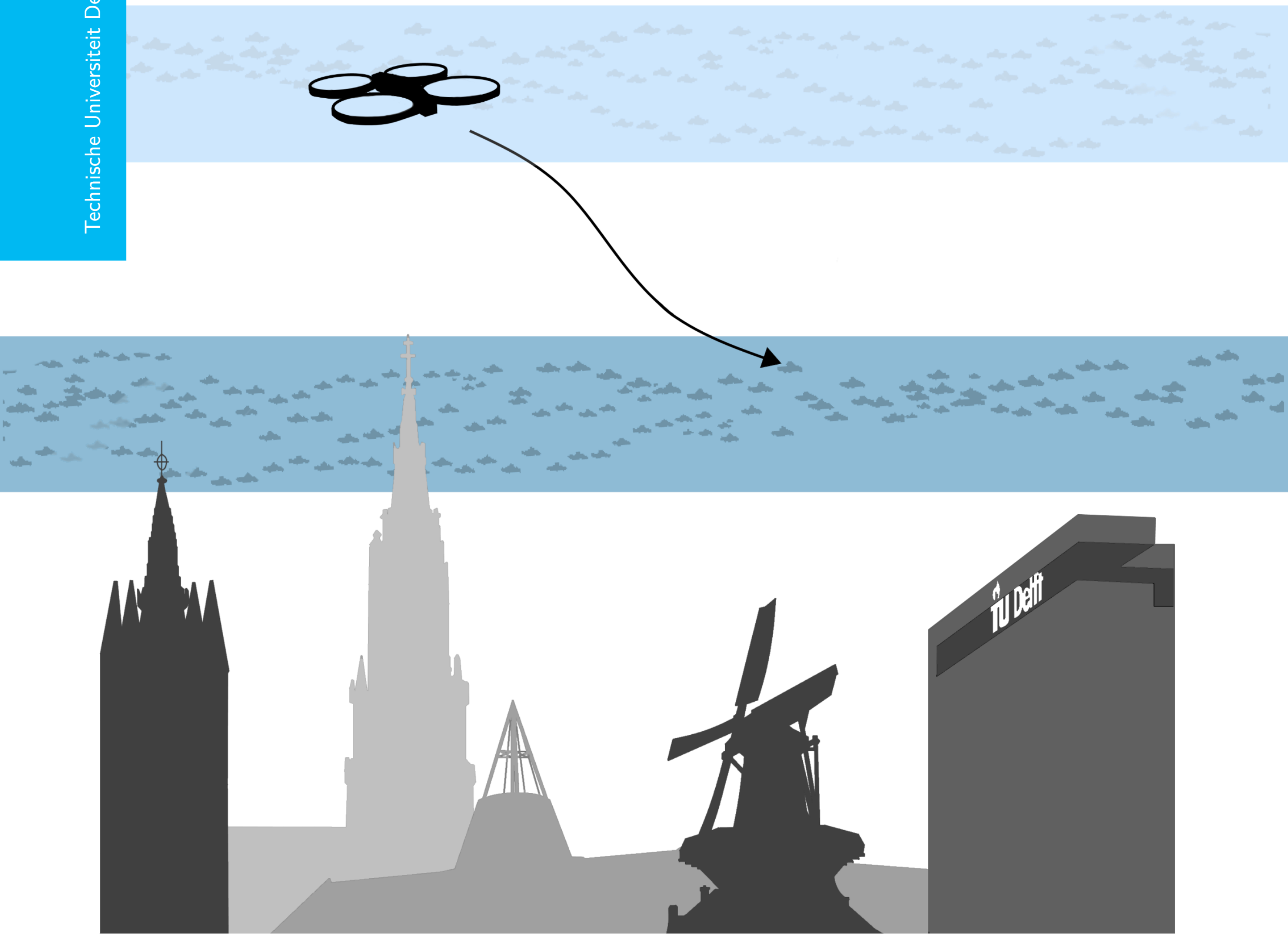
Improving Safety of Vertical Manoeuvres in a Layered Airspace with Deep Reinforcement Learning

MSc Thesis

Jan Groot
Supervision:
Marta Ribeiro, Joost Ellerbroek, Jacco Hoekstra



Improving Safety of Vertical Manoeuvres in a Layered Airspace with Deep Reinforcement Learning

by

Jan Groot

To obtain the degree of Master of Science
at the Delft University of Technology

Aerospace Engineering, Control & Simulation Department

Student number	4353583
Supervision	M.J. Ribeiro
	J. Ellerbroek
	J.M. Hoekstra

Acknowledgements

First of all, I would like to thank Jacco, Joost and Marta for giving me the opportunity to work on this incredible project with such freedom. I love how you all were always there for me when I had questions or to give advice when I got stuck, but never frowned upon my ideas and gave me the freedom to work them out myself. I would also like to thank Marta for the amount of time and thought you put into all of our meetings and being available almost every moment in time (I still don't understand how you can answer all my questions so quickly, even if I asked them at 11 p.m. or during the weekend), a special thanks also to the rate at which you were able to provide feedback at my work, as that really allowed me to keep my pace. I want to thank Joost for your great constructive criticism, critical questions and your ability to immediately pickup on any subject, even if it was a long time ago since our last meeting, without you the final thesis would not be at the level that it is now. I was really fortunate to have you as my supervisors. Finally I would like to thank everyone who I could come to when I was stuck with either motivation problems, program errors or claustrophobia from sitting in the same room for 10 months, be it friends, family or my lovely girlfriend Lemin, without all of you I don't think I would have managed to finish this Thesis.

Contents

Part I

Scientific Paper

Improving Safety of Vertical Manoeuvres in a Layered Airspace with Deep Reinforcement Learning

Jan Groot
Control and Simulation, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands
Supervision: Marta Ribeiro, Joost Ellerbroek and Jacco Hoekstra

***Abstract*—Current estimates show that the presence of unmanned aviation is likely to grow exponentially over the course of the next decades. Even with the more conservative estimates, these expected high traffic densities require a re-evaluation of the airspace structure to ensure safe and efficient operations. One structure that scored high on both the safety and efficiency metrics, as defined by the Metropolis project, is a layered airspace, where aircraft with an intended heading are assigned to a specific altitude layer. However, a problem arises once aircraft start to vertically traverse between these layers, leading to a large number of conflicts and intrusions. One way to potentially reduce the number of intrusions during these operations is by using conventional conflict resolution algorithms. These algorithms however have also been shown to lead to instabilities at higher traffic densities. As recent years have shown tremendous growth in the capabilities of Deep Reinforcement Learning, it is interesting to see how well these methods perform in the field of conflict resolution. This research investigates and compares the performance of multiple Soft Actor Critic models with the Modified Voltage Potential algorithm during vertical manoeuvres in a layered airspace. The final obtained performance of the trained models is comparable to that of the Modified Voltage Potential algorithm and in certain scenarios, the trained models even outperform the MVP algorithm. Overall, the results show that DRL can improve upon the current state of conflict resolution algorithms and provide new insight into the development of safe operations.**

***Keywords*—Conflict Detection and Resolution (CD&R), Deep Reinforcement Learning (DRL), Modified Voltage Potential (MVP), Unmanned Traffic Management (UTM), Self-Separation, BlueSky ATC Simulator**

I. Introduction

With the current rise in market demands for faster parcel delivery combined with the incentive to reduce the cost of such a delivery service, more and more companies have started researching the viability of using drones for these so-called last mile delivery operations [1], [2]. Estimates for the drone delivery market in Paris range widely from 110.000 to 275.000 drones operating per hour in the city by 2035 [3]. Even at the lowest estimates, this far surpasses the traffic densities of the current aviation standards. Therefore the Federal Aviation Administration (FAA) and the International Civil Aviation Organisation (ICAO) have required drones to be capable of detect and avoid manoeuvres without the need of a human controller [4].

Constantly requiring avoid manoeuvres, which likely move aircraft away from their nominal path, is inefficient and can lead to instabilities. Therefore, it is worthwhile to research the effect of airspace structures on the intrinsic safety of the airspace. The Metropolis project researched a variety of different structures differing in complexity and restrictions. This project showed that a layered airspace, separating traffic vertically based on their heading, leads to a high intrinsic safety without (heavily) impacting the efficiency of the air traffic operations [5]. This increase in safety can be attributed to the separation and alignment effect [6]. The problem that arises, however, is that vertically manoeuvring aircraft do not benefit from this separation and alignment effect. This results in these vertical operations leading to a large increase in conflicts and intrusions, which is also demonstrated in different studies [7], [8].

One potential solution to this rise in conflicts would be to use conventional conflict resolution algorithms. However, at higher traffic densities, these algorithms might potentially lead to instabilities [9]. Conflict resolution at high traffic densities essentially is a multi-agent coordination problem. Deep Reinforcement Learning (DRL) has been shown to successfully learn how to operate in these environments by adapting to emergent behaviour that follows from these continuous interactions. Furthermore, studies have also shown that DRL can be used for lane changing and merging of cars in highway scenarios, which can be considered a 2D version of the layer transition problem [10]. Because of this, this research will investigate the capabilities of DRL for improving the safety of vertical manoeuvres in a layered airspace structure through direct control of the drones.

The DRL models will be trained for a variety of degrees of freedom in large scale simulations, simulating both package deliveries and take-offs. The performance of the final converged models will then be compared to the Modified Voltage Potential (MVP) conflict resolution algorithm to see if there is a benefit to using DRL over conventional methods. It is decided to use the MVP algorithm as previous research has shown that it is optimal at resolving conflicts whilst minimizing additional travel distance [11].

II. Problem Formulation

The anticipated increase in numbers of drones and personal aircraft in the airspace has caused renewed interest in the field of airspace structuring to accommodate these traffic densities. During the Metropolis project, it was found that a layered airspace, where aircraft are assigned to a specific, altitude bound, layer based on their heading, improved on the intrinsic safety of the airspace, without severely impacting the efficiency of the total operations [5]. The downside of such a layered system however is that heading changes will be coupled with altitude changes. It was found that these vertical manoeuvres negatively influence the safety of the airspace and, in certain airspace configurations, are even responsible for the majority of the intrusions [7], [8]. Improving the safety of these vertical manoeuvres means that the benefits of a layered airspace can be kept without the downside of vertical conflicts.

It is decided that Deep Reinforcement Learning (DRL) will be used as a novel approach for safe manoeuvring of aircraft due to the recent advancements in this field as well as because it was used to successfully teach an agent safe lane-changing behaviour for autonomous driving, a task that can be seen as a 2D version of the layer changing problem[10]. To analyse the results of the DRL model, the performance will be compared to the performance of the Modified Voltage Potential (MVP) algorithm, a conflict resolution algorithm based on the repelling force of charged particles [12].

To research the effectiveness of DRL and the MVP algorithm, these vertical operations will be simulated in the BlueSky Open Air Traffic Simulator [13]. In this simulation environment, drones will be tasked with either climb or descent commands to a specific target layer within this airspace. During these vertical manoeuvres, the goal of the model is to safely control the drone to the target layer while avoiding the other aircraft. From now on, individually controlled aircraft will be referred to as agents, whereas the model is used to define which policy is used by these individual agents.

The margin by which other aircraft must be avoided is based on a minimum horizontal and vertical separation, if any two aircraft are within these margins of each other, the aircraft are said to have an intrusion. In this research a conflict between two aircraft means that the distance at the predicted closest point of approach between these aircraft is smaller than the required separation margins, indicating a potential future intrusion.

How these conflicts are resolved depends on which model the agent that is used to control the individual aircraft is based. For the MVP models, when aircraft are in conflict, the shortest way out of the conflict is determined and translated to a set of actions that should be taken by the aircraft. The DRL models on the other hand are allowed to select actions for the aircraft each time-step of the simulation, independent on whether or not the aircraft is in conflict.

For this research the DRL and MVP models are further subdivided into different models based on the freedom they have in their actions, as it is currently unknown which set of actions will result in the optimal performance. In total three individual actions can be isolated: a change in vertical velocity, a change in horizontal velocity and a change in heading. These actions are combined to obtain the following 'sub-models':

- 'vert', control of the vertical velocity only
- 'vert+', control of the vertical and horizontal velocity
- 'hor', control of the horizontal velocity and heading
- 'full', or 3 degrees of freedom, control of all motions

III. Methods

In this section, the methods used for the experiments will be presented. First, the Markov Decision Processes (MDPs) are formulated to allow usage DRL methods. Then the used DRL algorithm, Soft Actor Critic, will be further elaborated. Finally, a quick overview of the used resolution baseline, MVP, will be given.

A. Markov Decision Process

To ensure that DRL can be used for the defined problem, this problem must first be formulated as an MDP. An MDP is a mathematical framework that can be used for decision making in systems with uncertainty. An important element of the MDP is the so-called Markov-property, which entails that the future states of the system should only be dependent on the current state of the system. For the scenario of conflict resolution with MVP this Markov-property holds, as for a specific conflict, the used resolution manoeuvre, and therefore future states, are independent of how these aircraft came to be in conflict. It is therefore assumed that this property also holds for DRL. This allows the problem to be formulated as an MDP, described by the quadruple (S, A, P, R): [14]

- S, the state space of the system. Contains information on the ownship as well as other aircraft in the airspace.
- A, the action space of the system. Gives the allowed actions to select for a given state. Is dependent on both the model, state and actions bounds.
- $P([s,a],s')$, the state transition function. Defines the probability of a state-action pair to transition to a specific new state. Which in this research is fully described by the dynamics implemented in BlueSky.
- $R(s,a,s')$, the reward function. Rewards (or penalizes) the agent depending on the state transition and selected action.

The goal of the model is to learn which action $a \in A$ given a state $s \in S$ maximizes the total reward $\sum r \in R$ over all the state transitions.

1) State: The state vector is a combination of the ownship states and the (relative) states of the intruders. As the number of aircraft in the state-space is variable, and the state representation has to be constant in size for the proposed method, the problem has to be converted to a partial observable MDP (POMDP). For this research, it is decided to use 5 aircraft in the state representation sorted by time until closest point of approach (T_{cpa}) with a maximum distance at closest point of approach (D_{cpa}) of 250m, which is 5 times the minimum horizontal separation (PZ_h) between 2 aircraft. This removes aircraft which are moving away from each other from the state and only includes the aircraft with the smallest T_{cpa} in the state, e.g, the aircraft that require the most imminent action. An exception to this is made for aircraft that are in conflict, these are prioritized over other aircraft and always included in the state, again sorted by T_{cpa}. All the horizontal states considered for state inclusion are given in Figure 1. CPA stands for closest point of approach, which is the point at minimum horizontal distance. Beyond this also the vertical distance, D_z, and relative vertical velocity, V_w, are considered.

For the ownship state, the height difference with the target layer (D_h), vertical velocity (V_z), horizontal velocity (V_h) and heading (hdg) are used for the state vector. The final state vectors for all the models are given in table I. Not all models are given the same state vector as it is assumed that a too-large state vector containing non-relevant information will negatively impact the required training time of the models.

Finally, all states are normalized with equation 1 before usage, which makes the distribution of all states have a zero mean and unit variance. This ensures that the initial weights for all states are of similar magnitude. In this equation, μ_s refers to the mean value of this state and σ_s to the standard deviation. The values for σ_s and μ_s are determined by observing 100.000 state transitions. An exception for the normalization of the state vector is made for the conflict boolean parameter, which is kept as a boolean.

$$S = \frac{s_i - \mu_s}{\sigma_s} \tag{1}$$

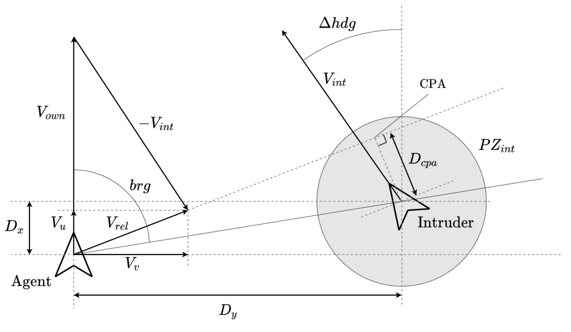


Figure 1. Visualization of the (horizontal) states related to an intruder.

TABLE I. The resulting state vector for the different experiments.

	vert	vert+	hor	full
Ownship States				
V_z	x	x	x	x
V_{own}		x	x	x
hdg			x	x
D_h	x	x	x	x
Intruder States (x5) ↓				
T_{cpa}	x	x	x	x
D_{cpa}	x	x	x	x
Conflict with ownship (boolean)	x	x	x	x
D_z	x	x	x	x
D_x		x	x	x
D_y		x	x	x
V_u		x	x	x
V_v			x	x
brg			x	x
Δhdg			x	x

2) Action Space: For the action space, the allowable actions and their limits have to be defined. The allowable actions are dependent on the different models, defined in section II. The limits for the different actions are given in table II. In this table the increment column indicates the maximum change in action per time-step of the simulation.

TABLE II. Allowed range and increments per time-step for each of the different actions.

Action	Range	Increment
Vertical Velocity (m/s)	[0, 5]	[0, 5]
Horizontal Velocity (m/s)	[5, 15]	[-1.5, 1.5]
Heading (deg)	[0, 360]	[-45, 45]

3) Reward: It is preferred to keep the reward function as simple as possible while encompassing all the requirements of the solution to the problem. [15] This philosophy leads to the reward function given in equation 2. Here s_{target} refers to a state in which the agent is in the corresponding target layer and s_{LoS} is a state in which a Loss of minimum Separation with the agent is present.

$$r = \begin{cases} 1 & \text{s} = \text{s}_{target} \\ -1 & \text{s} = \text{s}_{LoS} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

B. Deep Reinforcement Learning: Soft Actor Critic

To solve the (PO)MDP defined in section III-A use is made of the Soft Actor Critic (SAC) DRL algorithm. SAC is an off-policy, model-free, DRL algorithm, which means that it can learn from past experiences without explicitly knowing the environment dynamics or reward function. Off-policy is preferred over on-policy methods because they have a lower sample efficiency, which would result in slower learning. SAC also has shown to be very stable during training, even in environments with sparse-rewards, which makes it a prime candidate for this research [16].

C. Baseline Resolution Algorithm: Modified Voltage Potential

To better understand the performance of the DRL models and to put the performance into perspective, all scenarios are also simulated with the MVP conflict resolution algorithm [12]. MVP determines the closest point of approach of two aircraft, and, if the distance between the two aircraft at CPA is smaller than the minimum separation distance, a repelling 'force' is determined which changes the velocity vector such that the shortest way out of the conflict is determined. This is visually presented in Figure 2. This also entails that, unlike the DRL agents, MVP agents will only change the current course if the aircraft is in conflict, following the shortest way out. To ensure a fair comparison between the MVP and the DRL models, the MVP model will have the same constraints on their degrees of freedom imposed as their DRL counterpart.

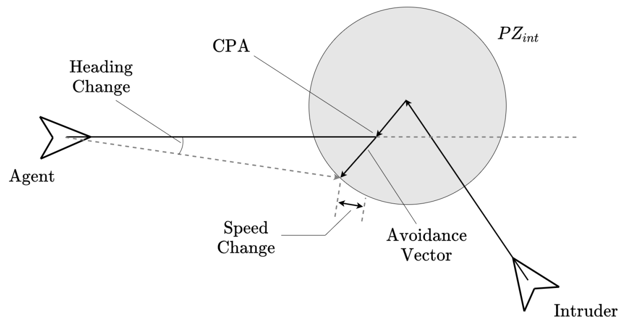


Figure 2. Graphical representation of the Modified Voltage Potential algorithm, adapted from [12].

IV. Experimental Setup

A. Experimental Scenario

For all of the conducted experiments, the goal of the agent is to traverse through the different layers in an altitude layered airspace and reach the target layer without having any intrusions.

The layered airspace in question consists of 2 sets of 8 altitude layers, each having an allowed heading range of 45 degrees, covering all the possible heading angles twice. The general use of the 2 sets of layers is that long-distance travel can be done at higher velocities in the top layers, whereas short-distance commute is allocated to the slower bottom layers [5]. For this research the different layers however function solely as a way to artificially generate the need for vertical manoeuvres. Between each layer exists a transition layer that can only be accessed by aircraft conducting vertical manoeuvres, which allows the agent to adapt to the new layer before merging. All layers are 25ft in height. This structure is presented visually in Figure 3.

Within this airspace, aircraft operating in the top 8 layers will have a chance to obtain a descent command to one of the 8 bottom layers, simulating the delivery of a package. Similarly, aircraft flying in the bottom 8 layers have a chance to get a climb command, simulating the return to a warehouse or place outside of the city. This chance is selected such that on average 5% of the aircraft in the airspace are conducting vertical manoeuvres at any given time. This means that at any given time roughly 5% of the aircraft in the airspace will be controlled by either DRL or MVP.

B. Traffic Density and Conflict Probability

The traffic density in the airspace is selected to be $55AC/NM^2$, equally distributed over all of the heading layers. Using equation 3, which is the ratio between the volume searched for conflicts and the total airspace volume ,the conflict probability between any two aircraft, p, can be obtained [9].Using a look-ahead time equal to the mean duration of a vertical manoeuvre, $t_l = 30s$, an airspace volume, $B_{total} = 8.1E8m^3$ ($1NM^2$ times airspace height both in meters), $PZ_h = 50m$, $PZ_v = 7.6m$, $\mathbf{E}\left(V_{r,h}\right) = \frac{4\cdot 10}{\pi} m/s$ and $\mathbf{E}\left(V_{r,v}\right) = 4m/s$ yields the conflict probability between a vertically manoeuvring and any other aircraft, $p_{h,v}$. Substituting this value in equation 4 to obtain the conflict probability for a specific aircraft with all other aircraft ($N_{ac} = 54$) yields a conflict probability, p_{conf}, of 9.9%. When not using any resolution methods this value should be roughly equal to the intrusion rate, which is confirmed in initial simulations (Figure **??**). It is assumed that this intrusion rate is high enough to observe changes in the safety with respect to no resolution, and low enough not to saturate the solution space.

$$p = \frac{B_{c,h} + B_{c,v}}{B_{\text{total}}} = \frac{4PZ_h PZ_v \mathbf{E}\left(V_{r,h}\right) t_l + \pi PZ_h^2 \mathbf{E}\left(V_{r,v}\right) t_l}{B_{\text{total}}} \tag{3}$$

$$p_{conf} = 1 - \left(1 - p_{h,v}\right)^{N_{ac}} \tag{4}$$

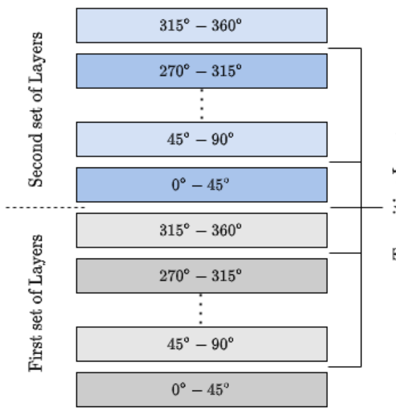


Figure 3. Figure showing the heading range of the different altitude layers in the airspace.

C. Control Variables

1) Simulation time-steps: The simulation is run with time-steps of 1.5 seconds. This entails that the DRL agent selects an action for the aircraft every 1.5 seconds. The MVP agent selects an action for the aircraft every 1.5 seconds only when in conflict.

2) Minimum Separation: The protected zone around all aircraft is set at 50m horizontally (PZ_h) and 25 feet vertically (PZ_v). These values are based on comparable work [17], as currently no standard for separation requirements has been specified for unmanned aviation.

3) Conflict Detection: For all experiments, instead of look-ahead time use is made of a 'search cylinder' with a radius of 500m, spanning from the agent's altitude to the altitude of the target layer. All aircraft within this cylinder with a $D_{cpa} < PZ_h$ are evaluated for potential conflicts. This is done by comparing the times in and out of the horizontal and vertical minimum separation. If there is overlap between these times the aircraft are labelled as in conflict. The choice for a look-ahead distance instead of look-ahead time is made to ensure that aircraft that are flying (almost) parallel to the agent, but that are very close in absolute distance, will not be overlooked for state inclusion. This has as a drawback that aircraft with a very high relative velocity, and therefore a much smaller T_{cpa} than other aircraft, might initially be ignored.

4) Default Speeds: All cruising aircraft will be flying at a constant horizontal velocity of 10m/s. The default vertical velocity during climb or descent is 4m/s.

5) Conflict Resolution: For all of the aircraft that are not conducting vertical manoeuvres the conflict resolution is turned off. This means that solely the agents are responsible for resolving any conflicts.

D. Dependent Variables

1) Safety: To assess the safety of the vertical manoeuvres, three safety parameters are used: the average number of conflicts encountered during a vertical manoeuvre, average time spent flying in conflicts and the average number of intrusions or losses of minimum separation. The latter is the most important as it directly relates to the safety of the operations. The number of conflicts encountered can give a good indication of the relative stability between the different methods and the percentage of time spent flying in conflict can be related to the efficacy of the performed resolution manoeuvres or for the DRL models also to the attributed urgency of a conflict.

2) Efficiency: To evaluate the efficiency of the models, two parameters will be used. The horizontal flight distance during vertical operations, and the total flight time of the vertical manoeuvre. It can be considered efficient to minimize both of these parameters. A low horizontal flight distance during vertical operations requires less spatial planning, as descent or ascent operations can be started just before reaching the target destination. Similarly, lower duration of the vertical manoeuvres reduces the required planning of the operations and has as an additional benefit that more of the flight time will be spent in cruise.

E. Experimental Hypotheses

1) Action Space Usage: For the action space usage, it is hypothesized that all the models that can control the vertical velocity will opt for a high mean vertical velocity. This hypothesis is based on the findings of Sunil and Tra where it is shown that a slower vertical velocity leads to more intrusions [7], [8]. This makes sense as a lower vertical velocity extends the duration of these operations, thus increasing the cumulative probability of an intrusion occurring.

For the horizontal velocity, it is expected that the mean horizontal velocity will be equal to the mean cruise velocity. This also implies that the mean horizontal velocity change will be equal to zero. This hypothesis stems from the fact that having a horizontal velocity equal to the cruise velocity lowers the relative horizontal velocity between the aircraft.

Finally, the heading changes are expected to be small of magnitude, as large heading changes will also change the observed aircraft by the agent considerably. From a predictability point of view this is unfavorable, as the agent has less control over the next state.

2) Differences in Performance: For the performance differences, it is hypothesized that the models with more degrees of freedom will outperform the models with fewer degrees of freedom at the cost of higher training time. This is because it was shown that having more degrees of freedom increase the performance of a DRL model in a lane-changing and merging task on the highway [10].

Furthermore, it is expected that the 'hor' model will perform worse when compared with the models controlling the vertical velocity of the agents. This is because during vertical conflicts a trivial solution to resolve the conflict exists by setting the vertical velocity to zero (given that the current vertical separation is more than the required separation and not accounting for secondary conflicts). Such a trivial solution does not exist when solely using heading and speed commands.

Finally, it is not expected that the DRL models will outperform the MVP models. This is hypothesized as MVP has been shown to very efficiently resolve conflicts, the DRL model on the other hand is an initial attempt in terms of MDP formulation and model selection/tuning. The results will however give an indication of the potential of DRL for establishing safe manoeuvring policies.

V. RESULTS

For the results, the performance of the final trained DRL models will be shown next to the performance of the MVP conflict resolution algorithm with the same degrees of freedom. First, the usage of the action space will be shown to allow for better understanding of the final trained policies. Then, the safety metrics will be shown followed by the efficiency of the different models. For the results, more than 10.000 vertical flight manoeuvres have been simulated for each model

A. Model Policy Differences

To better understand the differences in performance in terms of safety and efficiency the differences in policies between the different models is shown. This is done through a selected action boxplot for all 3 actions, given in Figure 4.

Analysis of the action space usage showed that the DRL model's policy is semi-independent on the Boolean conflict variable, which specifies if an agent is in conflict or not. Instead, a more notable correlation is observed with the T_{cpa} and D_{cpa} variables, as indicated in figure 5, which shows the mean magnitude of the actions for all DRL models against T_{cpa} and D_{cpa}. From this figure it is noticeable that the most prevalent differences in the policy can be observed for D_{cpa} $<$ 100m and $T_{cpa} <$ 20s.

This shows that the agent changes course even if the agent is not necessarily in conflict. Therefore it is decided to define the "Conflict Resolution" category in Figure 4 as all actions selected whilst T_{cpa} is smaller than 20 seconds and D_{cpa} smaller than 100 meters. The "Non-Conflict" category is then defined as all other actions. An exception to this definition is for the MVP model, which is called based on the Boolean conflict variable, for this model the category is simply: selected actions whilst in conflict.

The high number of outliers in Figure 4 is also a direct result of highlighting only the differences based on T_{cpa} and D_{cpa}, the influence of other state variables on the selected action can result in these frequent outliers. However, for clarity, it is decided to present the data in this fashion.

From Figure 4 it is visible that a broader area of the action space is utilized by the DRL models when resolving conflicts than during nominal flight, indicated by the larger whiskers and higher frequency of outliers. This shows that the DRL models successfully learn the concept of danger, and understand that continuing the current course of action might result in dangerous states or even intrusions.

Similarly, the narrow distribution during nominal flight conditions shows that the agent understands that, in principle, there is no need to change the actions during safe operations.

Figure 4a shows the selected vertical velocities by the different models. What is interesting to notice is the differences in policy between the 'vert' and the 'vert+' model. The 'vert' model seemingly prefers a low vertical velocity during nominal operations, during conflict resolution the mean vertical velocity is slightly lower whilst simultaneously having more outliers in the direction of increasing vertical velocity. The exact opposite policy is observed for the vert+ model. The reason for the lower vertical velocities of the 'vert' and 'full' models is currently unknown and contradicts the hypothesis that the models would prefer a higher vertical velocity to finish their vertical manoeuvre more quickly.

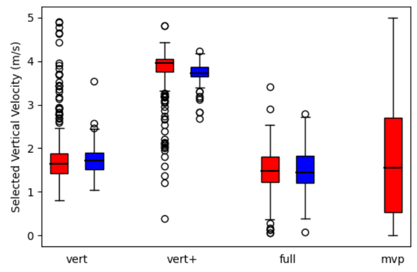
For the horizontal velocities it is visible that the mean velocity change is centered around zero for all models except the 'full' model. This partially confirms the hypothesis that the models will prefer a horizontal velocity equal to the horizontal cruise velocity. As the 'full' model has a lower mean vertical velocity it might be possible that the lower horizontal velocity is used to increase the climb/descent angle, although more research is required to see if this behavior is actually beneficial for the safety of the operations.

Overall it seems as if the differences in policy between "Conflict Resolution" and "Non-Conflict" are less noticeable for the 'full' model. It is possible that, because it can combine 3 different actions, the required magnitude of the actions to transition to a safer state is lower, indicated also by the relatively low spread in actions. Furthermore, as the DRL model does not necessarily resolve the conflict in a single time-step this smaller spread in selected actions does not indicate that the DRL model is capable of resolving the conflict with smaller total deviations than MVP.

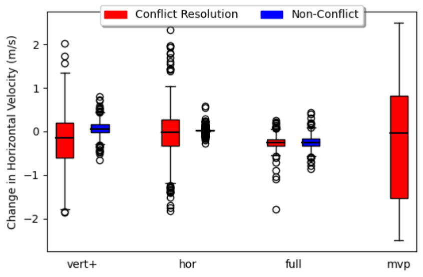
The difference is that MVP will always compute the required actions to resolve the conflict in a single time-step, the DRL model on the other hand can decide to wait before resolving or resolve a conflict in a series of small increments. This is also the reason why MVP models seemingly select actions of higher magnitude when compared with DRL models.

B. Safety Analysis

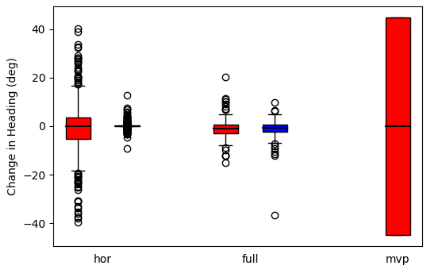
First, looking at the total number of conflicts shown in Figure 6, it becomes apparent that for all cases the total number of conflicts encountered during operations increases



(a)



(b)



(c)

Figure 4. Boxplots for the selected actions during conflict resolution and during non-conflict situations. a) Selected (absolute) vertical velocity. b) Selected horizontal velocity changes. c) Selected heading changes. Note that these are selected actions per time-step.

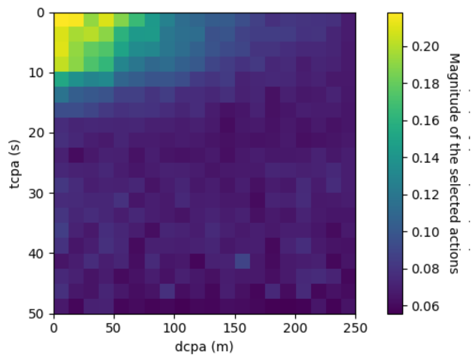


Figure 5. Color plot of the mean selected action magnitude, averaged for all the DRL models and different actions, normalized between 0 and 1. Plotted against T_{cpa} and D_{cpa}

with respect to the metrics for not using any conflict resolution. This is a common phenomenon often described as the Domino Effect [18]. In essence, the resolving manoeuvres conducted by the agents result in a larger volume of airspace being used. This in turn increases the number of potential conflict pairs when compared to flying in a straight line. The largest increase in the number of conflicts is observed in the 'vert' and 'full' models, and can be partially related to the overall lower vertical velocity these models have during operations, as shown in Figure 4a.

Because of these lower vertical velocities, the duration of vertical operations is also increased, which can result in an increase of conflicts encountered. It is interesting to note that this "Domino Effect" is less apparent in the 'vert+' and 'hor' DRL models than for their respective MVP models. It is hypothesized that this might be due to the fact the DRL model is capable of selecting different actions, even when not in conflict. This, for example, allows the agent to delay returning to nominal conditions after a resolution if it is observed that this would result in a new conflict. Because of this conflicts are prevented and therefore not observed for this metric.

Looking at the percentage of time spend flying in conflicts, shown in Figure 7, for most cases a decrease with comparison to no resolution is observed. This can be attributed to the conflict resolving actions both the MVP and DRL models use, which effectively shortens the duration of the conflicts. The only model for which this decrease is not observed is for the 'full' model. As the total number of conflicts is comparable to the 'vert' model, this indicates that the duration of conflicts for the 'full' model in general is longer. This might be caused by either the postponing of resolution manoeuvres or due to ineffective resolution manoeuvres.

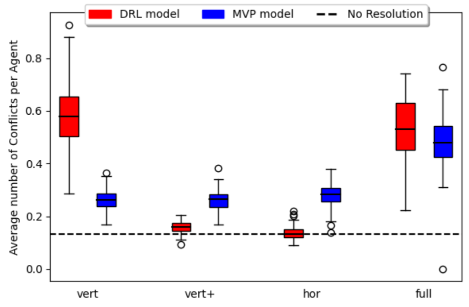


Figure 6. Average number of conflicts encountered during a vertical manoeuvre.

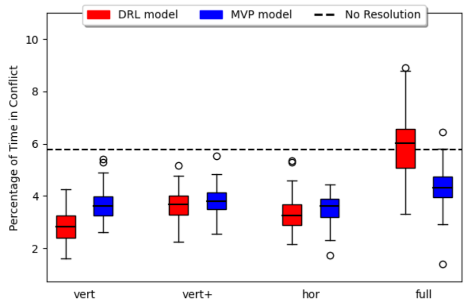


Figure 7. Average percentage of time spent in conflict while conducting vertical manoeuvres.

To shine a bit more light on this conflict duration, and to illustrate the difference between the DRL and MVP models with regards to resolution manoeuvres, it is interesting to look at the mean number of actions required before a conflict is resolved. This metric is indicated in table III. Here it is clearly visible that indeed the MVP models resolve the conflicts much faster, as explained by their larger action magnitude in figure 4. For the MVP models, the agent requires less actions when increasing the degrees of freedom, which makes sense, considering that the effectiveness of the resolution action should increase with more degrees of freedom (not considering secondary conflicts). The opposite is observed for the DRL models, this might indicate that with more degrees of freedom it is possible to delay the resolution manoeuvre even longer. The outlier in the DRL 'vert' model only requiring on average 2 actions to resolve a conflict when compared to the other DRL models can not yet be explained.

TABLE III. Mean number of actions taken by the agents before a conflict is resolved

	vert	vert+	hor	full
DRL	2.1	4.6	4.8	5.2
MVP	2.9	3.0	2.4	1.8

Finally, looking at the total number of intrusions per flight given in Figure 8, it can be seen that all of the models successfully reduce the total number of intrusions when compared to no resolution. The first thing that becomes apparent from this figure is that including heading manoeuvres for the MVP model decreases its performance, as shown by the 'hor' and 'full' MVP models. This may be a direct effect of the fact that the MVP model's heading actions are cut off to the bounds imposed on the action space, [±45°], leading to sub-optimal heading actions. This however would only be the case for last-second conflict resolutions, as larger heading changes could still be attained by splitting the required heading change over multiple time-steps. Another possibility is that large heading manoeuvres are sub-optimal in high-density layered airspace.[9]

Closer inspection of Figure 8 also shows that increasing the degrees of freedom does not necessarily result in a safer policy. This is interesting, as the policy of the DRL 'vert' model is part of the solution space of the 'vert+' and 'full' models. Similarly the policy conducted by the MVP models is also part of the solution space of their respective DRL models. Because the performance of the 'vert+' and 'full' models does not match the performance of better available policies (in this case the policy of the 'vert+' MVP model), it can be concluded that these models are likely stuck in a local optimum. This highlights one of the drawbacks of using Deep Reinforcement Learning for higher-dimensional problems, with more actions, the required exploration increases exponentially, increasing the required training time whilst decreasing the guarantee of convergence to the global (or a more optimal, local) optimum.

A final interesting remark is that the DRL model found a horizontal resolution method that outperforms the MVP model in terms of safety. Figure 4 already showed that the DRL models also acted when not in conflict. This might be a reason why the horizontal DRL model encounters fewer conflicts throughout the operations than the MVP model, which in return results in fewer conflicts potentially leading to an intrusion. Further analysis however also shows that a conflict encountered by the DRL model has a 22.8% chance of resulting in an intrusion versus 25.6% chance for the MVP model. This is not enough to state a significant difference, but it does show that the DRL model effectively resolves a comparable number of conflicts.

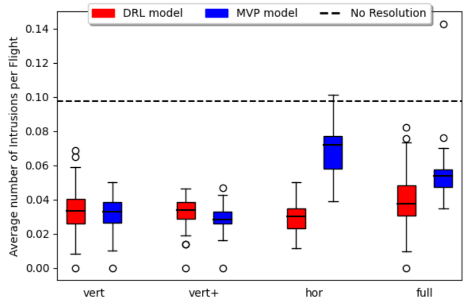


Figure 8. Number of intrusions per vertical manoeuvre.

C. *Efficiency Analysis*

Figure 9 shows the average duration of the vertical manoeuvres. As the duration of the vertical manoeuvres is directly related to the mean vertical velocity, this plot is highly correlated with Figure 4a. This image clearly shows that MVP is relatively constant in travel time regardless of the degrees of freedom used for resolving the encountered conflicts in comparison to the DRL models. This is largely because the MVP models are only called when in conflict, which from Figure 7 can be seen to equal roughly 4% of the travel time. As the DRL models select actions every time-step of the simulation (versus MVP which only selects actions when in conflict) the models are capable of flying at a lower vertical velocity throughout the entirety of the operation, this is why the 'vert' and 'full' DRL models take so much longer.

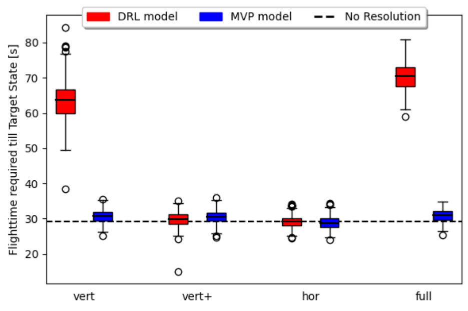


Figure 9. Time required to reach the target altitude layer.

As expected, the results for the horizontal travel distance, shown in Figure 10, are highly correlated with the duration of the vertical manoeuvres. One outlier however is the 'full' model, which seemingly reduces the mean horizontal velocity during nominal operations, reducing the horizontal travel distance to be closer to the baseline. This is interesting as the other models that controlled the horizontal velocity, 'vert+' and 'hor', did not change their horizontal velocity and instead adhered to the mean cruise velocity of the airspace. Further analysis is required to understand why the 'full' model opted for this behaviour.

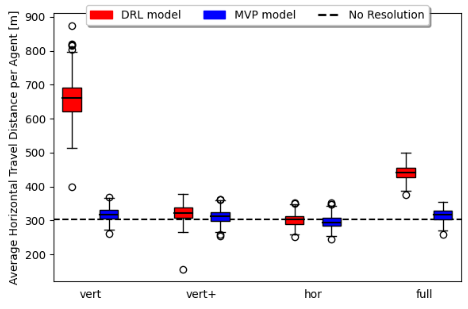


Figure 10. Average horizontal distance travelled during the vertical transition.

VI. Discussion

The results of this research have shown that Deep Reinforcement Learning can be used to train a model that learns to reduce the number of intrusions during vertical manoeuvres in a layered airspace. It learned this from a simple reward function that only rewarded successful operations and penalized intrusions. This shows that simple reward structures can be used in complex environments, which has as added benefit that it is easy to visualize what the desired behaviour of the agent is when compared to more complex reward structures.

During the safety analysis, the DRL model outperformed the MVP model in both the horizontal and 3 degrees of freedom scenarios. This performance difference can be attributed to the fact that the DRL model is able to perform conflict resolving manoeuvres at different moments. MVP on the other hand performs the resolution manoeuvre at the moment a conflict becomes apparent in the state vector, which is bounded by the look-ahead distance. DRL has the freedom to find the optimal moment for conflict resolution. In previous work, it has already been demonstrated that, at higher traffic densities, having a constant look-ahead time might not be optimal [19]. Furthermore, the DRL model has control over the selected actions during the recovery phase after resolving a conflict, allowing it to prevent new conflicts from happening.

Another element of the DRL model's policy that is interesting to further investigate is the fact that the model preemptively acted when aircraft would get close, but would not (yet) be in conflict. This behaviour effectively increases the minimum horizontal separation the model adheres to, which theoretically could decrease the stability of the manoeuvres by increasing the number of conflict resolution manoeuvres [9]. It is possible that the DRL model actively acts as a conflict prevention mechanism apart from resolving conflicts. This has two potential benefits, preventing conflicts requires smaller deviations from the current flight path than resolving conflicts and larger margins with other aircraft increases the available solution space in the case of new conflicts, potentially reducing the occurrence of multi-conflict scenarios where finding a solution is difficult. When looking at the total number of conflicts it seems as if this strategy does indeed lead to a minimal increase in secondary conflicts for both the 'vert+' and 'hor' models.

From the results, is is also visible that the DRL models are not converged to the global optimum. This means that the performance of the DRL model could potentially be further improved with either more training (unlikely if the model is stuck in a local optimum) or a better definition of the Markov Decision Process (MDP). One of the main problems with the current implementation of the MDP is the presence of partial observability in the state representation. Once a problem becomes partially observable, the theoretical guarantee of eventual convergence to the global optimum no longer holds. As the state-space in the simulation environment is continuous and the environment in real life would be unbounded, partial observability will always be a problem. However, expanding the state representation to include more aircraft, researching a more consistent representation and potentially also including historic states (k-th order history approach [15]) can all be done to decrease the impact of partial observability on the performance. Especially the inclusion of historic observations in the state can prove to be of utmost importance, as currently, it is possible that the agent resolves a conflict and is no longer aware of the existence of the old conflict in the next time step, causing the agent to revert to the old state and back into conflict. This indicates that with the current implementation of the MDP the problem has areas in which the Markov Property does not hold. Another possibility would be to still include aircraft, with which the agent previously was in conflict, in the state, even though there is no imminent danger anymore. A final interesting experiment would be to research the effect of number of aircraft in the state vector on the performance of the model. It might be that the performance gradually goes up indefinitely do to reduction of the partial observability, at the same time however a larger state will lead to longer duration of the training time. Optimizing this trade-off can potentially increase the performance further.

Finally, there are some limitations to the results. For example, the traffic scenarios can be adapted to have higher or variable traffic densities and include aircraft flying at different cruise velocities. Furthermore, it is difficult to anticipate how the model would perform in more complex traffic where not all aircraft would adhere to the altitude layers. Apart from this, elements such as static obstacles or maximum horizontal distance travelled during the vertical operations can also influence the effectiveness of the trained model. To estimate the true effectiveness of DRL for safe manoeuvring, it should be trained and tested in a variety of different traffic scenarios consisting of operations during all stages of flight (potentially using different models/policies for different conditions). An initial step in this direction would be the activation of conflict resolution for cruising aircraft. This extra element will remove much of the stationarity and therefore the predictability from the environment, and will better show the ability of the DRL model to deal with emergent behavior. This would however also lead to massive multi-agent operations, which will negatively impact the stability and duration of training. Overall, the results obtained however do show that DRL can potentially be used for improving the safety and can provide new insights into the understanding of safe operations.

VII. Conclusion

This paper analysed the capabilities of Deep Reinforcement Learning (DRL) for improving the safety of vertical manoeuvres in a layered airspace through direct control. It was shown that DRL is capable of learning policies that effectively reduce the number of intrusions for a variety of different degrees of freedom, even outperforming the Modified Voltage Potential algorithm in certain scenarios. This work shows that DRL can successfully be used for detect and avoid operations in high traffic density scenarios. More research is still required in the design of the Markov Decision Process as well as the DRL model selection to further improve on the obtained performance. Apart from that, more extensive analysis is required on the failure cases of the current DRL model to better understand the weaknesses and areas of improvement. Finally, future work should investigate the usage of DRL in more competitive and changing traffic scenarios such as non-uniform traffic densities and for different control tasks such as horizontal control.

References

[1] D. Pierce, "Delivery drones are coming: Jeff Bezos promises half-hour shipping with Amazon Prime Air," https://www.theverge.com/2013/12/1/5164340/delivery-drones-are-coming-jeff-bezos-previews-half-hour-shipping, 2013, [Online; accessed: 16.3.2021].

[2] "DHL Express", ""dhl expres launches its first regular fully-automated and intelligent urban drone delivery service." https://www.dhl.com/tw-en/home/press/press-archive/2019/dhl-express-launches-its-first-regular-fully-automated-and-intelligent-urban-drone-delivery-service.html, 2019, [Online; accessed: 16.3.2021].

[3] M. Doole, J. Ellerbroek, and J. Hoekstra, "Drone delivery: Urban airspace traffic density estimation," *8th SESAR Innovation Days, 2018*, 2018.

[4] "Organization, i.c.a. icao circular 328 - unmanned aircraft systems (uas). technical report, icao, 2011."

[5] E. Sunil, J. Hoekstra, J. Ellerbroek, F. Bussink, D. Nieuwenhuisen, A. Vidosavljevic, and S. Kern, "Metropolis: Relating airspace structure and capacity for extreme traffic densities," in *Proceedings of the 11th USA/Europe Air Traffic Management Research and Development Seminar (ATM2015), Lisbon (Portugal), 23-26 June, 2015*. FAA/Eurocontrol, 2015.

[6] J. M. Hoekstra, J. Ellerbroek, E. Sunil, and J. Maas, "Geovectoring: reducing traffic complexity to increase the capacity of uav airspace," in *International conference for research in air transportation, Barcelona, Spain*, 2018.

[7] E. Sunil, J. Ellerbroek, J. M. Hoekstra, and J. Maas, "Three-dimensional conflict count models for unstructured and layered airspace designs," *Transportation Research Part C: Emerging Technologies*, vol. 95, pp. 295–319, 2018.

[8] M. Tra, E. Sunil, J. Ellerbroek, and J. Hoekstra, "Modeling the intrinsic safety of unstructured and layered airspace designs," in *2017 ATM R&D Seminar*, 2017.

[9] E. Sunil, J. Ellerbroek, and J. M. Hoekstra, "Camda: Capacity assessment method for decentralized air traffic control," in *Proceedings of the 2018 International Conference on Air Transportation (ICRAT), Barcelona, Spain*, 2018, pp. 26–29.

[10] C.-J. Hoel, K. Wolff, and L. Laine, "Automated speed and lane change decision making using deep reinforcement learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2148–2155.

[11] M. Ribeiro, J. Ellerbroek, and J. Hoekstra, "Review of conflict resolution methods for manned and unmanned aviation," *Aerospace*, vol. 7, no. 6, p. 79, 2020.

[12] J. M. Hoekstra, R. N. van Gent, and R. C. Ruigrok, "Designing for safety: the 'free flight'air traffic management concept," *Reliability Engineering & System Safety*, vol. 75, no. 2, pp. 215–232, 2002.

[13] 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. FAA/Eurocontrol USA/Europe, 2016, p. 132.

[14] R. Bellman, "A markovian decision process," *Journal of mathematics and mechanics*, vol. 6, no. 5, pp. 679–684, 1957.

[15] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[16] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

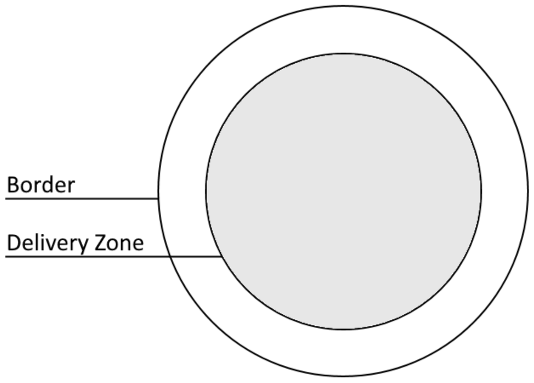
[17] M. Ribeiro, J. Ellerbroek, and J. Hoekstra, "Velocity obstacle based conflict avoidance in urban environment with variable speed limit," *Aerospace*, vol. 8, no. 4, p. 93, 2021.

[18] K. Bilimoria, K. Sheth, H. Lee, and S. Grabbe, "Performance evaluation of airborne separation assurance for free flight," in *18th Applied Aerodynamics Conference*, 2000, p. 4269.

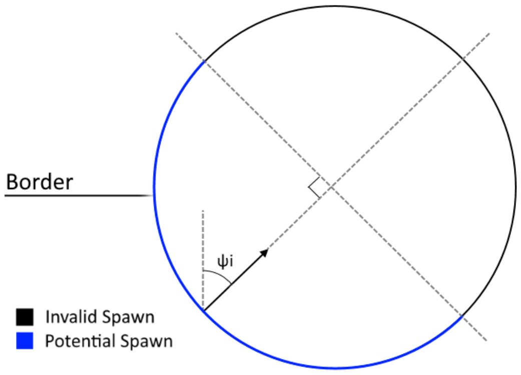
[19] M. Ribeiro, J. Ellerbroek, and J. Hoekstra, "Determining optimal conflict avoidance manoeuvres at high densities with reinforcement learning," *10th SESAR Innovation Days*, 2020.

Appendix A
Additional Simulation Parameters

All of the simulation are conducted in an enclosed airspace with a radius of 1.62NM. Within this enclosed airspace a second area is established which is used for the experiments, called the delivery area, to ensure a more uniform traffic density during the experiments. The radius of this second area is 1.35NM. This is visually presented in figure 11a



(a)



(b)

Figure 11. a) Simulation Area and Experiment Zone. b) Representation of the locations where an aircraft with intended heading ψ_i can spawn.

To sustain the traffic density of 55AC/NM2 aircraft are spawned randomly on the outer border of the environment. The spawn process of the aircraft is as follows; first the intended heading, ψ_i, is determined by means of a random integer, $\psi_i \in [0,360]$, with a uniform probability for all values. Then a second random variable, $y \in [0,1]$, will determine if the aircraft will be located in the top or bottom set of layers. The altitude of the aircraft can then be determined by means of equation 5. Note that the element $h_{layer}\left\lfloor \frac{\psi_i}{\psi_{layer}} \right\rfloor$ is multiplied by 2 in the equation in order to prevent spawning aircraft in the transition layers.

$$h_i = h_{min} + 2h_{layer}\left\lfloor \frac{\psi_i}{\psi_{layer}} \right\rfloor + \frac{y(h_{min}+h_{max})}{2} + \frac{h_{layer}}{2} \tag{5}$$

For the spawn location there is a uniform chance that the aircraft will be spawn at the border of the environment, given that the aircraft's intended heading will ensure that the aircraft enters the simulation area. This is shown in figure 11b. This means that the bounds for the allowed spawn bearing with respect to the center of the environment are $[90+\psi_i, 270+\psi_i]$, from which the final spawn location will be randomly selected.

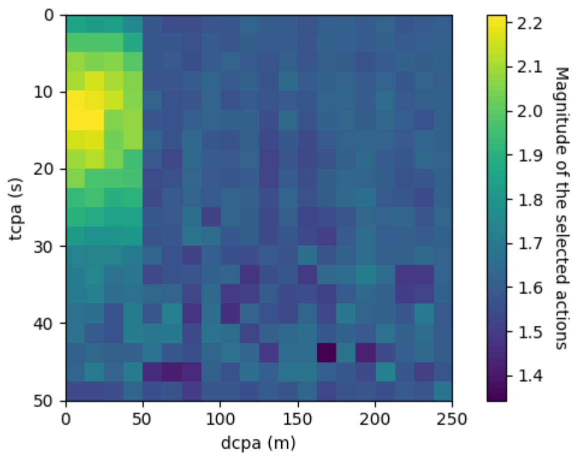
Appendix B
Model Parameters

TABLE IV. Hyperparameters used for the actor and critic networks of the SAC model

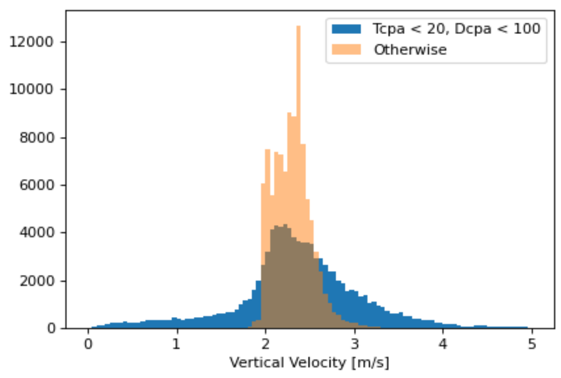
Parameter	Value
Learning Rate (α & β)	6E-5
Number of Layers	2
Number of Neurons per Layer	256
Memory Buffer Size	1E6
Sample Size	256
Optimizer	Adam
Activation Function	ReLU
Reward Scale	20
Discount Factor (γ)	0.99
Target Smoothing Coefficient (τ)	0.005

APPENDIX C
SELECTED ACTION HEATMAPS AND HISTOGRAMS

In Figure 13a the action color map for the 'vert' model is shown, interestingly, this model seems to deviate from the other models be clearly basing the policy on the conflict boolean. Figure 13b contains the histogram of the selected actions for the 'vert' model.



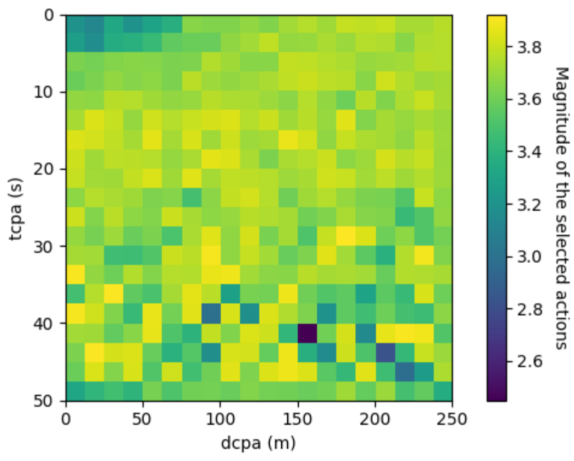
(a)



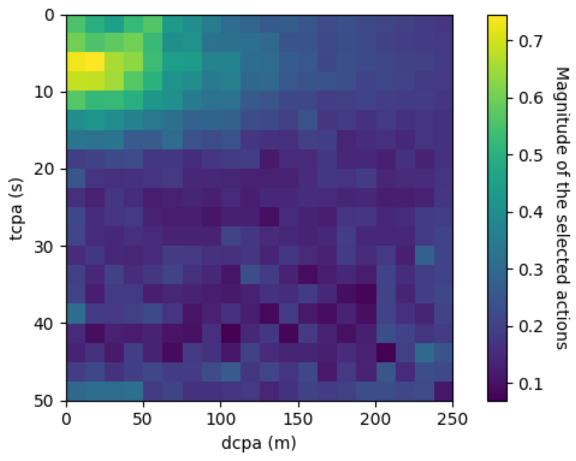
(b)

Figure 12. Action heatmap and histogram for the 'vert' model

In Figure 13 the colormaps for both actions of the 'vert+' model are shown. Figure 14 contains the corresponding histograms to these heatmaps to illustrate certain differences a bit better.

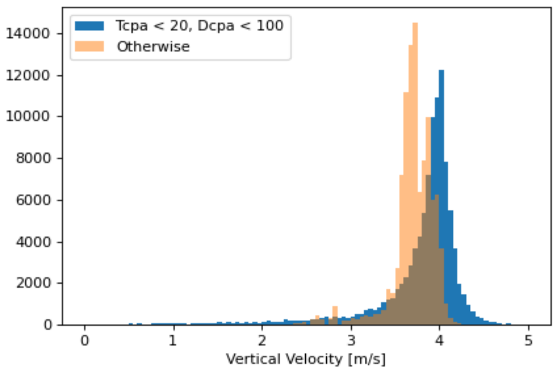


(a)

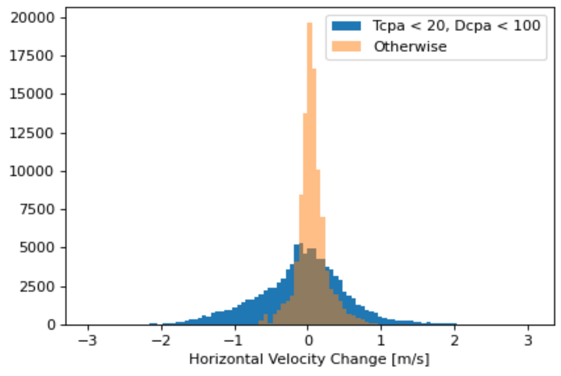


(b)

Figure 13. Action heatmaps for the 'vert+' model. a) vertical velocity, b) absolute horizontal velocity changes



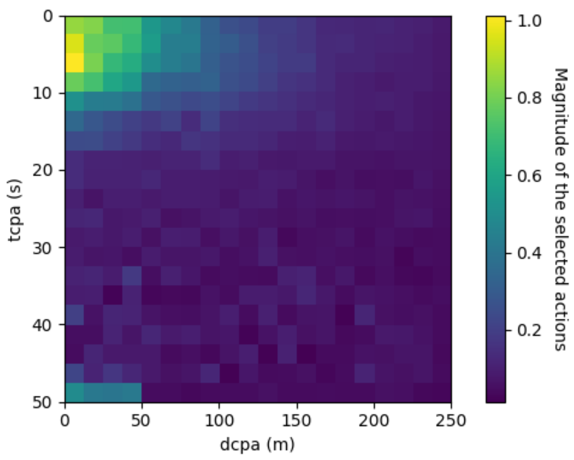
(a)



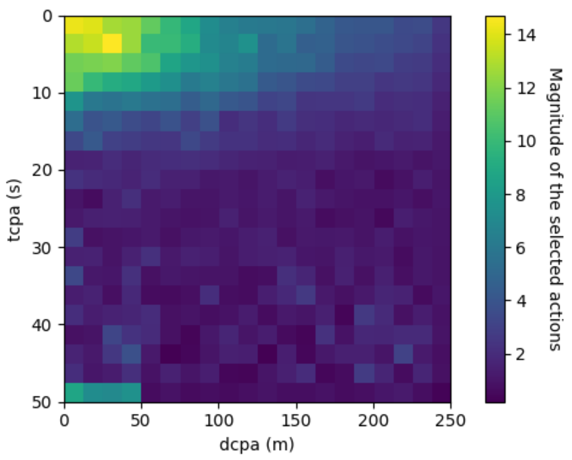
(b)

Figure 14. Action histograms for the ‘vert+’ model. a) vertical velocity, b) horizontal velocity changes

Figures 15 and 16 illustrate the colormaps and histograms of the selected actions for the ‘hor’ model respectively. It is interesting to observe the similarities in the policies for both actions with the main visible difference being that heading change are apparently used even at larger D_{cpa}.

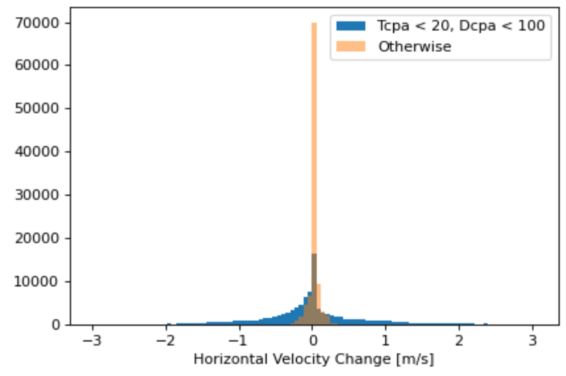


(a)

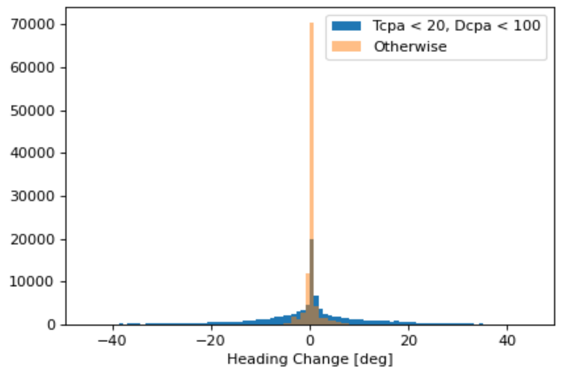


(b)

Figure 15. Action heatmaps for the ‘hor’ model. a) absolute horizontal velocity changes, b) absolute heading changes



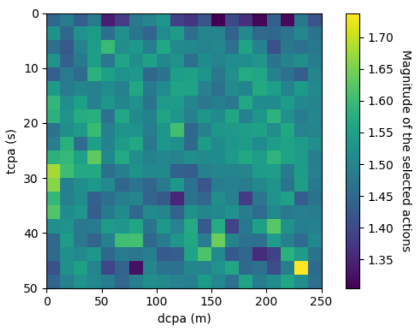
(a)



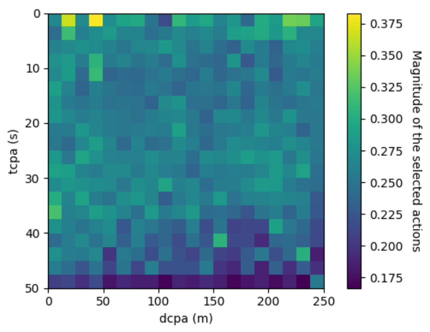
(b)

Figure 16. Action histograms for the ‘hor’ model. a) horizontal velocity changes, b) heading changes

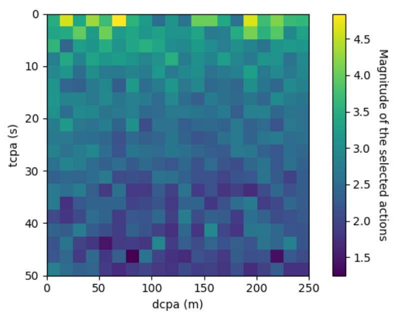
Figures 17 and 18 illustrate the colormaps and histograms of the selected actions for the ‘full’ model respectively. These colormaps illustrate that the policy of the ‘full’ model is indeed difficult to separate between standard operations and conflict resolution operations as also visible in Figure 4.



(a)

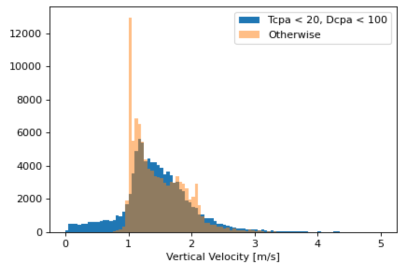


(b)

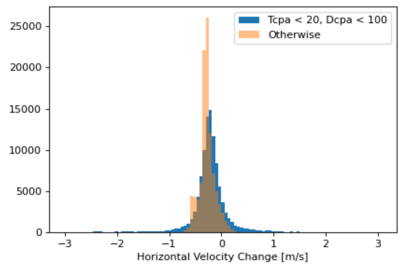


(c)

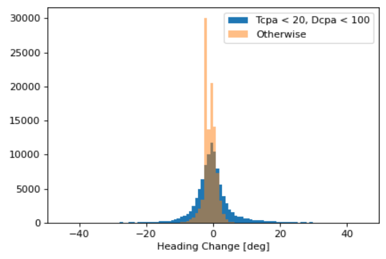
Figure 17. Action heatmaps for the ‘full’ model. a) vertical velocity, b) absolute horizontal velocity changes, c) absolute heading changes



(a)



(b)



(c)

Figure 18. Action histograms for the ‘full’ model. a) vertical velocity, b) horizontal velocity changes, c) heading changes

Part II

Preliminary Report - Already Graded

List of Figures

List of Tables

Nomenclature

Abbreviations

ANN	Artificial Neural Network
ATCo	Air Traffic Controller
ATM	Air Traffic Management
CDR	Conflict Detection & Resolution
CR	Conflict Resolution
Dcpa	Distance at closest point of approach
DDPG	Deep Deterministic Policy Gradient
DEP	Domino Effect Parameter
DNN	Deep Neural Network
DRL	Deep Reinforcement Learning
FAA	Federal Aviation Administration
ICAO	International Civil Aviation Organisation
LoS	Loss of Separation
MAMDP	Multi Agent Markov Decision Process
MARL	Multi-Agent Reinforcement Learning
MDP	Markov Decision Process
MVP	Modified Voltage Potential
OU	Ornstein Uhlenbeck
PAV	Personal Air Vehicle
POMDP	Partially Observable Markov Decision Process
Tcpa	Time till closest point of approach
TLoS	Time till Loss of Separation

List of Symbols

ϵ	Ratio between cruising and climbing/descending aircraft
γ	Discount factor
$\mathcal{A}$	Action space of the MDP

$\mathcal{S}$	State space of the MDP
μ	Actor network
μ'	Target actor network
π	Policy
ψ	Heading of an aircraft
ψ_v	Relative heading between two velocity vectors
τ	Soft target network update parameter
θ^w	DNN with weights w
a_t	Selected action at time t
du	Relative velocity in longitudinal direction
G	Return, sum of the rewards from a single episode
h_{layer}	Height of a single layer in the airspace
M	Replay buffer
$N_{aircraft}$	Number of aircraft included in the state representation
P	State transition function of the MDP
Pz_h	Radius of the horizontal protection zone surrounding an aircraft
Pz_v	Height of the vertical protection zone surrounding an aircraft
Q	Critic network
Q'	Target critic network
q_π	State-action value function for a given policy π
R	Reward function of the MDP
r	Reward determined by the reward function R
s_t	State at time t
V_h	Horizontal velocity
V_z	Vertical velocity
v_π	State value function for a given policy π
Vu	Longitudinal velocity of an aircraft
y	Expected return following reward r

1. Thesis Framework

This introductory chapter of the report functions to inform the reader about the background of the research, the goals and the research approach defined. First a general introduction will be given. Secondly, the research objective and questions will be formulated. Then the research approach is given followed by some general assumptions. Finally, the outline of the rest of the report will be given.

1.1. Introduction

Recent years have seen more and more companies researching the commercial viability of using drones for their operations, mostly for last-mile delivery [1, 2]. Current estimates for the number of drones operating by 2035 range from 100.000 to 275.000 drone-based operations per hour for the city of Paris, which equates to 40-100 movements per km^2. Additionally, the European Drone Outlook Study [3] estimates a total of 400.000 drones operating commercially in Europe by 2050. Because these numbers and the resulting air traffic densities are too large for human air traffic controllers to be managed, the Federal Aviation Administration (FAA) and the International Civil Aviation Organisation (ICAO) have required that these drones are capable of avoiding each other without human control [4]. This steers the airspace away from centralized Air Traffic Management (ATM), into the direction of decentralized ATM or the Free Flight concept, which has been shown to potentially increase airspace capacity [5]. In order to facilitate these air traffic densities and self-separation requirements, the Metropolis project researched the influence of airspace structure on the intrinsic safety and efficiency of the airspace. From this research, it was found that an airspace structured in vertical layers with a predefined heading range, resulted in the highest airspace capacity with limited negative effects on the overall flight efficiency [6]. This capacity increasing effect can be attributed to both separation and alignment of aircraft, which are both shown to increase the intrinsic safety of an airspace [7]. Other studies however indicated that climbing and descending aircraft did not benefit from this separation and alignment and therefore have a higher chance of being in conflict than cruising aircraft. It was even found that the majority of conflicts between aircraft in the airspace contained at least one aircraft that was conducting vertical maneuvers [8]. To satisfy the requirements of the FAA and ICAO it is important that these conflicts get resolved by the aircraft involved, or even better, that these conflicts are prevented.

Conventional Conflict Detection & Resolution (CDR) methods could be used for resolving the conflicts, however as traffic density increases the chance that resolving one conflict leads to a new conflict increases. This can lead to emergent behavior that is very complex to predict with analytical models. Because Deep Reinforcement Learning (DRL) and specifically Deep Neural Networks (DNN) have shown to perform well on pattern recognition tasks, it is decided to research the capabilities of Deep Reinforcement Learning for self-separation of vertically maneuvering aircraft in the layered airspace. As this is a new approach, it will first be researched if DRL can be used for self-separating without this emergent behavior, before introducing this domino effect of subsequent conflict resolutions, gradually increasing complexity. The field of DRL has seen incredible advancements over the last couple of years. In 2013 it was demonstrated that DRL is capable of learning how to play Atari games from raw pixel inputs [9]. Only six years later, in 2019, researchers from OpenAI FIVE managed to beat the Dota2[1] world champions in a best of 3 matches with a sophisticated DRL model [10]. This leap in performance shows how fast DRL is developing. Furthermore, it is shown that DRL can be used to train self-driving cars to safely conduct lane change maneuvers in a simulation environment [11–13], a problem that can be seen as a 2D version of this 3D vertical maneuvering problem. Previous research has shown that DRL can tune the parameters of existing conflict resolution methods on a case-by-case basis to increase the safety [14]. In this research however the aircraft will be directly controlled by the DRL model, similarly to the lane change cases.

1.2. Objective and Research Question

Previous works have shown that a relatively large number of the intrusions and conflicts that arise for the altitude layer airspace are between climbing/descending aircraft and cruising aircraft. This research will look at the specific case of using DRL to find safe vertical maneuvers to reduce the number of intrusions. The research objective is formulated as:

"Add to the current knowledge in the field of high density airspace safety by researching the performance and behavior of a Deep Reinforcement Learning agent tasked with conducting vertical maneuvers within the altitude layered airspace as safely as possible."

From this research objective, a main research question is formulated that will help to guide the research. This research question is:

"How much can the detrimental effect of vertical maneuvers in the altitude layered airspace on the safety be reduced when using Deep Reinforcement Learning over methods previously used for the vertical maneuvers?"

[1]A fast-paced 5v5 real-time strategy game within a continuous environment, having perceptually an infinite number of states to consider, ranging from different characters, items and location of the different agents.

1.3. Research Approach

To successfully answer the research question and increase the probability of achieving the goal of the research a variety of research activities, each with their own sub-questions, have been defined. First, these different research activities will be further elaborated, then in the subsequent section, the relation between these activities will be shown in the form of a research framework.

1.3.1. Research Activities

This section will describe the different research activities followed by an explanation of why these activities are conducted and what questions should be answered after the completion of these activities.

Research Activity 1
Research the different Deep Reinforcement Learning algorithms available.

Many different DRL algorithms have been developed in recent years. In order to correctly implement them, it is important to research the strengths and weaknesses of the different available algorithms and select a potential algorithm that is suited for the problem. Furthermore, it is beneficial to be aware of the potential pitfalls of the selected algorithm such that corresponding contingency measures can be taken.

(a) *Which DRL algorithm would be suited for the problem at hand?*
(b) *What are potential pitfalls of the selected algorithm and how can they be mitigated?*

Research Activity 2
Define the concepts of the thesis and the assumptions/limitations.

Properly defining the boundaries of the research is important for enhancing the overall feasibility. Furthermore, well-defined concepts can ensure that the obtained results can be related to previous studies.

(a) *Which parameters of the layered airspace affect the intrusion rate of vertically maneuvering aircraft?*
(b) *Which assumptions can be made about the airspace without invalidating the research?*
(c) *How should the research concepts be defined such that it covers a broad area of relevance, whilst remaining specific?*

Research Activity 3
Define the problem such that it can be solved by Deep Reinforcement Learning.

DRL is a strong tool if used for the correct problems, accurately defining the problem however does require some additional research. To make the problem solvable by DRL, it must be formulated as a so-called Markov Decision Process (MDP). During this research activity, the different elements of an MDP will be researched and the overall problem will be formulated as an MDP. Furthermore, it will be researched if formulating the problem as a pure MDP is the best approach or if there are elements that can be added to increase potential performance or convergence.

(a) *How should each element of the Markov Decision Process be designed?*
(b) *How much information should be included in the state representation, and how should this be presented, such that the agent is capable of making safe decisions?*
(c) *How to correctly implement safety in the reward function?*
(d) *Are there additional strategies that can be used to enhance the pure Markov Decision Process?*

Research Activity 4
Design the experiments by combining the established concepts and the formulated MDP.

During this research activity the experiments will be designed. These experiments will be formulated such that each experiment's outcome contributes to answering the defined research question. Furthermore, the experiments will focus more on obtaining a better insight on how different elements affect the safety of the maneuvers, rather than optimizing the safety based on a specific set of maneuvers. It is expected that this will contribute more to the thesis objective of adding to the current knowledge than optimizing a specific behavior.

(a) *What is an appropriate setup of the experiments such that convergence is promoted?*
(b) *How to create unbiased conflict scenarios in order to correctly evaluate the models?*
(c) *What are the independent variables?*

Research Activity 5
Conduct the experiments defined during research activity 4 and analyze their results.

The experiments that have been defined will be conducted in BlueSky. After they have been conducted, the results will be analyzed to see if the hypotheses defined for the different experiments hold and if interesting results/behavior can be found.

(a) *How do the obtained results relate to the defined hypotheses?*
(b) *How do the different approaches relate to each other with respect to safety and behavior?*
(c) *What are the dependent variables?*
(d) *What performance metrics are relevant for assessing the performance of the proposed method?*
(e) *Is DRL with the current setup capable of improving the safety of vertical maneuvers?*

Research Activity 6
Based on the results from the experiments test the most promising approaches on a variety of different tasks and air traffic structures.

To evaluate the robustness of the developed model it will be extensively tested on a variety of tasks and traffic densities. Also, the model will be trained in an environment where other aircraft interact with each other through conflict resolution to research if DRL is capable of recognizing patterns in seemingly complex emergent behavior. After this research activity it should also be possible to provide an initial answer to the main research question.

(a) *How robust is the obtained model?*
(b) *What is the impact of different airspace structures and traffic densities on the different models?*
(c) *Is DRL able to identify and adapt to emergent behavior resulting from conflict resolution maneuvers of other aircraft?*
(d) *How do the solutions found by the DRL models compare to existing CDR methods?*

Research Activity 7
Perform extensive data analysis on the final results in order to evaluate performance and identify failure cases to write proper recommendations.

The final research activity is a data analysis based on the obtained results. This data analysis should provide additional insights beyond the general number of intrusions. For example, specific failure cases can be identified. The main goal of this research activity is to provide additional information for the discussion, conclusion and recommendations.

(a) *Are there specific failure cases in which the models cannot perform?*
(b) *Is there apparent behavior of the agents that can be identified, isolated and applied to existing CDR methods?*

1.3.2. Research Framework

How the different research activities relate to each other can be seen in Figure 1. In this figure it can also be seen that activities 1-4 have been conducted during the preliminary phase, the remaining activities are reserved for the main phase of the research. An exception for this is made for a Proof of Concept experiment, which is also included in the preliminary phase in order to demonstrate some initial capabilities. Furthermore, an iterative approach is included between phases 3, 4 and 5. Based on the results of the experiments it is possible that alterations have to be made to the MDPs or the experimental setup. Another possibility is that based on the results of the experiments it becomes interesting to conduct additional experiments to gain more understanding regarding certain results. Hence the iterative approach.

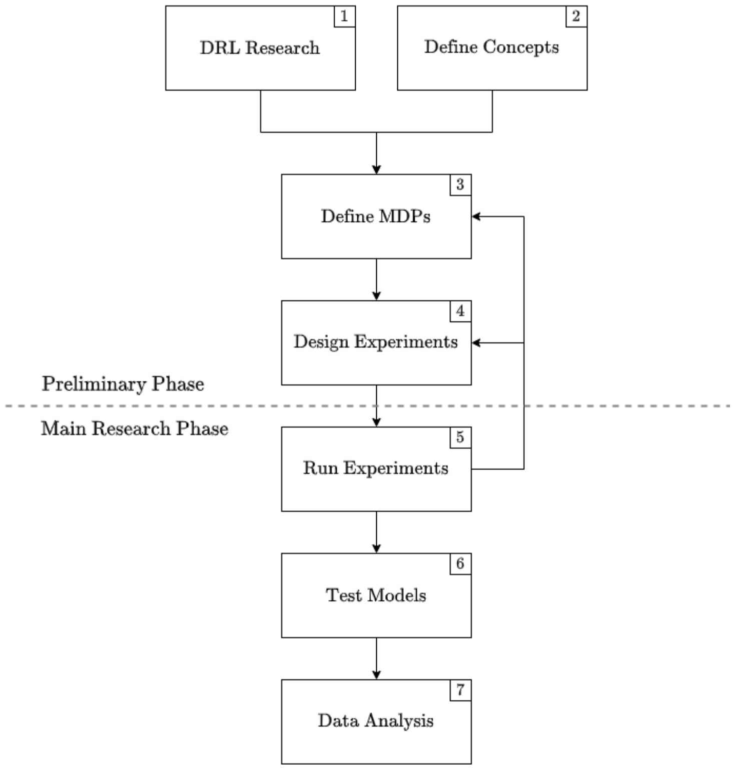


Fig. 1. Research framework established based on the defined research activities.

1.4. Research Scope

This section will provide the list assumptions that have been made in order to ensure that the research remains feasible. For a more extensive list on concepts and assumptions made, the reader is referred to Chapter 3.

Assumption 1
Only a single type of aircraft will be used throughout all of the experiments and simulations.

Using a single aircraft will simplify many elements of the research whilst keeping the more important elements such as airspace structure and interaction of the model with the environment. This way apparent results can not be attributed to the interaction of different aircraft types and better conclusions can be drawn from the research.

Assumption 2
Everything in the environment will be deterministic unless otherwise specified.

This assumption means that there will be no weather influences, rogue aircraft or sensor uncertainties. Because this research is mostly focused on identifying the capabilities of DRL under ideal circumstances, introducing stochasticity would make it difficult if not impossible to identify the effect of different parameters on the performance. Later studies could introduce stochastic behavior to evaluate the robustness and adaptability of the obtained models from this study.

Assumption 3
The altitude layers are are situated above static obstacles. (e.g buildings)

For the airspace it is assumed that there are no static obstacles within the airspace. This ensures that the agent is free to select the actions it wants without being limited by the environment. This is inline with the airspace of the research of Sunil et al. [6].

Assumption 4
Aircraft cruising in their corresponding layer will not change heading or altitude unless otherwise specified.

Because it is assumed that there exist no static obstacles it is also assumed that aircraft will fly a direct route to their destination. Because of this no heading or altitude changes are required by aircraft in their cruise phase. This has as added benefit that the environment becomes more predictable for the DRL model.

Assumption 5
There is no delay between observation and action selection.

Once an agent observes an observation is it assumed that the selected actions follow immediately. This assumption simplifies the implementation of the overall experiment as no artificial lag between observation and execution has to be built in.

1.5. Outline of the Preliminary Report

The main purpose of this report is to provide the reader information with regards to the preliminary phase of this thesis. First, a summary of the conducted literature study will be given. Based on the knowledge obtained during this literature study the concepts and assumptions have been defined together with the experimental design. This is further elaborated in Chapters 3 and 4 respectively. The methodology that is to be used for the experiments during the main phase of this thesis is then given in Chapter 5. Chapter 6 will describe the setup and results of the proof of concept experiment, which has been conducted during the preliminary phase of this research. Some foreseeable problems (based on literature and initial experiments) will then be described in Chapter 7 together with potential contingency measures. Finally, Chapter 8 will contain the conclusion of this report.

2. Literature Study

This chapter will go over the relevant literature for this report and provide some background information for the used methodologies regarding Deep Reinforcement Learning (DRL). First, the altitude layered airspace and its concepts will be introduced, as they provide the foundation for the research scenarios. Then in the subsequent sections, a more theoretical foundation for Markov Decision processes (MDPs) and Deep Reinforcement Learning as a potential solution method will be given. Finally, some applications of DRL will be shown to serve as a proof of concept and as inspiration for the methodology of this research.

2.1. High Traffic Density Airspaces

Recent years have seen increased interest in the usage of drones for commercial purposes, with multiple companies researching the possibilities of drone-based deliveries [1, 2]. Estimates for the city of Paris range from 110,224 to 275,599 drones based operations occurring per hour by 2035 (approximately 40 - 100 movements / km^2) [15]. This leads to air traffic densities unprecedented by today's standards. In order to mitigate the problems that potentially arise from these traffic densities, multiple projects, including the Metropolis project, are researching potential solutions in the form of novel airspace design concepts with a focus on high capacity and intrinsic safety [6].

2.1.1. Metropolis

The Metropolis project is a research initiative funded by the European Unions Seventh Framework Program [16]. The project is mentioned to have two distinguishable goals: "Exploratory research into future urban airspace design" and "Provide a better understanding of air traffic by looking at extreme scenarios". In order to achieve the goals, airspace concepts that differ significantly from today's standards are analyzed for unprecedented traffic densities, complexities and constraints. Four different types of concepts of increasing complexity and restrictions have been defined for further research; Full-Mix, Layers, Zones and Tubes [17]. A graphical representation of the four different concepts is given in Figure 2. Currently, most knowledge with regards to ATM is based around manned aviation, at lower traffic densities, which is separated in sectors in order to reduce the workload of the Air Traffic Controllers (ATCo). For drones or Personal Air Vehicles (PAV) with decentralized air traffic control, this benefit of sectors and the standard of traffic densities no longer holds, validating the need to look at different ways of separating aircraft.

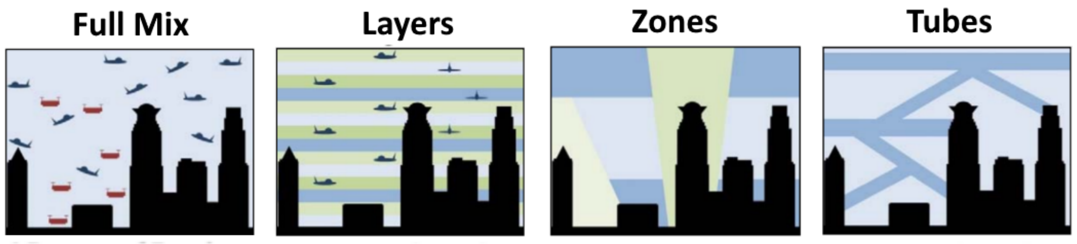


Fig. 2. Illustration of the four different concepts as defined for the Metropolis project, image adapted from [7]

2.1.2. Altitude Layer Airspace

One of the potential solutions from the Metropolis project, scoring both high on efficiency and safety metrics, is the Layers, or altitude layer airspace concept [6, 18]. Within the altitude layer airspace, the airspace is divided into multiple vertical layers with a predefined allowed heading range for each layer. It is to some extent an extension of the already established hemispherical rule, which separates east and westbound traffic based on altitude [19]. In the research by Sunil et al. two sets of eight layers are used to cover the entire heading range (0 - 360°), leading to a total of 16 layers with 45 degrees of heading each. The upper set of layers are being used for long-distance travel and the lower layers reserved for short-distance commerce [6]. An illustration of this is given in Figure 3.

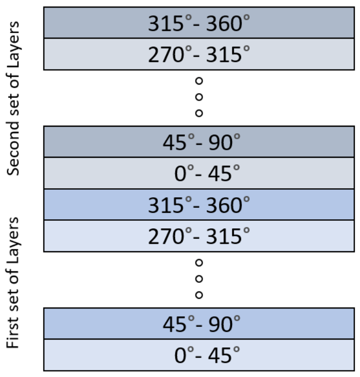


Fig. 3. Graphical representation of the layer airspace as used in the works by Sunil on Metropolis [6]

An explanation for why the altitude layer airspace concept scores high on the safety metrics is given in the work on Geovectoring by Hoekstra et al.[7]. In essence, two main elements can be distinguished that reduce the complexity of air traffic, resulting in higher intrinsic safety. Segmentation, which reduces the conflict rate by decreasing the number of possible conflict pairs and traffic density for each segment (adding layers). And alignment, which decreases the relative velocity by introducing homogeneity in the velocity vectors which in return has been shown to decrease the conflict rate (similar heading). In multiple works regarding the altitude layers airspace, it is found that a large portion of the intrusions[1] that occur are between cruising and climbing/descending aircraft [6, 8, 20, 21]. In most referred works, this can also be attributed to the lack of conflict prevention/resolution rules during transition between layers. For example, in the work by Sunil, climbing and descending aircraft are not obliged to adhere to the heading rules, it is possible that these conflicts[2] therefore can partially be attributed to the violation of segmentation and alignment.

[1]An intrusion, or Loss of Separation (LoS), occurs when two aircraft are within the vertical and horizontal safety margins of each other.

[2]A conflict is a predicted intrusion or LoS within a set look-ahead time.

2.1.3. Altitude Layer Airspace Capacity and Safety

Hoekstra et al. [7] and Sunil et al. [22] analyzed the effect of layers on the airspace capacity. Both works showed that there is a negative correlation between conflict rate and allowed heading range per layer, e.g. for smaller heading ranges the capacity increases. Sunil however also looked at the Domino Effect Parameter (DEP), a parameters that measures the increase in total conflicts when using conflict resolution (CR) versus without CR, e.g. the number of secondary conflicts arising from CR maneuvers [23]. It was found that the DEP increases when reducing the heading range below 180°, Sunil argues that this is a potential mismatch of the used CR method and the layered airspace, and that better tailored CR methods should be researched for the layered airspace. Ribeiro et al. also showed that there is a correlation between the airspace capacity and selected CR method [24], backing up the hypothesis of Sunil.

In their work on modeling of the intrinsic safety of layered and unstructured airspaces, Tra et al. [25] showed comparable results to Hoekstra and Sunil for the reduction in conflict rate when not using any CR methods. Furthermore, Tra also investigated the effect of vertical maneuvers and velocity changes on the total conflict rate. It was shown that vertical maneuvers have a detrimental effect on the safety of the airspace, it was also demonstrated that this detrimental effect worsened for slower climb rates. For the velocity changes, no significant differences in conflict rate were found when allowing aircraft to fly between 450-550 kts when compared to a standard velocity of 500 kts, suggesting that allowing changes in the horizontal velocity has less of an impact on the intrinsic safety of the airspace.

Sunil et al. [8] also showed that slower climb rates lead to a higher total number of conflicts, reporting an increase of more than x2.5 the amount of conflicts when reducing the flightpath angle from 5.6° to 1.4°.

2.1.4. Presence of Static Obstacles

The works of Sunil on the altitude layer airspace assumed that the air traffic was conducting all of the cruise-related maneuvers above buildings. This effectively eliminates the presence of static obstacles, which is a valid assumptions for many European cities where high-rise is either limited due to regulations (e.g. Paris) or soil conditions (e.g. Amsterdam) [26]. In cities such as Hong Kong where high-rise buildings are the preferred solution for housing, the assumption of flying above the building might not be valid. In these cases, it might be better to have the air traffic following the already existing road infrastructure, as is done in the works by Doole and Ribeiro [20, 21].

In his work, Doole demonstrated that even in the presence of static obstacles, separation of the traffic to altitude layers based on the cardinal directions (North, East, South and West) improved the safety without (noticeably) affecting the efficiency and throughput of the airspace [21]. This result is also found in the works by Ribeiro [20].

Apart from the altitude layers, Ribeiro also implemented so-called transition layers, shown in figure 4. These transition layers were constructed in order to promote safer vertical maneuvers, allowing heading and velocity changes within these layers, before merging with the cruise layers. Nevertheless, it was still found that a majority of conflicts and intrusions that arose were between the cruising and vertically maneuvering aircraft.

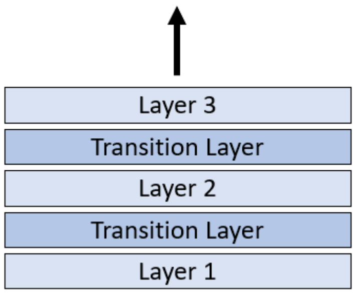


Fig. 4. General depiction of the transition layers as used by Ribeiro [20]

2.1.5. Emergent Behaviour

When applying traditional conflict resolution methods, proven to be successful for manned aviation at lower traffic densities, to higher or extreme traffic densities, emergent behaviour might occur that has an unpredictable and sometimes destabilizing effect on the airspace [14]. It is not uncommon for emergent behaviour to follow from a simple set of rules once the number of elements to which these rules apply increases, perhaps the most famous example being Conway's Game of Life [27]. Ribeiro et al. stated that DRL might potentially find patterns in this emergent behaviour and determine optimal strategies accordingly. DRL has not been extensively researched for safe maneuvers of aircraft, but it has been demonstrated to successfully outperform rule-based methods in overtaking and lane-changing tasks for cars on freeways [11, 28]. As adjacent layers in the layer airspace have a relatively comparable heading range, changing layers can be seen as a 3D version of the 2D lane changing problem. Therefore it is decided to further research the capabilities of DRL for the problem at hand.

2.2. Markov Decision Processes

A Markov Decision Process (MDP) is a mathematical process that can be used as a framework for decision making. It can be broken down into two elements, the environment and the agent (decision maker). In principle a process can be modeled as an MDP if there is a (finite) state-space $\mathcal{S}$, a (finite) action space $\mathcal{A}$, a (probabilistic) state transition function P and a reward function R that provides a reward based on this state transition [29].

The goal of the agent is then to find a strategy (policy, see section 2.2.3) that selects an action a_t, given a current state s_t such that the expected reward over time is maximized.

2.2.1. Environment

Because an MDP is completely described by the tuple $\langle \mathcal{S}, \mathcal{A}, P, R \rangle$, it is important to understand these elements before designing an MDP.

The **state representation** at time t s_t of the environment fully describes the system and contains all information required for choosing the optimal action a_t. Previous states or the previous action that led to this state is not relevant (comparable to a chessboard). If this is the case, an MDP is said to have the Markov Property.

The **action space** is the set of actions that can be selected, given a state s_t. The action space can therefore be dependent on the state. In classical MDPs the action space is discrete and only one of the available actions from the action space is selected.

The **state transition function** P is a (probabilistic) function that maps any selected action $a_t \in \mathcal{A}$ for a given state $s_t \in \mathcal{S}$ to a new state $s_{t+1} \in \mathcal{S}$. This function fully describes the dynamics of the system and is given in equation 2.1.

$$P\left(s', s, a\right) = \Pr\left\{S_{t+1} = s' \mid S_t = s, A_t = a\right\} \tag{2.1}$$

Finally, the **reward function** R (equation 2.2) calculates the corresponding reward, r, for a given state s, action a and new state s'. The reward function rewards certain actions and punishes others depending on the state and new state corresponding to that action. It therefore has a direct influence on the policy of the agent and is one of the most important elements when it comes to the agent achieving the desired behaviour [30], for more information on reward function design see section 2.2.2.

$$r_{t+1} = R\left(S_{t+1} = s', A_t = a, S_t = s\right) \tag{2.2}$$

2.2.2. Reward Function Design

The reward function defines the reward the system obtains after each timestep. As the goal of the agent is to maximize the sum of the rewards obtained over the entire episode [3], it is important to properly design a reward function. An incorrect reward function can lead to sub-optimal behavior or even cause the agent to never achieve the intended goal at all.

Barto and Sutton [30] argue that it is important that the reward function only specifies what the agent should be doing, and not how it should be done. For example in the game of chess rewards should only be given for winning and penalties for losing, no penalties or rewards for losing and taking pieces respectively. This is to prevent introduction of human biases into the reward function.

This approach can however lead to a so-called plateau problem, first named by Minsky in 1961 [31]. This plateau problem is characterized by a reward function that is so sparse that the agent might wander around aimlessly for an extensive period of time without receiving any rewards. One way to counter this problem is by shaping the reward, essentially giving small rewards for steps in the right direction. Although this contradicts the initial statement by Barto and Sutton, it is shown that it can substantially reduce training time [32]. They do however acknowledge that it is important to realize the impact this shaping can have on the learned policy.

Mataric [33] argues that there are 2 main ways to shape a reward function. The first way is to define multiple goals that should be achieved and giving rewards when each sub-goal is achieved. The second method is by utilizing so-called progress estimators, which gives a small reward every time a step is taken in the correct direction. In order to ensure that this progress

[3] An episode is defined as the entire set of state transitions, starting at an initial state and ending when a terminal state has been obtained

estimator is not abused by the system, it is important to ensure at least a zero-sum scenario where negative progress is penalized equally or harder.

2.2.3. Agents, Policies and Value Functions

Section 2.2.1 elaborated on all the elements that define the environment. The other important element of an MDP is the so-called agent or decision-maker. The interaction between this agent and the environment is graphically depicted in figure 5.

The goal of the agent is to learn a **policy** π that maps a state $s \in \mathcal{S}$ to an action $a \in \mathcal{A}$ such that the sum of rewards r (also known as the **return** G_t) is maximized over time. The most common mathematical definition for the return G_t is given in equation 2.3, where $0 \leq \gamma < 1$ is called the discount factor and describes the importance of immediate rewards versus future rewards. This discount factor also bounds the return such that it cannot divert to infinity [30].

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{k+t+1} \tag{2.3}$$

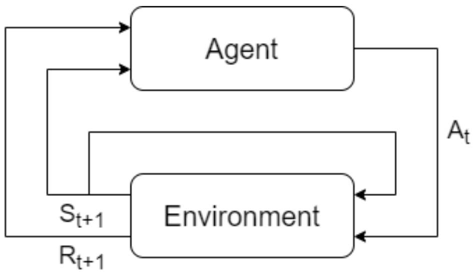


Fig. 5. Flowchart of the interaction between agent and environment for a standard MDP at time t, adapted from [30]

To learn from experiences and improve the policy, value functions are used. There are two main types of value functions: *state value functions* and *state-action value functions*, also known as the Q-value function. The state value function outputs the expected return given a state s following policy π (equation 2.4). The state-action value function outputs the expected return given a state s, action a and following a policy π (equation 2.5). These value functions can then be used to evaluate policies and see if the expected return increases for a new policy π [30].

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s \right] \tag{2.4}$$

$$q_\pi(s, a) = \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right] \tag{2.5}$$

Multiple Agents

There exists a specific subset of MDPs that deals with processes where multiple agents interact, and sometimes learn, independently from each other in the same environment. This is called Multi Agent MDPs (MAMDPs). A problem that often arises with MAMDPs is the fact that the environment seemingly changes with the agent, which can result in previously learned behavior becoming obsolete, leading to convergence problems [34]. Nevertheless successes have also been found in multi-agent domains, for example for drone swarming, demonstrating that they can learn emergent behavior, even if this behavior changes over time [35, 36].

2.2.4. Partial Observability

In the previous sections it is assumed that the process is fully observable, e.g all the state elements are available to the agent and that the state is bounded in size. In many applications this is not the case however, and additional steps are to be considered in order to make the process more 'Markovian'. These problems are called Partially Observable Markov Decision Processes (POMDPs).

A common approach for dealing with partial observability is by including historical observations in the state representation which has been demonstrated to make the problems more Markovian [9, 37]. This is done by either including the entire history of state observations or by including only the kth most recent observations, also called the kth order history approach. Another approach to dealing with partial observability is by using function approximators (such as Deep Neural Networks) for the value functions [30].

2.3. Deep Reinforcement Learning - Theory

As stated in section 2.2.4, Deep Neural Networks (DNNs) can be used as a function approximator for the value functions in order to mitigate the detrimental effects of partial observability. This specific solution leads to a field within reinforcement learning called Deep Reinforcement Learning (DRL). Apart from being a potential solution to POMDPs, DRL has a variety of other benefits (and drawbacks) that will be discussed in this section. First, an introduction into DNNs and it's capabilities is given. This is followed by a discussion regarding discrete and continuous methods. Finally, the working principles of the Deep Deterministic Policy Gradient (DDPG) algorithm will be given, as it is the algorithm used throughout this research.

2.3.1. Deep Neural Networks

DNNs are a special sub-class of Artificial Neural Networks (ANNs) containing more than one hidden layer between the input and output layer of the network [38]. They have been shown to perform successfully at a wide variety of tasks ranging from image recognition and natural language processing to regression and classification tasks. A generalized structure of a DNN is shown in Figure 6.

Each neuron shown in Figure 6 multiplies its inputs with a weight vector, calculates the sum, adds a bias and feeds this value through an activation function. This value is then used as the output of a single neuron. This process is graphically depicted in Figure 7. Many different types of activation functions exist, for this research the tanh and Rectified Linear Unit (ReLU) are considered. The tanh activation function is used in the output layer in order to bound the outputs between -1 and 1 (Which can then be converted to the corresponding action values). The ReLU activation function is used in the other layers as it is shown to improve both stability and performance of DNNs [39].

DNNs learn by comparing their own output for a certain input to a target value. This is done with a so-called loss function. The loss computed is then fed backward through the DNN

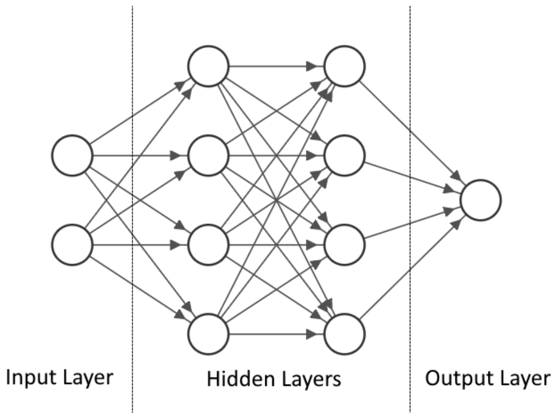


Fig. 6. DNN network architecture with 2 inputs, 1 output and 2 hidden layers of 4 neurons each.

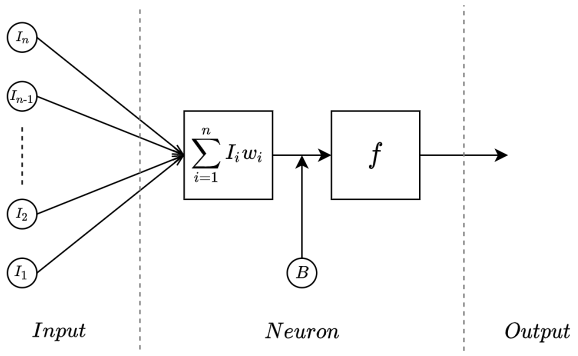


Fig. 7. Generalized depiction of the mathematical operations conducted in a single artificial neuron.

and the weights and biases are adjusted accordingly, this process is called back-propagation [40]. The rate at which the weights and biases are being updated is controlled by a parameter called the learn-rate [38].

One of the main advantages of using DNNs for reinforcement learning is their capability of generalizing complex states and therefore providing adequate outputs for states that they have never encountered before. A good example is given in the work of Mnih et al. where it was analyzed how different states of the game 'Space Invaders' would be valued by a DNN. They found that comparable, but not identical, states would be valued similarly [41]. An example of this is given in Figure 8.

This generalizing capability of DNNs also has the added benefit that during runtime computation time is slim, as it does not require going through an extensive look-up table containing all the possible states of the system. Instead, it only requires the computation of a system of linear equations and a set of activation functions.

A downside of DNNs however is the required training time before adequate performance is obtained. Even when combining a wide variety of advancements made in the area of DRL, Hessel et al [42] required an average of 18 million frames before reaching human-level performance in Atari games. This equates to roughly 80 hours of gameplay to achieve a performance level that a human is capable of achieving in a couple of minutes.

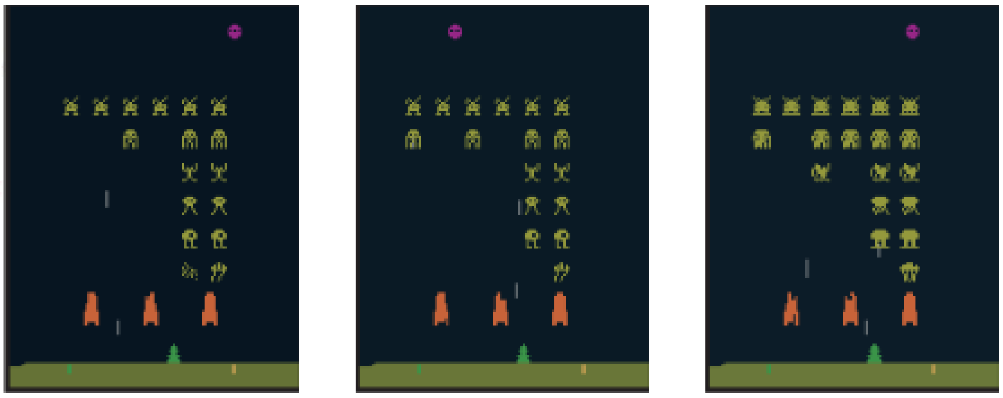


Fig. 8. Similar states of the game 'Space Invaders' being grouped together by a DNN, image retrieved from [41]

2.3.2. Discrete vs Continuous Action Spaces

Over the years many different DRL algorithms have been developed. A common distinguishing factor between these algorithms is whether they apply to discrete or continuous action spaces. For discrete action spaces, often DQN (introduced in [9]) or one of its derivatives is used. This is because these methods use argmax operations over the state-action values for all actions, making a discrete action selection process possible. For continuous action spaces, such operations would not work and therefore require discretization of the action space. This discretization however suffers from the "curse of dimensionality" and would quickly result in extremely large action spaces for increasing degrees of freedom.

For this reason, Lilicrap et al. [43] proposed the DDPG algorithm. This algorithm combines the advances of the DQN algorithm with the Deterministic Policy Gradient algorithm [44] (a method that works for continuous action spaces), resulting in an algorithm that can utilize the generalizing capabilities of the DQN algorithm in continuous action spaces. One of the problems of DDPG however, is that it suffers from instability issues, especially in environments with sparse rewards [45, 46]. Different algorithms for continuous action spaces do exist such as; SAC [47], A3C [48] and PPO [49], each with their own advantages and disadvantages over DDPG. Finding an optimal DRL algorithm however is deemed beyond the scope of this research, as the goal of the research is to research the potential of DRL for the problem at hand, rather than finding an optimal solution.

Because the control of a vehicle in high density scenario's is often a complex problem, it is assumed that allowing continuous control over parameters such as climb-rate and cruise velocity will allow for better tailored solutions. Therefore it is decided to use a continuous action space with the DDPG algorithm. The DDPG algorithm is chosen because it does not require a model of the environment, can learn from old experiences (or even artificially generated experiences) and has been successfully used for a variety of comparable control tasks [13, 14].

2.3.3. Deep Deterministic Policy Gradient

The Deep Deterministic Policy Gradient (DDPG) algorithm, first introduced by Lilicrap et al. in [43], combines the advances in deep learning with the Deterministic Policy Gradient algorithm, and is designed to work with continuous action spaces and high dimensional state-spaces. This section will provide the pseudo code to the algorithm and go over its most important elements. The algorithm is given in Algorithm 1.

Actors and Critics

DDPG is a so-called Actor-Critic method. This means that the DDPG agent maintains two networks, each with a distinguishable task. The actor network takes the state representation as input, and provides an action based on this state, hence the name actor. The critic network on the other hand learns to estimate the return based on a given state and a selected action (see section 2.2.3 on state-action value functions). It can in essence evaluate the selected action given a certain state, and is used during training for determining the gradient that is used for training the actor.

Replay Buffer

The replay buffer stores each state transition tuple, (s_t,a_t,r_t,s_{t+1}), in a fixed-length array M. Once the array is full, the oldest samples get discarded. During each step of training a random batch of size N is sampled from M. This allows the network to learn from uncorrelated data, which increases stability and improves performance [41].

Target Network and Target Values

The use of target networks to calculate the target values is a concept that was first introduced by Mnih et al. [9]. The target networks $Q'(s,a|\theta^{Q'})$ and $\mu'(s|\theta^{\mu'})$ are a copy of the critic and actor-network respectively. During training the target networks then follow a "soft-update" policy, given in equation 2.6, with respect to the learned networks. This ensures that the target values for a similar input do not deviate for each step and enhances the stability of the network. The target values are then calculated using equation 2.7. Here y_i is the expected return, given reward r_i, discount factor γ, state s_{i+1} and following the current policy of target actor $\mu'(s|\theta^{\mu'})$.

$$\begin{aligned} \theta^{Q'} &\leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \\ \theta^{\mu'} &\leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \end{aligned} \tag{2.6}$$

$$y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1} \mid \theta^{\mu'}\right) \mid \theta^{Q'}\right) \tag{2.7}$$

Ornstein Uhlenbeck Exploration Noise

A common problem with DRL is the exploration-exploitation dilemma[4]. In discrete action spaces, it is possible to use a greedy policy with a probability to select a suboptimal action for

[4]The exploration-exploitation dilemma is a common theme in Reinforcement Learning where there has to be a trade-off between the agent doing what it knows will work (exploitation) and searching for potential better solutions (exploration)

exploration, as is done in [9, 41]. For continuous action spaces this is however not possible. The DDPG algorithm tackles this problem by sampling noise from a noise process $\mathcal{N}$ and adding this noise to the selected action. For their implementation of the DDPG algorithm, Lillicrap et al. used an Ornstein Uhlenbeck (OU) noise process to sample temporary correlated noise. The noise is temporary correlated in order to aid with the exploration.

Algorithm 1: Deep Deterministic Policy Gradient, obtained from [43]

```
initialize critic $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ networks;
initialize target networks $Q'$ and $\mu'$ with the same weights as $Q$ and $\mu$;
initialize replay buffer $M$;
for episode = 1, M do
    initialize exploration noise $\mathcal{N}$;
    while $s$ != $s_{terminal}$ do
        do action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$;
        get reward $r_t$ and new state $s_{t+1}$;
        store ($s_t$,$a_t$,$r_t$,$s_{t+1}$) in $R$;
        sample a random batch ($s_i$,$a_i$,$r_i$,$s_{i+1}$) of size $N$ from $M$;
        calculate the target $y_i$ by using the target networks;
        update critic by minimizing the loss with respect to $y_i$;
        update the actor policy using the sampled policy gradient;
        update the target networks using the new actor and critic weights and
          "soft-target" parameter $\tau$
    end
end
```

2.4. Deep Reinforcement Learning - Applictations

Before designing, implementing and testing a novel DRL framework for vertical maneuvers in the altitude layer airspace, it is important to research current advancements in comparable fields of study. This section will provide some information on the current advances and highlight some solutions found in their research that can be beneficial for this research.

2.4.1. Lane Change Decision Making with DRL

Not many bodies of literature exist that address the vertical transitions between different layers within the altitude layer airspace. However, due to the alignment effect of the heading ranges between the layers, the problem can be approximated as a 3D version of the 2D lane change or on-ramp merging problem for autonomous cars, which has been researched more extensively.

Many different approaches have been used to solve this highway problem. For the action space both discrete [11, 12, 50] and continuous [13, 51] solutions were found. Sallab et al. researched the difference between a discrete (DQN) and continuous (DDPG) approach for the control of autonomous vehicles, they found that although the DQN algorithm converges faster, the overall performance is better when using DDPG [52]. The state representation for these different researches however differed more significantly, ranging from an array of state vectors [11, 13], to multiple boolean matrices of vehicle positions and velocity vectors [50] or even visual representations [12, 28]. This variety in state representations shows the generalizing capabilities of DRL, being able to learn successful policies from information, regardless of how this information is provided.

Hoel et al.[11] researched the effect of degrees of freedom in the action space of the agent on the overall performance. Two agents were trained on a lane-changing task. The first agent only controlled the lateral velocity (lane change) of the ego vehicle with the longitudinal velocities being controlled by a secondary algorithm. The second agent was free to control both the lateral and longitudinal velocity of the ego vehicle. It was found that agent 2 outperformed agent 1, at the cost of a slower convergence rate. Furthermore, they also demonstrated that consistency in the state-representation is important for both convergence rate and overall performance.

In their research on rule-based constraints, Wang Junjie et al. [12] demonstrated that using rule-based constraints can be used to learn correct policies when penalizing and correcting unsafe actions, ensuring safe maneuvers even during training. In the paper it is however not made clear if the intended or the corrected action is stored in the memory buffer for learning, which is essential for reproducibility. Furthermore, Wang Pin et al. showed that only penalizing, but not correcting, unsafe actions eventually lead to a robust and safe lane changing policy without limitations on the action space [13], demonstrating that the approach used by Wang Junjie et al might not be necessary.

Another interesting feature used in the research by Wang Junjue et al. is the use of two replay buffers. Whenever sparse rewards occurred these rewards would be stored in a secondary replay buffer. This essentially allowed the algorithm to sample sparse rewards more frequently, reducing the impact of sparse rewards on convergence and performance.

2.4.2. Conflict Resolution and Airspace Safety Research with DRL

Conflict resolution for aircraft has already been extensively studied, with many different methods being proposed. A comparative study by Kuchar & Yang in 2000 surveyed over 60 different conflict resolution methods [53]. Nevertheless, current developments in DRL have sparked new interest into the area of conflict resolution.

Brittain researched the capabilities of Multi-Agent Reinforcement Learning (MARL) for conflict resolution, researching if it is possible for two agents to implicitly learn how to cooperate for conflict resolution. He showed that the agents are capable of learning correct policies for speed-based conflict resolution in horizontal conflict cases [54, 55]. The second research from Brittain on this topic [54] utilized a complex state processing network, allowing variable length state representation, depending on the environment. Comparatively their first research used a fixed-length state vector which only included those aircraft with which a conflict can occur, discarding "useless" information [55]. When compared to the same case it is found that the model of the first study converges to a better policy in 7.500 episodes compared to 30.000 episodes of the second study. Again demonstrating the importance of consistency in the state representation, previously also mentioned by Hoel et al. [11].

In contrast to directly controlling the aircraft as done by Brittain, Ribeiro et al. [14] researched the potential of DRL controlling the resolution parameters of the Modified Voltage Potential (MVP) algorithm [56]. It was shown that allowing a DRL agent control the parameters of the look-ahead time and resolution methods to be used, allowed for tailored resolution advisories that resulted in fewer intrusions. In their research Ribeiro et al. trained the DRL agent on a moderate traffic density scenario and then tested the agent on a low, moderate and high traffic density scenarios. For the low and moderate scenarios better performance than the benchmark was found, this was not the case for the high density scenario. This behaviour might indicate that it is beneficial to train the model on the most difficult scenarios it can encounter.

In another research, Ribeiro et al. opted for a different approach with regards to DRL for improving airspace safety [20]. Instead of controlling aircraft maneuvers (either directly or indirectly) for conflict resolution, the agent controlled the speed limits in sectors of the airspace. Similar to the variable speed-limits already present on some highways, which have been shown to improve safety [57]. It was found that introducing variable speed-limits to high air-traffic density scenarios can increase the safety of the airspace, although diminishing returns on the effect on safety were found for density scenarios that differed from the scenario that the agent trained on.

3. Concepts and Assumptions

Before the experiments are introduced, it is important to specify the concepts and assumptions on which the experiments will be based. This chapter will introduce these concepts. First, the concepts and assumptions with regards to the aircraft will be given. Then, the airspace to be used throughout the experiments and simulations will be shown and finally, all parameters and assumptions regarding Conflict Detection & Resolution and Safety will be given. For a list of more general assumptions the reader is referred to Section 1.4, Research Scope.

3.1. Aircraft

In order to simplify the experimental scenarios, it is decided that only a single type of aircraft will be used throughout all the simulations. Furthermore, it is assumed that all the aircraft in the airspace that are not controlled by an agent will be flying at a constant velocity and will not change their heading during cruise. This has as an added benefit that it will not be required to provide information regarding aircraft type to the agent and their consistent flight path will help with making the environment more stationary[1] with respect to the agent. The selected aircraft model and its parameters is given in Table 1, and is based on the same model used by Ribeiro et al. [20].

Table 1. Parameters used for the aircraft model

Aircraft type:	DJI Mavic Pro
Speed:	(5 - 15) m/s
Vertical Speed:	(-5 - 5) m/s
Turn rate:	instantaneous
Acceleration:	instantaneous

It is decided to limit the speed of the aircraft to positive values of 5 to 15m/s, even though the selected model can theoretically fly backwards. This decision is made in order to make the outcomes of this research more relevant for future studies regarding fixed-wing aircraft and because it is likely that in a mixed environment of fixed-wing and rotor aircraft there will be a minimum imposed velocity for safety reasons.

The vertical speed is changed such that the maximum climb and descend velocities are equal. This is done to increase the symmetry in climb and descend scenarios, allowing the same policy to control the aircraft. This way a descend scenario for the agent is similar to a climb scenario. It is expected that this symmetry will aid in faster convergence of the agent's policy.

The range in distance and battery life of the aircraft are both assumed to be infinite. This assumption is made as including these elements in the current stage of the research is not relevant and would likely introduce a wide variety of complexities and limitations. Furthermore, the turn rate and acceleration of the aircraft are made artificially large such that they are not limiting the intended maneuvers of the agent. These assumptions are required to simplify the initial research. If it is found that the agent is capable of learning safe climb and descend behavior

[1]An environment is considered stationary when only actions the agent does alter the state of the environment, non-stationarity means that there is apparent behavior in the environment that the agent cannot explain based on its own actions. For example, if another aircraft changes its heading

using instantaneous response, e.g. there is no lag between the intended velocities and actual velocities due to model dynamics (acceleration, turn rate), future studies could investigate the effect of real aircraft dynamics on the performance of the agent.

3.2. Airspace

For the airspace it is assumed that the entire experimental airspace is a cylinder free of static obstacles, in order to justify this assumption, the airspace will be based around the city of Paris similar to Sunil et al. [6]. Flying over the buildings is a valid assumption for the city of Paris as there exist regulations that require all buildings around the center to be smaller than 37m tall [58]. Table 2 gives all the important parameters with regards to the airspace, later in this section, the individual parameters and how they were established are further explained.

Table 2. Main parameters of the airspace in which the experiments are conducted

Location:	Based on Paris
Radius:	3 km (1.62 NM)
Minimum Altitude:	200 ft
Maximum Altitude:	975 ft
Altitude per Layer	25 ft
Number of Heading Layers:	16
Number of Transition Layers:	15
Heading per Layer:	45

Radius

Determining the radius requires a trade-off between two elements: computing power and maneuvering freedom. Creating airspace that is too small will potentially result in the agent flying out of the environment unintentionally, negatively influencing the rate at which usable data for learning is gathered. On the other hand, increasing the airspace radius will result in an exponential increase in the number of aircraft required for the same air traffic densities, increasing required computational power.

From empirical experiments regarding number of aircraft and required computational power, an airspace radius of 3km was found to result in a proper trade-off between maneuvering freedom and required computational power. With a 3km radius, an aircraft with a horizontal velocity, V_h, of 10m/s and a vertical velocity,V_z, of 4m/s ($790 ft/min$) is capable of traversing all the layers roughly 10 times before leaving the environment. It is expected that this is sufficient and a proper trade-off between maneuvering freedom and required computational power. Furthermore, the empirical experiments demonstrated that using conflict detection for all aircraft in the airspace drastically increases the required computational power.

Number of layers & Heading per Layer:

The number of layers and heading is based on the same setup as used by Sunil [6], where 2 sets of 45° layers were used for the PAV experiments. The 45° per layer is deemed sufficient for this research as previous research by Sunil et al has shown that the total number of conflicts between cruising and climbing/descending aircraft is independent on the heading range, making the results of this research applicable to all heading ranges [8]. Additionally, the transition layers are added, in line with the work done by Ribeiro and Doole [20, 21]. This results in 16 cruise layers where the top 8 are used for long-distance travel, and 8 short distance layers, located at the bottom. Between these 16 layers, 15 transition layers will be in place for conducting the heading maneuvers. In these transition layers the agent can also temporarily reduce its vertical velocity to zero in order to wait if the next layer is densely occupied.

Altitude

The altitude bounds are based on the minimum height requirements for the city of Paris and the height of each layer. The minimum height requirements for the city of Paris were found to be 37m (120ft). Because the minimum altitude is actually a trivial decision (range of the aircraft is assumed infinite, hence there is no impact from air-density on the aircraft and wind is also assumed to be zero) it is decided to maintain the nearest hundred after 120ft as the minimum altitude, leading to a minimum altitude of 200ft. The maximum altitude then follows from the altitude per layer, 25ft, which is based on previous work by Doole et al [21]. Multiplying this altitude per layer with the total number of layers, 31 (16 altitude + 15 transition layers), an altitude range of 200ft - 975ft is obtained. Finally, it is assumed that all cruising aircraft will be flying at the mean altitude of its corresponding layer with a compliance rate of 100%.

3.3. Conflict Detection & Resolution and Safety

Previous research has shown that conflict resolution methods might influence the overall stability of the airspace [18]. In order to ensure that the airspace is not affected by other components than the policy of the agent(s), it is decided to not use any CR methods, especially during the initial experiments. This ensures that the results obtained can be completely attributed to the DRL model.

Disabling CR has the additional benefit that it increases the stationarity of the environment. If non-ownship aircraft were to interact with each other it is likely that the agent will not be able to explain this behavior, decreasing the convergence rate or making the agent's policy unstable.

Because no CR is going to be used, the selection of the look-ahead time becomes trivial. Nevertheless, it is still used for the performance metrics. Therefore a look-ahead time of 30 seconds is selected for the performance metrics, in line with [20] and [21]. Conflict detection will be done through a state-based approach[2]. For experiment 4, CR will only use speed based resolutions

Finally, the separation requirements are chosen to be 50m for horizontal separation, in line with [20, 59] and a vertical separation requirement of 25ft, equal to the height of a single layer is to be used initially as this has been found to work successfully by Doole et al [21].

[2]State-based conflict detection only looks at the current state of all the aircraft, it does not take intended future course changes or velocity alterations into account.

4. Experiment Design

An important element of all research is the setup and execution of the experiments. This chapter will go over all the experiments that are going to be conducted and the scenario in which these experiments will take place. First, the scenario will be explained in detail with respect to the setup and required air traffic densities. Then the different experiments and their main purpose. Finally, the hypotheses for the different experiments will be given. The methodology corresponding to these experiments will be given in the next chapter, Chapter 5

4.1. Experimental Scenarios

This section will first explain the main experimental scenario that will form the basis for all the experiments. After this scenario has been sketched and elaborated, the air traffic density for the scenarios and how they have been obtained will be explained.

4.1.1. Main Scenario

The main experimental scenario will simulate an airspace with a predefined delivery area. Once an aircraft crosses the border of the delivery zone it has a probability to receive an altitude command simulating deliveries and take-offs. Figure 9 provides a graphical representation of this delivery zone within the main experimental airspace. The radius of the delivery area is 2.5km. It is decided to use a delivery area in order to ensure a more homogeneous traffic density around the agent during the experiment, as giving an altitude command immediately after spawn will result in no aircraft being present around the aircraft outside the border of the environment.

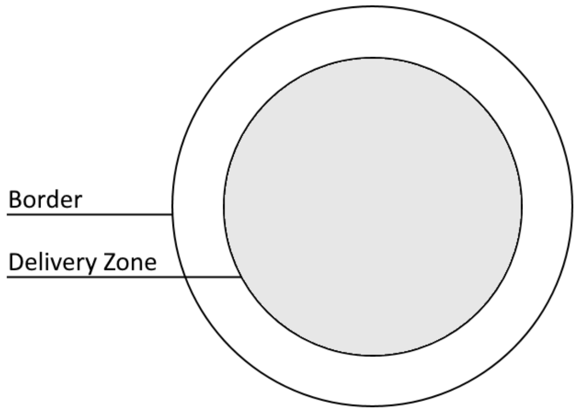


Fig. 9. Illustration of the environment border and delivery area, aircraft entering the delivery area will have a probability to obtain an altitude command, simulating delivery and take-off.

Aircraft will continuously be spawned on the border of the environment in order to maintain the required air traffic density. The spawn process of the aircraft is as follows; first the intended heading, ψ_i, is determined by means of a random integer, $\psi_i \in [0, 360]$, with a uniform probability for all values. Then a second random variable, $y \in [0, 1]$, will determine if the aircraft will be located in the top or bottom set of layers. The altitude of the aircraft can then be determined by means of equation 4.1. Note that the element $h_{layer}\left\lfloor\frac{\psi_i}{\psi_{layer}}\right\rfloor$ is multiplied by 2 in the equation in order to prevent spawning aircraft in the transition layers.

$$h_i = h_{min} + 2h_{layer}\left\lfloor\frac{\psi_i}{\psi_{layer}}\right\rfloor + \frac{y(h_{min}+h_{max})}{2} + \frac{h_{layer}}{2} \tag{4.1}$$

For the spawn location there is a uniform chance that the aircraft will be spawn at the border of the environment, given that the aircraft's intended heading will ensure that the aircraft enters the simulation area. This is shown in figure 10. This means that the bounds for the allowed spawn bearing with respect to the center of the environment are $[90+\psi_i, 270+\psi_i]$. Aircraft spawned outside of these bearing ranges would directly exit the environment, indicated by the invalid spawn areas in figure 10.

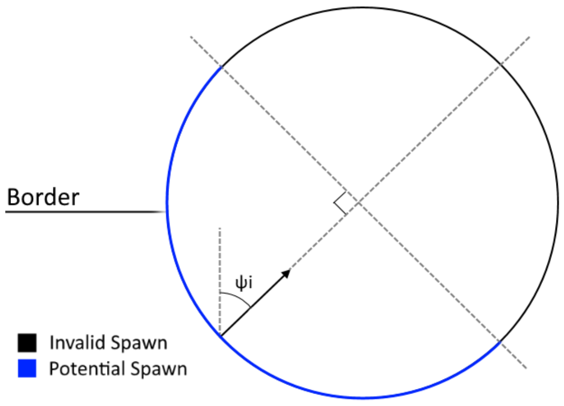


Fig. 10. Representation of the locations where an aircraft with intended heading ψ_i can spawn.

Once the required air traffic density is obtained, a setting period will follow to ensure that the traffic is uniformly distributed around the environment. After this setting period, aircraft entering the "Delivery Zone" (shown in figure 9) will have a predetermined probability, $P_{command}$, to either get a climb or descend command to a specified altitude. Aircraft existing in the top set of the layers can only obtain descend commands, simulating a delivery and consequently, aircraft residing in the bottom set of layers can only obtain climb commands, simulating a take-off to the long-distance travel layers. The climb and descend procedures followed after obtaining the command will be variable and is what will be researched in the specific experiments. For the experiments the value for $P_{command}$ is set to 5%. It is assumed that this will lead to a (local) ratio between cruising and climbing/descending aircraft, ϵ, of roughly 0.95. The probability of two vertically maneuvering aircraft intruding each other instead of a cruising aircraft, P_{vv}, can then be calculated by using the expression given in equation 4.2, adapted from [8]. For an ϵ of 0.95, this results in a probability of 5.3% that an intrusion of an agent will involve another climbing/descending aircraft or a probability of 94.7% that it involves a cruising aircraft. It is assumed that this value of 5.3% is low enough such that the problem will behave (mostly) as a single agent problem, increasing the convergence rate.

$$P_{vv} = 2\left(\frac{(1-\epsilon)^2}{4}\right) / \left(\frac{\epsilon-\epsilon^2}{2}\right) \tag{4.2}$$

Once an aircraft reaches its target layer, the aircraft will take on the mean heading and altitude of that layer and the horizontal velocity will be changed such that it is in line with the other cruising aircraft. All data gathered during the climb and descend will be stored in a memory file which is used during the performance metrics and data analysis parts of the experiments.

4.1.2. Air Traffic Density

When considering the air traffic densities it is important to consider the difficulty of the task and the frequency of intrusions. Reducing the air traffic density too much might result in a trivial solution by the DRL model where it will always climb as fast as possible because the chance for an intrusion is too small. This will also simultaneously lead to a memory buffer that contains almost no conflict/intrusion states, making learning to deal with these scenarios more difficult. It is therefore important to select the air traffic density such that the occurrence rate of these intrusions during random/trivial behavior is large enough to obtain notable representation in the memory buffer. Because no CR will be used throughout the experiments the air traffic density will not be limited by instabilities introduced through the domino effect [1].

In order to determine the appropriate air traffic densities, the average number of intrusions of a climbing aircraft for varying air traffic densities was determined from empirical experiments. In these experiments an aircraft flying in the lower set of heading layers is given a climb command of 400ft (mean expected altitude command) with a vertical velocity of 2.5m/s and 5.0m/s for both adhering and not adhering to the heading rules. The aircraft were cruising at a horizontal velocity of 10m/s. This experiment is run for a variety of air traffic densities. The average number of intrusions for the climbing aircraft is then observed for 1000 climb maneuvers which is used as an approximation of the LoS probability (PLoS / 400ft of climb). The results are seen in table 3. The results found are in line with the findings of Tra et al. [25], indicating a decrease in LoS probability at higher vertical velocities.

From these results the default, initial, traffic density is selected to be 15 AC/km2 (5̃0 AC/NM2). It is assumed that the resulting conflict rate will be sufficient for the model to learn from the intrusions whilst not being too difficult to solve. The rate of intrusions is still low, but counter measures such as memory buffer prioritization can be taken in order to artificially inflate the occurrence rates of these events.

Table 3. Effect of different traffic densities on the probability of LoS for multiple scenarios and ascend strategies.

AC/km2	PLoS Vz = 2.5m/s Heading Rules ⊠	PLoS Vz = 5.0m/s Heading Rules ⊠	PLoS Vz = 2.5m/s Heading Rules ☑	PLoS Vz = 5.0m/s Heading Rules ☑
5	2.0 %	1.2 %	1.4 %	1.1 %
7.5	4.3 %	2.4 %	2.6 %	2.3 %
10	5.7 %	3.8 %	4.0 %	3.4 %
15	8.5 %	6.3 %	8.2 %	5.9 %
20	10.6 %	8.3 %	9.0 %	7.2 %

[1]The domino effect occurs when resolving one conflict results in one or more additional secondary conflicts

4.2. Experiments

In total, three main experiments have been defined that will research the effect of degrees of freedom on the learned policy of the agent, shown in figure 11. In this figure the arrows indicate the movements that are controlled by the agent, all other movements are controlled by constant values. The different degrees of freedom controlled by the agent are also summarized in table 4. Apart from demonstrating the capabilities of DRL for learning safe climb and descend maneuvers, these different degrees of freedom might also provide additional understanding with regard to safe vertical maneuvers. After the main experiments, an additional experiment will be conducted to identify the capabilities of DRL with regards to recognizing complex emergent behavior

In this section, first a proof of concept experiment will be defined which will function as a basis for DRL with BlueSky. Then the three main experiments and their corresponding baseline will be shown. Finally the fourth experiment, analyzing DRL and emergent behavior will be explained. For all of the baseline models it is decided that the climb speed is constant and equal to the maximum climb speed as it was found that this results in the lowest conflict rate [8, 25]. This is done in order to prevent a trivial solution of the DDPG agent, such as always climbing at the maximum velocity (versus a baseline that flies at a lower climb speed), to outperform the baseline. Furthermore for all agents and baselines (except experiment 2b) it is decided to exempt the climbing and descending aircraft from the heading rules, in line with Sunil et al [6, 18]. This is also done as external changes to the agent will lead to non-stationarity and will result in inconsistent intruding aircraft state representation when using state-based conflict detection, both negatively influencing the models learning capabilities. For more information on the implementation of the MDPs for the different experiments see section 5.2.1.

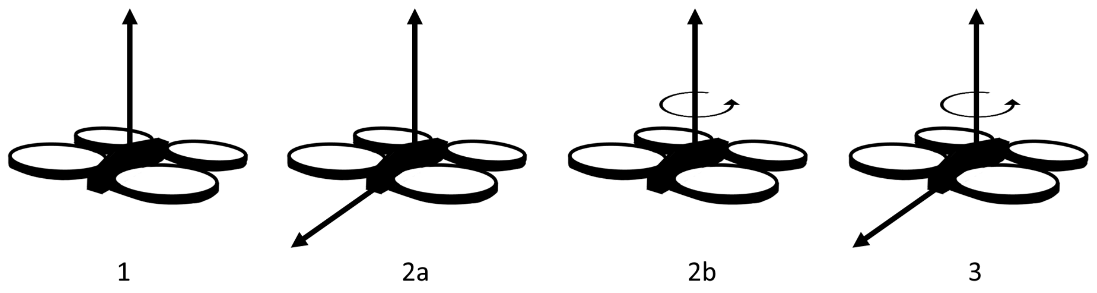


Fig. 11. Illustration of the different degrees of freedom for each of the experiments. (1) the agent only controls the vertical velocity of the drone, heading and horizontal velocity are constant. (2a) the agent controls the vertical and horizontal velocity of the drone, heading is constant. (2b) the agent controls the vertical velocity and heading of the drone, horizontal velocity is constant. (3) the agent controls all 3 degrees of freedom of the drone.

Table 4. Degrees of freedom controlled by the agent for the different experiments

Experiment	1	2a	2b	3
Vertical Velocity	x	x	x	x
Horizontal velocity		x		x
Heading			x	x

4.2.1. Proof of Concept (Climb & Decelerate)

The proof of concept experiment will serve as a baseline experiment for the DDPG algorithm to see if it is capable of controlling the aircraft towards the target state. In this experiment, the action-space and state representation will be identical to experiment 1. In this experiment, the agent will not be penalized for conflicts or intrusions, it will instead be penalized at each time step (e.g. the longer the agent takes to reach the target layer, the more penalties it will obtain) and should enter its target layer at a specified climb rate in order to receive the final reward. Its main function is to demonstrate whether or not the agent is capable of learning policies based on the information provided and if it is capable of discarding useless information.

Because the required policy (climb/descend fast and decelerate before reaching the target layer) is fairly straightforward forward it will be easy to verify if the agent is improving and approaching the intended behavior.

4.2.2. Experiment 0 (No penalties, just rewards)

Experiment 0 is comparable to the proof of concept experiment, but instead of having a reward function that stimulates a “go fast, decelerate fast” behavior, the same reward function will be used as for experiment 1, but without the penalties for intrusions. Because of this, the behaviour change of the agent due to penalties for intrusions can easily be observed through comparison with the behaviour of experiment 0.

4.2.3. Experiment 1 - One Degree of Freedom, Vertical Control

Experiment 1 will be the simplest experiment of the main experiments, with just a single degree of freedom, the climb speed. During this experiment, the agent obtains penalties for intrusions and rewards for reaching the target layer. The learned policy is then compared with experiment 0 in order to investigate if the agent changes policy due to potential intruders. In summary experiment 1 can be described by the following bullet points:

Baseline:

- Fly at maximum vertical velocity towards target layer
- Do not adhere to heading rules
- Constant horizontal velocity

DDPG:

- Free to choose vertical velocities between 0 & Vzmax in the direction of the target layer
- Simplest possible state representation
- Do not adhere to heading rules
- Constant horizontal velocity

4.2.4. Experiment 2 - Two Degrees of Freedom, Speed versus Heading Control

Experiment 2 will consist of 2 separate experiments, each with an additional degree of freedom over the vertical velocity selection.

In experiment 2a the agent will be allowed to control the horizontal velocity and vertical velocity. The state representation will be expanded such that decision-making with regards to horizontal velocity is better supported. Comparing the results of experiment 2a with experiment 1 will show if control over the horizontal velocity is beneficial. Furthermore, the results might show that there is a trivial solution (in terms of constant horizontal velocity) that limits the conflict rate during vertical maneuvers.

Baseline:

- Fly at maximum vertical velocity towards target layer
- Do not adhere to heading rules
- Constant horizontal velocity

DDPG:

- Free to choose vertical velocities between 0 & Vzmax in the direction of the target layer
- Free to choose horizontal velocities between Vmin & Vmax
- State representation expanded to support the horizontal velocity decision making
- Do not adhere to heading rules

Experiment 2b is comparable to 2a as it increases the degrees of freedom of the agent with respect to experiment 1. In this experiment, the agent also controls the heading or heading change of the agent. Similar to 2b the state representation will be adjusted accordingly. Comparing the results of this experiment with experiment 1 will provide insight into the importance of adhering to the heading rules during climb maneuvers, for example, if the agent will adhere to the heading rules without being told explicitly.

The baseline for this experiment will adhere to the heading rules in order to still allow for a fair comparison.

Baseline:

- Fly at constant vertical velocity towards target layer
- Constant horizontal velocity
- Adhere to heading rules

DDPG:

- Free to choose vertical velocities between 0 & Vzmax in the direction of the target layer
- Free to choose heading change for each time step
- State representation expanded to support the heading change decision making
- Constant horizontal velocity

4.2.5. Experiment 3 - Full Motion Control

In experiment 3 the agent will be given the full degrees of freedom with a corresponding state representation. If the model converges to a successful policy it will be interesting to compare the results to experiments 2a and 2b in order to see if the adapted strategies with regards to horizontal velocity changes and heading changes are comparable to the behavior learned in those experiments.

Baseline:

- Fly at constant vertical velocity towards target layer
- Constant horizontal velocity
- Adhere to heading rules

DDPG:

- Free to choose vertical velocities between 0 & Vzmax in the direction of the target layer
- Free to choose heading change for each time step
- Free to choose horizontal velocities between Vmin & Vmax
- Full state representation

4.2.6. Experiment 4 - Understanding Emergent Behavior

Experiment 4 will analyze the capabilities of a DDPG agent with regards to identifying and learning patterns in emergent behavior. For this experiment, the model with the best performance will be selected and tested (without additional training) in a scenario where conflict resolution is used for cruising aircraft. A new model with the same MDP as the aforementioned model will then be trained in simulations with conflict resolution in order to identify if the new model can outperform the other model.

4.3. Experimental Hypotheses

This section will describe the hypothesized outcome of the aforementioned experiments. Each hypothesis will be mentioned followed by the reasoning. In the hypotheses, each agent will be named after the number of its corresponding experiment

Hypothesis 1
Agent 1 will have fewer intrusions than agent 0

In experiment 1 the agent gets a negative reward for each intrusion. It is assumed that the agent is capable of learning a policy where at least in a fraction of the conflicts the agent can prevent intrusions by altering its vertical velocity in order to reduce the number of negative rewards obtained. This automatically leads to hypothesis 2.

Hypothesis 2
Agent 1 will have a lower average absolute vertical velocity than agent 0

If hypothesis 1 holds and agent 1 indeed resolves conflicts by altering its vertical velocity, it should result in a lower vertical velocity in comparison to agent 0. Because agent 0 only gets a reward for achieving the target layer it will likely prioritize getting there, resulting in a higher vertical velocity.

Hypothesis 3

Agent 2a will have a higher average absolute vertical velocity than agent 1

As previous research has demonstrated that having a higher vertical velocity will result in a lower conflict rate for vertical maneuvers [8, 25], and that random alterations in horizontal velocity have no noticeable detrimental effect on the conflict rate, it is assumed that agent 2a will prioritize a high vertical velocity combined with conflict resolution by means of horizontal speed variations. This will ultimately result in a higher average absolute vertical velocity when compared to agent 1.

Hypothesis 4
Agent 2b will adapt its heading to follow the heading rules of airspace

Hoekstra et al [7] demonstrated the importance of comparable heading on the relative velocity and resulting conflict rate. It is therefore assumed that agent 2b will also adopt the strategy of adhering to the heading rules during vertical maneuvers in order to reduce its relative velocity with other aircraft.

Hypothesis 5
Both agent 2a and 2b will outperform agent 1 in terms of intrusions at the
cost of a lower convergence rate

In the research on combined speed and lane change decision making by Hoel et al. [11], it was found that an agent capable of controlling both the lateral and longitudinal speed of a road vehicle outperformed an agent that only controlled the lateral velocity of the vehicle (the longitudinal velocity was controlled by a rule-based car-following model, preventing longitudinal collisions) at the cost of a lower convergence rate. This seemingly shows that increasing the degrees of freedom of the agent results in better and safer policies.

Hypothesis 6
Agent 3 will suffer from extremely slow convergence in comparison with the other experiments.

Because agent 3 receives an additional degree of freedom the total combination of possible actions increases exponentially. This will likely result in an increase in exploratory steps required before a properly trained state-action value function estimator (or Critic Network) can be obtained, decreasing the convergence rate.

5. Methodology

In this section, the methods used for conducting the experiments will be explained. First, the BlueSky Open Air Traffic Simulator will be introduced. Then the state representations and reward functions for the experiments, introduced in Section 4.2, will be formulated. Subsequently, the implementation and parameters of the used DDPG model will be given followed by the performance metrics that will be used in order to evaluate the performance of the different experiments.

5.1. BlueSky

In this research, the BlueSky Open Air Traffic Simulator will be used as the simulation platform for conducting all of the experiments. BlueSky[1] is an open-source air traffic control simulator developed by researchers of the TU Delft using the Python programming language [60]. It is capable of simulating a large number of aircraft and their interactions simultaneously and comes with a wide variety of additional features. Such as build-in CDR methods, data logging and a framework for easy implementation of self-written plugins. Making it an ideal candidate for ATM research such as this.

5.2. Markov Decision Processes

In order to apply DRL to the scenarios sketched in the experiments (Section 4.2), each scenario must be formulated as an MDP. In this section, the design of the MDP environment, previously described in Section 2.2.1, will be given for each of the experiments. First, the general considerations for the state representation will be given followed by the specific state representation used for each of the individual experiments. This will then also be done for the action space and the reward function. Note that the state transition function P is completely described by the dynamics of the system and therefore already implemented in BlueSky by default.

5.2.1. State Representation

The state representation of the environment is subdivided into 3 distinguishable categories, ownship, intruder and non-ship. All the elements that are considered for state representation will be given for their respective category including the experiment for which they are considered. The final resulting state vectors for each of the experiments will be provided at the end of this section. All of the elements in the state representation are formulated such that the climb and descend scenarios are as symmetrically as possible. As this is assumed to increase the convergence rate. Furthermore it is decided to use a fixed length state representation based on the results of Brittain et al. [54, 55], demonstrating that it leads to faster convergence and better performance. This requires the inclusion of a fixed number of aircraft ($N_{aircraft}$) in the state representation. The initial value for the experiments is $N_{aircraft} = 5$.

[1]download available via: https://github.com/ProfHoekstra/blueksy

Ownship

The main elements that are considered for inclusion into the ownship representation are the (absolute) vertical velocity (experiments 1, 2, 3), the horizontal velocity (2a, 3) and the current heading (2b, 3) (which is combined with the heading of the current layer to calculate the difference in heading with respect to the layer). The vertical and horizontal velocities are included such that the agent can evaluate the impact a certain action might have on the next state, e.g. changing the vertical velocity from 5m/s to 1m/s will have a higher impact than changing it from 2m/s to 1m/s.

Intruder

The representation elements of an intruder are dependent on the selected reference frame, two potential representation methods are provided in Figure 12. Although the same information can be determined from the elements provided in the two reference frames, certain reference frames facilitate specific resolution methods intrinsically. This would remove the need from the model to determine these relations.

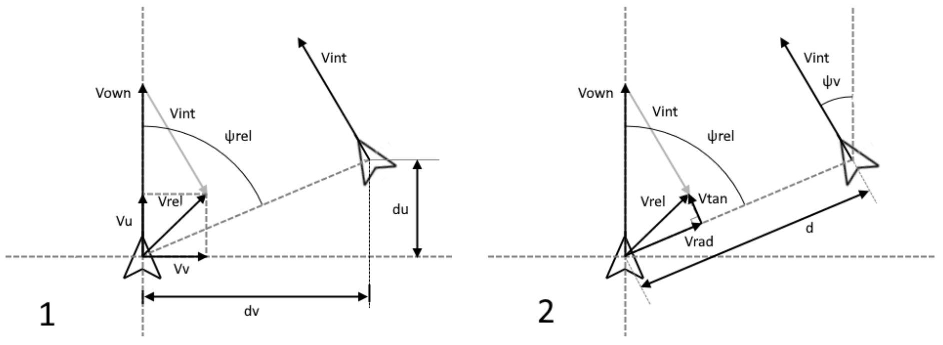


Fig. 12. Two ways to define the state representation of the intruder aircraft, (1) uses a Cartesian representation and (2) uses a Radial representation

Reference frame 1 is assumed to be better suited for horizontal velocity-based CR. For example, provided that $du > Pz_h$ where Pz_h is the horizontal safety margin, then a reduction of the ownships velocity by Vu will result in $Vu = 0$, resolving the conflict. Similarly, reference frame 2 is assumed to be better for heading-based CR, changing the heading with ψ_v will make $Vrad$ equal to zero, resolving the conflict (given that $d > Pz_h$).

All the elements included in the state representation can be seen in Table 5. In the table, Tcpa and TLoS stand for Time till closest point of approach and Time till Loss of Separation respectively. Tcpa is used if aircraft are not in conflict, otherwise, TLoS is used. Dcpa stands for Distance at closest point of approach. These values and their differences are graphically depicted in figure 13, here it can be seen that in the case of a conflict (the ownship flying into the horizontal protection zone of the intruder) TLoS is smaller than Tcpa. Hence why TLoS is included over Tcpa when available, it better represents urgency of action.

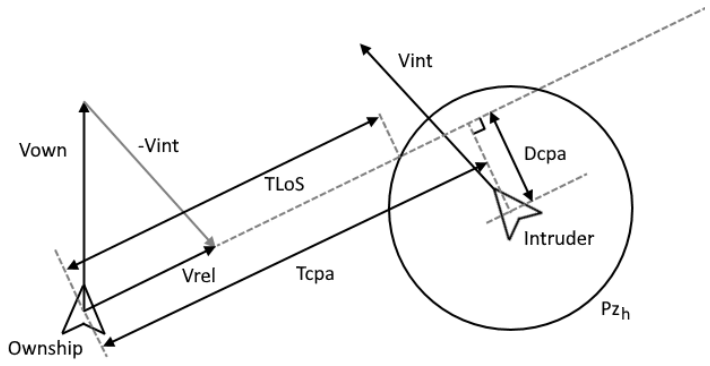


Fig. 13. Graphical representation of the elements Dcpa, Tcpa and TLoS. Note that in this figure Tcpa and TLoS are only an indication of their relative magnitude not a depiction of their actual value.

Table 5. Features of the intruding aircraft included in the state-representation for the different experiments

Parameter	1	2a	2b	3
Tcpa / TLoS	x	x	x	x
Dcpa	x	x	x	x
Conflict (boolean)	x	x	x	x
Vertical Distance	x	x	x	x
Longitudinal Distance		x		x
Relative Longitudinal Velocity		x		x
Relative Radial Velocity			x	x
Relative Heading			x	x
Relative Bearing				x
Distance				x

Intruder Inclusion

To enhance the consistency and relevancy of the state representation, a selection procedure is used to determine which aircraft are included in the state representation and in which order. First a vector containing all aircraft in a radius R_{look} around the ownship is created. ($R_{look} = 600m$, 30 seconds of lookahead combined with 2 times the mean horizontal velocity of 10m/s) Then, all aircraft that are in conflict with the ownship within this vector are included in the state, sorted for TLoS. If the number of aircraft in conflict is lower $N_{aircraft}$, the remaining elements will be filled with other aircraft sorted for Tcpa (Tcpa because if there is no conflict, there is no TLoS). The complete procedure can be found in Algorithm 2. Initially, it is decided to limit $N_{aircraft}$ in the state representation to 5, although this is subject to change depending on the results of the initial experiments. If not enough aircraft are available for inclusion in the state the remaining elements will be filled with zeros.

Algorithm 2: Sorting and Inclusion Algorithm used for Intruding Aircraft

```
create vector $\mathcal{A}$ containing all aircraft within radius $R_{look}$ around ownship;
create vector $\mathcal{C}$ of all aircraft in $\mathcal{A}$ that are in conflict with ownship;
sort $\mathcal{C}$ based on TLoS;
for intruder in $\mathcal{C}[0 : N_{aircraft}]$ do
    append corresponding information to the state vector;
end
if length($\mathcal{C}$) smaller than $N_{aircraft}$ then
    create vector $\mathcal{A}_2$ containing all aircraft in $\mathcal{A}$ and not in $\mathcal{C}$ that are in or between the
      target layer and the ownship layer;
    sort $\mathcal{A}_2$ based on Tcpa;
    while included aircraft smaller than $N_{aircraft}$ do
        append next aircraft information to the state vector;
    end
end
```

Non-Ship

The non-ship state representation contains all relevant information that cannot be attributed to either the ownship or interaction from the ownship with intruders. For now, only two features are considered: height difference to the target layer (experiments 1, 2, 3) and the heading difference with respect to the mean heading of the current layer (experiments 2b, 3).

Total State Vector

The total resulting state vectors for experiments 1-3 can be found in Table 6. The total size of the state vectors, including 5 intruding aircraft, for experiments 1, 2a, 2b and 3 are 22, 33, 33 and 54 elements respectively.

5.2.2. Action Space

The action space for the different experiments is defined by the degrees of freedom for each experiment and limited by the performance parameters of the selected aircraft model (see Section 3.1). The action spaces for all the experiments are provided in Table 7 and is independent of the state vector (unsafe actions or actions that result in the agent flying out of bounds can be performed).

5.2.3. Reward Function

In order to accurately assess the performance differences between the different experiments, it is decided that the reward function for all of the experiments is identical. If however convergence or performance issues are discovered for a specific experiment, reward shaping can be introduced (see the chapter on Potential Problems and Contingency Measures, chapter 7).

The reward function consists of 2 elements, a positive reward of 1 for reaching the target layer, a negative reward of -5 for having an intrusion (larger in magnitude due to the fact that it

Table 6. The resulting state vector for the different experiments.

Parameter	1	2a	2b	3
Vertical Velocity	x	x	x	x
Horizontal Velocity		x		x
Heading Difference with Mean Heading current Layer			x	x
Height Difference with Target Layer	x	x	x	x
x $N_{aircraft}$				
Tcpa / TLoS	x	x	x	x
Dcpa	x	x	x	x
Conflict (boolean)	x	x	x	x
Vertical Distance	x	x	x	x
Longitudinal Distance		x		x
Relative Longitudinal Velocity		x		x
Relative Radial Velocity			x	x
Relative Heading			x	x
Relative Bearing				x
Distance				x

Table 7. The action space for the different experiments.

Action	1	2a	2b	3
Vertical Velocity (0-5 m/s)	x	x	x	x
Horizontal Velocity (5-15 m/s)		x		x
Heading Change (-45°, 45°)			x	x

is awarded less frequently) and 0 otherwise, equation 5.1. This is in line with the reward design philosophy of Sutton and Barto [30]. It is decided to not penalize conflicts, as previous work by Ribeiro et al. demonstrated that an agent avoiding conflicts does not necessarily result in fewer intrusions [14].

$$r_i = \begin{cases} 1 & \text{Aircraft in Target Layer} \\ -5 & \text{Aircraft in Intrusion} \\ 0 & \text{Otherwise} \end{cases} \tag{5.1}$$

5.2.4. Agent

The goal of the agent is to maximize the total obtained reward (determined by the reward function) per episode, also called the return. An episode for the given environment is defined to start when a drone obtains the initial altitude change command, and ends when the drone reaches the target layer corresponding to this altitude change. Each drone is said to be controlled by an agent which observes the environment and selects appropriate actions. This action selection is done every 1.5 seconds[2] by all the agents simultaneously. If it is not the first time a specific agent is called, then also the old state, action and reward corresponding to this agent are observed

[2] 1.5 seconds is selected because at the maximum vertical velocity of 5 m/s the aircraft traverses 24.6 ft every 1.5 seconds, ensuring that the model provides an action to the aircraft at least every layer.

and stored in the replay buffer. This replay buffer can be accessed by all agents and is used by the overarching model to learn and determine the policy that all of the agents have to follow.

5.3. Deep Deterministic Policy Gradient

This section will provide information on the implementation of the DDPG algorithm. First, the general implementation of the DDPG algorithm in Python and the integration with BlueSky will be given, then the (initial) hyper-parameters that will be used for the experiments will be provided.

5.3.1. Implementation

The structure of the DDPG algorithm is build using the Tensorflow open-source machine learning library. The DDPG algorithm is used as the main model that determines the policy of the agents acting in the environment, it does this by periodically learning from the past experiences of the agents.

Learning is done every 30 seconds in order to ensure that the replay buffer has enough new samples every iteration. The effect of these new samples will become negligible over time but it is hypothesized that this will speed up the initial phase of learning where the policies are still changing rapidly.

5.3.2. Hyper-Parameters

For the initial hyper-parameters, it is decided to follow the hyper-parameters of the original authors of the DDPG algorithm, Lilicrap et al. [43]. This is done because they demonstrated that with their hyper-parameters the model was capable of training agents that demonstrated successful behavior over a wide variety of tasks. The selected hyper-parameters are shown in table 8. For the OU noise a standard deviation $\sigma = 0.2$ and a momentum parameter $\theta = 0.15$ are used, again in line with Lilicrap et al.

Table 8. Parameters used for the construction and trainig of the DDPG model

Parameter	Value
Actor & Critic Network Hidden Layers	2
Hidden Layer 1 Neurons	400
Hidden Layer 2 Neurons	300
Hidden Layer Activation	ReLU
Output Layer Activation	tanh
Learn rate	10^{-4} (Actor), 10^{-3} (Critic)
Discount Factor γ	0.99
Soft Update Parameter τ	0.001

5.4. Performance Metrics

This section will summarize all the performance metrics that are used for evaluating the learned behavior of the different agents. After the models are done training each model will be tested on the same scenario which will provide the data for these metrics.

Number of Intrusions

The number of intrusions will simply be determined by observing the intrusions of the agents with other aircraft. This number of intrusions is then used to calculate the expected climb/descend distance per intrusion and expected climb/descend time per intrusion.

Number of Instantaneous Conflicts

The number of instantaneous conflicts (containing at least one climbing/descending aircraft) will give an indication regarding the preventive measures taken by the agent. For example a large number of instantaneous conflicts with a low number of intrusions would indicate that the model has developed a policy where intrusions are avoided last minute.

Average Vertical Velocity and Standard Deviation

The average vertical velocity will be determined in order to evaluate the behavior of the different models. The standard deviation will provide additional insight into the triviality of the selected policy. A higher standard deviation will indicate that the policy does not output the same/a comparable vertical velocity regardless of the input.

Severity of Intrusions

The severity of intrusions is calculated using equation 5.2 and is a measure for how close aircraft get to each other [20]. It can be used to potentially identify differences in behavior between the different models.

$$LoS_{sev} = \frac{R - D_{cpa}}{R} \tag{5.2}$$

Convergence Rate

By using the same reward function for all experiments (or by still monitoring the original reward if reward-shaping is used) it is possible to compare the convergence rates during training between the different models used in the experiments.

6. Proof of Concept Experiment

The proof of concept experiment functions as an initial test in order to see if the used DDPG implementation is capable of learning desired behavior. Three different ways of describing this desired behavior are provided to the agent in the from of different reward functions, providing insight into how rewards should be presented to the agent. For this experiment, the agent's goal is to get to the target layer as quickly as possible while maintaining a velocity lower than 2m/s when entering the target layer.

The scenario and methods used for this experiment are identical with the scenario and methodology as described for experiment 1 (see Sections 4.1.1 & 4.2.3 and Chapter 5). The only changes made for this experiment are to the reward function, in order to formulate the previously defined task. First the defined reward functions will be described, followed by hypotheses corresponding to these reward functions. Finally the results for the different experiments will be given and discussed.

6.1. Reward Function

For this experiment, it is decided to use the same methodology as used for experiment 1 (One Degree of Freedom, Vertical Control). This is decided in order to verify if the model is able to learn from the elaborate state representation. The only change is made to the reward function. Instead of penalizing the agent for intrusions, it gets penalized at each timestep and only gets a reward when entering the target layers at $Vz < 2.0$m/s. This stimulates a "Go Fast, Decelerate" approach. The total reward function can be seen in equation 6.1. This experiment can be seen as the baseline approach for reward functions as defined by Barto and Sutton [30].

$$r_i = \begin{cases} 1 & \text{Aircraft in Target Layer, } Vz < 2.0m/s \\ 0 & \text{Aircraft in Target Layer, } Vz \geq 2.0m/s \\ -0.05 & \text{Otherwise} \end{cases} \tag{6.1}$$

Apart from this, two additional reward functions have been defined to identify the effect of reward shaping. The first provides an additional reward when the vertical velocity is between specified bounds, depending on the layer the aircraft is residing in, stimulating a gradual deceleration. The entire reward structure can be seen in equation 6.2. This experiment approaches reward shaping from the sub-goal point of view [33].

$$r_i = \begin{cases} 1 & \text{Aircraft in Target Layer, } Vz < 2.0m/s \\ 0.05 & \text{Aircraft 1-2 layers away from the Target Layer, } 1.0 < Vz < 2.5m/s \\ 0.05 & \text{Aircraft 2-4 layers away from the Target Layer, } 1.5 < Vz < 3.0m/s \\ 0.05 & \text{Aircraft more than 4 layers away from the Target Layer, } 2.5 < Vz < 4.5m/s \\ -0.1 & \text{Otherwise} \end{cases} \tag{6.2}$$

The second shaped reward uses a mathematical function to define the desired path, the reward per step is then determined based on the deviation from this path as seen in equation 6.3. Here V_{target} is calculated by equation 6.4 (an altered Sigmoid function), which is graphically depicted in figure 14. This experiment approaches reward shaping from the progress estimators' point of view [33].

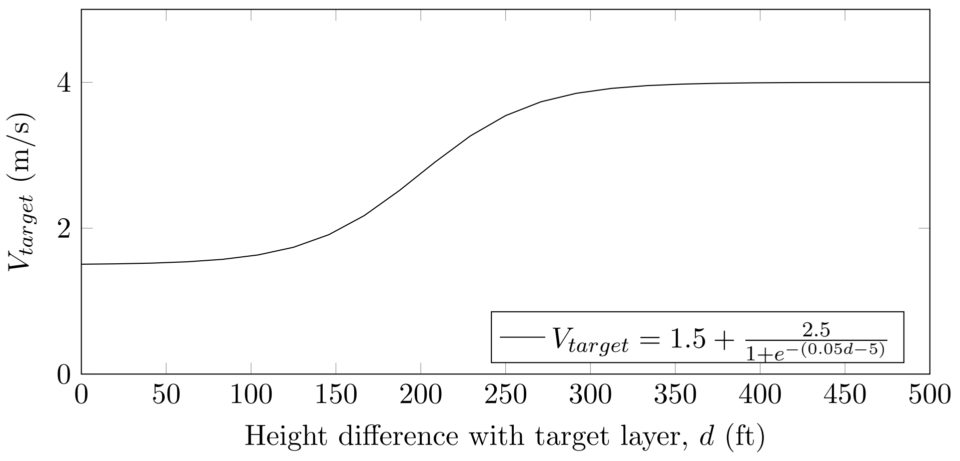


Fig. 14. Intended velocity profile for the reward

$$r_i = \begin{cases} 1 & \text{Aircraft in Target Layer}, Vz < 2.0m/s \\ 0.2(0.5 - |V_{target} - V_z|) & \text{Otherwise} \end{cases} \tag{6.3}$$

$$V_{target} = 1.5 + \frac{2.5}{1 + e^{-(0.05d-5)}} \tag{6.4}$$

6.2. Hypotheses

Based on the defined reward functions, three hypotheses have been established that will be introduced and explained in this section. Letters have been used instead of numbers in order to distinguish these hypotheses from the hypotheses formulated in Section 4.3.

Hypothesis A
The model trained on the simple reward function will suffer from plateau problems and favor faster velocities.

Based on the defined reward functions it is hypothesized that the initial reward function will suffer from plateau problems (Section 2.2.2) due to the sparsity of the positive reward. Because of the penalty for existing, it is expected that the model will prefer faster vertical velocities.

Hypothesis B
The models with reward shaping will demonstrate faster convergence.

Because the models with reward shaping obtain (different) rewards more often, it is expected that slight changes in the policy result in a larger change to the return when compared to sparse rewards. Because steps in the right direction obtain positive reinforcement it is more likely that the model will keep doing this and combined with exploration will find more optimal policies faster.

Hypothesis C
The models with reward shaping will outperform the simple reward model on it's own task.

As the reward shaped models get steered into the direction of the optimal policy for the simple reward it is hypothesized that both reward shaped models will outperform the model that has to discover the optimal policy completely on it's own.

6.3. Results and Discussion

The algorithm has trained on the different three different rewards (Equations 6.1, 6.2 and 6.3) for a total of 150.000 actions each. The selected actions have been plotted with their corresponding distance to the target layers in Figure 15. From these plots, it can clearly be seen that the spread of selected actions decreases overtime for all of the three reward functions. This seemingly indicates that they all converge to a policy, instead of randomly selecting actions. This convergence is also visible in the plots of Figure 16. These plots show the total return (rolling average of 500) obtained after the completion of each episode. In both plots for the return of the reward shaped models, initially a steep climb in return is observed before flattening, this is an indication of convergence. This behavior is not visible for the model trained on the first reward function, shown in the top-left of Figure 16. This could indicate convergence problems, but based on the selected actions, shown in the top left of Figure 15, it is assumed that this more likely is an indication of premature convergence to a sub-optimal, unstable, policy.

The selected actions for the simple reward functions, shown in the top-left of Figure 15, stay below the 2m/s limit for the entirety of the last 100.000 episodes. This indicates that the model becomes greedy towards the guaranteed +1 reward obtained at the end of the episode. The penalty of -0.05 for each timestep that the agent exists does not provide enough incentive to try and reach the target state quicker, even though this would result in a larger overall return. To demonstrate the reward potential lost by this trivial behavior, the number of required actions to complete an episode are compared for the different models. The (average) number of required actions to complete an episode are plotted in Figure 17. From this figure it can be seen that the model trained on the simple reward structure often required more than 20 additional actions to complete the episode compared to the other 2 models. Adding 20 additional actions, with a penalty of -0.05 for each selected action shows that the model gets penalized more than -1 in total, negating the benefit of the greedy behavior completely. It can therefore be concluded that from the perspective of this reward function the agent did not only select a trivial behavior, but it did not even select the best trivial behavior, which would be to always select the highest vertical velocity. This trivial behavior confirms hypothesis a partially, although instead of preferring a high velocity, penalty avoiding, behavior, the model demonstrated greedy low velocity behavior.

Nevertheless, it is still possible that the model will eventually converge to the optimal policy due to exploration. This trend is however not clearly visible when looking at the average return obtained during the first 4500 episodes as shown in the top-left of Figure 16. If the model were to convergence to the optimal policy a clearer uptrend in the return should likely have been visible

Increasing the frequency at which rewards are obtained by utilizing reward shaping techniques removes the trivial behavior of the first experiment. As indicated by the selected action plots for the two reward shape experiments, top-right and bottom of Figure 15. Looking at the average return plots of these two experiment (top-right and bottom plots of Figure 16), using progress estimators seemingly increases the stability of learning when compared to sub-goals. These plots also confirm hypothesis b, showing that both reward-shaped models converge faster than the simple reward model. From these initial results it is difficult to identify whether sub-goals or progress estimators is better when using reward shaping. At first glance it seems like the progress estimator performs better at decelerating fast, as is the desired behavior (bottom plot of Figure 1). It is however possible that with narrower and more frequent sub-goals similar/-better performance can be obtained with sub-goals as with progress estimators. To effectively determine this, additional research is required.

To efficiently compare the three different models, all of the selected actions for the three models are evaluated against the reward function of the first experiment, given in Equation 6.1. The results for this are plotted in Figure 18. From this plot it can be seen that both models that were trained when utilizing reward shaping techniques performed better at the task of the first model, without ever training on that task. This confirms hypothesis c and shows how reward shaping can be used to improve performance and convergence.

This proof of concept experiment demonstrated that with the current implementation, DDPG is capable of learning within the BlueSky environment. It was shown that shaping rewards speeds up learning and potentially improves overall performance. Finally, it was found that too sparse rewards in combination with DDPG can lead to trivial behavior. If proper reward shaping proofs to be difficult for avoiding intrusions, it might be worthwhile to research different DRL algorithms (SAC, TD3) to see how they deal with sparse rewards like the first scenario.

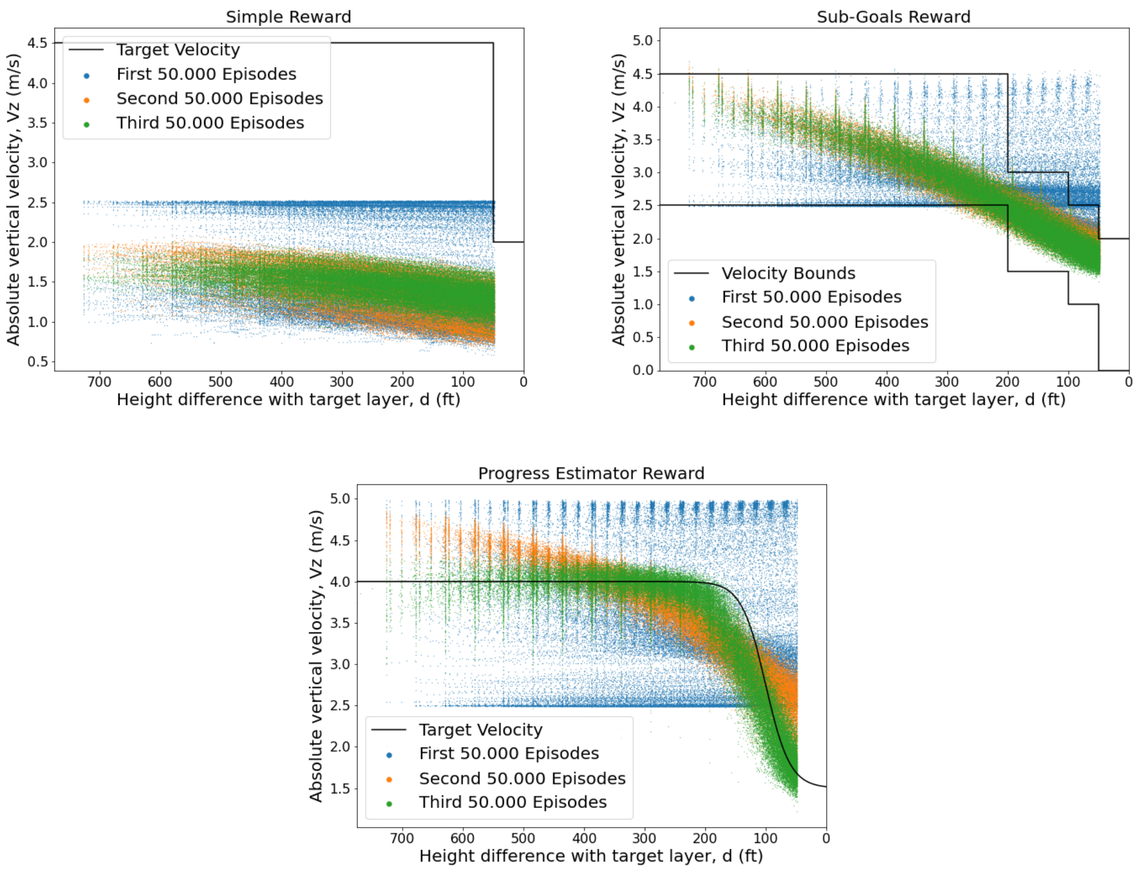


Fig. 15. Selected actions plotted against distance remaining to the Target Layer for the 3 different reward functions. The results have been split into 3 sets of actions in order to visualize the learning process of the model. Top-left is the selected actions corresponding to the simple, sparse reward. Top-right is the selected action when the reward function was shaped using sub-goals. Bottom is the selected actions when using progress estimators as selected reward shaping technique.

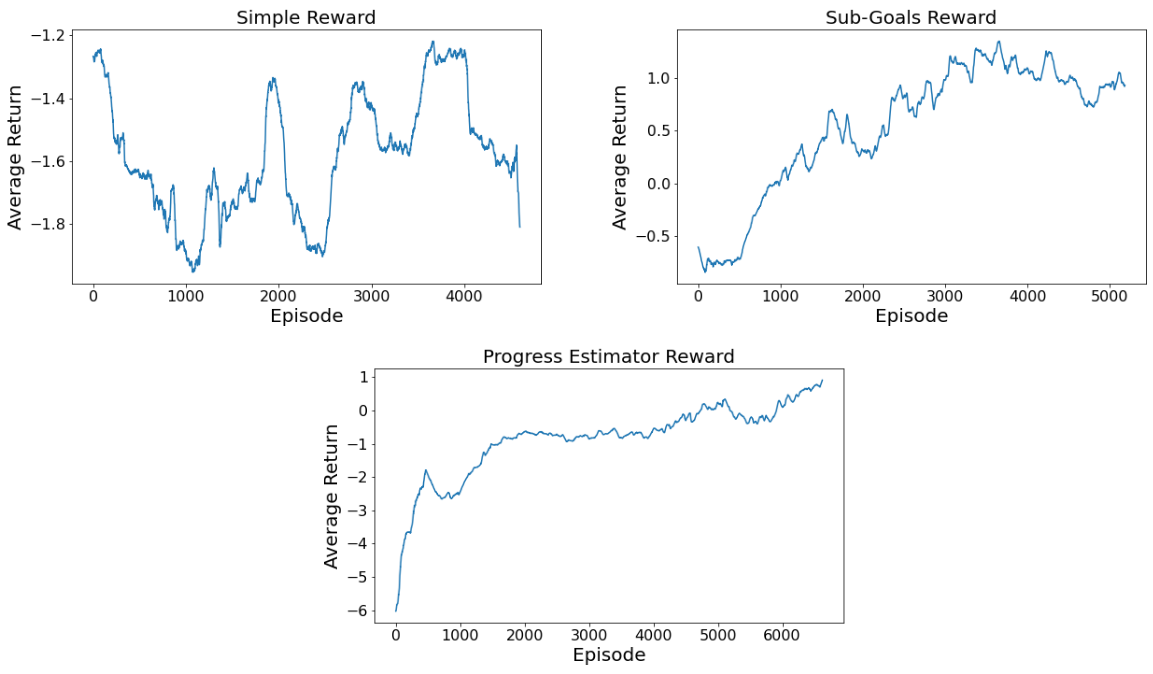


Fig. 16. Rolling average of 500 on the return for the 3 different reward functions. The returns can not directly be compared because different reward functions were used, they still give information regarding the convergence and improvement rates. Top-left is the average return corresponding to the simple, sparse reward. Top-right is the average return when the reward function was shaped using sub-goals. Bottom is the average return when using progress estimators as selected reward shaping technique.

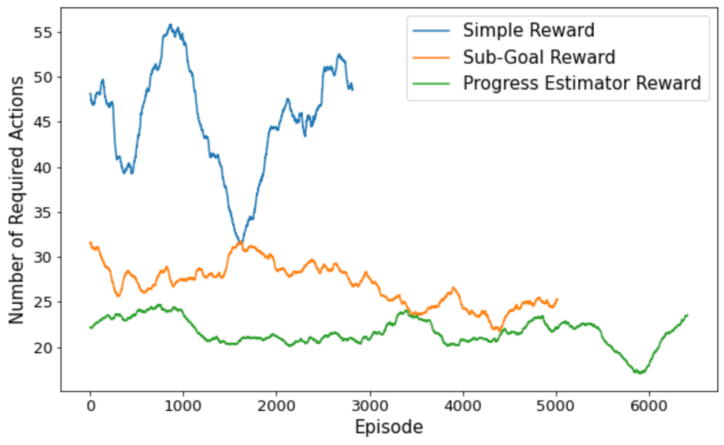


Fig. 17. Rolling average of 500 for the number of actions required to complete a single episode (from obtaining the altitude command, to reaching the target layer) for the different reward functions. Note that because the total actions is limited to 150.000 per model, not all models got to finish the same amount of episodes.

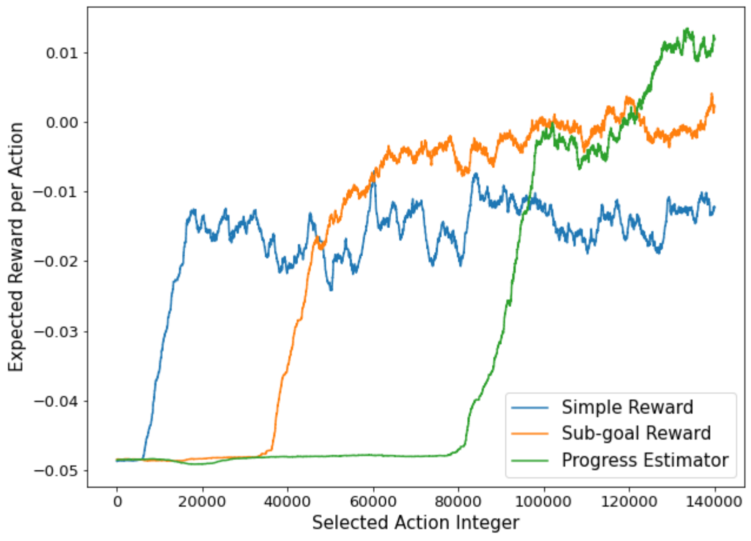


Fig. 18. Expected reward for each selected action for the different models, evaluated for the simple reward function

7. Potential Problems and Contingency Measures

As with any planning, setbacks and potential problems are bound to happen. This chapter will go over some of the more likely foreseen problems and discuss the contingency measures that will be taken if they occur.

7.1. Performance and Convergence Issues

Performance issues can be described as an agent that is incapable of obtaining better performance than the baseline or a previous model with fewer degrees of freedom. Convergence issues on the other are better described by learning slowly with frequent plateaus/not learning. These problems however can both have similar causes. Incorrect network parameters, sparsity in the rewards and incorrect state representation are the main expected causes and will therefore be discussed in this section.

Hyper-Parameter Tuning

The main hyper-parameters that are expected to affect the performance and convergence of the models are the learn rate, the number of hidden layers and the number of neurons in each layer. Adaptions to these parameters will be done in large steps in order to directly observe the effect of these parameters on performance and convergence, rather than fine-tuning the parameters. It is expected that hyper-parameter tuning will mostly be useful for convergence issues and have moderate benefits in terms of performance, which is more often dependent on the problem formulation.

Memory Buffer Prioritization

Another element that might lead to performance and convergence issues is the sparsity of the rewards, or in this case penalties. It is possible that the model does not have a large enough percentage of samples resulting in intrusions to effectively learn from these events. This problem can partially be counteracted by sampling these sparse events more frequently from the memory buffer. This was shown to improve performance for DDPG in a variety of tasks by Zha et al [61] and also used in the works of Wang Junjie et al. on lane change decision making [12].

Reward Shaping

Sparse rewards can also be counteracted by a process called reward shaping. It was shown that it can significantly reduce training time and potentially solve the plateau (convergence) problem described by Minsky [31]. Reward shaping can be done by using so-called progress estimators, for example by penalizing the agent for increasing conflict severity (e.g. TLoS or Dcpa decreases relatively to the previous state), or by rewarding it for decreasing conflict severity. If this approach of reward shaping is used it is important to ensure that no exploit is introduced into the MDP, for example by ensuring that positive rewards can not be obtained without first obtaining an equal or worse negative reward.

State Alteration

Performance or convergence issues can also be an indicator of incorrect or incomplete state representation. First, an analysis will be performed to see if there is a pattern distinguishable for the intrusions that occur. This pattern will be compared to the corresponding states that the agent observed in order to potentially identify changes that have to be made to the state.

7.2. Trivial Behavior

Trivial Behavior can be seen as an agent outputting the same action regardless of the state. This can be a potential problem caused by sparse rewards, for which potential solutions have been described in the previous Section (7.1).

Customized Noise Processes

In the scenario that the trivial behavior is not caused by sparse rewards, a potential solution can be customized noise processes. In the traditional implementation of DDPG, the same noise process is used for all of the actions, it is possible however that certain actions require more or more extreme exploration. For this reason, a potential solution would be to increase/change the action noise for the specific action that shows trivial behavior whilst keeping the noise for the other action(s)

7.3. Training Instabilities

A commonly mentioned problem with DDPG is the instability that can occur during training. If this occurs two main contingency measures have been defined.

Hyper-Parameter Tuning

DDPG uses target networks that slowly follow the updated weights of the main networks, these target networks are used for training and have shown to increase the stability of the model. A potential solution for instability (at the cost of a slower convergence rate) would be to decrease the soft update parameter τ, decreasing the rate at which the target networks follow the main networks.

Another common instability issue comes from the back-propagation process during weight updates, often indicating too high of a gradient. The instabilities resulting from this can be mitigated by reducing the learn rate of the actor and critic network or by changing the used activation functions.

Noise Degradation

Noise degradation can also be used to reduce the instability. Noise degradation will ensure that the standard deviation of the Ornstein Uhlenbeck noise reduces over time, reducing the intensity of the noise. This process is already commonly used in other DRL algorithms that use different exploration strategies (DQN [41])

7.4. Deteriorating Performance

Where training instability shows a lot of oscillations during training, deteriorating performance actually indicates a consistent decrease in performance in comparison with a previous policy.

Memory Buffer Design

One potential reason for deteriorating performance is that the agent got so good at completing the task that the memory buffer becomes saturated with positive rewards. This in turn results in a critic function that no longer is capable of identifying threats and starts to value all states with high expected rewards. Essentially the model forgets dangerous situations and becomes overly optimistic. In order to prevent this from happening it can be beneficial to always maintain a memory buffer that contains experiences from the initial exploration phase.

Weight Resetting

Something that also has been shown to help mitigate deteriorating performance (especially before convergence) is weight resetting, as demonstrated by Dasagi et al. [62].

7.5. Extreme Computation Time

Finally, it is expected that extreme computational requirements occurs, especially during the later experiments, which can result in time constraints with regards to the research.

Smaller Airspace

The first and simplest solution would be to decrease the airspace, this way the traffic densities can be maintained while reducing the number of required aircraft. This will most likely be beneficial for Experiment 4 (Section 4.2.6) where CD&R will be used for all aircraft in the airspace.

Hardware Upgrade

If the main limiting factor in the computational requirements can not be resolved by changing the scenarios or code it might be worthwhile to run the experiments on a different computer with more computational power.

8. Conclusion

Due to the anticipated increase in commercially utilized drones and their self-separation requirements, it is important that the conducting of vertical maneuvers does not have an as large impact on the safety of the airspace as that current research shows. This report proposed to investigate the usage of Deep Reinforcement Learning as a potential solution for self-separation during these maneuvers. Apart from being a potential solution, observing the behavior of the model might provide additional insights into what is regarded safe and can ultimately lead as a start for further research.

To increase the chance of successfully achieving the research goal, seven research activities have been defined. The first four research activities relate to the formulation of the problem such that Deep Reinforcement Learning can be used to solve them. These activities have been completed and are included in this preliminary report, although it is an iterative process and will likely be revisited in later stages of the research. The final three activities comprise the main phase of this research: Initial Experiments and Model Generation; Additional Testing and Data Gathering; Data, Behavior and Failure Mode Analysis.

For this research, it has been decided to initially use the Deep Deterministic Policy Gradient algorithm, which is capable of operating in continuous action space environments. For the simulation environment, the BlueSky Open Air Traffic Simulator will be used. Initial experiments have demonstrated that it is possible to use DDPG to control individual, simulated, aircraft. Although properly defining the goal of the model is shown to be a crucial element for success.

During the later experiments, it will be researched what the effect of different degrees of freedom will be on the safety and convergence rates. It is hypothesized that more degrees of freedom lead to safer operations at the cost of a longer training time. If problems occur during these experiments, the initial contingency measures to be taken have also been defined in this report. The outcome of these experiments will then provide the required information to answer the research question. Note that the results are only valid within the defined research scope, and additional research is required to evaluate the capabilities of Deep Reinforcement Learning in more complex and realistic environments.

Bibliography

[1] David Pierce. *Delivery drones are coming: Jeff Bezos promises half-hour shipping with Amazon Prime Air*. URL: `https://www.theverge.com/2013/12/1/5164340/delivery-drones-are-coming-jeff-bezos-previews-half-hour-shipping`. (accessed: 16.3.2021).

[2] DHL Express. *DHL Expres launches its first regular fully-automated and intelligent urban drone delivery service*. URL: `https://www.dhl.com/tw-en/home/press/press-archive/2019/dhl-express-launches-its-first-regular-fully-automated-and-intelligent-urban-drone-delivery-service.html`. (accessed: 16.3.2021).

[3] *Sesar Joint Undertaking. European drones outlook study - Unlocking the value for Europe. Technical report, Sesar Joint Undertaking, 2016.*

[4] *Organization, I.C.A. ICAO Circular 328 - Unmanned Aircraft Systems (UAS). Technical report, ICAO, 2011.*

[5] Jacco M Hoekstra, Ronald NHW van Gent, and Rob CJ Ruigrok. "Designing for safety: the 'free flight'air traffic management concept". In: *Reliability Engineering & System Safety* 75.2 (2002), pp. 215–232.

[6] Emmanuel Sunil et al. "Metropolis: Relating airspace structure and capacity for extreme traffic densities". In: *Proceedings of the 11th USA/Europe Air Traffic Management Research and Development Seminar (ATM2015), Lisbon (Portugal), 23-26 June, 2015*. FAA/Eurocontrol. 2015.

[7] Jacco M Hoekstra et al. "Geovectoring: reducing traffic complexity to increase the capacity of uav airspace". In: *International Conference for Research in Air Transportation, Barcelona, Spain*. 2018.

[8] Emmanuel Sunil et al. "Three-dimensional conflict count models for unstructured and layered airspace designs". In: *Transportation Research Part C: Emerging Technologies* 95 (2018), pp. 295–319.

[9] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[10] Christopher Berner et al. "Dota 2 with large scale deep reinforcement learning". In: *arXiv preprint arXiv:1912.06680* (2019).

[11] Carl-Johan Hoel, Krister Wolff, and Leo Laine. "Automated speed and lane change decision making using deep reinforcement learning". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 2148–2155.

[12] Junjie Wang et al. "Lane change decision-making through deep reinforcement learning with rule-based constraints". In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2019, pp. 1–6.

[13] Pin Wang, Hanhan Li, and Ching-Yao Chan. "Continuous control for automated lane change behavior based on deep deterministic policy gradient algorithm". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 1454–1460.

[14] Marta Ribeiro, Joost Ellerbroek, and Jacco Hoekstra. "Determining Optimal Conflict Avoidance Manoeuvres At High Densities With Reinforcement Learning". In: *10th SESAR Innovation Days* (2020).

[15] MM Doole, Joost Ellerbroek, and JM Hoekstra. “Drone delivery: Urban airspace traffic density estimation”. In: *8th SESAR Innovation Days, 2018* (2018).

[16] -. *Metropolis - overview.* URL: `https://homepage.tudelft.nl/7p97s/Metropolis`. (accessed: 4.6.2021).

[17] J Hoekstra et al. “Metropolis-Urban Airspace Design”. In: *Technical University of Delft National, Tech. Rep.* (2014).

[18] Emmanuel Sunil et al. “Analysis of airspace structure and capacity for decentralized separation using fast-time simulations”. In: *Journal of Guidance, Control, and Dynamics* 40.1 (2017), pp. 38–51.

[19] RL Ford. “On the use of height rules in off-route airspace”. In: *The Journal of Navigation* 36.2 (1983), pp. 269–287.

[20] Marta Ribeiro, Joost Ellerbroek, and Jacco Hoekstra. “Velocity Obstacle Based Conflict Avoidance in Urban Environment with Variable Speed Limit”. In: *Aerospace* 8.4 (2021), p. 93.

[21] Malik Doole et al. “Constrained Urban Airspace Design for Large-Scale Drone-Based Delivery Traffic”. In: *Aerospace* 8.2 (2021), p. 38.

[22] E Sunil. “Analyzing and Modeling Capacity for Decentralized Air Traffic Control”. PhD thesis. Delft University of Technology, 2019.

[23] Karl Bilimoria et al. “Performance evaluation of airborne separation assurance for free flight”. In: *18th Applied Aerodynamics Conference*. 2000, p. 4269.

[24] Marta Ribeiro, Joost Ellerbroek, and Jacco Hoekstra. “Review of conflict resolution methods for manned and unmanned aviation”. In: *Aerospace* 7.6 (2020), p. 79.

[25] Martijn Tra et al. “Modeling the intrinsic safety of unstructured and layered airspace designs”. In: *2017 ATM R&D Seminar*. 2017.

[26] Geert Woestenburg. “High-Rise Buildings on Compressible Soil: Research on the Structural Behavior and Substantiation of the Mitigating Measures”. In: (2020).

[27] John Conway. “The game of life”. In: *Scientific American* 223.4 (1970), p. 4.

[28] Dhruv Mauria Saxena et al. “Driving in dense traffic with model-free reinforcement learning”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 5385–5392.

[29] Richard Bellman. “A Markovian decision process”. In: *Journal of mathematics and mechanics* 6.5 (1957), pp. 679–684.

[30] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[31] Marvin Minsky. “Steps toward artificial intelligence”. In: *Proceedings of the IRE* 49.1 (1961), pp. 8–30.

[32] Andrew Y Ng, Daishi Harada, and Stuart Russell. “Policy invariance under reward transformations: Theory and application to reward shaping”. In: *Icml.* Vol. 99. 1999, pp. 278–287.

[33] Maja J Mataric. “Reward functions for accelerated learning”. In: *Machine learning proceedings 1994*. Elsevier, 1994, pp. 181–189.

[34] Jakob N Foerster et al. “Learning to communicate with deep multi-agent reinforcement learning”. In: *arXiv preprint arXiv:1605.06676* (2016).

[35] Maximilian Hüttenrauch, Adrian Šošić, and Gerhard Neumann. “Guided deep reinforcement learning for swarm systems”. In: *arXiv preprint arXiv:1709.06011* (2017).

[36] Maximilian Hüttenrauch, Sosic Adrian, Gerhard Neumann, et al. “Deep reinforcement learning for swarm systems”. In: *Journal of Machine Learning Research* 20.54 (2019), pp. 1–31.

[37] Karl Johan Åström. “Optimal control of Markov processes with incomplete state information”. In: *Journal of Mathematical Analysis and Applications* 10.1 (1965), pp. 174–205.

[38] Ian Goodfellow et al. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016.

[39] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep sparse rectifier neural networks”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 315–323.

[40] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), pp. 533–536.

[41] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *nature* 518.7540 (2015), pp. 529–533.

[42] Matteo Hessel et al. “Rainbow: Combining improvements in deep reinforcement learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[43] Timothy P Lillicrap et al. “Continuous control with deep reinforcement learning”. In: *arXiv preprint arXiv:1509.02971* (2015).

[44] David Silver et al. “Deterministic policy gradient algorithms”. In: *International conference on machine learning*. PMLR. 2014, pp. 387–395.

[45] Guillaume Matheron, Nicolas Perrin, and Olivier Sigaud. “The problem with DDPG: understanding failures in deterministic environments with sparse rewards”. In: *arXiv preprint arXiv:1911.11679* (2019).

[46] Yan Duan et al. “Benchmarking deep reinforcement learning for continuous control”. In: *International conference on machine learning*. PMLR. 2016, pp. 1329–1338.

[47] Tuomas Haarnoja et al. “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1861–1870.

[48] Volodymyr Mnih et al. “Asynchronous methods for deep reinforcement learning”. In: *International conference on machine learning*. PMLR. 2016, pp. 1928–1937.

[49] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[50] Guan Wang et al. “Cooperative lane changing via deep reinforcement learning”. In: *arXiv preprint arXiv:1906.08662* (2019).

[51] Zine el abidine Kherroubi et al. “Novel off-board decision-making strategy for connected and autonomous vehicles (Use case highway: on-ramp merging)”. PhD thesis. Lyon, 2020.

[52] Ahmad EL Sallab et al. “Deep reinforcement learning framework for autonomous driving”. In: *Electronic Imaging* 2017.19 (2017), pp. 70–76.

[53] James K Kuchar and Lee C Yang. “A review of conflict detection and resolution modeling methods”. In: *IEEE Transactions on intelligent transportation systems* 1.4 (2000), pp. 179–189.

[54] Marc W Brittain and Peng Wei. “One to Any: Distributed Conflict Resolution with Deep Multi-Agent Reinforcement Learning and Long Short-Term Memory”. In: *AIAA Scitech 2021 Forum*. 2021, p. 1952.

[55] Marc Brittain and Peng Wei. “Autonomous air traffic controller: A deep multi-agent reinforcement learning approach”. In: *arXiv preprint arXiv:1905.01303* (2019).

[56] JM Hoekstra, RCJ Ruigrok, and RNHW Van Gent. “Free flight in a crowded airspace?” In: *Proceedings of the 3rd USA/Europe Air Traffic Management R&D Seminar*. 2001.

[57] Chris Lee, Bruce Hellinga, and Frank Saccomanno. “Evaluation of variable speed limits to improve traffic safety”. In: *Transportation research part C: emerging technologies* 14.3 (2006), pp. 213–228.

[58] Mairie de Paris. *Paris land use plan, Regulations General Urban Zone*. https://cdn.paris.fr/paris/2020/02/ 2016.

[59] D Alejo et al. “Multi-UAV collision avoidance with separation assurance under uncertainties”. In: *2009 IEEE international conference on mechatronics*. IEEE. 2009, pp. 1–6.

[60] Jacco M Hoekstra and Joost 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. FAA/Eurocontrol USA/Europe. 2016, p. 132.

[61] Daochen Zha et al. “Experience replay optimization”. In: *arXiv preprint arXiv:1906.08387* (2019).

[62] Vibhavari Dasagi et al. “Ctrl-z: Recovering from instability in reinforcement learning”. In: *arXiv preprint arXiv:1910.03732* (2019).