Investigating Cyclic Airspace Corridor Optimization for UAS Traffic Management based on Deep Reinforcement Learning

Rodolphe Fremond*, Matthieu Verdoucq*, Zeynep Bilgin‡, Murat Bronz*†
*ENAC Airbus Sopra Steria Drones and UTM Research Chair
†Fédération ENAC ISAE-SUPAERO ONERA, Université de Toulouse, Toulouse, France
‡Chair of Rotorcraft and Vertical Flight, Technical University of Munich, Munich, Germany
{rodolphe.fremond, matthieu.verdoucq, murat.bronz}@enac.fr
zeynep.bilgin@tum.de

***Abstract*—As drone traffic increases in urban and suburban environments, more complex airspace structures will be required to accommodate rising demand. Unmanned Aircraft System (UAS) Traffic Management (UTM) service providers risk becoming overwhelmed by the time and resource demands of granting flight authorizations, for both human and automated systems. This research proposes the Cyclic Corridor Concept (C^3): a strategic, centralized UTM service architecture designed to offload flight planning trade-offs. The C^3 introduces two vertically separated, unidirectional cyclic corridors within Very Low Level (VLL) airspace. These corridors loop clockwise and counterclockwise through urban areas, passing close to multiple vertiports to ensure fair access and enhance operational efficiency in congested airspace. We investigate the use of Reinforcement Learning (RL) to optimize the corridor's trajectory, accounting for urban constraints such as no-fly zones derived from risk assessments. The proposed method rapidly generates feasible solutions under diverse airspace configurations, enabling near real-time applicability. Results highlight the benefits of C^3 in reducing UTM provider workload as vertiport network complexity increases. The system generates visually intuitive and safe corridors, reducing the need for strategic deconfliction and rerouting. Although the C^3 design may incur additional travel distance, this is offset by faster flight authorizations and more evenly distributed delays across dense traffic flows.**

***Keywords*—UAS Traffic Management; Airspace Design; Reinforcement Learning; Flight Planning; Operational Efficiency.**

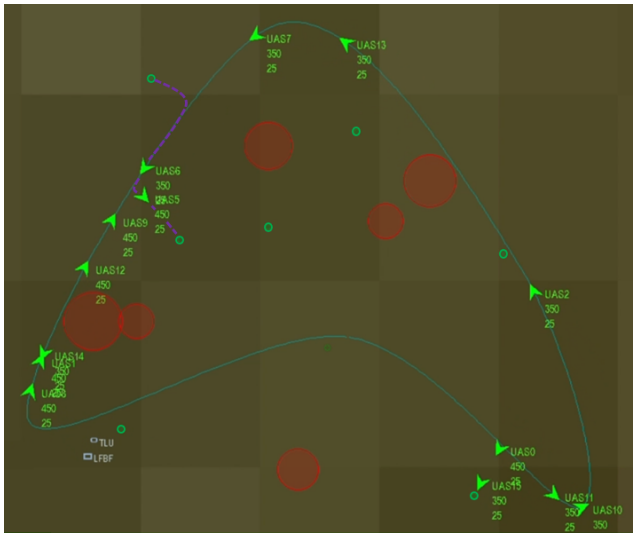


Fig. 1. Cyclic Airspace Corridor generated using RL (Scenario II), applied within the BlueSky ATM software [5]. Fifteen operations safely transit between vertiports via the shortest clockwise or counterclockwise route distributed across two distinct altitudes.

I. Introduction

UTM is an emerging paradigm designed to complement Air Traffic Management (ATM) by enabling operations in VLL airspace. This expansion aims to accommodate the growing demand for UAS and Urban Air Mobility (UAM) operations across urban, suburban, and rural environments. The research, industrialization, and deployment of UTM systems advance through distinct maturity levels, reflecting progress in technology and autonomy in UAS and UAM operations.

In the United States, the FAA and NASA are collaborating on the development of UTM as part of the NextGen initiative, which aims to modernize the National Airspace System (NAS). This industry-driven approach allows for flexible testing and gradual implementation of UTM through a federated model [1]. In contrast, Europe follows a regulation-driven approach with the U-Space framework, which designates VLL airspace exclusively for UAS and UAM operations [2]. Between 2023 and 2030, the focus is on deploying U-Space 2 (U2), which introduces a set of "initial services" [3] to democratize Beyond Visual Line of Sight (BVLOS) operations in controlled, low-density airspace where crewed aviation is currently restricted and developing U-Space 3 (U3) [4].

Airspace design is a core component of UTM, ensuring safe and efficient operations. Rather than being a specific service or system, airspace design represents the intersection of multiple interdependent elements. Effective airspace design requires identifying demand, assessing infrastructure for ground-based fleets, mapping high-risk areas, evaluating airspace capacity, and defining procedural interfaces with ATM when necessary.

Research has identified key constraints in urban airspace design, emphasizing the need for strong coordination of UTM services at strategic, pre-tactical, and tactical levels [6]. Airspace corridors help spatially and temporally delimit operations within defined volumes, a concept already used in ATM and now envisioned for UTM [7]. Several corridor

This research is funded by ENAC - Airbus - Sopra Steria, Drones and UTM Research Chair.

configurations have been proposed. In the U-Space concept, the operational volume is limited to a single operation [4]. In contrast, the shared corridor concept proposed by the FAA and NASA introduces multiple lanes with designated passing zones, enabling different vehicles to share airspace [1], [3]. Each approach has trade-offs. Exclusive operational volumes maximize safety but can create inefficiencies in dense airspace, prioritizing early entries and delaying others. Conversely, shared corridors, like a highway, offer higher throughput and fairer access, while reducing the UTM workload by limiting the need for constant flight plan regeneration, only separation minima must be maintained. However, this model demands strong coordination to ensure safety. NASA has explored shared airspace corridors with point-to-point structures and clearly defined entry/exit zones [8]. Airspace is often discretized using air matrices, air networks, or air-tubes [9]. Recent studies emphasize efficiency based on route length, travel time savings, and population exposure through multi-objective optimization using genetic algorithms [10]. Optimizing such airspace remains challenging due to trade-offs among cost, workload, and safety, typically addressed via mathematical solvers and weighted objectives.

Deep RL (DRL) offers a promising approach to explore airspace design paradigms beyond the limits of human cognition and traditional optimization. Among DRL methods, the Soft Actor-Critic (SAC) algorithm has shown strong performance in several applications [11]. This off-policy algorithm promotes training stability and yields satisfactory solutions by maximizing a reward that includes an entropy regularization term to encourage exploration [12]. While SAC is relatively new compared to methods like DQL or DDPG in UTM and ATM, it has mostly been used in tactical applications such as navigation, UAV control [13], [14], and collision avoidance [15]. In contrast, Proximal Policy Optimization (PPO), an on-policy algorithm, is widely adopted due to its empirical robustness [16]. In airspace design, many feasible solutions exist, but global optima are hard to identify. DRL approaches like SAC and PPO are particularly suited to this context because they balance exploration and exploitation, and operate in continuous action spaces for fine waypoint adjustments without resorting to coarse discretization [17].

Although widely used in ground transportation systems (e.g., roundabouts and ring roads), the concept of a cyclic corridor has not yet been explored as an airspace design for UTM. Yet it presents several hypothetical advantages. First, it is simple to manage: all operations use the same shared airspace, and flight planning is streamlined by always engaging with a central corridor. Strategic conflict resolution is minimized, making separation minima the primary concern. Multiple lanes, tailored to different vehicle performance levels, reduce the likelihood of tactical conflicts, which can be managed through simple separation policies (e.g., assigning vehicles to lanes based on capability). Second, while the design may introduce delays for some operations, these delays are equitably distributed across all traffic, avoiding the prioritization or penalization of specific operations.

In this research, we investigate the novel concept of C^3 under the hypothesis of centralized strategic management of dense urban airspace at VLL. The main contributions of this work are as follows:

- We develop and deploy DRL methods based on PPO and SAC to solve the airspace design optimization problem. These lightweight models are capable of generating a functional cyclic airspace corridor in under one second, given environmental constraints such as vertiport locations and no-fly zones.
- We employ Catmull-Rom spline interpolation to model realistic corridor geometries. The splines are discretized into multiple waypoints that define the corridor path. Although aircraft performance models are not explicitly integrated, corridor curvature is included in the optimization to ensure operational applicability.
- We benchmark the C^3 design against direct routing and the Traveling Salesman Problem (TSP) approach to analyze trade-offs in efficiency, computation time, and UTM workload. Efficiency metrics include extra flight distance, total corridor length, and vertiport-to-corridor transition distance. UTM workload is assessed in terms of processing time and the number of rerouting operations avoided due to conflict prevention by the C^3.
- We compare our DRL-based approach with classical optimization strategies, including gradient-based and local minimization algorithms: BFGS, L-BFGS-B, Powell, Nelder-Mead, SLSQP, and TNC, as well as global optimizers: Differential Evolution and Dual Annealing. We evaluate each method using the metrics above, and demonstrate the superior resilience and scalability of the PPO-based RL approach.
- We validate our approach through proof-of-concept use cases across three synthetic large-scale scenarios, where the PPO algorithm generates cyclic corridors in realistic urban layouts. These scenarios illustrate the scalability and applicability of C^3 as the number of vertiports and no-fly zones increases. Finally, operational safety is qualitatively assessed at the tactical level through simulation.

II. Methods

This section presents the methodology used in this research. Sec. II-A introduces foundational RL theory. Sec. II-B details how RL is adapted for the design and optimization of the proposed C^3. Sec. II-C introduces how the corridors are generated. Finally, Sec. II-D provides a high-level description of additional comparative optimization methods.

A. Foundational Reinforcement Learning

This subsection begins by formally defining the RL problem setting in Sec. II-A1 and the general RL objective in Sec. II-A2, and then briefly introduces the specific characteristics of the two DRL algorithms used in this study: SAC in Sec. II-A3 and PPO in Sec. II-A4.

1) Problem Formulation : The RL problem is formulated as a Markov Decision Process (MDP). In this context, if the airspace is in an initial state, its next configuration after applying a sequence of changes depends solely on the current state and not on the sequence of previous states that led to it.

The MDP is defined as a tuple $\mathcal{E}$ containing the elements $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$.

- The state space $\mathcal{S}$ represents all possible airspace configurations, with $s \in \mathcal{S}$ denoting a particular state.
- The action space $\mathcal{A}$ defines the set of possible actions $a \in \mathcal{A}$ used to parametrize changes in the airspace layout.

- $\mathcal{P}$ is the state transition probability, which specifies the likelihood $\mathbb{P}(s' \mid s,\, a)$ of transitioning to state s' from state s after taking action a.
- The reward function $\mathcal{R}$ assigns a scalar value to each state-action pair $(s,\, a)$, reflecting the quality of the resulting configuration.
- The discount factor $\gamma \in [0, 1]$ controls the trade-off between immediate and future rewards during the optimization process.

2) Objective : The objective in RL is to train a policy π that maximizes the expected return, defined as the cumulative reward an agent receives through interaction with the environment. The return is computed as a discounted sum of future rewards $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$.

The optimization goal is to find the optimal policy parameters ϕ^* such that the behavior of the resulting policy π_{ϕ^*} is best suited to the task. This corresponds to maximizing the expected return: $\phi^* = \arg\max_\phi \mathbb{E}_{\tau\sim\pi_\phi}\left[\sum_t r(s_t,\, a_t)\right]$. Policy gradient methods aim to directly optimize ϕ by exploiting the differentiability of the policy π_ϕ, typically expressed as: $\nabla_\phi J(\phi) = \mathbb{E}_{\pi_\phi}\left[\nabla_\phi \log \pi_\phi(a \mid s)\Psi\right]$. Here, $\nabla_\phi \log \pi_\phi(a \mid s)$ is the score function, which measures the sensitivity of the policy to its parameters, and Ψ is a scalar weighting term that modulates the update magnitude.

In actor-critic frameworks, Ψ is estimated by a critic parameterized by θ, using either a state-value function V_θ, an action-value function Q_θ, or a combination of both to guide the actor in improving π_ϕ. These functions estimate expected returns and support policy updates, defined respectively as $V^\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t) \mid s_0 = s\right]$ and $Q^\pi(s, a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t) \mid s_0 = s,\, a_0 = a\right]$.

This research focuses on the SAC and PPO RL algorithms, which are discussed in Sec. II-A3 and Sec. II-A4, respectively. These methods are described concisely, as the goal of this work is to adapt them to airspace design, rather than to revisit their theoretical foundations. Extensive prior literature provides detailed theoretical treatment for readers less familiar with these RL algorithms.

3) Soft Actor Critic : SAC optimizes a stochastic, off-policy policy π to maximize a trade-off between the expected return and an entropy term $\mathcal{H}$, which promotes exploration [11]. The resulting optimized policy π^* aims to capture the most appropriate airspace design under the given environment.

The entropy term is defined as $\mathcal{H}(\pi(\cdot \mid s)) = -\log \pi(a \mid s)$. The temperature parameter $\alpha > 0$ controls the balance between exploration and exploitation in the learned policy.

Since our airspace design problem involves a high-dimensional continuous domain, we adopt a modern variant of SAC that uses twin Q-functions to stabilize learning and improve convergence [18]. The policy π_ϕ is parameterized by ϕ, and the Q-functions Q_{θ_1} and Q_{θ_2} are each parameterized by separate networks with parameters θ_1 and θ_2, respectively. The full procedure is detailed in Algorithm 1, where operations that are vectorized across parallel environments and optimization steps are visually marked with vertical lines for clarity.

4) Proximal Policy Optimization : PPO is a state of the art, on policy RL algorithm designed to improve stability and reliability during policy updates [19]. Unlike SAC, described in Sec. II-A3, which is off-policy and entropy regularized, PPO directly optimizes a surrogate objective based on the probability ratio between the new and old policy. This ratio is constrained via a clipping mechanism to prevent large, destabilizing updates.

Algorithm 1 Soft Actor Critic

Hyper-parameters: α, τ, γ, λ_ϕ, λ_θ, λ_α, $\mathcal{D}_{size}$, $\mathcal{B}_{size}$, K, W, T_{end}, $\mathcal{E}_{max}$
Initialization: π_ϕ, $(Q_{\theta_k})_{k\in\{1,\,2\}}$, $(Q_{\theta'_k})_{k\in\{1,\,2\}}$
while $\mathcal{D}_{size} < W$ **do**
  reset the environment, s_0
  for optimization step t till step limit T_{end} **do**
    $a_t \sim \pi_\phi(\cdot|s_t)$
    $r_t = R(s_t,\, a_t)$
    $s_{t+1} \sim p(s_{t+1} \mid (s_t,\, a_t))$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t,\, a_t,\, r_t,\, s_{t+1})\}$
  end for
end while
for epoch $\mathcal{E}$ till $\mathcal{E}_{max}$ **do**
  reset the environment, s_0
  for optimization step t till step limit T_{end} **do**
    $a_t \sim \pi_\phi(\cdot|s_t)$
    $r_t = R(s_t,\, a_t)$
    $s_{t+1} \sim \mathbb{P}\left(s_{t+1} \mid (s_t,\, a_t)\right)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t,\, a_t,\, r_t,\, s_{t+1})\}$
  end for
  for K update iterations **do**
    $(s,\, r,\, s') \sim \mathcal{B}_{batch} \subset \mathcal{D}$
    $a',\, \log \pi_\phi(a'|s') \sim \pi_\phi(\cdot|s')$
    $\tilde{a},\, \log \pi_\phi(\tilde{a}|\, s) \sim \mathcal{N}(\pi_\phi(\cdot|\, s))$
    $\hat{Q}(s', a') = r + \gamma\mathbb{E}\left[\min_{k\in\{1,2\}} Q_{\theta'_k}(s', a') - \alpha \cdot \log \pi_\phi(a'|s')\right]$
    $J_Q(\theta_k) = \mathbb{E}\left[\frac{1}{2}\left(Q_{\theta_k}(s',\, a') - \hat{Q}(s',\, a')\right)^2\right],\ k \in \{1,\, 2\}$
    $J_\pi(\phi) = \mathbb{E}\left[\alpha \log \pi_\phi(\tilde{a}|\, s) - \min_{k\in\{1,\,2\}} Q_{\theta_k}(s,\, \tilde{a})\right]$
    $J(\alpha) = -\mathbb{E}\left[\alpha \log \pi_\phi(a|\, s) + \hat{\mathcal{H}}\right],\ \hat{\mathcal{H}} = -dim(\mathcal{A})$
    $\theta_k \leftarrow \theta_k - \lambda_\theta \hat{\nabla}_{\theta_k} J_Q(\theta_k),\ k \in \{1,\, 2\}$
    $\theta'_k \leftarrow \tau\theta_k - (1-\tau)\theta'_k,\ k \in \{1,\, 2\}$
    $\phi \leftarrow \phi + \lambda_\phi \hat{\nabla}_\phi J_\pi(\phi)$
    $\log \alpha \leftarrow \log \alpha - \lambda_\alpha \hat{\nabla}_{\log \alpha} J(\alpha)$
  end for
end for

Algorithm 2 Proximal Policy Optimization

Hyper-parameters: α, ϵ, λ, γ, λ_θ, $\lambda_{\mathcal{H}}$, $\mathcal{E}_{max}$, T_{end}, K, $\mathcal{B}_{minisize}$
Initialization: π_ϕ, V_θ
for epoch $\mathcal{E}$ **to** $\mathcal{E}_{max}$ **do**
  reset the environment: s_0, $\mathcal{D} = \{\varnothing\}$
  for optimization step t **to** step limit T_{end} **do**
    $a_t,\, \log \pi_{old}(a_t|\, s_t) \sim \pi_\phi(\cdot|s_t)$ and $V_t \sim V_\theta(s_t)$
    $r_t = R(s_t,\, a_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t,\, \log \pi_\phi(a_t|\, s_t),\, r_t,\, V_t)\}$
    $s_{t+1} \sim \mathbb{P}\left(s_{t+1} \mid (s_t,\, a_t)\right)$
  end for
  for each couple $(r_t,\, V_t)$ **in** $\mathcal{D}$ **do**
    $\hat{A}_t \leftarrow r_t + \gamma V_{t+1} - V_t + \gamma\lambda\hat{A}_{t+1}$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\hat{A}_t)\}$
  end for
  for K update iterations **do**
    for each mini-batch $\mathcal{B} \subset \mathcal{D}$ **do**
      $\left(s,\, \log \pi_{\text{old}}(a\,|s),\, V,\, \hat{A}\right) \leftarrow \mathcal{B}$
      $\eta(\phi) = \frac{\pi_{\text{new}}(a\,|s)}{\pi_{\text{old}}(a\,|s)}$ with $\log \pi_{\text{new}}(a|\, s) \sim \pi_\phi(\cdot|\, s)$
      $\mathcal{L}^{\text{CLIP}}(\phi) = \mathbb{E}_{t\in\mathcal{B}}\left[\min\left(\eta(\phi)\hat{A}_t,\, \text{clip}\left(\eta(\phi),\, 1-\epsilon,\, 1+\epsilon\right)\hat{A}_t\right)\right]$
      $\mathcal{L}^{\text{VF}}(\theta) = \mathbb{E}_{t\in\mathcal{B}}\left[\left(V_\theta(s_t) - (\hat{A}_t + V_t)\right)^2\right]$
      $\mathcal{L}^{\mathcal{H}}(\phi) = \mathbb{E}_{t\in\mathcal{B}}\left[\mathcal{H}[\pi_\phi(\cdot|s_t)]\right]$
      $\mathcal{L} = -\mathcal{L}^{\text{CLIP}} + \lambda_\theta\mathcal{L}^{\text{VF}} - \lambda_{\mathcal{H}}\mathcal{L}^{\mathcal{H}}$
      $\phi \leftarrow \phi - \alpha\nabla_\phi\mathcal{L}$ and $\theta \leftarrow \theta - \alpha\nabla_\theta\mathcal{L}$
    end for
  end for
end for

The optimization is driven by a clipped objective function $\mathcal{L}^{\text{CLIP}}$, which ensures that the updated policy π_ϕ does not diverge excessively from the previous policy π_{old}. This constraint stabilizes training by bounding the probability ratio $\eta_t(\phi)$, while still allowing meaningful updates when the advantage estimate $\hat{A}_t$ is large. PPO uses the Generalized Advantage Estimator (GAE) [20], a method that combines the discount factor γ and a smoothing parameter λ to compute a weighted average of multi-step estimators ψ, balancing bias and variance. Specifically, as $\lambda \to 0$, GAE ignores value function accuracy; as $\lambda \to 1$, it reduces variance but may introduce bias. The discount factor $\gamma \in [0, 1]$ controls the emphasis on long-term rewards: $\gamma \to 1$ promotes long-term planning, while $\gamma \to 0$ favors short-term gains.

The policy is trained alongside a state-value function V_θ, which contributes to a value loss term $\mathcal{L}^{\text{VF}}$. An entropy bonus term $\mathcal{L}^{\mathcal{H}}$ is also included to encourage exploration. The total loss function combines these terms with corresponding weights λ_θ and $\lambda_{\mathcal{H}}$.

In our airspace optimization case study, PPO is applied to continuous state and action spaces, leveraging its ability to produce smooth and stable policies. By iteratively collecting rollout data and performing clipped updates, PPO ensures policy improvement without diverging from prior behavior. The full procedure is detailed in Algorithm 2, where vectorized steps across parallel environments and update phases are visually marked with vertical lines.

B. Reinforcement Learning Adaption to Airspace Design

This section presents how the MDP formulation described in Sec. II-A1 is instantiated for the airspace corridor optimization problem. We detail the construction of the state space in Sec. II-B1, action space in Sec. II-B2, state transition mechanism in Sec. II-B3, and reward function in Sec. II-B4.

1) State Space : The environment is modeled as a two dimensional airspace bounded by the geographic coordinates $(\lambda_{\min}, \phi_{\min})$ and $(\lambda_{\max}, \phi_{\max})$. It incorporates the following components:

- **Vertiports**: n_v vertiports are modeled as static points in the airspace, each defined by its position $v = (\lambda_v, \phi_v)$.
- **No-fly zones**: The environment contains n_g circular no-fly zones, each defined by a center and radius: $g = (\lambda_g, \phi_g, r_g)$.
- **Airspace corridor**: A cyclic airspace corridor is represented by N control points C_i, where $i \in [\![1, N]\!]$. These serve as the optimization variables and are used to construct the corridor spline, as detailed later in Sec. II-C.

The state representation s is a composite vector comprising three components: $s = [s_v, s_g, s_c]$, illustrated in Fig. 2. s_v encodes the spatial relationships between control points and vertiports (Eq. 1, Fig. 2a).

$$\begin{aligned} s_v &= \left[\mathbf{s}_v^{(C_1)}, \mathbf{s}_v^{(C_2)}, \ldots, \mathbf{s}_v^{(C_{n_v})}\right], \\ \mathbf{s}_v^{(C_i)} &= \left(d_{v_{\min}}^{(C_i)}, d_{v_{\max}}^{(C_i)}, \hat{d}_v^{(C_i)}, \bar{d}_v^{(C_i)}, \theta_{v_{\min}}^{(C_i)}, \theta_{v_{\max}}^{(C_i)}, \bar{\theta}_v^{(C_i)}\right) \end{aligned} \tag{1}$$

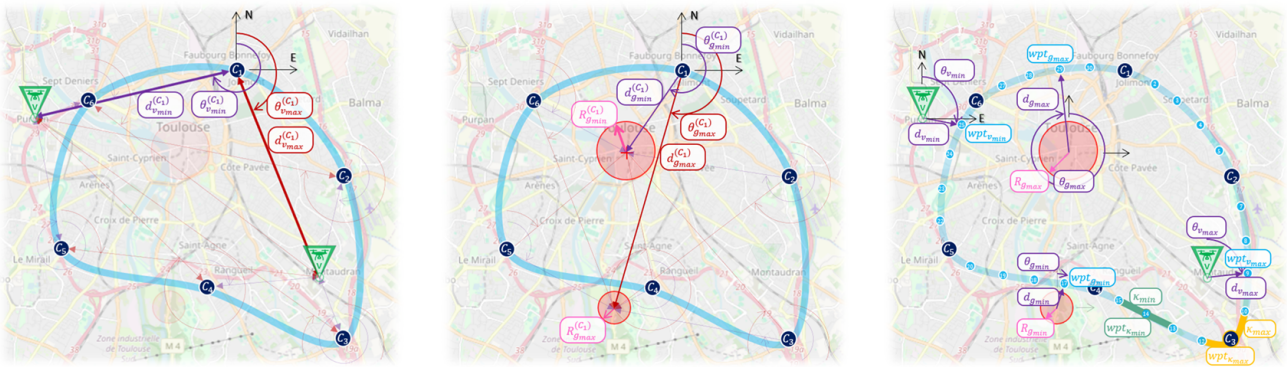
s_g encodes the spatial relationships between control points and no-fly zones (Eq. 2, Fig. 2b).

$$\begin{aligned} s_g &= \left[\mathbf{s}_g^{(C_1)}, \mathbf{s}_g^{(C_2)}, \ldots, \mathbf{s}_g^{(C_{n_v})}\right], \\ s_g^{(C_i)} &= \left(d_{g_{\min}}^{(C_i)}, d_{g_{\max}}^{(C_i)}, \bar{d}_g^{(C_i)}, R_{g_{\min}}^{(C_i)}, R_{g_{\max}}^{(C_i)}, \theta_{g_{\min}}^{(C_i)}, \theta_{g_{\max}}^{(C_i)}, \bar{\theta}_g^{(C_i)}\right) \end{aligned} \tag{2}$$

s_c captures the relationship of both vertiports and no-fly zones with the fully discretized cyclic corridor, including curvature features (Eq. 3, Fig. 2c).

Each control point C_i includes spatial features derived from its relationship to vertiports and no-fly zones. The features $d_{v_{\min}}^{(C_i)}$, $d_{v_{\max}}^{(C_i)}$ and $d_{g_{\min}}^{(C_i)}$, $d_{g_{\max}}^{(C_i)}$ denote the minimum and maximum distances to the closest and furthest elements, respectively. Angular orientations relative to geographic north are captured by θ features. Geofence radii are included as $R_{g_{\min}}^{(C_i)}$ and $R_{g_{\max}}^{(C_i)}$.

To complement these extrema, the state includes the average distances $\bar{d}_v^{(C_i)}$ and $\bar{d}_g^{(C_i)}$ to the observed vertiports and no fly zones, respectively. For vertiports, a weighted separation term $\hat{d}_v^{(C_i)}$ is also included, which emphasises more distant vertiports using a sigmoid based weighting scheme: $w_{v_j}^{(C_i)} = \frac{e^{d_{v_j}^{(C_i)}}}{\sum_{k=1}^{n_v} e^{d_{v_k}^{(C_i)}}}$. The average orientations $\bar{\theta}_v^{(C_i)}$ and $\bar{\theta}_g^{(C_i)}$ are computed using softmax weighted averages over angular directions to account for circularity through $\bar{\theta}_v^{(C_i)} = \operatorname{atan2}\left(\sum_{k=1}^{n_v} w_{v_j}^{(C_i)} \sin\theta_{v_j}^{(C_i)}, \sum_{k=1}^{n_v} w_{v_j}^{(C_i)} \cos\theta_{v_j}^{(C_i)}\right)$ and similarly for $\bar{\theta}_g^{(C_i)}$, but using a softmax on negative distances to give more importance to closer no fly zones: $w_{g_j}^{(C_i)} = \frac{e^{-d_{g_j}^{(C_i)}}}{\sum_{k=1}^{n_v} e^{-d_{g_k}^{(C_i)}}}$.



(a) Vertiports are modeled as static points in the airspace.

(b) The environment contains circular no-fly zones.

(c) A discretized cyclic airspace corridor is represented by control points.

Fig. 2. State Space Representation. The state representation s encodes the spatial relationships between control points and vertiports, no-fly zones, and the fully discretized cyclic corridor.

While s_v and s_g focus on the relationship between control points and environmental elements, s_c, as defined in Eq. 3, captures the configuration of the discretized corridor waypoints relative to both vertiports and no fly zones. It reuses the same types of features discussed above; distances and angular orientations, but this time relative to each waypoint of the corridor rather than the control points. In addition s_c, incorporates curvature related features of the corridor, namely the minimum, maximum, and average curvatures κ_{min}, κ_{max}, $\bar{\kappa}$ respectively. These are complemented by waypoint identifiers wpt, which specify the exact waypoint at which each indexed curvature or spatial extremum occurs.

$$s_c = [d_{g_{\min}}, d_{g_{\max}}, \theta_{g_{\min}}, \theta_{g_{\max}}, R_{g_{\min}}, R_{g_{\max}}, wpt_{g_{\min}}, wpt_{g_{max}}, d_{v_{\min}}, d_{v_{\max}}, \theta_{v_{\min}}, \theta_{v_{\max}}, wpt_{v_{\min}}, wpt_{v_{\max}}, \bar{d}_g, \bar{d}_v, \bar{\theta}_g, \bar{\theta}_v, \kappa_{min}, wpt_{\kappa_{min}}, \kappa_{max}, wpt_{\kappa_{max}}, \bar{\kappa}] \tag{3}$$

2) Action Space : The action space $\mathcal{A}$ governs the positioning of the N control points $(C_i)_{i\in[\![1,N]\!]}$ that define the geometry of the cyclic airspace. As detailed in Sec. II-C, these control points are used to generate spline based representations of the corridor. For a given policy π, the number of control points is fixed, yielding an action space of dimension $2N$, where each control point is defined by its geographic coordinates (λ, ϕ). Accordingly, the action space is expressed as:

$$\mathcal{A} = [a_{1,\lambda},\, a_{1,\phi},\, a_{2,\lambda},\, a_{2,\phi},\, \ldots,\, a_{N,\lambda},\, a_{N,\phi}]$$

with each $a_{i,\lambda}, a_{i,\phi} \in (-1, 1)$. These actions control the displacement of each control point within the normalised airspace domain.

Each update modifies the current position of $C_i(t)$ to the next position $C_i(t+1)$ as follows:

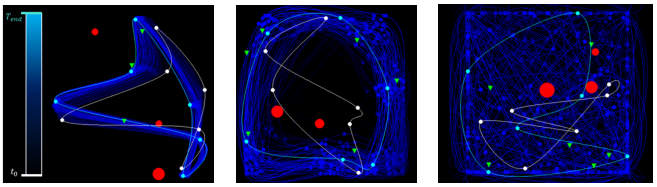
$$C_i(t+1) = \text{clip}\left(C_i(t) + \delta \begin{bmatrix} a_{i,\lambda} \\ a_{i,\phi} \end{bmatrix} \begin{bmatrix} \lambda_{\max} - \lambda_{\min} \\ \phi_{\max} - \phi_{\min} \end{bmatrix}, \begin{bmatrix} \lambda_{\min} \\ \phi_{\min} \end{bmatrix}, \begin{bmatrix} \lambda_{\max} \\ \phi_{\max} \end{bmatrix}\right)$$

where $\delta \in (0, 1]$ is a scaling factor that ensures smooth and bounded updates. The "clip" function enforces hard constraints to keep the control points within the defined airspace bounds.

3) State Space Transition : The scaling factor δ is decisive in shaping the temporal evolution of the control points and, consequently, the learning dynamics. A small value of δ ensures fine grained transitions between states, promoting training stability but requiring many steps to reach an optimal configuration; especially if the initial design is suboptimal. Conversely, larger δ values allow more significant updates and faster exploration of the design space but may lead to unstable or erratic behavior.

This trade-off is illustrated in Fig. 3, which compares the control point trajectories across 100 optimization steps for three values of δ.

The selection of δ thus becomes a sensitive hyperparameter in the RL training process. Larger δ values offer fast convergence potential, but the induced high variance can lead to convergence instability. On the other hand, smaller δ values improve trajectory smoothness but can hinder convergence, particularly when the agent's initial state is far from optimal. While increasing the number of steps per epoch may partially compensate for a small δ, RL algorithms often suffer from insufficient reward signal, since the difference between adjacent states may be minimal.



(a) $\delta = 0.01$ (b) $\delta = 0.1$ (c) $\delta = 1.0$

Fig. 3. State transitions across 100 steps with varying scaling factor δ. The scaling factor balances stability and speed in learning, with smaller values enabling gradual, stable updates, while larger values promote faster, but potentially unstable changes.

We will later explore how our PPO and SAC approaches respond differently to this C^3 optimization under varying δ configurations.

4) Reward Function : It guides the optimization of the cyclic airspace corridor by scalarizing multiple design objectives into a single feedback signal. It penalizes behaviors that are undesirable in operational airspace design, such as incursions into no fly zones (r_g), sharp curvature variations (r_κ), excessive corridor length (r_l), and self-intersections (r_s), while rewarding proximity to designated vertiports (r_v). These components are linearly combined into a global scalar reward r, returned to the agent at each optimization step, as $r = W_g r_g + W_\kappa r_\kappa + W_l r_l + W_s r_s + W_v r_v$.

The term r_g penalizes any incursion of the corridor into no fly zones. It is defined as $r_g = -\frac{1}{n_g}\sum_{k=1}^{n_g} \mathbb{1}_{\{d_{g_k} \leq R_{g_k}\}}$, where $\mathbb{1}_{\{d_{g_k} \leq R_{g_k}\}}$ is an indicator function that returns 1 if at least one waypoint of the corridor lies within the no fly zone k, and 0 otherwise. Here, d_{g_k} denotes the minimum distance between a waypoint and the center of obstacle k, while R_{g_k} is its associated geofence radius.

The reward term r_κ, related to the curvature of the corridor, uses an L_2 smoothness criterion to encourage gradual directional changes. It is defined as $r_\kappa = -\frac{1}{r_{\kappa_{\max}}}\left(\sum_{k=2}^{N} \left|\kappa_{wpt_k} - \kappa_{wpt_{k-1}}\right|^2 + \left|\kappa_{wpt_1} - \kappa_{wpt_N}\right|^2\right)$, where κ_{wpt_k} is the local curvature at waypoint k, and $r_{\kappa_{\max}}$ is the maximum absolute value of this reward observed during training, used for normalisation.

The reward r_l penalizes long corridors to reduce operational inefficiency. It is given by $r_l = -\frac{d_\mathcal{C}}{r_{l_{\max}}}$, where $d_\mathcal{C}$ is the total length of the corridor and $r_{l_{\max}}$ is the maximum length observed during training, used for scaling.

The reward r_s penalizes corridor path that self-intersect which leads to unsafe and inefficient corridor with poor realism relevance. It is given by $r_s = -\mathbb{1}_{\{d_\mathcal{C} \text{ self-intersect}\}}$.

The vertiport related reward r_v promotes the proximity of the corridor to the vertiports. It is defined as $r_v = \frac{1}{n_v}\sum_{k=1}^{n_v} \text{clip}\left(\frac{d_{v_{\max}} - d_{v_k}}{d_{v_{\max}} - d_{v_{\min}}}, 0, 1\right)$, where d_{v_k} is the closest distance between vertiport k and the corridor, $d_{v_{\max}}$ is the distance beyond which no reward is granted, and $d_{v_{\min}}$ is the threshold under which additional proximity yields no further gain.

The rewards r_κ, r_l, r_s, and r_g are defined in the range $[-1, 0]$, reflecting penalization, while r_v is valued in $[0, 1]$, representing a positive incentive. The parametrization of these reward components is detailed in Sec. III-A. In this configuration, greater emphasis is placed on r_s and r_g, which target critical safety concerns such as self-intersections and

obstacle incursions. The term r_v, while weighted less heavily, serves to increase the overall reward by promoting proximity to vertiports. The curvature r_κ and length r_l rewards are assigned relatively small weights and are primarily used to refine the policy's behavior once the key safety objectives have been met.

C. Generating the Splines to Design Cyclic Corridor

In the context of UTM, defining smooth and continuous airspace corridors respond to the realism of UAS operations flight envelops. Among various curve interpolation techniques, Catmull-Rom splines offer a practical approach to design our airspace [21]. Catmull-Rom splines provide local control while ensuring C^1 continuity, making them suitable for real time updates in airspace management using Soft Actor Critic to move the control points of these splines. Additionally, the Catmull-Rom splines ensures that the splines pass on the different control points that can be more easily interpreted as waypoints.

A Catmull-Rom spline is a type of interpolating spline that smoothly passes through a given set of control points $(P_i)_{i\in[\![1,\,N]\!]}$ while maintaining local control over curve segments. Given four consecutive control points P_{i-1}, P_i, P_{i+1}, and P_{i+2}, the interpolated curve segment between P_i and P_{i+1} is defined by the following Eq. 4 where $S(t)$ is the position along the spline (i.e., between P_i and P_{i+1}) at the parameter t with $t \in [0,\,1]$ the normalized segment length

$$S(t) = TMP \tag{4}$$

T is the parameter vector that defines the cubic interpolation $[t^3,\,t^2,\,t,\,1]$. M and P are respectively the Catmull-Rom basis matrix and the control point matrix that defines the local geometry of the spline segment as depicted in the following Eq. 5 where τ is the tension, set at 0.5, that controls how tightly the curve follows the control points.

$$M = \begin{bmatrix} -\tau & 2-\tau & \tau-2 & \tau \\ 2\tau & \tau-3 & 3-2\tau & -\tau \\ -\tau & 0 & \tau & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, P = \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix} \tag{5}$$

Generating the cyclic splines using Catmull-Rom (i.e., C^1 continuity at each point) requires to wrap the control points $(P_i)_{i\in[\![1,\,N]\!]}$ as define by P_W like $P_W = [P_{N-1},\,P_N,\,P_0,\,P_1,\,...,P_N,\,P_0,\,P_1]$. Therefore the airspace requires at least 4 control points to be generated.

D. Modular Optimization Framework for Cyclic Airspace Corridors

In addition to the learning based approaches (PPO and SAC), we evaluate a set of traditional optimization methods as baselines for comparison. These include gradient based and local minimisation algorithms such as BFGS, L-BFGS-B, Powell, Nelder-Mead, SLSQP, and TNC, as well as global optimizers like Differential Evolution and Dual Annealing. All methods are accessed via the `scipy.optimize` module, enabling modular integration within the same environment. While the internal workings of these methods vary, ranging from quasi-Newton updates to population, based stochastic search, they are treated here as black box solvers with a common cost function aligned with the opposite of the reward formulation described in Sec. II-B4.

III. Experiments

The experimental section begins with the setup of the C^3 scenario, detailing the airspace design configuration and the parametrization of our RL methodology in Sec. III-A. Section III-B presents training insights from our RL approach, which are then compared with other optimization methods in Sec. III-C. Finally, the PPO method is deployed in a large-scale case studies in Sec. III-D. The data are available in an online repository for further investigation, along with visual representation samples of the different approaches [22].

A. Setup

1) Corridor Benchmark : We consider a corridor design $\mathcal{C}$ implicitly considering two vertically separated and unidirectional cyclic corridors as depicted in Fig. 1: a corridor taken clockwise by the operators at 300 ft and a counterclockwise-taken corridor at 450 ft, the latter being at the upper boundary of the VLL airspace.

Our corridor $\mathcal{C}$ being the object of our optimization following an RL approach or one of those introduced in Sec. II-D and, is benchmarked against two baseline strategies, illustrated in Fig. 4:

- A Traveling Salesman Problem based (TSP) cyclic corridor that passes through each vertiport only once and minimizing the itinerary while neglecting airspace constraints such as no-fly zones. TSP represents an idealized case where no transitions outside the corridor is permitted.
- A straight-route approach that directly connects each origin–destination pair using a point-to-point (P2P) strategy, without any corridor structuring. This represents the ideal case from an operational efficiency perspective.

The Fig. 4 illustrates an example of the flight path differences between a corridor $\mathcal{C}$, the TSP corridor, and the P2P configurations for a case with three vertiports. For clarity, transition phases between the vertiports and the corridor are omitted. In practice, transitions vary depending on the direction (clockwise or counterclockwise), but the transition length is assumed constant as we always select the nearest waypoint on the corridor.

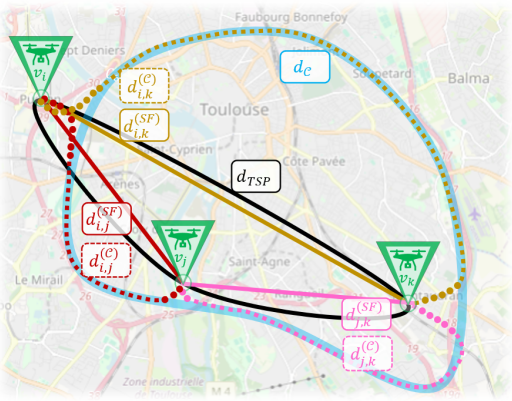


Fig. 4. Benchmarking Operation Efficiency. The optimized corridor $\mathcal{C}$, is evaluated against two baselines: a TSP-based cyclic corridor and a P2P strategy. Resultant flight path structures for an example case with three vertiports.

In Sec. III-D, our PPO approach is tested and evaluated through three large scale case studies taking the size of a

reasonable city with different configuration of vertiports and obstacles with progressive no-fly zone densities as summarized in Table I: The scenarios I and II share the same scale with different no fly zone densities while Scenario III has a higher scale with more homogeneous distribution of the vertiports and no-fly zones.

TABLE I. Large Scale Case Studies

Scenario		**I**	**II**	**III**
Scale	*km*	10.0	10.0	15.0
Vertiports	#	6	8	15
No fly Zones	#	4	6	10
No fly Zones	%	1.96	3.62	1.48

2) Control Variables : The design complexity of a qualitative cyclic airspace depends on the following control variables. Strategically, the number of vertiports n_v, the number of no-fly zones n_g and their spatial density $\mathcal{D}(n_g)$ has an important role during the training and deployment of our models. Tactically, design quality is affected by the separation minima between entering and exiting drones, which in turn relates to vertiport capacity, though capacity constraints are beyond the scope of this study.

3) Metrics : To evaluate the performance of our C^3, we propose the following metrics, which compare operational efficiency when using the corridor versus other routing strategies:

- Transition Phase Distance $d^{\mathcal{C}}_{v_i}$: Distance from vertiport v_i to the closest waypoint on the corridor $\mathcal{C}$. The cumulative transition distance is $d_v = \sum_{i=1}^{n_v} d^{\mathcal{C}}_{v_i}$.
- Transition Phase Proportion $d^{\mathcal{C}}_{v_{\%}}$: For a pair (v_i, v_j), this is the percentage of the total trip spent in transition between vertiport and corridor:

$$d_{v_{\%}}(v_i, v_j) = 100 \left(\frac{d^{\mathcal{C}}_{v_i} + d^{\mathcal{C}}_{v_j}}{d^{\mathcal{C}}_{i,j}} \right)$$

- Extra Flown Distance $d_{\mathcal{C}/SF}$: The distance overhead when using the corridor $\mathcal{C}$ compared to the P2P benchmark:

$$d_{\mathcal{C}/SF}(v_i, v_j) = d^{\mathcal{C}}_{i,j} - d^{SF}_{i,j}, \quad d^{\%}_{\mathcal{C}/SF}(v_i, v_j) = 100 \left(\frac{d^{\mathcal{C}}_{i,j}}{d^{SF}_{i,j}} - 1 \right)$$

- Exploited Corridor Region $X_{\%}$: The percentage of the corridor that is used during operations. Calculated by the ratio of used waypoints to total waypoints N in $\mathcal{C}$.
- Omission of Route Creation: The number of routes that would need to be generated in the case of P2P. This is estimated as $n_v(n_v + 1)$.
- Prevented Rerouting: The number of intersecting routes and no-fly zone incursions observed in the P2P strategy that are avoided by using the corridor.

Additionally, we consider the no-fly zone incursion occurrence n_{risk}, the corridor self-intersection occurrence n_s (e.g., "8" shape), the average flight distance of an operation $d^{\mathcal{C}}_{op}$, and the time to generate an optimized corridor $t_{process}$

4) Parametrization : During training, the cyclic airspace corridor is discretized into $N = 300$ waypoints. For evaluation and for comparison with other optimization methods, a finer discretization of $N = 500$ waypoints is used.

On the RL side, both PPO and SAC are trained using 96 parallel environments. The number of vertiports n_v and no-fly zones n_g is randomly sampled between 3 and 8, and 1 and 8, respectively. Each environment runs for $T_{\text{end}} = 100$ optimization steps, followed by $K = 12$ policy updates. The reward weights used are: $W_g = 3.05 \times 10^{-1}$ for no-fly zone avoidance, $W_\kappa = 5.00 \times 10^{-3}$ for curvature minimization, $W_l = 1.00 \times 10^{-2}$ for length reduction, $W_s = 4.80 \times 10^{-1}$ for self-intersection avoidance, and $W_v = 1.95 \times 10^{-1}$ for vertiport proximity. We remind that other optimizers used in this paper used the same weight while the cost has the opposite sign compared with the reward function. The actor and critic networks for both algorithms use two hidden layers of 512 neurons each with ReLU activation.

SAC is parametrized with a warm-up period of $W = 2.5 \times 10^4$ iterations and a replay memory size $\mathcal{D}_{\text{size}} = 1.0 \times 10^6$. The temperature is set to $\alpha = 1.5$, the buffer batch size is $\mathcal{B}_{\text{size}} = 256$, and the step interval is $\delta = 1.0$. The learning rates are $\lambda_\phi = \lambda_{\mathcal{H}} = 7 \times 10^{-5}$ for the actor and entropy coefficient, and $\lambda_\theta = 1 \times 10^{-5}$ for the critic. Critic target networks are updated using soft updates with $\tau = 5 \times 10^{-3}$.

PPO is configured with a clipping ratio $\epsilon = 0.20$, a mini-batch size of $\mathcal{B}_{\text{minisize}} = 32$, and a GAE parameter $\lambda = 0.95$. The learning rate is $\alpha = 8 \times 10^{-5}$, with critic and entropy coefficients set to $\lambda_\theta = 0.3$ and $\lambda_{\mathcal{H}} = 10^{-4}$, respectively.

B. Training

Both PPO and SAC models have been trained for more than 30,000 epochs before showing stability in their reward returns. While the SAC model exhibited early performance limitations, PPO in Fig. 5 has shown better performance in our implementation with reward signal reaching highest possible scores but with a relative volatility involved by the reward related to the vertiport proximity with the corridor r_v.

The reward related to the no-fly zone incursions with the corridor r_g shows volatility as well but with a much lower reward signal because, when it happens, only one or maximum two are met while the reward is normalized for n_g areas. Self-intersection rapidly converges to the highest score for PPO and stayed stable, the same for the curvature and the length of the corridor shows relatively stability while this last one does not look optimal at the training stage.

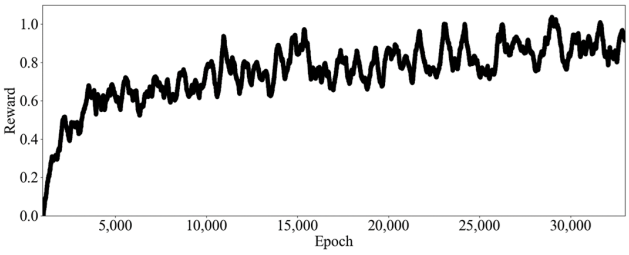
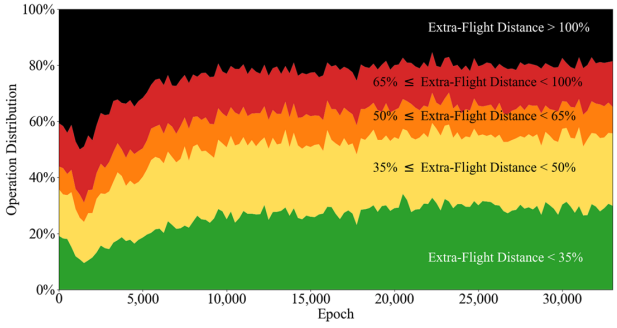


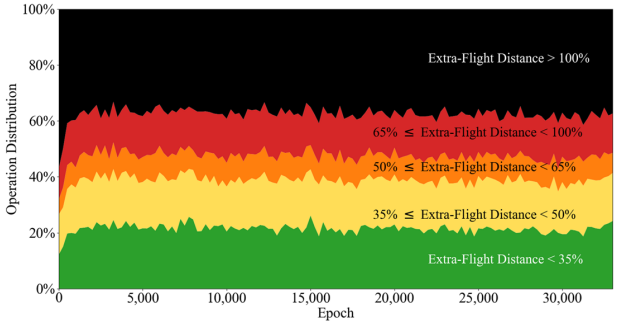
Fig. 5. PPO Reward Evolution throughout Training Epochs

Fig. 6 focuses on the efficiency aspect of the operation when comparing what the operations may ensue by flying between every combination of pairs of vertiports compared with straight route approach. The PPO approach shows efficiency improvement throughout the training but stabilizing early than the reward from the previous figure : The ideal behavior would minimizes the black area related to doubled flight distance using the corridor compared with straight route. This inefficiency represents approximately 20% of the operations at the end of the training compared with 23% of the operations with less than 35% of extra flight distance where

the values between 35% and 100% remains quite stable with light increase of the proportion over the epochs.



(a) PPO



(b) SAC

Fig. 6. Operational Efficiency Distribution Using the Corridor Throughout Training Epochs Compared with Straight Routes. The PPO approach shows efficiency improvement throughout the training but stabilizing earlier than the reward.

C. Comparative Analysis

When compared with others optimizers in Table II, RL approach using PPO algorithm demonstrates a favorable balance between performance and computational efficiency: It achieves low no-fly zone violation and self-intersection rates, high corridor exploitation (99.83%), and real time responsiveness (0.55 seconds per design). In contrast, SAC, while also computationally efficient, suffers from more frequent airspace violations and a higher number of self-intersections, indicating less stable convergence in this application.

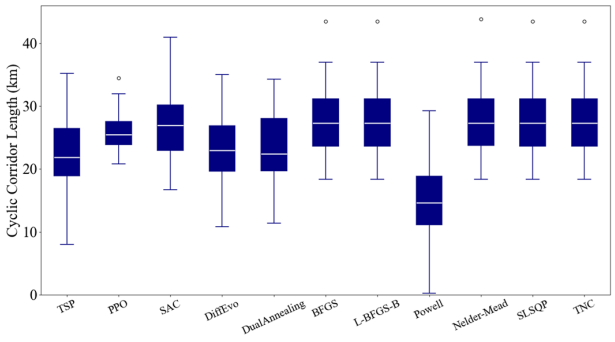
Global optimizers such as Differential Evolution and Dual Annealing perform competitively in terms of solution quality but require significantly higher computation time several orders of 5min per iteration. Gradient based methods (e.g., BFGS, L-BFGS-B) converge faster but are more prone to suboptimal solutions and constraint violations due to local minima. It is supported through Fig. 7b and Fig. 7c where all the minimization functions, at the exception of Powell, show similar optimization and convergence behavior even through their computation times are the shortest.

The TSP baseline offers the shortest transitions from vertiports to corridors but as expected, it completely ignores airspace constraints therefore performs poorly in safety related metrics such as self-intersection and airspace incursions.

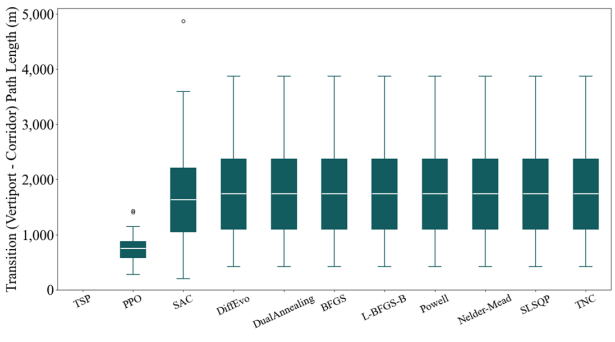
Fig. 7 complements the numerical analysis by illustrating differences in corridor length, transition distance from vertiports, and transition proportion (i.e., time spent outside the corridor) across 25 optimization runs. PPO consistently demonstrates a balanced and realistic corridor layout with minimal deviation from optimal metrics and high adaptability across trials.

TABLE II. Optimization Approach Comparison

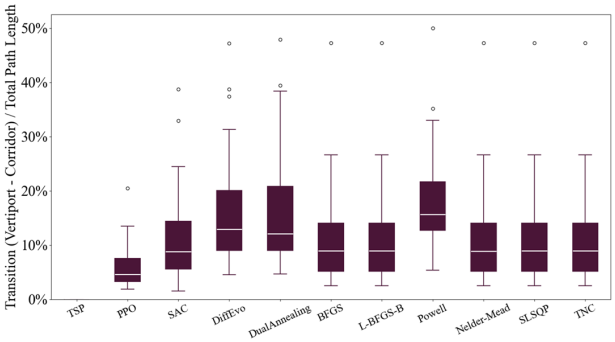
Approach	$d_{op}^{\mathcal{C}}$	$d_{\mathcal{C}/SF}$	n_{risk}	n_s	$X_{\%}$	$t_{process}$
TSP	2.54	0.05	0.03	0.23	100	0.01
PPO	3.06	0.13	0.06	0.03	99.83	0.55
SAC	4.02	0.23	0.17	3.26	99.85	0.58
DiffEvo	2.80	0.09	0.17	0.45	99.86	386.41
DualAnnealing	2.84	0.09	0.03	0.59	99.86	284.08
BFGS	3.96	0.23	0.21	0.83	99.86	0.16
L-BFGS-B	3.96	0.23	0.21	0.83	99.86	0.17
Powell	2.31	0.04	0.07	0.62	99.86	8.46
Nelder-Mead	3.96	0.23	0.21	0.79	99.86	1.75
SLSQP	3.96	0.23	0.21	0.83	99.86	0.16
TNC	3.96	0.23	0.21	0.83	99.86	0.16



(a) Cyclic Corridor Length

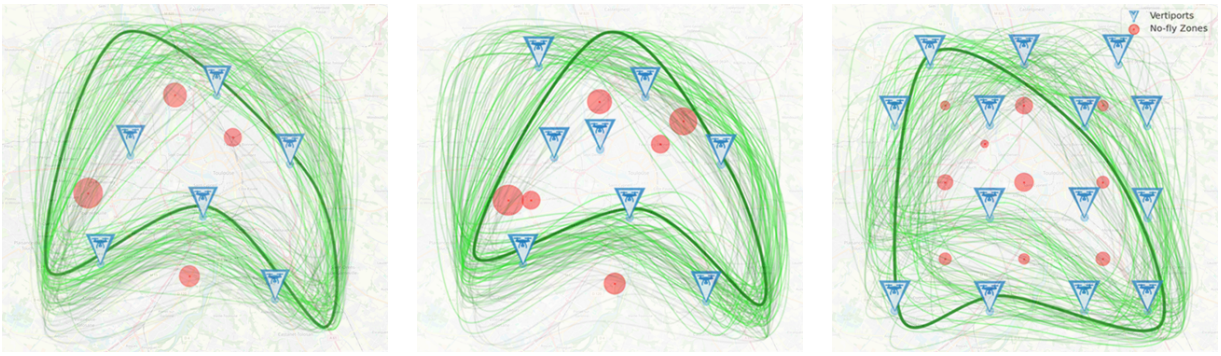


(b) Transition Length



(c) Transition Proportion

Fig. 7. Comparative Operational Design through 25 iterations. Corridor length, transition distance from vertiports, and transition proportion are compared among optimizers. PPO consistently demonstrates a balanced and realistic corridor layout with minimal deviation from optimal metrics and high adaptability across trials.



(a) Scenario I (b) Scenario II (c) Scenario III

Fig. 8. The PPO algorithm was deployed on three large-scale urban airspace scenarios, each with 100 optimization trials. The resulting airspace corridors are visualised with a green gradient where transparency indicates lower reward and thickness reflects higher reward. The darkest and thickest green trajectory corresponds to the highest reward solution.

D. Large Scale Case Studies

In Sec. III-C, PPO has demonstrated superior performance compared to the other tested approaches and has therefore been deployed across the large scale scenarios previously summarized in Table I. Due to the randomness in control point initialization, each scenario was run for 100 trials and may end up lightly differently. Fig. 8 illustrates the resulting airspace corridors for each scenario, where the brighter solution corresponds to the trial that achieved the highest reward. Each proposed corridor required approximately half a second to generate. While a minority of iterations resulted in violations of no-fly zones, the majority produced realistic airspace corridor designs.

The total corridor lengths for the three scenarios are 33.94, 30.72, and 46.09 km, respectively. In all cases, the corridor regions enable operations between various combinations of vertiports. The average horizontal distances between vertiports and the corridor boundaries are 502, 869, and 1,289 m for Scenarios I, II, and III, respectively. These represent 1.84%, 2.41%, and 1.28% of the average total operation length.

Assuming uniform demand between vertiport pairs, 60.00% and 50.00% of operations in Scenarios I and II, respectively, are more than twice as long as their direct route equivalents. This proportion drops significantly to 20.95% in Scenario III. Similarly, 26.67%, 32.17%, and 52.38% of operations are up to 50% longer than the direct distance in Scenarios I, II, and III, respectively. For Scenarios II and III, the corridor generated by the PPO agent surpasses the corresponding TSP derived baseline in terms of compactness and suitability.

Most importantly, PPO shows much lower volatility in its efficiency in comparison with others optimization approaches.

As illustrated earlier in Fig. 1 for Scenario II, the large-scale scenarios were deployed in a synthetic simulator, involving tactical operations navigating between two vertiports using the designed corridor. This helps validate our concept and also highlights potential areas for improvement in tactical traffic management, which could form the basis for further research. A qualitative assessment of the traffic will be discussed at the end of Sec. IV.

IV. Discussion

This section provides a deeper interpretation of the experimental results in Sec. IV-A, followed by a discussion of the research limitations and future directions in Sec. IV-B.

A. Results

PPO-trained model shows a promising fit to handle complex scenarios with tens of vertiports while it has been trained with environments varying from 2 to 8 vertiports and obstacles from 1 to 8; routes are quite realistic with smooth curvatures and visually intuitive for any UTM service provider that may have to manage a complex flight network in an urban airspace area. The RL algorithm validates most of our criterion with a prompt response compared to competitive global optimizers that takes 5min and more in complex scenarios showing promises for dynamic aspect of this C^3 considering operational demand and airspace capacity in real-time.

For Scenario I, the point-to-point vertiport itineraries lead to numerous intersections that must be resolved at the strategic management level, with only five out of fifteen vertiport connections remaining intersection-free. These limitations highlight the potential of the C^3 approach, particularly in Scenario II, where it prevents 23 reroutings out of 28 connections. In Scenario III, C^3 prevents 95 rerouting among 105 connections when compared to point-to-point itineraries. When using point-to-point itineraries, the scenarios involve resolving 11, 46, and 1,127 flight plan intersections for Scenarios I, II, and III respectively, along with 2, 5, and 31 no-fly zone incursions. In contrast, C^3 successfully eliminates most of these conflicts within a second.

Additionally in a point-to-point flight planning strategy, each vertiport may need to simultaneously manage arrivals up to $n_v - 1$ other directions, significantly increasing the risk of in air conflicts at the arrival.

The above results support the use of the C^3 approach to reduce the workload associated with UTM service provision, particularly in flight authorization protocols. From an efficiency perspective, the use of a corridor in Scenarios I and II comes at the cost of increased flight distances. Up to 60.00% and 50.00% of operations, respectively, experience a doubling of flight distance compared to straight point-to-point itineraries. However, only 26.67% and 32.17% of operations in Scenario I and Scenario II experience more than a 50% increase in flight distance, indicating that most operations remain relatively efficient. Notably, C^3 becomes significantly less penalizing and even advantageous in more complex

vertiport configurations, as illustrated in Scenario III, where only 21.00% of operations result in doubled flight distances, and 52.38% of operations incur less than a 50% increase. The benefit is even more compelling when considering that many point-to-point itineraries are practically infeasible, often resulting in longer actual flight distances. In this context, C^3 offers a more realistic and efficient benchmark.

From a safety perspective, the three scenarios were deployed in a simulator with 15 operations each, using a basic vertiport management system. This system authorizes takeoff only when the surrounding corridor is available and no conflicting operations are expected at the entry time. The simulations show the importance of vertiport proximity to the corridor: In Scenario I, the corridor passes above a vertiport, enabling efficient routing but resulting in local traffic densification. Six operations experienced a horizontal loss of separation below 2,000 ft, although no hazardous separation are visible throughout. Scenario II represents the best case, with consistently safe separation between all operations and more uniform transition distances across all vertiports. Scenario III also maintains safe separation; however, one no-fly zone incursion was observed during the departure and arrival phases of the top-right corner vertiport, as shown in Fig. 8c.

B. Limitations and Future Work

PPO and SAC has been fairly set up to handle the same problem but the implementation of the SAC may lack of maturity to reveal a definitive performance of it. Additionally including the PPO, the positive results may also be underestimated while our experiments used only 6 control points in maximum, we struggled to achieve stable model convergence over 6 control points to get a more complex cylic shape with higher dimensional neural network size.

On efficiency aspect, the efficiency of proposed solutions can be debated depending on the complexity of the urban airspace; somehow it could be very inefficient in simple cases where a very few vertiports are implemented or very convenient in complex case studies where prompt flight-authorization may compensate extra-flight distance induced by the shared-corridor. One option was to incorporate these metrics directly into the reward function; however, this significantly slowed down the training process.

From a safety perspective, it is important to propose a quantitative tactical validation of our cyclic corridor design. This would further support the validity of C^3 by introducing safety metrics and assessing the impact of implementing parallel lanes, which depend on aircraft performance to maintain separation minima effectively. Additionally, further work is required to address flight transitions between vertiports and the corridor when no-fly zones are located in between, as observed in Scenario III. This includes managing entry points into the corridor and ensuring safe separation of operations when vertiports are situated in close proximity.

Because the truth often lies somewhere in the middle, a promising direction would be to propose a hybrid approach that combines RL with other methods to address more complex cyclic designs, such as C^4, which may involve concentric features and dynamically adapting corridors based on demand. This strategy could incorporate P2P routing technics where the approach proves more relevant or efficient.

V. Conclusion

This work introduced the Cyclic Corridor Concept C^3, a novel approach for urban airspace design that leverages DRL to generate safe, efficient, and scalable shared routes for UAS operations. By offloading strategic deconfliction and enabling near real-time design updates, C^3 demonstrates strong potential to reduce UTM workload while maintaining operational feasibility in complex environments. Our results confirm that DRL, particularly PPO can robustly optimize airspace layouts under realistic constraints.

The future research explores hybrid methods combining learning-based and rule-based strategies may further enhance adaptability, enabling future extensions such as C^4 to dynamically respond to demand and integrate with point-to-point routing where appropriate.

References

[1] Federal Aviation Administration, "Urban Air Mobility, Concept of Operations," Release, Version 2, 2023.

[2] SESAR Joint Undertaking, "U-Space Blueprint," 2019, accessed online: 2024-12-19.

[3] ——, "U-space ConOps and architecture, Edition 4," 2023.

[4] European commission, "Regulation 2021/664 U-space Regulatory Framework," 2021.

[5] J. Hoekstra and J. Ellerbroek, "BlueSky ATC simulator Project: An Open Data and Open Source Approach," in 7^{th} *International Conference on Research in Air Transportation*, 2016.

[6] A. Bauranov and J. Rakas, "Designing airspace for urban air mobility: A review of concepts and approaches," *Progress in Aerospace Sciences, vol. 125, 100726*, 2021.

[7] International Civil Aviation Organization, "International standards and recommended practices: Annex 15 to the Convention on International Civil Aviation – Aeronautical Information Services, 16^{th} ed." 2015.

[8] D.-S. Jang, C. Ippolito, S. Sankararaman, and V. Stepanyan, "Concepts of Airspace Structures and System Analysis for UAS Traffic flows for Urban Areas," in *AIAA SciTech Forum*, 2017.

[9] U.-J. Lee, S.-J. Ahn, D.-Y. Choi, S.-M. Chin, and D.-S. Jang, "Airspace Designs and Operations for UAS Traffic Management at Low Altitude," *Aerospace*, vol. 10, no. 9, 2023.

[10] Z. Zhang, Y. Zhengand, C. Li, B. Jiang, and Y. Li, "Designing an Urban Air Mobility Corridor Network: A Multi-Objective Optimization Approach Using U-NSGA-III," *Aerospace*, vol. 12, no. 3, 2025.

[11] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in 35^{th} *International Conference on Machine Learning, PMLR 80:1861-1870*, 2018.

[12] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2019.

[13] G. M. Barros and E. L. Colombini, "Using soft actor-critic for low-level UAV control," *arXiv preprint arXiv:2010.02293*, 2020.

[14] M. H. Lee and J. Moon, "Deep reinforcement learning-based model-free path planning and collision avoidance for UAVs: A soft actor–critic with hindsight experience replay approach," *ICT Express*, vol. 9, no. 3, pp. 403–408, 2023.

[15] M. Ribeiro, J. Ellerbroek, and J. Hoekstra, "Improving Algorithm Conflict Resolution Manoeuvres with Reinforcement Learning," *Aerospace*, vol. 9, no. 12, 847, 2022.

[16] P. Razzaghi, A. Tabrizian, W. Guo, S. Chen, A. Taye, E. Thompson, A. Bregeon, A. Baheri, and P. Wei, "A survey on reinforcement learning in aviation applications," *Engineering Applications of Artificial Intelligence, 136, 108911*, 2024.

[17] Y. Xu, D. Hu, L. Liang, S. McAleer, P. Abbeel, and R. Fox, "Target Entropy Annealing for Discrete Soft Actor-Critic," *arXiv preprint arXiv:2112.02852*, 2021.

[18] OpenAI, "Soft Actor-Critic, *Spinning Up*," 2018.

[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms, arxiv 1707.06347," in *arXiv:1707.06347 [cs.LG]*, 2017.

[20] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-Dimensional Continuous Control Using Generalized Advantage Estimation," *CoRR*, 2015.

[21] C. Yuksel, S. Schaefer, and J. Keyser, "Parameterization and applications of Catmull–Rom curves," *Computer-Aided Design*, vol. 43, no. 7, pp. 747–755, 2011.

[22] R. Fremond, "Research Data, *DASC2025, Github*," 2025.