A Meta-reinforcement Learning Method for Adaptive Payload Transportation with Variations

Jingyu Chen[a,1,*], Ruidong Ma[b], Meng Xu[c], Fethi Candan[b], Lyudmila Mihaylova[b], John Oyekan[d]

[a]*Institute of Software, Chinese Academy of Sciences, Beijing, China*
[b]*Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield, United Kingdom*
[c]*School of Information Technology & Management, University of International Business and Economics, Beijing, China*
[d]*Department of Computer Science, University of York, York, United Kingdom*

Abstract

The safe transport of cable-suspended payloads by a group of Unmanned Aerial Vehicles (UAVs) depends on their capacity to effectively respond to fluctuations in the dynamics caused by external variations, such as wind gusts. For group transportation with obstacles, internal variations, such as changes in formation, can also alter the space occupancy of the system related to collision detection. However, traditional adaptive learning methods are challenging to adapt to these two variations. In this paper, we present a learning-based method for collision-free dual-UAV-payload transportation in the presence of varied wind force and formation change. It consists of an adaptive trajectory tracking controller based on meta-model-based reinforcement learning with online adaptation and a novel correction policy, and a path planner that can sample collision-free goal states of the system for the controller based on the meta-collision predictor. The simulation results demonstrate that the proposed trajectory tracking controller outperforms state-of-the-art model-free, model-based, and variational inference methods in terms of payload tracking error reduction and robustness when dealing with the variations mentioned above. Specifically, the proposed controller reduces the average payload tracking error to less than 0.1 metres in most tasks without obstacles. Furthermore, by following the adapted paths generated by the path planner, the trajectory tracking controller can effectively track the payload while ensuring collision-free safety of the dual-UAV-payload system during navigation among obstacles. The success rate of the proposed method is more than 80% in all scenarios with obstacles. Our project website can be seen at `https://sites.google.com/view/meta-payload-fly/` and the source code is available at `https://github.com/wawachen/Meta-load-fly`.

Keywords: Reinforcement learning, Meta-learning, Cooperative transportation, Trajectory tracking, Path planning

1. Introduction

In the last few decades, the transportation of cable-suspended payloads by multiple UAVs has been well investigated. Taking into account the coupled dynamic effects resulting from the interaction between a multi - UAV system and a slung load, previous literature has endeavored to tackle this challenge through various methods. Some studies adopted a decentralised leader - follower approach [1]. Others have attempted to address it by applying a full dynamics model on a nonlinear manifold [2]. Additionally, [3][4] explored the utilization of swarm behaviours to deal with this problem. However, these approaches rely heavily on expert knowledge regarding the underlying physics of the system. Recently, researchers have increasingly focused on applying model-free reinforcement learning (MFRL) [5] to realise various robotics tasks such as flight control [6], navigation [7] [8], hovering [9], landing [10][11].

In MFRL methods, an explicit model is not required. Instead, the optimal policy is learned through extensive interactions with the environment. The applications of MFRL on cable-suspended transportation has been explored in [12][13]. However, model-free RL methods suffer from low data efficiency and high costs, and they may encounter potential risks during exploration. On the other hand, model-based reinforcement learning (MBRL), that make use of deep learning [14] to generate a black-box dynamics model, has found extensive applications in creating data-driven dynamics models for small legged robots [15], robot arms [16], helicopters [17] and quadrotors [18][19]. The data-driven methods for system identification are useful when domain knowledge is scarce or deriving system equations is hard. In cooperative payload transportation, the system has highly non-linear and coupling features, making precise mathematical modeling very challenging. Moreover, the cooperation among UAVs requires effective coordination, highlighting the utility of data-driven methods for this task.

The UAVs involved in payload transportation are often subjected to disturbances such as wind gusts, especially in outdoor environments. These wind gusts are often unpredictable and as a result, a non-adaptive controller would not be able to ad-

*Corresponding author
Email addresses: chenjingyu@iscas.ac.cn (Jingyu Chen), rma17@sheffield.ac.uk (Ruidong Ma), xumeng@uibe.edu.cn (Meng Xu), fcandan@uidaho.edu (Fethi Candan), l.s.mihaylova@sheffield.ac.uk (Lyudmila Mihaylova), john.oyekan@york.ac.uk (John Oyekan)

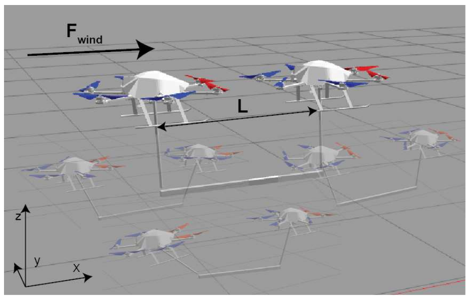


Figure 1: Showing the considered variations wind gust force (F_{wind}) and the distance (L) between individuals during payload transportation in this work. Both variations result in changes to the dynamics and space occupancy of the cooperative UAV-payload system.

just their parameters to account for such unpredictable changes. Furthermore, in a dual-UAV-payload system, the distances between the individual UAVs have an impact on the system dynamics [20]. As a result, both the wind gusts and the distance between the individual UAVs need to be considered in the task of payload transportation as shown in Fig.1.

In order to address these variations, traditional controllers have been focused on identifying the unmodeled dynamics of the system [21][22]. In addition to the dynamics model, the aforementioned variations also affect the space occupancy of the UAV-payload system. This is related to collision detection when planning a navigable path in an environment with obstacles. In this case, incorrect collision constraints can pose a threat to the system. Towards solving this, MFRL methods often adopt the strategy of online searching for collision-free trajectories by massive interactions with the environment [23][24]. In MBRL, an uncertainty collision predictor was learned to estimate the risk and penalise the sampling of actions that lead to catastrophic states. However, when the space occupancy of the UAV-payload system is changed by the variations, the complexity of collision avoidance increases significantly.

In this paper, we propose a meta-reinforcement learning method for adaptive payload transportation with collision avoidance. The proposed method involves effectively tracking safe trajectories from a path planner using a trajectory tracking controller under variations. To account for different state distributions caused by variations, we transform the learning of the dynamics model into a few-shot learning problem. This allows the neural network-based model to be quickly adapted to a new scenario using a few context samples. The meta-dynamics model is first trained by offline human demonstrations using meta-learning. Subsequently, the model is adapted online in a model-predictive-control (MPC) controller to obtain the optimal action toward goal states. Finally, the optimal action from MPC is fine-tuned by a model-free action correction policy online for better performance. To guarantee collision-free transportation, the path planner relies on a self-awareness collision predictor to create safe trajectories for the UAVs to follow. This predictor uses a neural network to determine whether a given spatial point is occupied by the robot's body. As a result, by querying the spatial points from obstacles, we can evaluate whether there are any collisions between the UAV-payload system and obstacles along the sampled trajectories generated by the rapidly-exploring random tree (RRT) algorithm. This enables us to avoid unsafe trajectories during the sampling of collision-free paths.

Specifically, our key contributions include:

- Proposing a design using a virtual leader for meta-model-based reinforcement learning training with offline human demonstrations in dual-UAV-payload systems. This enables online adaptation of the learned meta-dynamics model for the trajectory tracking of both UAVs and the payload;
- Incorporating a model-free action correction policy into meta-model-based reinforcement learning to enhance trajectory tracking performance by mitigating the impact of inaccurate meta-models. Towards this, we offer two different correction strategies for the payload tracking scenario and the UAV-payload tracking scenario. Our simulation results demonstrate that the proposed trajectory tracking controller outperforms state-of-the-art methods in the relevant field.
- A novel path planner that integrates a meta-collision predictor with the traditional Rapidly-exploring Random Tree (RRT) algorithm. The planner samples collision-free goal states, ensuring high-success-rate collision-free payload transportation via our trajectory tracking controller.
- Self-modeling of a dual-UAV-payload system and its relation to obstacles for collision detection and avoidance while participating in cooperative payload transportation. This is unlike previous literature that focused mainly on geometrical methods.

The rest of the paper is organised as follows: In Section 2, we briefly introduce the background of meta-model-based reinforcement learning and the applications of obstacle avoidance. The problem is formulated in Section 3. Then, Section 4 illustrates our proposed method based on an adaptive trajectory tracking controller and a path planner. The experiment results and discussions are shown in Section 5 followed by the final conclusion in Section 6.

2. Related Work

In model-based reinforcement learning (MBRL), both the transition model of the system and the policy are learned and optimized to improve decision-making. The transition model is often approximated by non-parametric methods like Gaussian Processes (GPs) [32] and neural networks (NN) [33]. However, traditional MBRL methods can struggle with generalizing to new tasks due to their reliance on a fixed state distribution assumption. Gradient-based meta-learning [34] has been widely studied in multi-task learning for classification and regression problems where it adapts the meta-model into a task-specific one by several gradient steps. Moreover, meta-learning of the latent variables [27][35] and embeddings [36] can be quickly adapted for new tasks by an online inference algorithm using the maximum likelihood estimation of new samples. However, it may require intractable probability calculations.

Table 1: Comparisons between the latest related works and our proposed method

Research References	Payload	Controller	Learning	Cooperative	Disturbance	Obstacle avoidance
[25]	✗	model-based RL	✓	✗	✗	✗
[26]	✗	meta-learning & PID	✓	✗	✓	✗
[27]	✓	variational inference	✓	✗	✓	✗
[28]	✗	neural network	✓	✗	✓	✗
[29]	✓	MPC	✗	✓	✗	✗
[30]	✓	MPC	✗	✓	✗	✓
[31]	✓	model-free RL	✓	✓	✗	✗
Ours	✓	**model-based RL, meta-learning**	✓	✓	✓	✓

In order to address the limitations of MBRL, [15] introduced a meta-learning approach that requires parallel environments to collect batch data for the online training of the robot's meta-dynamics model. Nevertheless, this is difficult to satisfy especially when computation power is limited. Parallel training can be replaced by learning different tasks sequentially [37]. However, it requires longer training times. Therefore, [26] combined an offline meta model with a PID controller to realise online adaptation. Furthermore, [28] found a wind-invariant basis function by meta-learning and designed a composite adaptive law based on it to mitigate the aerodynamic effects. In robotics tasks, most meta-learning methods focus on building a perfect dynamics model called regression-oriented methods. The alternative approach is the control-oriented method in which meta-learning is applied in a closed control loop to design an adaptive controller [38]. In this work, we adopt the architecture of offline meta-training and online adaptation but focus on the solution for a MPC controller. Prior literature has noted that reducing the prediction error of the dynamics model does not necessarily result in higher rewards for downstream tasks [39]. Therefore, instead of improving the meta-dynamics model, we aim to incorporate a model-free policy to correct the optimal actions produced by the MPC controller. This enables us to counteract the bias of the adapted model. In [40], they focused on designing a corrective policy for a simple PID controller using deep reinforcement learning for autonomous UAV landing tasks. However, they did not consider the variations that might arise from the environment.

Obstacle avoidance can be achieved by reactive methods like potential field [41], model predictive control (MPC) [42] and velocity obstacle [43] which effectively handle constraints and enable real-time planning. Furthermore, path planning algorithms such as A* [44], rapidly-exploring random tree (RRT) [45] and probabilistic roadmap [46] can sample optimal paths that guide robots from a given starting point to a desired goal while considering collision constraints. The safety problem has been considered in the model-based RL, especially for some hazardous tasks like autonomous driving [47] and aerial navigation tasks [48]. These works often pre-train a collision predictor to avoid visiting catastrophic states when sampling the actions. In MFRL, adding high-risk experiences into the training can improve the performance of obstacle avoidance [49]. However, the literature has rarely discussed solutions for situations where the robot's space occupancy may change due to variations in the environment. Therefore, we aim to develop a collision predictor that can be adaptive to variations in the space occupancy of the system. The recent progress in neural representation learning can reconstruct the signed distance functions very well through the use of a neural network (NN) with periodic activation functions [50]. The signed distance function is often utilised to construct the occupancy map for collision detection. Taking advantage of the code vector [51], we design a neural network to predict space occupancy by encoding the state information of the robot system within the workspace. This process can be regarded as a self-modelling strategy that has been utilised in the motion planning of humanoids [52]. The most similar work is [53] where authors proposed a visual representation of a robot arm in 3D space. However, we focus on the cooperative UAV-payload system with 2D collision avoidance. Additionally, [54] developed a reachability estimator to facilitate the RRT method for obstacle avoidance. Their estimator predicts the travelling time to a state but we focus on the prediction of the space occupancy.

In order to improve the clarity of contributions, as shown in Tab.1, we have compared our approach with the current state-of-the-art research. We focused on aspects such as the controller design, whether the approach is applicable to multiple drones, its consideration of disturbances, and the effectiveness of obstacle-avoidance capabilities.

3. Problem definition

According to Fig.1, we formulate variations as v_d to be the distance between two UAVs (neighbour distance) and v_f to be the wind force generated by the static wind gusts in the x axis of the inertial coordinate system. We generate different tasks by creating various combinations (v_f, v_d) from a set $\mathbf{S} := (v_f, v_d)$ for training and validation. The dynamics and space occupancy of the dual-UAV payload system differ across tasks. The main objective of this work is to design a learning-based trajectory tracking controller that can adapt to **varied dynamics models** and a path planner that can adapt to **varied space occupancy** of the system during the testing tasks. We discuss this further as follows:

(1). Adaptation to varied dynamics model: We build the dynamics model of a UAV-payload system as $\mathbf{s}' = f(\mathbf{s}, \mathbf{a})$ where $\mathbf{s}$ is the full state of the system and $\mathbf{a}$ is the action. This model f can be utilised to predict the future state $\mathbf{s}'$ based on the current

s and **a**. Our goal is to track the payload trajectory by finding a policy $\mathbf{a}(t)$ that enables the payload position $p_l \in \mathbf{s}$ to converge to its reference goal $g_l(t)$ based on f. In an environment with obstacles, we also consider additional constraints to ensure that the positions of two UAVs, $p_{u1}, p_{u2} \in \mathbf{s}$, converge to their reference goals $g_{u1}(t)$ and $g_{u2}(t)$ for UAVs. We assume the system maintains stability after executing each action. To handle the variations in system dynamics across different tasks, adapting f into the task-specific model f' is required for downstream control tasks.

(2). Adaptation to varied space occupancy: The collision predictor ϕ will assist the rapidly-exploring random tree (RRT) algorithm in evaluating the safety of sampled goal trajectories $g(t)$ where $g_l(t)$, $g_{u1}(t)$ and $g_{u2}(t) \in g(t)$. It is defined as $\hat{sd} = \phi(x|\mathbf{z})$, where ϕ takes the spatial point x and the system's goal state $\mathbf{z}$ as inputs to output a predicted signed distance function $\hat{sd}$, which is utilised to evaluate collisions between the system and obstacles. To ensure that the collision predictor remains accurate across different tasks, it extracts latent information from $\mathbf{z}$ to adapt to changes in space occupancy of the UAV-payload system.

(3). Collision-free UAV-payload trajectory tracking: In this study, we assume that the information of obstacles is known and perfectly observed. The obstacles $O_{1:m}$ are modeled as cylinders with infinite height. As a result, the collision constraints only need to be checked between the dual-UAV-payload system and obstacles in a $2d$ $x-y$ plane. The global path P_d of the payload is predefined, and the proposed path planner is required to re-plan a collision-free path P_{dc} for two UAVs and payload while fulfilling the safety constraints under different tasks. Furthermore, the reference goal at time t, $g(t) \in P_{dc}$. Since we maintain the orientation of each drone, the UAV-payload system operates as an omnidirectional vehicle controlled through 3D position tracking, and no additional motion constraints are required. In non-obstacle scenarios, only the payload is tracked. However, in the presence of obstacles, full state tracking for the UAV-payload system is required. The collision-free policy is achieved by tracking the sampled trajectories P_{dc} using a controller with acceptable accuracy.

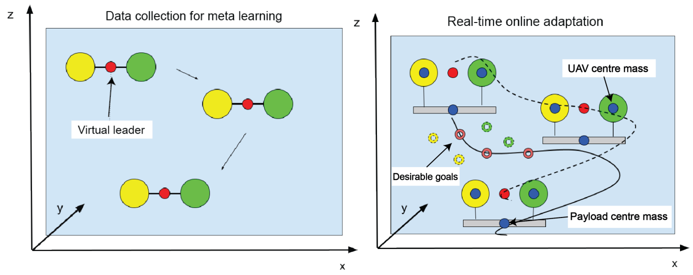


Figure 2: The design of virtual leader. Left: data collection for the offline meta-model; Right: Real-time adaptation for tracking the payload and UAVs

4. Methodology

4.1. Reinforcement learning setup

We formulate the problem of the UAV-payload system in a Markov Decision Process (MDP). This UAV-payload system is actuated by two hexacopters. Here, we consider the tuple (S, A, S', R, P, H) where S is the current state set; S' is the next state set; R is the reward, P is the transition probability of the environment and H is the horizon. The current state $\mathbf{s} \in S$ captures the full state of the system, $\mathbf{s} = [p_l, p_{u1}, p_{u2}, \Omega]$ where $p_l \in \mathbb{R}^3$ is the payload position, $p_{u1} \in \mathbb{R}^3$ and $p_{u2} \in \mathbb{R}^3$ are the positions of two UAVs respectively. Ω is the Euler angle (roll, pitch, yaw) of the payload. A neural network f is used to approximate the transition probability $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$, also known as the dynamics model. Two UAVs are commanded to follow a virtual leader situated between them using a position controller called *Lee controller* [55], which maintains a stable formation for the system. When the position of the virtual leader is given as $p_v = [x_v, y_v, z_v]$ in the inertial coordinate, the position of UAV_1 and UAV_2 are computed by $p_{u1} = [x_v + L/2, y_v, z_v]$ and $p_{u2} = [x_v - L/2, y_v, z_v]$ where L is the distance between two UAVs. We define the action $\mathbf{a} \in A$ as the relative position change with respect to the virtual leader where $\mathbf{a} = [\delta p_x, \delta p_y, \delta p_z]$. Thus, the reference goal of the virtual leader is updated by $p_g = [x_v + \delta p_x, y_v + \delta p_y, z_v + \delta p_z]$. As shown in Fig.2, we aim to design a policy to command the position of the virtual leader p_g (red points) to ensure that the centre mass of the payload (blue point) converges to the desirable payload trajectory (orange ring). In the scenarios with obstacles, the centre mass of the UAVs (blue point) must converge to the desirable UAV trajectories (yellow and green rings) because of collision constraints.

4.2. Offline meta training for dynamics model

Model-agnostic meta-learning (MAML) [34] is applied to train the meta-dynamics model f in an offline fashion. We collect the dataset containing states **s** and actions **a** defined in the last section for various tasks. We use a Xbox 360 controller to change the position of the reference goal of the virtual leader. For each time step, the position of the reference goal is changed within ±0.03m in the $3d$ axis by the joystick signal. Here, several random trajectories over the workspace are sampled by the open-loop human control without taking the position of the payload into account. For each training task, we collect 2500 data points every 0.15 seconds during the operation of the human.

For each training task $S_{tr}^{i=1:n}$, we divide the whole trajectory P^i into five sub-paths P^i_0,P^i_1,...,P^i_5. Inspired from [56], each trajectory τ_i chosen from P^i is further regarded as a new task where $\tau_i = [\mathbf{s}_{\tau_i}(t), \mathbf{a}_{\tau_i}(t), \mathbf{s}_{\tau_i}(t+1)]_{t=1:T}$. We use past K samples $\tau_i(t-K)$ to predict the future Q samples $\tau_i(t+Q)$. Thus, the training dataset consisting of total N trajectories is organised into samples x and labels y as shown in Eq.(1), where t is current timestep in trajectory τ_i, ∀i∈ $[1, N]$ and ∀ t∈ $[1, T]$. T is the total timestep in τ_i. The change of the state transition is predicted by the concatenation of the current state and action. All states and actions are normalised into $[0, 1]$.

$$\begin{aligned} x_{\tau_i} &= [\mathbf{s}_{\tau_i}(t), \mathbf{a}_{\tau_i}(t)] \\ y_{\tau_i} &= \mathbf{s}_{\tau_i}(t+1) - \mathbf{s}_{\tau_i}(t) \end{aligned} \tag{1}$$

We consider the meta-learning problem as a n-way-$(K+Q)$-shot few-shot learning. The initial meta learner is approximated by a feed-forward neural network f_θ with parameter

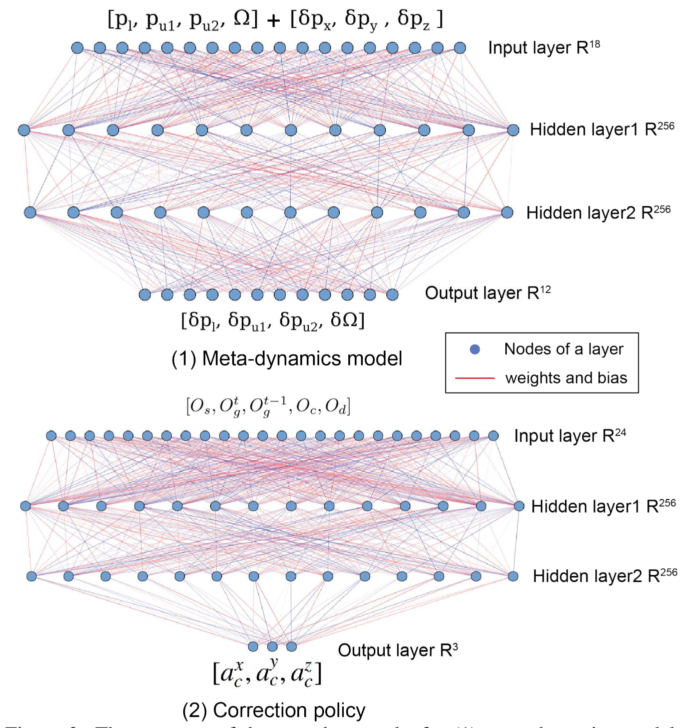


Figure 3: The structure of the neural networks for (1) meta-dynamics model and (2) actor network of the correction policy. $[\delta p_l, \delta p_{u1}, \delta p_{u2}, \delta\Omega]$ is the state change. $[a_c^x, a_c^y, a_c^z]$ is action of the correction policy which is different from the action $[\delta p_x, \delta p_y, \delta p_z]$ of the MPC policy.

θ. The task loss is defined as the mean squared error $\mathcal{L}_{\tau_i} = MSE(f_\theta(x_{\tau_i}), y_{\tau_i})$. In each task τ_i, the meta network f_θ is firstly updated by the context K samples $\tau_i(t-K)$ (Eq.(2)).

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\tau_i(t-K)}(f_\theta) \tag{2}$$

Then, the meta loss across all tasks is minimised by the query Q samples $\tau_i(t+Q)$(Eq.(3)).

$$\theta = \theta - \beta \nabla_\theta \sum_{\tau_i \in P} \mathcal{L}_{\tau_i(t+Q)}(f_{\theta_i'}) \tag{3}$$

where α and β are the learning rates for each step. We randomly sample n tasks from the dataset $(x_{\tau_i}, y_{\tau_i})_{i=1:N}$ and organise them into batch data for the training. As shown in (1) of Fig.3, the meta-model f_θ consists of two hidden layers with 256 nodes and ReLU activation functions.

4.3. Control with online adaptation

At the online test time, we sample a testing task S_{te}^j from the task distribution $\rho(S_{te})$. This task can be in or out-of-distribution of the training tasks. The online adaptation follows the same rule Eq.(2) as the offline training. Here, an adaptation buffer D_a is incorporated to store the past trajectories. For each timestep t, we use the past K samples $\tau_i(t-K)$ from D_a to adapt the meta-model θ obtained from the Section 4.2 into θ_a. The adapted dynamics model f_{θ_a} is utilised in the model predictive control (MPC) for downstream trajectory tracking tasks. The optimisation of action sequences sampled by the cross-entropy method (CEM) depends on the adapted model, where M populations of actions are sampled over a horizon of T. The cost function J of each stage k is calculated by Eq.(4) for non-obstacle scenarios and Eq.(5) for obstacle scenarios,

$$J(\mathbf{s}_k, \mathbf{a}_k) = |\hat{p}_l^{k+1} - g_l^{k+1}| + \eta |a_k|^2 \tag{4}$$

$$J(\mathbf{s}_k, \mathbf{a}_k) = |\hat{p}_l^{k+1} - g_l^{k+1}| + |\hat{p}_{u1}^{k+1} - g_{u1}^{k+1}| + |\hat{p}_{u2}^{k+1} - g_{u2}^{k+1}| + \eta |a_k|^2 \tag{5}$$

where $\hat{p}_l^{k+1}$, $\hat{p}_{u1}^{k+1}$ and $\hat{p}_{u2}^{k+1}$ represent the predicted positions of the payload, UAV1 and UAV2 respectively using f_{θ_a}. a_k is the current action. g_l^{k+1}, g_{u1}^{k+1} and g_{u2}^{k+1} are the corresponding goal positions of the payload and UAVs provided by the path planner. η is the penalty coefficient for penalising sampled actions with large values. The action populations with the lowest costs will be selected. Only the first action of the optimal action sequence will be executed for control. The analytical solution of the optimal action a^* of the MPC controller is shown in Eq.6 where $\mathbf{s}_{t+T}$ is the terminal state.

$$\mathbf{a}_t^* = \arg\min_a \left(\sum_{k=t}^{t+T-1} J(\mathbf{s}_k, \mathbf{a}_k) + J(\mathbf{s}_{t+T}) \right) \tag{6}$$

4.4. Action correction by PPO

In order to further improve tracking performance and overcome adaptation errors, we propose a correction policy π_c to fine-tune the optimal action a^* from a MPC controller. An extra corrective feedback a_c generated by π_c is incorporated and the final output action for the position controller is shown in Eq.7.

$$\mathbf{a}_t^* = \mathbf{a}_t^* + \mathbf{a}_c \tag{7}$$

We formulate this online action correction problem in the new MDP which is independent from the MDP defined in the Section 4.1. The goal is to find a policy $\mathbf{a}_c = \pi_c(\mathbf{s}_c)$ to maximise the long term cumulative reward $J(\pi_c) = E_{\pi_c}[\sum_{t=0}^{\infty} \gamma^t R(\mathbf{s}_c^t, \mathbf{a}_c^t)]$. This policy operates independently from the previous MPC policy. As shown in the block B of Fig.4, it uses different input states that focus on information specific to the MPC controller.

For the **tracking tasks without obstacles**, the state of the correction policy is defined as $\mathbf{s}_c = [O_s, O_g^t, O_g^{t-1}, O_c, O_d] \in \mathbb{R}^{24}$ which consists of current system state $O_s \in \mathbb{R}^{12}$, current goal $O_g^t \in \mathbb{R}^3$, past goal $O_g^{t-1} \in \mathbb{R}^3$, the corrected action $O_c \in \mathbb{R}^3$ and three-channel tracking error $O_d \in \mathbb{R}^3$ of the payload.

For the **tracking tasks with obstacles**, the state of the correction policy is defined as $\mathbf{s}_c = [O_s, O_g^t, O_g^{t-1}, O_{g1}^t, O_{g2}^t, O_c, O_d, O_{d1}, O_{d2}] \in \mathbb{R}^{32}$ where O_{g1}^t and O_{g2}^t represent the goals of UAV1 and UAV2, respectively, and O_{d1} and O_{d2} represent the tracking errors of the UAV1 and UAV2. Compared to the non-obstacle scenario, additional states are included in the observation to enable UAV tracking for collision detection.

The correction action is a three-dimensional feedback signal $\mathbf{a}_c = [a_c^x, a_c^y, a_c^z]$ where a_c^x, a_c^y and $a_c^z \in [-0.1, 0.1]$ metre. For non-obstacle scenarios, the reward $r \in R$ is defined as the negative Euclidean distance between the goal position of the payload and the actual payload position. For obstacle scenarios, the reward function is augmented by adding penalties for the deviations of the UAVs' positions. We choose an off-the-shelf algorithm called proximal policy gradient (PPO) [57] to train the correction policy. This algorithm is based on the actor-critic architecture and uses the importance sampling technique to utilise the previous fixed-length samples from a replay buffer

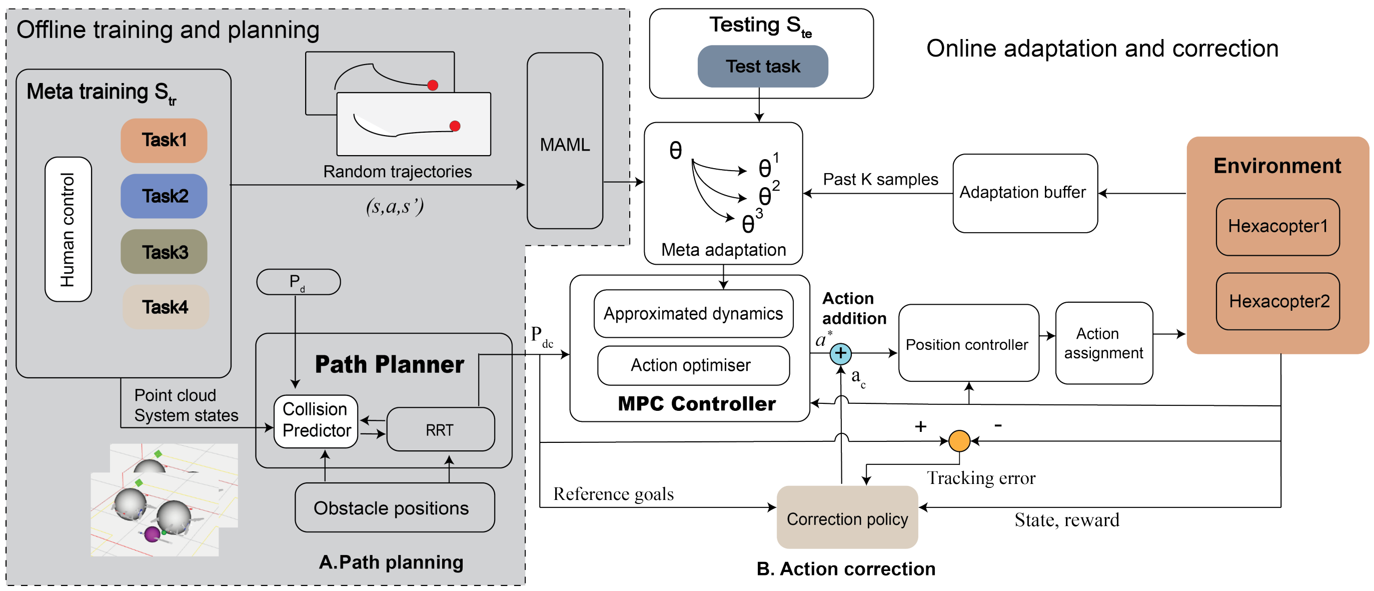


Figure 4: Overview of the proposed system with trajectory tracking and path planning. The grey area represents the offline processes that are involved in training the meta-dynamics model and the collision-predictor. The white area represents the online processes that are involved in adapting to new situations and correcting actions in real time

D_c to update the policy. For the actor neural network, the parameter θ_c of the policy is directly optimised by the clipped objective function in Eq.8.

$$J(\theta_c) = E_{(\mathbf{s}_c^t,\mathbf{a}_c^t)\sim D_c}[min(r_t(\theta_c)\hat{A}_t, clip(r_t(\theta_c), 1-\epsilon, 1+\epsilon)\hat{A}_t) + cS^{\pi_{\theta_c}}(\mathbf{s}_c^t)] \tag{8}$$

Where probability ratio $r_t(\theta_c) = \frac{\pi(\mathbf{a}_c^t|\mathbf{s}_c^t)}{\pi_{\theta_{old}}(\mathbf{a}_c^t|\mathbf{s}_c^t)}$, ϵ is the hyperparameter, $S^{\pi_{\theta_c}}(\mathbf{s}_c^t)$ is the entropy of current policy and c is the corresponding coefficient. In this work, a generalised advantage estimation (GAE) A_t in Eq.9 is used to represent the reward level.

$$A_t = \delta_t + (\gamma\lambda)\delta_{t+1} + ... + (\gamma\lambda)^{T-1}\delta_{t+T-1} = \sum_{n=0}^{T-1}(\gamma\lambda)^n\delta_{t+n} \tag{9}$$

Where the temporal difference error δ is computed by $\delta_t = \gamma V(s_{t+1}) + r_t - V(s_t)$. For the critic neural network, the value function is optimised by the mean squared error loss $L_v = MSE(V_t^{target} - V(s_t))$. The actor and critic networks are composed of two hidden layers with 256 neurons and ReLU activation functions (The actor policy is shown in (2) of Fig.3). The output activation function of the actor is Tanh. During the training of correction policy, the online adaptation is executed simultaneously to generate the final action. The detailed architecture of the proposed method is shown in Fig.4.

4.5. *Safe transportation by self-awareness collision-predictor*

To avoid collisions, we use a strategy that combines offline collision-free path planning with online trajectory tracking. The online trajectory tracking controller, which is based on meta-reinforcement learning, ensures that the tracking error is small enough to meet safety constraints under different tasks. The offline planning is responsible for sampling valid collision-free trajectories for the system, which are checked by the proposed

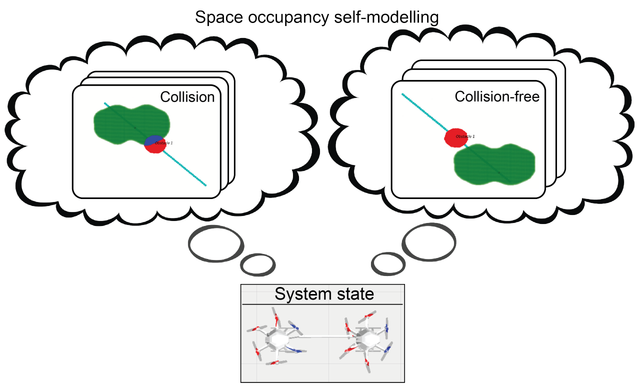
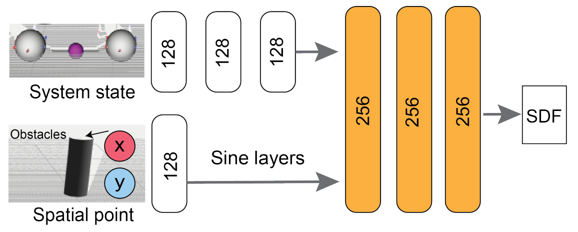
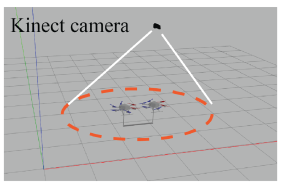


Figure 5: The collision predictor actively predicts the space occupancy of the UAV-payload system from its current state to determine whether the corresponding system state will result in a collision. This process can be regarded as a self-modelling process. The left scenario depicts a collision scenario where we can see the penetration (blue area) of an obstacle (red area) into the predicted system body (green area). The right scenario depicts a collision-free scenario.

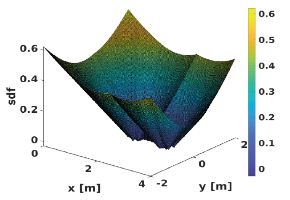
collision predictor. Moreover, the collision checking for each goal state (trajectory) should reveal the true safety constraint before executing any trajectories in the physical world. As shown in Fig.5, given the state of a UAV-payload system, the collision predictor ϕ can predict the space occupancy of the system, specifically the signed distance function (SDF) of the system. We define the body of the dual-UAV-payload system in the $2d$ $x-y$ plane. The 2D signed distance function (SDF) value $sd \in \mathbb{R}$ represents the shortest distance from a spatial point $x \in \mathbb{R}^2$ to the body of the system. If any spatial points x are found to be on or inside the surface of the body of the system, their SDF value is equal to or less than 0. If the spatial point x is outside the boundary of the body, its SDF value is larger than 0. Therefore, the SDF value can be used to determine the collision of a spatial point. Theoretically, we try to



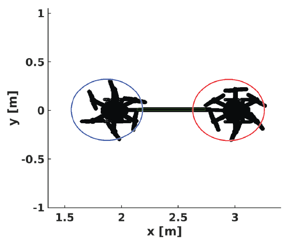
(a) The structure of collision predictor



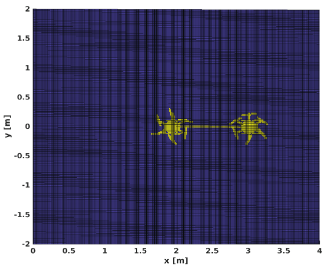
(b) Point cloud collection



(e) signed distance function



(c) Point cloud



(d) Occupancy map

Figure 6: Overview of the collision predictor

approximate the mapping from a 2D location to its corresponding SDF value $\hat{sd} = \phi(x)$.

4.5.1. *Training of the meta-collision predictor*

In order to train the predictor, we build a dataset $\boldsymbol{X} := [(x, \mathbf{z}), sd]$ where x is the 2D spatial points over the workspace, $\mathbf{z}$ is the current state of the UAV-payload system and sd is the ground truth SDF value. The additional variable $\mathbf{z}$ captures the change in space occupancy when the system state transitions in different tasks. As $\mathbf{z}$ is a code vector, training of the predictor can be regarded as meta-learning based on the conditional vector. We used a top-down Kinect camera to collect point cloud data (spatial points) of the workspace as well as the corresponding system state. The data collection is performed on different tasks while the UAV-payload system follows random trajectories controlled by a human operator. During data collection (as shown in (b) of Fig.6), the camera is centred at the base position of the payload. We extract the spatial points x_{pc} of the body of the system and transform them into the inertial coordinate (see (c) of Fig.6).

In order to obtain the corresponding label sd, we first merge all the collected spatial points x_{pc} into a 401×401 2D point grid M_{sp} within the range of the workspace by a nearest-neighbour tree search technique. Then, points from the collection called on-surface points are labelled with the occupancy value of 1 and the rest points in the grid are called off-surface points with the occupancy value of 0 ((d) of Fig.6). The last step is to calculate the shortest normalised pixel distance (SDF) to the surface of the on-surface points for each point in the grid ((e) of Fig.6).

The prediction function ϕ of SDF on all spatial points in M_{sp} is approximated by a SIREN neural network [50] $\hat{sd} = \phi_\theta(x|\mathbf{z})$ and we minimise the mean square error by $L_{sdf} = MSE(sd, \hat{sd})$ in the supervised learning. We choose $z = [p_l^{xy}, p_{u1}^{xy}, p_{u2}^{xy}]$ where $p_l^{xy} \in \mathbb{R}^2$, $p_{u1}^{xy} \in \mathbb{R}^2$ and $p_{u2}^{xy} \in \mathbb{R}^2$ are the 2d xy positions of the payload, UAV1 and UAV2 respectively (because of the 2D collision constraints). In the training, we randomly sample 1500 data points from all on-surface points and all off-surface points respectively. The architecture of the predictor is shown in (a) of Fig.6. As we can see, the spatial points are fed into one hidden sine layer with 128 neurons, and the state vector is processed by three sine hidden layers with 128 neurons. Finally, these two outputs are concatenated followed by three sine hidden layers with 256 neurons.

4.5.2. *Collision-free path sampling*

A rapidly-exploring random tree (RRT) is utilised to sample the collision-free goal states of the system guided by the trained collision predictor. For offline path planning, as only the payload trajectory P_d is given, we recover the code vector $\mathbf{z}$ from each trajectory of P_d. We conducted empirical experiments in order to calculate the feasible distance between the payload and two UAVs in the x and y axes of the inertial coordinate under different testing tasks. This is done to ensure the hovering stability of the system. By querying the spatial points P_o from the known obstacles, the collision predictor scans each code vector $\mathbf{z}$ of the payload trajectory P_d. For each code vector $\mathbf{z}$, a SDF grid occ over all the obstacle points P_o is formed. If the predicted SDF value σ in the grid is larger than the distance threshold α, its corresponding occupancy is set to 0. Otherwise, the occupancy is 1. The determination of an unsafe payload trajectory point is made if the sum of occupancy of occ is larger than the collision threshold τ. After the scan, we find the unsafe payload trajectory segments $l_{j=1,...,n}$ which will lead to collisions. Then, the RRT method is employed to plan a collision-free trajectory l_{c_j} based on the starting point p_s and ending point p_e of $l_{j=1,...,n}$ in the 2D $x - y$ plane, while keeping the position in the z axis of the inertial coordinate constant. The newly sampled point of the payload will also be recovered to the code vector $\mathbf{z}$ and the collision detection process is similar to the scanning process. The sampled trajectories of the payload will be saved unless it is detected as a collision-free point by the predictor ϕ. The corresponding 2D positions of UAVs in code vector $\mathbf{z}$ will also be saved for deriving the full collision-free trajectory P_{dc}. The whole process of the collision-free path planning is shown in Algorithm 1. The derived P_{dc} is executed by the proposed trajectory tracking controller to achieve safe transportation under different tasks.

5. Experiments and Results

5.1. *Task description*

The simulation environment is built in the *Gazebo* simulator with the reinforcement learning toolbox *openai_ros* [58] and robot operating system (ROS). The payload transportation task with two hexacopters is carried out in a 4×4 metre workspace

Algorithm 1 Collision-free transportation by path planning with collision predictor

```
Input: The global payload trajectory $P_d$
Output: The collision-free full trajectory $P_{dc}$
Require: Obstacles knowledge $O_{1:m}$
Require: Exploitation coefficient $\epsilon$, step size $\delta d$
Require: Collision threshold $\tau$, distance threshold $\alpha$
 1: Check feasibility of each point $P_d^i$
 2: for Point number $i = 1, ..., N$ do
 3:     Get desirable system state $z_i$ from $P_d^i$
 4:     Sample spatial points $P_o$ from $O_{1:m}$
 5:     for Each obstacle point $j = 1, ..., n$ do
 6:         $\sigma_j = \phi(\mathbf{z}_i, P_o^j)$
 7:         if $\sigma_j > \alpha$ then
 8:             $occ(j) = 0$
 9:         else
10:             $occ(j) = 1$
11:         end if
12:     end for
13:     if $sum(occ) > \tau$ then
14:         label $P_d^i$ as a unsafe point
15:     else
16:         label $P_d^i$ as a safe point
17:     end if
18: end for
19: Re-plan $P_d$ for corrective trajectory $P_{dc}$
20: for unsafe trajectory segments $l_{j=1,...,n}$ do
21:     Find start point $p_s$ and endpoint $p_e$
22:     Calculate $lc_j = RRT(p_s, p_e, \epsilon, \delta d, \phi)$
23: end for
24: replace all segments $l$ in $P_d$ by $lc$ to get $P_{dc}$
```

with a maximum height of 2 metres. A bar payload weighing 0.2 kg is connected to the centre mass of two UAVs by two 0.3-metre massless cables. The dimensions of the payload is $0.6 \times 0.02 \times 0.02$ metres. A generated constant wind force is only applied to all simulated objects in the x axis of the inertial coordinate. As shown in Tab.2, we design four training tasks with different combinations of wind force and neighbour distance.

Table 2: Task configurations

Training tasks	1	2	3	4
Wind force [N] (v_f)	0.0	0.3	0.5	0.8
Neighbour distance [m] (v_d)	0.6	1.0	0.8	1.2
Testing tasks	1	2	3	
Wind force [N] (v_f)	0.0	1.0	0.6	
Neighbour distance [m] (v_d)	0.6	0.8	1.4	

The testing tasks contain the tasks that have been seen in the training (Task 1) or the newly designed tasks outside the training distribution (Task 2 and Task 3), which aims to examine the adaptation ability of the proposed algorithm. The space occupancy of the system under different testing tasks is visualised in Fig.7. Under all the training tasks, the training of the meta-dynamics model and collision predictor are executed by a laptop with Nvidia 3070Ti and Pytorch. Our proposed method is validated in all testing tasks, where the UAV-payload system is required to follow the goal paths generated by the path planner in environments with or without obstacles. In an obstacle-free environment, the path planner does not process the original path. Instead, it tracks only the payload trajectory. As shown in (c) and (d) of Fig.9, the goal paths of *figure8* (in the x-y plane of the inertial coordinate) and square (in the x-z plane of the inertial coordinate) are tracked. In order to distinguish the figure reference and the goal path, we use italic font style to address that *figure8* is a goal path, not a figure reference.

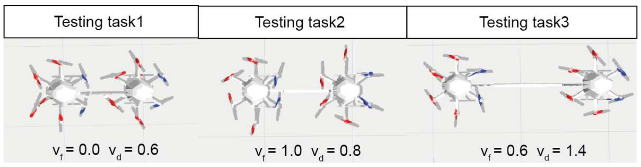


Figure 7: Space occupancy of UAV-payload system under different testing tasks

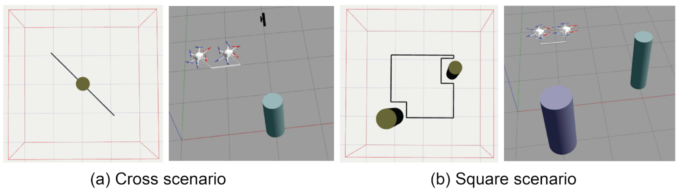


Figure 8: Two scenarios with obstacles:(a) Cross (b) Square. The predefined goal trajectories P_d are represented by black lines.

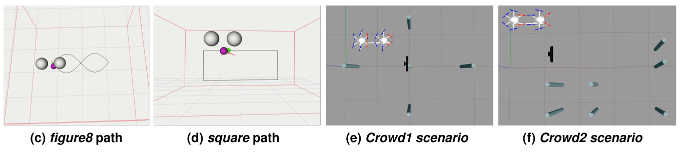


Figure 9: Two scenarios without obstacles:(c) *figure8* (d) Square (without obstacles). Two crowded scenarios:(e) Crowd1 scenario (f) Crowd2 scenario

When obstacles exist, the UAV-payload system needs to track the collision-free trajectories of UAVs and the payload generated by the path planner. As shown in Fig.8, two obstacle scenarios called cross and square are presented (the height of the goal path is kept the same to be 0.8 metres). In the square scenario, two obstacles with a radius of 0.2m and 0.3m are placed at $(3.0, 0.5)$ and $(1.0, -1.0)$ respectively. While in the cross scenario, only one obstacle with a radius of 0.2m is placed at the position $(2.0, 0.0)$. Furthermore, as shown in (e) and (f) of Fig.9, two additional challenging scenarios involving more than two obstacles are tested. The waypoints of payload paths are spaced every 0.02 metre evenly and moved forward every time step after the start of a test. The UAV-payload system starts at the same location in each path, and the test is terminated when the last waypoint is reached or if the system fails, such as by moving out of the workspace or experiencing a catastrophic crash.

5.2. *Results of meta-dynamics model*

The training data for offline meta-learning of the dynamics model consists of the states and actions of the system, which

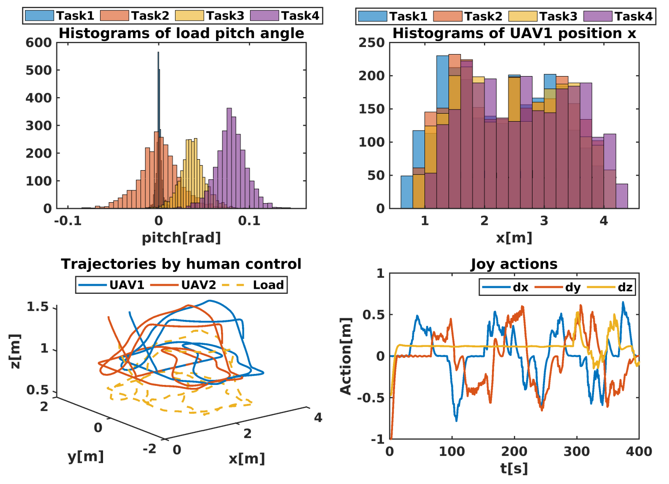
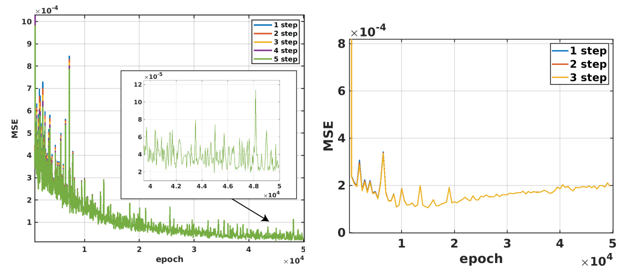


Figure 10: Visualisation of the training data for meta-learning: the upper two histograms are the pitch angle of the payload and the position *x* of UAV1. The bottom two graphs are part of the recorded trajectories and joy actions from training Task 1



(a) (b)

Figure 11: Offline training of the dynamics model using MAML: (a) the curve of training loss (b) the curve of validation loss

were recorded from human demonstrations. In order to understand the distribution of training data, we visualise the human demonstrations from different training tasks in Fig.10. It shows that the payload's pitch angle in all training tasks tends to form a normal distribution. Due to the effects of wind gusts and formation change, the mean of each distribution is shifted. Tasks with larger wind forces tend to have larger mean values of the pitch angle (Tasks 3 and 4 have larger means than that of Tasks 1 and 2). The mean of Task 2 is almost the same as that of Task 1, but Task 2 has a larger standard deviation, revealing the influence of neighbour distance. Meanwhile, the mean position of UAV1 is concentrated at a higher value in the task with a larger wind force because the wind is applied along the positive *x* axis of the inertial coordinate. At the bottom of Fig.10, the part of the recorded trajectories of the UAVs and the payload and joystick action signals are demonstrated respectively. After conducting an analysis of the data, we trained our meta-dynamics model. This model was adapted to various testing tasks with varying distributions of state or state transitions.

We used MAML [34] to train the offline meta-dynamics model. This is because we found that its adaptation speed is fast enough for real-time online reinforcement learning. In our study, we defined the problem in the context of 4-way-30-shot few-shot learning. We selected four distinct types of tasks and set the number of context samples K and query samples Q to 15. The Adam optimiser was utilised with the 0.001 learning rate for the inner loop (Eq.(2)) and outer loop (Eq.(3)). During each iteration of the outer loop, we performed batch training on four training tasks, where the batch size was 32. The total number of training epochs was 50,000, and we recorded the mean of the training loss of the batch data using the mean square error (MSE) every 50 epochs. The loss was defined as the state change, as mentioned in Section 4.2. The validation data was validated every 500 epochs. We implemented the gradient adaptation 5 times for the training tasks and 3 times for the validation data. As shown in Fig.11, training loss was dramatically decreased to less than 4×10^{-5}. Although the model's performance degraded in the validation tasks, the MSE could drop to less than 2×10^{-4}. We did not observe any superior performance with a different number of gradient steps. Therefore, in the online adaptation part, we updated the context data by five gradient steps in order to obtain the adapted parameters for the dynamics model.

Table 3: Mean squared error (MSE) of the offline model in the testing dataset

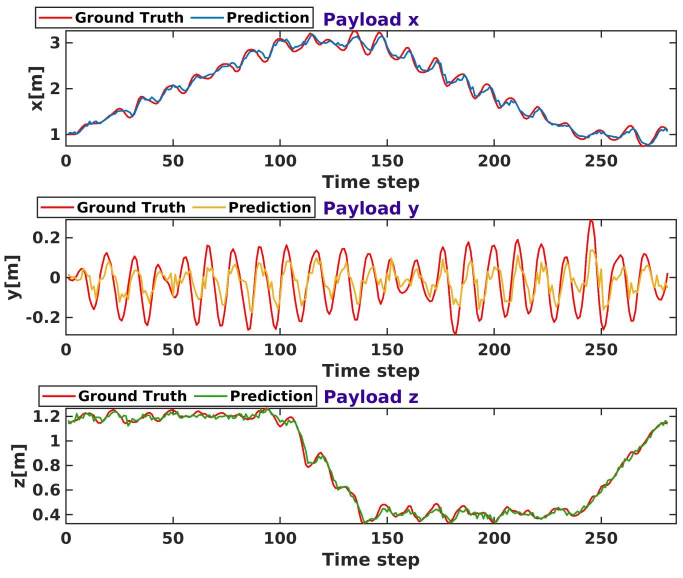
	Task 1	Task 2	Task 3
Mean	5.47e-05	2.47e-04	3.28e-04
Std	4.45e-05	1.04e-04	1.37e-04

Moreover, we tested the trained meta-model in all the testing tasks. The result was calculated by the mean accuracy of 10 batch data from the samples of different testing tasks (see Tab.3). In Tab.3, the MSE in the out-of-distribution tasks (task 2 and task 3) is less than 4×10^{-4}. This result shows that our meta-dynamics model can effectively adapt to the data from testing tasks, even when the training data differs from the samples encountered during online deployment in the testing tasks. To further validate our proposed algorithm, we implemented online adaptation followed by the MPC controller in testing Task 1. This involves tracking the square and *figure8* paths. During one episode of the trajectory tracking task, we recorded the ground truth position of the payload and the predicted positions of the payload using the adapted meta-dynamics model. We then plotted the 3D positions in Fig.12. The results indicated that the predicted positions match the real positions with small errors in three axes. However, the predicted y position of the payload in the square path exhibits relatively large fluctuations.

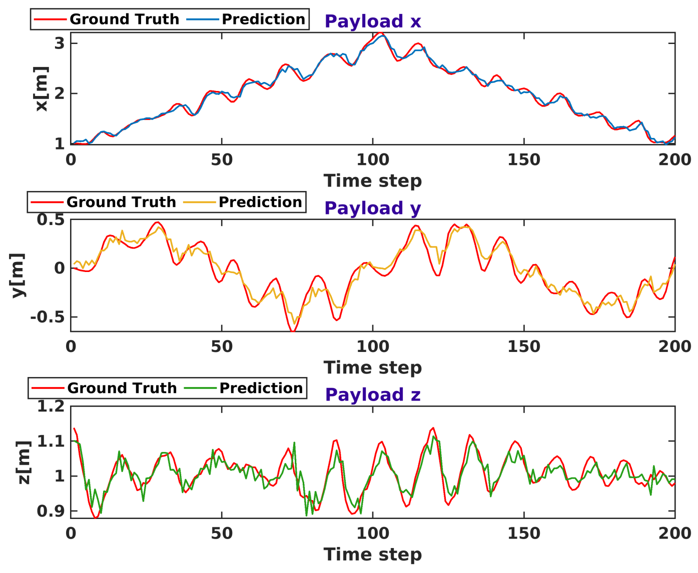
5.3. *Performance analysis in non-obstacle environment*

5.3.1. *Tracking performance*

In the online adaptation, the past 15 samples are extracted from the adaptation buffer D_a to adapt the offline meta-model at each time step. Given the real-time constraints, we set the prediction horizon T of the MPC to 5. In addition, we used a total population M of 1000 sampled action sequences in the MPC, with 10 elite population members for optimal action optimisation. We tested our algorithm in two obstacle-free paths *figure8* and square of the payload. To prove the novel tracking performance of our proposed algorithm, we compared it with the proximal policy optimisation (**PPO**), probabilistic ensembles with trajectory sampling (**PETS**) [59] as well as fast adaptation through meta-learning embeddings (**FAMLE**). These baselines are state-of-the-art model-free, model-based and meta-learning



(a) Task1 and square path



(b) Task1 and *figure8* path

Figure 12: Online prediction of the payload by the adapted meta dynamics model in scenario (a) and scenario (b)

Table 4: Comparison of tracking error [metres] of the payload in *figure8* and Square paths

Path	*figure8*					Square				
Task	PETS	FAMLE	PPO	MAML	Proposed	PETS	FAMLE	PPO	MAML	Proposed
1	0.23±0.0168	0.17±0.0124	0.16±0.0030	0.18±0.0091	**0.09±0.0060**	0.23±0.0147	0.18±0.0158	0.18±0.0035	0.18±0.0059	**0.08±0.0064**
2	0.21±0.0088	0.23±0.0953	0.28±0.0009	0.19±0.0257	**0.14±0.0063**	0.29±0.1077	0.22±0.0776	0.37±0.0012	0.17±0.0086	**0.16±0.0078**
3	0.19±0.0068	0.12±0.0078	0.12±0.0029	0.14±0.0085	**0.07±0.0036**	0.29±0.0649	0.12±0.0179	**0.07±0.0009**	0.12±0.0075	0.08±0.0121

methods. We removed the correction policy from the proposed algorithm and retained only the online meta-adaptation part by **MAML** in order to evaluate whether the performance degraded. For each scenario, we conducted one episode of testing and measured the **average tracking error**. This error is calculated as the mean Euclidean distance between the centre of mass of the payload and the reference goal over all time steps. All results in Tab.4 are computed using data from 10 repeated experiments. The numbers in red indicate that the task is partly finished, while the bold numbers represent the best performance in each scenario.

In the *figure8* path, the proposed method outperforms other methods in all testing tasks, exhibiting the smallest error and variance. The average tracking error can be less than 0.08 meters, and the performance improved by up to 60% compared to the worst method (PETS). The performance of meta-learning methods MAML and FAMLE is almost identical in all tasks, except for Task 2, where the tracking error of FAMLE is larger than that of MAML. In most cases, meta-learning methods are superior to model-free methods. The model-based method PETS has the largest tracking error in all testing tasks, performing even worse than the model-free PPO method. This suggests that the online update of the dynamics model makes a limited contribution to the tracking performance. Most methods perform worse in Task 2, which has the largest wind force compared to other tasks. As meta-learning methods depend on the prior knowledge of the meta-model, the quality of the training data from Task 2 may affect its performance in online testing.

In the square path, the proposed method also outperforms other methods. The performance of FAMLE degrades in Task 2, while the PPO method performs best in Task 3, indicating that the model-free method is less stable in adaptive tasks compared to the proposed method. We observed that the tracking error of our proposed method did not significantly decrease in Task 2. Compared to the *figure8* path, the square path's performance is slightly worse. Fig.13 shows the mean actual payload trajectories for all methods by 10 repeated experiments. The path is divided into several areas by placing three and four nodes in the *figure8* and square paths, respectively. The sequence of the path is A-B-C-B-A for the *figure8* and A-B-C-D-A for the square. In some situations, the entire path was not completed, resulting in only partial plotting (such as FAMLE with Task 1 and Task 2 in the square path). Overall, our action correction policy in our proposed method further decreases the tracking error based on online adaptation by MAML in most testing scenarios.

5.3.2. *Analysis of correction policy*

In order to better understand the correction action in the proposed method, we first compared MAML (the online adaptation method only) with our proposed online adaptation and correction method in Fig.14. We implemented the experiments in testing Task 2 with the *figure8* path and testing Task 3 with the square path, respectively. For each path, we plot the actions, which are defined as the 3D position displacements of the virtual leader by these two methods over one testing trial, and show the corresponding tracking error curve. Moreover, at

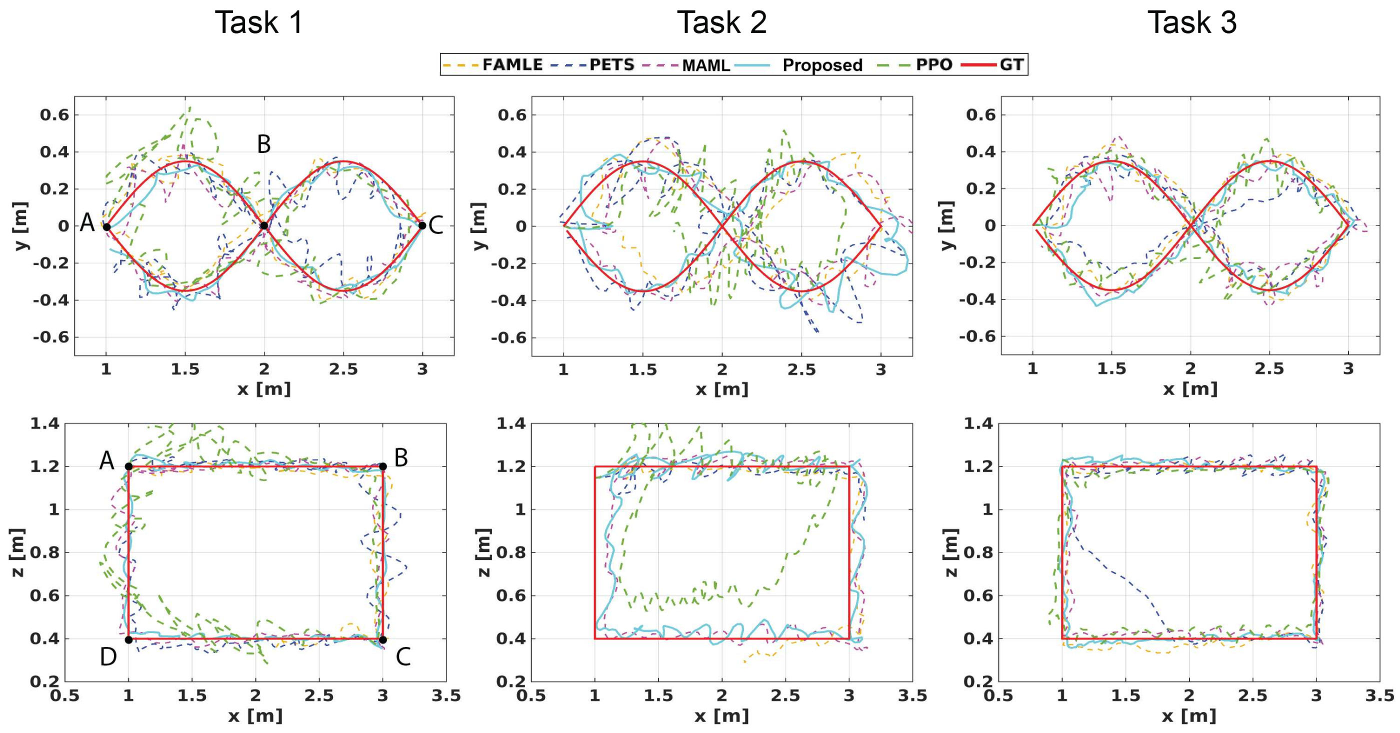


Figure 13: Visualisation of the mean actual trajectories of the payload by all methods in all scenarios.

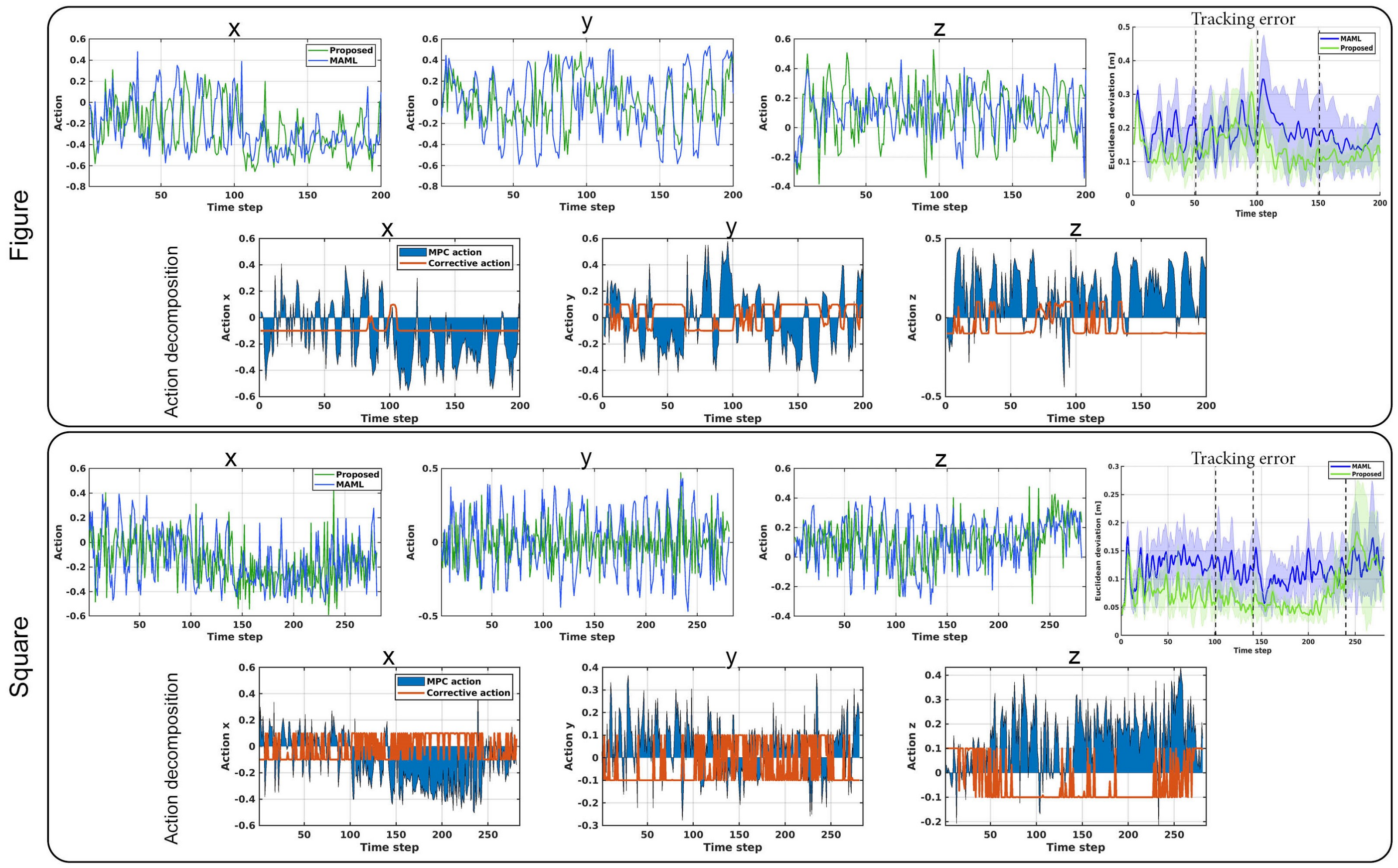


Figure 14: The comparison of actions between MAML and the proposed method. In the proposed online adaptation and correction, we decompose the action into the MPC action and the corrective action and visualise them in the second row of each block

the bottom of each block, we decompose the action a of the proposed method into the action a^* (the blue curve) from MPC and the correction action a_c (the red curve), where $a = a^* + a_c$, and visualise them. In the first scenario, we can observe the distinct actions of the two compared methods. The incorporation of the correction policy can further decrease the tracking error after the process of online adaptation. A negative value of the correction action is applied to the action x as the wind pushes the payload in the positive x direction. The action z is decreased by the correction action compared to the original output of MPC. The reason may be that the increasing wind force raises the height of the payload slightly. As for the second scenario, the correction actions fluctuate significantly compared to the first scenario. Due to the larger neighbour distance in the testing Task 3, the stability of the payload-UAV system changes. Therefore, the correction action needs to be adjusted frequently to ensure tracking accuracy. Overall, the action correction policy can improve the performance of the tracking tasks by correcting the action from the MPC controller in time using the observation of the system.

5.3.3. *Performance in time-varying wind conditions*

In order to emulate real-world unsteady wind conditions in the simulator, we set the neighbour distance to 0.6m and vary the wind force using a sinusoidal function. The amplitude of the sinusoidal function was set to 0.2, 0.5, and 0.8, respectively (different functions are shown in Fig.15). We tested the setup on a *figure8* path without obstacles. The experiments are repeated for five times. The mean Euclidean error (the Euclidean distance between the payload's center of mass and its goal) during the tasks is shown in Fig.16. The average tracking error of one test iteration is shown in Tab.5.

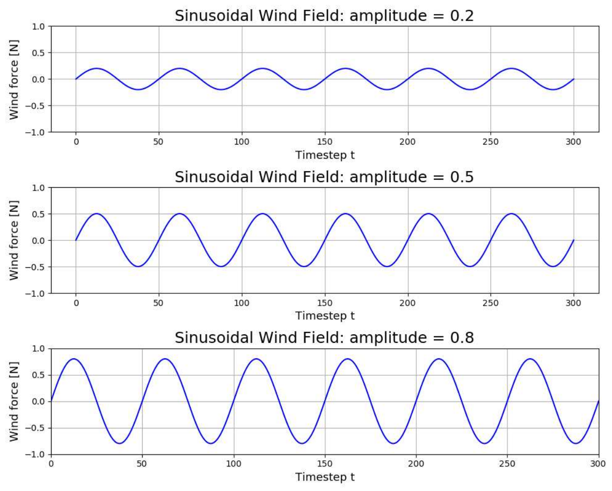


Figure 15: Sinusodal wind field with different amplitudes

Table 5: Average tracking error in one test iteration

Experiment	Amplitude 0.2	Amplitude 0.5	Amplitude 0.8
Mean Euclidean error [m]	0.13	0.17	0.32

Overally, in the simulation, our proposed model trained in the static uniform wind force can also achieve good performance (with less than 0.18 metre) in the time-varying wind field

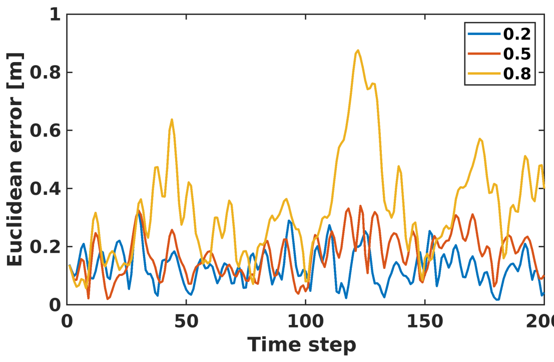


Figure 16: Mean Euclidean tracking error of the payload in the sinusodal wind fields

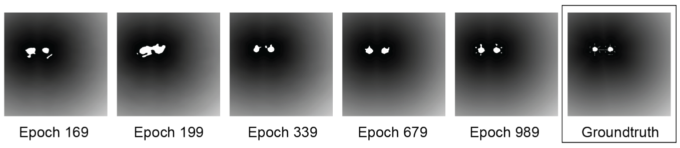


Figure 17: The change of the occupancy space of a validation sample during the training process

when the amplitude is not very large, which further prove the potential of our model to be deployed in the real-world with fast changing wind conditions. However, we still need to improve the robustness of our algorithm in future work to achieve better performance in the real-world.

5.4. *Performance analysis of collision-free transportation*

5.4.1. *Results of meta-collision predictor*

The collision predictor ϕ is trained for 1000 epochs using the Adam optimiser with a learning rate of 0.00005. The training data and validation data from the four training tasks are split in a 9:1 ratio. The batch size of the training is 55. During the training, we visualise the occupancy space of a validation sample using the collision predictor to demonstrate the model's improvement compared to the ground truth (see Fig.17). As the number of training epochs increases, we can observe that the prediction of the space occupancy of the UAV-payload system becomes more accurate.

In order to validate the performance of the collision predictor, we test the derived model on the dataset from the three testing tasks. The mean squared error (MSE) between the predicted SDF and the real SDF is 7.28×10^{-5} for Task 1, 1.20×10^{-3} for Task 2, and 2.19×10^{-4} for Task 3. For visualisation, we input a code vector (system state) into the predictor and obtain the predicted SDF over the grid M_{sp} (the 2D workspace) by querying all the spatial points. Results of visualisation are shown in Fig.18. The space occupancy of the system is constructed by the prediction. The determination of the on-surface points follows the Algorithm 1. We tuned the distance threshold α to obtain an inflated space occupancy that accounts for potential tracking deviations from the trajectory tracking controller when checking for collisions with obstacles. The results in Fig.18 are evaluated on random samples from the testing tasks, which demonstrate that the collision predictor has acceptable precision and good performance in adapting to different tasks.

Table 6: Statistics performance of the proposed path planner compared with some baselines

Scenario	Cross obstacle								
	Testing task 1			Testing task 2			Testing task 3		
Metrics	RRT-circle	RRT-point	**Proposed**	RRT-circle	RRT-point	**Proposed**	RRT-circle	RRT-point	**Proposed**
Success rate	100%	0%	100%	60%	0%	80%	0%	0%	100%
Detour distance[m]	11.28±0.63	8.97±0.79	12.69±0.57	11.28 ±0.63	8.97±0.79	13.94±0.97	11.28 ±0.63	8.97±0.79	15.09±0.22
CPU time[s]	38	14	125	38	14	220	38	14	168
Scenario	Square obstacle								
	Testing task 1			Testing task 2			Testing task 3		
Metrics	RRT-circle	RRT-point	**Proposed**	RRT-circle	RRT-point	**Proposed**	RRT-circle	RRT-point	**Proposed**
Success rate	60%	0%	100%	0%	0%	100%	0%	0%	100%
Detour distance[m]	17.61±1.26	18.45±0.11	24.33±0.58	17.61±1.26	18.45±0.11	23.29±1.64	17.61±1.26	18.45±0.11	21.96±1.20
CPU time[s]	39	5	155	39	5	181	39	5	255

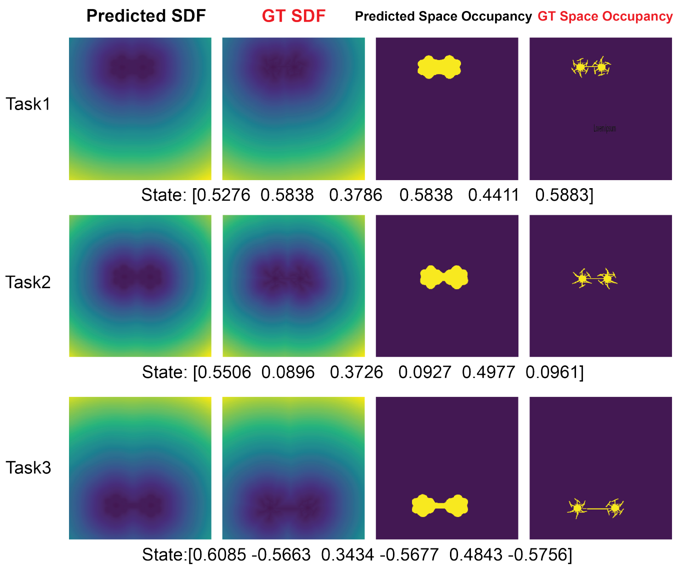


Figure 18: Visualisation of the collision predictor by random samples

5.4.2. *Results of collision-free transportation*

Following the details of the Algorithm 1, we use the trained collision predictor to derive the collision-free trajectories of the UAVs and the payload. As shown in Fig.19, we identify the unsafe segments (red lines) from original paths P_d and re-plan the collision-free paths by RRT and collision predictor in the considered square and cross scenarios. As for the parameters of RRT, exploitation coefficient ϵ and step size δd are set to 0.5 and 0.04 respectively. The collision threshold τ and distance threshold α are adjusted for different scenarios.

The final obtained collision-free paths P_{dc} are shown in the first row of Fig.20. To validate the performance of the sampling-based path planner, we sampled five paths for each task and scenario. In order to provide clarity, we only plot the payload trajectories to demonstrate the differences in the planned paths across different tasks. The derived collision-free paths are influenced by the predicted space occupancy of the UAV-payload system. In Task 1, where there is no wind disturbance and the neighbour distance is small, the space occupied by the system is also small. However, in Task 3, the space occupancy of the system increases due to the increased neighbour distance and wind force. As a result, the sampled paths of Tasks 2 and 3 have a longer distance to the obstacle than that of Task 1 because the system with an increased occupied area requires a detour to avoid the obstacles.

The proposed tracking method tracks the sampled trajectories from the path planner in the cross and square scenarios with static obstacles. The offline meta-dynamics model defined and trained in the last section is reused here. We jointly trained the correction policy with random trajectories across all three testing tasks. We recorded the real trajectories of the UAVs and the payload while running the tracking tasks in each scenario. From the five sampled paths, we chose one path and visualised the real trajectories and goals in the second row of Fig.20. Additionally, we recorded the real height of the payload with a goal height of 0.8 meters in the third row of Fig.20. Although there are fluctuations, the tracking of the goals appears to be within acceptable limits.

As illustrated in Fig.21, our proposed method's average tracking errors of the payload are no more than 0.15 meters. Furthermore, the error of the UAVs is less than the payload and less than 0.09 meters. Tab.6 shows the statistical results of the above validations. We compared our proposed path planning with two non-adaptive baselines to demonstrate the adaptation capability of the proposed method. The RRT-circle method samples the paths with a fixed circular collision constraint with a 0.5-meter radius (the centre of the circle is the mass centre of

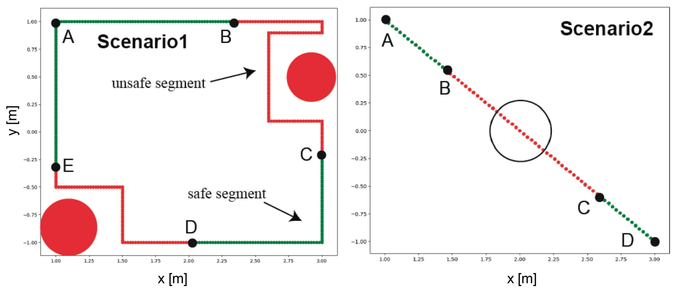


Figure 19: The collision detection of the provided two global paths. Safe segments (A-B, C-D, E-A for scenario 1 and A-B, C-D for scenario 2) and unsafe segments (B-C, D-E for scenario 1 and B-C for scenario 2)

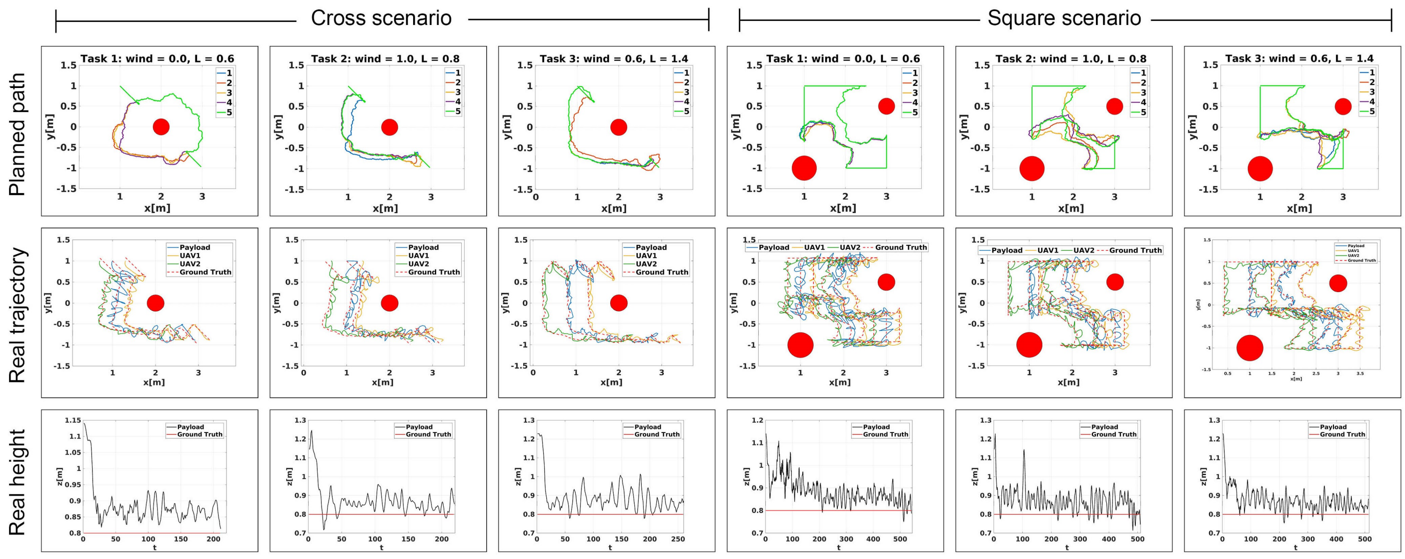


Figure 20: Visualisation of the planned paths and the corresponding real trajectories and height of the system. **First row**: planned collision-free paths; **Second row**: real trajectories of the system; **Third row**: real height of the payload

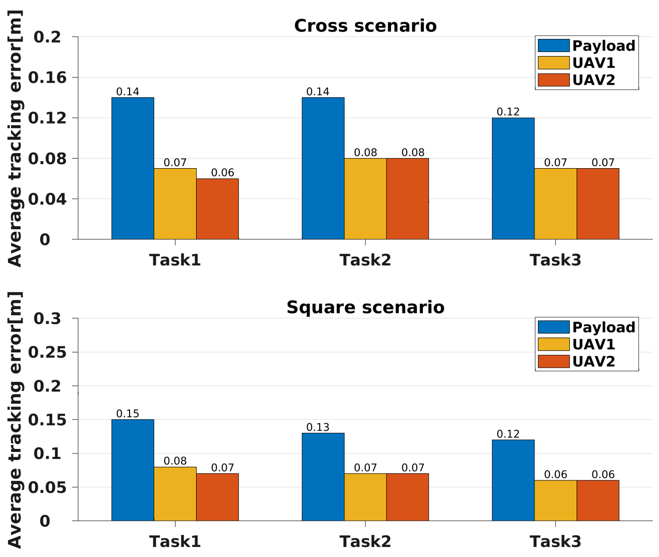
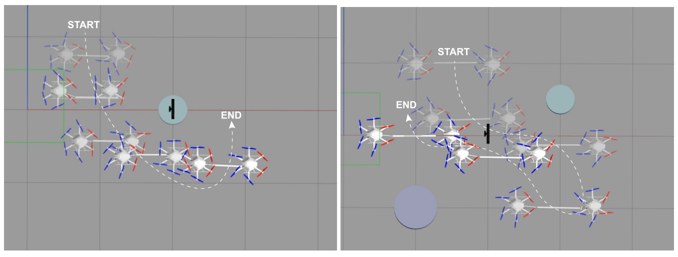


Figure 21: Tracking errors of the UAVs and payload

the payload). For the RRT-point method, we ignore the body constraints of the system and assume it is a point mass. We recorded the **success rate** (whether the system collides with obstacles or not), **detour distance** (the travelling distance minus the real distance of the planned path), and **CPU time** (computation time for one collision-free path). All the results are calculated from five repeated sampled paths.

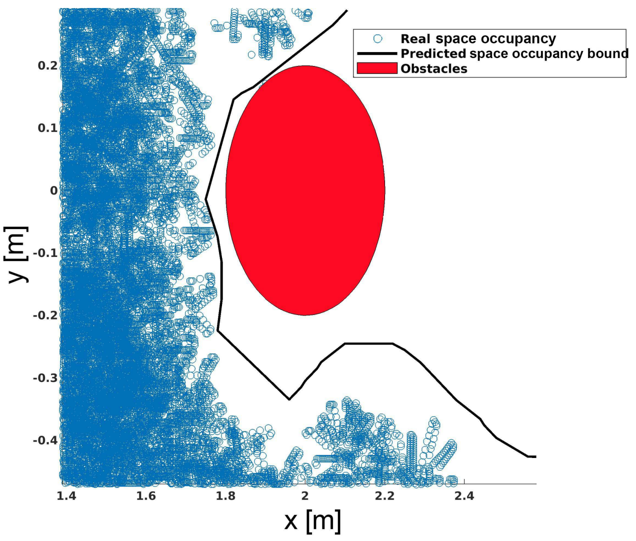
Overall, the proposed method has a success rate of almost 100% in all scenarios. The RRT-circle method has good performance in Task 1 or Task 2. However, it becomes invalid in Task 3 with a 0% success rate. The reason is that the collision constraints by a fixed body cannot satisfy the new constraints resulting from variations such as larger neighbour distances. The RRT-point method fails in all scenarios, which highlights the importance of the correct constraints of the robot's space occupancy. Although the proposed path planner takes a longer CPU time to compute the collision-free path, its success rate is higher than the RRT-circle or RRT-point, which cannot estimate the correct space occupancy of the system when detecting the collision. The proposed planner can adjust the collision constraints by the meta-collision predictor across different tasks and choose the proper detour to avoid the obstacles. Fig.22 shows screenshots of some tasks where the proposed path is tracked by the proposed adaptive controller to achieve the collision-free transportation.



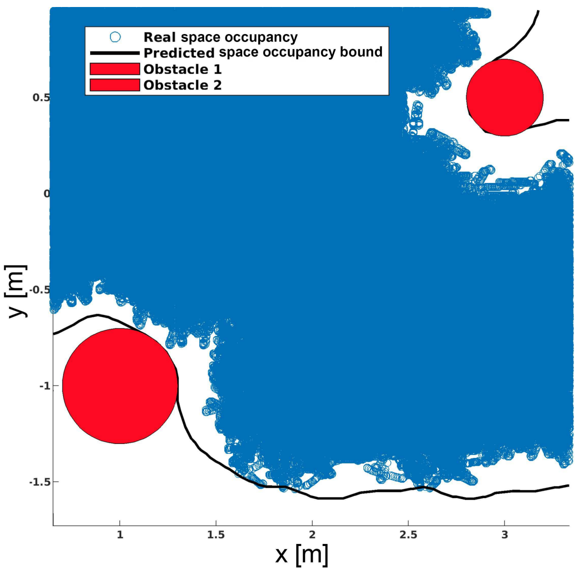
(a) Cross scenario (b) Square scenario

Figure 22: Screenshot of the collision-free transportation

In Fig.23, we visualise the predicted space occupancy based on the sampled collision-free paths and the real occupancy space (blue points) of the system by the top-down Kinect camera during the tasks. The boundary (black lines) of the predicted occupancy space by the collision predictor is plotted. We can observe that our controller can track the goal states with small tracking errors and the collision predictor has a good prediction accuracy. As a result, the difference between the predicted and real space occupancy of the system is negligible. The black boundary of the predicted occupancy space perfectly detours the obstacle. Since it is the inflated space occupancy, the spatial points of the real space occupancy does not exceed that boundary. The results indicate that the combination of the trajec-



(a)



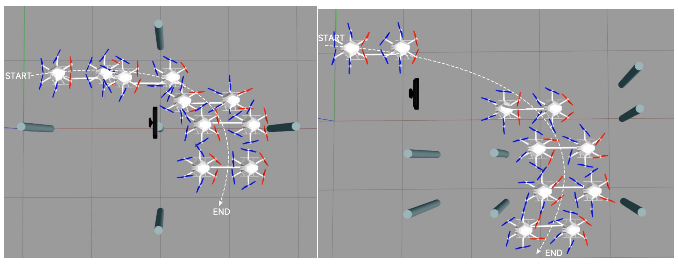
(b)

Figure 23: Visualisation of the predicted space occupancy (only the boundary of the spatial points is shown in the black line) and the real space occupancy (blue spatial points collected by the top Kinect camera) during the tracking tasks in the (a) cross and (b) square scenarios

tory tracking controller and the offline path planner can achieve good online performance without online interactions with obstacles.

5.4.3. *Transportation in crowded scenarios*

To further validate the performance of our proposed algorithm, we conduct experiments in scenarios with more than two obstacles. To adapt to crowded scenarios, we maintain the neighbor distance at 0.6m to test the proposed algorithm under different wind gusts (following the original task setup). Screenshots of the two crowded scenarios are shown in Fig.24. The first scenario contains five obstacles, and the second scenario contains seven obstacles. We can see that although the number of obstacles increases, the collision-free transportation task can still be achieved using our proposed algorithm. The average tracking errors of different scenarios are shown in Fig.25. In the Crowd1 scenario, the success rates for task 1,2,3 are 100%, 80% and 100% respectively. In the Crowd2 scenario, the success rates for task 1,2,3 are 100%, 80% and 80% respectively.



(a) Crowd1 scenario (b) Crowd2 scenario

Figure 24: Screenshot of the transportation in crowded scenarios

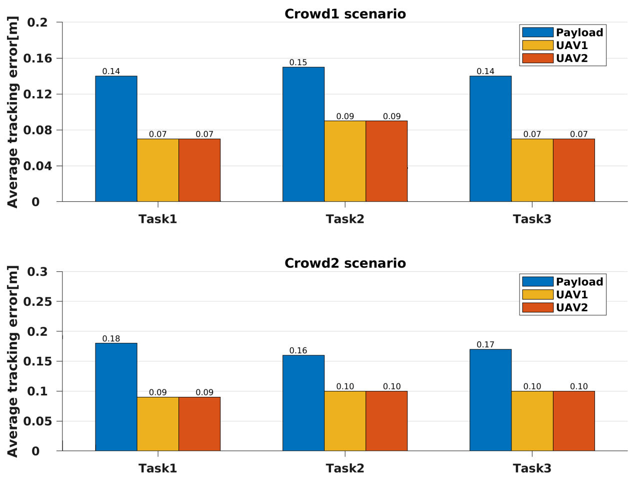


Figure 25: Tracking errors of the UAVs and payload in crowded scenarios

6. Conclusion

In this work, we propose an adaptive method based on meta-model-based reinforcement learning for payload trajectory tracking by two UAVs subject to variations during transportation. A novel correction policy is utilised to fine-tune the actions from the adapted meta-dynamics model in an online fashion to improve tracking performance. The superior performance of our proposed method enables the dual-UAV system to follow the predefined trajectories with less than 0.1 meters deviation and variance in all tested scenarios. Compared with the meta-learning method without action correction, the tracking error of the payload by the proposed method has been reduced by almost 50%. Compared with other model-based and variational inference methods, the tracking error of the payload has been decreased by up to 60%. Additionally, we developed an RRT method combined with our newly designed collision predictor in order to obtain a safe collision-free path in different tasks. The results validate the feasibility of the path planner and reveal the reference paths are adjusted properly to achieve collision-free transportation. The success rate of the proposed method is more than 80% and the corresponding average tracking errors of UAVs and the payload are less than 0.1 metres and 0.15 metres respectively.

Nevertheless, while learning-based controllers have shown promise in certain aspects, they still have notable flaws. For example, they often demand a large amount of data for training, which can be time-consuming and costly to obtain. The quality of the offline meta-model heavily relies on the distribution of the training data. The algorithm struggles with testing tasks that differ significantly from the training task distribution. Moreover, the adaptation ability of our framework is hard to extend to places outside of our predefined workspace. As for the new path planning method, we believe its performance can be further improved if we can deploy the collision predictor into the MPC to achieve the prevention of the risk in an online fashion rather than offline roll-outs with RRT. Regarding the design of the virtual leader, we discover that the action space can be designed in a decentralized rather than a centralized manner. However, as pointed out in [31], the training stability is poor due to the insufficient exploration of the policy. As a result, these learning-based designs need to strike a balance between implementation simplicity, precision and computational complexity [60]. In future work, we plan to extend the number of UAVs for the swarm system and develop a better mechanism to combine the model-free and model-based methods towards decreasing the tracking deviation as well as improving the generalisation results.

References

[1] M. Gassner, T. Cieslewski, D. Scaramuzza, Dynamic collaboration without communication: Vision-based cable-suspended load transport with two quadrotors, in: Proceedings of 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2017, pp. 5196–5202.

[2] F. A. Goodarzi, T. Lee, Stabilization of a rigid body payload with multiple cooperative quadrotors, Journal of Dynamic Systems, Measurement, and Control 138 (12) (2016).

[3] K. Huang, J. Chen, J. Oyekan, Decentralised aerial swarm for adaptive and energy efficient transport of unknown loads, Swarm and Evolutionary Computation 67 (2021) 100957. `doi:https://doi.org/10.1016/j.swevo.2021.100957`.
URL `https://www.sciencedirect.com/science/article/pii/S221065022100119X`

[4] J. Chen, J. Oyekan, Behavioural swarm optimisation for stable slung-load aerial transportation, in: 2023 IEEE Congress on Evolutionary Computation (CEC), 2023, pp. 1–8. `doi:10.1109/CEC53210.2023.10254023`.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.

[6] H. Kim, M. Jordan, S. Sastry, A. Ng, Autonomous helicopter flight via reinforcement learning, Advances in Neural Information Processing Systems 16 (2003).

[7] H. X. Pham, H. M. La, D. Feil-Seifer, L. V. Nguyen, Autonomous uav navigation using reinforcement learning, arXiv preprint arXiv:1801.05086 (2018).

[8] J. Chen, R. Ma, J. Oyekan, A deep multi-agent reinforcement learning framework for autonomous aerial navigation to grasping points on loads, Robotics and Autonomous Systems 167 (2023) 104489. `doi:https://doi.org/10.1016/j.robot.2023.104489`.
URL `https://www.sciencedirect.com/science/article/pii/S0921889023001288`

[9] T. Sugimoto, M. Gouko, Acquisition of hovering by actual uav using reinforcement learning, in: Proceedings of 2016 3rd International Conference on Information Science and Control Engineering (ICISCE), IEEE, 2016, pp. 148–152.

[10] R. Polvara, S. Sharma, J. Wan, A. Manning, R. Sutton, Autonomous vehicular landings on the deck of an unmanned surface vehicle using deep reinforcement learning, Robotica 37 (11) (2019) 1867–1882.

[11] M. B. Vankadari, K. Das, C. Shinde, S. Kumar, A reinforcement learning approach for autonomous control and landing of a quadrotor, in: Proceedings of 2018 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, 2018, pp. 676–683.

[12] A. Faust, I. Palunko, P. Cruz, R. Fierro, L. Tapia, Automated aerial suspended cargo delivery through reinforcement learning, Artificial Intelligence 247 (2017) 381–398.

[13] I. Palunko, A. Faust, P. Cruz, L. Tapia, R. Fierro, A reinforcement learning approach towards autonomous suspended load manipulation using aerial robots, in: Proceedings of 2013 IEEE International Conference on Robotics and Automation, IEEE, 2013, pp. 4896–4901.

[14] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[15] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, C. Finn, Learning to adapt in dynamic, real-world environments through meta-reinforcement learning, arXiv preprint arXiv:1803.11347 (2018).

[16] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, S. Levine, Visual foresight: Model-based deep reinforcement learning for vision-based robotic control, arXiv preprint arXiv:1812.00568 (2018).

[17] P. Abbeel, A. Coates, A. Y. Ng, Autonomous helicopter aerobatics through apprenticeship learning, International Journal of Robotics Research 29 (13) (2010) 1608–1639. `doi:10.1177/0278364910371999`.

[18] S. Bansal, A. K. Akametalu, F. J. Jiang, F. Laine, C. J. Tomlin, Learning quadrotor dynamics using neural network for flight control, Proceedings of 2016 IEEE 55th Conference on Decision and Control, CDC 2016 (0931843) (2016) 4653–4660. `arXiv:1610.05863`, `doi:10.1109/CDC.2016.7798978`.

[19] N. O. Lambert, D. S. Drew, J. Yaconelli, S. Levine, R. Calandra, K. S. Pister, Low-level control of a quadrotor with deep model-based reinforcement learning, IEEE Robotics and Automation Letters 4 (4) (2019) 4224–4230.

[20] V. Spurny, M. Petrlik, V. Vonasek, M. Saska, Cooperative transport of large objects by a pair of unmanned aerial systems using sampling-based motion planning, in: Proceedings of 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), IEEE, 2019, pp. 955–962.

[21] L. Qian, H. H. Liu, Path-following control of a quadrotor uav with a cable-suspended payload under wind disturbances, IEEE Transactions on Industrial Electronics 67 (3) (2019) 2021–2029.

[22] R. A. S. Fernández, S. Dominguez, P. Campoy, L 1 adaptive control for wind gust rejection in quad-rotor uav wind turbine inspection, in: Proceedings of 2017 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, 2017, pp. 1840–1849.

[23] J. Zhi, J.-M. Lien, Learning to herd agents amongst obstacles: Training robust shepherding behaviors using deep reinforcement learning, IEEE Robotics and Automation Letters 6 (2) (2021) 4163–4168.

[24] J. Ou, X. Guo, M. Zhu, W. Lou, Autonomous quadrotor obstacle avoidance based on dueling double deep recurrent q-learning with monocular vision, Neurocomputing 441 (2021) 300–310.

[25] N. O. Lambert, D. S. Drew, J. Yaconelli, S. Levine, R. Calandra, K. S. Pister, Low-Level Control of a Quadrotor with Deep Model-Based Reinforcement Learning, IEEE Robotics and Automation Letters 4 (4) (2019) 4224–4230. `arXiv:1901.03737`, `doi:10.1109/LRA.2019.2930489`.

[26] E. Yel, N. Bezzo, A meta-learning-based trajectory tracking framework for uavs under degraded conditions, in: Proceedings of 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2021, pp. 6884–6890.

[27] S. Belkhale, R. Li, G. Kahn, R. McAllister, R. Calandra, S. Levine, Model-based meta-reinforcement learning for flight with suspended payloads, IEEE Robotics and Automation Letters 6 (2) (2021) 1471–1478.

[28] M. O'Connell, G. Shi, X. Shi, K. Azizzadenesheli, A. Anandkumar, Y. Yue, S.-J. Chung, Neural-fly enables rapid learning for agile flight in

strong winds, Science Robotics 7 (66) (2022) eabm6597.

[29] J. Wehbeh, S. Rahman, I. Sharf, Distributed model predictive control for uavs collaborative payload transport, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020, pp. 11666–11672. doi:10.1109/IROS45743.2020.9341541.

[30] N. Rao, S. Sundaram, Integrated decision control approach for cooperative safety-critical payload transport in a cluttered environment, IEEE Transactions on Aerospace and Electronic Systems 59 (6) (2023) 8800–8811. doi:10.1109/TAES.2023.3312065.

[31] B. Xu, F. Gao, C. Yu, R. Zhang, Y. Wu, Y. Wang, Omnidrones: An efficient and flexible platform for reinforcement learning in drone control, IEEE Robotics and Automation Letters 9 (3) (2024) 2838–2844.

[32] S. Kamthe, M. Deisenroth, Data-efficient reinforcement learning with probabilistic model predictive control, in: Proceedings of International Conference on Artificial Intelligence and Statistics, PMLR, 2018, pp. 1701–1710.

[33] K. Chua, R. Calandra, R. McAllister, S. Levine, Deep reinforcement learning in a handful of trials using probabilistic dynamics models, Advances in Neural Information Processing Systems 31 (2018).

[34] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: Proceedings of International Conference on Machine Learning, PMLR, 2017, pp. 1126–1135.

[35] K. Rakelly, A. Zhou, C. Finn, S. Levine, D. Quillen, Efficient off-policy meta-reinforcement learning via probabilistic context variables, in: Proceedings of International Conference on Machine Learning, PMLR, 2019, pp. 5331–5340.

[36] R. Kaushik, T. Anne, J.-B. Mouret, Fast online adaptation in robotics through meta-learning embeddings of simulated priors, in: Proceedings of 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2020, pp. 5269–5276.

[37] C. Finn, A. Rajeswaran, S. Kakade, S. Levine, Online meta-learning, in: Proceedings of International Conference on Machine Learning, PMLR, 2019, pp. 1920–1930.

[38] S. M. Richards, N. Azizan, J.-J. Slotine, M. Pavone, Adaptive-control-oriented meta-learning for nonlinear systems, arXiv preprint arXiv:2103.04490 (2021).

[39] N. Lambert, B. Amos, O. Yadan, R. Calandra, Objective mismatch in model-based reinforcement learning, arXiv preprint arXiv:2002.04523 (2020).

[40] L. Wu, C. Wang, P. Zhang, C. Wei, Deep reinforcement learning with corrective feedback for autonomous uav landing on a mobile platform, Drones 6 (9) (2022) 238.

[41] Z. Pan, D. Li, K. Yang, H. Deng, Multi-robot obstacle avoidance based on the improved artificial potential field and pid adaptive tracking control algorithm, Robotica 37 (11) (2019) 1883–1903.

[42] B. Brito, B. Floor, L. Ferranti, J. Alonso-Mora, Model predictive contouring control for collision avoidance in unstructured dynamic environments, Proceedings of IEEE Robotics and Automation Letters 4 (4) (2019) 4459–4466.

[43] J. Van Den Berg, S. J. Guy, M. Lin, D. Manocha, Reciprocal n-body collision avoidance, in: Robotics Research: The 14th International Symposium ISRR, Springer, 2011, pp. 3–19.

[44] P. E. Hart, N. J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Transactions on Systems Science and Cybernetics 4 (2) (1968) 100–107.

[45] S. LAVALLE, Rapidly-exploring random trees: a new tool for path planning, Research Report 9811 (1998).

[46] L. E. Kavraki, P. Svestka, J.-C. Latombe, M. H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Transactions on Robotics and Automation 12 (4) (1996) 566–580.

[47] J. Zhang, B. Cheung, C. Finn, S. Levine, D. Jayaraman, Cautious adaptation for reinforcement learning in safety-critical settings, in: Proceedings of International Conference on Machine Learning, PMLR, 2020, pp. 11055–11065.

[48] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, S. Levine, Uncertainty-aware reinforcement learning for collision avoidance, arXiv preprint arXiv:1702.01182 (2017).

[49] F. Hart, O. Okhrin, Enhanced method for reinforcement learning based dynamic obstacle avoidance by assessment of collision risk, Neurocomputing (2023) 127097.

[50] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, G. Wetzstein, Implicit neural representations with periodic activation functions, Advances in Neural Information Processing Systems 33 (2020) 7462–7473.

[51] J. J. Park, P. Florence, J. Straub, R. Newcombe, S. Lovegrove, Deepsdf: Learning continuous signed distance functions for shape representation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 165–174.

[52] A. Leylavi Shoushtari, Robot body self-modeling algorithm: a collision-free motion planning approach for humanoids, SpringerPlus 5 (1) (2016) 1–18.

[53] B. Chen, R. Kwiatkowski, C. Vondrick, H. Lipson, Fully body visual self-modeling of robot morphologies, Science Robotics 7 (68) (2022) eabn1944.

[54] H.-T. L. Chiang, J. Hsu, M. Fiser, L. Tapia, A. Faust, Rl-rrt: Kinodynamic motion planning via learning reachability estimators from rl policies, IEEE Robotics and Automation Letters 4 (4) (2019) 4298–4305.

[55] T. Lee, M. Leok, N. H. McClamroch, Control of complex maneuvers for a quadrotor uav using geometric methods on se (3), arXiv preprint arXiv:1003.2005 (2010).

[56] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, C. Finn, Learning to adapt in dynamic, real-world environments through meta-reinforcement learning, Proceedings of 7th International Conference on Learning Representations, ICLR 2019 (2019) 1–17arXiv:1803.11347.

[57] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347 (2017).

[58] A. Ezquerro, M. A. Rodriguez, OpenAI ROS package, Accessed Jan.5, 2023.
URL http://wiki.ros.org/openai_ros

[59] K. Chua, R. Calandra, R. McAllister, S. Levine, Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models, Advances in Neural Information Processing Systems 2018-Decem (Nips) (2018) 4754–4765. arXiv:1805.12114.

[60] S. C. Barakou, C. S. Tzafestas, K. P. Valavanis, A review of real-time implementable cooperative aerial manipulation systems, Drones 8 (5) (2024). doi:10.3390/drones8050196.
URL https://www.mdpi.com/2504-446X/8/5/196

CRediT authorship contribution statement

Jingyu Chen: Writing – review editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. Ruidong Ma: Methodology, Conceptualization. Meng Xu: Writing – review editing, Validation. Fethi Candan: Validation, Software. Lyudmila Mihaylova and John Oyekan: Writing – review editing, Funding Acquisition, Resources, Project Administration

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgment

We would like to acknowledge the support of the Engineering and Physical Sciences Research Council (EPSRC) funding: DigiCORTEX (EP/W014688/1), NanoMan (EP/V055089/1) and Researcher in Residence Fellowship with CPI (RIR16B221020+2) for the work carried out in this manuscript.

Appendix A. Supplementary materials

Algorithm 2 Offline Meta Model-based Reinforcement Learning with Online Adaptation and Model-free Correction

```
Require: Training tasks $S_{tr}^{i=1:4}$, testing tasks $S_{te}^{i=1:3}$
Require: Offline dataset $D$, adaptation buffer $D_a$
Require: Replay buffer $D_c$
 1: Randomly initialise raw meta-model $f_{\theta^r}$
 2: // Data collection
 3: for Each training task $S_{tr}^{i}$ do
 4:     Manually collect paths $P^i = \{P_0^i, P_1^i, ..., P_5^i\}$
 5:     $D \leftarrow D \cup P^i$
 6: end for
 7: // Offline meta learning
 8: Update $f_{\theta^r}$ into $f_\theta$ using offline dataset $D$
 9: Sample a testing task $S_{te}^{j} \sim \rho(S_{te})$
10: // Online adaptation
11: Obtain $f_{\theta'}^{j}$ from online adaptation
12: for Adaptation timestep $t = 1, ...$ do
13:     Evaluate $\nabla_\theta \mathcal{L}_t(f_\theta)$ by last $K$ samples from $D_a$
14:     $\theta' \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_t(f_\theta)$
15:     Execute $a_t^*$ generated by MPC using $f_{\theta'}^{j}$
16:     $D_a \leftarrow D_a \cup \{s_t^j, a_t^j, s_{t+1}^j\}$
17: end for
18: // Online correction
19: Start training correction policy $f_c^j$
20: for Training episode $i = 1, ..., N$ do
21:     Reset environment
22:     for Timestep $t = 1, ...$ do
23:         Compute $a_t^*$ from MPC with meta-model $f_{\theta'}^{j}$
24:         Use $f_c^j$ to generate corrective action $a_c$
25:         Execute final action $a_t = a_t^* + a_c$
26:         Save $(s_t^j, a_c^j, s_{t+1}^j, r_t^j, d)$ into a replay buffer $D_c$
27:         if Enough samples are collected then
28:             Update policy and reset replay buffer
29:         end if
30:         if Terminal signal $d$ then
31:             break
32:         end if
33:     end for
34: end for
```

In this section, we first give the complete algorithm of our proposed tracking method called offline meta-model-based reinforcement learning with online adaptation and correction in Algorithm 2. More details of the setup of the training baselines in Sec.5.3.1 are introduced below,

Proximal policy optimisation (PPO): A simple model-free reinforcement learning algorithm using policy gradient.
Probabilistic ensembles with trajectory sampling (PETS): A famous model-based reinforcement learning method which employs the uncertainty-aware neural networks and a sampling-based MPC [59].
Fast Adaptation through Meta-Learning Embeddings (FAMLE): An algorithm learns a set of initialised parameters of the meta-learning for better adaptation when the diversity of the dynamics distributions among different tasks is too large [36].

For the implementation of PPO, it is trained by an Adam optimiser with the learning rate 3×10^{-4} for both the actor and critic nets. The discounted factor is 0.99. The smoothing parameter for GAE is 0.95. Moreover, the clipped range ϵ for the objective function is 0.95 and the coefficient c of the entropy is 0.005. The batch size of the replay buffer is 2048, and we divide it into 32 batches for each training process. The observation of PPO is changed into $s = [O_s, O_g^r]$ where O_g^r is the 3D relative position between the payload and the reference goal. The dataset for meta-learning in FAMLE is the same as in our proposed method.

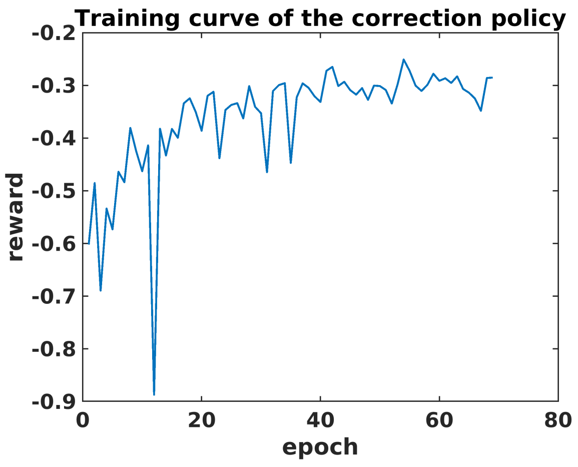


Figure A.26: The training curve of the correction policy

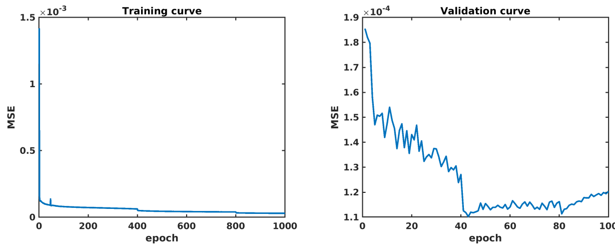


Figure A.27: The training and validation curves of the collision predictor

As shown in Fig.A.26, the training curve of the correction policy is demonstrated. This policy is trained by the proximal policy optimisation and the increasing reward shows the improvement of the policy. The training and validation curves of the collision predictor are shown in Fig.A.27. The mean square error decreases significantly with the increase of the epoch in both curves. The tracking error curves in all scenarios are shown in Fig.E.33.

Appendix B. Ablation Studies

For the learning-based tracking controller, the ablation experiments are represented by the methods PETS and MAML as shown in Tab.4. PETS, introduced as a model for approximating the dynamics function of MPC using a fully-connected neural network, operates effectively within a single task distribution. However, PETS is limited to this specific task distribution. In contrast, we replace the fully-connected neural network framework with a meta-learning model, MAML, which is capable of rapid adaptation to diverse task distributions. Finally, we integrate a correction policy with MAML to formulate our proposed design. We calculate the mean error of each method across all paths and tasks. As shown in Tab.B.7, the performance of the only meta-learning method surpasses that of the non-meta-learning method, demonstrating that the meta-learning framework is more effective than a fully-connected neural network for adapting to different tasks. Furthermore, our proposed method, which incorporates the correction policy, outperforms the only meta-learning method, validating the effectiveness of our correction strategy.

Table B.7: Ablation experiments of the tracking controller

Ablation	Euclidean error [m]
No meta learning	0.24
Only meta learning	0.163
Meta learning with the correction policy	0.103

For the path planner, we conduct ablation experiments by comparing two non-adaptive baseline methods: RRT-circle and RRT-point. In the RRT-circle approach, we eliminate the collision predictor and use a fixed circular collision constraint with a 0.5-meter radius. In the RRT-point method, there is no collision detection, and the system is represented as a point mass. We calculate the mean success rate of each method across all paths and tasks (the only RRT method is represented by the RRT-circle method). As illustrated in Tab.B.8, the performance of the only RRT method is inferior due to the absence of collision detection mechanisms. Our proposed path planner demonstrates superior performance compared to the only RRT method by employing a meta-collision predictor that dynamically adjusts collision constraints across different tasks.

Table B.8: Ablation experiments of the path planner

Ablation	Success rate
Only RRT	36%
RRT with the collision predictor	96%

Appendix C. Challenges on Simulation-to-Reality

The main challenge of real-world experiments lies in setting up diverse tasks. Specifically, we need to address how to generate a uniform wind force field and create an environment suitable for the dual-drone carrying system. In this section, firstly, we discuss the feasibility and efficiency of transferring our proposed algorithm from simulation to real-world applications.

- *3D physics simulator*: In this work, we implemented the algorithm in a 3D physics simulator that closely approximates real-world physics. The control modules are defined within the Robot Operating System (ROS), allowing for straightforward integration with real-world hardware with minimal modifications. Additionally, our approach does not rely on complex data structures such as images. Instead, it uses only vectors for positions and orientations. Consequently, when transitioning to real-world applications, the communication time with sensors is unlikely to increase significantly.
- *Computation efficiency of the learning-based tracking controller*: During testing, the MPC controller operates with a prediction horizon of 5 time steps, and the online inference speed is approximately 0.0244 seconds per iteration, which meets the requirements for real-world deployment. The correction policy runs concurrently with the MPC controller to generate optimal action outputs, with an inference time of 0.0002 seconds, significantly faster than the MPC inference time. The PPO algorithm collects interaction data and stores it in a replay buffer for subsequent updates to the actor and critic neural networks. Assuming each time step in the environment has a complexity of O(1), generating N time steps for the replay buffer has both time and space complexities of O(N). Policy loss calculation involves computing the probability ratio between new and old policies for each sample, with a time complexity of O(N). Updating the critic involves calculating the mean squared error (MSE) between the target and current value functions for each sample, also with a time complexity of O(N). Although the training updates for the actor and critic neural networks take approximately 1.2 seconds, this impact is negligible due to the low update frequency.
- *Computation efficiency of the path planner*: We do not expect the path planner's inference speed to meet real-time requirements, as the goal trajectories will be computed and finalized before the flight. Additionally, training the collision predictor requires approximately 2500 timesteps (about 10 minutes) of point cloud data for each training task. In real-world applications, a camera like the Intel RealSense, placed above the workspace, can be used to capture point cloud data for training the collision predictor and for accessing obstacle information.

Secondly, we address the challenges we encounter and show some preliminary information for our future real-world experiments. We plan to use two DJI Tello drones with custom grasping tools to achieve indoor transportation in varied wind conditions. In order to generate wind forces for indoor testing, we used a fan as shown in Fig.C.28.

However, the wind force produced by the fan is not uniform; it varies with distance from the fan and is stronger closer to it. This assumption is different from that used in the simulation and as a result, caused difficulties in transferring from simulation to the real-world. While our proposed algorithm performs well in simulation under the assumption of a static and uniform wind field, real-world conditions involve unknown and rapidly

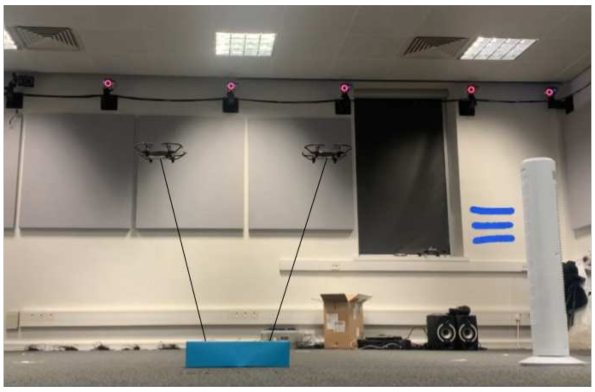
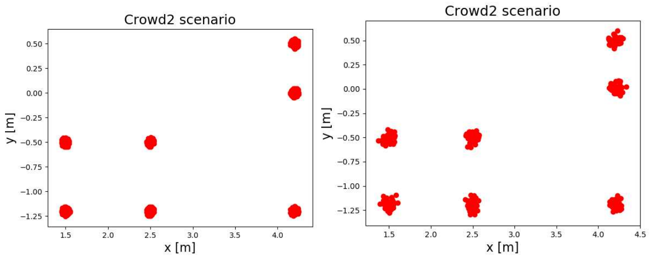


Figure C.28: The setup of the future real-world experiments

changing wind forces. To demonstrate the adaptability of our algorithm to these time-varying wind conditions, we implement the additional experiment **in simulation**, which can be found in Section 5.3.3.

Appendix D. Noisy observation of obstacles

We considered static obstacles in our work. The observations of obstacles consisted of a collection of 2D spatial points, which can be captured by a Kinect camera or other devices that provide the initial global map of obstacles. Currently, the observation is obtained perfectly without noise or errors. However, observation errors can shift the position distribution of obstacles and may influence collision detection. As shown in Fig.D.29, we added Gaussian noise with a mean of 0 and a standard deviation of 0.03 to the original observation in the Crowd2 scenario. This results in an increase in the occupancy of the obstacles.



(a) Original observation (b) Noisy observation

Figure D.29: Adding Gaussian noise to observation of obstacles in the crowd2 scenario.

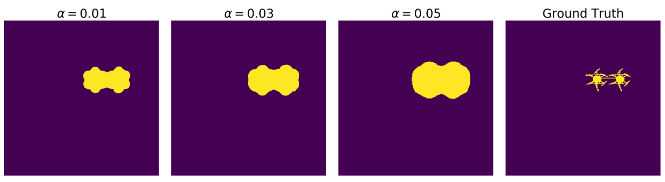


Figure D.30: Predicted space occupancy of the system with different distance thresholds

To address noise, we maintain the system state and adjust the distance threshold α defined in Algorithm 1 to get the inflated shape of the system, ensuring safety. The Fig.D.30 illustrates how the coefficient influences the predicted occupancy of the dual-drone payload system. Increasing the threshold decreases the safe distance for collision detection, thereby accommodating the noise in obstacle observations and ensuring collision-free navigation.

Appendix E. Prediction by the meta-dynamics model

In Fig.12, the prediction of the payload by the adapted meta-dynamics model during the online tracking has been shown. In this section, we demonstrate more results regarding the prediction of the UAVs and other states in Fig.E.32. The results show that the prediction of the position on the x and z axes of the inertial coordinate is more accurate than that on the y axis. Additionally, the prediction of the payload's pitch angle is more accurate than those for the roll and yaw angles.

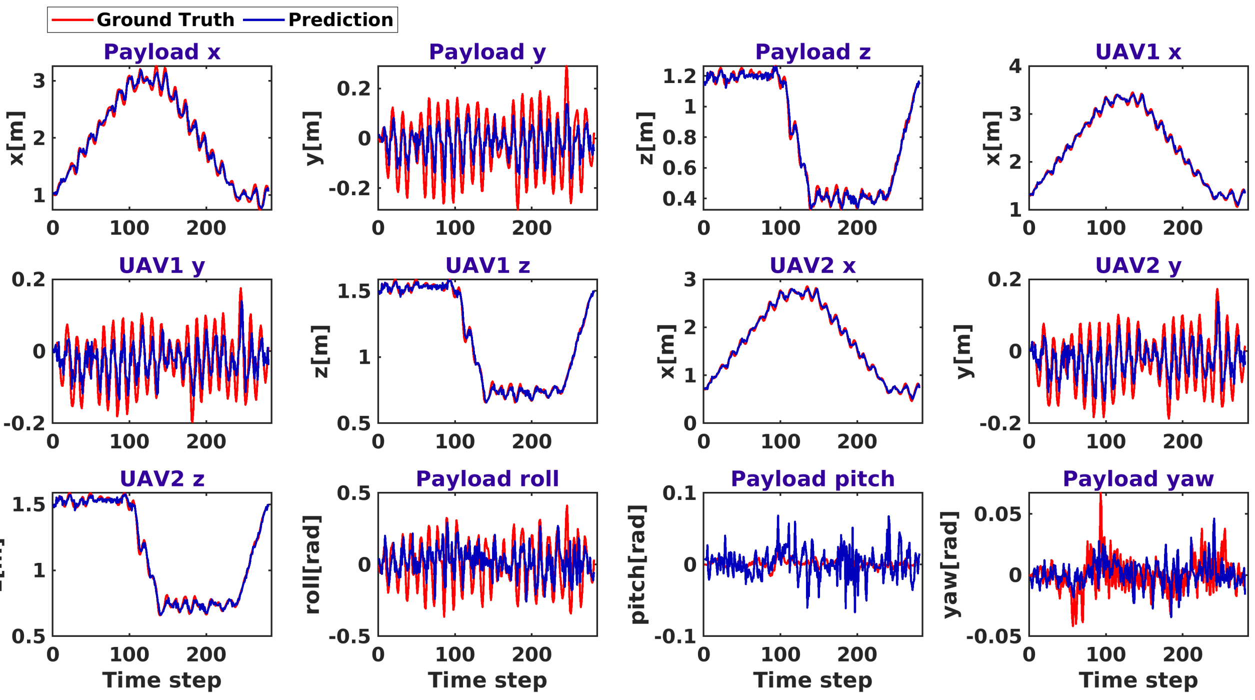


Figure E.31: Online prediction of the states in Task1 and square path

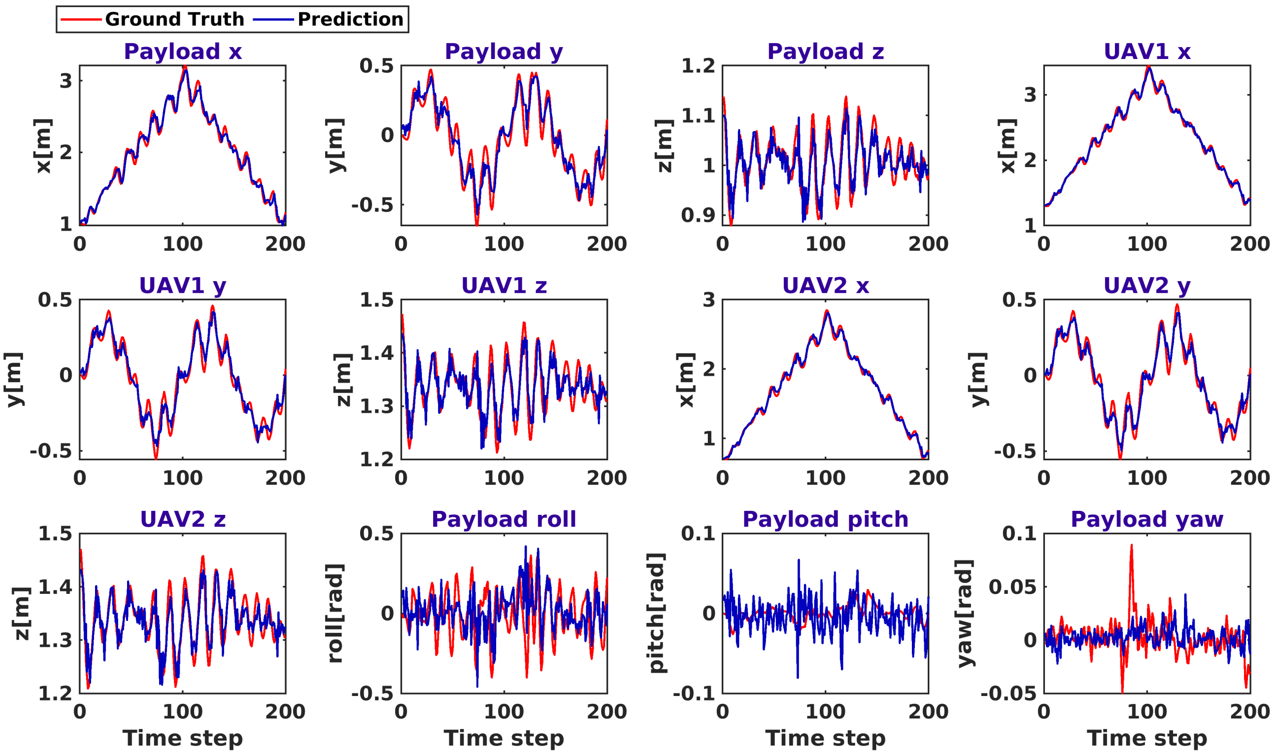


Figure E.32: Online prediction of the states in Task1 and *figure8* path

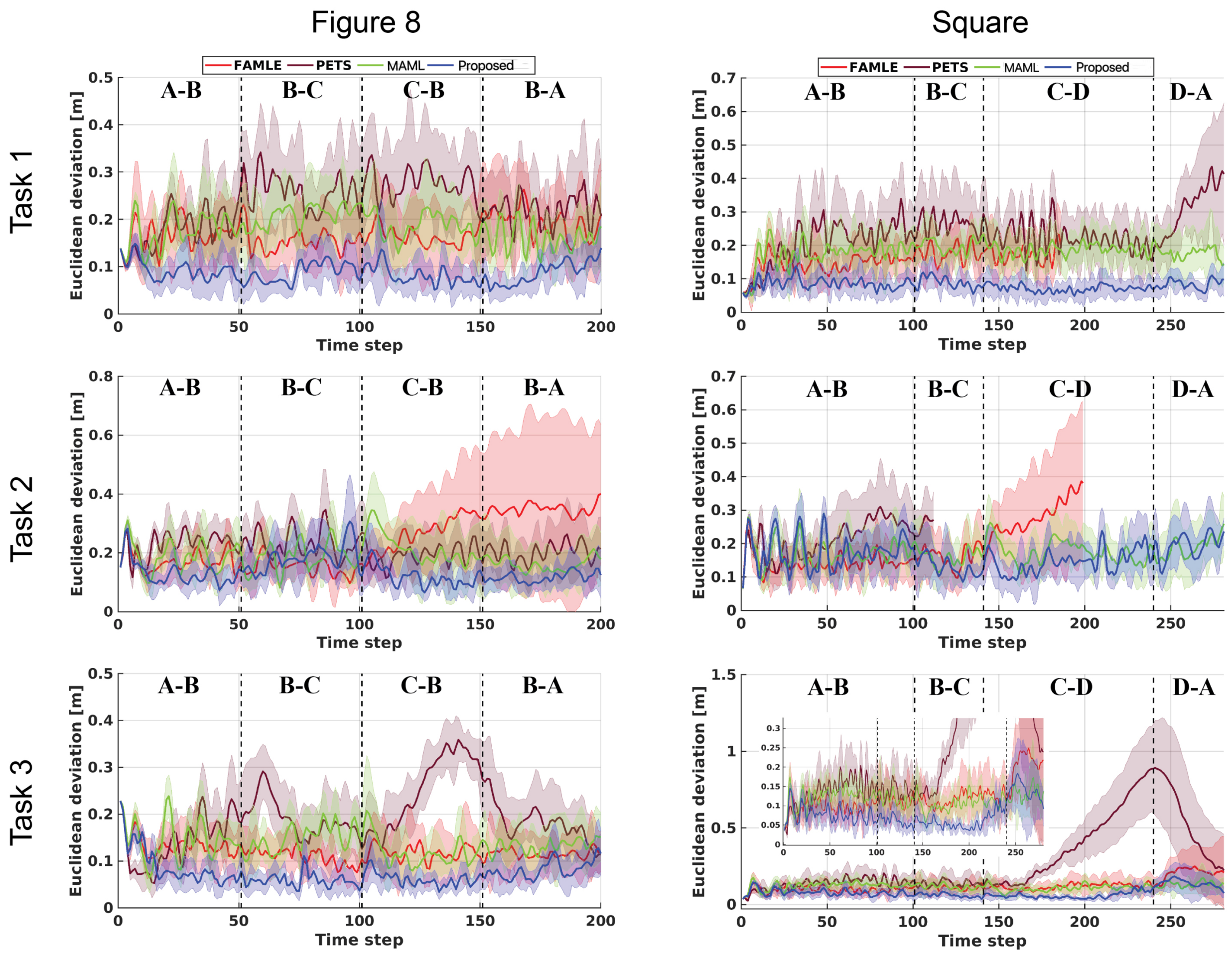


Figure E.33: Visualisation of the results with different methods in three testing tasks and two paths. The corresponding curves of the Euclidean deviation over all time steps in one testing trial. These curves serve as supplementary material for Fig.13.

Jingyu Chen received the M.Eng. degree in electrical and electronic engineering from the University of Leicester and the M.Sc. degree in robotics from The University of Sheffield. He received his Ph.D. degree with the Department of Automatic Control and Systems Engineering, The University of Sheffield. He is currently working in the Institute of Software, Chinese Academy of Science. His research interests include reinforcement learning, autonomous system and generative model.

Ruidong Ma received a B.Eng. degree in electrical and electronics engineering from the University of Liverpool and the M.Sc. degree in human and biological robotics from Imperial College London. He is currently pursuing the Ph.D. degree with the Department of Automatic Control and Systems Engineering, The University of Sheffield. His research interest includes human–robot collaboration in complex manual manufacturing processes.

Meng Xu received her B.E. degree from Northwestern Polytechnical University, Xi'an, in 2018, and her Ph.D. degree from Queen Mary University of London. She was also a research associate at the University of Sheffield. She is currently an assistant professor at the University of International Business and Economics (UIBE). Her research interests include spatial intelligence, machine learning, deep learning, augmented reality, and urban computing.

Fethi Candan received B.Sc. degrees in Electrical Electronics and Mechanical Manufacturing Engineering between 2011-2016, from Bilecik S.E. University, Bilecik, Turkiye. His M.Sc. degree is in Control and Automation Engineering in 2018, Istanbul Technical University, Istanbul, Turkiye. He received his PhD degree with the Automatic Control and Systems Engineering Department, Sheffield, United Kingdom. Currently, he is working at the University of Idaho as a postdoctoral fellow at the Department of Mechanical Engineering and Soil Water Systems. His research interests are in the areas of control theory, state estimation, unmanned air vehicles, swarm, intelligent systems, and their real-world applications.

Lyudmila Mihaylova is Professor of Signal Processing and Control at the Department of Automatic Control and Systems Engineering at the University of Sheffield, United Kingdom. Her research is in the areas of machine learning and autonomous systems with various applications such as navigation, surveillance and sensor network systems. She has given a number of talks and tutorials, including the plenary talk for the IEEE Sensor Data Fusion 2015 (Germany), invited talks University of California, Los Angeles, IPAMI Traffic Workshop 2016 (USA), IET ICWMMN 2013 in Beijing, China. Dr. Mihaylova is an Associate Editor of the IEEE Transactions on Aerospace and Electronic Systems and of the Elsevier Signal Processing Journal. She was elected in March 2016 as a president of the International Society of Information Fusion (ISIF). She is on the board of Directors of ISIF and a Senior IEEE member. She was the general co-chair IET Data Fusion Target Tracking 2014 and 2012 Conferences, Program co-chair for the 19th International Conference on Information Fusion, Heidelberg, Germany.

John Oyekan is a Chartered Engineer (CEng) and Senior Lecturer in Human-Centred AI for Autonomous Manufacturing in the Department of Computer Science. He was previously a Lecturer in Digital Manufacturing at the University of Sheffield. He received a Ph.D degree in Computer Science and Electronic Engineering from the University of Essex as well as a MSc Robotics and Embedded Systems from same. Prior to the University of Sheffield, he was an Engineer at the Manufacturing Technology Centre in Coventry were he developed software architectures and algorithms for Autonomous Systems and a Research Fellow at Cranfield University carrying out research in Manufacturing Informatics. He has worked in a startup, government funded catapult, automotive industry and his work is characterised by both fundamental and applied research in collaboration with partners in the automotive, aerospace and manufacturing sectors. He is passionate about translating blue sky research and thinking into practical solutions for industry challenges. As an academic, he has over 50 publications in the areas of swarm robotics, manufacturing informatics, bio-inspired algorithms and sensing.