# Hybrid Transformer Based Multi-Agent Reinforcement Learning for Multiple Unpiloted Aerial Vehicle Coordination in Air Corridors

Liangkun Yu, *Student Member, IEEE*, Zhirun Li, Nirwan Ansari, *Life Fellow, IEEE*, and Xiang Sun, *Member, IEEE*

***Abstract*—Advanced Air Mobility (AAM) seeks to establish a next-generation air transportation system by leveraging autonomous unpiloted aerial vehicles (UAVs) to transport passengers and cargo between locations previously underserved or unserved by traditional aviation. Achieving AAM at scale requires overcoming significant challenges in airspace management, classification, and traffic control to safely accommodate the increasing volume of UAV operations. This paper presents a comprehensive design for air corridors to facilitate efficient aerial transport and formulates a multi-UAV coordination problem within these corridors. The objective is to enable each UAV to autonomously make control decisions based on local observations gathered from onboard sensors. This decentralized control approach is modeled as a multi-agent partially observable Markov decision process (POMDP), aiming at minimizing UAV travel time while ensuring adherence to corridor boundaries and collision avoidance. To address the complexities posed by varying state dimensions and types, we propose a novel Hybrid Transformer-based Multi-agent Reinforcement Learning (HTransRL) architecture. HTransRL integrates a customized transformer model into an actor-critic network, effectively processing both sequential and non-sequential observed states of varying sizes while capturing their correlations. This enables safe and efficient UAV navigation. Simulation results show that in test environments similar to or simpler than training scenarios, HTransRL achieves a successful arrival rate exceeding 90% in worst-case test scenarios. In test environments more complex than training scenarios, HTransRL demonstrates superior scalability compared to two baseline methods, achieving higher arrival rates and comparable travel times.**

***Index Terms*—Reinforcement learning, transformer, PPO, autonomous control, UAV, air corridor.**

Received 27 June 2024; revised 24 December 2024; accepted 17 January 2025. Date of publication 21 January 2025; date of current version 7 May 2025. This work was supported in part by the National Science Foundation under Grant CNS-2430773, Grant OIA-2417062, Grant CNS-2323050, and Grant CNS-2148178, and Grant CNS-2148178, in part by funds from federal agency and industry partners as specified in the Resilient & Intelligent NextG Systems (RINGS) program. Recommended for acceptance by D. Niyato. *(Corresponding author: Xiang Sun.)*

Liangkun Yu, Zhirun Li, and Xiang Sun are with the SECNet Laboratory, Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM 87131 USA (e-mail: liangkun@unm.edu; zhirunli@unm.edu; sunxiang@unm.edu).

Nirwan Ansari is with the Advanced Networking Laboratory, Department of Electrical and Computing Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: nirwan.ansari@njit.edu).

The code for HTransRL is available at https://github.com/SECNetLabUNM/HTransRL

Digital Object Identifier 10.1109/TMC.2025.3532204

## I. Introduction

As the utilization of Unpiloted Aerial Vehicles (UAVs) continues to rise across various sectors, NASA and the FAA are collaborating to develop an advanced air transportation system dedicated to UAVs, enhancing their capabilities for efficient cargo delivery and passenger transportation. In densely populated urban airspace, UAVs are expected to autonomously adhere to specific flight regulations to ensure smooth and safe air traffic flow, while minimizing potential collisions. To meet these requirements, meticulously designed three-dimensional (3D) aerial highways, known as air corridors, are essential. These corridors should facilitate efficient transportation while upholding rigorous safety standards, serving as designated routes for UAVs traveling between vertiports [1]. The current state-of-the-art airspace management and air corridor design is still in its infancy, lacking detailed configurations. Building upon our previous work [2], we present our comprehensive design for air corridor structures, aimed at facilitating efficient movement of UAVs and air traffic management.

Given the air corridors, which essentially constrain the trajectory of a UAV from its source to its destination vertiport, one of the major challenges is controlling the UAVs in air corridors to meet three objectives: 1) to navigate within the predefined air corridors, 2) to avoid collisions in complex airspace, where some non-cooperative flying objects (NCFOs), such as birds and amateur drones, may exist, 3) to promptly arrive at their destinations. These objectives may conflict with each other, meaning that if a UAV aims to minimize its travel time to the destination, it may increase the probability of collisions or boundary breaches within air corridors. Hence, it is critical but challenging to optimize this tradeoff.

Existing automated aviation navigation systems are primarily implemented based on centralized control [3], [4], [5], where various aircraft and external sensing systems, such as ground radar or lidar systems, sense the airspace states. This information is then transmitted to a centralized controller, which processes the data received to generate actions (e.g., accelerations) for various aircraft. These actions are then relayed back to the corresponding aircraft to ensure efficient navigation and collision avoidance. Although the centralized control solution is capable of obtaining the global view of the whole airspace to potentially derive optimal actions, it suffers from scalability limitations and

1536-1233 © 2025 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Southeast University. Downloaded on November 10,2025 at 11:32:51 UTC from IEEE Xplore. Restrictions apply.

communications latency/failure challenges, and is thus unable to control UAVs in air transportation systems for Advanced Air Mobility (AAM), which attempts to manage UAVs in crowded airspace. Therefore, ensuring high scalability and tolerance to communication latency/failure is critical.

Decentralized control solutions are developed to enable each UAV to autonomously determine its actions based on some prior knowledge, such as the characteristics of its air corridors, and local observations, such as the velocities and locations of neighboring UAVs or NCFOs that are located within the UAV's sensing area, whose size is determined by its onboard sensors. Although decentralized control resolves the scalability issue and unsafe actions caused by communications latency/failure challenges, it requires UAV policies capable of collaboratively navigating air corridors to avoid collisions and boundary crossings based solely on local views. Intuitively, the decentralized control problem for a UAV in air corridor navigation can be formulated as a multi-agent partially observable Markov decision process (MAPOMDP) [6], [7]. Here, partially observable means that a UAV only has a local view within its sensing area, besides the prior static global knowledge, such as the structures of its current and next air corridors. In general, deep reinforcement learning (DRL) has been demonstrated as an efficient way to solve a Markov decision process.

However, the partially observable nature of the problem presents a challenge for utilizing DRL. Specifically, the dimension of observations varies over time due to fluctuations in the number of neighboring UAVs and NCFOs within the UAV's sensing area. This dynamic observation dimension results in varying input sizes, which traditional DRL algorithms cannot easily handle because multilayer perceptron (MLP) models, adopted by these algorithms, cannot adapt to varying input sizes. Normally, if the observation dimension is smaller than the input dimension of an MLP, padding the remaining input lines with zeros is an effective solution. Yet, if the observation dimension is larger than the input dimension, a common but ineffective solution is to drop some observations (e.g., the states of neighboring UAVs/NFCOs that are far away from the UAV) to accommodate the MLP's input size [8], [9]. This approach could lead to bad actions that finally lead to, for example, collisions. On the other hand, the transformer has been widely used in natural language processing to handle a sentence or a sequence of words with variable lengths.

Inspired by that, we propose the Hybrid Transformer based Multi-agent Reinforcement Learning (HTransRL) framework, which customizes the transformer framework based on the unique features of the air corridor traffic management application, and then integrates it into DRL. HTransRL is capable of not only handling the dynamic dimension of observations with sequential and non-sequential vectors but also efficiently analyzing their relationships to benefit the actor-critic network in DRL, deriving better policies to meet the three objectives. The major contributions of the paper are summarized below.

1) We propose a detailed air corridor design by modeling horizontal lanes as truncated cylinders and on-off ramps as partial tori. A truncated cylinder is defined by four parameters: anchor point, orientation, length, and radius, while a partial torus is characterized by five parameters: anchor point, orientation, tube radius, central point of the end plane, and the angle between the start and end planes. Unlike previous work, which often relies on simplified shapes [10] or 2D paths [11], our 3D design models realistic airspace, enabling effective air traffic management, smooth transitions between corridors, and support for the spatial complexity required in dense airspaces.
2) We formulate multiple UAV coordination in air corridors as an optimization problem, and propose the HTransRL framework to solve the problem efficiently and distributively. HTransRL customizes the transformer framework to efficiently capture the relations among a UAV's observations, which vary in size over time and include both sequential and non-sequential data. This customized transformer is then integrated into a DRL algorithm to ensure efficient and safe UAV navigation.
3) To improve the training efficiency, curriculum learning is customized and applied in training HTransRL. Extensive test simulations are conducted to demonstrate the performance and scalability of HTransRL.

The remainder of this paper is organized as follows: Section II reviews existing works on handling states with dynamic dimensions in DRL. Section III presents system models and problem formulation for multiple UAV coordination in air corridors. Section IV details the design of HTransRL. Section V demonstrates and analyzes HTransRL's performance through simulations. Finally, Section VI concludes the paper.

## II. Related Work

Deep reinforcement learning (DRL) has recently gained prominence as a powerful method for UAV control, particularly in complex and dynamic environments [12], [13], [14]. In multi-agent systems, advanced machine learning techniques have been employed to process high-dimensional, dynamic state information critical for achieving objectives such as collision avoidance. However, DRL models, typically implemented as multi-layer perceptrons, struggle to efficiently process inputs of varying sizes, as they are designed to handle fixed-size inputs. To address this challenge, Qin et al. [8] proposed a method where each agent selects a fixed number of the nearest neighboring agents' states as inputs, discarding the rest. This approach assumes that shorter distances are more likely to lead to collisions, but it overlooks other crucial factors such as velocity and acceleration. As shown in the simulation results in Section V, this method results in a high collision rate. Effectively addressing the challenge of varying input sizes is crucial for improving model performance, robustness, and scalability in real-world applications. In the following, we summarize and categorize existing solutions to this challenge into three key approaches.

### *A. Long Short-Term Memory Networks (LSTMs) for Varying Input Sizes*

Long Short-Term Memory (LSTM) networks, a type of recurrent neural network (RNN), are designed to model sequential

Authorized licensed use limited to: Southeast University. Downloaded on November 10,2025 at 11:32:51 UTC from IEEE Xplore. Restrictions apply.

data by maintaining memory cells that capture long-term dependencies [15]. LSTMs handle varying input sizes by processing input sequences one step at a time, while maintaining internal states that adapt to different sequence lengths. This capability enables the network to effectively capture temporal dependencies in dynamic inputs. LSTMs have proven useful in dynamic UAV environments, where input dimensionality fluctuates over time. For instance, Li and Wu [16] proposed integrating LSTMs to enhance environmental state approximation with varying sizes, leveraging historical observations to adapt to varying sequence lengths. This allows UAVs to accommodate dynamically changing observations, particularly in partially observable settings where input dimensions vary due to moving targets and environmental changes. Deniz et al. [17] applied LSTMs to process the states of neighboring UAVs modeled as a sequential vector, encoding the sequence into a fixed-length vector [17], which is then used as input to a conventional DRL model.

However, two significant limitations affect the applicability of LSTMs in this context. First, LSTMs are susceptible to the vanishing gradient problem, as discussed in [18], where longer sequences lead to a loss of gradient information, making it difficult to effectively learn from earlier elements in the sequence. Second, treating neighboring UAV states as a sequential vector can lead to order-dependent results, which is problematic for our application. For instance, if a UAV observes two neighboring UAVs, UAV-1 and UAV-2, it should generate the same action to avoid a collision regardless of their order in the input sequence. LSTM's sensitivity to sequence order can cause inconsistencies in action selection, i.e., $f^{LSTM}(UAV_1, UAV_2) \neq f^{LSTM}(UAV_2, UAV_1)$, where $f^{LSTM}(\cdot)$ is the function achieved by an LSTM.

## B. Deep Sets (DSs) for Varying Input Sizes

The DS model [19] was proposed to preprocess inputs of varying size while ensuring permutation invariance, i.e., $f^{DS}(UAV_1, UAV_2) = f^{DS}(UAV_2, UAV_1)$, where $f^{DS}(\cdot)$ is the function achieved by a DS model. This approach uses fully connected neural networks to extract features from the environment's states, which are then passed through a permutation-invariant function, such as row-wise max pooling, to produce a fixed-length vector representation [20], [21]. In multi-UAV tasks, where the focus is on UAV interactions rather than their observation order, DS can summarize spatial features into a fixed-size vector, regardless of the number of UAVs. This fixed-size representation can then be used by the DRL model to generate effective actions for various applications [22], [23].

However, a key limitation of the DS model is its dependence on a specific permutation-invariant function, which may overlook critical information required for effective collision avoidance. This rigid approach limits the model's flexibility and may fail to capture essential interactions, thereby reducing its effectiveness in dynamic UAV control scenarios.

## C. Transformers for Varying Input Sizes

Rather than relying on a simple permutation-invariant function, the transformer (or attention model) calculates attention scores to capture correlations among different states. The attention scores are then fed into a DRL model to generate actions for UAVs, enabling efficient and safe navigation. Lee et al. [24] proposed the Set Transformer, a permutation-invariant deep learning model designed to process dynamically sized inputs and outputs important correlations among the inputs. Hsu et al. [25] incorporated the transformer model into the Double Deep Q-Learning Network (DDQN) for counter-UAV applications, where a group of pursuer UAVs tracks and captures a group of evader UAVs. In this setup, the transformer preprocesses the observed or estimated states of the pursuer UAVs, and DDQN uses this information to derive the optimal action that maximizes the Q-value. In general, as compared to LSTMs and DSs, transformers excel in processing variable-sized inputs by efficiently capturing both local and global dependencies, overcoming the sequential processing limitations of LSTMs and the fixed aggregation constraints of DSs.

In summary, leveraging a transformer model to preprocess the observed states of an agent with dynamic size has shown outstanding performance. However, integrating the transformer into DRL for autonomous and distributive traffic management in air corridors remains unexplored and challenging due to the unique features of this application. First, each UAV must ensure its position within the air corridors, necessitating the inclusion of air corridor characteristics should be included in the input states to determine the UAV's actions. Unlike other state types that are permutation invariant or non-sequential (such as neighboring UAV states), the sequence of air corridor states is crucial for ensuring UAVs stay within the corridors, especially during transitions between contiguous corridors. Consequently, the observed states of a UAV comprise a mix of sequential and non-sequential vectors, and designing a transformer model to handle these hybrid states is an unresolved issue.

To address this, we propose a straightforward yet effective solution described in Section IV-A, which employs one-hot encoding for the sequential states related to UAV $i$'s air corridor characteristics. Second, the airspace is complex, with many NCFOs often present but ignored in previous works. These NCFOs are not controlled by the developed DRL models and may behave unpredictably, thus increasing the complexity of the state space. PPO, the DRL model we used, requires effective exploration of the state and action spaces. However, misaligned transformer outputs can impede this process, resulting in extended training durations or even divergence. To address this challenge, we implement curriculum learning, as detailed in Section IV-C, by incorporating intermediate rewards to facilitate efficient model training. Although our previous work [26] developed a transformer-based DRL model for traffic management in air corridors, it did not incorporate these unique features into the model design and simulations. Here, our proposed HTransRL customizes and incorporates the transformer framework based on the two features into the DRL model.

## III. System Models and Problem Formulation

Denote $\mathcal{I}$ as the set of UAVs in the airspace, where $i$ is the index of these UAVs. Each UAV possesses prior knowledge

Authorized licensed use limited to: Southeast University. Downloaded on November 10,2025 at 11:32:51 UTC from IEEE Xplore. Restrictions apply.

TABLE I
LIST OF KEY NOTATIONS

| Notation | Description |
|---|---|
| $\boldsymbol{\mathcal{I}}$ | Set of UAVs in the airspace |
| $i$ | Index of the UAVs in $\boldsymbol{\mathcal{I}}$ |
| $t$ | Index of time slots |
| $\boldsymbol{\mathcal{K}}_i(t)$ | Set of objects observed by UAV $i$ at time slot $t$ |
| $k$ | Index of the objects in $\boldsymbol{\mathcal{K}}_i(t)$ |
| $\boldsymbol{p}_i(t)/\boldsymbol{p}_k(t)$ | Position of UAV $i$/Object $k$ at time slot $t$ |
| $\boldsymbol{v}_i(t)/\boldsymbol{v}_k(t)$ | Velocity of UAV $i$/Object $k$ at time slot $t$ |
| $\boldsymbol{a}_i(t)$ | Acceleration of UAV $i$ at time slot $t$ |
| $\varrho_i$ | Radius of UAV $i$ |
| $\varrho_k$ | Radius of Object $k$ |
| $\boldsymbol{c}_i^{cyl}(t)$ | Anchor point of UAV $i$'s residing cylinder at time slot $t$ |
| $\boldsymbol{d}_i^{cyl}(t)$ | Orientation of UAV $i$'s residing cylinder at time slot $t$ |
| $l_i^{cyl}(t)$ | Length of UAV $i$'s residing cylinder at time slot $t$ |
| $r_i^{cyl}(t)$ | Radius of UAV $i$'s residing cylinder at time slot $t$ |
| $\boldsymbol{M}_i^{cyl}(t)$ | Transformation matrix w.r.t UAV $i$'s residing cylinder |
| $\boldsymbol{c}^{tor}(t)$ | Anchor point of UAV $i$'s residing torus at time slot $t$ |
| $\boldsymbol{d}^{tor}(t)$ | Orientation of UAV $i$'s residing torus at time slot $t$ |
| $r^{tor}(t)$ | Tube radius of UAV $i$'s residing torus at time slot $t$ |
| $e^{tor}(t)$ | Center of the end plane of UAV $i$'s residing torus |
| $\mu_{tor}(t)$ | Angle between the two planes of UAV $i$'s residing torus |
| $\boldsymbol{M}_i^{tor}(t)$ | Transformation matrix w.r.t UAV $i$'s residing torus |
| $t_i^{travel}$ | Overall travel time of UAV $i$ to its destination |
| $v^{max}$ | Maximum speed of a UAV |
| $a^{max}$ | Maximum acceleration of a UAV |
| $\rho_i(t)$ | Magnitude of UAV $i$'s acceleration at time slot $t$ |
| $\theta_i(t)$ | Polar angle of UAV $i$'s acceleration at time slot $t$ |
| $\phi_i(t)$ | Azimuthal angle of UAV $i$'s acceleration at time slot $t$ |

of its source and destination planes, as well as the trajectory specifying which air corridors it should traverse from the source to destination planes. Table I summarizes the key notations used in the system models and problem formulation.

### A. Aerodynamic and Collision Model of a UAV

Assume that a 3D Cartesian coordinate system is applied and each UAV is represented as a sphere. UAV $i$'s position at time slot $t$ is the center of the sphere, denoted as $\boldsymbol{p}_i(t) \in \mathbb{R}^3$, i.e., $\boldsymbol{p}_i(t) = [p_i^x(t), p_i^y(t), p_i^z(t)]$. Also, let $\boldsymbol{v}_i(t)$ and $\boldsymbol{a}_i(t)$ be the velocity and acceleration of UAV $i$ at time slot $t$, where $\boldsymbol{v}_i(t), \boldsymbol{a}_i(t) \in \mathbb{R}^3$. The aerodynamics of UAV $i$ follow $\boldsymbol{v}_i(t+1) = \boldsymbol{v}_i(t) + \boldsymbol{a}_i(t)\Delta t$ and $\boldsymbol{p}_i(t+1) = \boldsymbol{p}_i(t) + \frac{\boldsymbol{v}_i(t)+\boldsymbol{v}_i(t+1)}{2} \times \Delta t$, where $\Delta t$ is the duration of a time slot. We use these simple aerodynamic models of a UAV in our simulations. However, our proposed HTransRL model can accommodate more sophisticated aerodynamic models, though it may require retraining or fine-tuning.

Each NCFO is also represented as a sphere, where the position of NCFO $k$ at time slot $t$, denoted as $\boldsymbol{p}_k$, is the sphere's center. Assume that there are two types of NCFOs: 1) static NCFOs, which are statically deployed within the air corridors, and 2) mobile NCFOs, each flying from its source to destination waypoints with constant velocity $\boldsymbol{v}_k$. Upon reaching the destination waypoint, NCFO $k$ randomly selects a new destination waypoint within the vicinity of the air corridors.

With respect to the collision between two UAVs represented as two spheres, a collision occurs if the distance between the centers of two spheres is less than or equal to the sum of the radii of the two spheres, i.e., $\|\boldsymbol{p}_i(t) - \boldsymbol{p}_{i'}(t)\| \leqslant \varrho_i + \varrho_{i'}$, where $\varrho_i$ and $\varrho_{i'}$ are the radii of UAVs $i$ and $i'$. Similarly, the collision between UAV $i$ and NFCO $k$ is modeled as $\|\boldsymbol{p}_i(t) - \boldsymbol{p}_k(t)\| \leqslant \varrho_i + \varrho_k$, where $\boldsymbol{p}_k(t)$ is the position of NFCO $k$ at time slot $t$ and $\varrho_k$ is the radius of NFCO $k$. Without loss of generality, we assume that the radii of UAVs and NCFOs are the same, i.e., $\varrho = \varrho_i = \varrho_k$. Hence, to avoid collisions for UAV $i$ at time slot $t$, the following inequality should be met.

$$\forall k \in \boldsymbol{\mathcal{K}}_i(t), \|\boldsymbol{p}_i(t) - \boldsymbol{p}_k(t)\| \leqslant 2\varrho, \tag{1}$$

where $\boldsymbol{\mathcal{K}}_i(t)$ is the set of objects, including other UAVs and NCFOs, observed by UAV $i$ at time slot $t$.

### B. Structural Models of Air Corridors

Air corridor design is critical to efficiently and safely manage traffic in crowded airspace. Followed by our previous works [2], the whole airspace is divided into several horizontal layers with different altitudes. An air corridor is considered a one-way highway in the airspace, dictating where UAVs are permitted to fly. There are two types of air corridors: 1) horizontal lanes, which are organized in horizontal layers without intersection, and 2) on-off ramps, which connect these horizontal lanes to allow UAVs to change directions/altitudes. Fig. 1 provides an example of two horizontal lanes in different layers connected by an on-off ramp. In the following, we will provide mathematical models to characterize the horizontal lanes and on-off ramps.

*Horizontal lanes:* A horizontal lane is modeled as a truncated cylinder, which can be characterized by the following four parameters: 1) anchor point $\boldsymbol{c}^{cyl}$, which is the center of the truncated cylinder, 2) orientation $\boldsymbol{d}^{cyl}$, which is the direction of the truncated cylinder, 3) length $l^{cyl}$, which is the distance between the anchor point and the center point of start/end plane, and 4) radius $r^{cyl}$.

To facilitate the calculations and reduce the state space, if UAV $i$ is currently in a horizontal lane, we transform this horizontal lane into a standard truncated cylinder with anchor point $\bar{\boldsymbol{c}}^{cyl} = [0, 0, 0]$ and orientation $\bar{\boldsymbol{d}}^{cyl} = [0, 0, 1]$, i.e.,

$$\boldsymbol{M}_i^{cyl}(t)\left[\boldsymbol{c}_i^{cyl}(t), \boldsymbol{d}_i^{cyl}(t)\right]^T = \left[\bar{\boldsymbol{c}}^{cyl}, \bar{\boldsymbol{d}}^{cyl}\right]^T, \tag{2}$$

where $\boldsymbol{c}_i^{cyl}(t)$ and $\boldsymbol{d}_i^{cyl}(t)$ are the anchor point and orientation of UAV $i$'s current residing horizontal lane in the Cartesian coordinate system at current time slot $t$, and $\boldsymbol{M}_i^{cyl}(t) \in \mathbb{R}^{3\times 3}$ is the transformation matrix to convert this horizontal lane into the standard truncated cylinder. As long as $\boldsymbol{c}_i^{cyl}(t)$ and $\boldsymbol{d}_i^{cyl}(t)$ are known, $\boldsymbol{M}_i^{cyl}(t)$ can be calculated based on (2). Accordingly, the positions and velocities of all the UAVs and NCFOs as well as UAV $i$'s previous and next air corridors should also be transformed based on $\boldsymbol{M}_i^{cyl}(t)$. For example, UAV $i$'s position and velocity are updated via $\boldsymbol{p}_i^T(t) := \boldsymbol{M}_i^{cyl}(t)\boldsymbol{p}_i^T(t)$ and $\boldsymbol{v}_i^T(t) := \boldsymbol{M}_i^{cyl}(t)\boldsymbol{v}_i^T(t)$. The right side of Fig. 1 illustrates the conceptual process of transforming Air corridor-4 into the standard truncated cylinder. *Note that if UAV $i$ is in a truncated cylinder, unless otherwise specified, the positions, velocities, and accelerations of the UAVs and NCFOs, as well as the rest of the*

Authorized licensed use limited to: Southeast University. Downloaded on November 10,2025 at 11:32:51 UTC from IEEE Xplore. Restrictions apply.

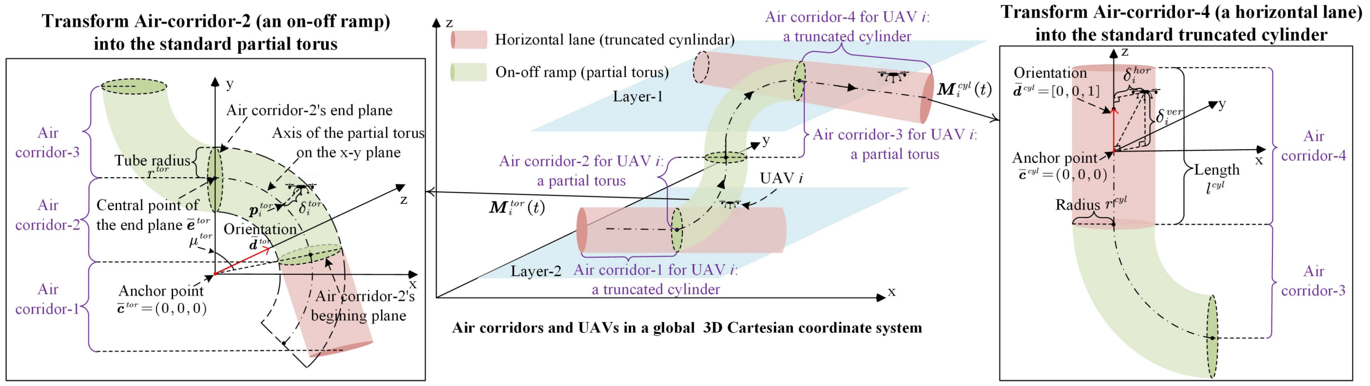

Transform Air-corridor-2 (an on-off ramp) into the standard partial torus
Air corridor-3
Air corridor-2
Air corridor-1
Tube radius $r^{tor}$
Air corridor-2's end plane
Axis of the partial torus on the x-y plane
Central point of the end plane $\bar{e}^{tor}$
$\mu^{tor}$
Orientation $\bar{d}^{tor}$
$p_i^{tor}$
$\delta_i^{tor}$
Anchor point $\bar{c}^{tor}=(0,0,0)$
Air corridor-2's begining plane
Horizontal lane (truncated cynlindar)
On-off ramp (partial torus)
Layer-1
Layer-2
Air corridor-4 for UAV $i$: a truncated cylinder
Air corridor-3 for UAV $i$: a partial torus
Air corridor-2 for UAV $i$: a partial torus
Air corridor-1 for UAV $i$: a truncated cylinder
UAV $i$
$M_i^{tor}(t)$
$M_i^{cyl}(t)$
Air corridors and UAVs in a global 3D Cartesian coordinate system
Transform Air-corridor-4 (a horizontal lane) into the standard truncated cylinder
Orientation $\bar{d}^{cyl}=[0,0,1]$
$\delta_i^{hor}$
$\delta_i^{ver}$
Anchor point $\bar{c}^{cyl}=(0,0,0)$
Length $l^{cyl}$
Radius $r^{cyl}$
Air corridor-4
Air corridor-3

Fig. 1. Illustration of air corridor design: 1) The central subfigure shows UAV $i$'s trajectory (dashed-dotted line) traversing four air corridors in a global 3D Cartesian coordinate system. Air corridor-1 and Air corridor-4 are modeled as truncated cylinders, while Air corridor-2 and Air corridor-3 are modeled as partial tori. 2) The left subfigure illustrates the conversion of Air corridor-2 from a partial torus in the global Cartesian coordinate system to a standard partial torus in a local Cartesian coordinate system. 3) The right subfigure illustrates the conversion of Air corridor-4 from a truncated cylinder in the global Cartesian coordinate system to a standard truncated cylinder in a local Cartesian coordinate system.

*air corridors for UAV i, are referenced in the standard truncated cylinder for the remainder of the paper.*

*On-off ramps:* An on-off ramp is modeled as a partial torus or two connected partial tori. For example, an on-off ramp in Fig. 1 comprises two partial tori, each being characterized by the following four parameters: 1) anchor point $\boldsymbol{c}^{tor}$, which is the center of the partial tori, 2) orientation $\boldsymbol{d}^{tor}$, which is the direction staring from the center of the torus around which it rotates (right-hand thumb rule), 3) tube radius $r^{tor}$, which is the perpendicular distance between the central path and the edge of the partial torus, 4) central point of the end plane $\boldsymbol{e}^{tor}$, and 5) angle $\mu^{tor}$ between the line from the central point of the end plane to the anchor point and the line from the central point of the beginning plane to the anchor point.

Similarly, if UAV $i$ is currently in an on-off ramp, we transform this ramp into the standard partial torus with anchor point $\bar{\boldsymbol{c}}^{tor} = [0,0,0]$, orientation $\bar{\boldsymbol{d}}^{tor} = [0,0,1]$, and the central point of the end plane $\bar{\boldsymbol{e}}^{tor}$ located on the $y$-axis, i.e.,

$$\boldsymbol{M}_i^{tor}(t)\left[\boldsymbol{c}_i^{tor}(t), \boldsymbol{d}_i^{tor}(t), \boldsymbol{e}_i^{tor}(t)\right]^T = \left[\bar{\boldsymbol{c}}^{tor}, \bar{\boldsymbol{d}}^{tor}, \bar{\boldsymbol{e}}^{tor}\right]^T, \tag{3}$$

where $\boldsymbol{c}_i^{tor}(t)$, $\boldsymbol{d}_i^{tor}(t)$, and $\boldsymbol{e}_i^{tor}(t)$ are the anchor point, orientation, and central point of the end plane for UAV $i$'s current residing on-off ramp, respectively, $\bar{\boldsymbol{e}}^{tor} = [0, \|\boldsymbol{e}_i^{tor}(t)\|, 0]$, and $\boldsymbol{M}_i^{tor}(t) \in \mathbb{R}^{3\times 3}$ is the transformation matrix to convert this ramp into the standard partial torus. As long as $\boldsymbol{c}_i^{tor}(t)$, $\boldsymbol{d}_i^{tor}(t)$, and $\boldsymbol{e}_i^{tor}(t)$ are known, $\boldsymbol{M}_i^{tor}(t)$ can be calculated based on (3). Accordingly, the positions and velocities of all the UAVs and NCFOs as well as UAV $i$'s previous and next air corridors should also be transformed based on $\boldsymbol{M}_i^{tor}(t)$. The left side of Fig. 1 illustrates the conceptual process of transforming Air corridor-2 into the standard partial torus.

*Note that if UAV i is in a partial torus, unless otherwise specified, the positions, velocities, and accelerations of the UAVs and NCFOs, as well as the rest of the air corridors for UAV i, are referenced in the standard partial torus for the remainder of the paper.* We apply two transition matrices, $\boldsymbol{M}_i^{cyl}(t)$ and $\boldsymbol{M}_i^{tor}(t)$, to map the truncated cylinder and partial torus, where UAV $i$ is currently residing, from a global coordinate system to a local standard coordinate system. The motivation to implement this transition is to reduce the state space required to represent the truncated cylinder and partial torus. For example, two truncated cylinders with identical shapes (same orientation, length, and radius) but different anchor points would be treated as separate entities in the global coordinate system, but are equivalent when transformed into the local standard coordinate system. Importantly, this transition preserves all critical information needed for controlling UAVs, as they only need to consider their relative positions with respect to the boundaries of their respective air corridors.

## C. UAV Crossing Air Corridor Boundary

One objective of UAV control is to ensure they navigate within their predefined air corridors. Therefore, it is crucial to evaluate whether UAV $i$ crosses its air corridor boundary, which can be divided into two scenarios.

*UAV i in a horizontal lane:* If the horizontal lane and UAV $i$'s position are transformed into the standard truncated cylinder, determining whether UAV $i$ crosses the air corridor boundary becomes straightforward. This is done by checking two inequalities: if both are satisfied, UAV $i$ is in the air corridor at time slot $t$; otherwise, it has crossed the boundary.

$$\begin{cases} \delta_i^{ver}(t) \leqslant \frac{l_i^{cyl}(t)}{2}, \\ \delta_i^{hor}(t) \leqslant r_i^{cyl}(t), \end{cases} \tag{4}$$

where $l_i^{cyl}(t)$ and $r_i^{cyl}(t)$ are the length and radius of the truncated cylinder that UAV $i$ is located at time slot $t$, and $\delta_i^{ver}(t)$ and $\delta_i^{hor}(t)$, as shown in Fig. 1, are the vertical and horizontal distance between UAV $i$'s position at time slot $t$ and the anchor point of the standard truncated cylinder, respectively, which can

Authorized licensed use limited to: Southeast University. Downloaded on November 10,2025 at 11:32:51 UTC from IEEE Xplore. Restrictions apply.

be calculated based on

$$\begin{cases} \delta_i^{ver}(t) = \parallel \bar{\boldsymbol{d}}^{cyl} \cdot \boldsymbol{p}_i(t) \parallel, \\ \delta_i^{hor}(t) = \sqrt{\|\boldsymbol{p}_i(t)\|^2 - (\delta_i^{ver}(t))^2}, \end{cases} \tag{5}$$

where $\bar{\boldsymbol{d}}^{cyl} = [0, 0, 1]$, $\boldsymbol{p}_i(t)$ is the position of UAV $i$ at time slot $t$, and $\cdot$ is the dot product.

*UAV $i$ in an on-off ramp:* If the ramp and UAV $i$'s position are both transformed into the standard partial torus, the following two inequalities can be used to determine whether UAV $i$ crosses the air corridor boundary. If both inequalities are satisfied, UAV $i$ remains within the air corridor at time slot $t$; otherwise, it has crossed the boundary.

$$\begin{cases} \parallel \boldsymbol{b}_i^{tor}(t) - \boldsymbol{p}_i(t) \parallel \leqslant r_i^{cyl}(t), \\ \pi/2 - \mu_i^{tor}(t) \leqslant \arctan\left(p_i^y(t)/p_i^x(t)\right) \leqslant \pi/2, \end{cases} \tag{6}$$

where $\boldsymbol{b}_i^{tor}(t)$ is the perpendicular point at the axis of the standard partial torus for UAV $i$'s current position $\boldsymbol{p}_i(t)$ as shown in Fig. 1, $r_i^{cyl}(t)$ and $\mu_i^{tor}(t)$ are the tube radius and angle of the partial torus that UAV $i$ is located at time slot $t$, and $p_i^x(t)$ and $p_i^y(t)$ are the UAV $i$'s current position along the $x$ and $y$ axes, respectively. Here, $\parallel \boldsymbol{b}_i^{tor}(t) - \boldsymbol{p}_i(t) \parallel$ calculates the shortest distance between UAV $i$'s current position and the axis of the partial torus, where $\boldsymbol{b}_i^{tor}(t) = \|\bar{\boldsymbol{e}}_i^{tor}(t)\| \times \frac{\boldsymbol{p}_i(t)}{\|\boldsymbol{p}_i(t)\|}$. Also, $\arctan\left(p_i^y(t)/p_i^x(t)\right)$ calculates the angle between the $x$-axis and the line that connects the standard anchor point [0,0,0] and the point $[p_i^x(t), p_i^y(t), 0]$, which represents the projection of UAV $i$'s current location $\boldsymbol{p}_i(t)$ onto the x-y plane.

### D. Problem Formulation

In crowded airspace, different UAVs fly from their source locations to destinations via predefined air corridors. The goal of the system is to minimize the overall travel time for all the UAVs to their destinations while avoiding collisions and crossing air corridor boundaries. Hence, we formulate the multiple UAV coordination problem as follows.

$$\boldsymbol{P0}: \arg\min_{\boldsymbol{a}} \sum_{\forall i \in \boldsymbol{\mathcal{I}}} t_i^{travel}, \tag{7}$$

$$\text{s.t.,} \ \forall t, \forall i \in \boldsymbol{\mathcal{I}}, 0 \leq \|\boldsymbol{v}_i(t)\| \leq v^{\max}, \tag{8}$$

$$\forall t, \forall i \in \boldsymbol{\mathcal{I}}, 0 \leq \|\boldsymbol{a}_i(t)\| \leq a^{\max}, \tag{9}$$

$$\forall t, \forall i \in \boldsymbol{\mathcal{I}}, \forall k \in \boldsymbol{\mathcal{K}}_i(t), \|\boldsymbol{p}_i(t) - \boldsymbol{p}_k(t)\| \leqslant 2\varrho, \tag{10}$$

$$\forall t, \forall i \in \boldsymbol{\mathcal{I}}, Eq. (4) \text{ or } (6), \tag{11}$$

where $t_i^{travel}$ is the overall travel time of UAV $i$ to its destination. In $\boldsymbol{P0}$, the objective function is to minimize the travel time of all the UAVs; Constraints (8) and (9) define the feasible velocity and acceleration of a UAV, respectively, where $v^{\max}$ and $a^{\max}$ are the maximum speed and acceleration of a UAV, respectively; Constraint (10) avoids the collisions between UAVs and their observed flying objects; Constraint (11) prevents each UAV from crossing its current air corridor's boundary. Here, the choice between (4) and (6) in Constraint (11) depends on whether UAV $i$'s current air corridor is a truncated cylinder or a partial torus.

## IV. Hybrid Transformer MARL for Multiple UAV Coordination

Solving $\boldsymbol{P0}$ is difficult since there is no close-form expression to calculate $t_i^{travel}$. Deep reinforcement learning (DRL) has gained significant popularity in controlling robots, which are formulated as Markov decision processes, due to its ability to learn complex tasks and adapt to dynamic environments through trial and error. Inspired by that, we first convert $\boldsymbol{P0}$ into a multi-agent partially observable Markov decision process (MAPOMDP), which models the decision-making problem of a single UAV based on its limited observations. MAPOMDP comprises three major aspects, the observation, action, and reward of the agent in terms of UAV $i$. Specifically,

**Observations** of UAV $i$ at time slot $t$, denoted as $\boldsymbol{s}_i(t)$, represent the states of the environment that can be observed by UAV $i$ at time slot $t$. Here, $\boldsymbol{s}_i(t)$ is further divided into three parts, i.e., $\boldsymbol{s}_i(t) = \{\boldsymbol{s}_i^{self}(t), \boldsymbol{s}_i^{cor}(t), \boldsymbol{s}_i^{other}(t)\}$. Specifically, $\boldsymbol{s}_i^{self}(t)$ is **UAV $i$'s self-state**, which includes UAV $i$'s current position $\boldsymbol{p}_i(t)$ and velocity $\boldsymbol{v}_i(t)$. $\boldsymbol{s}_i^{cor}(t)$ is **the structure features of UAV $i$'s air corridors**; it is known a priori if the source and destination of UAV $i$ are determined. Having the air corridor information is critical to prevent UAV $i$ from crossing the boundary. Yet, feeding features of all the air corridors in UAV $i$'s trajectory to the control model is not necessary and may increase the size and complexity of the control model. Here, we only include the features of 4 consecutive air corridors for UAV $i$ in $\boldsymbol{s}_i^{cor}(t)$, i.e., $\boldsymbol{s}_i^{cor}(t) = \{\boldsymbol{s}_i^{cor\_pre}(t), \boldsymbol{s}_i^{cor\_cur}(t), \boldsymbol{s}_i^{cor\_next}(t), \boldsymbol{s}_i^{cor\_next2}(t)\}$, where $\boldsymbol{s}_i^{cor\_pre}(t)$, $\boldsymbol{s}_i^{cor\_cur}(t)$, $\boldsymbol{s}_i^{cor\_next}(t)$, and $\boldsymbol{s}_i^{cor\_next2}(t)$ represent the feature vector of UAV $i$'s previous, current, next one, and the one following the next air corridors, respectively, at time slot $t$. The feature vector of each air corridor has 4 elements if it is a truncated cylinder, or 5 elements if it is a partial torus, as detailed in Section III-B. $\boldsymbol{s}_i^{other}(t)$ is **states of other UAVs and NCFOs within UAV $i$'s sensing area**, which includes the positions and velocities of all these UAVs and NCFOs, i.e., $\boldsymbol{s}_i^{other}(t) = \{k \in \boldsymbol{\mathcal{K}}_i(t) | \boldsymbol{s}_k^{other}(t)\}$, where $\boldsymbol{s}_k^{other}(t) = \{\boldsymbol{p}_k(t), \boldsymbol{v}_k(t)\}$ is the state of object $k$.

**Action** of UAV $i$ at time slot $t$, denoted as $\hat{\boldsymbol{a}}_i(t)$, represents the acceleration of UAV $i$. For ease of calculation, $\hat{\boldsymbol{a}}_i(t)$ is expressed in spherical coordinates as $\hat{\boldsymbol{a}}_i(t) = [\rho_i(t), \theta_i(t), \phi_i(t)]$, representing the magnitude, polar angle, and azimuthal angle of UAV $i$'s acceleration, respectively. Yet, $\hat{\boldsymbol{a}}_i(t)$ will finally be converted into Cartesian coordinates, i.e., $\boldsymbol{a}_i(t) = [a_i^x(t), a_i^y(t), a_i^z(t)]$ based on the following equation.

$$\boldsymbol{a}_i(t) = \begin{bmatrix} a_i^x(t) \\ a_i^y(t) \\ a_i^z(t) \end{bmatrix} = a^{\max} \times \begin{bmatrix} \rho_i(t)\sin(\theta_i(t))\cos(\phi_i(t)) \\ \rho_i(t)\sin(\theta_i(t))\sin(\phi_i(t)) \\ \rho_i(t)\cos(\theta_i(t)) \end{bmatrix}. \tag{12}$$

As compared to directly generating $\boldsymbol{a}_i(t)$ by the DRL model, calculating $\hat{\boldsymbol{a}}_i(t)$ first and converting it into $\boldsymbol{a}_i(t)$ simplifies constraining $\rho_i(t), \theta_i(t)$, and $\phi_i(t)$ in $\hat{\boldsymbol{a}}_i(t)$ within their feasible domain: $\rho_i(t) \in [0, 1]$, $\theta_i(t) \in [-\pi, \pi]$, and $\phi_i(t) \in [0, \pi]$, $\forall t$, ensuring compliance with Constraint (9).

Authorized licensed use limited to: Southeast University. Downloaded on November 10,2025 at 11:32:51 UTC from IEEE Xplore. Restrictions apply.

**Rewards** of UAV $i$ at time slot $t$, denoted as $r_i(t)$, is the feedback for its action taken in a given observation at time slot $t$. $r_i(t)$ is pivotal in guiding a UAV/agent to learn and optimize its actions over time to maximize the return. Here, we define $r_i(t)$ based on the following rules:

- *Arrival:* UAV $i$ is rewarded +160 upon reaching its intended destination, emphasizing the primary objective of ensuring successful arrival.
- *Intermediate Arrival:* UAV $i$ receives a +40 reward upon reaching the end of its current air corridor. This intermediate reward enhances guidance for successfully navigating toward the destination, significantly improving training efficiency, especially when the path involves multiple air corridors.
- *Air Corridor Boundary Crossing:* UAV $i$ flying beyond its designated air corridors incurs a $-140$ penalty. This penalty aims to avoid violations of Constraint (11).
- *Collision:* UAV $i$ colliding with another UAV or NFCO results in a $-80$ penalty per incident. This penalty aims to avoid Constraint (10) violations.
- *Time Penalty:* A $-0.2$ penalty is imposed at each time slot until UAV $i$ reaches its destination, incentivizing faster travel as highlighted in Moore's study on efficient memory-based learning [27]. This time penalty aligns with the objective function in $\boldsymbol{P0}$, which aims to minimize the travel time of all UAVs.
- *Liability:* A $-10$ penalty is applied if UAV $i$ observes a collision involving another UAV. Assuming equivalent visibility for all UAVs, each UAV's actions are based on the surrounding environment. This liability penalty encourages UAV $i$ to consider the safety of neighboring UAVs, discouraging self-centered actions that could increase collision risks for others, as discussed in [28].

It is important to note that, based on our observations from numerous simulations, the exact reward values have minimal impact on reinforcement learning using Proximal Policy Optimization with Generalized Advantage Estimation (PPO-GAE), as all raw rewards are standardized before being used in the learning process.

Many DRL solutions have been developed to efficiently train policies, such as advantage actor critic [29], deep deterministic policy gradient [30], and proximal policy optimization [31]. These methods typically employ a dual neural network structure: an actor and a critic. Based on the input state values, the actor network outputs an action for UAV $i$ to maximize the return, expressed as $\sum_t \gamma^t r_i(t)$, where $\gamma$ represents the discount factor. The critic network, also utilizing state values as inputs, is focused on generating a state value $V(\boldsymbol{s}_i(t))$, which evaluates the effectiveness of the action suggested by the actor network.

Two primary obstacles prevent traditional DRL methodologies from solving the proposed MAPOMDP problem. First, a variable dimension issue arises with the observation $\boldsymbol{s}_i^{other}(t)$ in $\boldsymbol{s}_i(t)$, as this dimension fluctuates based on the quantity of other UAVs and NCFOs within UAV $i$'s observational range. This presents a significant challenge because the input dimensions for both actor and critic networks are fixed and cannot adapt to these fluctuations. Finding a method to consistently integrate the dynamic dimension of $\boldsymbol{s}_i(t)$ into the fixed dimension of these networks remains a complex problem. Second, low learning efficiency results from reward sparsity. The defined reward function $r_i(t)$ is predominantly sparse, which diminishes the efficiency of the learning process. Hence, it is critical to explore and integrate the strategies that can provide additional guidance or incentives to the agent, helping it learn more efficiently in sparse reward settings and enabling it to discover and exploit rewarding behaviors effectively.

To efficiently solve the proposed MAPOMDP problem, we design the Hybrid Transformer based multi-agent Reinforcement Learning (HTransRL) framework, which incorporates the hybrid transformer model and curriculum learning into the existing DRL model, i.e., Proximal Policy Optimization with Generalized Advantage Estimation (PPO-GAE) [31], [32]. Here, the hybrid transformer addresses the variable dimension issue, and curriculum learning resolves the sparse reward challenge. Fig. 2 shows the structure of the HTransRL framework, which comprises two models, i.e., hybrid transformer and actor-critic network, both being trained via curriculum learning.

## A. Hybrid Transformer

Transformers are widely employed in addressing sequence-to-sequence tasks, such as natural language processes (NLP) [33], leveraging the encoder-decoder structure to accommodate input sentences with different word lengths. Inspired by that, we apply the transformer structure to handle the variable dimension of UAV $i$'s observation $\boldsymbol{s}_i(t)$. By analogy, $\boldsymbol{s}_i(t)$ is considered as a sentence, while $\boldsymbol{s}_i^{self}(t)$, $\boldsymbol{s}_i^{cor}(t)$, and $\boldsymbol{s}_i^{other}(t)$ within $\boldsymbol{s}_i(t)$ represent words in a sentence. Different values of $\boldsymbol{s}_i^{self}(t)$, $\boldsymbol{s}_i^{cor}(t)$, and $\boldsymbol{s}_i^{other}(t)$ correspond to different words. We expect that the transformer structure is capable of analyzing the correlation between UAV $i$'s self-state $\boldsymbol{s}_i^{self}(t)$ and other observations, i.e., UAV $i$'s air corridor states $\boldsymbol{s}_i^{self}(t)$ and other observed UAVs and NCFOs' states $\boldsymbol{s}_i^{other}(t)$. This correlation information, which implies the chance of having collisions and air corridor crossing, will be used by the actor and critic network to derive optimal actions of UAV $i$.

In NLP applications, positional encoding is commonly applied to the words in a sentence since sequences of these words are critical when analyzing their relationships using transformers. Conversely, in the multiple UAV coordination problem, the sequence of some observations is not important, such as the states of other UAVs and NCFOs observed by UAV $i$. We refer to these types of observations as non-sequential observations. Yet, the sequence of elements in air corridor observation $\boldsymbol{s}_i^{cor}(t)$ is critical as they represent the features of four different air corridors, i.e., UAV $i$'s previous, current, next one, and the one following the next air corridors at time slot $t$. The position information of these air corridors is required to be embedded into their feature vectors. We refer to the air corridor observation $\boldsymbol{s}_i^{cor}(t)$ as sequential observations.

To efficiently handle the mixture of sequential and non-sequential observations, we propose the hybrid transformer model to customize the traditional transformer for NLP. Specifically, as shown in Fig. 2, for the sequential observations $\boldsymbol{s}_i^{cor}(t)$,

Authorized licensed use limited to: Southeast University. Downloaded on November 10,2025 at 11:32:51 UTC from IEEE Xplore. Restrictions apply.

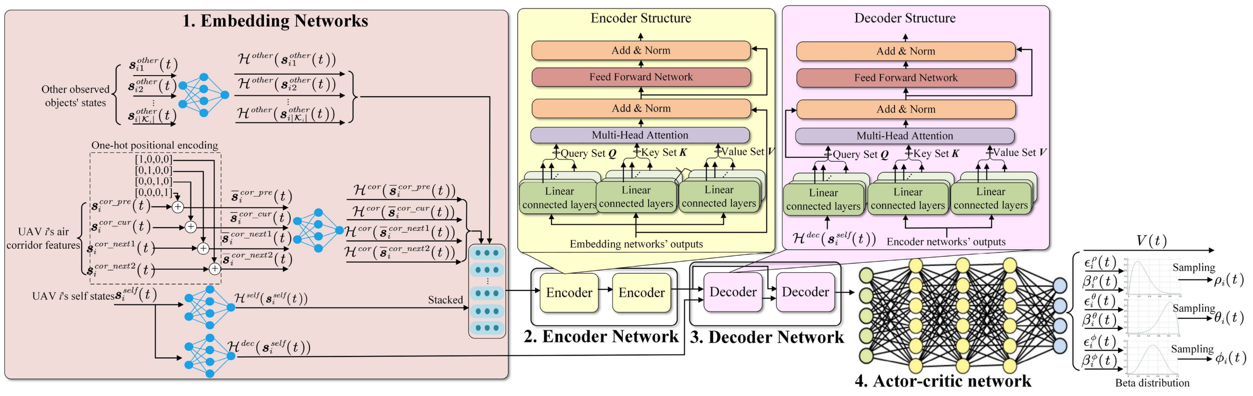


Fig. 2. The HTransRL architecture, which comprises embedding networks, a hybrid transformed with two encoders and two decoders, and an actor-critic network.

one-hot positional encoding is used to concatenate the feature vectors of UAV $i$'s four air corridors, i.e., $\boldsymbol{s}_i^{cor\_pre}(t)$, $\boldsymbol{s}_i^{cor\_cur}(t)$, $\boldsymbol{s}_i^{cor\_next}(t)$, and $\boldsymbol{s}_i^{cor\_next2}(t)$ by their respective one-hot vectors. Denote $\bar{\boldsymbol{s}}_i^{cor\_pre}(t)$, $\bar{\boldsymbol{s}}_i^{cor\_cur}(t)$, $\bar{\boldsymbol{s}}_i^{cor\_next}(t)$, and $\bar{\boldsymbol{s}}_i^{cor\_next2}(t)$ as the feature vectors of UAV $i$'s four air corridors after one-hot positional encoding. The hybrid transformer comprises three major modules, a set of **embedding networks**, an **encoder network**, and a **decoder network**.

1) The sequential and non-sequential observations are fed into their respective **embedding networks**, which are used to normalize the values and sizes of different observations. Each embedding network is implemented as a fully connected neural network with batch normalization. Denote $\mathcal{H}^{self}()$, $\mathcal{H}^{cor}()$, and $\mathcal{H}^{other}()$ as the functions implemented by three types of embedding networks for self-state, air corridor, and other flying object observations. Let $\mathcal{H}^{self}(\boldsymbol{s}_i^{self}(t))$ be the outputs of the embedding network for self-state observation. Note that UAV $i$ may observe multiple flying objects. The states of these objects are parallelly fed into the same embedding network $\mathcal{H}^{other}()$ to generate a set of outputs, denoted as
$$\left\{\mathcal{H}^{other}(\boldsymbol{s}_{i1}^{other}(t)),\ldots,\mathcal{H}^{other}(\boldsymbol{s}_{i|\mathcal{K}_i|}^{other}(t))\right\}, \quad (13)$$
where $\boldsymbol{s}_{ik}^{other}(t)=\{\boldsymbol{p}_k(t),\boldsymbol{v}_k(t)\},\forall k\in\mathcal{K}_i$, is the state of object $k$ observed by UAV $i$. Similarly, four air corridors' feature vectors are parallelly fed into the same embedding network $\mathcal{H}^{cor}()$ to generate corresponding results, i.e.,
$$\left\{\begin{array}{l}\mathcal{H}^{cor}(\bar{\boldsymbol{s}}_i^{cor\_pre}(t)),\mathcal{H}^{cor}(\bar{\boldsymbol{s}}_i^{cor\_cur}(t)),\\ \mathcal{H}^{cor}(\bar{\boldsymbol{s}}_i^{cor\_next1}(t)),\mathcal{H}^{cor}(\bar{\boldsymbol{s}}_i^{cor\_next2}(t))\end{array}\right\}. \quad (14)$$
The outputs of these embedding networks will be stacked together to form the inputs of the **encoder network**. In addition, self-state observation $\boldsymbol{s}_i^{self}(t)$ is fed into another embedding network, whose output, denoted as $\mathcal{H}^{dec}(\boldsymbol{s}_i^{self}(t))$, will be one of the inputs to generate the query for the **decoder network** to inquire the correlation between $\boldsymbol{s}_i^{self}(t)$ and other observations.

2) The **encoder network** is used to generate a high-dimensional vector based on the outputs from the **embedding networks**. This vector integrates interrelationships among different observations at time slot $t$. In the proposed **encoder network**, there are two encoders, each following the traditional encoder structure, i.e., i) three types of linear neural networks to generate query ($\boldsymbol{Q}$), key ($\boldsymbol{K}$), and value ($\boldsymbol{V}$) for each output element from the **embedding networks**, e.g., $\mathcal{H}^{self}(\boldsymbol{s}_i^{self}(t))$, $\mathcal{H}^{other}(\boldsymbol{s}_{i1}^{other}(t))$, etc; ii) a multi-head attention model followed by add & norm to capture various relationships and patterns among different $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ sets; iii) a feed-forward neural network followed by add & norm to process the multi-head attention outputs.

3) The **decoder network** is used to generate a vector representing the relationships between UAV $i$'s self-state $\boldsymbol{s}_i^{self}(t)$ and other observations. This relationship includes, for example, the evaluation of having collisions and air corridor boundary crossing. Two decoder models are designed in the **decoder network**. Each decoder follows the traditional decoder structure, i.e., i) three types of linear neural networks to generate query ($\boldsymbol{Q}$), key ($\boldsymbol{K}$), and value ($\boldsymbol{V}$), where $\boldsymbol{K}$ and $\boldsymbol{V}$ are generated based on the output of the encoder network, and $\boldsymbol{Q}$ is generated based on the output $\mathcal{H}^{dec}(\boldsymbol{s}_i^{self}(t))$; ii) a multi-head attention model followed by add & norm; iii) a feed-forward neural network followed by add & norm.

## B. Actor-Critic Network

The **actor-critic network** takes the outputs from the **decoder network** to generate 1) the estimated state value $V(t)$ that is used to evaluate the current policy, and 2) three distributions corresponding to UAV $i$'s action $\hat{\boldsymbol{a}}_i(t)=[\rho_i(t),\theta_i(t),\phi_i(t)]$, i.e., $B(\epsilon_i^\rho(t),\beta_i^\rho(t))$, $B(\epsilon_i^\theta(t),\beta_i^\theta(t))$, and $B(\epsilon_i^\phi(t),\beta_i^\phi(t))$. These three distributions are used to sample the actions $\rho_i(t)$, $\theta_i(t)$, and $\phi_i(t)$, respectively. Here, instead of using a Gaussian distribution, we apply a Beta distribution to allow the agent to explore different actions, where $\epsilon$ and $\beta$ in $B(\epsilon,\beta)$ are the two parameters to control the shape of the distribution. The major reason for using a beta distribution instead of a Gaussian distribution for exploration is that a beta distribution naturally generates values between 0 and 1. This feature is particularly useful as it ensures the generated acceleration $\boldsymbol{a}_i(t)$ remains

Authorized licensed use limited to: Southeast University. Downloaded on November 10,2025 at 11:32:51 UTC from IEEE Xplore. Restrictions apply.

below the maximum acceleration $a^{\max}$ when applied through (12). As such, defining a penalty for Constraint (9) is unnecessary. For Constraint (8), which defines feasible values for UAV $i$'s velocity $\boldsymbol{v}_i(t)$, $\boldsymbol{v}_i(t)$ is calculated based on $\boldsymbol{a}_i(t)$ generated by HTransRL. If $\|\boldsymbol{v}_i(t)\| \leq v^{\max}$, the velocity is clipped as $\boldsymbol{v}_i(t) := v^{\max} \cdot \frac{\boldsymbol{v}_i(t)}{\|\boldsymbol{v}_i(t)\|}$ to ensure compliance with Constraint (8). For the **actor-critic network** implementation, the loss functions of actor and critic as well as the estimation of advantage follow the same design in PPO-GAE [31], [32]. However, instead of creating two independent neural networks, we combine the actor and critic networks into a single neural network, which can potentially lead to more efficient training.

### C. Curriculum Learning

Curriculum learning is a training strategy that begins with simpler tasks and progressively increases task complexity, facilitating more effective learning and convergence to better solutions [28], [34]. In the early stage of training HTranRL, UAVs often face difficulty in reaching their destinations and consequently struggle to earn intermediate arrival and final arrival rewards. This absence of arrival-based incentives poses challenges in policy development. On the other hand, simplifying the task leads to dense triggering of positive rewards, effectively motivating agents to reach their destinations. A pivotal aspect of curriculum learning involves defining a metric for task complexity, denoted as $\xi$, which ranges from 0.1 (least complex) to 1 (most complex). The training begins in the least complex environment and continues until the model reaches a defined **milestone**, such as an average successful arrival rate above a predefined threshold. Once this **milestone** is achieved, task complexity increases incrementally, allowing the model to adapt to progressively more challenging scenarios.

The definition of task complexity may vary among different applications. In air corridor traffic management, it is reasonable to define the task complexity based on the length of a truncated cylinder and partial torus. Specifically,

*Task complexity for a cylinder:* In each episode, the length $l^{cyl}$ of the cylinder is chosen randomly for every cylinder. It follows a uniform distribution $l^{cyl} = U(l^{\min}, l^{\min} + \Delta l \times \zeta)$, where $l^{\min}$ and $\Delta l$ are preset values. Therefore, a larger $\zeta$ indicates the creation of a longer truncated cylinder, signifying increased task complexity.

*Task complexity for a partial torus:* In each episode, the angle $\mu^{tor}$ of a partial torus is randomly selected for every torus, following a uniform distribution $\mu^{tor} = \frac{\pi}{2} \times U(\max(0.1, \zeta - 0.1), \zeta + 0.1)$. Therefore, a larger $\zeta$ value results in a more broadly spanning torus, indicating higher task complexity.

Algorithm 1 summarizes the curriculum learning process for training HTransRL.

**Algorithm 1:** Training HTransRL With Curriculum Learning.

```
1  Task complexity level $\xi = 0.1$;
2  while $\xi \leq 1$ do
3      while the milestone for the current task complexity
         level has not been reached do
4          while the reply buffer has not been fully
             occupied do
5              Randomly generate several air corridors
                 based on the current task complexity level;
6              Place NCFOs via random waypoint models;
7              while all UAVs have reached their
                 destinations or $t > T^{max}$ do
8                  Control the UAVs in air corridors based
                     on the current HTransRL model and
                     save the transitions in the reply buffer;
9                  $t = t + 1$;
10             end
11         end
12         Divide the transitions in the replay buffer into
             several mini-batches;
13         Train HTransRL using PPO-GAE with the
             mini-batches;
14         Evaluate if the milestone has been reached;
15         Empty the reply buffer;
16     end
17     $\xi = \xi + 0.1$
18 end
```

## V. Simulation Results

### A. Simulation and Training Setups

During each training episode, we randomly selected the number of UAVs in the system between 4 and 12, all being released at the same starting plane and following the same path, which comprises 5 air corridors. The radii of the air corridors are the same, i.e., $r^{cly} = r^{tor} = 2$ meters. The lengths of these air corridors are randomly generated in each episode and gradually increase over episodes based on the task complexity defined in Section IV-C. Meanwhile, we guarantee that these 5 randomly generated air corridors are interconnected, with the type of interconnection randomly chosen from "cttct", "tcttc", or "ttctt", where "t" denotes a partial torus and "c" denotes a cylinder. Moreover, there are 7 NCFOs, where 4 of them are static and randomly placed in the air corridors and 3 of them maneuver in the airspace. Each NCFO follows a random waypoint model, whose destination is randomly selected in the airspace, and the velocity is randomly chosen between 0.5 and 2 m/s. Hence, there are a total of 11-19 flying objects in the airspace. Each UAV has an observation area defined by a sphere with a radius of 6 meters, allowing it to monitor the states of other UAVs and NCFOs located within this sphere. To verify HTransRL's collision avoidance capability, we train and test the model in the worst-case scenario: all the UAVs are simultaneously released from the same starting plane, traverse the same air corridor path, and arrive at the same end plane. To ensure collision-free initial states, the UAVs' positions are randomly selected from the 37 red spots arranged in a hexagonal grid on the starting plane, as shown in Fig. 3(a), with a radius of 2.0 meters. Adjacent spots

Authorized licensed use limited to: Southeast University. Downloaded on November 10,2025 at 11:32:51 UTC from IEEE Xplore. Restrictions apply.

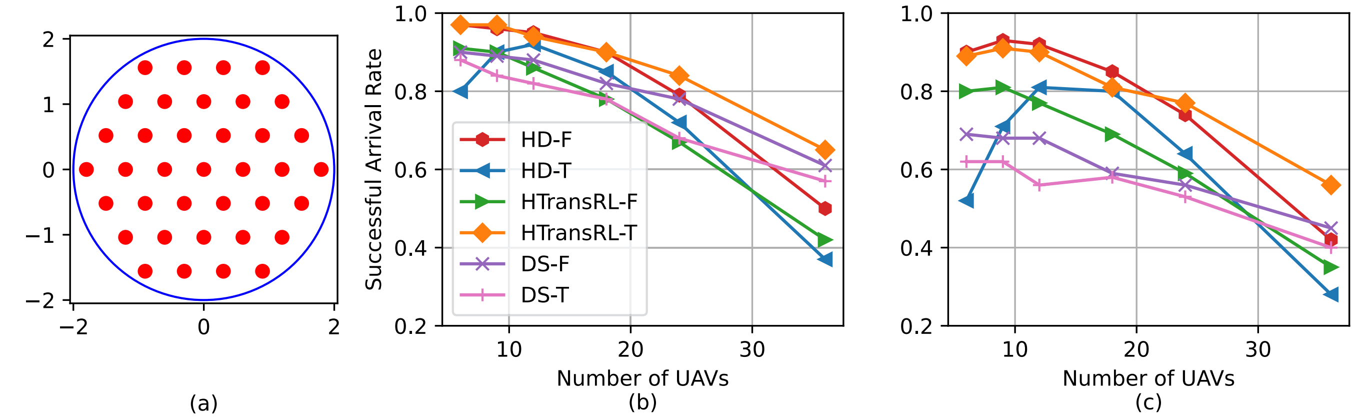


Fig. 3. Average arrival rate of the UAVs, where (a) shows the layout of the initial positions of the UAVs at the starting plane, and (b) and (c) show the average arrival rate of the UAVs when they go through the "cttcttc" and "cttcttcttc" paths, respectively.

are spaced 0.6 meters apart, slightly above the 0.4-meter safety threshold, ensuring a collision-free configuration. If fewer than 37 UAVs are used, each UAV selects an unoccupied position uniformly[1] from the red spots. Thus, the initial positions of the UAVs vary across different Monte Carlo simulations. Note that the parameters for the air corridors and UAVs are carefully selected to ensure that navigating multiple UAVs within the corridors is both feasible and challenging in optimizing travel time while avoiding collisions and boundary crossings.

The detailed architecture of HTransRL, which comprises three primary components, is outlined below:

- *Embedding Networks:* These consist of four fully connected neural networks, each with the same two-hidden-layer structure. The input layer, with 32 neurons, processes raw observed states such as corridor descriptions and UAV states. Each hidden layer contains 64 neurons with ReLU activation, and the output layer includes 128 neurons with a SoftMax activation function for normalization.
- *Transformer:* The transformer component includes two encoders and two decoders, arranged in an encoder-encoder-decoder-decoder sequence. Each encoder and decoder share the same architecture, comprising a multi-head self-attention mechanism with 4 heads and a single-layer position-wise fully connected network with 128 neurons. Each module in the sequence receives input from its predecessor.
- *Actor-Critic Network:* This network features three hidden layers, each containing 128 neurons. The output layer consists of 6 neurons representing the parameters of three Beta distributions, which are used to sample UAV $i$'s actions $\rho_i(t)$, $\theta_i(t)$, and $\phi_i(t)$, respectively. A softplus activation function is applied to the output layer to ensure that all parameter values are strictly positive.

The HTransRL model contains a total of 1,226,822 parameters in float32. Leveraging the NVIDIA 3090 GPU's capability of up to 35.5 TFLOPs, the model's inference time can theoretically be under 1 ms. To enable HTransRL to run on edge computing devices, its model size and inference time can be significantly reduced through quantization, which converts float32 parameters to lower precision formats such as float16 or int8, thereby reducing computational demands without compromising performance. Future work will focus on exploring quantization methods for optimizing the HTransRL model.

With respect to the training settings for the actor-critic in HTransRL, there are 16,192 transitions collected in each iteration. These transitions are then divided into a number of mini-batches, each containing 1,024 transitions. The training process alternates between updates to the actor and critic, with each module undergoing 4 training epochs. Distinct learning rates are assigned to each: the actor is trained with a learning rate of $1.5 \times 10^{-3}$, while the critic is trained with a smaller learning rate of $1.5 \times 10^{-4}$. The critic plays a crucial role in stabilizing training by effectively distinguishing beneficial actions from detrimental ones, thus supporting the goal of achieving a collision-free action policy. Regarding curriculum learning settings in HTransRL, the initial task complexity is set to $\zeta = 0.1$. The complexity level increments by 0.1 each time HTransRL achieves the **milestone**, defined as an average arrival rate exceeding 80% over 50 episodes, and progresses until reaching the maximum complexity level of $\zeta = 1.0$. The discount factor $\gamma$ is set to 0.99, as recommended in [23], which has proven effective in multi-agent UAV control environments. Other simulation settings and hyperparameters are listed in Table II. The source code for HTransRL and corridor visualization are available at.[2]

## B. Comparison Algorithms

Besides the proposed HTransRL method described in Fig. 2, we present two other baseline solutions for comparison.

*Hybrid decoder-only multi-agent reinforcement learning (HD)* utilizes the decoder-network-only architecture without

[1]We also evaluated other distributions, such as Gaussian and Poisson, for UAV position selection. The model's performance remained consistent across these distributions. Therefore, the HTransRL model, referred to later, is trained using a uniform distribution for better generalization.

[2]https://github.com/SECNetLabUNM/HTransRL

Authorized licensed use limited to: Southeast University. Downloaded on November 10,2025 at 11:32:51 UTC from IEEE Xplore. Restrictions apply.

TABLE II
SIMULATION AND TRAINING SETUPS

| Parameter | Value |
|---|---|
| Minimum length of a truncated cylinder ($l^{min}$) | 5 meters |
| Maximum length of a truncated cylinder ($l^{min}+\Delta l$) | 20 meters |
| Radius of a truncated cylinder ($r^{cyl}$) | 2 meters |
| Maximum velocity of a UAV ($v^{max}$) | 1.5 m/s |
| Maximum acceleration of a UAV ($a^{max}$) | 0.3 m/s$^2$ |
| Radius of a UAV/NCFO ($\varrho$) | 0.2 m |
| Maximum duration of an episode ($T^{max}$) | 500 time steps |
| Duration of a time step ($\Delta t$) | 1 s |

having the encoder network. The decoder network in HD comprises three decoders, each following the same structure as the decoder shown in Fig. 2. HD applies the same embedding networks and actor-critic network in HTransRL.

*Hybrid Deep Set based multi-agent reinforcement learning (DS)* utilizes the Deep Set model [19] to replace the transformer in HTransRL to analyze the correlations among a UAV's observations with various dimensions. DS applies the same embedding networks and actor-critic network in HTransRL.

In addition, to investigate the effectiveness of one-hot positional encoding, we test each method's performance both with and without it. A method using one-hot positional encoding is indicated by the suffix "-T", while those without it is indicated by the suffix "-F".

Note that we also try to train **nearest neighbor observation (NNO)** [8], which aims to fix the observation size by selecting the states of the three observed UAVs/NCFOs closest to UAV $i$. Any additional UAVs/NCFOs observed by UAV $i$ are discarded. If fewer than three are observed, zeros are padded. These observations, along with $\boldsymbol{s}_i^{self}(t)$ and $\boldsymbol{s}_i^{cor}(t)$, are fed into the actor-critic network without being preprocessed by the transformer. NNO applies the same embedding networks and actor-critic network in HTransRL. Yet, NNO cannot achieve an average arrival rate higher than 80% when task complexity $\zeta = 0.4$. Consequently, NNO's performance is expected to be worse than others that can successfully raise $\zeta$ to 1.0 during curriculum learning. Hence, NNO will not be included in performance comparisons with others during testing.

### C. Test Results

After the models have been trained via curriculum learning, we test their performance in a different environment, where multiple UAVs simultaneously transverse over the same air corridor path with 4 static and 3 mobile NCFOs. Two types of paths are used, i.e., "cttcttc" and "cttcttcttc", where "t" represents a partial torus and "c" represents a cylinder. The parameters of the air corridors in each path are randomly generated in each test episode. Fig. 3(b) and (c) show the average arrival rate of a UAV after 300 Monte Carlo simulation episodes for the two types of air corridor paths, respectively.

*1) The Effect of One-Hot Positional Encoding:* As we mentioned before, we argue that the sequence of the air corridor observation is critical, and so one-hot positional encoding is used to concatenate the feature vectors of UAV $i$'s four air corridors with their respective one-hot vector. Here, we investigate how one-hot positional encoding affects the performance of different methods. From Fig. 3(b) and (c), we can observe that 1) HTransRL with one-hot positional encoding, i.e., HTransRL-T, achieves the highest arrival rate in most cases; 2) HTransRL-T consistently achieves a higher arrival rate than HTransRL without having one-hot positional encoding, i.e., HTransRL-F. Yet, it is surprising to observe that having one-hot positional encoding achieves a lower arrival rate than without having one-hot positional encoding in HD and DS. This phenomenon can be explained as follows. HTransRL is the only method to explicitly analyze correlations among different air corridors by calculating their respective attention scores, and these correlations are critical to controlling a UAV, preventing it from boundary crossing, especially when the UAV transits between two air corridors. Providing the sequence of these four air corridors is critical in correctly analyzing their correlations, and thus HTransRL-T outperforms HTransRL-F. On the other hand, HD and DS do not analyze the correlations among different air corridors. For example, HD only calculates the attention scores between the states of UAV $i$ $\boldsymbol{s}_i^{self}(t)$ and the rest of the observations. Hence, using one-hot positional encoding is equivalent to adding noise to UAV $i$ air corridor feature vectors to jeopardize the correlation analysis. As a result, the arrival rates of HD-T and DS-T are higher than those of HD-F and DS-F. In the following, unless otherwise specified, we will use the best configuration, i.e., HTransRL-T, HD-F, and DS-F, to compare their performance.

*2) Scalability Analysis:* The models are trained based on the scenario with 4-12 UAVs passing the path with 5 air corridors. To evaluate the scalability of different models, we test their performance by increasing 1) the number of UAVs and 2) the number of air corridors in the path. In Fig. 3(b), HTransRL-T and HD-F exhibit similar performance with an arrival rate above 90%, but both outperform DS-F when the number of UAVs is fewer than 12. Yet, as the number of UAVs increases, the gap between HTransRL-T and HD-F/DS-F increases. Fig. 3(c) shows a similar trend, except that HD-F outperforms HTransRL-T when the number of UAVs is 12. One possible reason to explain this exception is that HTransRL-T has higher model complexity than HD-F. While HD-F is well-trained, HTransRL-T may require more episodes to achieve comparable performance. Overall, HTransRL-T demonstrates greater scalability compared to HD-F and DS-F.

Fig. 4(a) details the incidents causing UAVs to fail to reach their destinations in four different scenarios, respectively. In each bar graph, the $x$-axis represents the number of incidents, the $y$-axis represents three methods, and four bar colors represent four types of incidents leading to failure. From the figure, we observe that the primary reason for HTransRL achieving higher scalability is its slower increment in collision incidents as the number of UAVs/air corridors increases.

To further analyze the results in Fig. 4(a), we calculate the average speed among the UAVs that have successfully arrived at the destinations through the "cttcttcttc" path for 300 episodes. As shown in Fig. 4(b), HD consistently maneuvers the UAVs' speed close to the maximum speed of 1.5 m/s and does not reduce the UAVs' speeds as the number of UAVs increases. Note that the

Authorized licensed use limited to: Southeast University. Downloaded on November 10,2025 at 11:32:51 UTC from IEEE Xplore. Restrictions apply.

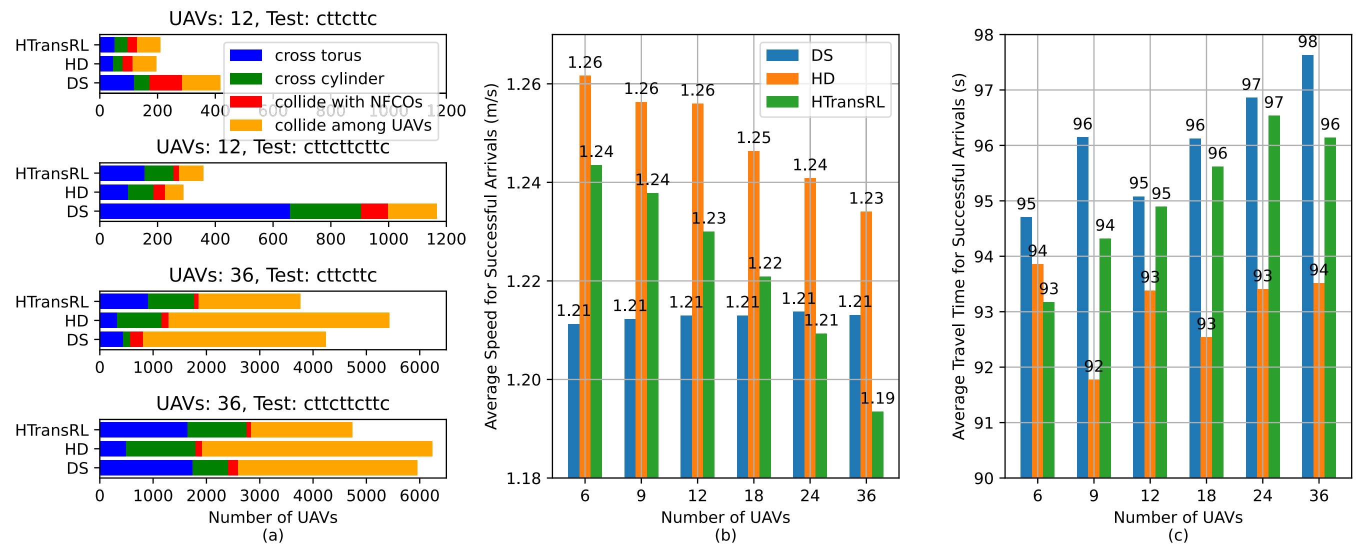


Fig. 4. Test results for different methods in 300 episodes, where (a) shows the numbers of three incident types, and (b) and (c) show the average travel speed and time of the UAVs, respectively, when they go through the "cttcttcttc" path.

UAVs will inevitably cross the boundary of a torus when they are at the maximum speed since even the maximum deceleration cannot provide a sufficient centripetal force to maintain the UAVs within the torus. As a result, UAVs having a higher speed leads to a higher probability of collisions and boundary crossings when they transverse a torus and the number of UAVs is large. This can be observed by comparing the second and fourth bar graphs in Fig. 4(a), where the number of incidents incurred by HD is the lowest for 12 UAVs but the highest for 36 UAVs, indicating that HD shows the largest increase in incidents as the number of UAVs increases. On the other hand, DS does not change the average speed of the UAVs, implying that DS is not adaptive to the number of UAVs. HTransRL can adaptively reduce the average speed of the UAVs, thus leading to the smallest increase in incidents as the number of UAVs increases.

*3) Relationship Among Average Arrival Rate, Travel Speed, and Travel Time:* One of the objectives of the multiple UAV coordination problem is to minimize the overall travel time. Hence, we calculate the average travel time of the UAVs that successfully reach their destinations via the "cttcttcttc" path over 300 episodes. As illustrated in Fig. 4(c), all three algorithms achieve very similar average travel time, with HD generally having the smallest value, except when the number of UAVs is six. Also, the average travel time does not change significantly as the number of UAVs increases for all three algorithms. In addition, a high average speed generally results in a low average travel time, as shown by comparing the results in Fig. 4(b) and (c), except when the number of UAVs is six. However, there is no clear relationship between average arrival rate and travel speed/time. For instance, HD has the fewest incidents, highest average travel speed, and lowest average travel time with 12 UAVs. In contrast, with 36 UAVs, HD has the most incidents, despite maintaining the highest average travel speed and lowest average travel time.

Based on the simulation results, we conclude that when the training environment matches or exceeds the test environment in complexity, HD performs similarly to but slightly better than HTransRL, with both outperforming DS. However, when the test environment is more complex than the training environment, HTransRL significantly outperforms both HD and DS, achieving fewer incidents and similar travel times, demonstrating its scalability.

## VI. Conclusion and Future Work

In this paper, we have formulated a multiple UAV coordination problem in a 3D air corridor environment. To solve this problem, we have designed HTransRL, which incorporates 1) a transformer to handle dynamic observation dimensions, and 2) curriculum learning to improve the training efficiency. The test results show that when the test environment is simpler or similar to training, HTransRL enables UAVs to promptly reach their destinations while maintaining a successful arrival rate of over 90%. When the test environment is more complex than the training environment, HTransRL shows much better scalability than the other two baseline methods with a much higher arrival rate and similar travel time.

While HTransRL outperforms baseline methods in both performance and scalability, challenges remain for real-world deployment. First, UAV $i$'s observations may be inaccurate or delayed due to sensor hardware limitations, potentially causing HTransRL to generate suboptimal actions that lead to collisions or air corridor boundary crossings. To address this, we will explore applying federated learning [35] to enable multiple UAVs to collaboratively fine-tune HTransRL parameters in real time, leveraging data with inaccuracies or delays encountered during flight. Incorporating perturbations during fine-tuning can further enhance the model's generalization, enabling it to better handle delayed or noisy feedback. Second, while HTransRL

Authorized licensed use limited to: Southeast University. Downloaded on November 10,2025 at 11:32:51 UTC from IEEE Xplore. Restrictions apply.

achieves a successful arrival rate exceeding 90%, this performance remains insufficient for real-world deployment. A key limitation is that DRL algorithms, including HTransRL, are not inherently designed to guarantee safe actions that ensure compliance with constraints. A common DRL approach to mitigate unsafe actions is to penalize them during training, discouraging constraint violations. However, this method cannot guarantee the model will always produce safe actions, as its primary objective is to maximize cumulative rewards. If the reward for an unsafe action exceeds the penalty, the agent may still select it. While increasing penalties reduces unsafe actions, it can also restrict exploration, limiting the discovery of optimal policies. Achieving a balance between ensuring safety and encouraging exploration remains a critical and challenging task in DRL. To address this challenge, we propose integrating control barrier functions [8] into HTransRL. This approach will evaluate UAV $i$'s states across multiple future time slots and generate actions that ensure its states remain within the safe set.

## References

[1] U.S. Department of Transportation Federal Aviation Administration, *Urban Air Mobility (UAM): Concept of Operations*, 2023. [Online]. Available: www.faa.gov/sites/faa.gov/files/Urban%20Air%20Mobility%20%28UAM%29%20Concept%20of%20Operations%202.0_0.pdf

[2] K. Prabhath et al., "Invited paper: Ground-based communication support for air corridors," in *Proc. IEEE 34th Annu. Int. Symp. Pers. Indoor Mobile Radio Commun.*, 2023, pp. 1–6.

[3] F. Fabra, J. Wubben, C. T. Calafate, J. C. Cano, and P. Manzoni, "Efficient and coordinated verticaltakeoff of UAV swarms," in *Proc. IEEE 91st Veh. Technol. Conf.*, 2020, pp. 1–5.

[4] C. Sastre, J. Wubben, C. T. Calafate, J.-C. Cano, and P. Manzoni, "Safe and efficient take-off of VTOL UAV swarms," *Electronics*, vol. 11, no. 7, 2022, Art. no. 1128, doi: 10.3390/s22145437.

[5] G. Wang, W. Yao, X. Zhang, and Z. Li, "A mean-field game control for large-scale swarm formation flight in dense environments," *Sensors*, vol. 22, no. 14, 2022, Art. no. 5437, doi: 10.3390/s22145437.

[6] S. Bhattacharya, S. Kailas, S. Badyal, S. Gil, and D. Bertsekas, "Multiagent rollout and policy iteration for POMDP with application to multi-robot repair problems," in *Proc. 2020 Conf. Robot Learn.*, J. Kober, F. Ramos, and C. Tomlin, Eds., PMLR, 2021, pp. 1814–1828. [Online]. Available: https://proceedings.mlr.press/v155/bhattacharya21a.html

[7] N. Golowich, A. Moitra, and D. R. Ohatgi, "Learning in observable POMDPs, without computationally intractable oracles," in *Proc. Adv. Neural Inf. Process. Syst.*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., Curran Associates, Inc., 2022, pp. 1458–1473. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/099607cd970f4e1ac2fdd30624dffff8-Paper-Conference.pdf

[8] Z. Qin, D. Sun, and C. Fan, "Sablas: Learning safe control for black-box dynamical systems," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 1928–1935, Apr. 2022.

[9] J. Hu, H. Zhang, L. Song, R. Schober, and H. V. Poor, "Cooperative internet of UAVs: Distributed trajectory design by multi-agent deep reinforcement learning," *IEEE Trans. Commun.*, vol. 68, no. 11, pp. 6807–6821, Nov. 2020.

[10] Z. Ma, Z. Wang, A. Ma, Y. Liu, and Y. Niu, "A low-altitude obstacle avoidance method for UAVs based on polyhedral flight corridor," *Drones*, vol. 7, no. 9, 2023, Art. no. 588. [Online]. Available: https://www.mdpi.com/2504-446X/7/9/588

[11] S. I. Muna et al., "Air corridors: Concept, design, simulation, and rules of engagement," *Sensors*, vol. 21, no. 22, 2021, Art. no. 7536. [Online]. Available: https://www.mdpi.com/1424-8220/21/22/7536

[12] O. Bouhamed, H. Ghazzai, H. Besbes, and Y. Massoud, "Autonomous UAV navigation: A DDPG-based deep reinforcement learning approach," 2020. [Online]. Available: https://arxiv.org/abs/2003.10923

[13] S. Bhagat and P. Sujit, "UAV target tracking in urban environments using deep reinforcement learning," in *Proc. 2020 Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, 2020, pp. 694–701.

[14] M. Sherman, S. Shao, X. Sun, and J. Zheng, "Counter UAV swarms: Challenges, considerations, and future directions in UAV warfare," *IEEE Wireless Commun.*, early access, Aug. 26, 2024, doi: 10.1109/MWC.003.2400047.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] B. Li and Y. Wu, "Path planning for UAV ground target tracking via deep reinforcement learning," *IEEE Access*, vol. 8, pp. 29064–29074, 2020.

[17] S. Deniz, Y. Wu, Y. Shi, and Z. Wang, "A reinforcement learning approach to vehicle coordination for structured advanced air mobility," *Green Energy Intell. Transp.*, vol. 3, 2024, Art. no. 100157. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2773153724000094

[18] S.-H. Noh, "Analysis of gradient vanishing of RNNs and performance comparison," *Information*, vol. 12, no. 11, 2021, Art. no. 442. [Online]. Available: https://www.mdpi.com/2078-2489/12/11/442

[19] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Proc. Adv. Neural Inf. Process. Syst.*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017, pp. 3394–3404. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/f22e4747da1aa27e363d86d40ff442fe-Paper.pdf

[20] Z. Qin, K. Zhang, Y. Chen, J. Chen, and C. Fan, "Learning safe multi-agent control with decentralized neural barrier certificates," 2021, *arXiv:2101.05436*.

[21] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *J. Mach. Learn. Res.*, vol. 21, no. 178, pp. 1–51, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-081.html

[22] J.-E. Pierre, X. Sun, D. Novick, and R. Fierro, "Multi-agent deep reinforcement learning for countering uncrewed aerial systems," in *Distributed Autonomous Robotic Systems*, Cham, Switzerland: Springer Nature, 2024, pp. 394–407.

[23] J.-E. Pierre, X. Sun, and R. Fierro, "Multi-agent partial observable safe reinforcement learning for counter uncrewed aerial systems," *IEEE Access*, vol. 11, pp. 78192–78206, 2023.

[24] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *Proc. 36th Int. Conf. Mach. Learn.*, K. Chaudhuri and R. Salakhutdinov, Eds., PMLR, 2019, pp. 3744–3753. [Online]. Available: https://proceedings.mlr.press/v97/lee19d.html

[25] C. D. Hsu, H. Jeong, G. J. Pappas, and P. Chaudhari, "Scalable reinforcement learning policies for multi-agent control," in *Proc. 2021 IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2021, pp. 4785–4791.

[26] L. Yu, Z. Li, J. Yao, and X. Sun, "Transformer-based multi-agent reinforcement learning for multiple unmanned aerial vehicle coordination in air corridors," in *Proc. Workshop Cooperative Commun. Comput. Space-Air-Ground-Sea Integr. Netw.*, 2024, pp. 505–510, doi: 10.1109/ICCWorkshops59551.2024.10615924.

[27] A. W. Moore, "Efficient memory-based learning for robot control," Univ. Cambridge, Cambridge, U.K., Tech. Rep. TR-209, 1990.

[28] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel, "Reverse curriculum generation for reinforcement learning," in *Proc. 1st Annu. Conf. Robot Learn.*, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., 2017, pp. 482–495.

[29] P. Yu et al., "Intelligent-driven green resource allocation for industrial Internet of Things in 5G heterogeneous networks," *IEEE Trans. Ind. Informat.*, vol. 18, no. 1, pp. 520–530, Jan. 2022.

[30] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. 31st Int. Conf. Mach. Learn.*, E. P. Xing and T. Jebara, Eds., Bejing, China: PMLR, 2014, pp. 387–395.

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

[32] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015, *arXiv:1506.02438*.

[33] A. Vaswani et al., "Attention is all you need," in in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.

[34] X. Wang, Y. Chen, and W. Zhu, "A survey on curriculum learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 4555–4576, Sep. 2022.

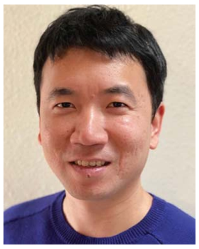
[35] L. Yu, R. Albelaihi, X. Sun, N. Ansari, and M. Devetsikiotis, "Jointly optimizing client selection and resource management in wireless federated learning for Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4385–4395, Mar. 2022.

Authorized licensed use limited to: Southeast University. Downloaded on November 10,2025 at 11:32:51 UTC from IEEE Xplore. Restrictions apply.

**Liangkun Yu** (Student Member, IEEE) received the BE and ME degrees in communications engineering from Fuzhou University, in 2014 and 2017, respectively. Following this, he joined China Telecom as a wireless network engineer from 2017 to 2019. He commenced his doctoral studies with the SENet Lab, University of New Mexico, in 2019. Specializing in machine learning and wireless networks, his research explores various topics, including reinforcement learning for UAV swarm traffic management in aerial corridors, wireless IoT federated learning, queueing theory application in wireless networks, and drone-assisted mobile access networks.

**Zhirun Li** received the bachelor's degree from the South China University of Technology, China, in 2020, and the master's degree from the University of Missouri, in 2022. He is currently working toward the PhD degree with the Department of Electrical and Computer Engineering, University of New Mexico. His research interests lie at the intersection of machine learning and control systems, with a particular focus on reinforcement learning and its applications in autonomous UAV (unmanned aerial vehicle) control. His work aims to address challenges in multi-agent coordination, real-time decision-making, and safe navigation in dynamic environments.

**Nirwan Ansari** (Life Fellow, IEEE) received the BSEE (summa cum laude with a perfect GPA) degree from NJIT, the MSEE degree from the University of Michigan, and the PhD degree from Purdue University. He is distinguished professor of electrical and computer engineering with the New Jersey Institute of Technology (NJIT). He is also a fellow of National Academy of Inventors. He authored Green Mobile Networks: A Networking Perspective (Wiley-IEEE, 2017) with T. Han, and co-authored two other books. He has also (co-)authored more than 750 technical publications, with more than half of them published in widely cited journals and magazines. He has guest-edited numerous special issues covering various emerging topics in communications and networking. He is the editor-in-chief of *IEEE Wireless Communications Magazine* and has served on the editorial/advisory board of over ten journals. His current research focuses on green communications and networking, cloud computing, drone-assisted networking, and various aspects of broadband networks. He was elected to serve on the IEEE Communications Society (ComSoc) Board of Governors as a member-at-large. He has served as the director of ComSoc Educational Services Board, chaired various technical and steering committees within ComSoc, and served on many committees such as the IEEE fellow Committee. He has actively participated in organizing numerous IEEE International Conferences/Symposia/Workshops. Among his recognitions are several excellence in teaching awards, multiple best paper awards, the NCE Excellence in Research Award, several ComSoc TC technical recognition awards, the NJ Inventors Hall of Fame Inventor of the Year Award, the Thomas Alva Edison Patent Award, the Purdue University Outstanding Electrical and Computer Engineer Award, the NCE 100 Medal, the NJIT Excellence in Research Prize and Medal, and designation as a COMSOC distinguished lecturer. He has also been granted more than 40 U.S. patents.



**Xiang Sun** (Member, IEEE) received the PhD degree in electrical engineering from the New Jersey Institute of Technology, in 2018. He is an assistant professor with the Department of Electrical and Computer Engineering, University of New Mexico. His research interests span a variety of topics, including free space optics, wireless networks, Internet of Things (IoT), edge computing, UAV swarm control, and drug discovery. Additionally, he focuses on developing innovative methodologies in deep reinforcement learning, federated learning, generative adversarial networks, and various non-convex optimization techniques to address diverse research challenges. He has received several honors and awards, including 2016 IEEE International Conference on Communications Best Paper Award, 2018 NJIT Hashimoto Price, 2019 IEICE Communications Society Best Tutorial Paper Award, 2019 NJIT Outstanding Doctoral Dissertation, and 2022 Digital Communications and Networks Outstanding Associate Editor Award. He is an associate editor of *IEEE Open Journal of the Computer Society* and *Digital Communications and Networks*.

Authorized licensed use limited to: Southeast University. Downloaded on November 10,2025 at 11:32:51 UTC from IEEE Xplore. Restrictions apply.