Deep Reinforcement Learning for Drone Conflict Resolution Under Uncertainty

Sasha Vlaskin, Emmanuel Sunil, Dennis Nieuwenhuisen
Aircraft Operations, Air Traffic Management and Airports
Netherlands Aerospace Centre
Amsterdam, the Netherlands
{sasha.vlaskin, emmanuel.sunil, dennis.nieuwenhuisen}@nlr.nl

Muhammad Fazlur Rahman, Joost Ellerbroek, Jacco Hoekstra
CNS/ATM, Operations and Environment
TU Delft
Delft, the Netherlands
{m.f.rahman, j.ellerbroek, j.m.hoekstra}@tudelft.nl

***Abstract*—Drone operations are expected to become a key facet of urban life, through contributions to emergency response missions, alleviation of urban traffic through parcel delivery, and aerial inspection. High demand is predicted to cause traffic densities never before seen in classical aviation, exceeding human capabilities and requiring autonomous solutions for separation management. In flight, drones are to perform maneuvers to avoid other vehicles - this is Conflict Resolution. In most research, the Conflict Detection and Resolution (CD&R) component is modeled without sensor error, which is unrealistic, as the State-Based conflict detection is to rely on GNSS-derived position information. Previous work has shown that conventional methods such as the Modified Voltage Potential (MVP) are negatively impacted by this error. This paper examines the applicability and performance of Reinforcement Learning (RL) for this conflict detection and resolution task, with positional error modeled. The RL model outputs velocity and heading commands (actions) on the basis of its state information, and that of other vehicles. Two models are trained - one with, and one without gaussian noise applied to the observed vehicle position. Both of these are tested against the MVP algorithm as a benchmark. The difference between the RL model trained with and without noise is minor in terms of losses of separation (safety). Compared to MVP, the model consistently shows a lower loss of separation count, demonstrating better noise robustness. The RL models favor pure speed changes to resolve conflicts, while staying on the same course.**

***Keywords*—drones, U-Space, conflict detection and resolution, reinforcement learning, uncertainty**

I. Introduction

By the year 2050 [1], 7 million drones are expected to operate in European low-level airspace. The main drivers are expected to be drone delivery and other important applications such as emergency response, causing unprecedented traffic volumes that exceed the capacity of conventional air traffic control. A largely autonomous *system* [2] therefore needs to take this task on, with centralised pre-flight elements handling the strategic (pre-flight) deconfliction of traffic. If there is a deviation from these plans due to uncertainty, decentralised (on-board) components are needed to tactically separate the vehicles. A key means of doing this is state-based Conflict Detection and Resolution (CD&R), which resolves conflicts as they occur. While heavily researched and proven to work in stringent conditions, most current CR methods neglect sensor error or communication uncertainty altogether, assuming perfect state knowledge, with the noise-induced error to be solved iteratively. In reality, uncertainty arises in part due to the communication and position determination methods that are used for sharing state information. Drones are expected to communicate state information using the Automatic Dependent Surveillance - Light (ADS-L) protocol [3], similar to that used on commercial aircraft (ADS-Broadcast). The automatic component refers to the fact that aircraft broadcast their information periodically to other aircraft and drones - this can be lost in transmission. More crucially, the word *dependent* alludes to the state information being in large part derived from Global Navigation Satellite System (GNSS) sensors. These sensors have an associated error, therefore all position and speed data derived from them also carries the same uncertainty level. Studies [4] have shown that position uncertainties can impact the ability of Conflict Detection and Resolution algorithms to separate drones successfully. Notably, some algorithms are more robust than others, with the Modified Voltage Potential (MVP) outperforming Velocity Obstacle (VO) methods in intrusion prevention rate and intrusion severity [4]. This becomes especially important in the drone context, where the scale of GNSS error is greater compared to the drone's protected zone. Two open questions that this work aims to answer are:

1) Can a learning-based model outperform a classical iterative method (MVP) in terms of safety on a conflict resolution task with GNSS sensor error?
2) Does training a reinforcement learning model for CR on episodes with GNSS position noise improve its resultant performance?

This work will train an RL model for this CD&R task, in a BlueSky-Gym [5] PettingZoo [6] environment, and benchmark the resulting model against the Modified Voltage Potential algorithm.

II. Methodology

A. RL and Simulation Environment

The simulated environment used in this work models several flights inside a polygonal airspace sector, which must safely reach their destinations outside the sector. The randomly-initialized vehicles are denoted by the small rectangles, and are surrounded by a circle denoting *(*half) of their protected zone (thus when the circles overlap, there is a loss of separation). These must fly to waypoints, set outside the boundary of the sector in which they are initialized, shown as the outer polygon. The situation can be visualized in Figure 1.

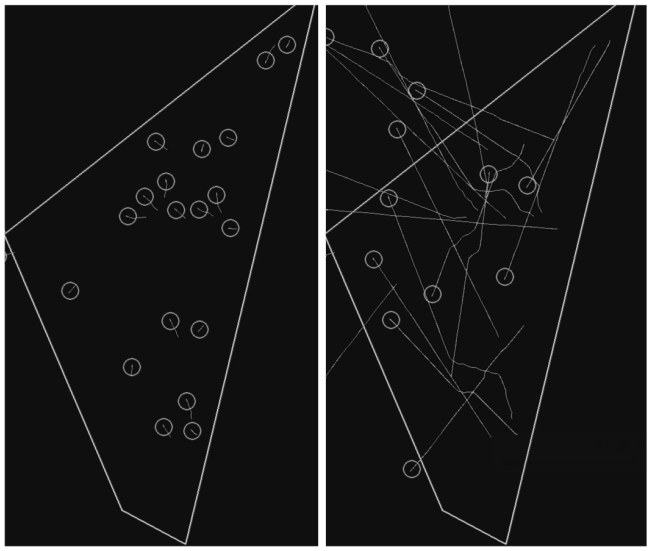


Figure 1. Environment Visualization. The left image shows the starting conditions, the right shows the end of the episode. The trails show the paths taken by each of the agents.

B. Reinforcement Learning Algorithms

Soft Actor Critic (SAC) [7], as implemented in detail in [8], is used as the principal algorithm in this work. Initially, Proximal Policy Optimization (PPO) [9] was also attempted for this task, but only a single-agent implementation was feasible with Stable-Baselines3 [10] and BlueSky-Gym [5]. It was found early on that a single-agent implementation for both SAC and PPO [9] showed worse performance when trained with position noise, and especially trailed in performance when compared to the Modified Voltage Potential baseline. This work thus utilizes an attention mechanism [11] [8] instead of a fully-connected neural network, trained with the SAC algorithm.

C. Attention Mechanism

The attention mechanism [11] used is the *additive* attention variant. While the state of the art for machine translation is set by scaled dot-product attention mechanisms [11], the additive attention variant is shown to work better [8] for CR tasks, as the dot product operation produces certain undesired effects, such as setting the attention score of traffic at perpendicular paths to zero. In essence, this mechanism allows for variable input sizes like Recurrent Neural Networks (RNNs) would, but with the removal of the inherent inefficiency due to their recurrent nature. The attention mechanism uses Query $\mathbf{Q}$, Key $\mathbf{K}$, and Value $\mathbf{V}$ in the computation of the hidden state h. For each drone, the Query $\mathbf{Q}$ is computed from the ownship state observation, while the Key $\mathbf{K}$ and Value $\mathbf{V}$ are derived from the relative observations from other agents.

For each drone, let the *ownship observation* be $X \in \mathbf{R}^{d_x}$ and the *relative states of other agents* be $Y = [y_1, \ldots, y_n] \in \mathbf{R}^{n \times d_y}$. The additive attention projections are then computed as:

$$Q = W_q X, \quad K_i = W_k y_i, \quad V_i = W_v y_i, \quad i = 1, \ldots, n$$

where $W_q \in \mathbf{R}^{d_a \times d_x}$, $W_k, W_v \in \mathbf{R}^{d_a \times d_y}$ are learnable weight matrices. The attention score for each other agent is calculated as:

$$e_i = v_a^\top \tanh(Q + K_i), \quad \alpha_i = \frac{\exp(e_i)}{\sum_j \exp(e_j)}$$

and the *context vector* representing the attended information is

$$C = \sum_i \alpha_i V_i$$

For *multi-head attention*, multiple independent sets of Q, K, V projections (heads) are computed in parallel:

$$\text{head}_h = \sum_i \alpha_i^{(h)} V_i^{(h)}, \quad h = 1, \ldots, H$$

The outputs of all heads are concatenated and linearly projected to produce the final hidden representation:

$$H_{\text{attn}} = W_o \, \text{Concat}(\text{head}_1, \ldots, \text{head}_H)$$

This representation is then used to compute the *action output* for the drone policy, combining attended information from all other agents in a single, parallelizable step.

D. Observation Vector

A single observation vector consists of the ownship state, as seen in Table I. This vector is computed for each vehicle independently, and added to the *global* observation vector. The attention mechanism, introduced in section II-C, computes the importance of the relative states of the other vehicles. As opposed to a fully-connected neural network model, the relative states are computed within the module itself, and thus do not need to be explicitly given in the observation vector of the environment at hand. The x and y position (x, y), velocity components (v_x, v_y) and the airspeed are standard components. The drift angle δ, defined as the difference between the target heading to the defined waypoint and the current heading is shown in Figure 2.

TABLE I: Observation Vector

Category	Variable	Description	Size
Ownship	$\cos(\delta)$	Cosine of the drift angle	1
	$\sin(\delta)$	Sine of the drift angle	
	v	Ground Speed (m/s)	
	x	x-position (m)	
	y	y-position (m)	
	v_x	x-component of velocity (m/s)	
	v_y	y-component of velocity (m/s)	

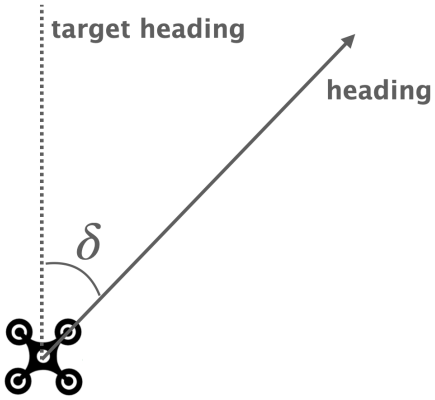


Figure 2. Drift angle definition

E. Action Space

The problem's action space is horizontally constrained. While it is possible to perform resolutions in all 3 dimensions, therefore including an altitude change as a possible action, other work has shown that this causes high unpredictability, especially in dense urban areas. The vertical component of U-Space is also expected to be highly constrained, due to a low maximum altitude allowed for these autonomous vehicles (namely below 120 meters). For this reason, the action space is limited to heading and velocity changes, thus making the problem 2-dimensional. The action space, consisting of a heading and speed change, as well as its limits, are shown in Table II.

TABLE II: Allowed actions and their bounds

Action	**Bounds**
Heading Change	[-1,1] $\rightarrow$ [-22.5°, +22.5°]
Speed Change	[-1,1] $\rightarrow$ [-10 kts, +10 kts]

F. Reward Function

The reward function, evaluated at every timestep, is given in Equation 1.

$$r = R_{drift} \cdot \phi_{dr} + \begin{cases} 0 & \text{if not in intrusion} \\ -1 & \text{if in intrusion} \end{cases} \quad (1)$$

There are several components to this function. R_{drift} (with value of -0.1), penalizes every radian of drift ϕ_{dr}, ensuring path efficiency. Lastly, every timestep spent in intrusion is penalized with a value of -1. This intrusion, or Loss of Separation penalty, is the safety term. Note that this is allocated per simulation timestep and therefore the total intrusions metric is more accurately described by "simulation times-teps spent in intrusion".

III. Experiment Setup

A. Randomization of Spawn Parameters

The spawn parameters are randomized at the start of every episode, thus meaning that every individual episode (full execution of the maneuver) is different. This is done to prevent *overfitting*, that is the model learning a sequence (only applicable to the initial conditions) instead of a generalized policy. These randomized parameters are:

- Sector Geometry, bounded by an upper and lower area bound (0.25 to 1 NM^2)
- Initial vehicle position within the sector
- Destination Waypoint position (outside the sector)

B. Simulation and Software Tools Used

This project uses a branch of BlueSky-Gym [5], more specifically a *PettingZoo* [6] implementation which is a set of environments and wrappers enabling Reinforcement Learning on the BlueSky simulator [12].

C. Modified Voltage Potential

Once the models are trained, they must be compared against existing geometric algorithms that are already proven to work for the conflict resolution task. Such algorithms include variations of the Modified Voltage Potential [[13], [14], [15]] and Velocity Obstacle [16] methods. Previous work [4] has shown that the Modified Voltage Potential (MVP) method outperforms the Velocity Obstacle algorithm in scenarios with sensor uncertainty, and that this robustness is inherent to the algorithm implementation. Therefore, the most robust algorithm has been selected to be tested against - MVP.

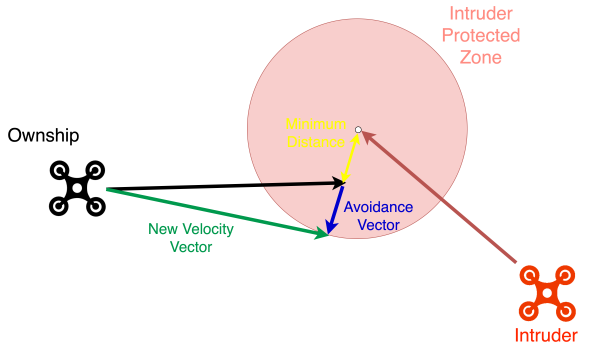


Figure 3. MVP resolution, shown graphically (adapted from [17])

Conflict detection with MVP is *state-based*, with the current state of the vehicle extrapolated with a given lookahead time. Thus, if it is expected to lose separation to another vehicle within that interval, a *conflict* is detected. Modified Voltage Potential resolves conflicts iteratively, by performing this state-based check periodically and applying a velocity change. The resultant maneuver in the velocity plane can be visualized in Figure 3.

D. Control Variables

The control variables are kept constant for both the training and testing phase.

- *Drone Performance Model*: the DJI Matrice 600 quadcopter performance model is used [12]. Especially important to note is the ability of this drone to hover, since it is a quadcopter.
- *Environment Geometry:* despite the relative intruder positions being randomized, the randomized test sector area, and the drone spawn boundaries and velocities are kept constant.
- *MVP Algorithm Settings:* the look-ahead time ($t_{lookahead}$) for the MVP CR algorithm is set to 15 seconds, and the Protected Zone Radius is set to 0.15 nautical miles (approx. 280 m).
- *Number of Vehicles:* this is set to 20 for training, and 30 for testing conditions.

E. Independent Variables

The independent variables are:

- *CR Module Used:* there are three usable conflict resolution modules - two RL-derived attention models (trained on position noise and without position noise), and one classical CR method, the Modified Voltage Potential.
- *Testing Observation Conditions:* this defines whether the environment on which testing occurs includes positional noise or not.

Simply put, the goal is for each module type (RL with and without noise) to be tested under both noisy and ideal observation conditions, across multiple noise levels defined by their standard deviations.

F. Dependent Variables

The dependent variables are:

- Timesteps in Loss of Separation (LoS) count [-]
- Average Drift [rad]
- Intrusion Severity [-]

These values describe the performance of the CR algorithms - the LoS timestep count is the safety metric, and the average drift shows the efficiency of the algorithm, as excessive heading inputs are not desired. Intrusion severity highlights the extent to which the LoS were unsafe.

G. Assumptions Made

Several of the test parameters, such as the protected zone radius R_{pz}, are direct results of assumptions and simplifications. Values of 50m are often used in similar drone-related work [4], [18], while here 0.15 nautical miles (approximately 280 meters) is used. This is selected through trial and error, and is linked to the update period Δt_{update}. To make the comparison fair, both the RL models and MVP are allowed to take actions every Δt_{update} seconds, which in this case is 5 seconds. This value is a computational consideration, but also an approximation of how often an ADS-L update could be received. Crucially, the distribution governing the position noise is assumed to be Gaussian, with a mean at zero. Coordinate conversions are linearized for simplicity. Finally, the standard deviation of the position noise, σ_{noise}, is taken to be 3.5 meters. This is a reasonable estimate from several technical briefs and GNSS L-band assessment works, notably [19], where RMS values are quoted to be approximately 3 meters.

H. Summary of Test Settings Used

The settings used for the test phase of the simulation, kept constant across methods, are as follows: The R_{pz} denotes

Variable	Value	Unit
R_{pz}	0.15	NM
n_{veh}	20	—
dt_{look}	15	s
$\Delta t_{\mathrm{update}}$	5	s
σ_{noise}	3.5	m

TABLE III: Summary of parameters used in the simulation.

the protected zone radius, n_{veh} is the number of vehicles in training/testing, dt_{look} is the lookahead time, $\Delta t_{\mathrm{update}}$ is the action update period, and σ_{noise} is the standard deviation of the position noise.

I. Noise Application and Conflict Detection

The position noise is applied somewhat differently across MVP and the CR models, and therefore merits its own explanation. The MVP implementation in the BlueSky simulator consists of an conflict detection module (*statebased.py*) and a conflict resolution module, in this case *MVP.py*. The approach here is to leave the MVP module as is, but only modify the observation in *statebased.py*. The logic for the MVP test runs is applied in Algorithm 1. For the learning-based models, testing and training need to have modeled

noise included. The noise is applied exactly as prior in Algorithm 1, except that the spherical coordinate conversion is not needed, as the positions in the observation vector are encoded in cartesian coordinates by default. Naturally, the LoS themselves are computed from the unperturbed ground-truth data for all cases. Thus, only MVP and the models "observe" the perturbed position, while LoS are computed from the actual position.

Algorithm 1: Applying Gaussian Position Noise to State-Based CD

Input: Ownship position $(\text{lat}_o, \text{lon}_o)$, Intruder position $(\text{lat}_i, \text{lon}_i)$, noise standard deviation σ in meters

Output: Noisy positions $(\text{lat}'_o, \text{lon}'_o)$, $(\text{lat}'_i, \text{lon}'_i)$

foreach *aircraft* $a \in \{ownship, intruder\}$ **do**

Sample noise in meters: $(dx, dy) \sim \mathcal{N}(0, \sigma^2 I_2)$;

Latitude at reference point: $\phi \leftarrow a.\text{lat}$;

Convert to degrees using spherical Earth approximation:;

$$d\phi \leftarrow \frac{dy}{111320.0};$$

$$d\lambda \leftarrow \frac{dx}{111320.0 \cdot \cos(\phi \cdot \pi/180)};$$

Apply noise:;

$a.\text{lat}' \leftarrow a.\text{lat} + d\phi$;

$a.\text{lon}' \leftarrow a.\text{lon} + d\lambda$;

IV. Results and Discussion

A. Training Results

Both variants of the model are shown to have reached asymptotic performance, as seen in the reward evolution curve in Figure 4. In turn, Figures 5 and 6 show the Intrusion timesteps and Drift evolution. These plots are smoothed with a window of 300 episodes.

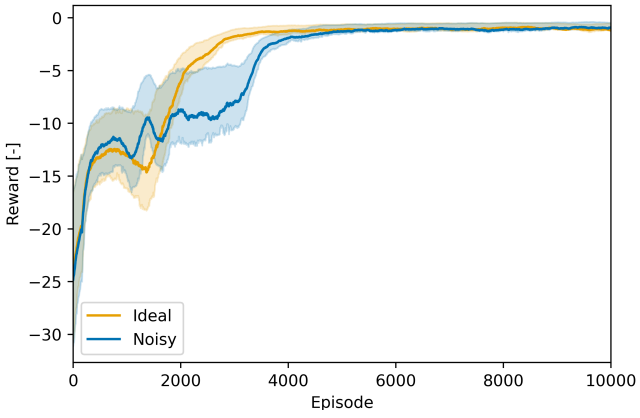


Figure 4. Reward Evolution

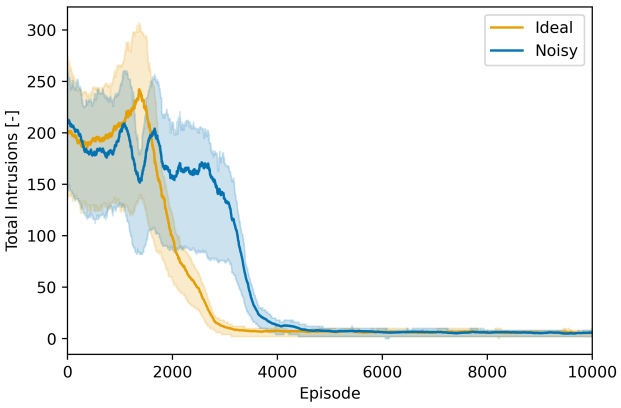


Figure 5. Intrusions Evolution

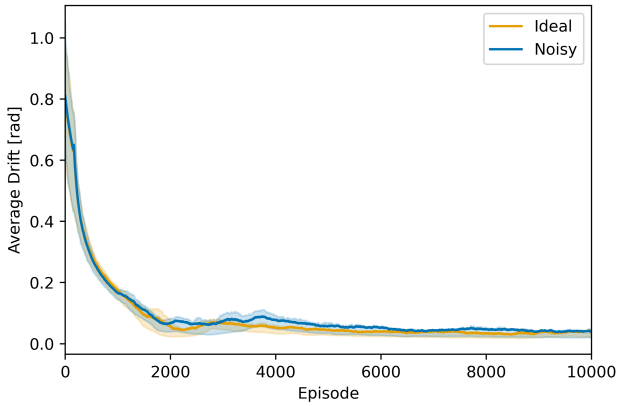
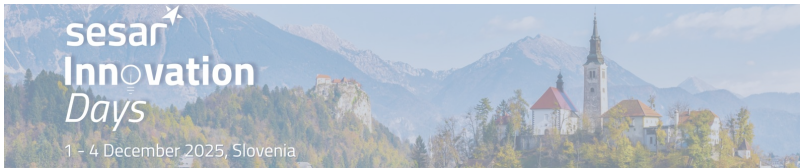


Figure 6. Drift Evolution

It is clear from this that the drift decrease is more stable than that of the intrusion count, and that a low drift policy is obtained from 2000 episodes onwards. The less opaque segments bounding the lines show the inter-quartile range of the data, and the curves themselves a moving average of the reward. Here, instability in policy is seen for both training conditions, as the inter-quartile range (IQR) is expansive for both. Towards 4000 episodes, both models stabilize in learning, with the IQR decreasing.

B. Policy Analysis

The resulting policies of ideal-trained (*log_cu*) and the uncertainty-trained (*log_uu*) can be compared in terms of the actions they took during testing. Over the 5000 test episodes, the heading and speed actions from the model are averaged across all agents, to return the kernel density estimates in Figure 7 and Figure 8 for the heading and speed respectively. From these figures, it is seen that the distributions in heading changes differ minimally, and the values are generally low. This means that the policy avoids heading changes when possible, and relies on speed inputs mostly, as confirmed by the Kernel Density Estimate (KDE) plot for the speed changes in Figure 8. Note that all speed actions are negative - meaning that the overall learned policy is to slow down in case of conflict, while ideally minimizing heading input. A small distinction seen in Figure 7 is that the heading input is mostly positive for the noise-trained model, showing



a tendency to turn right, whereas the ideal-trained model (blue) shows negative tendency, thus to turn left. The mean is squarely at zero - meaning that most episodes did not require heading changes.

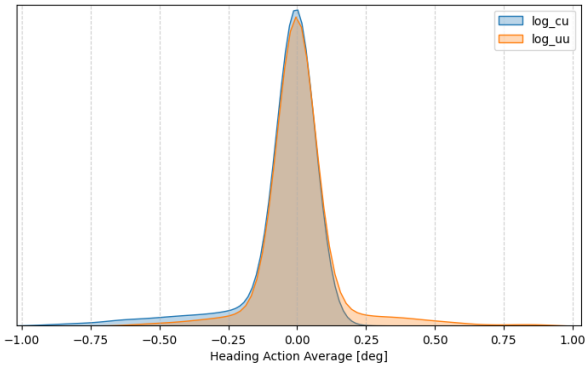


Figure 7. Average Heading Action KDE

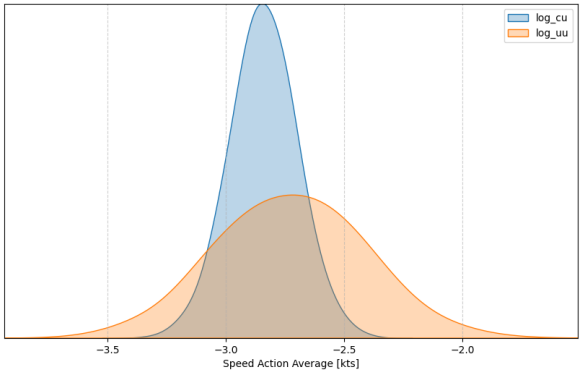


Figure 8. Average Speed Action KDE

C. Benchmark Analysis - Low Noise

Now that the models are trained, they are tested against the MVP algorithm (III-C). This is done by running both models and the MVP algorithm on 5,000 randomly generated scenarios, first with and then without positional error/noise. The noise value is set to a realistic GNSS value, as described in section III-G. The results themselves are seen below in the form of box plots, for total reward (Figure 9), Average Intrusion Timesteps (Figure 10) and Average Drift (Figure 11). These show a higher timestep in intrusion for MVP, in both noisy and ideal conditions. While the mean of the average track deviation (drift) per episode in the case of MVP is only the second highest, slightly trailing the noisy-trained model, it is less consistent overall, with some clear outliers and a larger inter-quartile range. In terms of total reward, the differences in mean are again very small.

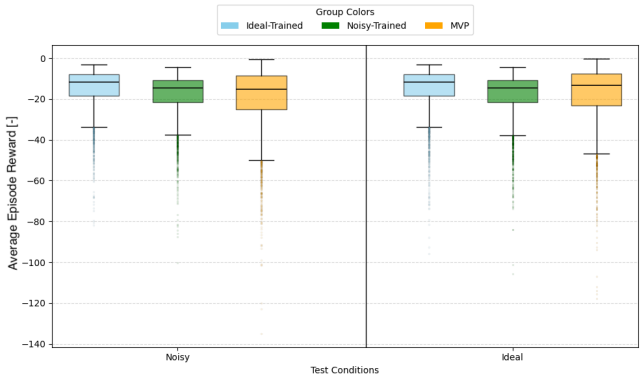


Figure 9. Total Reward

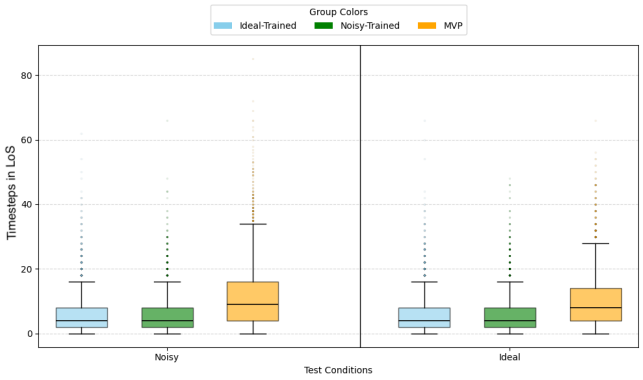


Figure 10. Intrusion Timesteps

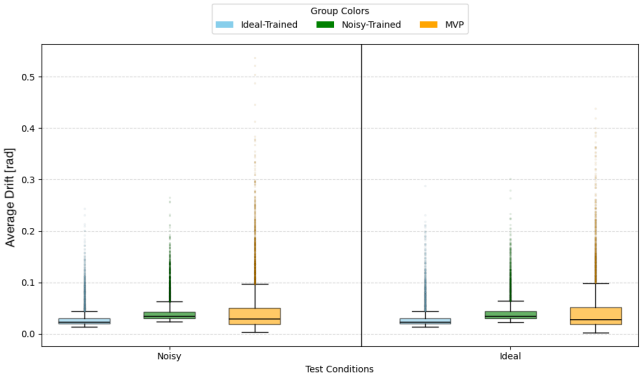


Figure 11. Average Drift

D. Benchmark Analysis - High Noise

Testing for a realistic noise level shows that a large gain is not obtained by training on a noisy environment. Therefore, the model is retrained and retested against MVP with a higher position noise standard deviation of σ_{noise} of 35 meters. A histogram of total intrusion time per episode for a standard deviation in position noise σ_n of 35 m is given in Figure 12. The x-axis bounds the bin bounds, and the count of corresponding instances to that intrusion time range is shown on the y-axis. The kernel density estimates for each of the methods are also drawn over the data as continuous lines.

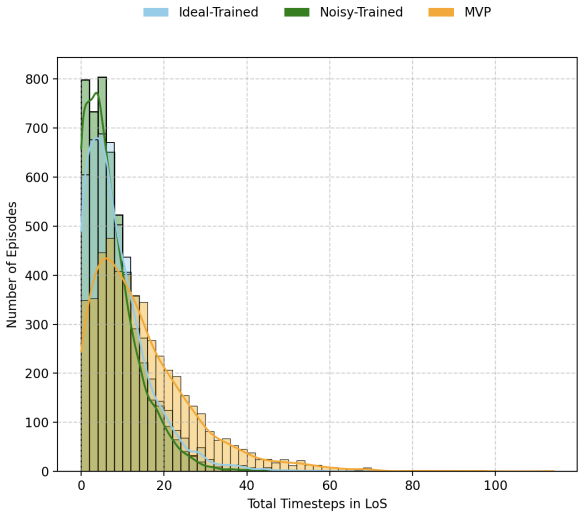


Figure 12. Intrusion Timestep Histogram, $\sigma_n = 35[m]$

What is clear in Figure 12 is that for episodes totaling 20 intrusion timesteps or more, the count is above 50% higher in the case of MVP than for the RL-based methods. Isolated instances with above 50 intrusion timesteps with those ranging all the way to 80 are also present for MVP while largely absent for the RL methods.

Ultimately, unique intrusion instances were also summarized, as were the intrusion severities. The intrusion severity is defined in Equation 2, where d_{CPA} is the distance at closest point of approach, and R_{PZ} is the protected zone radius.

$$I_{\text{sev}} = \frac{I}{R_{\text{PZ}}} = \frac{R_{\text{PZ}} - d_{\text{CPA}}}{R_{\text{PZ}}} \tag{2}$$

Table IV shows the means of LoS instances per method, as well as their severity.

TABLE IV: Unique LoS and Intrusion Severity Summary

Train	Test	Intrusions	Severity
Clean	Clean	2.69 ± 1.61	0.231 ± 0.158
Clean	Uncertainty	2.75 ± 1.67	0.232 ± 0.161
Uncertainty	Clean	2.45 ± 1.51	0.255 ± 0.167
Uncertainty	Uncertainty	2.43 ± 1.50	0.256 ± 0.168
MVP	Clean	2.99 ± 1.81	0.285 ± 0.162
MVP	Uncertainty	3.54 ± 2.17	0.288 ± 0.162

Thus, it is seen that the unique intrusion count is lowest for the uncertainty-trained model. Note that the number of unique intrusions is close across methods. Table V shows the total time spent in Los across all episodes, as well as the mean time in LoS per episode, as well as its standard deviations.

TABLE V: LoS timesteps Summary by Train/Test Regime

Train	Test	Total	Mean ± Std
Clean	Clean	40634	8.13 ± 7.16
Clean	Uncertainty	42590	8.52 ± 7.33
Uncertainty	Clean	34464	6.89 ± 6.46
Uncertainty	Uncertainty	34334	6.87 ± 6.41
MVP	Clean	46794	9.36 ± 7.96
MVP	Uncertainty	72062	14.41 ± 12.30

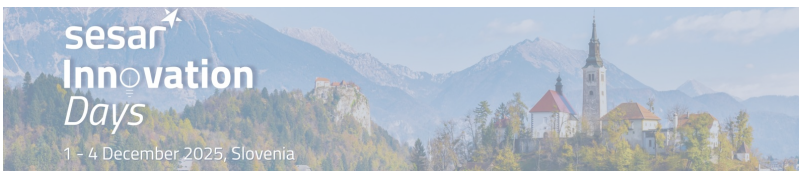
For unique LoS instances, MVP shows 40% more when compared to the Uncertain-trained model, meaning the gap is larger in mean time in LoS, where the average LoS time for MVP is twice as long as that for the Uncertain-trained model.

E. Discussion and Operational Considerations

Some important qualitative observations and remarks are required to attempt to explain where the differences in performance between the RL models and MVP arise. Firstly, the nature of the methods themselves causes this difference. MVP's algorithmic formulation by definition employs a fixed policy - its resolution velocity is a balance of heading and speed changes. Contrary to this, results in the policy analysis clearly show the RL methods mostly relying on pure speed changes, which appear to be a preferred policy given that drones do not have a stall speed and can decelerate to a stop when needed. When on the verge of a LoS, under uncertainty especially, simply slowing down may well be a better policy to follow for drones. While implicitly coordinated (with the same priority rules across all vehicles), MVP is a purely iterative and *decentralized* method, operating on each drone independently. Therefore, the MVP algorithm itself is independent of the scenario in which it is operating, but also cannot adapt to it, unlike the RL models. Practically, the RL models are trained on high-density traffic scenarios, and as a result collect specific experience of how to deal with multi-vehicle conflicts, providing a strategy that is inherently cooperative. This means that without extensive hand-tuning of the parameters that MVP requires, such as its look-ahead time, protected zone radius, selected priority rules and safety margins, it is going to be difficult for such a method to match the performance of the RL methods. On the other hand, the behavior that MVP exhibits is purely deterministic and analytically explainable. This is a big strength for explainability purposes, as such a method is then likely preferable for real-world use, as its response in unseen conditions is predictable. This is a barrier for the RL models - explainability is key for real-world applications, which means in the short term, geometric methods are preferable. Future work is needed to make this more explainable, through, for instance, state-action heatmaps, and detailed a-posteriori analysis of individual agent actions and contributions to given scenarios. Finally, on generalization - the scenarios and environment here follow a random initialization procedure [20] in order to avoid overfitting [21] (learning a pattern to solve conflicts instead of a general strategy). While this avoids overfitting within this scenario, its ability to generalize for different types of traffic flows (for crossing traffic for instance [22] [23]) is yet to be tested.

V. Conclusion and Recommendations

This work's objective was to develop, train and compare an RL approach to Conflict Detection and Resolution under uncertainty, with an action space analogous to that given to a method such as MVP and VO. This has been largely



achieved for the given geometry, as both the ideal and noise-trained models have reached stable asymptotic performance. When compared to the Modified Voltage Potential algorithm on noisy scenarios, the RL-derived model performs noticeably better in terms of raw intrusion count, demonstrating better safety. It is also less impacted by noise overall, as the results show. As for the research questions initially posed, they can be answered as follows:

1) A learning method can outperform a classical MVP implementation under sensor error, under certain conditions.
2) Training a reinforcement learning model on noisy position data yields a visibly different policy, more prone to heading changes, however only a small margin safer than that trained in ideal conditions.

The difference in performance is likely due to the RL methods favoring negative speed changes as compared to MVP. From an operational standpoint, the flexible nature of the model resulting from the RL methods means that it can be deployed centrally from the ground, or act as an onboard self-separation method, in compliance with U-Space self-separation requirements. Nevertheless, certain limitations will be addressed in further work, with several crucial improvements in both the comparison to MVP and to general model performance. More extensive testing is required, with different conflict geometries, protected zones, and action periods, in order to improve the generalizability and performance of the RL methods. Analyses of other scenarios, such as purely pairwise conflicts, merging tasks and other geometry will be performed. It also hypothesized that training across a greater variety of scenarios will increase overall performance and generalizability, and will also be done in future work. The realism of the noise model will also be increased for future work, taking more varied noise sources into account, as well as applying noise to velocity. Most crucially, future work aims to broaden the scope of analysis by evaluating the LoS geometries across all methods tested here, as well as quantifying the exact effect of training noise on resulting RL policies, in order to make them more explainable and adaptable for use in U-Space self-separation. Finally, the effects of the resolution maneuvers taken by the trained agents to resolve conflicts result in the slowing of traffic. For this reason, further analysis is needed in order to evaluate the impact of this strategy on sector capacity as compared to standard geometric methods.

References

[1] SESAR Joint Undertaking, "European drones outlook study: unlocking the value for europe.," 2017.

[2] S. Vlaskin, E. Sunil, D. Nieuwenhuisen, J. Ellerbroek, and J. Hoekstra, "Autonomous separation management system for drones through optimal layer interactions," in *Proceedings of the ICRAT 2024 Conference*, July 2024. Conference Paper, Nanyang Technological University, Singapore.

[3] European Union Aviation Safety Agency (EASA), "Technical Specification for ADS-L transmissions using SRD-860 frequency band (ADS-L 4 SRD-860)," Tech. Rep. ED Decision 2022/024/R, European Aviation Safety Agency, Cologne, Germany, Dec. 2022.

[4] M. F. Rahman, A. Vlaskin, J. Ellerbroek, and J. Hoekstra, "Autonomous separation in u-space: Assessing the impact of position uncertainty," in *Proceedings of the SID 2024 Conference*, November 2024. Conference Paper.

[5] D. Groot, G. Leto, A. Vlaskin, A. Moec, and J. Ellerbroek, "Bluesky-gym: Reinforcement learning environments for air traffic applications," 2024.

[6] J. Terry, B. Black, N. Grammel, M. Jayakumar, A. Hari, R. Sullivan, L. Santos, R. Perez, C. Horsch, C. Dieffendahl, N. Williams, and Y. Lokesh, "Pettingzoo: Gym for multi-agent reinforcement learning," 2021.

[7] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[8] J. Groot, J. Ellerbroek, and J. Hoekstra, "Comparing attention-based methods with long short-term memory for state encoding in reinforcement learning-based separation management," *Engineering Applications of Artificial Intelligence*, vol. 159, 11 2025.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[10] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[12] J. M. Hoekstra and J. Ellerbroek, "Bluesky ATC simulator project: an open data and open source approach," in *Proceedings of the 7th international conference on research in air transportation*, vol. 131, p. 132, FAA/Eurocontrol USA/Europe, 2016.

[13] J. M. Hoekstra and J. Ellerbroek, "Aerial robotics: State-based conflict detection and resolution (detect and avoid) in high traffic densities and complexities," *Curr Robot Rep*, vol. 2, pp. 297–307, 2021.

[14] J. M. Hoekstra, R. C. J. Ruigrok, and R. N. H. W. van Gent, "Free flight in a crowded airspace?," in *Proceedings of the 3rd USA/Europe Air Traffic Management R&D Seminar*, (Naples, Italy), EUROCONTROL, 2000.

[15] M. G. Ballin, J. M. Hoekstra, D. J. Wing, and G. W. Lohr, "Nasa langley and nlr research of distributed air/ground traffic management," in *AIAA's Aircraft Technology, Integration, and Operations (ATIO) 2002 Technical Forum*, (Los Angeles, CA, USA), American Institute of Aeronautics and Astronautics, 2002.

[16] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[17] F. Bussink, J. Hoekstra, and B. Heesbeen, "Designing for safety: The 'free flight' air traffic management concept," *Cognition, Technology & Work*, vol. 2, no. 4, pp. 229–237, 2000.

[18] D. Alejo, R. Conde, J. Cobano, and A. Ollero, "Multi-UAV collision avoidance with separation assurance under uncertainties," in *2009 IEEE International Conference on Mechatronics*, pp. 1–6, 2009.

[19] S. Nik and M. Petovello, "Implementation of a dual-frequency glonass and gps l1 c/a software receiver," *THE JOURNAL OF NAVIGATION*, vol. 63, pp. 269–287, 04 2010.

[20] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, "Leveraging procedural generation to benchmark reinforcement learning," in *International conference on machine learning*, pp. 2048–2056, PMLR, 2020.

[21] C. Zhang, O. Vinyals, R. Munos, and S. Bengio, "A study on overfitting in deep reinforcement learning," *arXiv preprint arXiv:1804.06893*, 2018.

[22] S. Deniz, Y. Wu, Y. Shi, and Z. Wang, "A multi-agent reinforcement learning approach to traffic control at merging point of urban air mobility," in *AIAA AVIATION 2022 Forum*, p. 3912, 2022.

[23] S. Deniz, Y. Wu, Y. Shi, and Z. Wang, "A reinforcement learning approach to vehicle coordination for structured advanced air mobility," *Green Energy and Intelligent Transportation*, vol. 3, no. 2, p. 100157, 2024.