aerospace

MDPI

Article

Centralized Landing Flow Merging for Drones Using Deep Reinforcement Learning †

Sasha Vlaskin [1,2]*, Jan Groot [2], Emmanuel Sunil [1], Joost Ellerbroek [2], Jacco Hoekstra [2] and Dennis Nieuwenhuisen [1]

[1] Royal Netherlands Aerospace Centre (NLR), 1059 CM Amsterdam, The Netherlands
[2] Faculty of Aerospace Engineering, Operations and Environment, Delft University of Technology (TU Delft), 2628 CD Delft, The Netherlands
* Correspondence: sasha.vlaskin@nlr.nl
† Presented at the 15th EASN International Conference, in Madrid, Spain, 14–17 October 2025.

Abstract

Drones are expected to support applications such as emergency response, parcel delivery, and infrastructure monitoring in dense urban airspaces, creating traffic levels that are unmanageable for human operators. Autonomous separation management is therefore essential, combining strategic and tactical control to prevent conflicts. This paper addresses the tactical landing phase by introducing a centralized landing flow manager—a reinforcement learning (RL) agent that adjusts drone speed and heading to merge landing flows safely and efficiently prior to a final approach fix. The objective of the work was to demonstrate the potential of reinforcement learning in this novel context, by implementing and evaluating it in simulation and testing its capabilities with 10 concurrent landing drones. The RL agent learns to successfully separate traffic, thereby lowering intrusion counts compared to the baseline autopilot, but is outperformed in safety by the decentralized Modified Voltage Potential (MVP) method due to outlier scenarios. Nevertheless, the RL-based system achieves faster scenario completion and thus a higher overall throughput, by speeding up the vehicles towards the final approach fix. Future work will explore improved network architectures, transfer learning across varied scenarios, and algorithmic fine-tuning to further enhance safety performance.

Keywords: drones; reinforcement learning; BlueSky; UTM; flow merging; U-space

1. Introduction

Seven million drones in total are expected to operate within EU airspace by 2050 [1], of which five hundred thousand are expected to be commercial. Drone delivery is expected to be a large driver for this, with companies such as Amazon investing heavily in the sector. Nevertheless, this poses a great challenge in terms of separation—there will be too many drones for human operators to efficiently guide. For this, an autonomous separation management system is required, through which we can combine safety layers from strategic pre-flight sequencing to in-flight conflict resolution in order to safely manage drone traffic. This paper, more specifically, investigates the landing segment for these drones. Previous work [2] has shown that landing conflicts are a risk in constrained landing areas, where traffic converges. For this, an efficient algorithm is needed—if a landing schedule is breached and drones need to return safely to their point of origin, centrally controlled landing merging may become a necessity. That is, all drones give control to the landing

Academic Editor: Gustavo Alonso-Rodrigo

Received: 12 January 2026
Revised: 17 February 2026
Accepted: 28 February 2026
Published: 3 March 2026

management entity, which guides them to the vertiports safely with sufficient separation while having complete knowledge of the states of the incoming drones. A variety of related problems have been investigated using reinforcement learning [3–10], as the field has seen an ever-increasing amount of interest. Nevertheless, centralized merging of more than two vehicles simultaneously is still a difficult task for reinforcement learning agents. Much of the work approaches this from a multi-agent perspective [7], with a focus on exclusively pairwise conflicts and existing airspace corridors. For a vertipad in a high-density environment with inbound traffic, however, this may be difficult, as many drones' flight paths will converge to the same pad. Due to sensor errors and lack of intent knowledge of other vehicles due to sensor inaccuracies and limits, decentralized control could be difficult in this setting. Here, it may be beneficial for drones to be controlled centrally by a ground entity in order to avoid reactive conflicts. Another structural difference will be that the maneuvers are performed in a terminal maneuvering area, instead of adhering to specific corridors. This paper will outline the implementation of a relevant environment, and train a neural network model to perform the flow merging maneuvers in a centralized way using reinforcement learning.

Another objective of this paper is to compare the extent to which centralized and decentralized geometric landing flow control differ in their performance and actions. The centralized reinforcement learning agent is therefore compared to a decentralized approach, which simply uses the physics-inspired Modified Voltage Potential method. Here, drones self-separate on an individual basis, by using the shortest-out adjustment to the velocity vector. Therefore, this requires no training, which is another advantage over the RL-derived centralized agent. Two different learning algorithms are trialed for the centralized landing flow merging agent, and compared to the MVP method in terms of intrusions, behavior, and total maneuver execution time. Note that centralized methods are often at a disadvantage from a processing perspective, as they need to gather all incoming data, compute a central command, and send this to the vehicles to execute. This means that those methods are likely to be inherently slower as compared to a decentralized method, which is reflected in the action time. This work also does not aim to present a certifiable system—while work is ongoing in the domain to certify reinforcement learning methods in the ATM context [11], and to make them interpretable to human operators [12], the goal of this work is to identify a working application for centralized landing flow merging, implement it, and demonstrate its initial strengths and limitations as compared to decentralized methods.

2. Methodology

The goal of this paper is to implement a centralized landing flow merging agent—a ground agent that gives speed and heading commands to all drones of an incoming landing flow. Although theoretically a single-agent problem, it has added complexity when compared to traditional purely single-agent implementations. To develop this new environment, we must first find its closest peers. This scenario is similar to BlueSky-Gym's [13] MergeEnv-v0 in objective, but extends the observation vector significantly. BlueSky-Gym [13] is a reinforcement learning (RL) platform for the BlueSky ATC simulator [14], based on Gymnasium [15], and allows for quick prototyping of different RL algorithms on community scenarios. For this trial, using BlueSky-Gym [13], a 2D horizontal environment was built: *CentralizedMergeEnv-v0*.

2.1. Environment Logic and Randomization

The environment is set up to model a landing situation for drones. In a constrained environment and for high traffic densities, delays at landing need to be managed through

active arrival management. Arrival management consists of two components, namely slot adherence (timing) and geometric merging. This paper's goal is to investigate the applicability of reinforcement learning, firstly for the purely geometric component of the merging task. The main goal here is to evaluate the potential of an RL-based centralized landing flow merging agent within a terminal maneuvering area (TMA). In the test environment, drones are arriving from the right-hand side of the visualization. In order to land with sufficient spacing from each other at the same point, they must first reach a final approach fix (FAF). From that point on, the drones should travel in a line towards the pad. This is visualized in Figure 1.

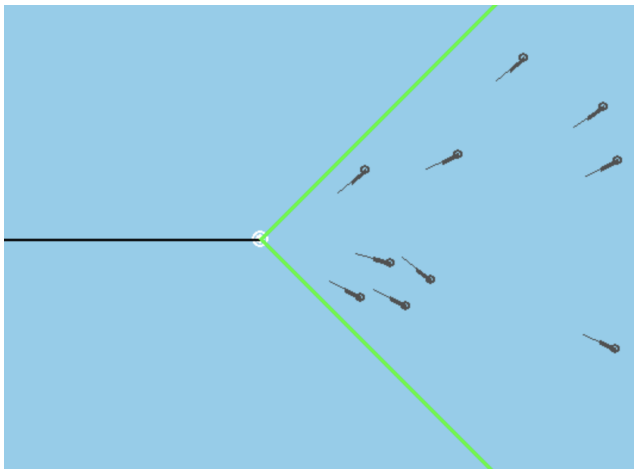


Figure 1. Render of the *CentralizedMergeEnv-v0* TMA environment.

Drones are shown as black rectangles with heading vectors indicating travel direction. The final approach fix (FAF) is a small white circle at the frame center. Green lines at $\pm 45°$ from the FAF mark the terminal maneuvering area (TMA) limits. The black line on the left shows the required path to the landing point.

Overfitting [16,17] is a prevalent problem in reinforcement learning research, and occurs when agents learn a specific action pattern instead of a general strategy. As this occurs in static scenarios, each training scenario used here is fully procedurally generated [18]: drones are initialized at positions determined by a randomized bearing (BEARING_TO_POS, $-45°$ to $45°$) and distance (DISTANCE_TO_POS, 5–20 NM) to the FAF. The model aims to merge drones while maintaining safe separation. Separation keeps drones outside each other's protected zones, circular areas into which no intruder may enter. An intrusion occurs when a protected zone is breached.

2.2. Reinforcement Learning Problem Formulation

2.2.1. Observation Vector

The base observation vector for this environment is given in Table 1. There are two sets of states: ownship (taken with respect to the current waypoint) and intruder, taken with respect to the other aircraft.

All of the ownship relative positions and velocities are taken with respect to the final approach fix, prior to it being reached. Once it is reached, the pad is used as the reference point with respect to which ownship states are calculated. For the intruders, information is encoded in relative terms to the ownship. The observation vector is visualized graphically in Figure 2. Here, the states are explained, with green denoting the FAF-relative ownship states and red the intruder-relative set of states. The angles are encoded in terms of sine and cosine to ensure normalization. As for the other values, coarse normalization is used to ensure stable learning.

Table 1. Observation vector for centralized flow merging agent.

	Variable	Description	Size
Ownship	$\cos(\phi_{dr})$	Cosine of the drift angle	*N_drones*
	$\sin(\phi_{dr})$	Sine of the drift angle	
	v	Airspeed	
	FAF_reach	FAF-reached Boolean vector	
	$\cos(\psi_{tr})$	Cosine of the track angle	
	$\sin(\psi_{tr})$	Sine of the track angle	
	d_{wpt}	Distance to current waypoint	
Intruder	d_{int}	Distance to intruder's vector	$N\_drones * N\_int$
	θ_{int}	Bearing to intruder's vector	
	$v_{x,int}$	X-component of velocity relative to intruder	
	$v_{y,int}$	Y-component of velocity relative to intruder	

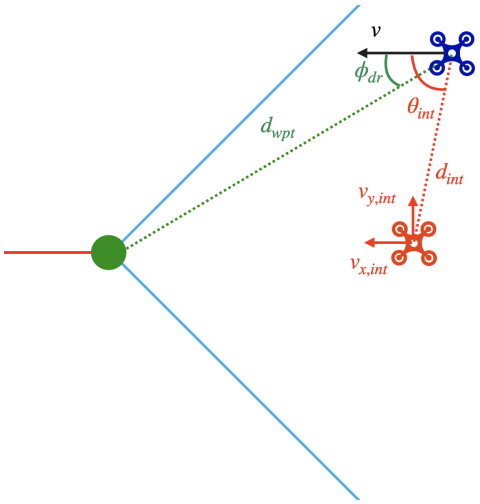


Figure 2. Observation vector visualization. The ownship is shown in blue and the intruder in red. The ownship-to-FAF states are summarized in green, and intruder relative states in red.

2.2.2. Action Space

The centralized agent is able to control the speed and heading of each of the drones. Thus, the action space in vector form has $2 * N_{drones}$ elements, 2 per drone. Note that each of the actions is bounded between -1 and 1 in the algorithm, but they are multiplied and translated to knots and degrees, respectively. The action space, and its bounds for every timestep, is shown in Table 2. The performance model used is that of the DJI Matrice M600, as used in the BlueSky simulator [14].

Table 2. Allowed actions and their bounds.

Action	Bounds
Heading change	$[-1, 1] \rightarrow [-25°, +25°]$
Speed change	$[-1, 1] \rightarrow [-0.5$ kts, $+0.5$ kts$]$

All of the vectoring maneuvers are deliberately constrained to the horizontal plane. The main reason for this is the 120 m AGL operational ceiling defined by EASA UAS rules [19], which leaves little maneuver room in the vertical direction. Delivery concepts also envision altitude layers, such as the outbound layers seen in [2], which would restrict maneuver room near vertiports further. The time between actions is taken to be 50 s (compared to MVP's 10 s) in this work. The reason for this is twofold—firstly, there is an inherent delay for a centralized system such as this, incurred by receiving the state information from the drones, computing the resolution maneuver, and transmitting the

resolution maneuver to the drones, which subsequently execute it. Decentralized methods do the processing onboard and forego the re-transmission. This longer time also helps in training, where a shorter update time would lead to significantly longer training runs.

2.2.3. Reward Function

The reward function is formulated as

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

The term R_{dr} takes the value of -0.1, and ϕ_{dr} is the drift, or the difference between the current heading and the heading line to the FAF (as seen in Figure 2). The intrusions are penalized with -1 per timestep spent in intrusion. The constants of the reward function were selected through manual tuning.

2.3. Independent Variables

The independent variables are as follows:

- Algorithm for separation assurance: RL agent (SAC or PPO), Modified Voltage Potential, or none.
- Protected zone radius (RPZ): 0.15 and 0.075 nautical miles.

2.4. Dependent Variables

Several metrics are used to quantify the effectiveness and safety of the resulting policies. First is the overall reward, as an indicator of learning. Then, the number of intrusion timesteps is counted—this is the prime indicator of safety. An intrusion, or loss of separation (LoS), is defined as occuring when a drone enters another drone's circular protected zone, that is, the value of the distance to the other drone is less than the protected zone radius R_{pz}. The drift is used as an efficiency measure. The time to execute a full scenario for a set of drones, or episode, is also recorded and is referred to as the episode length.

2.5. Reinforcement Learning Algorithms

In this paper, the Proximal Policy Optimization (PPO) [20] and the Soft Actor–Critic (SAC) [21] algorithms were used to train the RL model, as implemented in STABLE-BASELINES3 [22]. The PPO algorithm enhances stability when compared to earlier gradient methods. It is an on-policy algorithm, meaning that it learns only from the most recent interactions that its current policy has experienced from the environment. Such an algorithm can "forget" certain positive behaviors, as it discards past data. SAC is an off-policy algorithm, designed for continuous control tasks. Being off-policy, it has a replay buffer, hence storing both recent and past experiences in a memory. How it differs from other earlier continuous control algorithms is through its use of a maximum entropy reinforcement learning framework, which encourages exploration to a greater extent. Both algorithms, while resembling the Actor–Critic framework, are fundamentally different. The actor component of both networks is a stochastic policy, with the output being a Gaussian distribution for continuous action spaces. They differ mainly in that SAC uses entropy regularization to ensure exploration, and PPO clips updates in order to prevent them being too large. PPO's critic computes the value function $V(s)$, the expected return from state s, as seen in Equation (2).

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s\right] \quad (2)$$

The SAC algorithm, on the other hand, uses $Q(s,a)$, which is the expected return of taking action a, seen in Equation (3).

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^{t} r_{t} \mid s_{0} = s, a_{0} = a\right] \tag{3}$$

Here, r_t denotes the reward at time t, γ is the discount factor, π denotes the current policy, s the state, and a the action. The expectation of the total sum of the reward at a given policy is denoted by $\mathbb{E}_{\pi}$. The value function $V(s)$ is typically smoother and easier to estimate because it averages over actions, while the Q-function $Q(s,a)$ is essential for off-policy methods, where state information is needed for learning from past experience (stored in the replay buffer). It is also worth noting that due to the replay buffer implementation, the SAC algorithm is more sample-efficient than PPO, as it learns from a random sample in its replay buffer, therefore requiring less fresh data. The temporal correlation between different parts of the data is also broken in this fashion—the model is then less likely to learn patterns that are too dependent on this specific sequence, rather resulting in a more general solution. Fundamentally, the single-agent implementation of this landing flow merging agent (where only one aircraft is steered into an existing traffic stream) showed that SAC outperformed PPO by a significant margin in terms of reward [13]. Therefore, the same is expected here in terms of result. The goal of this work is to compare an RL implementation to a classical decentralized iterative geometric approach. For this reason, no extensive hyperparameter tuning or reimplementation of the algorithms at hand is performed. The hyperparameters are kept as standard for STABLE-BASELINES3 [22], and are shown in Table 3.

Table 3. Hyperparameters for SAC and PPO, as used in STABLE-BASELINES3 [22].

Parameter	PPO	SAC
learning_rate	3×10^{-4}	3×10^{-4}
gamma	0.99	0.99
batch_size	64	256
ent_coef	0.0	auto
max_grad_norm	0.5	–

2.6. Decentralized Approach: Modified Voltage Potential

The Modified Voltage Potential method, as developed for the NLR-NASA Free-Flight study [23], is a state-based decentralized conflict resolution (CR) method, which is used as a representative method to compare the centralized agent to. Since this is a purely geometric task and scheduling is not taken into account, it is also plausible that agents may be allowed to self-separate. The method uses the fastest-out solution, which is geometrically depicted in Figure 3.

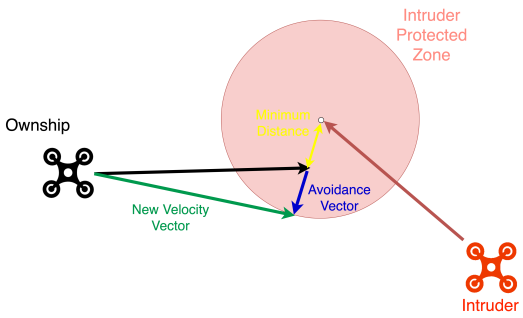


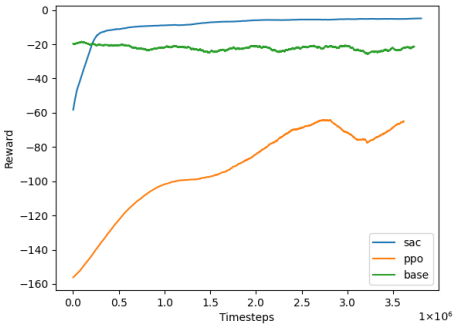
Figure 3. MVP resolution, shown graphically (adapted from [23]).

While other decentralized methods such as Velocity Obstacle [24] can also perform well, studies [25] have consistently demonstrated that MVP is still the best-suited method for decentralized self-separation.

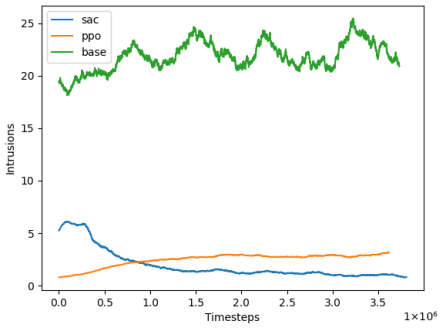
3. Results and Discussion

3.1. Reward Evolution (4M Timesteps)

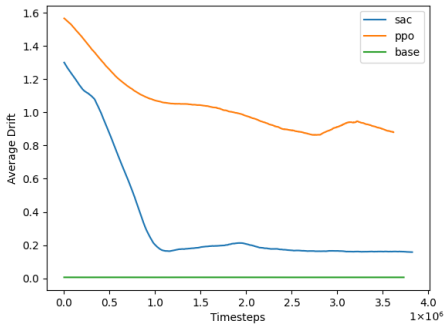
The reward evolution, as well as its individual components, are plotted as learning curves in Figure 4. It is clear that SAC achieves asymptotic performance, as evidenced by its total reward, drift, and intrusion plots showing no change. PPO shows no such stability in policy, with the policy not yet having converged. The green curve is introduced as a reference, and shows the result of not applying any conflict resolution method to the flights at all. Here, the drift is zero and the reward does not fluctuate significantly. Note that this initial baseline condition of doing nothing leads to a minimum average of 20 timesteps in loss of separation per episode, and no drift, as the straight-line path is maintained.



(**a**) Reward evolution (4M timesteps)



(**b**) Intrusion evolution (4M timesteps)



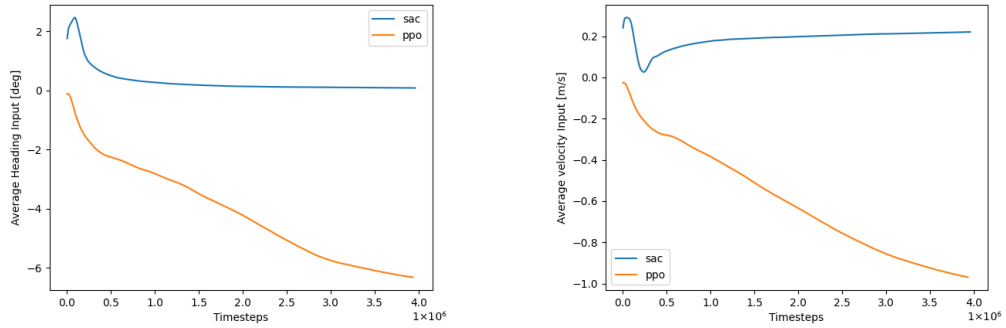
(**c**) Drift evolution (4M timesteps)

Figure 4. Evolution of key performance metrics over 4 million training timesteps: (**a**) reward, (**b**) intrusions, and (**c**) drift.

3.2. Actions and Their Evolution

The average actions performed by the agent across all drones in terms of velocity and heading were also analyzed to see their overall effect. The outputs for SAC and PPO are plotted in Figure 5.

What can be noted from Figure 5 is that SAC achieves a stable policy, whereas PPO fails to learn a policy that is consistent with this task. Note also that the heading input for SAC is consistently positive, whereas PPO learns to turn in the opposite direction. In terms of speed input, Figure 5b in fact shows the SAC-derived model preferring to speed the vehicles up on average.

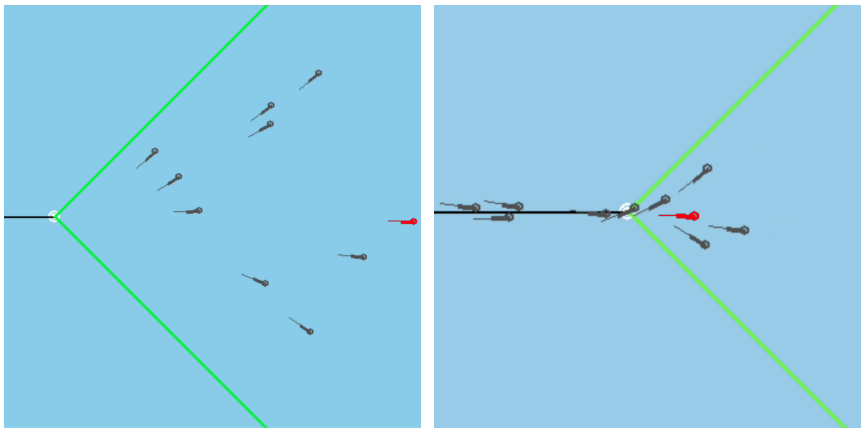


(**a**) Heading input evolution (**b**) Velocity input evolution

Figure 5. Evolution of (**a**) heading and (**b**) velocity inputs during training for PPO and SAC.

3.3. Policy Analysis

The resulting policy has several interesting behaviors, which will be discussed here. First, the policy allows for drones overtaking each other. This is visualized in Figure 6.



(**a**) Initial situation (**b**) After overtake (zoomed in)

Figure 6. Overtake situation render—tracking the drone in red. Other drones are shown in black, the FAF is the white circle, the green lines depict TMA limits. The left image depicts starting conditions, with the tracked drone being the furthest from the FAF. The right image shows the end of the scenario, demonstrating the overtake performed by the red drone.

More concretely, the change in vehicle order due to these overtaking maneuvers can be summarized in terms of mean square error from the start indices to the final order such that

$$MSE_{order} = \frac{(index\_list_{init} - index\_list_{final})^2}{N_{DRONES}} \tag{4}$$

where ind_list_{init} is the list of drone indices sorted in order of increasing distance to the waypoint at the start of the episode, and ind_list_{final} that at the end of the episode. Plotted for 1000 episodes for SAC, PPO, and the MVP baseline, for 10 drones and a protected zone of 0.075 nautical miles, we obtain Figure 7.

This shows that the order does change, but that the extent of the change is consistent with what would occur in an emergent way for a decentralized method. Since the initialization conditions are consistent, as are the initial speeds of the drones, it is likely that the algorithm learns to utilize space in the same way that would emerge from using MVP.

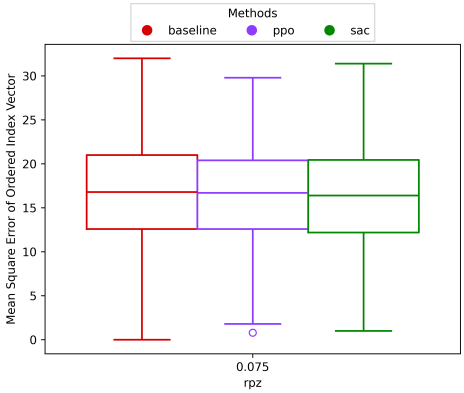
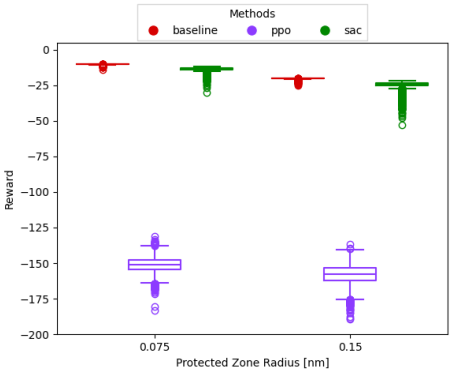


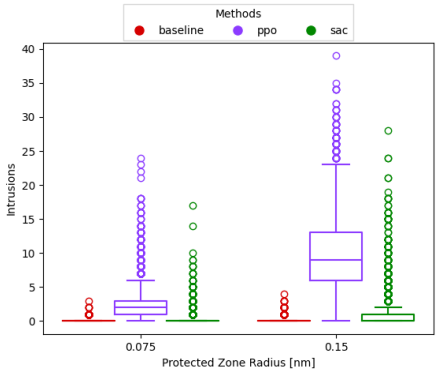
Figure 7. Average mean square error of change in drone arrival order.

3.4. Sensitivity Analysis—Protected Zone

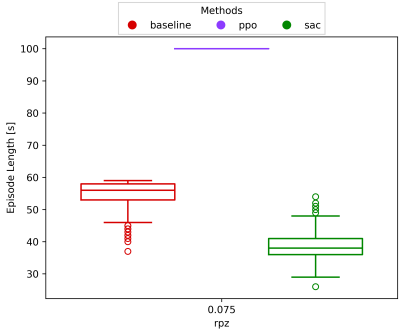
Intrusion rates depend strongly on the protected zone (PZ) size. Training used a 0.15 NM (278 m) PZ, while testing employed 0.075 NM. It was expected, based on other studies [26], that training on a higher protected zone radius would yield better performance on the intended operational PZ value. This is indeed consistent with the findings here, as Figure 8 shows. Across 10,000 episodes (Figure 8), both MVP and SAC achieved a mean of zero intrusions at 0.075 NM, with SAC showing 0.015% more outliers but 25% shorter episode durations. This means that the RL method completes the maneuvers significantly faster than the decentralized baseline, at the cost of safety.



(**a**) Reward vs. PZ



(**b**) Intrusions vs. PZ



(**c**) Episode duration

Figure 8. Sensitivity of model performance to protected zone radius: (**a**) total reward, (**b**) mean intrusions, and (**c**) episode duration.

3.5. Discussion

The learning outcomes, in terms of successful navigation to the FAF, reduction in intrusions/LoS, and visualized policy all demonstrate that a centralized RL agent imple-

mentation for merging landing flows is feasible by using SAC, and that it improves on the "do nothing" policy. Interestingly, PPO is unable to reach the same results, and although it demonstrates learning, it fails to learn to avoid conflicts entirely, and adopts a holding pattern instead of guiding drones to the final approach fix. This behavior may be due to the lack of a replay buffer for this algorithm; however, it is unclear whether additional training alone will help overcome these issues. These findings are consistent with other testing between the SAC and PPO algorithms, where SAC seems to show a performance edge [13] under the default hyperparameters. Further investigation is needed to see by how much this can be improved.

For the main question posed in the research, the comparison to the baseline Modified Voltage Potential method, the SAC-trained model performs similarly to MVP in terms of safety when the mean is considered. There are, however, some clear additional outliers outside of this mean value, accounting for 0.015% more intrusions. A metric in which the SAC-derived model is able to outperform the MVP method is overall execution time—the drones are guided to the pad quicker, as evidenced by the shorter average episode time. Despite this, MVP is still the marginally safer method to use. An important factor to consider is the update time for both methods—the reinforcement learning methods are only allowed to execute every 50 s, whereas MVP is able to execute every 10 s. The reason for this longer update time is twofold: it allows for faster scenario computation while providing a first-order delay model as would be expected for a centralized method. It is hypothesized that this may be in part what causes the outlier losses of separation, but further verification is needed. These outliers could also very well be a result of the dual nature of the reward function, which incorporates both efficiency and safety, which may lead to some safety tradeoffs by the model in favor of efficiency. Future work will investigate this Pareto front of safety/efficiency ratios, and find whether a less efficient but safer solution is possible.

It is also notable that training on a 0.15-nautical mile PZ and using a separation minimum of 0.075 subsequently closes the performance gap between the methods, as seen in the intrusion plot (b) in Figure 8. Likely, longer training and a model with greater breadth could close the gap in performance further. For generalizability, the paper utilizes randomized drone positions and velocities at the initialization of every episode, which guarantees generalization over the sample space that the environment offers. Due to the limitations of the model architecture (fully connected neural network), the method is inflexible to changes in drone number. Thus, testing across different drone numbers was not feasible at this time. Future work will utilize the framework developed and showcased for this paper in conjunction with the attention mechanism [27], which would allow for better scalability and testing across different vehicle numbers. For real-world generalizability, recent work [28] on conflict resolution with reinforcement learning shows that the gap is difficult to predict, and that training on lower-fidelity scenarios can help improve performance in those closer to the real world. In addition, sensor uncertainty appears to play an important part in the performance of conflict resolution algorithms [29], where learning-based algorithms appear to exhibit an edge in safety over geometric algorithms. It is recommended that future studies consider pre-training on lower-fidelity scenarios, train across a larger variety of conflict geometries, and include sensor noise.

4. Conclusions and Recommendations

This paper showcases the successful implementation of a learning-based centralized flow merging agent. It is seen that the Soft Actor–Critic-derived model is able to avoid intrusions when compared to the base autopilot while successfully steering traffic to the final approach fix. While faster at performing the merge maneuver, as evidenced by the episode duration plots, the RL agent shows additional outlier intrusion episodes when

compared to the MVP baseline, which is strongly undesirable. The policy employed is similar to that observed in the MVP scenarios, in that the centralized agent in large part resorts to speed commands—the main difference here lies in its tendency to speed vehicles up instead of slow them down. In terms of further work, it would be beneficial to investigate whether a larger model or extensive hyperparameter tuning could yield better overall performance. Furthermore, priority rules for vehicles need to be implemented, along with simulated anomalies (such as weather) that could cause go-arounds. Training the model on more varied geometry, such as a generic conflict resolution task within a sector, is also worth exploring, as it may improve safety further. Sensor noise and lower-fidelity pre-training are also seen as important avenues for the improvement of this specific method. Crucially, more work needs to be done on explainability in order to make the model certifiable in the future. Finally, utilizing different model architectures and mechanisms for the actor network could also help improve safety performance. An attention mechanism is recommended in place of the existing fully connected network, as it could likely provide improvement in both scalability and safety performance—this will be the main focus for future work.

Author Contributions: Conceptualization, S.V. and J.G.; methodology, S.V. and J.G.; software, J.G. and S.V.; validation, S.V., E.S. and J.E.; formal analysis, S.V.; investigation, S.V. and E.S.; resources, D.N.; data curation, S.V.; writing—original draft preparation, S.V.; writing—review and editing, S.V., E.S. and J.E.; visualization, S.V.; supervision, E.S., J.E., J.H. and D.N.; project administration, S.V., J.E. and J.H.; funding acquisition, D.N. and E.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: All source code is available at https://github.com/svlaskin/bluesky-gym-sasha.git (accessed on 11 January 2026), and is forked from https://github.com/TUDelft-CNS-ATM/bluesky-gym.git (accessed on 11 January 2026).

Acknowledgments: The authors acknowledge the use of GenAI (ChatGPT GPT-4o) in this work, limited to the spell-checking and rephrasing of some of the text in order to improve legibility.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MVP	Modified Voltage Potential
SAC	Soft Actor–Critic
PPO	Proximal Policy Optimization
RL	Reinforcement learning
LoS	Loss of separation
FAF	Final approach fix
TMA	Terminal maneuvering area

References

1. SESAR Joint Undertaking. *European Drones Outlook Study: Unlocking the Value for Europe*; SESAR Joint Undertaking: Bruxelles, Belgium, 2017.
2. Vlaskin, S.; Sunil, E.; Nieuwenhuisen, D.; Ellerbroek, J.; Hoekstra, J. Comparison of Strategic Conflict Prevention Methods for Departure Planning in Drone Delivery. In *2024 AIAA DATC/IEEE 43rd Digital Avionics Systems Conference (DASC)*; IEEE: Piscataway, NJ, USA, 2024; pp. 1–8. [CrossRef]
3. Razzaghi, P.; Tabrizian, A.; Guo, W.; Chen, S.; Taye, A.; Thompson, E.; Bregeon, A.; Baheri, A.; Wei, P. A survey on reinforcement learning in aviation applications. *Eng. Appl. Artif. Intell.* **2024**, *136*, 108911. [CrossRef]

4. Wang, P.; Chan, C.Y. Formulation of deep reinforcement learning architecture toward autonomous driving for on-ramp merge. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*; IEEE: Piscataway, NJ, USA, 2017. [CrossRef]
5. Deniz, S.; Wu, Y.; Shi, Y.; Wang, Z. A Multi-Agent Reinforcement Learning Approach to Traffic Control at Merging Point of Urban Air Mobility. In *AIAA AVIATION 2022 Forum*; American Institute of Aeronautics and Astronautics: Reston, VA, USA, 2022; p. 3912.
6. Deniz, S.; Wu, Y.; Shi, Y.; Wang, Z. A reinforcement learning approach to vehicle coordination for structured advanced air mobility. *Green Energy Intell. Transp.* **2024**, *3*, 100157. [CrossRef]
7. Brittain, M.; Wei, P. Autonomous air traffic controller: A deep multi-agent reinforcement learning approach. *arXiv* **2019**, arXiv:1905.01303. [CrossRef]
8. Schlichting, M.R.; Notter, S.; Fichter, W. Long Short-Term Memory for Spatial Encoding in Multi-Agent Path Planning. *J. Guid. Control Dyn.* **2022**, *45*, 952–961. [CrossRef]
9. Notter, S.; Gall, C.; Müller, G.; Ahmad, A.; Fichter, W. Deep Reinforcement Learning Approach for Integrated Updraft Mapping and Exploitation. *J. Guid. Control Dyn.* **2023**, *46*, 1997–2004. [CrossRef]
10. Isufaj, R.; Omeri, M.; Piera, M.A. Multi-UAV conflict resolution with graph convolutional reinforcement learning. *Appl. Sci.* **2022**, *12*, 610. [CrossRef]
11. Ribeiro, M.J.; Tseremoglou, I.; Santos, B.F. Certification of Reinforcement Learning Applications for Air Transport Operations Based on Criticality and Autonomy. In *AIAA SciTech 2024 Forum*; AIAA 2024-1463; American Institute of Aeronautics and Astronautics (AIAA): Reston, VA, USA, 2024. [CrossRef]
12. Nunes, T.; Borst, C.; van Kampen, E.J.; Hilburn, B.; Westin, C. Human-interpretable input for Machine Learning in tactical air traffic control. In Proceedings of the SESAR Innovation Days 2021, Online, 7–9 December 2021.
13. Groot, D.; Leto, G.; Vlaskin, A.; Moec, A.; Ellerbroek, J. BlueSky-Gym: Reinforcement Learning Environments for Air Traffic Applications. In Proceedings of the SESAR Innovation Days 2024, Rome, Italy, 12–15 November 2024.
14. Hoekstra, J.M.; Ellerbroek, J. Bluesky ATC simulator project: An open data and open source approach. In *Proceedings of the 7th International Conference on Research in Air Transportation*; FAA/Eurocontrol USA/Europe: Washington, DC, USA, 2016; Volume 131, p. 132.
15. Towers, M.; Kwiatkowski, A.; Terry, J.; Balis, J.U.; Cola, G.D.; Deleu, T.; Goulão, M.; Kallinteris, A.; Krimmel, M.; KG, A.; et al. Gymnasium: A Standard Interface for Reinforcement Learning Environments. *arXiv* 2024, arXiv:2407.17032. [CrossRef]
16. Zhang, C.; Vinyals, O.; Munos, R.; Bengio, S. A study on overfitting in deep reinforcement learning. *arXiv* **2018**, arXiv:1804.06893. [CrossRef]
17. Zhang, A.; Ballas, N.; Pineau, J. A Dissection of Overfitting and Generalization in Continuous Reinforcement Learning. *arXiv* **2018**, arXiv:1806.07937. [CrossRef]
18. Cobbe, K.; Hesse, C.; Hilton, J.; Schulman, J. Leveraging procedural generation to benchmark reinforcement learning. In *International Conference on Machine Learning*; PMLR: New York, NY, USA, 2020; pp. 2048–2056.
19. European Union Aviation Safety Agency (EASA). *Easy Access Rules for Unmanned Aircraft Systems*; Technical Report; European Union Aviation Safety Agency: Cologne, Germany, 2021. Available online: https://www.easa.europa.eu/en/document-library/easy-access-rules/online-publications/easy-access-rules-unmanned-aircraft-systems?page=4 (accessed on 11 February 2026).
20. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347. [CrossRef]
21. Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. Soft actor-critic algorithms and applications. *arXiv* **2018**, arXiv:1812.05905.
22. Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M.; Dormann, N. Stable-baselines3: Reliable reinforcement learning implementations. *J. Mach. Learn. Res.* **2021**, *22*, 1–8.
23. Hoekstra, J.M.; Ruigrok, R.C.J.; van Gent, R.N.H.W. Free Flight in a Crowded Airspace? In Proceedings of the 3rd USA/Europe Air Traffic Management R&D Seminar, Naples, Italy, 3–6 June 2000.
24. Fiorini, P.; Shiller, Z. Motion Planning in Dynamic Environments Using Velocity Obstacles. *Int. J. Robot. Res.* **1998**, *17*, 760–772. [CrossRef]
25. Balasooriyan, S. Multi-Aircraft Conflict Resolution Using Velocity Obstacles. Master's Thesis, Delft University of Technology, Delft, The Netherlands, 2017.
26. Groot, J.; Ellerbroek, J.; Hoekstra, J. Analysis of the impact of traffic density on training of reinforcement learning based conflict resolution methods for drones. *Eng. Appl. Artif. Intell.* **2024**, *133*, 108066. [CrossRef]
27. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. *arXiv* **2023**, arXiv:1706.03762. [PubMed]

28. Moec, A.; Groot, D.J.; Ellerbroek, J. Mixed-Fidelity Reinforcement Learning for Aircraft Conflict-Resolution. In Proceedings of the 15th SESAR Innovation Days 2025, Bled, Slovenia, 1–4 December 2025; pp. 1–9. Available online: https://sesar.eu/sites/default/files/documents/sid/2025/papers/SIDs_2025_paper_17-final.pdf (accessed on 11 February 2026).
29. Vlaskin, S.; Rahman, M.F.; Sunil, E.; Ellerbroek, J.; Hoekstra, J.; Nieuwenhuisen, D. Deep Reinforcement Learning for Drone Conflict Resolution Under Uncertainty. In Proceedings of the SESAR Innovation Days 2025, Bled, Slovenia, 1–4 December 2025. Available online: https://www.sesarju.eu/sites/default/files/documents/sid/2025/papers/SIDs_2025_paper_72-final.pdf (accessed on 11 February 2026).