown environments. *IEEE Trans. Neural Netw.* **19**(7), 1279–1298 (2008). <https://doi.org/10.1109/TNN.2008.2000394>
21. Chahine, M., Hasani, R., Kao, P., Ray, A., Shubert, R., Lechner, M., Amini, A., Rus, D.: Robust flight navigation out of distribution with liquid neural networks. *Sci. Robot.* **8**(77), 8892 (2023). <https://doi.org/10.1126/scirobotics.adc8892><https://www.science.org/doi/pdf/10.1126/scirobotics.adc8892>
22. Sajjadi, S., Bittick, J., Janabi-Sharifi, F., Mantegh, I.: A robust and adaptive sensor fusion approach for indoor uav localization. In: 2023 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 441–447 (2023). <https://doi.org/10.1109/ICUAS57906.2023.10156526>
23. Negru, S.A., Geragersian, P., Petrunin, I., Guo, W.: Resilient multi-sensor uav navigation with a hybrid federated fusion architecture. *Sensors* **24**(3) (2024). <https://doi.org/10.3390/s24030981>
24. Liu, C., Zhang, G., Guo, W., He, R.: Kalman prediction-based neighbor discovery and its effect on routing protocol in vehicular ad hoc networks. *IEEE Trans. Intell. Transp. Syst.* **21**(1), 159–169 (2020). <https://doi.org/10.1109/TITS.2018.2889923>
25. Bae, S., Han, D., Park, S.: A new approach to lidar and camera fusion for autonomous driving. In: 2023 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC), pp. 751–753 (2023). <https://doi.org/10.1109/ICAIIIC57133.2023.10066963>
26. Kacker, T., Perrusquia, A., Guo, W.: Multi-spectral fusion using generative adversarial networks for uav detection of wild fires. In: 2023 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC), pp. 182–187 (2023). <https://doi.org/10.1109/ICAIIIC57133.2023.10067042>
27. Spaan, M.T.J.: Partially Observable Markov Decision Processes. In: Wiering, M., Otterlo, M. (eds.) *Reinforcement Learning: State-of-the-Art*, pp. 387–414. Springer, Berlin, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-27645-3\\_12](https://doi.org/10.1007/978-3-642-27645-3_12) Accessed 2024-10-12
28. Osogami, T.: Robust partially observable Markov decision process. In: *Proceedings of the 32nd International Conference on Machine Learning*, pp. 106–115. PMLR, ??? (2015). ISSN: 1938-7228. <https://proceedings.mlr.press/v37/osogami15.html> Accessed 2024-10-12
29. Tessler, C., Efroni, Y., Mannor, S.: Action Robust Reinforcement Learning and Applications in Continuous Control. In: *Proceedings of the 36th International Conference on Machine Learning*, pp. 6215–6224. PMLR, ??? (2019). ISSN: 2640-3498. <https://proceedings.mlr.press/v97/tessler19a.html> Accessed 2024-03-04
30. Pinto, L., Davidson, J., Sukthankar, R., Gupta, A.: Robust Adversarial Reinforcement Learning. In: *Proceedings of the 34th International Conference on Machine Learning*, pp. 2817–2826. PMLR, ??? (2017). ISSN: 2640-3498. <https://proceedings.mlr.press/v70/pinto17a.html> Accessed 2025-05-28
31. Korkmaz, E.: Adversarial Training Blocks Generalization in Neural Policies. (2021). <https://openreview.net/forum?id=fXGimmbtD9c> Accessed 2025-03-26
32. Korkmaz, E.: Adversarial Robust Deep Reinforcement Learning Requires Redefining Robustness. *Proceed. AAAI Conf. Artif. Intell.* **37**(7), 8369–8377 (2023). <https://doi.org/10.1609/aaai.v37i7.26009> Accessed 2024-09-28
33. Korkmaz, E.: Understanding and Diagnosing Deep Reinforcement Learning. In: *Proceedings of the 41st International Conference on Machine Learning*, pp. 25265–25275. PMLR, ??? (2024). ISSN: 2640-3498. <https://proceedings.mlr.press/v235/korkmaz24a.html> Accessed 2025-06-23
34. Havens, A.J., Jiang, Z., Sarkar, S.: Online Robust Policy Learning in the Presence of Unknown Adversaries. *arXiv:1807.06064* [cs, stat] (2018). Accessed 2024-10-09
35. Panda, D.K., Guo, W.: Action Robust Reinforcement Learning for Air Mobility Deconfliction Against Conflict Induced Spoofing. *IEEE Trans. Intell. Transp. Syst.* **25**(12), 21343–21355 (2024). <https://doi.org/10.1109/TITS.2024.3454354>. Accessed 2025-06-30
36. Peng, Z., Song, X., Song, S., Stojanovic, V.: Spatiotemporal fault estimation for switched nonlinear reaction–diffusion systems via adaptive iterative learning. *Int. J. Adapt. Contr. Signal Process.* **38**(10), 3473–3483 (2024) <https://doi.org/10.1002/acs.3885>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/acs.3885>. Accessed 2025-03-26

37. Song, X., Wu, C., Song, S., Stojanovic, V., Tejado, I.: Fuzzy wavelet neural adaptive finite-time self-triggered fault-tolerant control for a quadrotor unmanned aerial vehicle with scheduled performance. *Eng. Appl. Artif. Intell.* **131**, 107832 (2024) <https://doi.org/10.1016/j.engappai.2023.107832>. Accessed 2025-03-26
38. Hickling, T., Aouf, N., Spencer, P.: Robust Adversarial Attacks Detection Based on Explainable Deep Reinforcement Learning for UAV Guidance and Planning. *IEEE Trans. Intell. Veh.* **8**(10), 4381–4394 (2023). <https://doi.org/10.1109/TIV.2023.3296227>. Accessed 2024-08-27
39. Hickling, T., Aouf, N.: Real-Time Detection of White and Black Box Adversarial Attacks on Deep Reinforcement Learning UAV Guidance. In: 2024 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 420–425 (2024). <https://doi.org/10.1109/ROBIO64047.2024.10907751>. ISSN: 2994-3574. <https://ieeexplore.ieee.org/document/10907751/> Accessed 2025-06-30
40. Clark, G., Doran, M., Glisson, W.: A Malicious Attack on the Machine Learning Policy of a Robotic System. In: 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), pp. 516–521 (2018). <https://doi.org/10.1109/TrustCom/BigDataSE.2018.00079>. ISSN: 2324-9013. <https://ieeexplore.ieee.org/document/8455947> Accessed 2025-06-30
41. Dasgupta, S., Ghosh, T., Rahman, M.: A Reinforcement Learning Approach for GNSS Spoofing Attack Detection of Autonomous Vehicles. *Transport. Res. Record: J. Transport. Res. Board* **2676**(12), 318–330 (2022) <https://doi.org/10.1177/03611981221095509>. [arXiv:2108.08628](https://arxiv.org/abs/2108.08628) [eess]. Accessed 2025-06-30
42. Wang, Z., Aouf, N.: Efficient adversarial attacks detection for deep reinforcement learning-based autonomous planetary landing GNC. *Acta Astronaut.* **224**, 37–47 (2024) <https://doi.org/10.1016/j.actaastro.2024.07.052>. Accessed 2025-06-30
43. Hosseini, A., Houti, S., Qadir, J.: Deep Reinforcement Learning for Autonomous Navigation on Duckietown Platform: Evaluation of Adversarial Robustness. In: 2023 International Symposium on Networks, Computers and Communications (ISNCC), pp. 1–6 (2023). <https://doi.org/10.1109/ISNCC58260.2023.10323905>. ISSN: 2768-0940. <https://ieeexplore.ieee.org/document/10323905> Accessed 2025-06-30
44. Zhang, Z., Duan, T., Lin, Z., Huang, D., Fang, Z., Sun, Z., Xiong, L., Liang, H., Cui, H., Cui, Y., Gao, Y.: Robust Deep Reinforcement Learning in Robotics via Adaptive Gradient-Masked Adversarial Attacks. [arXiv:2503.20844](https://arxiv.org/abs/2503.20844) [cs] (2025). <https://doi.org/10.48550/arXiv.2503.20844>. Accessed 2025-06-30
45. Lütjens, B., Everett, M., How, J.P.: Certified Adversarial Robustness for Deep Reinforcement Learning. [arXiv:1910.12908](https://arxiv.org/abs/1910.12908) [cs] (2020). 10.48550/arXiv.1910.12908. Accessed 2025-06-30
46. Ibrahim, A.D.M., Hussain, M., Hong, J.-E.: Deep learning adversarial attacks and defenses in autonomous vehicles: a systematic literature review from a safety perspective. *Artif. Intell. Rev.* **58**(1), 28 (2024). <https://doi.org/10.1007/s10462-024-11014-8>. Accessed 2025-06-30
47. Deng, Y., Zhang, T., Lou, G., Zheng, X., Jin, J., Han, Q.-L.: Deep Learning-Based Autonomous Driving Systems: A Survey of Attacks and Defenses. *IEEE Trans. Industr. Inf.* **17**(12), 7897–7912 (2021). <https://doi.org/10.1109/TII.2021.3071405>. Accessed 2025-06-30
48. Ilahi, I., Usama, M., Qadir, J., Janjua, M.U., Al-Fuqaha, A., Hoang, D.T., Niyato, D.: Challenges and Countermeasures for Adversarial Attacks on Deep Reinforcement Learning. [arXiv:2001.09684](https://arxiv.org/abs/2001.09684) [cs] (2021). <https://doi.org/10.48550/arXiv.2001.09684>. Accessed 2025-06-30
49. Standen, M., Kim, J., Szabo, C.: Adversarial Machine Learning Attacks and Defences in Multi-Agent Reinforcement Learning. *ACM Comput. Surv.* **57**(5), 1–35 (2025). <https://doi.org/10.1145/3708320>. Accessed 2025-05-28
50. Korkmaz, E.: Deep Reinforcement Learning Policies Learn Shared Adversarial Features Across MDPs. [arXiv:2112.09025](https://arxiv.org/abs/2112.09025) [cs, stat] (2021). Accessed 2024-09-28
51. Korkmaz, E., Brown-Cohen, J.: Detecting Adversarial Directions in Deep Reinforcement Learning to Make Robust Decisions. [arXiv:2306.05873](https://arxiv.org/abs/2306.05873) [cs, stat] (2023). Accessed 2024-09-29
52. Towers, M., Kwiatkowski, A., Terry, J., Balis, J.U., De Cola, G., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J.J., Tan, H., Younis, O.G.: Gymnasium: A Standard Interface for Reinforcement Learning Environments. [arXiv:2407.17032](https://arxiv.org/abs/2407.17032) [cs] (2024). Accessed 2024-08-27
53. Kim, S.-G., Lee, E., Hong, I.-P., Yook, J.-G.: Review of Intentional Electromagnetic Interference on UAV Sensor Modules and Experimental Study. *Sensors* **22**(6), 2384 (2022). <https://doi.org/10.3390/s22062384>. Number: 6 Publisher: Multidisciplinary Digital Publishing Institute. Accessed 2023-07-27
54. Fujimoto, S., Hoof, H., Meger, D.: Addressing Function Approximation Error in Actor-Critic Methods. [arXiv:1802.09477](https://arxiv.org/abs/1802.09477) [cs, stat] (2018). Accessed 2024-02-18
55. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal Policy Optimization Algorithms. [arXiv:1707.06347](https://arxiv.org/abs/1707.06347) [cs] (2017). Accessed 2022-11-03
56. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. *Neural Comput.* **9**(8), 1735–1780 (1997). <https://doi.org/10.1162/neco.1997.9.8.1735>. Accessed 2024-03-28
57. Hafner, D., Pasukonis, J., Ba, J., Lillicrap, T.: Mastering Diverse Domains through World Models. [arXiv:2301.04104](https://arxiv.org/abs/2301.04104) [cs, stat] (2023). Accessed 2023-12-06
58. Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N.: Stable-Baselines3: Reliable Reinforcement Learning Implementations. *J. Mach. Learn. Res.* **22**(268), 1–8 (2021). Accessed 2024-10-09
59. Wisniewski, M., Rana, Z.A., Petrulin, I., Holt, A., Harman, S.: Towards Fully Autonomous Drone Tracking by a Reinforcement Learning Agent Controlling a Pan-Tilt-Zoom Camera. *Drones* **8**(6), 235 (2024). <https://doi.org/10.3390/drones8060235>. Number: 6 Publisher: Multidisciplinary Digital Publishing Institute. Accessed 2024-08-23
60. Zahavy, T., Zrihem, N.B., Mannor, S.: Graying the black box: Understanding DQNs. [arXiv:1602.02658](https://arxiv.org/abs/1602.02658) [cs] (2017). <https://doi.org/10.48550/arXiv.1602.02658>. Accessed 2025-02-13
61. DeMay, C.R., White, E.L., Dunham, W.D., Pino, J.A.: AlphaDogfight Trials: Bringing Autonomy to Air Combat. *Johns Hopkins APL Technical Digest* **36**(2) (2022)
62. Hafner, D., Lee, K.-H., Fischer, I., Abbeel, P.: Deep Hierarchical Planning from Pixels. [arXiv:2206.04114](https://arxiv.org/abs/2206.04114) [cs, stat] (2022). Accessed 2024-04-24
63. Pope, A.P., Ide, J.S., Mićović, D., Diaz, H., Twedt, J.C., Alcedo, K., Walker, T.T., Rosenbluth, D., Ritholtz, L., Javorsek, D.: Hierarchical Reinforcement Learning for Air Combat at DARPA's AlphaDogfight Trials. *IEEE Trans. Artif. Intell.* **4**(6), 1371–1385 (2023). <https://doi.org/10.1109/TAI.2022.3222143>. Conference Name: IEEE Transactions on Artificial Intelligence. Accessed 2025-04-03
64. Lechner, M., Hasani, R., Amini, A., Henzinger, T.A., Rus, D., Grosu, R.: Neural circuit policies enabling auditable autonomy. *Nature Mach. Intell.* **2**(10), 642–652 (2020). <https://doi.org/10.1038/s42256-020-00237-3>. Number: 10 Publisher: Nature Publishing Group. Accessed 2024-01-31
65. Hasani, R., Lechner, M., Amini, A., Rus, D., Grosu, R.: Liquid Time-constant Networks. [arXiv:2006.04439](https://arxiv.org/abs/2006.04439) [cs, stat] (2020). Accessed 2024-01-17

66. Chahine, M., Hasani, R., Kao, P., Ray, A., Shubert, R., Lechner, M., Amini, A., Rus, D.: Robust flight navigation out of distribution with liquid neural networks. *Sci. Robot.* **8**(77), 8892 (2023). <https://doi.org/10.1126/scirobotics.adc8892>. Publisher: American Association for the Advancement of Science. Accessed 2025-03-26
67. Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Bochoon, S., Birchfield, S.: Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization. *arXiv:1804.06516* [cs] (2018). Accessed 2021-07-13
68. Wisniewski, M., Rana, Z.A., Petrunin, I., Holt, A., Harman, S.: Drone Detection using Deep Neural Networks Trained on Pure Synthetic Data. *arXiv. arXiv:2411.09077* [cs] (2024). <https://doi.org/10.48550/arXiv.2411.09077>. Accessed 2025-07-02

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Mariusz Wisniewski** is a Research Fellow in Trust and Sensing for Autonomy at Cranfield University. He holds an MEng in Aeronautical Engineering from Loughborough University and a PhD from Cranfield University on using synthetic data for drone detection using machine learning. He has professional experience working on wind tunnels in Red Bull Racing, and simulation of avionics for flight simulators in L3Harris. His interests include deep reinforcement learning, adversarial attacks, synthetic data, and UAV autonomy.

**Paraskevas Chatzithanos** is a Doctoral Researcher and Research Assistant at Cranfield University, working on cognitive AI partners in human–robot teams and explainable interfaces. He holds a Master's degree in Robotics and Autonomous Systems from University of Lincoln. His research focuses on human–robot interaction, trust in robotic systems, and the design of transparent, explainable AI tools to enhance teamwork between humans and autonomous agents. His interests include human-centred AI and developing intelligent and trustworthy robotic systems.

**Weisi Guo** Professor Guo is the Head of Centre for Assured & Connected Autonomy and Head of Human Machine Intelligence Group at Cranfield University. He obtained his MEng, MA, and PhD degrees from the University of Cambridge. Since 2012, he has built an award-winning team at the University of Warwick (2012–19) and then Cranfield (2019–now). His research group has won the IET Innovation Award winner (2015) and been a runner-up in the Bell Labs Prize three times (2014, 2017, 2019). He has been PI on £8.2m and investigator on over £35m of research funding, and was a Turing Fellow at the Alan Turing Institute. He has published 160+ journal papers (total IF 860+) and 100+ IEEE/ACM conference papers, with over 7000 citations (h-index 42). This includes a Nature, Nature Communications, Nature Machine Intelligence, Nature Comp.Sci., and several cover issues in Royal Society, Nature Publishing Group, and IEEE journals. He currently serves as editor on several IEEE & Royal Society journals, and is a Full Member of the EPSRC peer-review college, as well as reviewing for UKRI FLF, ESRC, MRC, Royal Society, and Leverhulme. His research expertise is in connected intelligence, linking: data science, communication systems, and machine learning and his research interests is in how AI and information systems can be designed for specific challenging environments or tasks, whilst integrating human domain knowledge.

**Antonios Tsourdos** is a Professor of Autonomous Systems and Control with Cranfield University. He was appointed Director of Research - Aerospace, Transport and Manufacturing in 2015 and Director of AI, Robotics and Space Theme, Faculty of Engineering and Applied Science in 2024. He was member of the BSI Future Flight Standards Hub (2023–2024) and the AIAA Advancing Aerial Mobility Certification Task Force (2023–2024). Professor Tsourdos is a Fellow of the Royal Aeronautical Society, he has served (2017–2023) as chair of the IFAC Technical Committee on Aerospace Control. Professor Tsourdos research is focussed on autonomy. He has published widely on guidance, control and navigation for single and multiple autonomous vehicles as well as mission planning and multi-agent systems.



**DSPACE**

<https://dspace.org/>

## **Benchmarking deep reinforcement learning for navigation in denied sensor environments**

**Wisniewski, Mariusz; Chatzithanos, Paraskevas; Guo, Weisi; Tsourdos, Antonios**

**2025-09-15**

Springer

<https://doi.org/10.1007/s10846-025-02309-1>

Wisniewski M, Chatzithanos P, Guo W, Tsourdos A. (2025) Benchmarking deep reinforcement learning for navigation in denied sensor environments. Journal of Intelligent & Robotic Systems, Volume 111, Issue 3, September 2025, Article number 103