# Using Relative State Transformer Models for Multi-Agent Reinforcement Learning in Air Traffic Control

D.J. Groot & J. Ellerbroek, and J. Hoekstra
Control and Simulation, Faculty of Aerospace Engineering
Delft University of Technology, The Netherlands

***Abstract*—Deep Reinforcement Learning has seen more usage in the field of Air Traffic Control over the last couple of years. As the number of aircraft in a given sector of airspace is not constant, there is a need for methods to be invariant to the number of agents in the system. Often this is done by making a selection of the aircraft that will be included in the state, which introduces human biases. Another option that has been used is Recurrent Neural Networks to process the entire sequence of aircraft present. These methods however are sequence-dependent and can give different results depending on the order that the aircraft are given, which is undesirable. Methods that solely rely on attention mechanisms, such as transformers, allow sequential data to be processed in a sequence-invariant manner by using multi-head attention mechanisms. However, because traditional Transformers operate on individual tokens, this does not allow for relative state information to be encoded into the hidden state. This paper shows that by performing a transformation operation on the key and value tokens, it is possible to use Transformers on relative states, at the cost of a factor (N-1) additional attention computations, where N is the number of agents in the system. This adaptation allows relative state Transformers to obtain significantly higher performance than standard Transformers. The results also showed that using attention mechanisms to construct the initial observation vector out of a total of 20 agents results in similar, but slightly lower, performance to handcrafted observation vectors, without requiring manual selection of the important agents. Future research should investigate whether additional changes to the attention mechanisms and their training can result in higher performance.**

***Keywords*—Air Traffic Control, Multi-Agent Reinforcement Learning, Transformers, Artificial Intelligence**

## I. Introduction

The last couple of years has shown increasing interest in research on utilizing Deep Reinforcement Learning (DRL) methods for conflict resolution and safe multi-agent navigation for Air Traffic Control (ATC) operations [1]. One issue with conventional neural networks underlying the DRL methods, however, is the requirement of a fixed-length input vector. Most of the current research therefore does a pre-selection on the number of aircraft that will be considered for the model's input or keeps the number of agents/aircraft in the environment constant in an artificial way, which limits the application of trained models in environments with variable number of aircraft [1].

One way to overcome this issue is by encoding the information graphically in fixed-size images, and using Convolutional Neural Networks (CNNs) to distill useful information, similarly to ATC screens [2]. Another possibility is to use Recurrent Neural Networks (RNNs) that operate over the entire set of aircraft in the environment, encoding relevant information into a fixed-size hidden state. Both Long-Short Term Memory networks (LSTMs) [3] and Gated Recurrent Units (GRUs) [4], have been shown to work very well for the purpose of conflict resolution. However, as RNNs go over the input data sequentially, they have the issue that the output is sequence-dependent [5]. This is important for data structures that have an inherent sequence, such as natural language processing (NLP) or time series, but can lead to unexpected behaviour in situations where the input sequence should not matter, for example in most Multi-Agent Reinforcement Learning (MARL) applications, such as robot or traffic studies.

Transformers have been introduced as an alternative to RNNs for sequential input processing to allow easier parallel training [6]. Since the introduction, it has seen a lot of success in the area of NLP (for example all versions of GPT [7] and Google's Bard powered by PaLM 2 [8]). Traditional Transformers are sequence invariant, removing the sequence dependency that RNNs have for MARL applications. Therefore, it is interesting to research the applicability of the Transformer network architectures for the ATC.

Transformers compute the relative importance of so-called tokens (aircraft states in this study) using a mechanism called dot-product attention. This process results in a scalar weight value for each token/aircraft state pair, which determines how much of the other aircraft state gets stored in a hidden state. As the relation between the two aircraft states gets reduced to a single-weight scalar, this allows only the absolute state information of the other aircraft to be stored in the hidden state. This multiplicative attention has already been shown to effectively control aircraft based on the absolute states [9].

However, most MARL applications in ATC utilize relative states instead of absolute states. To accommodate this requires an alteration of the conventional Transformer architecture.

This paper introduces a modification to the conventional transformer architecture that allows it to operate on relative states, at the cost of a factor of (N-1) additional attention computations, where N is the number of agents in the system. This modified transformer architecture is then tested against three other network architectures on the task of conflict resolution

SESAR Innovation Days
27-30 November 2023
Seville

and navigation in high-density airspace, all trained with the Soft Actor-Critic (SAC) DRL algorithm. The other network architectures are a conventional transformer architecture and two fully connected neural network architectures. One using the absolute state of the three closest aircraft and the other using the relative state of the three closest aircraft.

The resulting attention mechanism is similar to that used in the recent work of Brittain et al., which has been developed in parallel to this study and uses some relative states and discrete actions [10]. This study already showed that including additional relative information improves the performance over conventional 'absolute' attention. The main goal of this research is to identify whether Transformers can be used for DRL application in ATC as an alternative to RNN based methods. As a secondary goal, this research also investigates the importance of relative states over absolute states in MARL for ATC.

## II. Methods

In the Methods section, first the Markov Decision Process (MDP) formulations of the problem are described, with differences between the different observation vectors used. Then the used Reinforcement Learning algorithm, Soft Actor-Critic is briefly introduced, followed by the different network architectures: a Fully Connected Feed Forward Neural Network (FNN), a standard single-block Transformer Network and a Transformer Network with relative state based self-attention mechanisms.

### A. Markov Decision Process Abstraction

To solve the problem of safely navigating through a dense airspace, the problem is first abstracted into the form of a (multi-agent) MDP. This requires a description of the state-space representation, action-space and reward function, which are all given in this section. Furthermore, the state-transition function is required, which is obtained from the used ATC simulator described in section III-A.

#### 1) State-Space Representation

To provide the state-space representation, fixed size observation vectors are created that contain information of the ownship, and in the case of the FNN methods also that of the three closest other agents. This does provide limitations on the input, increasing the partial observability of the problem for the FNN methods when compared with the Transformer methods, which utilize the states of all agents in the environment. For this research, two different types of observation vectors are used, absolute and relative observation vectors.

##### *Absolute State Observation*

The absolute state observation uses the coordinates and velocities of the agents in an environment-centered reference system. For each individual aircraft, their own observation is given in Table I. Here $x$ and $y$ are the x- and y-positions with respect to the origin of the environment in meters, with the origin at the center of the environment and y pointing north. $v_x$, $v_y$ and $v$ are the velocities in the x- and y-directions and the overall magnitude respectively. Finally $\sin(\delta)$ and $\cos(\delta)$ are the sine and cosine components of the drift angle, which is the angular deviation of the optimal, scenario-dependent, path in radians. The drift angle is transformed to ensure a continuous derivative of the drift over all angles.

TABLE I. Observation vector for the absolute state observations.

| Variable | symbol |
|---|---|
| x position of the ownship | $x_i$ |
| y position of the ownship | $y_i$ |
| x velocity of the ownship | $v_{x_i}$ |
| y velocity of the ownship | $v_{y_i}$ |
| own absolute velocity | $v_i$ |
| sin of the drift | $\sin(\delta_i)$ |
| cos of the drift | $\cos(\delta_i)$ |
| *information regarding 3 closest agents:* | |
| x position of the other agent | $x_j$ |
| y position of the other agent | $y_j$ |
| x velocity of the other agent | $v_{x_j}$ |
| y position of the other agent | $v_{y_j}$ |

For the Fully Connected Feed Forward Network (see section II-C1), additional information regarding the 3 closest agents is concatenated to the observation vector. The concatenated vector for each aircraft $j$ in the list of closest aircraft is: $[x_j,\ y_j,\ v_{x_j},\ v_{y_j}]$. Resulting in a total observation vector per agent of size: (7 + 3*4) = 1x19.

##### *Relative State Observation*

The relative state observation vector uses the same information as the absolute state observation vector, but transformed into the ownship's frame of reference, with positive x pointing in the direction of flight. Additionally, the relative distance is added for the 3 closest agents. Finally, as the frame of reference is now the ownship's frame of reference, the x and y position and velocity information of the ownship have been removed from the state vector as they will always be equal to zero or redundant (e.g. velocity in the x direction will be the same as the own absolute velocity). The resulting state representation is given in table II. This results in a total observation vector per agent of size: (3 + 3*5 ) = 1x18.

TABLE II. Observation vector for the relative state observations.

| Variable | symbol |
|---|---|
| own absolute velocity | $v_i$ |
| sin of the drift | $\sin(\delta_i)$ |
| cos of the drift | $\cos(\delta_i)$ |
| *information regarding 3 closest agents:* | |
| relative x position of the other agent | $x_{ij}$ |
| relative y position of the other agent | $y_{ij}$ |
| relative x velocity of the other agent | $v_{x_{ij}}$ |
| relative y position of the other agent | $v_{y_{ij}}$ |
| relative distance of the other agent | $d_{ij}$ |

*Normalization of the Observations*

To enhance the stability of the learning process of the neural networks, all of the observations are normalized [11]. This is done using z-score normalization where the mean and standard deviation of the individual observation elements are approximated through 50,000 state transitions under a random actor.

*2) Action Space*

The action space of the environment is continuous and allows for 2 types of actions per agent, a heading change and a speed change. The bounds of these actions are given in Table III. It is noted that the values given in this table are not realistic considering actual aircraft dynamics, however, they are used to allow more flexibility in the paths generated by the agents, resulting in the performance only being evaluated on the quality of the paths, and not the understanding of the environment dynamics.

TABLE III. ACTION BOUNDS FOR THE DIFFERENT AVAILABLE ACTIONS TO THE RL METHODS.

| Action | Bounds |
|---|---|
| Heading | [-1,+1] transforms to [-25°,+25°] |
| Speed | [-1,+1] transforms to [-50m/s,+50m/s] |

*3) Reward Function*

The reward function consists of 2 individual cost functions and is strictly negative. This ensures that the maximum reward over an infinite time horizon for a perfect agent approaches zero instead of positive infinity. Yielding more stable learning as the gradient over time decreases instead of increases.

The total reward function can be seen in equation 1. The first part of the equation penalizes the drift of the agent with respect to its destination. The second part of the equation is a simple boolean operation which equals 0 when the aircraft is intrusion-free and -1 if the aircraft is not. The magnitude of the weight $w_{drift}$ is set at -0.1, to ensure that rotating in the same spot yields a lower score than flying perfectly straight with the maximum number of intrusions. (This behaviour was observed as a local optimum for drift weight values magnitudes lower.)

$$r = w_{drift} \cdot |\delta| + \begin{cases} -1 & \text{if intrusion} \\ 0 & \text{if not intrusion} \end{cases} \tag{1}$$

## B. Soft Actor-Critic Algorithm

For the training of the different models, a version of the Soft Actor-Critic (SAC) algorithm with automatic entropy regulation is used [12]. SAC is a widely used RL algorithm that has been shown to be effective at previous RL applications for intrusion-free navigation [13], [14].

In total, each model, indicated by their different network architectures, is trained for a total of 10,000 episodes (1,500,000 steps), using the hyperparameters given in table IV. The models will then be compared for their best-obtained performance under a window with a rolling average of 100 during training.

TABLE IV. HYPERPARAMETERS USED FOR THE SAC ALGORITHM

| Parameter | Value |
|---|---|
| Optimizer | Adam |
| Alpha Learnrate | 3e-4 |
| Actor Learnrate | 3e-4 |
| Critic Learnrate | 3e-3 |
| Discount factor ($\gamma$) | 1.0 |
| Memory buffer size | 20e6 |
| Sample size | 2048 |
| Smoothing coefficient ($\tau$) | 5e-3 |
| Network update frequency | 8 |

## C. Network Architectures

For this study, a total of 3 different network architectures are used. Additionally, for the standard feed-forward network architecture, both the absolute and relative state observation vectors are used. This results in a total of 4 different methods being used for this research. An overview of the 4 different methods is given in Figure 1. The size of the different network architectures in terms of memory allocation is given in Table V, to ensure an equal comparison of the different architectures in terms of computational complexity and approximation power it is attempted to keep them similar in size.

TABLE V. SIZE OF THE DIFFERENT NETWORK ARCHITECTURES DURING TRAINING (ACTOR + CRITIC NETWORKS) AND EVALUATION (ACTOR ONLY)

| Network Architecture | Training Size | Evaluation Size |
|---|---|---|
| Absolute State, FNN | 1436 kB | 289 kB |
| Relative State, FNN | 1436 kB | 289 kB |
| Absolute State, Transformer | 1447 kB | 291 kB |
| Relative State, Transformer | 1471 kB | 296 kB |

*1) Standard Feed-Forward network architecture*

The first two methods tested in this research use a standard feed-forward network architecture. This means that the observation vectors, according to the description of section II-C2, are directly used as input into a simple network of 2 fully connected hidden layers with 256 neurons each. Both hidden layers use the ReLU activation function. The output of the last hidden layer then goes into a 1x2 output layer with a tanh activation function to map it to a range of [-1,+1].

*2) Transformer Architecture*

The transformer network architecture, introduced by A. Vaswani et al [6], uses scaled dot-product attention to compute the weight/importance of the tokens[1] (keys) in the input with respect to a reference token, the query. The weights for all of the tokens are then normalized using a soft-max operation to ensure that the sum of all the weights equals 1. Then, multiplying these weights with the tokens (values) and summing them, results in a new token. This token is referred to as the latent space of the Transformer, to stay consistent with deep learning terminology. This process is graphically shown in figure 2. For this method, the query tokens are the individual

[1]In this research the tokens are individual aircraft states, however, in the original paper tokens are vector encodings of strings

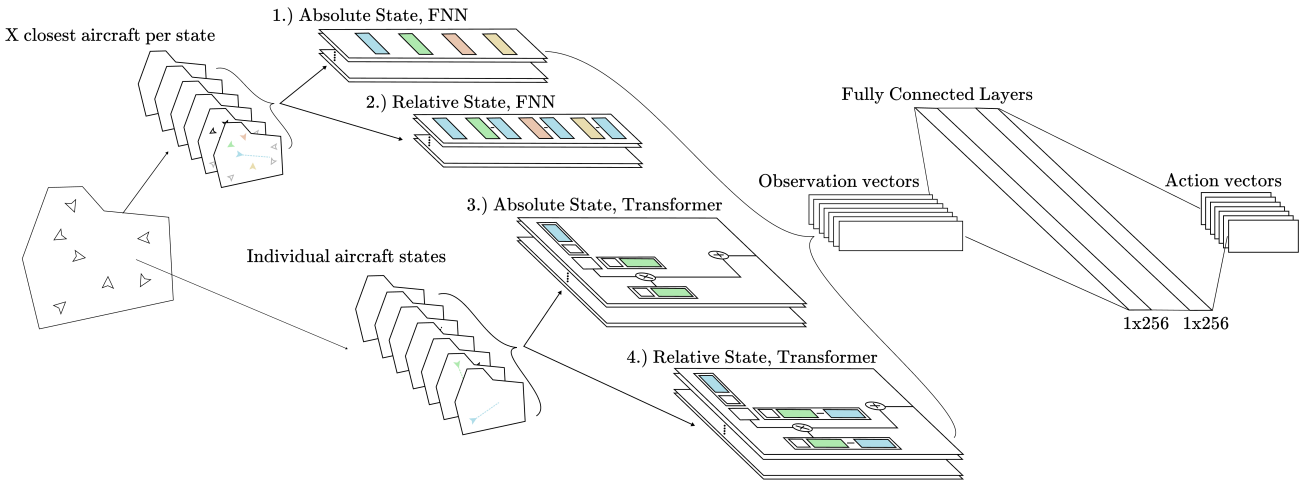


Figure 1. The four different methods/network architectures used for this research. 1.) Standard feed-forward network with the concatenated absolute states described in section II-A1. 2.) Standard feed-forward network with the concatenated relative states described in section II-A1. 3.) Default Transformer model with the tokens for the attention mechanisms being the agent's absolute state. 4.) Transformer model with modified attention mechanism that includes mapping of the key and value tokens to the reference frame of the query token. Note that in this figure the relative state operation is only shown as a subtraction of the different states, it however also includes a rotation operation that has been omitted for the clarity of the figure

'ownship' observations, and the key and value tokens are the absolute intruder observations, both described in section II-A1.

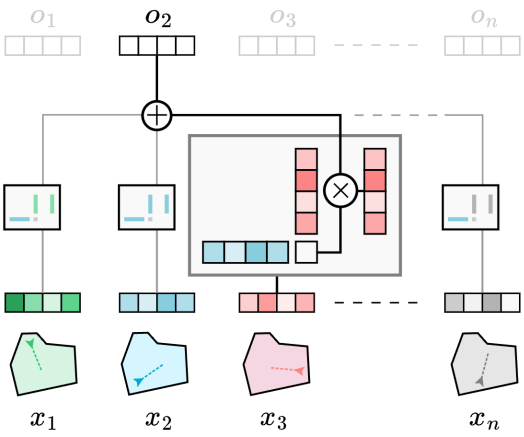


Figure 2. Illustration of a single head of the dot-product attention mechanism used for the transformer architecture. Illustration based on the illustration by Peter Bloem: https://peterbloem.nl/blog/transformers

To ensure that relevant features are used for determining the weights and the latent space, this attention network introduces 3 additional sets of learn-able weight matrices: the Query ($\mathbf{Q}$, size $d_q, d_k$), the Key ($\mathbf{K}$, size $d_k, d_k$) and the Value ($\mathbf{V}$, size $d_v, d_v$) matrices. Here $d_q$, $d_k$ and $d_v$ are the length of the query $q$, key $k$ and value $v$ tokens respectively. For this research the $q$ tokens are the normalized absolute ownship information given in table I and the $k$ and $v$ tokens are identical tokens containing the normalized absolute intruder information given in table I. Equations 2, 3 and 4 give a generalized mathematical representation of the process illustrated in figure 2, using these matrices. In equation 2 a scaling factor of $\frac{1}{\sqrt{d_k}}$ is used to counter the growth of the potential absolute value with increasing dimensionality $d_k$, and is likely not necessary for the input token sizes used in this research, but is still used for completeness of the method.

$$w_{ij} = \frac{\mathbf{Q}q_i \cdot \mathbf{K}k_j^T}{\sqrt{d_k}} \tag{2}$$

$$s(w_{ij}) = \frac{e^{w_{ij}}}{\sum_{n=1}^{N} e^{w_{nj}}} \tag{3}$$

$$o_i = \sum_{n=1}^{N} s(w_{nj})\mathbf{V}v_n \tag{4}$$

This process is then done in parallel h times, where h is the number of 'heads' of the architecture. Each head has a unique set of $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ matrices. The output of the heads are then concatenated to obtain tokens of size $(d_k \cdot h)$. In the original implementation, these tokens are then projected back to size $d_k$ using a learnable matrix of size $(d_k \cdot h, d_k)$, however, for this research this step is skipped such that the dimensionality of the latent space is similar in size to the observation vectors used by the absolute and relative state FNN methods.

Finally, the original aircraft state, $x_i$, is concatenated to the latent space, giving the resulting observation vector of size $(d_q + h \cdot d_v)$, that is used as input for a standard FNN network shown in figure 1. Given the size of $d_q = 7$, $d_v = d_k = 4$ and the number of heads, $h = 3$, the resulting observation vector is: (7 + 3*4) = 1x19, which is the same as the observation vector used by the absolute state FNN.

*3) Relative Attention Transformer Architecture*

One of the issues with conventional transformers is that scaled dot-product attention mechanisms are unable to store the relative relation between the query and key tokens in the latent space, because their relation gets reduced to a single weight scalar. This weight scalar then gets multiplied with the value token according to equation 4, which can only contribute absolute state information to the latent space.

To counter this issue, this method applies the attention mechanism to the relative state tokens, obtained by translating and rotating the encoded state information, which is graphically shown in figure 3. Doing this transforms the coordinate system of the key and value states from environment reference frame to 'query' reference frame. Additionally, to the resulting key and value tokens, a mapping of the relative distance is added. This mapping function is given in equation 5 and ensures that distances closer to 0 approach 1 and the value asymptotically goes to zero for larger values of the relative distance. For the relative state transformers the information in the query token is reduced to the same information as given in table II, combined with the addition of the distance results in $d_q = 3$, $d_v = d_k = 5$ and the number of heads, $h = 3$, the resulting observation vector is: (3 + 3*5) = 1x18, which is the same as the observation vector used by the relative state FNN to ensure a fair comparison between the different methods.

$$f(d) = \frac{1}{e^{-1+5*(d-0.2)}} \tag{5}$$

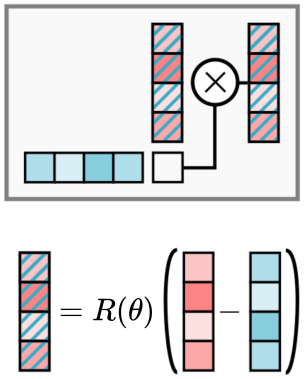


Figure 3. Graphical representation of the change added to the key and value tokens in the relative state transformer models. Here $R(\theta)$ is the rotation operation to the query reference frame.

This operation increases the input size for the attention network from (N, T) to (N, N-1, T), where N is the number of agents/tokens and T is the token length. Because the attention matrices $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ only operate on the T dimension, it is possible to flatten the input into a shape of (N * (N-1), T) before performing the initial matrix multiplications, which does not introduce sequence-order effects due to the dot-product operation. The resulting tokens can then be restored to the original shape of (N, N-1, T) to compute the latent space representations as shown in figure 2. This way, an increase of factor N-1 for the number of attention calculations is required for this method when compared with conventional multi-head attention.

## III. Experiment: Importance of Relative States on RL Method Efficacy for Conflict Resolution Tasks

### A. Simulation Environment

To train the models and evaluate their performance, the ATC environment, introduced for the Eurocontrol Innovation Masterclass was used[2]. This simulator does not consider aircraft dynamics, instead, the states are linearly propagated in time with a time-step of 5 seconds and any changes to the states are applied directly. Because of this, the trained models will only be evaluated for the quality of their paths. Future research will have to test the models in higher fidelity simulators to see if similar results can be obtained when the models also have to learn the environment dynamics.

### B. Simulation Scenario

The simulation scenario consists of 20 aircraft flying in procedurally generated airspace, with a given track that they have to follow, whilst ensuring that a minimum separation of 5 Nautical Miles (9.26 km) is maintained. A violation of this requirement is called an intrusion.

Each airspace is randomly generated to have an area of 10.000 km$^2$ with a margin of 10%. Traffic is then initialized with random positions and orientations such that the initial conditions are intrusion-free. Each scenario is then rolled-out for 150 time-steps before a new scenario is generated.

Figure 4 shows the initial conditions of an example scenario that is generated, in this figure the circles have a radius of half the minimum separation requirements. The traffic density and number of aircraft for this research are higher than those used in comparable studies to not trivialize the task of the attention mechanisms.

### C. Independent Variables

The only independent variable that is changed throughout the experiments is the network architecture and state representation used for the SAC algorithm. These methods are: 1.) Absolute State FNN, 2.) Relative State FNN, 3.) Conventional Transformer, and 4.) Relative Attention Transformer. More information and details about these methods are given in section II-C.

### D. Dependent Measures

To analyse the performance of the different models, 3 different parameters, and their evolution over time are evaluated.

[2]Source code for the original environment is available at: "https://github.com/ramondalmau/atcenv". The version used for this paper can be found at: "https://github.com/jangroter/atcenv/tree/Master".

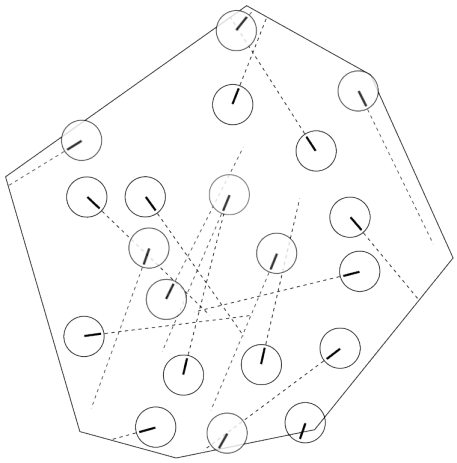

Figure 4. Initial conditions of an example scenario generated for training of the different methods. The dotted lines indicate their destination, the thicker lines the heading and the circles indicate their protected zone.

*1) Reward*

The reward parameter shows the mean of the total sum of total rewards obtained by all agents in a single episode. It therefore directly shows the performance of the model, based on the given reward function. The reward is the only objective measure of the model's performance, as it is the only information available to the model during training, and is directly used for optimizing the policy.

*2) Intrusions*

The intrusions measure is given as the total number of timesteps a single agent is, on average, in a state of intrusion per episode. This measure is used to evaluate the safety of the trained policy.

*3) Drift*

The drift is measured as the average deviation from the optimal path at over the entire episode. It is measured in absolute values such that a positive and negative drift will not cancel each other out. This measure is used to evaluate the path-efficiency of the policy.

### *E. Hypotheses*

*1) Transformer Network Architectures will outperform FFN*

Because Transformers use attention mechanisms to learn which tokens, and thus which aircraft contain relevant information, it is hypothesized that they are capable of constructing a better state representation of the environment than the handcrafted state representation done by humans, leading to better performance.

*2) Relative State Observation Vectors will outperform Absolute State Observation Vectors*

Relative states effectively reduce the size of the problem space, as illustrated in figure 5. Therefore it is hypothesized that it is easier to learn patterns and generalize between different scenarios than when using absolute states.

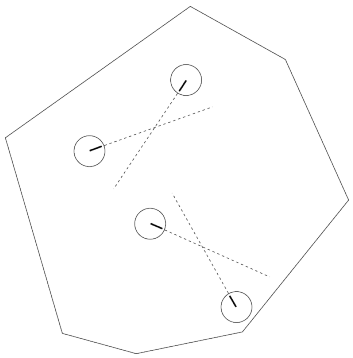

Figure 5. Illustration of 2 conflict situations that are identical when observed in relative state space, but different in absolute state space.

## IV. Experiment: Results

### *A. Evolution of Reward over Time*

Figure 6 shows the evolution of the reward over the number of episodes. It can immediately be observed that the methods utilizing relative state information have a much steeper initial learning curve and higher asymptotic performance than the absolute state methods. Another observation that can be made is that the relative FNN method reaches the highest performance much faster than the relative Transformer method, but then starts to decrease in performance.

Table VI shows the maximum attained reward during training as a rolling average of 100 for the different methods, and the episode at which this reward was attained. For reference, this table also includes the values for a policy that always flies with 0 degrees of drift, and for a fully random policy. This table shows that the maximum attained performance of the relative FNN method is higher than that of the relative Transformer method, indicating better performance. Furthermore, this maximum occurs earlier during training, showing a faster training curve for the FNN method.

This table also highlights the importance of the relative states, as both of the relative state methods outperform their absolute state counterparts.

TABLE VI. Rewards observed during training for the different methods

| Method | Max reward (ao100) | Episode |
|---|---|---|
| Relative state FNN | $-3.65$ | 2661 |
| Absolute state FNN | $-15.10$ | 2018 |
| Relative state Transformer | $-4.25$ | 8149 |
| Absolute state Transformer | $-20.73$ | 1114 |
| *Reference values* | | |
| No Drift Policy | $-22.40$ | - |
| Random Policy | $-25.10$ | - |

### *B. Number of Intrusions for the different methods*

In figure 7 the average number of intrusion time-steps per agent is shown as a function of the episodes. This figure clearly highlights the difference between the relative and absolute

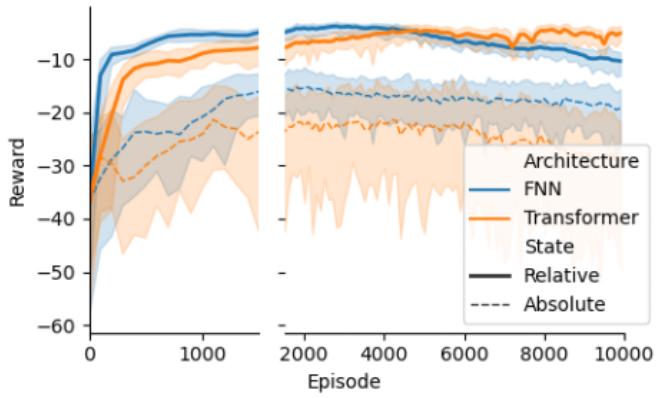


Figure 6. Evolution of the reward over time for the different methods, the initial 1500 episodes on the left are slightly enlarged to highlight the different initial training curves.

methods in terms of efficacy for conflict resolution. Both of the relative methods decrease the number of intrusions rapidly, with the relative FNN method obtaining the lowest number of intrusions.

These results are again shown in table VII, where the lowest number of intrusions and the corresponding episode are given for the different methods and the reference methods. This table shows that the lowest number of intrusions per episode for the relative state FNN method is almost half that of the Transformer equivalent. Interestingly, the observed discrepancy for the episode at which these minima occur is even larger than it is for the reward (2661 & 8149 for the reward and 1213 & 9380 for the number of intrusions for the FNN and Transformer respectively). This shows that the relative state Transformer model requires a larger set of state transitions to learn proper conflict-resolving actions than the FNN method.

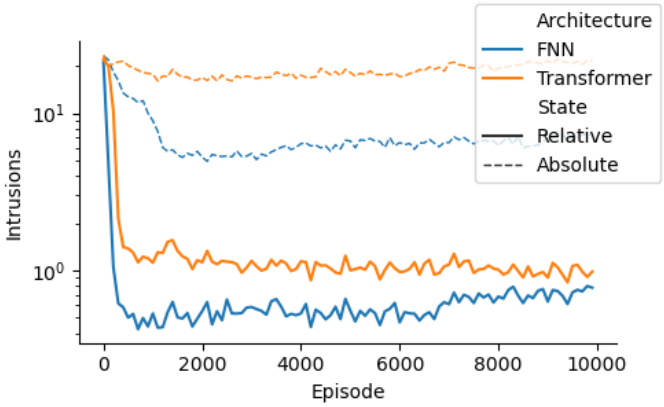


Figure 7. Evolution of the (log)number of intrusion time-steps, per agent per episode, over time for the different methods.

### C. Average Drift for the different methods

The evolution of the drift error over time is given in figure 8, in this figure the difference between the relative and absolute

TABLE VII. INTRUSIONS OBSERVED DURING TRAINING FOR THE DIFFERENT METHODS

| Method | Min intrusions (ao100) | Episode |
|---|---|---|
| Relative state FNN | 0.40 | 1213 |
| Absolute state FNN | 4.85 | 2037 |
| Relative state Transformer | 0.77 | 9380 |
| Absolute state Transformer | 15.52 | 1114 |
| *Reference values* | | |
| No Drift Policy | 20.11 | - |
| Random Policy | 21.99 | - |

methods is less prominent. This can be explained by the fact that the drift state is the same for all of the methods. Furthermore, from this figure it also becomes clear that the decrease in observed performance for the relative state FNN method in figure 6 is caused by an increase in drift angle over time.

Looking at the results given in table VIII, the absolute methods attain a lower drift than the relative methods, but the relative difference for the lowest obtained drift between the different methods is less notable than for the reward and number of intrusions.

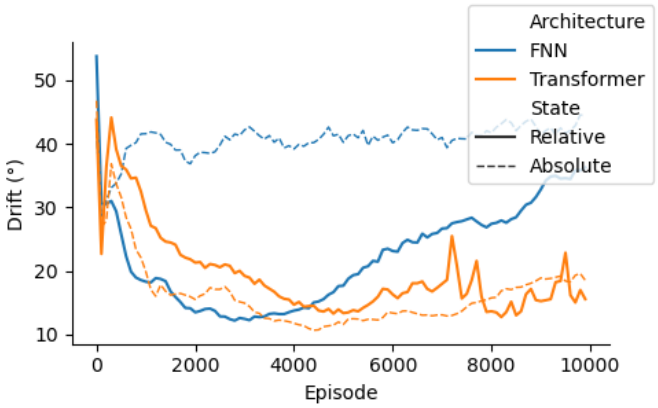


Figure 8. Evolution of the average drift over time for the different methods.

TABLE VIII. DRIFT ERROR DURING TRAINING FOR THE DIFFERENT METHODS

| Method | Min drift (ao100) | Episode |
|---|---|---|
| Relative state FNN | 12.14 | 2825 |
| Absolute state FNN | 27.66 | 132 |
| Relative state Transformer | 12.58 | 8231 |
| Absolute state Transformer | 10.49 | 4360 |
| *Reference values* | | |
| No Drift Policy | 0.00 | - |
| Random Policy | 54.15 | - |

## V. DISCUSSION

The results show that the highest performance is obtained when using the relative state FNN method. This observation contradicts the hypothesis that Transformer Network Architectures will outperform FNN methods. The initial learning curve for the FNN method being steeper than the Transformer method can be explained by the random initialization of the

attention mechanisms for the Transformer method. Unlike the FNN method, for which the observation vectors are handcrafted, the initial observation vectors for the Transformer method are essentially random. Because of this, a slower initial learning rate is expected to allow the attention mechanisms to learn what information is relevant to the latent space. However, it was expected that this learned latent space would result in higher asymptotic performance for the Transformer method than the FNN method, which is not observed.

It is, in theory, possible that the Transformer model will still obtain a better performance if the number of episodes is increased, as indicated by the episode numbers at which the best performance for the different metrics are observed (2661, 1213 and 2825 for the FNN and 8149, 9380 and 8231 for the Transformer), and the fact that training for the Transformer model appears stable throughout the 10.000 episodes, something that is not observed for the FNN method. This, however, does increase the overall training and computation time, which might be undesirable. Instead, it might be better to focus more on the attention mechanisms underlying the Transformer model, as they control the construction of the latent space used for determining the actions.

The original paper introducing the Transformer architecture mentions the usage of additive attention instead of scaled dot-product attention as an alternative to the attention mechanisms. Considering that, in contrast to word tokens, the tokens used in this research contain actual physical information on the problem, it could be that additive attention yields more logical results from a physical perspective (e.g. $x_i - x_j$ containing more useful information than $x_i \cdot x_j$).

Furthermore, constantly changing attention mechanisms weights create a degree of stochasticity for the latent space from the perspective of the Fully Connected Layers. Because of this, it is more difficult to fine-tune the weights responsible for action selection when compared to a deterministic input vector as used for the FNN methods. One way to counteract this is by either decreasing the learnrate for the attention mechanisms over time or freezing the attention mechanism weights after a certain period. The latter has as a consequence that the optimal attention mechanism is dependent on the policy of the method, and will result in a lower theoretical performance limit.

Additionally, as this was an initial attempt at using Transformer architectures in conjunction with DRL the obtained performance is likely not yet at the maximum attainable limit. However, future studies on this topic should also compare the results with RNNs of similar model size to identify the performance differences between the different methods. This might then provide a more conclusive result on whether or not Transformers have any useful role in the field of MARL over RNNs and FNNs.

Finally the method should be tested under different scenarios and different higher fidelity simulators, to see how the attention mechanisms adapt and compare against the other methods. This is relevant as aircraft dynamics, or scenarios such as merging paths, require the attention mechanism to consider longer time horizons, which complicates the decision process.

## VI. Conclusion

This paper introduced an alteration to the attention mechanisms of Transformer models to allow them to operate on relative state information. It was found that this alteration improved the performance when compared with Transformers operating on the absolute state information of the surrounding agents. The final obtained performance however did not exceed that of a simple FNN structure which used a handcrafted observation vector with the relative state information of the 3 surrounding aircraft.

It is possible that with more training the performance of the Transformer method will exceed that of the FNN method, as performance improvements for the FNN method stagnated at a much earlier stage than the Transformer. However, as increasing the duration of training comes at a cost of more computational expense, it is instead suggested to further investigate the underlying attention mechanisms.

Future research should investigate whether methods that only rely on attention mechanisms, such as Transformers, can obtain better performance in the field of MARL for safe navigation than traditional FNNs and RNNs. Nevertheless, they might still have a use-case for problems where selecting the important aircraft is not as straightforward, or the problem requires a fully observable environment and where the output should be independent of the input sequence.

## References


[1] Z. Wang, W. Pan, H. Li, X. Wang, and Q. Zuo, "Review of deep reinforcement learning approaches for conflict resolution in air traffic control," *Aerospace*, vol. 9, no. 6, p. 294, 2022.

[2] P. Zhao and Y. Liu, "Physics informed deep reinforcement learning for aircraft conflict resolution," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 8288–8301, 2021.

[3] M. W. Brittain and P. 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.

[4] R. Dalmau and E. Allard, "Air traffic control using message passing neural networks and multi-agent reinforcement learning," *Proceedings of the 10th SESAR Innovation Days, Virtual Event*, pp. 7–10, 2020.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[7] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[8] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen *et al.*, "Palm 2 technical report," *arXiv preprint arXiv:2305.10403*, 2023.

[9] M. W. Brittain, X. Yang, and P. Wei, "Autonomous separation assurance with deep multi-agent reinforcement learning," *Journal of Aerospace Information Systems*, vol. 18, no. 12, pp. 890–905, 2021.

[10] M. W. Brittain, L. E. Alvarez, and K. Breeden, "Improving autonomous separation assurance through distributed reinforcement learning with attention networks," *arXiv preprint arXiv:2308.04958*, 2023.

[11] J. Sola and J. Sevilla, "Importance of input data normalization for the application of neural networks to complex industrial problems," *IEEE Transactions on nuclear science*, vol. 44, no. 3, pp. 1464–1468, 1997.

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

[13] C. Badea, D. Groot, A. M. Veytia, M. Ribeiro, J. Ellerbroek, J. Hoekstra, and R. Dalmau, "Lateral and vertical air traffic control under uncertainty using reinforcement learning," in *12th SESAR Innovation Days*, 2022.

[14] J. Groot, M. Ribeiro, J. Ellerbroek, and J. Hoekstra, "Improving safety of vertical manoeuvres in a layered airspace with deep reinforcement learning," in *Proceedings of the 10th International Conference for Research in Air Transportation (ICRAT), Tampa, FL, USA*, 2022, pp. 19–23.