In: AIAA SciTech Forum 2023, 23-27 January 2023, National Harbor, Maryland, USA
DOI: 10.2514/6.2023-0967

Optimization of a Robust Reinforcement Learning Policy

Bilkan Ince[1] and Hyo-Sang Shin[2]
School of Aerospace, Transport and Manufacturing, Cranfield University, MK430AL, U.K.

Antonios Tsourdos[3],
School of Aerospace, Transport and Manufacturing, Cranfield University, MK430AL, U.K.

A major challenge for the integration of unmanned air vehicle (UAV) in the current civil applications is the sense-and-avoid (SAA) capability and the consequent possibility of mid-air collision avoidance. Although UAS have been shown to be efficient under different and varied conditions, their safety, reliability, and compliance with aviation regulations remain to be proven. In autonomous collision avoidance, UAS sense hazards with the sensors equipped on them and make decisions on manoeuvres autonomously for collision avoidance at the minimum safe time before impact. Thus, it is required for each individual UAS to have capabilities to recognize urgent threats and undertake the evasive manoeuvres immediately. Most of the current sense and avoid algorithms are composed of separated obstacle detection and tracking algorithm and decision-making algorithm on avoidance manoeuvre. Implementing artificial intelligence (AI), reinforcement learning (RL) algorithm combines both sense and avoid functions through state and action space. An autonomous agent learns to perform complex tasks by maximizing reward signals while interacting with its environment. It may be infeasible to test a policy in all contexts since it is difficult to ensure it works as broadly as intended. In these cases, it is important to trade-off between performance and robustness while learning a policy. This work develops an optimization method for a robust reinforcement learning policy for a nonlinear small unmanned air systems (sUAS), in AirSim using a model-free architecture. Using an on-line trained reinforcement learning agent, the difference of an optimized robust reinforcement learning (RRL) policy together with a conventional RL and RRL algorithm will be reproduced.

I. INTRODUCTION

In the current climate, unmanned air vehicles (UAV) are in growing demand due to their autonomous capabilities to accomplish various operations such as medical deliveries, commercial package delivery, critical infrastructure inspection, and search and rescue operations. It is required for each individual unmanned aerial system (UAS) to have capabilities to detect urgent threats while deciding on collision avoidance manoeuvres. To enhance the sense and avoid performance, it is required to develop an algorithm which makes decisions immediately and directly from sensor measurements. Reliable integrated sense and avoid algorithm for UAS is in growing importance for many autonomous applications. In particular, safety within detect and avoid (DAA) presents challenges, due to the variety of airborne obstacles and the need to find non-cooperative detection technologies that are accurate and have a desirable response. Combining image processing and avoidance, reinforcement learning provides a novel method for DAA capabilities for UAS, integrating both detect and avoid scheme and eliminating the lag between the implementation of many algorithms and integrating these into one individual algorithm.

Reinforcement learning (RL) aims to improve the action of an agent based on the reward received from an environment [1]. The trained RL agent performs an action to obtain reward from the environment and adjusts its policy based on the reward. By continuously communicating with the synthetic environment, the agent can self-

[1]Doctoral Researcher , School of Aerospace, Transport and Manufacturing, Cranfield University, b.g.ince@cranfield.ac.uk
[2]Professor , School of Aerospace, Transport and Manufacturing,Cranfield University, h.shin@cranfield.ac.uk
[3]Professor, School of Aerospace, Transport and Manufacturing, Cranfield University, a.tsourdos@cranfield.ac.uk

Published by AIAA. This is the Author Accepted Manuscript issued with: Creative Commons Attribution License (CC:BY 4.0).
The final published version (version of record) is available online at DOI:10.2514/6.2023-0967. Please refer to any applicable publisher terms of use.

learn the optimal policy to obtain the maximum cumulative reward [2]. RL is based on solving the current problem using the Markov Decision Process (MDP), and through obtaining an optimal policy for this problem [3]. MDP determines the probability that the agent will move from one state to another, this can be defined through utilising the concept of state-transition probabilities. In practice, the estimation of state-transition probabilities could introduce many inaccuracies. These transition errors may limit the application of MDP in new environments and lead to model deteriorations. Therefore, the use of finite-state and finite-action can be an important factor in decision making aspect of the uncertainties within the state transition probability matrix[4].

An example of the application challenge for a reinforcement learning agent includes using a drone in various tasks. Using drones for payload operations in a windy environment, at the same time flying the drone in indoor environment. The new environmental conditions and the possible deterioration of the drone components due to their usage may result in a poor, if not catastrophic, performance of the learned controller [5]. Furthermore, incorporating a reinforcement learning agent to experimental tests presents further challenges due to their sensitivity to distributional model parameters, typically unknown, which will require estimation. All these factors mentioned above will produce small adversarial perturbations, external disturbances, and unpredictable sensing noise which will cause vulnerabilities and uncertainties in action-state values. Considering robustness alongside performance, agents can limit performance degradation due to different training and testing environments.

Contributions. This paper addresses the issue of uncertainty within MDPs': A Markov decision problem has been considered in which the transition probabilities themselves are uncertain and seek a robust decision for it. While many works investigating the performance/robustness trade-off exist in both the RL and control theory literature for the model-based setting, few results are known for the model-free scenario. There are several real-world scenarios where models are not available, inaccurate, or too expensive to use, however robustness is fundamental. Thus, in this paper, the first data-efficient, robust, model-free RL method-based optimization of a robust learning policy is introduced. In particular, these are our individual contributions:

- The formulation of the robust, model-free RL as an optimization problem.
- Consideration of the Markov decision problem in which the transition probabilities themselves are uncertain and seek a robust decision for it.
- Proposal of the uncertainty sets on existing RL frameworks for learning robust policies.
- Efficient problem solving with expected hyper-volume improvement (EHI).
- Introduction of a disturbance model for uncertain parameters within MDPs'.
- Demonstrate that the proposed non-robust policy outperforms the performance of optimization for robustness.

Related Work. In control theory, robustness has been a widely investigated topic [6], and standard robust control techniques for linear systems include loop transfer recovery, H_∞ control. For nonlinear systems, robust control techniques exist with some utilizing feedback linearization and backstepping control for neural networks [7]. However, these control strategies when implemented in big nonlinear systems have not achieved better performances due to the high nonlinear behaviour, the presence of parametric uncertainties and unmodeled dynamics, and many assumptions that must be considered for obtaining simplified mathematical models for control purpose design [8]. In [9], robust backstepping control for nonlinear systems using neural networks is implemented with the need of parameter assumption by estimating certain nonlinear function for each stage of the neural net.

The rest of the paper is as follows: Section 2, defines the nominal problem for reinforcement learning (RL) using MDPs' for both a conventional RL and a robust RL, and a definition of the neural network structure. Furthermore, the robust control problem is devised by defining uncertainties and modelling these uncertainties through a disturbance model. Section 3 expands the robust RL and a proposed optimization method on robustness is considered, particularly implementing a protagonist and an adversarial agent. A pseudo algorithm is represented to depict the proposed optimization methodology. Section 4 discusses the RL and robust RL simulation training process and results for robust policy optimization. Finally in section 5, conclusions are presented regarding the optimization and robust control results.

II. REINFORCEMENT LEARNING ARCHITECTURE SETUP

A. Nominal Problem

The mathematical model of RL problems can be expressed using Markov equations, proposed by Bellman [10]. The decision-making process is constructed combining Markov reward process, Bellman equation and Markov properties. There are 5-tuples in which derives from the relationship of the agent and the environment for an MDP [1] S, the state that an agent can attain in a specific environment; A, the action that an agent must execute to move from one state to another; $P(s_{t+1}|s_t, a_t)$, the probability of executing an action a to move an agent from states s to states s'; $R(s, a, s')$ the received reward by an agent after executing action and transition from states s; γ the discount factor that determines the importance of current and future rewards.

Q-Learning. After constructing the problem as an MDP model, the next step is to solve for the optimal policy. This maximizes the value of all states at the same time. More common techniques for deriving the optimal policy are by using on and off policies. Q-learning [11] and Sarsa [12] are the most known RL. Discrete time steps are defined as follows: t =1,2,3…, where at each time step t, the agent interacts with the environment to obtain the environment state S_t. Given a reward function r_t and the current state S_t, the agent selects an action a_t till the next state of S_{t+1}. This process is looped continuously until the agent reaches the terminal, supposing the episode is from time t to T; giving the cumulative reward below:

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_t \tag{1}$$

where γ is a discount factor, $0 \leq \gamma \leq 1$, this allows the adjustment of the weight of the future rewards to be relayed on to the cumulative reward. Through maximizing the expected cumulative reward, the control policy, $\pi(a_t \mid s_t)$, can be formulated as given in Eq. (2-3), where $\tau = (s_0, a_0, s_1, a_1, \dots)$ is the trajectory sampled using policy π.

$$\pi^* = arg \max_{\pi} E_{\tau\sim\pi}[R(\tau)] \tag{2}$$

$$\pi^* = arg \max_{\pi} E_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^{t'-t} r_t \,\middle|\, \pi\right] \tag{3}$$

The action value function represents the cumulative reward obtainable by the agent if the agent executes action a at state s and always follows the policy π to the end of the episode given above and it is defined as follows:

$$Q(s, a) = E[R_t|s_t = s, a_t = a, \pi] \tag{4}$$

$$Q(s, a) = E\left[\sum_{t'=t}^{T} \gamma^{t'-t} r_t \,\middle|\, s_t = s, a_t = a, \pi\right] \tag{5}$$

In a model-free setting, the use of Q-Value enables updated state estimation from the action/value function represented as, $Q(s, a) \leftarrow Q^{\pi}(s, a) + \alpha[R_{t+1} + \gamma \max_{a'} Q(s', a') - Q(s, a)]$. Although this architecture can eventually achieve reasonable results, it tends to overestimate the action values and takes a long time to train. Expanding the scope of Q-learning [2], Deep Q-Network (DQN) is shown to be directly trainable from raw images [13], it provides a valid option for a detect and avoid (DAA) algorithm. The DQN has two networks with the same hyperparameters. The evaluation network uses $Q(s, a; \theta)$ as the Q function to approximate the action value function while the target network uses $Q(s, a; \theta^-)$, where θ and θ^- define the parametric states for the evaluation network and target network, respectively. DQN solves the problem of overestimation of action values, the updated Q-value can be represented as follows:

$$Q(s, a) \leftarrow Q^{\pi}(s, a) + \alpha[R_{t+1} + \gamma Q(s', \alpha') - Q(s, a)] \tag{6}$$

$$a = \max_{a'} Q(s', \alpha') \tag{7}$$

$$q_{estimated} = Q'(s', \alpha) \tag{8}$$

Q function is selecting the best action a with a maximum Q–value of the next state. Q' function is for calculating expected Q-value by using the action a. Learning rate, α, is neglected when updating the Q-values as this will be utilized in the optimization stage of parameter updates. In traditional DQN only one stream of fully connected layers is constructed after the convolution layers to estimate the Q-value of each action-state pair, given the current state. However, in the duelling network, two streams of fully connected layers are built to compute the value and advantage functions separately, [14] proposes a double DQN (DDQN) which uses two different deep neural networks, Deep Q-Network (DQN) and target network, Q_{tnet}.

$$Q_{qnet}(s,a) \leftarrow R_{t+1} + \gamma Q(s',\alpha') \tag{9}$$

$$a = \max_{a'} Q_{qnet}(s',\alpha') \tag{10}$$

$$q_{estimated} = Q_{tnet}{}'(s',\alpha) \tag{11}$$

The work of [15], combines the ideas of duelling network and double deep Q-network for two streams of fully connected layers which are finally combined for computing Q-values. Specifically, it has three convolutional layers, specified with filter size (height, width, channel), and three fully connected layers for two streams of duelling architecture.

The DDQN structure for a depth image classifier can be represented as shown in Fig. 1, consisting of depth image prediction network and Q network. Within the Q network, there are two fully connected layers, two output layers and the action space.

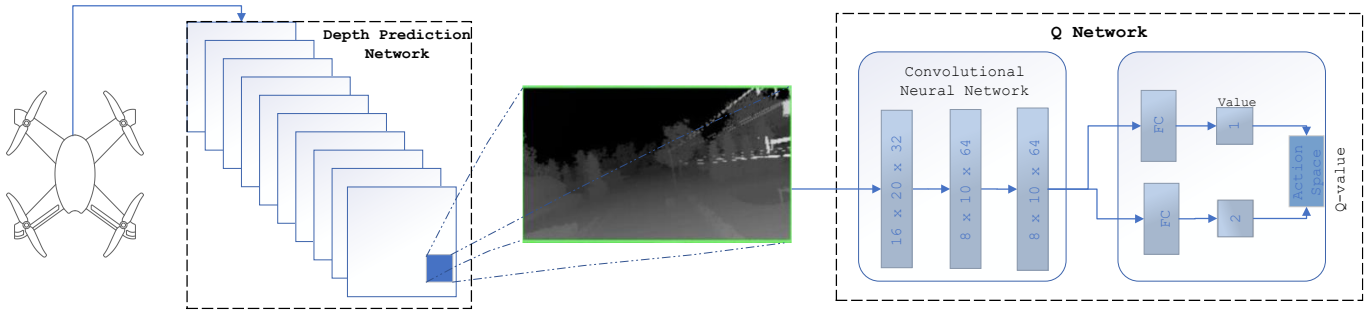


Fig. 1 DDQN Model Architecture with depth prediction and Q- Network structures.

B. Robust Control Problem

Recently, robustness has drawn attention in data-driven settings, giving rise to the field of robust model-based RL. Robust Markov decision processes study the RL problem when the transition model is subject to known and bounded uncertainties [5]. In [16] the use of parametric uncertainties is considered, mostly using model-based methods. Therefore, formulating the uncertainties within a reinforcement learning agent is necessary to achieve a more robust dynamic system. Three types of uncertainties can be modelled to create a robust architecture: Model, Parameter and Inherent.

Parameter uncertainty, $u'_{parameter}$, arises when the model parameter values are specified under imprecise knowledge or lack of direct measurements. States and actions are parameter dependent, and the objective is to determine the optimal parameters along with the corresponding optimal policy. The unknown parameters that the state and action variables depend upon are such that the cumulative cost is minimized.

$$\max_{\pi} \min_{P \in possible\ MDP\ parameters} \mathbb{E}^{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right] \tag{12}$$

Equation (12) finds the policy that maximises the performance within the worst possible model. Through modelling a max-min problem, the defined policy, π, and the most adverbial parameters are set to solve the expectation parameter.

Model uncertainty, u'_{model}, in comparison, captures how well a model fits all possible observations from the environment. This model is typically high in applications with limited training data, or with test data that is far from the training data. Thus, the model uncertainty captures cases in which a model fails to generalize to novel test data and hints when the network predictions are not reliable.

$$\max_{\pi} \min_{possible\ models} \mathbb{E}_{model}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right] \tag{13}$$

Inherent uncertainties, $u'_{inherent}$, consider optimal risk sensitive action is selected by applying established risk criteria, such as the conditional value at risk, to the learned state action return distributions.

$$\max_{\pi} \rho\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right] \tag{14}$$

ρ is a risk measure, Markovic risk measure, which is expectation minus a constant time the variance.

Implementation of inherent uncertainties can be defined by utilising a probabilistic event map and creating a consequential risk criterion. The consideration of flight envelopes and off-nominal conditions are used when there is an emergency threat present. Table. 3 inspired by [17] shows the accumulative understanding of consequential risk assessment:

Table. 1 Probabilistic event likelihood.

Improbable	Unlikely	Sporadic	Frequent
$0 \leq P_E < 0.01$	$0.01 \leq P_E < 0.2$	$0.2 \leq P_E < 0.6$	$0.6 \leq P_E < 1$

Table. 2 Probabilistic likelihood obstacles.

Obstacles	Probability, P_E
Small Foliage (Bushes,)	0.35
Trees	0.6
Buildings	0.7
Intruders (Birds, e.g.,)	0.1
Ground Vehicle	0.15

Table. 3 Risk and severity assessment.

Risk ↓ Severity→	Minimal Index:[1]	Moderate Index:[2]	Major Index:[4]
Frequent ($0.6 \leq P_E < 1$)			
Sporadic ($0.2 \leq P_E < 0.6$)			
Unlikely ($0.01 \leq P_E < 0.2$)			
Improbable ($0 \leq P_E < 0.01$)			

P_E represents the probability of an event occurring, definition of the severity indexes is given as follows:

1. *Minimal*: Low-level damage to the environment due to collision.
2. *Moderate*: Non-serious or mild damage to the sUAS and the surrounding obstacle due to impact.
3. *Major*: Fatal damage to the sUAS and the surrounding obstacle due to impact.

Severity indexes focus on the surrounding object damage and the potential damage to the sUAS. This concept allows the hazard interpretation of the third party and the likelihood of the event. To understand this concept in terms of numerical and qualitative analysis, sensor hazard assessment is utilized. Representation of the uncertainty sets can be represented as given in Fig. 2.

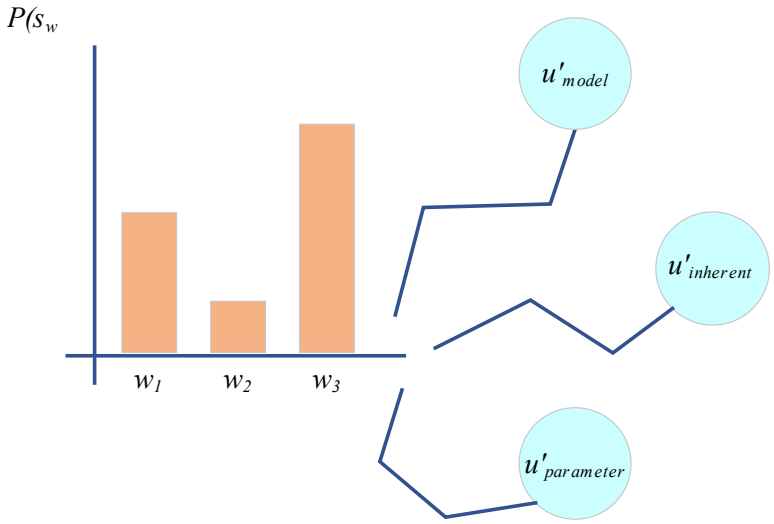


Fig. 2 State disturbance model affecting uncertain parameters.

The system's sensitivity to its maximum disturbances is then minimized. Robust RL has drawn inspiration not only from robust optimization but also from H_∞-control [18], [19]. Formulations based on robust optimization are closely related to game theory. In two-player zero-sum games, a protagonist, i.e., an agent or controller, minimizes an objective function, while an opposing player maximizes the same objective. This competitive framework known as a mini-max game corresponds to the worst-case design [20]. In order to be able to implement the aforementioned uncertainties, a disturbance model is constructed as shown in Fig. 3. Using the disturbance model a disturbance reward is derived, and the observation functions are changed.

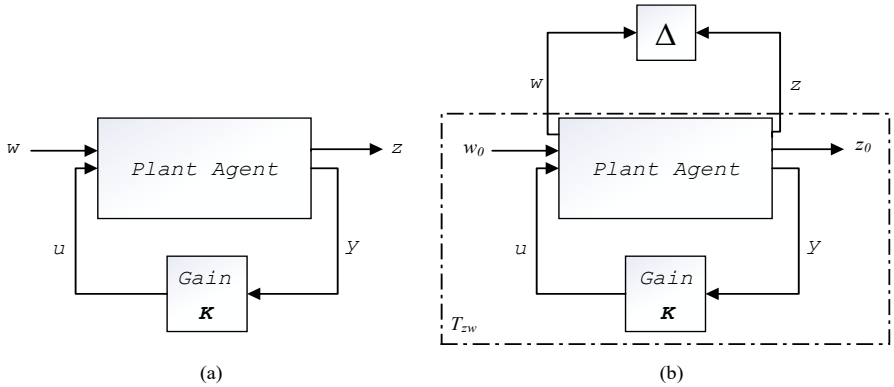


Fig. 3 (a) A plant without model uncertainty, where w is a vector signal containing external noise, disturbances, and the reference signal. The system output is given in z. Measurements are represented by y, while u is the control signal. (b) A plant with all possible model uncertainty expressed as Δ. Here w_0 depicts external noise, disturbances, and the reference signal. Now w is a signal representing parameter perturbations and model uncertainty.

One of the common ways of implementing robustness is through using the concept of H_∞ -control. Through this concept we can describe parameter changes in the environment as disturbances or through approximation errors. Given a disturbance, Δ, we can model the disturbance parameter, w.

In order to model the disturbance, $w(t)$, as a parameter, the uncertainties defined in Fig. 2, will be considered. Including the parameter, inherent and model uncertainties, we can represent the disturbance model as follows:

$$w(t) = r_w\big(A_w\big(x(t)\big); v^w\big) + n^w(t) \tag{15}$$

where $A_w\big(x(t)\big); v^w$ is the function approximator, v^w is the parameter vectors and the n^w is the noise disturbance term for exploration. The noise term can be devised as given,

$$\dot{v_i}^w = -\eta^w \delta(t) \eta_i^w(t) \frac{\partial A_w(x(t); v^w)}{\partial v_i^w} \tag{16}$$

where η^w is the learning rate. Therefore, the disturbance signal can be further defined as a generic reward given as, $r_w = \gamma^2 w^T w$. Augmented value function with the disturbance reward:

$$V[x(t)] = \int_t^\infty \big(z^T(t)z(t) - r_w(t)\big)\, dt \leq 0 \tag{17}$$

$$V[x(t)] = \int_t^\infty (z^T(t)z(t) - \gamma^2 w^T(t)w(t))\, dt \leq 0 \tag{18}$$

III. Optimisation Design for A Robust Agent

Most research focuses on the deployment of RL in competitive games instead of robustness. The framework, however, is also valid for robust RL. The core difference is the formulation of the opposing player, the adversary. Controlling uncertainty and disturbances through the adversary produces robust protagonists [20]. Through utilizing the min-max concept we can introduce adversarial reinforcement learning.

A. Robust Policy Optimisation

Our goal is to learn the policy of the RL agent (protagonist) denoted by, π, such that it has higher reward and better robust (generalizes better to variations in test settings). In the standard reinforcement learning setting, for a given transition function $\mathcal{P}$ we can learn policy parameters θ^π such that the expected reward is maximized.

$$\rho(\pi; \theta^\pi, \mathcal{P}) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) | s_0, \pi, \mathcal{P}\right] \tag{19}$$

In standard-RL settings, the transition function is fixed (since the physics engine and parameters such as mass, friction is fixed). However, in our setting, we assume that the transition function will have modelling errors and that there will be differences between training and test conditions. Therefore, in our general setting, we should estimate policy parameters θ such that we maximize the expected reward over different possible transition functions as well.

$$\rho(\pi; \theta^\pi) = \frac{\mathbb{E}}{\mathcal{P}}\left[\mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) | s_0, \pi, \mathcal{P}\right]\right] \tag{20}$$

Optimisation of the expected reward over all transition functions allows the assumption of known disturbances over model parameters, which in turn optimizes the average performance. [20] implements the idea of optimisation through conditional value at risk (CVaR), represented as follows:

$$\rho_{RC} = \mathbb{E}[\rho |\, \rho \leq Q_a(\rho)] \tag{21}$$

where $Q_a(\rho)$ is the a -value of ρ-values. For robust control we want the worst possible case scenario for ρ-values. We can achieve the worst possible scenario by sampling the change of state/observation parameters.
Instead, an adversarial agent is implemented varying the state functions such that the reward of the original RL agent is minimised.

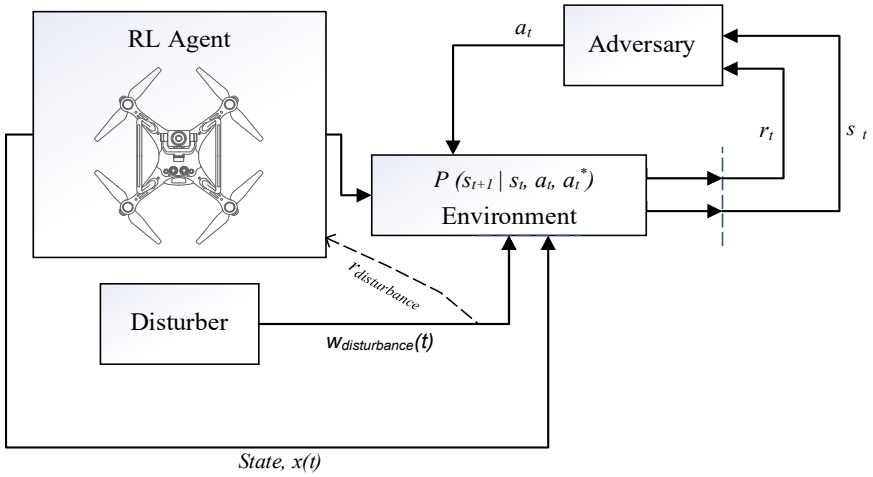


Fig. 4 DDQN Model Architecture with depth prediction and Q- Network structures.

For every step, t, the RL agent and the adversarial agent observe the state s_t, and take the relevant actions, $a_{1_t} \sim \pi(s_t)$, and $a_{2_t} \sim \bar{\pi}(s_t)$. Reward can be formulated as such $r_t = r(s_t, a_{1_t}, a_{2_t})$ for the state transitions and environment. In zero-sum game, the RL agent gets a reward $r_{1_t} = r_t$, while the adversary gets a reward $r_{2_t} = -(r_t + r_w)$. Each step of the MDP can be represented as $(s_t, a_{1_t}, a_{2_t}, r_{1_t}, r_{2_t}, s_{t+1})$.

We introduce a loss function that takes the state and value function into account for optimization of the robust policy, the single loss function will be a superset of many common robust loss functions. [21] A single continuous-valued parameter in our general loss function can be set such that it is equal to several traditional losses and can be adjusted to model a wider family of functions. This allows us to generalize algorithms built around a fixed robust loss with a new robustness hyperparameter that can be tuned or annealed to improve performance.

For the robust policy optimization, π^*, we can further express the function, $f(x, \alpha, c, \pi^*)$. Solving a term for the optimization of the robust policy is expected to improve the efficiency and reliability of the robust reinforcement learning agent [20], [22].

Therefore, we introduce a robust MDP, $(S, A, \mathbb{C}, \gamma, P, T_p, \mathbb{P}_0)$, where T_p and $\mathbb{P}_0$ are state transition function for $p \in P$, and an initial state distribution, respectively. $S, A, \mathbb{C}, \gamma,$ and P defining states, action, cost function, discount factor and an ambiguity set, respectively. In other ways, we can further evaluate the MDP as given: $(s_t, a_{1_t}, a_{2_t}, r_{1_t}, r_{2_t}, P, T_p, \mathbb{P}_0, s_{t+1})$, with the average augmented MDP $(s_t', a_{1_t}', a_{2_t}', r_{1_t}, r_{2_t}, P', T_p', \mathbb{P}_0', s_{t+1}')$ [23]. We can define the cost function in the form $c_{t+1} \sim \mathbb{C}$, where c_{t+1} is a random variable, hence the sum of the costs discounted by γ is called a loss $C = \sum_{t=0}^{T} \gamma^t c_t$.

The loss function to control the robustness of the loss where, $C > 0$, is as given below.

$$f(x, \alpha, C) = \frac{\alpha - k}{\alpha} \left(\left(\frac{\left(\frac{x}{C}\right)^2}{|\alpha - k|} + 1 \right)^{\frac{\alpha}{k}} - 1 \right) \tag{22}$$

As with the standard robust reinforcement learning settings [24], [25], [26], p is generated by a model parameter distribution $\mathbb{P}(p)$ that captures our subjective belief about the parameter values of a real environment.

$$\mathbb{E}_{C,p}[C] = \sum_{p} \mathbb{P}(p)\, \mathbb{E}_C[C,p] \tag{23}$$

$\mathbb{E}_C[C,p]$ is the expected loss on an MDP, $(S, A, \mathbb{C}, \gamma, P, T_p, \mathbb{P}_0)$ in which the parametrized p is the transition probability. In order to optimise, we can minimise the expected loss function using parameter set, θ, representing the soft robust loss.

$$\min_{\theta} \mathbb{E}_{C,p}[C] \tag{24}$$

B. Proposed Method

This work aims to train and test both on simulation and experimentally, robust reinforcement learning for a single agent. Given the Markov decision equation for reinforcement learning,

$$R(\pi, \boldsymbol{P}) = \mathbb{E}^{\pi,\boldsymbol{P}}\left[\sum_{t=0}^{\infty} \lambda^t r_{s_t a_t} \middle| s_0 \sim p_0\right] \tag{25}$$

where s_0 and p_0 the robust policy can be shown as below,

$$\pi^* = arg\max_{\pi} \min_{\bar{\pi}} R(\pi, \boldsymbol{P}) \tag{26}$$

where a transition kernel $\boldsymbol{P}$ gives transition probabilities $P_{sa} \in R_+^{|\mathbb{S}|}$ for all state-action pair *(s, a),* some rewards r_{sa} for each state-action pair *(s, a)* and a discount factor $\lambda \in (0, 1)$. For estimating the unknown distributional model parameters, we derive a confidence region that contains the unknown parameters with a prespecified probability 1-β. By construction, this policy achieves or exceeds its worst-case performance with a confidence of at least 1-β. The algorithm updates the protagonist through the adversarial agent using the following alternating procedure. First phase, we learn the RL agents' policies while the adversarial is constant. The algorithm below, depicts a pseudo code using CVaR constraint, a DDQN structure is utilized with the option policies.

Input: $N_{iter}, N_{epi}, \theta_{\pi}, \theta_{\pi_w}, w_0$

Initialize action-value function Q with parameters θ and policy π

Initialize target action value function Q' with parameters θ' and policy π^*

for $i=1,2 \ldots N_{iter}$ **do**

$\theta_i^{\pi} \leftarrow \theta_{i-1}^{\pi}$

for $k=0,1, \ldots, N_{epi}$ **do**

Sample trajectory $\tau_k = \{s_t^i, w_t, a_t, s_{t+1}\}$

$(s_t^i, a_{1i_t}, a_{2i_t}, r_{1i_t}, r_{2i_t}) \leftarrow (\overline{\pi}_{\theta_i^{\pi}}, \pi_{\theta_i^{\pi}}, \tau_k)$

end for

$\theta_i^{\bar{\pi}} \leftarrow \theta_{i-1}^{\bar{\pi}}$

for $k=0,1, \ldots, N_{epi}$ **do**

Fig. 5 Proposed Algorithm Logic.

The initial parameters, θ_{π_0} are sampled from random distribution. For each iteration, N_{iter}, we sample the trajectories to estimate the state-action values and find parameters. We utilise the disturbance model represented in the reward function of the adversary to update the states. Then, for each episode we sample the trajectories of the augmented MDP parameters, this allows the optimisation of the robust MDP, following we update the states. Finally, the robust action Q' value is updated for each episode. Two exploration/exploitation strategies are presented: ϵ -greedy method chooses between the probability $1-\epsilon$, and a random probability, ϵ for the best action.

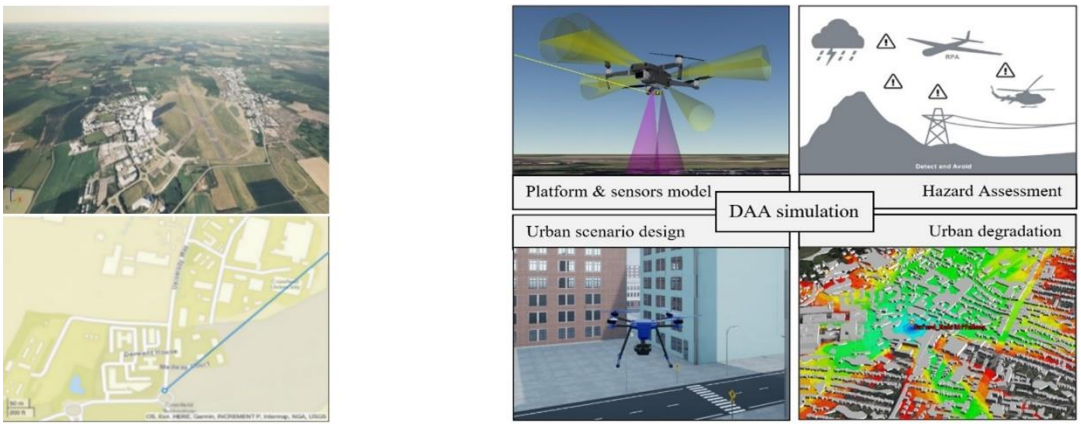
$$A \leftarrow \begin{cases} arg\max_{a} Q(s,a) & with\ probability\ (1-\epsilon) \\ a\ random\ action & \epsilon \end{cases} \tag{26}$$

IV. SIMULATION RESULTS

Based on the simulation results, this section addresses the challenges urban environments pose for sUAS. DAA explores practical considerations to support safer operations. The analysis is structured as DAA hazard assessment results through sense and avoid simulation, complemented by the communications and navigations identified challenges for DAA cooperative technologies. Finally, a set of applicable safe practices is listed based on the reviewed literature and obtained simulation results.

A. *Mission Environment*

For simulation purposes Unreal Engine/AirSim (Cesium World Dynamic) are used to build a custom 3D dynamic world map platform for the UAS, linked to Python. Utilising AirSim platform, three accurate quadcopter models (dynamic and aero-propulsion specifications), representative urban scenes (3D OSM Buildings) including sample urban objects (buildings, vegetation, vehicles) and airspace intruders (sUAS) can be modelled. The complex geometry and noise of real-world environments greatly impacts DAA performance (numerous obstacles) and signal propagation. Therefore, the dynamic real-world environment in AirSim/Unreal allows several practical considerations. Weather and noise are a critical enabler for the definition of practical considerations within the dynamic synthetic environment.



(a) (b)

Fig. 6 (a) Synthetic Simulation Environment, Cranfield (AirSim/Cesium and Bing Maps) (b) DAA simulation overview.

The obtained results provide a study over the extension of reinforcement learning to real-life experiments through optimization and robustness. In addition to ensure compliance with operational requirements with the airspace legislation and support sUAS. Software in the loop tests will comprise of several different simulation environments, including locations in AirSim/Cesium starting with Digital Aviation Research and Technology Centre, Cranfield, UK (DARTeC). For each test mission, the flight phases will consist of take-off, cruise, and landing. The flight phases factors define each mission's success, which are collected and statistically processed for major DAA events, enabling effective mitigations.

Analysis of the findings and simulation results leads to a holistic approach to implementation of sUAS operations, focusing on extracting critical DAA capability for safe mission completion, like minimum field-of-view and detection probability of the sensor system, and minimum manoeuvrability of the guidance and control system.

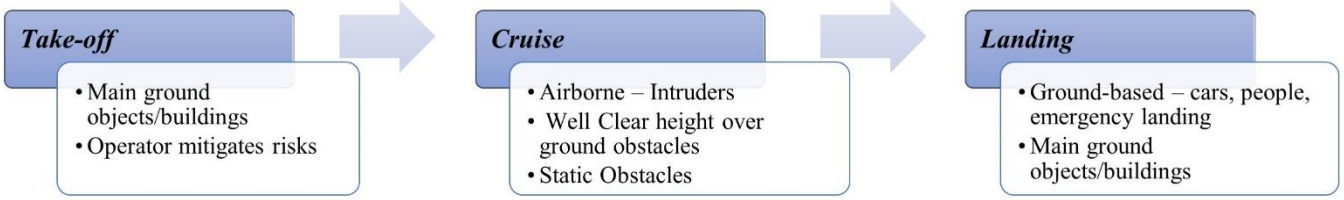


Fig. 7 Flight phases.

From the obtained results in simulation, the following points are presented to be considered for practical applications of the detect and avoid systems [27]:

1. **Noise or bias on sensors to detect obstacles:** The performance of the detection algorithm is dependent on the noise and bias the sensors are subject to. As a result, the avoidance algorithm is as well affected since the avoidance is conducted based on the target information, which is estimated by the detection algorithm of the sUAS. Noise and external unpredicted weather parameters can have a potential to directly affect obstacle detection, causing failure to avoid and assess risks hence, it is important to use a high-fidelity and noisy environment to do the simulation test before hardware in the loop testing.

2. **Casualty/Environmental Damage Estimation:** The importance of third-party damage estimation is essential for real-life sUAS integration. This allows an understanding of the prediction of damage, or the casualty imposed on the obstacles as well as the sUAS. Ideally, population densities should be considered for civil drone applications, for better integration for experimental validation.

3. **Computational delay***:* Fast and sudden approaching obstacles detection is conditioned by the computational capabilities from the DAA system, and therefore, performance must be assessed for delays reduction and mitigation. For heavy computing of software in the loop systems, it is crucial to be able to minimize the computational delay caused by complex environments. Minimization of lag will allow for better DAA performance.

4. **Weather effects:** The attitude control of the sUAS is affected by wind; additionally, rain can increase the noise on the camera image (detection). Light rain, while within tolerable flight safety conditions, can create noise on light-based sensors, and therefore reduce the DAA system capabilities.

B. Results

Based on the simulation results, this section addresses the RL agent training process and the relevant comparisons between the nominal problem and the proposed optimization method. Further we express the challenges of experimental tests and address the practical considerations stated in the previous chapter to support safer operations.

The analysis is structured as RL training results, RRL and RL performance metrics and the optimized RRL metrics. RL agent was trained over 3000 episodes, considering a scenario with one UAVs with multiple targets (dynamic and static). Fig. 8, the blue curve shows the cumulative reward gained at the end of each episode, the orange line demonstrates the average of the last 50 cumulative rewards. We observed that the trained RL agent converges at 600 episodes. Changing the learning rate affects the instantaneous reward as the variance rate of exploration increases which may result in several decreased rewards. Overall, less success is achieved. Prior to training, the hyperparameters were defined as given in Table. 4.

Table. 4 Hyperparameters of the DDQN agent.

Hyperparameter	Value	Description
training steps	500,000	total number of interactions with environment
minibatch size	32	stochastic gradient descent step size
replay memory	100,000	memory size of the most recent buffer
buffer size	500,000	improve sample efficiency for large buffer
target factor τ	0.01	update frequency from neural network to target
learning rate α	0.00025	optimizer learning rate
discount factor γ	0.98	balance rate of last reward and historical

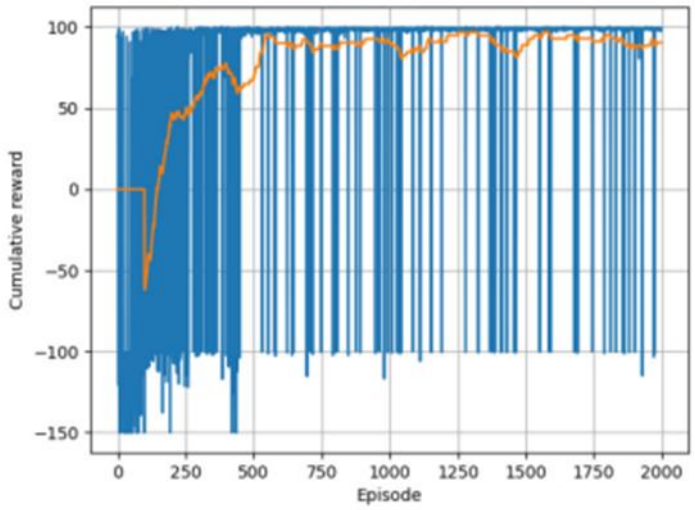


Fig. 8 Reinforcement Learning Accumulative Reward Training.

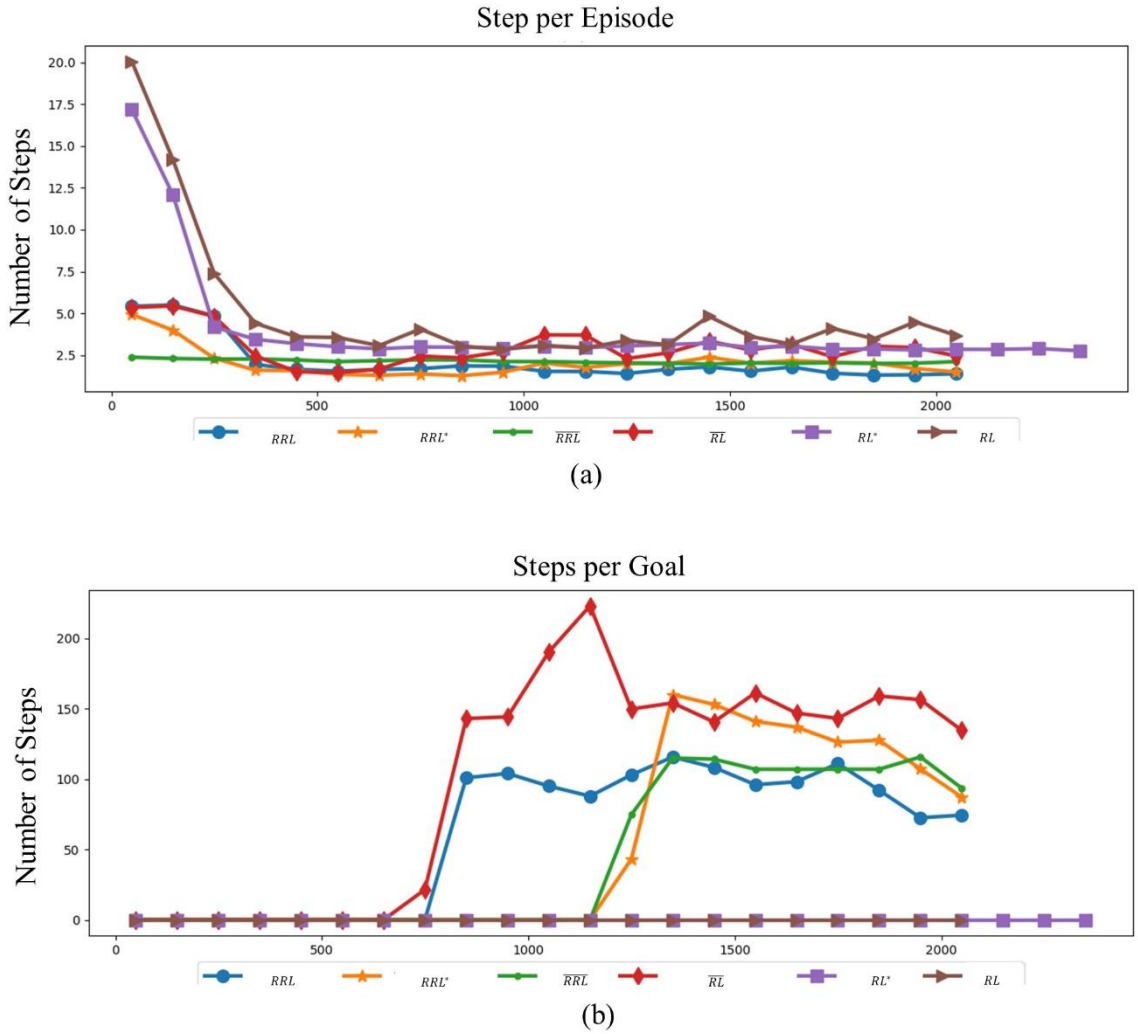


Fig. 9 (a) Number of steps per episode for different trained agents within robust reinforcement learning (RRL) and conventional reinforcement learning (RL) (b) Number of steps to reach goal for different trained agents within robust reinforcement learning (RRL) and conventional reinforcement learning (RL).

Fig. 9 represents the time steps for reaching goal and episode for RRL and RL agents. RRL agents defined in the plots are given as: robust reinforcement learning (*RRL)*, proposed optimised robust reinforcement learning (RRL^*) and robust reinforcement learning with a reward variation ($\overline{RRL}$). RL agents defined in the plots are given as: conventional reinforcement learning (*RL)*, reinforcement learning using loss function (RL^*) and reinforcement learning with a reward variation ($\overline{RL}$). When we start in state, *s,* we want to take the action that gives us the best total reward taking into account not only the current, or next state, but all possible next states until we reach the goal. These are the time steps, i.e., each action taken is done in a time step. And when we learn the policy, we try to consider as many time steps as possible to choose the best action.

Over 2500 episodes were trained to show the time-steps taken for each action. $\overline{RRL}$ and $\overline{RL}$ with a reward variation are represented in Eq. 27-28. For both RL agents, the revision of the reward function was due to the length of training and oscillation of instantaneous reward. Therefore, removing the event probability (P_E) term proved to be less oscillatory and more efficient. In addition, as can be observed from Fig. 9, changing the reward improves the steps taken for episodes which reduces the convergence significantly.

$$r_{t_1} = 1000 * a \ - (1000 * \beta) * P_E - D_{goal} + D_{correction} * y \tag{27}$$

$$r_{t_2} = 1000 * a \ - (1000 * \beta) - D_{goal} + D_{correction} * y \tag{28}$$

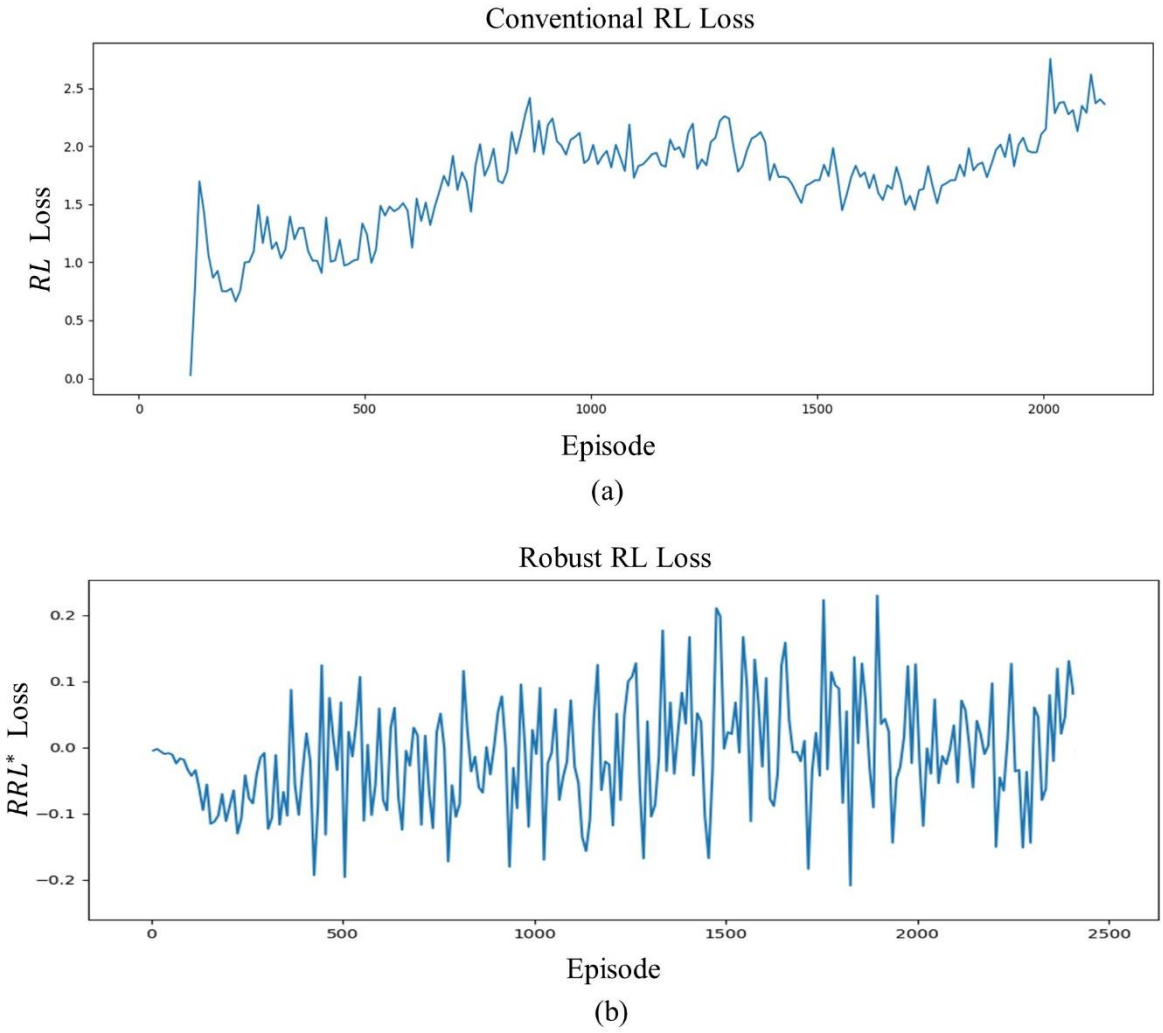


Fig. 10 (a) Loss of conventional reinforcement learning (RL) (b) Loss of optimized robust reinforcement learning (RRL*).

In Fig. 11 (c), we observed that the increase of the number of intruders to ten leads to reduce the percentage of success to arrive at the destination without collisions to 60% and 80% success rate for RL and for Robust RL, respectively.
The reason for this is that both RL and Robust RL does not account the dynamic changes of the UAVs.
Conventional RL has higher probability of collision as it does not consider change in dynamics of the environment setting whereas Robust RL allows the probability of unknown disturbances to be limited to a constraint to allow better avoidance.

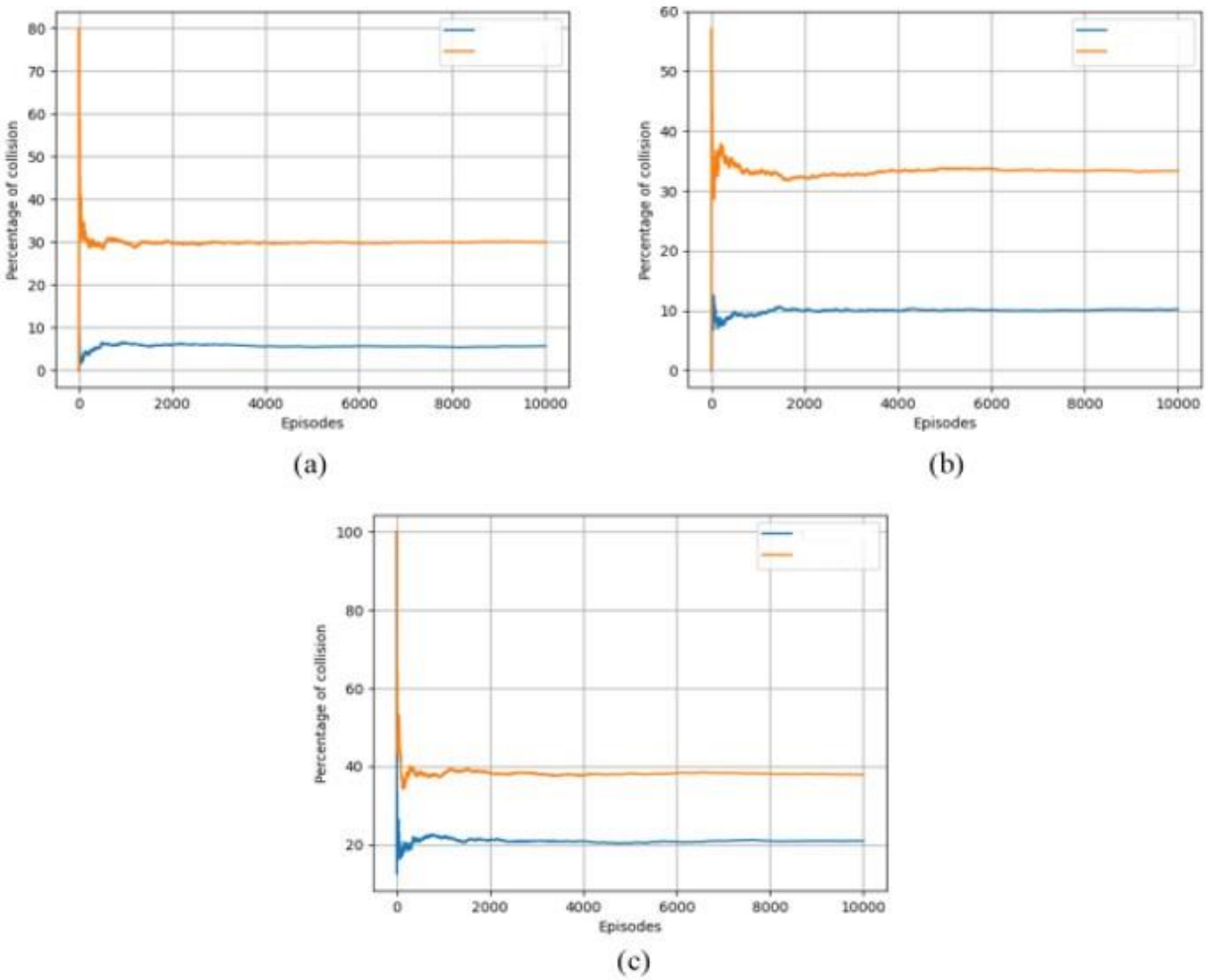


Fig. 11 Percentage of collision for Robust Reinforcement Learning and Conventional RL (a) Two intruders (b) Five intruders (c) Ten intruders.

V. CONCLUSIONS AND DISCUSSION

This paper presents a comprehensive review of the state-of-the-art detect and avoid (DAA) technologies in conjunction with the simulation of realistic urban scenarios for DAA potential challenges assessment. Different missions are designed and executed for representative scenes accounting for the common threads for obstacles in the sight of sUAS. Reinforcement learning training incorporated relevant factors such as noise on sensors, and dynamic real-world environments, including obstructed regions, complete the proposed simulation environment, complemented with DAA hazard assessment leading to effective threat identification.

The RL agents trained represent the success of collision avoidance and obstacle detection. Changes in the dynamic environment settings and implementation of dynamic obstacles have a great impact on the manoeuvre of a conventional RL agent**.** Conventional RL is impacted by 60-80% as more intruders are introduced to the environment, Robustness reduces the impact of the changes in the environment to the UAVs by 20%. Robust RL

has a better collision success rate compared to conventional RL agent. Robust RL is more successful due to factoring in the uncertainty parameters of the environmental changes within Airsim. The proposed method of the optimised RRL proved to be effective compared to both conventional and robust RL as the loss was found to be much lower.

In the future this work can be extended including **intruder** dynamics and to integrate in the **worst-case** scenarios regarding the intruder collision, by implementing and updating the policy hence the accumulative reward of the UAVs. Further improvements are needed for the collision **success rate** of the conventional RL and robust RL aiming for no collision. In addition, possible loss functions are needed for improvement of further uncertain parameters regarding the environment.

VI. ACKNOWLEDGEMENT

This work is supported by Thales UK and EPSRC funding, grant number 2454266.

VII. REFERENCES

[1] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction Second edition, in progress."

[2] Z. Hu, K. Wan, X. Gao, and Y. Zhai, "A Dynamic Adjusting Reward Function Method for Deep Reinforcement Learning with Adjustable Parameters," *Math Probl Eng*, vol. 2019, 2019, doi: 10.1155/2019/7619483.

[3] M. Marashi, A. Khalilian, and M. E. Shiri, "Automatic reward shaping in Reinforcement Learning using graph analysis," in *2012 2nd International eConference on Computer and Knowledge Engineering, ICCKE 2012*, 2012, pp. 111–116. doi: 10.1109/ICCKE.2012.6395362.

[4] A. Nilim and L. el Ghaoui, "Robust control of Markov decision processes with uncertain transition matrices," *Oper Res*, vol. 53, no. 5, pp. 780–798, Sep. 2005, doi: 10.1287/opre.1050.0216.

[5] M. Turchetta, A. Krause, and S. Trimpe, *Robust Model-free Reinforcement Learning with Multi-objective Bayesian Optimization; Robust Model-free Reinforcement Learning with Multi-objective Bayesian Optimization*. 2020. doi: 10.0/Linux-x86_64.

[6] K. Zhou, "ESSENTIALS OF ROBUST CONTROL," 1999.

[7] C. Kwan and F. L. Lewis, "Robust backstepping control of nonlinear systems using neural networks," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.*, vol. 30, no. 6, pp. 753–766, Nov. 2000, doi: 10.1109/3468.895898.

[8] *Renewable Energy Systems*. Elsevier, 2021. doi: 10.1016/C2019-0-00528-6.

[9] C. Finn, S. Levine, and P. Abbeel, "Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization," Mar. 2016, [Online]. Available: http://arxiv.org/abs/1603.00448

[10] R. Bellman, "A Markovian decision process," *Indiana University Mathematics Journal*, vol. 6, no. 4, pp. 679–684, 1957.

[11] C. J. C. H. Watkins and P. Dayan, "Q-Learning," 1992.

[12] M. Niranjan, "On-Line Q-Learning Using Connectionist Systems," 1994. [Online]. Available: https://www.researchgate.net/publication/2500611

[13] L. Xie, S. Wang, A. Markham, and N. Trigoni, "Towards Monocular Vision based Obstacle Avoidance through Deep Reinforcement Learning."

[14] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning." [Online]. Available: www.aaai.org

[15] M. U. de Haag, C. G. Bartone, and M. S. Braasch, "Flight-test evaluation of small form-factor LiDAR and radar sensors for sUAS detect-and-avoid applications," in *AIAA/IEEE Digital Avionics Systems Conference - Proceedings*, Dec. 2016, vol. 2016-December. doi: 10.1109/DASC.2016.7778108.

[16] L. Tai, S. Li, and M. Liu, "A Deep-Network Solution Towards Model-less Obstacle Avoidance." [Online]. Available: http://www.ros.org

[17] E. Ancel, F. M. Capristan, J. v Foster, and R. Condotta, "Real-time Risk Assessment Framework for Unmanned Aircraft System (UAS) Traffic Management (UTM)."

[18] J. U. Morimoto and K. Doya, "Robust Reinforcement Learning."

[19] J. Andrew Bagnell, A. Ng Y., and J. Schneider G., "Solving Uncertain Markov Decision Processes," 2001.

[20] J. Moos, K. Hansel, H. Abdulsamad, S. Stark, D. Clever, and J. Peters, "Robust Reinforcement Learning: A Review of Foundations and Recent Advances," *Mach Learn Knowl Extr*, vol. 4, no. 1, pp. 276–315, Mar. 2022, doi: 10.3390/make4010013.
[21] J. T. Barron, "A General and Adaptive Robust Loss Function," Jan. 2017, [Online]. Available: http://arxiv.org/abs/1701.03077
[22] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust Adversarial Reinforcement Learning," Mar. 2017, [Online]. Available: http://arxiv.org/abs/1703.02702
[23] T. Hiraoka, T. Imagawa, T. Mori, T. Onishi, and Y. Tsuruoka, "Learning Robust Options by Conditional Value at Risk Optimization."
[24] E. Derman, D. J. Mankowitz, T. A. Mann, and S. Mannor, "Soft-Robust Actor-Critic Policy-Gradient," Mar. 2018, [Online]. Available: http://arxiv.org/abs/1803.04848
[25] K. Frans, J. Ho, X. Chen, P. Abbeel, and J. Schulman, "Meta Learning Shared Hierarchies," Oct. 2017, [Online]. Available: http://arxiv.org/abs/1710.09767
[26] G. N. Iyengar, *Robust Dynamic Programming*, 2nd ed., vol. 30. INFORMS, 2005.
[27] V. C. Martinez *et al.*, "Detect and Avoid Considerations for Safe sUAS Operations in Urban Environments," in *AIAA/IEEE Digital Avionics Systems Conference - Proceedings*, 2021, vol. 2021-October. doi: 10.1109/DASC52595.2021.9594407.

https://dspace.org/

Optimization of a robust reinforcement learning policy

Ince, Bilkan; Shin, Hyo-Sang; Tsourdos, Antonios

2023-01-19
American Institute of Aeronautics and Astronautics (AIAA)
https://doi.org/10.2514/6.2023-0967
Ince B, Shin H-S, Tsourdos A. (2023) Optimization of a robust reinforcement learning policy. In: AIAA SciTech Forum 2023, 23-27 January 2023, National Harbor, Maryland, USA. Paper number AIAA 2023-0967

Downloaded from a DSpace repository. https://dspace.org/