Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Motion Planning and Robust Decentralized Fault-Tolerant Tracking Control of Hybrid UAVs and Biped Robots Team System for Search and Rescue Usage

BOR-SEN CHEN[1,2], (Life Fellow, IEEE), TING-WEI HUNG[1]

[1]Department of Electrical Engineering, National Tsing Hua University, Hsinchu 30013, Taiwan

[2]Department of Electrical Engineering, Yuan Ze University, Taoyuan 32003, Taiwan

Corresponding author: Bor-Sen Chen (bschen@ee.nthu.edu.tw)

This work was supported by the Ministry of Science and Technology of Taiwan under Grant MOST 108-2221-E-007-099-MY3.

ABSTRACT In this study, we investigate a motion planning and robust fault-tolerant control of a hybrid UAVs and biped robots team system (URTS) for the purpose of search and rescue (S&R). The system architecture of URTS is proposed at first to illustrate the issues to be addressed in URTS and the relationships between them. The task allocation and path planning problems are first investigated. Next, we focus on the local motion planning problem for UAV flying and robot walking behavior, and then the tracking control problem for UAVs and robots in the hybrid team system. The relationship between local motion planning and tracking control, i.e., the transformation of the reference trajectory, is also explored in detail. By converting the dynamics models of UAV and robot into a unified agent dynamics model, the robust H_∞ decentralized observer-based feedforward fault-tolerant control (FTC) strategy is proposed for the agents in URTS. A novel smoothing signal model of fault signal is embedded into the linearized system to achieve the active FTC through observer estimation. Then, the design of the robust H_∞ decentralized observer-based feedforward FTC strategy of URTS is transformed into a linear matrix inequality (LMI) -constrained optimization problem of each agent in the hybrid team system which can be solved by a two-step design procedure. With the help of MATLAB LMI Toolbox, the H_∞ decentralized fault-tolerant tracking problem of each UAV and robot in URTS is effectively solved. Finally, the simulation results are used to demonstrate the operation of URTS and to verify the effectiveness of the proposed H_∞ decentralized fault-tolerant tracking control method of hybrid URTS under the external disturbance and the actuator and sensor fault.

INDEX TERMS biped robot, fault-tolerant control, heterogeneous multi-agent system, robust H_∞ control, S&R, smoothing signal model, UAV, hybrid UAVs-UGVs team system

I. INTRODUCTION

IN recent years, the unmanned vehicle (UV) has attracted attention due to the advances in communication technology, sensing devices, and computing power. It not only reduces labor costs and brings convenience to life, but more importantly, it can replace some dangerous jobs for humans. Due to these benefits, it has been widely used in many scenarios, such as S&R, battlefield, logistics and transportation, surveillance, etc. [1]. Compared with a single UV, multiple UVs can perform more complex tasks and are more robust due to a large number of agents [2]. However, the cost is that the design of such a multi-agent system (MAS) becomes more intricate as there are more problems to be resolved, such as formation, collision avoidance between agents, task allocation, and cooperation between agents [3]. In addition to the number increasment, a heterogeneous multi-agent system (HMAS) combining various types of UV is also valued [4]. Compared with homogeneous multi-agent system, it can adapt to a wider variety of application scenarios because each agent has different aptitudes.

To construct an unmanned HMAS, the three required key capabilities are perception, decision-making and control.

Perception is to obtain information through the sensor (e.g., localization or computer vision), decision-making is to make decisions through the sensor information, and control is to execute the decision through the actuator. To limit the scope of this paper, we focus on decision-making and control problem only. Three main problems of decision-making in an unmanned HMAS are task allocation, path planning and collision avoidance. Task allocation is to optimally assign tasks to each agent under some constraints such as agent capabilities, fuel cost, time cost, etc. [5]. Path planning is to optimally plan paths for each agent while subject to constraints such as agent kinodynamic properties, distance, obstacles collision, etc. [6]. Collision avoidance is to avoid collision with obstacles. Although collision avoidance is often concerned in path planning, the collision avoidance system is also independently studied because of the requirements for the safety and reliability of the actual system [7].

Although there are many types of UVs to make up an unmanned HMAS, Unmanned Aerial Vehicle (UAV) and Unmanned Ground Vehicle (UGV) have been the subject of major recent research because of their availability and applicability. Additionally, the complementarity between them also makes such a system more potential [8]. In other words, UAV is widely used in reconnaissance due to the high mobility. However, the carrying capacity of UAV is very low compared to UGV since there is no ground support. In contrast, UGV has higher carrying capacity but is easily restricted by ground obstacles and cannot move at high speed. For these reasons, a hybrid UAVs-UGVs team system will be more appealing. To discuss more concretely, we consider a hybrid UAVs-UGVs team system for S&R. For the need for search mobility, we choose quadrotor aircraft as UAV. In order to deal with the complex terrain of the S&R environment, we choose biped robot as UGV. Even though other types of UGVs like wheeled robots and vehicles are easier to handle than biped robot, the high degree of freedom and the compatibility of the human environment still makes it a good candidate of UGV in a S&R system.

To the best of the authors' knowledge, most of the literature focus on only one specific problem in such an unmanned multi-agent S&R system, such as task allocation problem, path planning problem or control problem. Additionally, few literatures illustrate the relationship between these problems. This leads us to propose a system architecture of hybrid UAVs and biped robots team system (URTS) for S&R usage. The flowchart of decision-making and control process of URTS is also given. We divide it into five main hierarchical processes, i.e., (i) task allocation, (ii) path planning, (iii) behavior layer, (iv) local motion planning and (v) tracking control. It is because the URTS needs to be able to assign different tasks to agents to perform first. After a task is assigned, if the task is to reach a goal location, a path to reach it needs to be planned. To make agent move on the path, a behavior corresponding to the environment is required to be determined. Then, a local motion corresponding to the behavior of the agent needs to be designed. Finally, a controller must be designed to track the trajectory of the motion. In order to further limit the scope of the study, we will focus on the latter two processes. But to illustrate how the whole system works, the first three problems are also briefly stated.

The local motion planning is the bridge between path planning and tracking control since the path found by path planning algorithm and the path enforced to follow by a controller are not necessarily the same. The reason is that path planning algorithm usually treats the agent as a point, while the actual agent in the physical world is a mechanical system for the tracking control design. A mechanical system means that there exist kinodynamic constraints. This makes certain paths impossible to follow for an actual agent, such as paths that are not smooth, have too large curvature, or require too large velocity and acceleration. Although some literatures directly tackle the kinodynamic constraints path planning problem [9], this paper splits path planning into three steps, i.e., (i) path planning, (ii) behavior layer and (iii) local motion planning for clarity. Through this decomposition, we can focus on the local motion planning of specific behaviors. The local motion planning of flying behavior for UAV and walking behavior for robot is studied in this paper, especially the latter. The local motion planning of biped robot walking, i.e., stable walking pattern generation, is a popular research topic due to its challenge [10].

The tracking control is to control an agent to follow a desired trajectory. There are many control strategies for MAS. According to the way of the design of controller, it can be divided into centralized control and decentralized control in control field [11]. Centralized control means there exists a powerful central controller in MAS to gather the state information of MAS and send the control command back to each agent to reach a global goal. Due to the powerful nature of the central controller, control commands can be determined well and quickly. But when it fails, the whole system will be completely paralyzed. In contrast, decentralized control means that each agent has its own controller to collect and control the agent's own state information. Under this architecture, although the global goal cannot be achieved, the possibility of paralyzing the entire system due to the failure of the controller can be avoided.

Besides, the formation control is also a topic in MAS [12]. Its purpose is to keep a MAS in a formation while moving. Although formation control provides a simple framework for the control of a large number of agents, considering the complexity of the disaster relief environment, formation will make the application of URTS inflexible. It is because we expect that agents in URTS need to organize multiple teams of different scales and types to deal with multiple tasks of different scales and types in a disaster relief environment. In this situation, it is more reasonable to treat each agent in URTS as an independent individual to follow a specific trajectory to form a team formation and specify an independent trajectory for each agent to follow.

In order to cope with the fault in the actual system, the fault-tolerant control (FTC) has also been widely studied. According to the way of handling the fault, it can be divided into the passive FTC and the active FTC [13]. The passive FTC treats the fault as an unknown system perturbation and designs a control law to tolerate it. In contrast, the active FTC will first estimate and identify the fault and then compensate it through the controller. Despite the extra complexity in controller design, the active FTC will outperform the passive FTC due to the extra estimation steps. Based on the foregoing, a robust H_∞ decentralized observer-based feedforward FTC is proposed to deal with the control problem in hybrid URTS.

The contributions of this study are described as follows:

1) A system architecture and system flow of hybrid URTS for the purpose of S&R are proposed so that the issues involved in hybrid URTS and their relationships can be defined and resolved.
2) A transformation between the trajectory generated by the path planning algorithm and the trajectory required for tracking control design is proposed to enable some common path planning algorithms can be applied to the team formation tracking control of agents in hybrid URTS.
3) A general agent dynamics model is proposed so that the robust H_∞ decentralized observer-based fault-tolerant tracking control problems of the heterogeneous agents in URTS, UAVs and biped robots, can be solved together.

The remainder of the paper is organized as follows. In Section II, a system architecture of URTS in S&R usage is proposed and the function and relationship among its components, i.e., task allocation, path planning, behavior layer, local motion planning and tracking control are described. In Section III, the dynamics models of agents in URTS are given to design the motion of UAV flying and biped robot walking behavior and the control strategy of agents. In Section IV, a robust H_∞ decentralized observer-based feedforward FTC is proposed for the agents in URTS with the help of a general agent dynamics model. In Section V, a simulation example is given to illustrate the operation of the system architecture of URTS and to verify the effectiveness of the proposed tracking control method. In Section VI, a conclusion is maded.

Notation 1: $diag(A_1, A_2, \dots, A_n)$: a block diagonal matrix with main diagonal blocks $A_1, A_2, \dots, A_n$. A^T: transpose of A. $A > 0$: a positive definite matrix. (a_n): a sequence. (a_{k_n}): a subsequence of a sequence (a_n). $[a_{j,k}]$: A matrix with the entries $a_{j,k}$ in the jth row and kth column. $|S|$: size of a set S. $\otimes$: Kronecker product. I_n: n-dimension identity matrix. $x(t) \in L_2\left[0, t_f\right]$ if $\int_0^{t_f} x^T(t)x(t)dt < \infty$. $Sym(A)$: sum of a matrix A and its tranposed, i.e., $Sym(A) = A + A^T$.

II. PRELIMINARIES OF URTS IN S&R USAGE

The URTS will start with a given S&R area, and end with mission completed. The URTS is composed of N_T teams and a ground station. Each team contains N_A agents with 1 UAV and $N_A - 1$ robots. Hence, the jth agent in the ith team is denoted as $\alpha_{i,j}$, where $i = 1, 2, ..., N_T$ and $j = 1, 2, ..., N_A$. The UAVs are chosed as the first agents in each team, i.e., $\alpha_{i,1}, i = 1, 2, \dots, N_T$. Each agent has environmental sensing capability and load capability, while the ground station is responsible for computing and decision-making. Besides, there are communication channels between agents and ground station through wireless network.

To complete search tasks, each team is designed to be responsible for a small area of the overall S&R area, and each agent will be assigned an appropriate path to cover the area. To complete rescue tasks, whenever a goal (e.g., victims or disaster area) is found by the machine vision of nearby agent, the ground station will assign some agents to the location of goal.

If a task is to reach a location of certain goal, we need to find a collision-free path to reach it. Hence, each agent will also sense distance-related information about its surrounding and send it back to the ground station. The ground station will combine this information with the goal location assigned by task allocation algorithm and then make a decision to avoid obstacles and other agents nearby.

We need to determine specific behavior to follow the path found by path planning algorithm according to the situation of the environment especially for agents with complex mechanical systems like robots. Since such a system has a high degree of freedom, there are many ways to follow the same path (e.g., a robot can walk or run to follow the same path). Furthermore, agents in URTS are not always following the path. Sometimes they need the behavior such as stop to look around, get supplies and put supplies. To meet these needs, a behavioral layer is necessary.

In order to make a behavior, we must design a corresponding motion by local motion planning. The motion is a prescribed reference trajectory for an actual mechanical system to follow. Finally, a tracking controller is designed for each agent to follow this desired reference trajectory.

The agents overall have the same system architecture in URTS except for some subprocess differences. Followed by the concept in [14], a system architecture for an agent employed for S&R is proposed as shown in Fig. 1. The Simultaneous Localization And Mapping (SLAM) block converts sensor information into the location of agents q_{start} and an occupancy map $\mathcal{C}$. The visual object recognition block provides distance information and object information through the analysis of sensor information. The object information provides agent machine vision that enables it to determine an appropriate behavior (e.g., a robot can see an obstacle and decides to climb through it). The detailed functions of remaining 5 blocks, i.e., Task Allocation, Path Planning, Behavior Layer, Local Motion Planning and Tracking Control, will be explained in the following subsections.

A. TASK ALLOCATION

In the URTS, it can be expected that each agent $\alpha_{i,j}$ will be assigned to several specific tasks, such as searching a specific

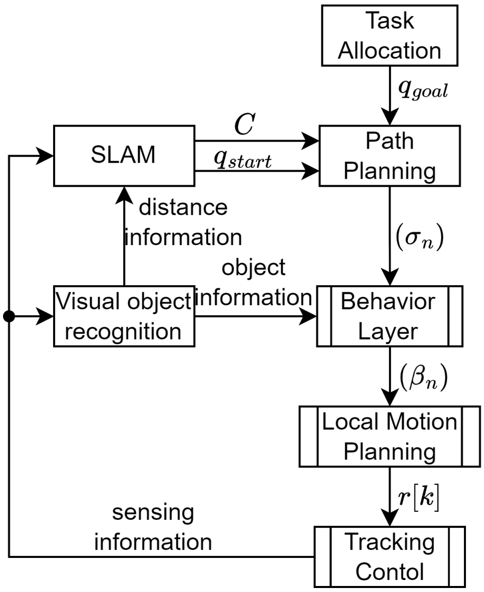

FIGURE 1: The system architecture of agent in URTS. The 2 blocks on left hand side are used to convert the low-level sensor information into high-level information, such as map, start point and object information. The 5 blocks on right hand side are the flow of an agent performing a S&R mission. From the top to the bottom, it is the decision of the task position, the planning of the path, the decision of the behavior, the planning of the local motion trajectory, and the low-level tracking control.

area, or delivering supplies to disaster area, etc. However, the number of agents and tasks is more than one, and each agent has different capabilities (e.g., moving speed or load capacity) and status (e.g., their own position or the amount of supplies carried), and each task has different characteristics (e.g., urgency, position, amout of supplies needed). Therefore, the results of task allocation can be "good or bad", which leads us to finding the optimal allocation. This problem is referred to a task allocation problem or multi-robot task allocation problem. A problem formulation and a mathematical model of problem can be found in [15]. Although many different formulations and models have been employed to solve the task allocation problem, the common goal is to find a set of agent-task pairs to achieve a specific cost function. In this paper, we assume that the tasks have been properly assigned so that every agent knows a destination q_{goal} it needs to go at every moment.

Remark 1: This block is like a commander since it is used to assign task for agents. Thus, if the real S&R system has human experts as commanders, he can replace its job or make decisions together with it to maximize the rescue value.

Remark 2: Although each agent has its own team, agents can also work across teams. For example, if the result given by the task allocation algorithm contains the agent-task pairs $(\alpha_{1,5}, T_1)$ and $(\alpha_{2,2}, T_1)$, then the agent $\alpha_{1,5}$ in team 1 and the agent $\alpha_{2,2}$ in team 2 will execute task T_1 together.

B. PATH PLANNING

After a destination q_{goal} is assigned for each agent, next step is to find a collision-free path from current position to it. There must exist multiple feasible paths to go. Similar to task allocation, we usually want to find an optimal path. There are several path planning algorithms to handle this problem. Due to the developmental and universal nature of roadmap-based path planning algorithm, this paper considers it as the path planning method of URTS. This method attempts to discretize the search space into interconnected roads and find the path on it.

According to the way of pathfinding, it can be divided into multi-query planner and single-query planner [16]. Multi-query planner will first construct a roadmap and then use a graph search method on it to query the best path, such as Probability Road Map (PRM), Visibility Graph, and Voronoi Diagrams [17]. Single-query planner will complete the pathfinding by constructing and querying simultaneously, such as Rapidly-exploring Random Tree (RRT), Expansive Space Tree (EST), and Ariadne's Clew [16]. However, the environment is dynamic rather static for URTS so some extra structures need to impose on the aforementioned planner. Some common dynamic planners can also be found in [16], such as PRM with D* search algorithm, dynamic RRT, and extended RRT. All of the above common roadmap-based planners can be applied to the URTS.

Remark 3: To avoid agents colliding with each other, the concept of multi-agent path planning is proposed [18]. However, URTS operates in a large environment so the probability of collision is small and the agents have the ability to communicate. Therefore, when the paths collide, the mechanism of waiting for the other side to pass can be used to avoid the collision problem.

Furthermore, the constraints imposed by the mechanical structure are needed to consider within pathfinding process mentioned by other literatures but it will be left to local motion planning block to deal with. The reason is that URTS works on a complex environment and therefore requires a variety of behaviors to respond, and different behaviors have different constraints (e.g., curvature constraints between running and walking behavior are expected to be different for robots). In this case, an unified planner will become overly complex and impracticable. Therefore, we divide path planning into three subprocesses, i.e., path planning, behavior layer and local motion planning. The path planning block becomes a global planner and regards an agent as a point without kinodynamic constraints. The local motion planning block becomes a local planner and considers the motion planning of a specific behavior.

By treating a roadmap-based path planning algorithm as a black box, the output is a sequence (or sayed waypoints), and the three inputs are current configuration q_{start}, goal configuration q_{goal}, and configuration space $\mathcal{C}$. Current configuration is obtained by GPS, inertial measurement unit, or other locating techniques. Goal configuration is obtained by the previous block, task allocation block. Configuration space

$\mathcal{C}$ is constructed by environment information through sensors of agents online or in advance by human knowledge offline. $\mathcal{C}$ is a space containing all possible configurations of agents which are composed of free space $\mathcal{C}_{free}$ and obstacle space $\mathcal{C}_{obs}$, where $\mathcal{C} = \mathcal{C}_{free} \cup \mathcal{C}_{obs}$ and $\mathcal{C}_{free} \cap \mathcal{C}_{obs} = \emptyset$. For a simpler explanation of how the URTS works, the following assumptions are made.

Assumption 2.1: The locating ability of the URTS is perfect so every agent can know its current configuration q_{start}.

Assumption 2.2: A Task Allocation algorithm is already designed so that every agent can know its goal configuration q_{goal}.

Assumption 2.3: The URTS is supposed to have a perfect real-time mapping ability so a real-time configuration space $\mathcal{C}$ can be obtained.

Assumption 2.4: UAVs do not consider obstacle collision, so the path of UAVs can be directly assigned rather than found by planner. Robots do not consider obstacle collision in the direction perpendicular to the ground.

From above assumptions, a path of agent can be expressed as a sequence

$$(\sigma_n), n \in \mathbb{Z} \cap [1, k_f], \sigma_n \in \mathcal{C} \tag{1}$$

where k_f is the time step when reaching goal. Since the path planning is dynamic, (σ_n) is composed of multiple segments actually. Let (σ_{k_n}) be the subsequence of (σ_n), where k_n is the time step when a replanning decision is occured. Then, the segments of path from the result of the replanning in time step k_n can be expressed as sequences $(\sigma_m), m \in \mathbb{Z} \cap [k_n, k_{n+1})$. For agents, the replanning decision can be due to a goal changing that is made by human or task allocation block. For robot, it can be a collision detected by a dynamic roadmap-based planner. The resulting path (σ_n) is passed to the next block, Behavior Layer.

C. BEHAVIOR LAYER

Path planning tells agents where to go but not how since we regard the agent as a point. Taking robot as an example, it may walk, run, climb, or jump to follow the path (σ_n) according to real scenario. These behaviors with changing position are classified as "moving" behavior in this paper. Besides, the agents in the hybrid URTS do not always moving. Somtimes they have to suspend to take an action (e.g., getting and putting supplies, rotating in place to collect more environment information) or deal with some unexpected situations (e.g., no path found, the robot falls). These behaviors without changing position are classified as "action" behavior in this paper. More behaviors can be added so that the agent can have more ways to act with environment but there must have a corresponding behavior every moment otherwise the agent will lose control. The sequence of these behaviors can be expressed as:

$$(\beta_n), n \in \mathbb{Z} \cap [1, k_f], \beta_n \in \mathcal{B} \tag{2}$$

where $\mathcal{B}$ is the set of behaviors. It means that the path (σ_n) is divided into many segments and each segment corresponds to a specific behavior. By the object information in Fig. 1, an appropriate behavior can be judged. However, it will be a rather complicated project, so this article only gives the structure of the behavior set. The behavior set $\mathcal{B}$ of agent in URTS can be roughly discribed in Fig. 2.

D. LOCAL MOTION PLANNING

After a specific behavior is determined, the next step is to design a motion to achieve the specific behavior. Local motion planning block is like path planning block but with smaller scale and higher resolution and precision. Collision checking is needed since we consider agent as a point in path planning block but it is a real mechanical body here. Furthermore, the kinodynamic constraint is handled in this block. Although motion planning and path planning are separated into two blocks, the technologies involved are similar and often with the same notion in other literature. Therefore, the output of this block is also a path $r[k]$. The flowchart of local motion planning block is shown in Fig. 3

In Fig. 3, some basic behaviors of UAV and robot mentioned before are given to futher illustrate the flow of this block. To limit the scope of this article, we only focus on the motion planning of flying behavior of UAV and walking behavior of robot. They are belong to the moving behavior in Fig. 3. A more detailed description will be given in the next section.

E. TRACKING CONTROL

To analyze the control problem in the continuous time domain, $r[k]$ will be first converted to a continuous signal by D/A convertor with a timescale, i.e., sampling period. By analyzing the dynamic model of each agent, a desired reference trajectory $r(t)$ is designed. $r(t)$ describes the position and orientation that need to be reached over time by a machine system governed by a dynamic equation. Note that the path and trajectory are distinguished in some literature. Different from path (e.g., (σ_n) or $r[k]$), a trajectory $r(t)$ has considered the time in physic world. We also distinguish them in this way in this article.

If each agent in hybrid URTS can track each own reference trajectory $r(t)$, then they can move in the physical world as we expect. To this end, a control method needs to be designed. It will be discussed detailly in Section IV. In addition, the sensing information collected by the sensors in this block is not only used for control but also sent back to the upper layer as shown in Fig. 1.

III. SYSTEM DESCRIPTION OF UAVS AND BIPED ROBOTS IN URTS

In order to design a reference trajectory $r(t)$ for the motion of UAV and robot in URTS, their dynamic models must be given first. After the system description of UAV and robot in URTS, the motion planning of flying and walking as shown in Fig. 4 will be discussed subsequently and separately in the next two subsections. Before the discussion of the motion

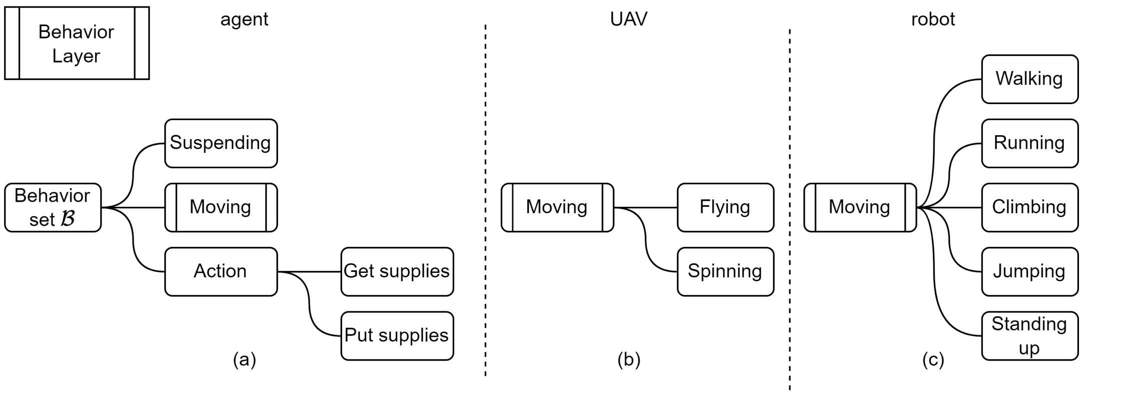

FIGURE 2: The structure of behavior layer. (a) The behavior set of each agent in URTS. The leaves of the tree structure are the possible behaviors an agent can take. The behavior to be took at every moment by each agent will be decided in this block. For UAVs, the moving behavior set is predefined in (b). For robots, the moving behavior set is predefined in (c). (b) The moving behavior set of each UAV. (c) The moving behavior set of each robot.

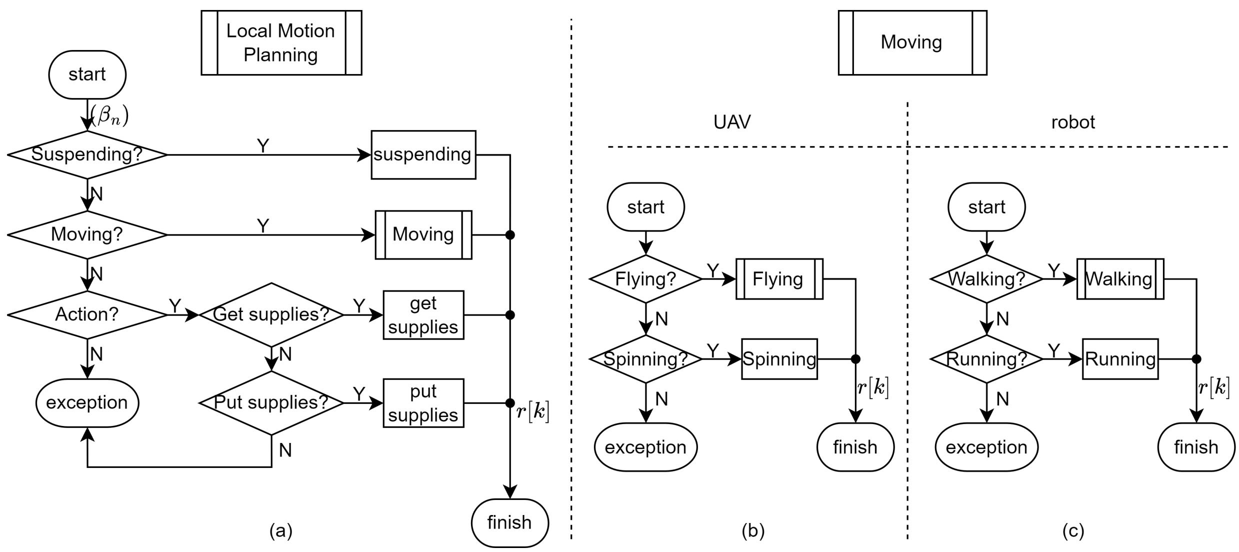

FIGURE 3: The flowchart of local motion planning. (a) The local motion planning of each agent in URTS . The corresponding motion planning of an agent will be executed according to the behavior determined by the behavior layer. The exception terminator in figure represents an exceptional condition that performs unconsidered behaviors. The reference path $r[k]$ is a sequence (or say a discrete signal) designed by a specific behavior block. For UAVs, the moving block is predefined in (b). For robots, the moving block is predefined in (c). (b) The moving block of each UAV where the flying block is predefined in 4 (c) The moving block of each robot where the walking block is predefined in 4

planning of these two behaviors, the following assumption is maded.

Assumption 3.1: The space between obstacles is large enough to eliminate the need for collision checking again, and the average speed of agents is slow enough to ignore dynamic constraints

However, for these two behaviors, there exists an inevitable kinematic constraint on the curvature of local motion. Although an accurate reference trajectory without breaking the curvature constraint can be designed, it is not easy to solve this problem. Additionally, it is not nessasry for these two behaviors in URTS since they are uesd to move from one location to another while the effect of error during moving caused by breaking the curvature constraint is relatively insignificant. At the same time, the *Assumption 3.1* makes sure the error will not cause collision. As an alternative, this problem can be handled by curve fitting which can be regarded as a post-process of the path (σ_n). The post-process will appear in the begining of motion planning process of these two behaviors as shown in Fig. 4.

A. MOTION PLANNING OF FLYING OF UAV

A dynamic model about how an UAV in URTS moves in the physical world is given first. By Newton-Euler equation, the

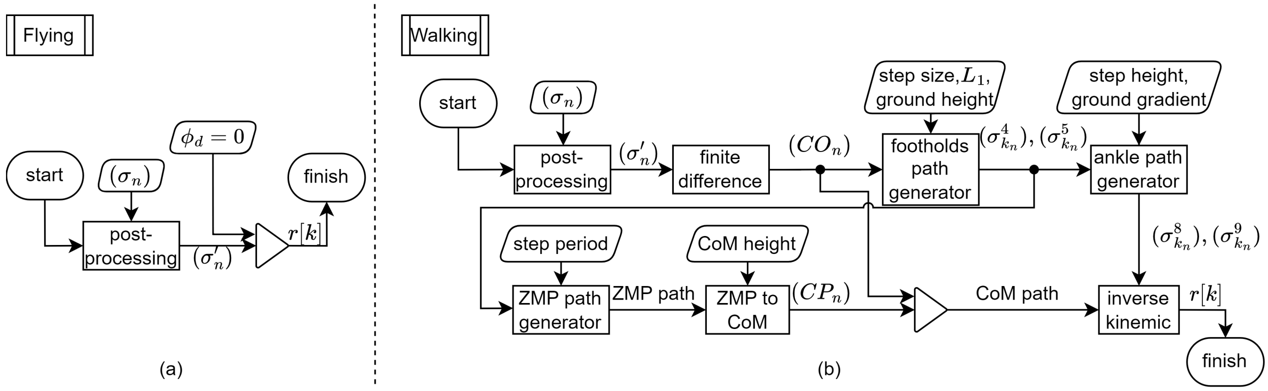

FIGURE 4: The flowchart of flying block and walking block in 3. In both blocks, a smoothed path is obtained by post-processing first. Then the respective motion of behavior is designed. (a) The flying block of each UAV. By combining the position path (σ'_n) and setting $\phi_d = 0$ (let UAV fly without spinning), we have the reference path $r[k]$. (b) The walking block of each robot. Follow the process in Section IV-B, we have the reference path $r[k]$.

dynamic model of each UAV in URTS can be formulated as [19]:

$$\begin{aligned}\begin{bmatrix} f_u \\ \tau_u \end{bmatrix} &= \begin{bmatrix} mI & 0 \\ 0 & J \end{bmatrix} \begin{bmatrix} \ddot{X} \\ \ddot{\Theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \dot{\Theta} \times (J\dot{\Theta}) \end{bmatrix} + \begin{bmatrix} f_g \\ 0 \end{bmatrix} \\ &+ \begin{bmatrix} K_F & 0 \\ 0 & K_\tau \end{bmatrix} \begin{bmatrix} \dot{X} \\ \dot{\Theta} \end{bmatrix}\end{aligned} \tag{3}$$

where $J = \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix}$, $K_F = \begin{bmatrix} K_x & 0 & 0 \\ 0 & K_y & 0 \\ 0 & 0 & K_z \end{bmatrix}$, $K_\tau = \begin{bmatrix} K_{\tau_x} & 0 & 0 \\ 0 & K_{\tau_y} & 0 \\ 0 & 0 & K_{\tau_z} \end{bmatrix}$, $f_u = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} = R(\Theta) \begin{bmatrix} 0 \\ 0 \\ F \end{bmatrix}$, $\tau_u = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}$, $X = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$, $\Theta = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}$, $f_g = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix}$, $R(\Theta) = R_z(\psi)R_y(\theta)R_x(\phi)$, $R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$, $R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$. g is the gravity acceleration, m and J are the mass and inertia matrix of UAV, respectively, τ_u and F are the total torque and force acting on UAV, respectively, Θ is the Euler angles in body frame, X is the postion of center of mass (CoM) in inertial frame, K_τ and K_F are the aerodynamic damping coefficients, and $R(\Theta)$ is the intrinsic rotation matrix from body frame to inertial frame. This model treats the UAV as a mass point and can control the total force F and the total torque τ_u. For UAV, the reference trajectory $r(t) = [x_r(t), y_r(t), z_r(t), \phi_r(t), \theta_r(t), \psi_r(t)]^T \in \mathbb{R}^6$ is in the task space, where the subscript r denotes the reference.

Remark 4: Since the UAV (quadrotor) has four rotors, we only have four control input. To simplify the model, the UAV dynamic model in (3) considers the actuator control input $u'(t) = [F, \tau_u^T]^T = [F, \tau_x, \tau_y, \tau_z]^T$ as the equivalent control input for the four rotors. Besides, since $u'(t) \in \mathbb{R}^4$ has two less degrees of freedom than $r(t) \in \mathbb{R}^6$, UAV is an underactuated system. This makes the two degrees of freedom in the reference trajectory of UAV cannot be assigned arbitrarily but be reversed from the dynamic equation in (3).

Now, suppose the UAV flying behavior occurs between time step t_1 and t_2, that is, $\beta_n = \mathit{flying}, n \in \mathbb{Z} \cap [t_1, t_2]$. The corresponding path $(\sigma_n), n \in \mathbb{Z} \cap [t_1, t_2]$ will be smoothed first by linear interpolation and then by cubic spline interpolation, which gives the smoothed path $(\sigma'_n), n \in \mathbb{Z} \cap [t_1, t_1 + D(t_2 - t_1)], \sigma'_n \in \mathbb{R}^3$, where $D \in \mathbb{Z}^+$ is the interpolation density. Then, (σ'_n) will be the position reference path $[x_r[k], y_r[k], z_r[k]]^T$. Subsequently, the orientation reference path, row angle $\phi_r[k]$, pitch angle $\theta_r[k]$ and yaw angle $\psi_r[k]$ are all considered. $\phi_r[k]$ is set to zero since no need for spinning when flying. $\theta_r[k]$ and $\psi_r[k]$ cannot be set beforehand since UAV is an underactuated system, which will be discussed in the next section. Finally, the reference path $r[k]$ of UAV can be obtained by combining them together, i.e., $r[k] = [x_r[k], y_r[k], z_r[k], \psi_r[k]]^T \in \mathbb{R}^4$. The flowchart is shown in Fig. 4.

B. MOTION OF WALKING OF ROBOT

By Lagrange equation, the dynamic model of a biped robot in URTS can be formulated as:

$$\tau_R = M_R(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) \tag{4}$$

where τ_R is the total torque on revolute joints, $q, \dot{q}, \ddot{q} \in \mathbb{R}^{12}$ are angular position, angular velocity, and angular acceleration vector of revolute joints, $M_R(q) \in \mathbb{R}^{12\times12}$ is the inertia matrix, $C(q, \dot{q}) \in \mathbb{R}^{12}$ is the Coriolis and centripetal force vector and $G(q) \in \mathbb{R}^{12}$ is the gravitational force vector. The detailed kinematic and dynamic parameters can be found in the online source [20]. For biped robot, the reference trajectory $r(t) = q_r(t) \in \mathbb{R}^{12}$ is in the joint space. Furthermore, the walking of biped robot suffers from the falling problem, i.e, how to find a stable walking pattern to prevent robot from falling. These make the design of walking motion more difficult. In this paper, a three-dimensional linear inverted pendulum Model (3D-LIPM) [21] is used to design the walking motion of each biped robot in URTS.

With 3D-LIPM, we can significantly reduce the amount of computation. Let us define the body frame of biped robot as $\{\widehat{X_b}, \widehat{Y_b}, \widehat{Z_b}\}$ as in [20]. Taking the forward direction of biped robot as $\widehat{X_b}$ direction, the left direction as $\widehat{Y_b}$ direction, and the torso direction as $\widehat{Z_b}$ direction in body frame, "Falling" means the moments on the robot in $\widehat{X_b}$ and $\widehat{Y_b}$ direction are not zero. More accuately, the biped robot will not fall if the zero moment point (ZMP) lies in the support polygon, i.e., the convex hull of face of supported foots. The ZMP in $\widehat{X_b}$ direction can be discribed as [22] (The ZMP in the $\widehat{Y_b}$ direction is the same form):

$$x_{zmp} = \frac{\sum_{i=1}^{12} (m_i(\ddot{z}_i + g)x_i - m_i\ddot{x}_i z_i - I_{iy}\ddot{\Omega}_{iy})}{\sum_{i=1}^{12} m_i(\ddot{z}_i + g)} \tag{5}$$

where m_i is the CoM, x_i, z_i are the linear position components, I_{iy} is the inertial component, and $\ddot{\Omega}_{iy}$ is the angular acceleration component of link i. However, it is difficult to directly calculate the analytical solution of $q_r(t)$ through (5). Since there exists a complex coordinate transformation between $q_r(t)$ and $x_i, z_i, \ddot{\Omega}_{iy}$. At the same time, it is necessary to ensure that x_{zmp} falls in the support polygon which also has a relationship with $q_r(t)$. To simplify this complex problem, an approximate solution can be derived through 3D-LIPM. We find CoM reference first and then obtain $q_r(t)$ by using inverse kinematic (IK) with given step size, step height, step period and CoM height. Many researchers have used this method to avoid complex calculations for ZMP of the actual robot dynamic model. Although there exists a model error between the actual dynamic model and 3D-LIPM, the design process will be more simple. The overall process is shown in Fig. 4.

Following the same step in UAV, the smoothed path (σ'_n) for biped robot can be obtained at first. For the convenience of explanation, suppose walking is occured between time step 1 and N, i.e., $n \in \mathbb{Z} \cap [1, N]$. Note that (σ'_n) is not actual CoM reference in robot case since CoM of robot need to "swinging" for balance. Despite of that, (σ'_n) tells the biped robot the position to go so the $\widehat{X_b}$ direction can be obtained by doing finite difference on (σ'_n) due to the expectation that the biped robot will move forward (rather than sideways or backward). To keep torso upright, the $\widehat{Z_b}$ direction is equal to the z-axis in the inertial frame $\widehat{Z_g}$. Given $\widehat{X_b}$ and $\widehat{Z_b}$, $\widehat{Y_b}$ can be obtained obviously through cross product. The sequence of body frame, i.e., CoM orientation path (CO_n) can be obtained through the above steps.

Remark 5: A frame in $\mathbb{R}^3$ can be determined by giving the "position" and "orientation" with respect to a reference frame. That is, given the position and orientation of two joints in a link with known kinematics, the position and orientation of joints between them can be found by IK. *Hence, we need to find the position path and orientation path, which compose the desired path.*

Let us denote the x and y component of σ'_n in (σ'_n) as the sequence $(\sigma^1_n), \sigma^1_n \in \mathbb{R}^2$. The left and right "envelopes", (σ^2_n) and (σ^3_n), of (σ^1_n) with a fixed distance L_1 can be found by (σ^1_n) and (CO_n) through the geometric relation among $(\sigma^i_n), i = 1,2,3$, where L_1 is the feet width (or shoulder width). Then the x and y component of the left and right foothold paths, $(\sigma^2_{k_n})$ and $(\sigma^3_{k_n})$, can be obtained by a given step size, which are the subsequence of (σ^2_n) and (σ^3_n), respectively. Finally, the left and right foothold paths $(\sigma^i_{k_n}), \sigma^i_{k_n} \in \mathbb{R}^3, i = 4,5$ are found by adding the z component which is given by ground height.

After foothold paths are obtained, ankle position path can also be obtained by the given step height which is customized by the designer or based on the height of the obstacle to be crossed. Taking the left foothold path as an example, x and y component of the highest position of ankle during stride are set as the middle point of two footholds $\sigma^2_{k_m}$ and $\sigma^2_{k_{m+1}}$ where $m \in \mathbb{Z} \cap [1, N-1]$, and the z component is given by the step height. By using the cubic spline interpolation, we have the left and right ankle position paths, (σ^6_n) and (σ^7_n). The ankle orientation path is found by the gradient of ground. Finally, the left and right ankle paths, (σ^8_n) and (σ^9_n), are found by combining the position and orientation path together. So far, the remaining work is to find out the CoM path and then to combine with the ankle path to calculate the joint path through IK.

To obtain the CoM postion path (CP_n), ZMP path needs to be obtained first. ZMP path can be obtained through foothold paths $(\sigma^4_{k_n})$ and $(\sigma^5_{k_n})$ since ZMP needs to lie in the support face and the foothold path points out when the feet are on the ground. Suppose the CoM height z_c of biped robot is kept constant when walking, then the biped robot model can be regarded as an 3D-LIPM [21]:

$$\begin{aligned} \ddot{x}_c &= \frac{g}{z_c}(x_c - p_x) \\ \ddot{y}_c &= \frac{g}{z_c}(y_c - p_y) \end{aligned} \tag{6}$$

where (x_c, y_c, z_c) is the position of CoM of the inverted pendulum, g is the gravity acceleration, and (p_x, p_y) is the position of ZMP on the x-y plane. Since z_c, g and (p_x, p_y) are given, (x_c, y_c) can be solved. From the dynamic equations in the x and y directions in (6), it can be found that they are decoupled and thus can be calculated separately. Therefore, only the solution in the x direction is given below (the y direction as the same). To solve it, a method is proposed to convert it to a servo problem [23]:

$$\begin{aligned} \dot{\bar{x}}_c &= A\bar{x}_c + Bu \\ p_x &= C\bar{x}_c \end{aligned} \tag{7}$$

where $\bar{x}_c = \begin{bmatrix} x_c \\ \dot{x}_c \\ \ddot{x}_c \end{bmatrix}$, $A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$, and

$C = \begin{bmatrix} 1 & 0 & -z_c/g \end{bmatrix}$. Our goal is to find a control input u in order that the output p_x can track a ZMP reference trajectory so that the solution x_c of ODE in (6) can be obtained, i.e., the CoM position path can be found. Unlike conventional methods, the problem is solved by the optimal control. The system is discretized first and the discrete LQ optimal tracker is employed to achieve the output tracking. The formulation can be found in TABLE 4.4-1 in [24]. The CoM position path (CP_n) can then be obtained by combining x_c and y_c with z_c.

By combining the CoM orientation path (CO_n) wtih the position path (CP_n), the CoM path can be obtained. Finally, the joint path, i.e., reference path $r[k] \in \mathbb{R}^{12}$ of robot can be found by solving IK.

IV. TRACKING CONTROL OF EACH AGENT IN HYBRID URTS

Before converting $r[k]$ to $r(t)$, we first convert the UAV dynamic system in (3) and robot dynamic system in (4) into a form called agent dynamic model in a hybrid team to analyze their team formation tracking control problems together. Through some appropriate variable transformations, we have:

$$M(x(t))\ddot{x}(t) + H(x(t), \dot{x}(t)) = u(t) \tag{8}$$

where $u(t) \in \mathbb{R}^n$ is the control input vector, $x(t) \in \mathbb{R}^n$ is the state vector, $M(x(t)) \in \mathbb{R}^{n\times n}$ is the inertia matrix, and $H(x(t), \dot{x}(t)) \in \mathbb{R}^n$ is the non-inertial force vector. The control law for an agent in URTS is given as:

$$u(t) = M(r(t))(\ddot{r}(t) + u_{fb}(t)) + H(r(t), \dot{r}(t)) \tag{9}$$

where $r(t) \in \mathbb{R}^n$ is the desired reference trajectory, $M(r(t))$, $\ddot{r}(t)$ and $H(r(t), \dot{r}(t))$ are the feedfoward control terms for canceling system nonlinearity, and $u_{fb}(t)$ is the feedback control law to be futher designed for improving system robustness. For each UAV in hybrid URTS, we have $u(t) = \begin{bmatrix} f_u \\ \tau_u \end{bmatrix}$, $x(t) = \begin{bmatrix} X \\ \Theta \end{bmatrix}$, $M(x(t)) = \begin{bmatrix} mI & 0 \\ 0 & J \end{bmatrix}$, $H(x(t), \dot{x}(t)) = \begin{bmatrix} 0 \\ \dot{\Theta} \times (J\dot{\Theta}) \end{bmatrix} + \begin{bmatrix} f_g \\ 0 \end{bmatrix} + \begin{bmatrix} K_F & 0 \\ 0 & K_\tau \end{bmatrix} \begin{bmatrix} \dot{X} \\ \dot{\Theta} \end{bmatrix}$, and $n = 6$ by (3). For each robot in hybrid URTS, we have $u(t) = \tau_R$, $x(t) = q$, $M(x(t)) = M_R(q)$, $H(x(t), \dot{x}(t)) = C(q, \dot{q})\dot{q} + G(q)$, and $n = 12$ by (4).

Remark 6: The difference between the control law $u(t)$ in 9 of this article and the traditional feedback linearization control is that we use the "feedward" linearization control, i.e., use $M(r(t))$ and $H(r(t), \dot{r}(t))$ instead of $M(x(t))$ and $H(x(t), \dot{x}(t))$. This is because the state $x(t)$ is assumed to be unavailable in this paper so $x(t)$ cannot be use in control law $u(t)$.

To complete the design of reference trajectory $r(t)$ of each agent, a D/A converter is used to transform the reference path $r[k]$ (output of local motion planning) into a continuous signal $r'(t)$ as shown in Fig. 5. For UAV, we get $r'(t) = [x_r, y_r, z_r, \psi_r]^T \in \mathbb{R}^4$. Besides, for an UAV in hybrid URTS, it can be seen that the control input $u(t) = [f_u^T, \tau_u^T]^T = [f_x, f_y, f_z, \tau_x, \tau_y, \tau_z]^T \in \mathbb{R}^6$ we design in (9) is different from the actuator control input $u'(t) = [F, \tau_x, \tau_y, \tau_z]^T \in \mathbb{R}^4$ for UAV actuator since UAV is an underactuated system. The two degrees of freedom we reserved in Section III-A, i.e., ϕ_r and θ_r, are just to solve this problem. By substituting $\Theta = \begin{bmatrix} \phi_r \\ \theta_r \\ \psi_r \end{bmatrix}$ into $\begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} = R(\Theta) \begin{bmatrix} 0 \\ 0 \\ F \end{bmatrix}$ from UAV dynacmis in (3), the 3 unknown variables F, ϕ_r and θ_r can be found from these 3 equations using inverse dynamic because f_x, f_y, f_z and ψ_r are given. Combining ϕ_r and θ_r with $r'(t)$, we obtain $r(t) = [x_r, y_r, z_r, \phi_r, \theta_r, \psi_r]^T$. For biped robot, we get $r'(t) = r(t) \in \mathbb{R}^{12}$ and $u'(t) = u(t) \in \mathbb{R}^{12}$ since biped robot is a fully actuated system. So far, the design of reference trajectory $r(t)$ of each agent in hybrid URTS is done. We define the process of converting $r'(t)$ and $u(t)$ into $r(t)$ and $u'(t)$ mentioned above as the reference generation block in Fig. 5.

Remark 7: Let $r_{i,j}(t)$ be the reference trajectory $r(t)$, $r_{i,j}[k]$ be the reference path $r[k]$, $(\beta_n)_{i,j}$ be the behavior sequence (β_n), $(\sigma_n)_{i,j}$ be the collision-free path (σ_n), $q_{start,i,j}$ be the current configuration q_{start}, and $q_{goal,i,j}$ be the goal configuration q_{goal} of the agent $\alpha_{i,j}$ in the hybrid URTS. As long as $\alpha_{i,j}$ can track $r_{i,j}(t)$, the hybrid URTS can work as we expect since the previous blocks, i.e, task allocation, path planning, behavior layer and local motion planning, have completed their respective responsibilities and found their corresponding values $q_{start,i,j}$, $q_{goal,i,j}$, $(\sigma_n)_{i,j}$, $(\beta_n)_{i,j}$ and $r_{i,j}[k]$. That is, $r_{i,j}(t)$ is the reference trajectory that can complete the specific task, follow the specific path and perform the specific behavior.

To make the model more realistic, the following external disturbances encountered in actual scenarios are considered:

1) For each agent, there exists coupling effect due to co-channel interference in communication between agents [25].
2) For each agent, there exists cyber-attack on communication network between agents and ground station.
3) For each agent, there exists sensor noise.
4) For each UAV, there exists wind disturbance [26].
5) For each robot, there exists ground reaction force [27].

Let $x_{i,j}(t), i = 1, 2, \ldots, N_T, j = 1, 2, \ldots, N_A$ denote the state vector of agents $\alpha_{i,j}$. The coupling disturbance $c(t)$ on each agent in hybrid URTS can be represented as

$$c(t) = \sum_{k=1, k\neq i}^{N_T} D_{i,1,k}(x_{i,1}(t))x_{k,1}(t) \tag{10}$$

for UAV $\alpha_{i,1}$ and

$$c(t) = \sum_{k=1, k\neq j'}^{N_A} D_{i,j',k}(x_{i,j'}(t))x_{i,k}(t) \tag{11}$$

for biped robot $\alpha_{i,j'}$ where $j' = 2, 3, ..., N_A$ [25]. Since the ground station is responsible for the calculation, the calculated control command in (9) will be transmitted to the agent through the network channel in hybrid URTS. Therefore,

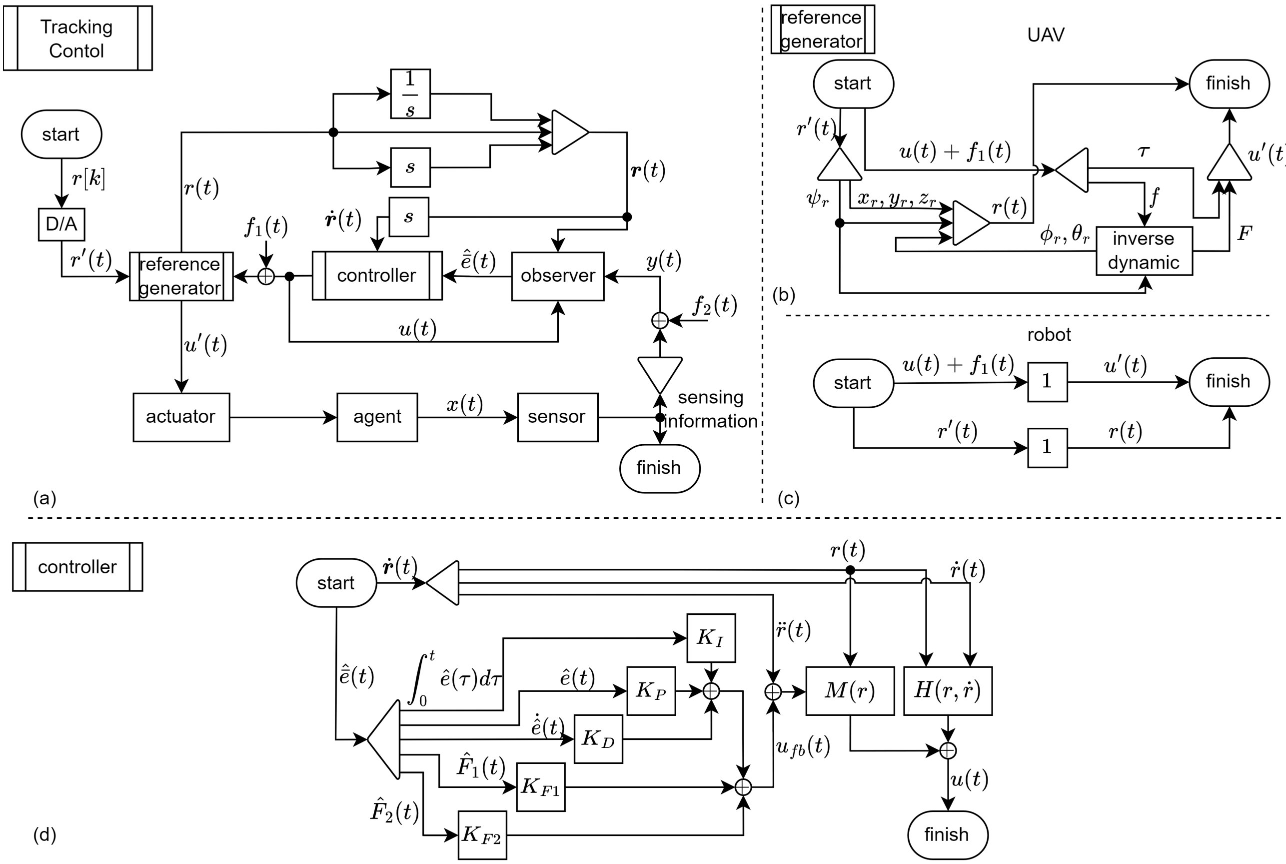

FIGURE 5: The flowchart of tracking control in the hybrid URTS. (a) The tracking control block of each agent in hybrid URTS. The controller block is predefined in (d). The controller block (the proposed general H_∞ decentralized observer-based feedforward FTC scheme) is designed for a fully actuated agent dynamic model in (8) while the UAV is an underactuated system. It makes the designed control law $u(t) \in \mathbb{R}^6$ and the actuator control input $u'(t) \in \mathbb{R}^4$ different for UAV. The reference generator block is introduced to deal with this problem. For UAVs, the reference generator block is predefined in (b). For robots, the reference generator block is predefined in (c). (b) The reference generator block for each UAV. $u'(t)$ and $r(t)$ can be calculated from $u(t)$ and $r'(t)$ by inverse dynamic through the UAV dynamic model in (3). (c) The reference generator block for each robot. $u'(t) = u(t)$ and $r(t) = r'(t)$ since the robot dynamic model in (4) is fully actuated. (d) The proposed general H_∞ decentralized observer-based feedforward FTC scheme for each agent in hybrid URTS.

the coupling effect due to co-channel interference and the cyber-attack signal will deteriorate the control command. In addition, the wind disturbance and the ground reaction force will apply extra force on an agent in (8). Therefore, through appropriate conversion, the above disturbances can be equivalent to a disturbance force $c(t) + d_1(t) \in \mathbb{R}^n$ where $d_1(t)$ is the non-coupling disturbance. The nominal system in (8) of an agent in hybrid URTS then be rewrited as the following real system:

$$M(x(t))\ddot{x}(t) + H(x(t), \dot{x}(t)) = u(t) + c(t) + d_1(t) \quad (12)$$

Now, substituting the control law $u(t)$ in (9) into (12) and subtracting $M(x(t))\ddot{r}(t)$ from the left and right sides, we have:

$$\begin{aligned} &M(x(t))(\ddot{x}(t) - \ddot{r}(t)) + H(x(t), \dot{x}(t)) \\ &= (M(r(t)) - M(x(t)))\ddot{r}(t) + M(r(t))u_{fb}(t) \\ &+ H(r(t), \dot{r}(t)) + c(t) + d_1(t) \end{aligned} \quad (13)$$

By Multipling $M(x(t))^{-1}$ from the left and right sides and with some arrangments, we have the tracking error differential equation as follow:

$$\ddot{e}(t) = u_{fb}(t) + f_1(t) \quad (14)$$

where $f_1(t) = M(x(t))^{-1}(-\Delta M(\ddot{r}(t) + u_{fb}(t)) - \Delta H + c(t) + d_1(t)) \in \mathbb{R}^n$ is considered as the actuator fault signal, $\Delta M \triangleq M(x(t)) - M(r(t))$ and $\Delta H \triangleq H(x(t), \dot{x}(t)) - H(r(t), \dot{r}(t))$ are the error terms from feedforward compensation, and $e(t) = x(t) - r(t)$ is the tracking error. Let us denote $\boldsymbol{e}(t) = \begin{bmatrix} \int_0^t e^T(\tau)d\tau & e^T(t) & \dot{e}^T(t) \end{bmatrix}^T \in \mathbb{R}^{3n}$, the tracking error differential equation in (14) can be rewrited as the following linear tracking error system:

$$\dot{\boldsymbol{e}}(t) = A\boldsymbol{e}(t) + B(u_{fb}(t) + f_1(t)) \quad (15)$$

where $A = A_0 \otimes I_n, B = B_0 \otimes I_n$ with $A_0 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, B_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$.

Through the above analysis, the tracking control problem of the nonlinear system of each agent in hybrid URTS with external disturbance in (12) is transformed into the regulation problem of the linear tracking error system in (15) with actuator fault signal $f_1(t)$ by the feedforward linearization control law $u(t)$. The remaining step is to design an appropriate feedback control law $u_{fb}(t)$ to make the linear tracking error system stable. In a real system, the feedback information is measured by sensor, i.e, the state $x(t)$ in (8) is unavailable. At the same time, the sensor fault on sensor also needs to be considered as mentioned before. Since the sensor information will transmitted back to the ground station for calculating control command through the network channel in URTS, not only the sensor noise but the cyber-attack signal are concerned. Let $\boldsymbol{x}(t) = \begin{bmatrix} \int_0^t x^T(\tau)d\tau & x^T(t) & \dot{x}^T(t) \end{bmatrix}^T$, the measurement output equation can be described as:

$$y(t) = C\boldsymbol{x}(t) + B_2 f_2(t) \tag{16}$$

where $y(t) \in \mathbb{R}^l$ is the output vector, $C \in \mathbb{R}^{l\times 3n}$ is the output matrix, $B_2 \in \mathbb{R}^{l\times o}$ is the input matrix of sensor fault signal $f_2(t) \in \mathbb{R}^o$. Let us define $\boldsymbol{r}(t) = \begin{bmatrix} \int_0^t r^T(\tau)d\tau & r^T(t) & \dot{r}^T(t) \end{bmatrix}^T$ to modify the output equation in (16) and combine it with (15), we have the following tracking error dynamic system of an agent in the hybrid URTS:

$$\begin{aligned} \dot{\boldsymbol{e}}(t) &= A\boldsymbol{e}(t) + B(u_{fb}(t) + f_1(t)) \\ y(t) &= C\boldsymbol{e}(t) + C\boldsymbol{r}(t) + B_2 f_2(t) \end{aligned} \tag{17}$$

To deal with the fault signals $f_i(t), i = 1, 2$, a smoothing signal model is introduced [28]:

$$\begin{aligned} \dot{F}_i(t) &= A_i F_i(t) + v_i(t) \\ f_i(t) &= C_i F_i(t) \end{aligned} \tag{18}$$

where $F_i(t) = \begin{bmatrix} f_i^T(t) & f_i^T(t-h) & \dots & f_i^T(t-wh) \end{bmatrix}^T \in \mathbb{R}^{(w_i+1)n_i}$, $A_i \in \mathbb{R}^{(w_i+1)n_i\times(w_i+1)n_i}$, $v_i(t)$ is the model error, $C_i = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix} \otimes I_{n_i}$, and w_i is the window size of smoothing signal model with $n_1 = n$ and $n_2 = o$. Substituting (18) into (17), we get the following augmented tracking error system of an agent in hybrid URTS:

$$\begin{aligned} \dot{\bar{e}}(t) &= \bar{A}\bar{e}(t) + \bar{B}u_{fb}(t) + \bar{v}(t) \\ y(t) &= \bar{C}\bar{e}(t) + C\boldsymbol{r}(t) \end{aligned} \tag{19}$$

where $\bar{e}(t) = \begin{bmatrix} \boldsymbol{e}(t) \\ F_1(t) \\ F_2(t) \end{bmatrix}$ is the augmented tracking error vector, $\bar{A} = \begin{bmatrix} A & BC_1 & 0 \\ 0 & A_1 & 0 \\ 0 & 0 & A_2 \end{bmatrix}$, $\bar{B} = \begin{bmatrix} B \\ 0 \\ 0 \end{bmatrix}$, $\bar{C} = \begin{bmatrix} C & 0 & B_2 C_2 \end{bmatrix}$, and $\bar{v}(t) = \begin{bmatrix} 0 \\ v_1(t) \\ v_2(t) \end{bmatrix}$. Since the fault signals become a state variable of the augmented tracking error system of an agent in (19), their corruption on the tracking error dynamic system in (17) can be avoided. A Luenberger observer is proposed to estimate them and origin state simutaniously to achieve an active FTC by the following estimation system:

$$\begin{aligned} \dot{\hat{e}}(t) &= \bar{A}\hat{e}(t) + \bar{B}u_{fb}(t) - L(y(t) - \hat{y}(t)) \\ \hat{y}(t) &= \bar{C}\hat{e}(t) + C\boldsymbol{r}(t) \end{aligned} \tag{20}$$

Assumption 4.1 ([28]): The augmented tracking error system (19) of an agent is observable, i.e., $rank \begin{bmatrix} zI - \bar{A} \\ \bar{C} \end{bmatrix} = 3n + (w_1 + 1)n + (w_2 + 1)o, \forall z \in eig(\bar{A})$.
The feedback control law $u_{fb}(t)$ of each agent in hybrid URTS then be designed as follow:

$$u_{fb}(t) = K\hat{e}(t) \tag{21}$$

where K is the control gain.
Remark 8: Let $u_{fb}(t) = K\hat{e}(t) = [K_I\ , K_P\ , K_D\ , K_{F_1}\ , K_{F_2}][\int_0^t \hat{e}^T(\tau)d\tau, \hat{e}^T(t), \dot{\hat{e}}^T(t), F_1^T(t), F_2^T(t)]^T$, we can find that the control gain K is composed of the PID control gains K_I, K_P and K_D for $\int_0^t \hat{e}(\tau)d\tau$, $\hat{e}(t)$ and $\dot{\hat{e}}(t)$, respectively, and the fault control gains K_{F_i} for $F_i(t)$ with $i = 1, 2$. This show that $u_{fb}(t)$ is a fault-tolerant PID controller.

Let us define the augmented estimation error $\tilde{e}(t) = \bar{e}(t) - \hat{e}(t)$, the augmented estimation error system can be obtained by (19) and (20):

$$\dot{\tilde{e}}(t) = \bar{A}\tilde{e}(t) + L\bar{C}\tilde{e}(t) = (\bar{A} + L\bar{C})\tilde{e}(t) \tag{22}$$

Combining (19), (21), and (22), we have the following augmented tracking and estimation error system of each agent in the hybrid URTS:

$$\dot{\tilde{x}}(t) = \tilde{A}\tilde{x}(t) + \tilde{v}(t) \tag{23}$$

where $\tilde{x}(t) = \begin{bmatrix} \bar{e}(t) \\ \tilde{e}(t) \end{bmatrix}$, $\tilde{A} = \begin{bmatrix} \bar{A} + \bar{B}K & -\bar{B}K \\ 0 & \bar{A} + L\bar{C} \end{bmatrix}$, and $\tilde{v}(t) = \begin{bmatrix} \bar{v}(t) \\ \bar{v}(t) \end{bmatrix}$

In order to enable the designed control gain K in (21) and observer gain L in (20) to achieve a specific performance for the augmented system in (23) under the disturbance $\bar{v}(t)$, the robust H_∞ decentralized observer-based tracking control strategy below a prescribed disturbance attenuation level ρ^2 for each agent in hybrid URTS is given as:

$$\frac{\int_0^{t_f} (\bar{e}^T(t)Q_1\bar{e}(t) + \tilde{e}^T(t)Q_2\tilde{e}(t) + u_{fb}^T(t)Ru_{fb}(t))dt - V(\tilde{x}(0))}{\int_0^{t_f} \tilde{v}^T(t)\tilde{v}(t)dt} \le \rho^2 \tag{24}$$

where t_f is the final time, $Q_1 \ge 0$ is the tracking error weighting matrix, $Q_2 \ge 0$ is the estimation error weighting matrix, $R > 0$ is the weighting matrix of control effort, $V(\tilde{x}(0))$ is the initial condition effect on the augmented system in (23), and $\tilde{v}(t)$ is the total disturbance needed to be attenuated. If we can find the control gain K and observer gain L such that (24) holds, then the effect of total disturbance $\tilde{v}(t)$ on augmented tracking error $\bar{e}(t)$ and augmented estimation error $\tilde{e}(t)$ can be attenuated to a prescribed level ρ^2 from the viewpoint of energy. Before analyzing the robust H_∞ decentralized observer-based tracking control problem of each agent in (24), the following lemmas are given:

Lemma 1 ([29]): For any matriices X and Y with appropriate dimensions, and matrix $R = R^T > 0$ the following inequality holds:

$$X^T Y + Y^T X \leq X^T R^{-1} X + Y^T R Y \tag{25}$$

Lemma 2 (Schur Complement [29]): For the matrices $X = X^T, Y = Y^T$ and matrix R with appropriate dimensions the following statement is true:

$$\begin{bmatrix} X & R \\ R^T & Y \end{bmatrix} > 0 \Leftrightarrow Y > 0, X - RY^{-1}R^T > 0 \tag{26}$$

Then, the following theorem is given.

Theorem 1: If there exists matrices $P = P^T > 0, K, L$ such that the following Riccati-like matrix inequality holds:

$$Q + P\tilde{A} + \tilde{A}^T P + \tilde{K}^T R \tilde{K} + \frac{1}{\rho^2} PP \leq 0 \tag{27}$$

where $\tilde{K} = \begin{bmatrix} K & -K \end{bmatrix}$, $Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix}$, then the H_∞ decentralized observer-based team formation tracking control strategy in (24) of each agent in the hybrid URTS can be achieved.

Remark 9: Since Riccati-like inequality of each agent in URTS has not involved the system information of other agents, therefore, the robust decentralized team formation tracking control can be achieved.

Proof. Choose the Lyapunov function $V(\tilde{x}(t)) = \tilde{x}^T(t) P \tilde{x}(t)$ for the augmented system (23) with $P = P^T > 0$, we have:

$$\begin{aligned}
&\int_0^{t_f} (\tilde{x}^T(t) Q \tilde{x}(t) + u_{fb}^T(t) R u_{fb}(t))dt \\
&= V(\tilde{x}(0)) - V(\tilde{x}(t_f)) + \int_0^{t_f} (\tilde{x}^T(t) Q \tilde{x}(t) \\
&+ u_{fb}^T(t) R u_{fb}(t) + \dot{V}(\tilde{x}(t)))dt \\
&\leq V(\tilde{x}(0)) + \int_0^{t_f} (\tilde{x}^T(t) Q \tilde{x}(t) + \\
&u_{fb}^T(t) R u_{fb}(t) + Sym(\dot{\tilde{x}}^T(t) P \tilde{x}(t)))dt
\end{aligned} \tag{28}$$

By (23) and Lemma 1, we have:

$$\begin{aligned}
&Sym(\dot{\tilde{x}}^T(t) P \tilde{x}(t)) \\
&= Sym((\tilde{A}\tilde{x}(t) + \tilde{v}(t))^T P \tilde{x}(t)) \\
&= \tilde{x}^T(t)(P\tilde{A} + \tilde{A}^T P + \frac{1}{\rho^2} PP)\tilde{x}(t) + \rho^2 \tilde{v}^T(t)\tilde{v}(t)
\end{aligned} \tag{29}$$

Substituting (21), (29) and $\tilde{x}^T(t) Q \tilde{x}(t) = \bar{e}^T(t) Q_1 \bar{e}(t) + \tilde{e}^T(t) Q_2 \tilde{e}(t)$ into (28), we get:

$$\begin{aligned}
&\int_0^{t_f} (\bar{e}^T(t) Q_1 \bar{e}(t) + \tilde{e}^T(t) Q_2 \tilde{e}(t))dt + u_{fb}^T(t) R u_{fb}(t))dt \\
&\leq V(\tilde{x}(0)) + \int_0^{t_f} (\tilde{x}^T(t)(Q + P\tilde{A} + \tilde{A}^T P + \tilde{K}^T R \tilde{K} \\
&+ \frac{1}{\rho^2} PP)\tilde{x}(t) + \rho^2 \tilde{v}^T(t)\tilde{v}(t))dt
\end{aligned}$$

Thus, if (27) holds then (24) holds ■

Although the sufficient condition (27) for the existence of the H_∞ decentralized observer-based tracking control strategy (24) have been found, it can not be solved easily since it is a bilinear matrix inequality (BMI) and exists strong coupling between the designed variables K and L. To solve the issue, a two-step design procedure is exploited.

Step 1: First, let the Lyapunov function of augmented system (23) be the sum of two Lyapunov function of subsystems (19) and (22), i.e., $V(\tilde{x}(t)) = \tilde{x}^T(t) P \tilde{x}(t) = \bar{e}^T(t) P_1 \bar{e}(t) + \tilde{e}^T(t) P_2 \tilde{e}(t)$. Substituting $P = \begin{bmatrix} P_1 & 0 \\ 0 & P_2 \end{bmatrix}$ and $Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix}$ into (27), we get:

$$\begin{aligned}
&\begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} + Sym(\begin{bmatrix} P_1 & 0 \\ 0 & P_2 \end{bmatrix} \begin{bmatrix} \bar{A} + \bar{B}K & -\bar{B}K \\ 0 & \bar{A} + L\bar{C} \end{bmatrix}) \\
&+ \begin{bmatrix} K^T R K & -K^T R K \\ -K^T R K & K^T R K \end{bmatrix} + \frac{1}{\rho^2} \begin{bmatrix} P_1 P_1 & 0 \\ 0 & P_2 P_2 \end{bmatrix} \\
&= \begin{bmatrix} M_{11} & -P_1 \bar{B} K - K^T R K \\ * & M_{22} \end{bmatrix} < 0
\end{aligned} \tag{30}$$

where $M_{11} = Q_1 + Sym(P_1(\bar{A} + \bar{B}K)) + K^T R K + \frac{1}{\rho^2} P_1 P_1$, $M_{22} = Q_2 + Sym(P_2(\bar{A} + L\bar{C})) + K^T R K + \frac{1}{\rho^2} P_2 P_2$. By the fact that $\begin{bmatrix} M_{11} & -P_1 \bar{B} K - K^T R K \\ * & M_{22} \end{bmatrix} < 0 \Rightarrow M_{11} < 0, M_{22} < 0$, the inequality $M_{11} < 0$ is used to find P_1, K. Premultiplying and postmultiplying $M_{11} < 0$ by $W_1 = P_1^{-1}$ and applying Lemma 2, we obtain:

$$\begin{bmatrix} Sym(\bar{A} W_1 + \bar{B} Y_1) + \frac{1}{\rho^2} & W_1 \sqrt[1/2]{Q_1} & Y_1^T \\ * & -I & 0 \\ * & * & -R^{-1} \end{bmatrix} < 0 \tag{31}$$

where $Y_1 = K W_1$. By solving the LMI in (31), we can obtain W_1, Y_1.

Step 2: Substituting $P_1 = W_1^{-1}$ and $K = Y_1 W_1^{-1}$ found in *Step 1* into (30) and applying Lemma 2, we obtain:

$$\begin{bmatrix} M_{11} & -P_1 \bar{B} K - K^T R K & P_2 \\ * & Q_2 + Sym(P_2 \bar{A} + Y_2 \bar{C}) + K^T R K & 0 \\ * & * & -\rho^2 I \end{bmatrix} < 0 \tag{32}$$

where $Y_2 = P_2 L$. By solving the LMI (32), we can obtain P_2, Y_2.

If we want to find the optimal H_∞ decentralized observer-based tracking control strategy for the augmented tracking and estimation error system in (23) of each agent in hybrid URTS , we need to solve the following LMIs-constrained optimization problem:

$$\begin{aligned} &\rho^{*2} = \min_{P,K,L} \rho^2 \\ &s.t.(31),(32) \end{aligned} \tag{33}$$

The design procedure of the optimal decentralized H_∞ observer-based feedforward FTC scheme for each agent in (12) is summarized as follows:

1) Apply the feedforward control in (9) to obtain the linearized tracking error dynamic system in (17) for each agent in hybrid URTS.
2) Construct the smoothing signal models (18) for the actuator fault $f_1(t)$ and sensor fault $f_2(t)$. Embed these smoothing signal models into the linearized system (17) to get the augmented tracking error system of each agent in (19).
3) Construct the robust FTC law in (21) and the augmented estimation error system (22) to obtain the augmented tracking and estimation error system of each agent in (23)
4) Solve the LMIs-constrained optimization problem (33) by the two-step design procedure to obtain the control gain K and observer gain $L = P_2^{-1}Y_2$ for each agent in the hybrid URTS.

The overall flowchart of tracking control of each agent in hybrid URTS is shown in Fig. 5. The reference generator is used to compute the desired reference trajectory $r(t)$ and actuator control input $u'(t)$ for each agent according to the continuous signal $r'(t)$ obtained by D/A and the control law $u(t)$ we design in (9). Passing $r(t)$ through the integrator and differentiator, we get $\boldsymbol{r}(t)$. $\boldsymbol{r}(t)$ is then passed to observer to calculate the error $\boldsymbol{e}(t)$. Its differential, $\dot{\boldsymbol{r}}(t)$, is then inputted to controller for feedforward control. The sensor measures not only the agent's own information (e.g., position or velocity) but also environmental information. The former, measurement output $y(t)$, is passed to observer to get the estimation $\hat{\bar{e}}(t)$ for feedback control. The letter is passed back to the high-level block for positioning, mapping and object recognition.

Remark 10: By the proposed agent dynamics model in (8) and the introduction of reference generator block in Fig. 5 (b), a general H_∞ decentralized observer-based feedforward FTC scheme for each agent $\alpha_{i,j}$ in hybrid URTS can be designed as shown in the controller block in Fig. 5 (c). The decentralized architecture also ensures the scalability of URTS scale. More specifically, let us introduce subscripts i and j to the corresponding variables of each agent $\alpha_{i,j}, i = 1,2,...,N_T, j = 1,2,...,N_A$ (e.g., the state $x_{i,j}(t)$, the reference trajectory $r_{i,j}(t)$, the control gain $K_{i,j}$, etc.). It can be seen that the number of teams $N_T > 0$ and the number of agents in a team $N_A > 0$ are scalable.

Although the control gain in (21) and observer gain in (19) for each agent in URTS can already be found through the previous steps, the calculation speed of solving the matrix inequality (27) and the online calculation speed of controller and observer can be further improved by reducing the dimensionality. Observing the matrices A, B, C, B_2 in the linearized system (17), it can be further split into n subsystems for each agent ($n = 6$ for UAV and $n = 12$ for robot) if the matrices C, B_2 in the output equation (16) have the same form to A, B and $l = l_0 n, o = o_0 n$, i.e., $C = C_0 \otimes I_n, B_2 = B_{2,0} \otimes I_n$ where $C_0 \in \mathbb{R}^{l_0\times 3}, B_{2,0} \in \mathbb{R}^{l_0\times o_0}$. Let us decompose the error $\boldsymbol{e}(t) = \sum_{i=1}^n \boldsymbol{e}_i(t) \otimes \mathbf{e}_i$, the control $u_{fb,i}(t) = \sum_{i=1}^n u_{fb,i}(t) \otimes \mathbf{e}_i$, the acuaor fault $f_{1,i}(t) = \sum_{i=1}^n f_{1,i}(t) \otimes \mathbf{e}_i$, the output $y(t) = \sum_{i=1}^n y_i(t) \otimes \mathbf{e}_i$, and the sensor fault $f_{2,i}(t) = \sum_{i=1}^n f_{2,i}(t) \otimes \mathbf{e}_i$ where $\boldsymbol{e}_i(t) \in \mathbb{R}^3$, $u_{fb,i}(t) \in \mathbb{R}$, $f_{1,i}(t) \in \mathbb{R}^1$, $y_i(t) \in \mathbb{R}^{l_0}$, $f_{2,i}(t) \in \mathbb{R}^{o_0}$ and $\mathbf{e}_i$ is standard unit column vectors in $\mathbb{R}^n$, we get the n subsystems:

$$\begin{aligned} \dot{\boldsymbol{e}}_i(t) &= A_0\boldsymbol{e}_i(t) + B_0(u_{fb,i}(t) + f_{1,i}(t)) \\ y_i(t) &= C_0\boldsymbol{e}_i(t) + B_{2,0} f_{2,i}(t) \end{aligned} \tag{34}$$

where $i = 1,2,...,n$.

Remark 11: If the linearized system (17) can be splited into n subsystems, this means that the error $\boldsymbol{e}_i(t)$ of each state variable $x(t)$ of each agent in (8), can be measured independently via sensors to obtain the independent outputs $y_i(t)$. In actual systems, this is usually done.

By Theorem 1 again, the form of subsystems in (34) shows that we can find the control gain $K_i \in \mathbb{R}^{1\times s}$ and observer gain $L_i \in \mathbb{R}^{s\times l_0}$ of the ith subsystem (34) that achieve the decentralized H_∞ observer-based feedforward FTC performance with a prescribed attenuation level ρ_i, where $s = 3 + (w_1 + 1) + (w_2 + 1)o_0$. The origin control gain K of the origin agent system can be reconstructed by $K = \begin{bmatrix} k_1 & k_2 & \dots & k_n \end{bmatrix}^T, k_i = K_i^T \otimes \mathbf{e}_i$. The origin observer gain L can be reconstructed in the same way.

In this case, the calculation speed of finding gains K, L of each agent can be improved since the dimensionality is decrease. Furthermore, the online calculation speed of controller and observer can be also improved since there are more zeros in the gains K, L found by this method while maintaining estimation and tracking robustness. More clearly, the number of elements in matrix K, i.e., the number of scalar gains, changes from $n \times sn$ to $n(1 \times s)$. For L, it changes from $sn \times l_0 n$ to $n(s \times l_0)$. The number of scalar gains to be designed between them is n times different.

V. SIMULATION RESULTS

In this section, a specific S&R procedure for URTS is given to illustrate the proposed URTS system architecture and demonstrate the effectiveness of motion planning and control strategy of a hybrid UAVs and biped robots team system. First, a S&R area divided into N_T areas, $area_i, i = 1,2,...,N_T$, is given as shown in Fig. 6. To simplify the description, we will focus on the UAV and robot in ith team. Suppose each team has 5 agents, i.e., $N_A = 5$, then we can denote ith team as a set, $team_i = \{\alpha_{i,j} | i = 1,2,...,5\}$.

At the beginning, the task allocation block will assign the agents in $team_i$ with some search tasks in $area_i$ to build the occupancy map and find goals. The search task is assumed to be obtained by dividing the unsearched region as shown in Fig. 7. Representing the search tasks as a set, $task_1 = \{T_j | j = 1, 2, ..., 5\}$, then the proper agent-task pairs can be obtained through the task allocation block, $allocation_i = \{(\alpha_{i,j}, T_j) | j = 1, 2, ..., 5\}$. Suppose a goal is found after a while as shown in Fig. 7. At this point, we have a rescue task T_6. The new task list $task_2 = task_1 \cup \{T_6\}$ is obtained by updating the old one. If the ground station assigns $\alpha_{i,5}$ and $\alpha_{i+1,2}$ to perform T_6 through the task allocation algorithm, then we have the new allocation $allocation_2 = (allocation_1 - \{(\alpha_{i,5}, T_2), (\alpha_{i+1,2}, T_3)\}) \cup \{(\alpha_{i,5}, T_6), (\alpha_{i+1,2}, T_6)\}$. Until the S&R mission is over, the task allocation block will continuously work in the similar way.

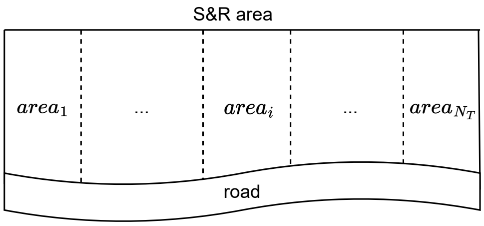

FIGURE 6: An example of a S&R area in URTS. This area is divided into N_T areas, and $team_i$ is responsible for $area_i$

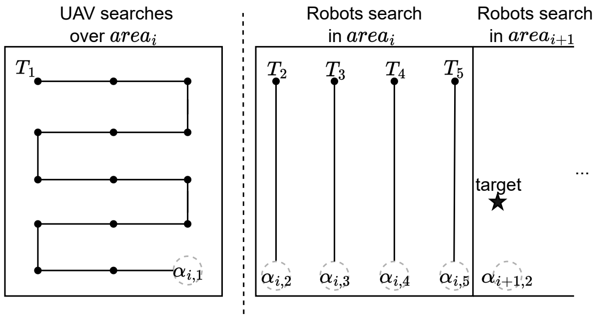

FIGURE 7: The search tasks in the ith team at the begining. The search tasks $T_j, j = 1, 2, ..., 5$ are to reach some consecutive destinations q_{goal} (black dots in figure) obtained from task allocation block. For robots, q_{goal} will be passed to path planning block to find collision-free paths (σ_n). For UAV, the sequence formed by q_{goal} is directly the path (σ_n) due to the no-collision assumption *Assumption 2.4*.

To explain the path planning block and behavior layer block, we choose the pairs $(\alpha_{i,5}, T_6)$ and $(\alpha_{i,1}, T_1)$ as example. For the UAV $\alpha_{i,1}$, the path $(\sigma_n), n \in \mathbb{Z} \cap [1, k_f], k_f = 9$ is directly assigned as shown in Fig. 9 without going through path planning by *Assumption 2.4*. The behavior sequence (β_n) is set as $\beta_n = \mathit{flying}, n \in \mathbb{Z} \cap [1, 9]$. For the robot $\alpha_{i,5}$, we have the goal configuration q_{goal} from the task T_6. With the current configuration q_{start} and configuration space $\mathcal{C}$ obtained by SLAM, the path $(\sigma_n), n \in \mathbb{Z} \cap [1, k_f], k_f = 27$ can be found as shown in Fig. 8. The behavior sequence (β_n) is set as $\beta_n = walking$ for $n \in \mathbb{Z} \cap [1, 5]$, $\beta_n = climbing$ for $n \in \mathbb{Z} \cap [6, 15]$ and $\beta_n = running$ for $n \in \mathbb{Z} \cap [16, 27]$. We choose the walking behavior to illustrate the local motion planning block of robots.

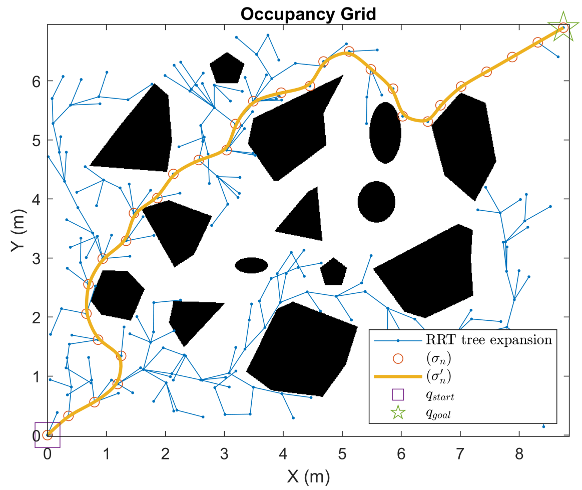

FIGURE 8: The path planning result of robot using RRT algorithm. The block polygons represent the obstacle space $\mathcal{C}_{obs}$.

After (β_n) is set, we can find the reference path $r[k]$ by local motion planning block. Following the procedure in Fig. 4, the results of local motion planning of UAV flying and robot walking are shown in Fig. 9 and 10, respectively.

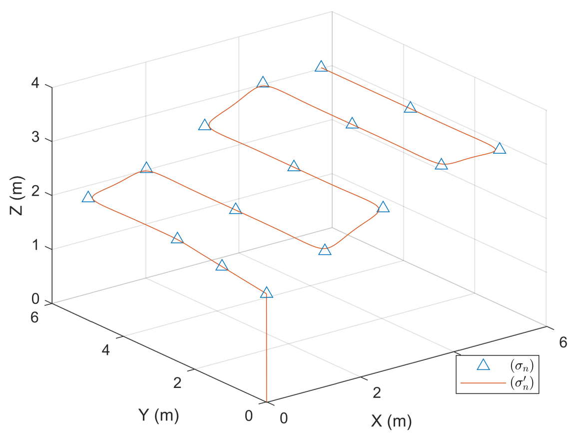

FIGURE 9: The result of local motion planning of UAV flying.

Through the previous steps and the help of reference generator block, the reference trajectory $r(t)$ of flying behavior for each UAV and walking behavior for each biped robot has been designed in URTS. The remaining parameters are set as follows:

Agents:

1) system parameters: Initial value $\tilde{x}(0) = [\boldsymbol{e}(0)^T, F_1(0)^T, F_2(0)^T, \tilde{e}(0)]^T = [[0.1X, ..., 0.1X]^T, 0, 0, 0]^T$ where $X \sim \mathcal{N}(0, 1)$. $C = C_0 \otimes I_n$ where $C_0 = I_3$.

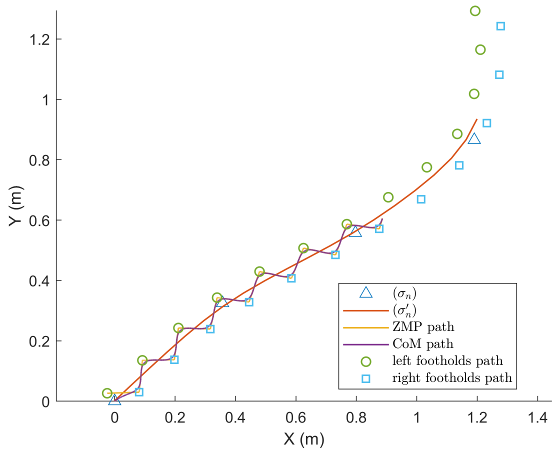

FIGURE 10: The top view of result of local motion planning of robot walking. The joint path, i.e., reference path $r[k]$ will be obtained by solving IK with CoM, left foothold and right foothold path.

2) designed parameters: $\rho^* = 30$. Actuator and sensor window size of smoothing signal model are $w_1 = 3$ and $w_2 = 4$, respectively.

UAVs:

1) system parameters: $B_2 = B_{2,0} \otimes I_n$ where $B_{2,0} = [0, 0.1, 1]^T$. $g = 9.81, m = 2, J_x = J_y = 1.25, Jz = 2.2, K_x = K_y = K_z = 0.01, K_\phi = K_\theta = K_\psi = 0.012$.
2) designed parameters: $Q_1 = 10diag(diag(1, 100, 10)\,, 0\,, 0)$, $Q_2 = diag(0.1diag([1\,, 100\,, 10]), diag(1, 0.1\,, 0.01), 20diag(1, 0.1, 0.01, 0.001))$, $R = 0.02$.
3) actuator coupling disturbance in (10): $c(t) = \sum_{k=1,k\neq i}^{N_T} D_{i,1,k}(x_{i,1}(t))x_{k,1}(t) \in \mathbb{R}^6$ where $D_{i,1,k}(x_{i,1}(t)) = diag(x_{i,1,1}(t)\,, \ldots\,, x_{i,1,6}(t))$ with $x_{i,1}(t) = [x_{i,1,1}(t)\,, \ldots\,, x_{i,1,6}(t)]^T$.
4) actuator non-coupling disturbance: $d_1(t) = [100\sin(3t)\,, \ldots\,, 100\sin(3t)]^T \in \mathbb{R}^6$.
5) sensor fault: $f_2(t)$ is set as a smoothed square wave as shown in Fig. 12.

Robots:

1) system parameters: $B_2 = B_{2,0} \otimes I_n$ where $B_{2,0} = [0, 0, 1]^T$. Remaining parameters can be found in Appendix E in [20].
2) designed parameters: $Q_1 = 50diag(diag(1, 100, 10), 0, 0)$, $Q_2 = 5diag(diag([1\,, 100\,, 10]), diag(1, 0.1\,, 0.01)\,, diag(1, 0.1, 0.01, 0.001))$, $R = 0.002$.
3) actuator coupling disturbance in (11): $c(t) = \sum_{k=1,k\neq j'}^{N_A} D_{i,j',k}(x_{i,j'}(t))x_{i,k}(t) \in \mathbb{R}^{12}$ where $D_{i,j',k}(x_{i,j'}(t)) = diag(x_{i,1,1}(t)\,, \ldots\,, x_{i,1,12}(t))$ with $x_{i,j'}(t) = [x_{i,j',1}(t)\,, \ldots\,, x_{i,j',12}(t)]^T$.
4) actuator non-coupling disturbance: $d_1(t) = [10\sin(3t)\,, \ldots\,, 10\sin(3t)]^T \in \mathbb{R}^{12}$.
5) sensor fault: $f_2(t)$ is set as a smoothed square wave as shown in Fig. 16.

The simulation results of tracking and estimation in tracking control block of the UAV $\alpha_{1,1}$ and the robot $\alpha_{1,2}$ in $team_1$ are given as follows:

UAV $\alpha_{1,1}$: The trajectories of reference, state and estimated state are shown in Fig. 11. The estimation of actuator fault $f_1(t)$ is shown in Fig. 12. The estimation of sensor fault $f_2(t)$ is shown in Fig. 13. The control effort is shown in Fig. 14.

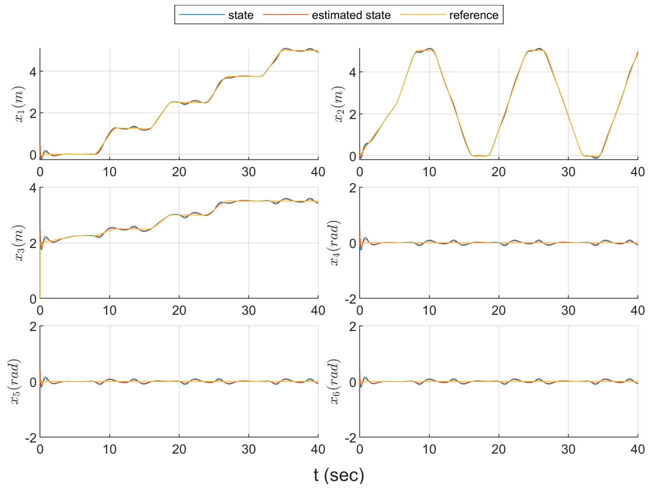

FIGURE 11: The trajectories of reference, state and estimated state of the UAV $\alpha_{1,1}$

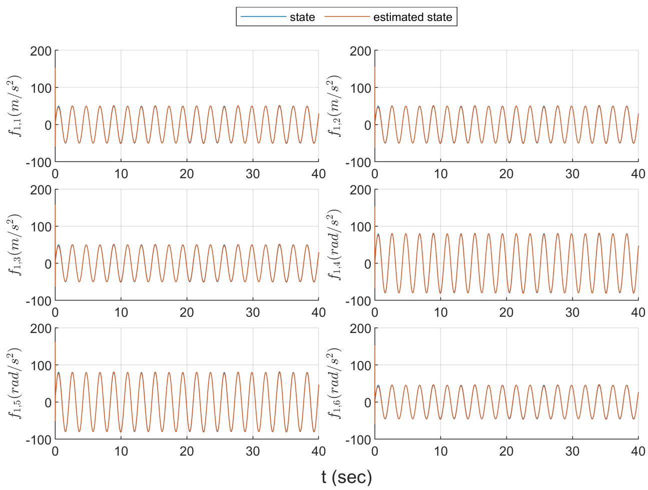

FIGURE 12: The estimation of actuator fault of the UAV $\alpha_{1,1}$

Robot $\alpha_{1,2}$: The trajectories of reference, state and estimated state are shown in Fig. 15. The estimation of actuator fault $f_1(t)$ is shown in Fig. 16. The estimation of sensor fault $f_2(t)$ is shown in Fig. 17. The control effort is shown in Fig. 18.

In Fig. 11, the tracking and estimation errors of UAV position and attitude reach the steady state all within 2 seconds. There is a brief jitter when the sensor fault signal changes drastically, but it returns quickly to the steady state. In Fig. 15, the tracking and estimation errors of robot joint

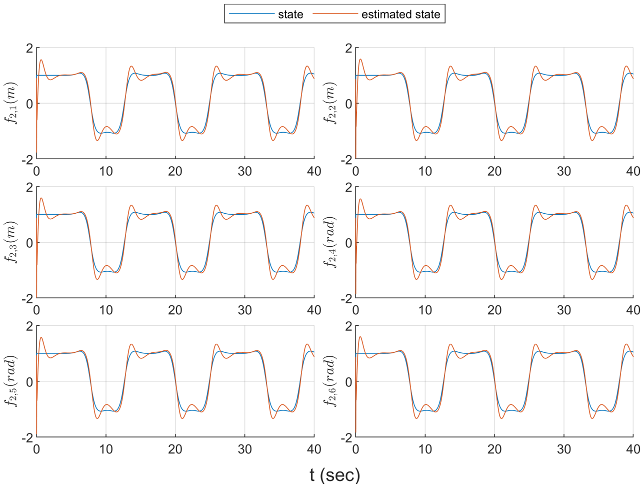

FIGURE 13: The estimation of sensor fault of the UAV $\alpha_{1,1}$

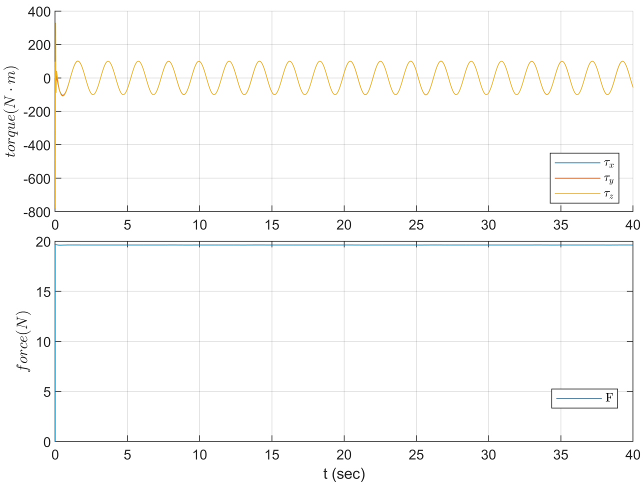

FIGURE 14: The control effort of the UAV $\alpha_{1,1}$

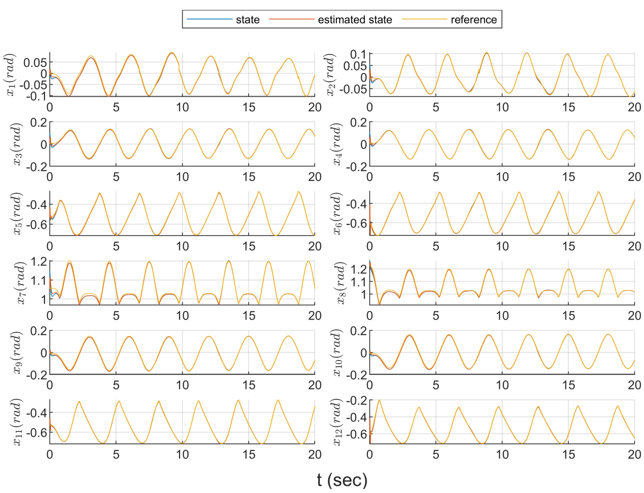

FIGURE 15: The trajectories of reference, state and estimated state of the robot $\alpha_{1,2}$

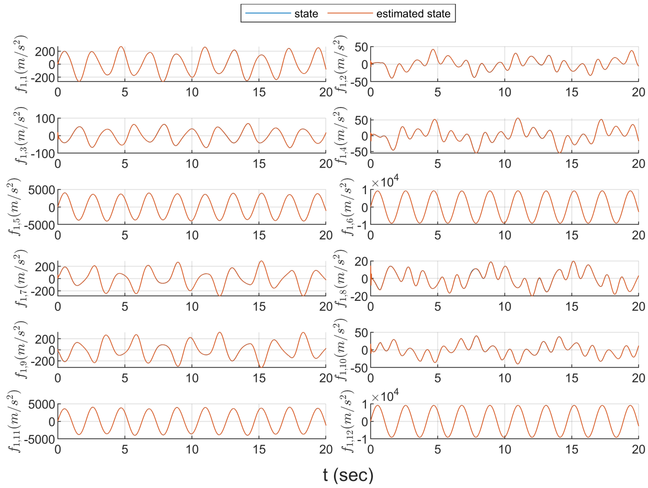

FIGURE 16: The estimation of actuator fault of the robot $\alpha_{1,2}$.

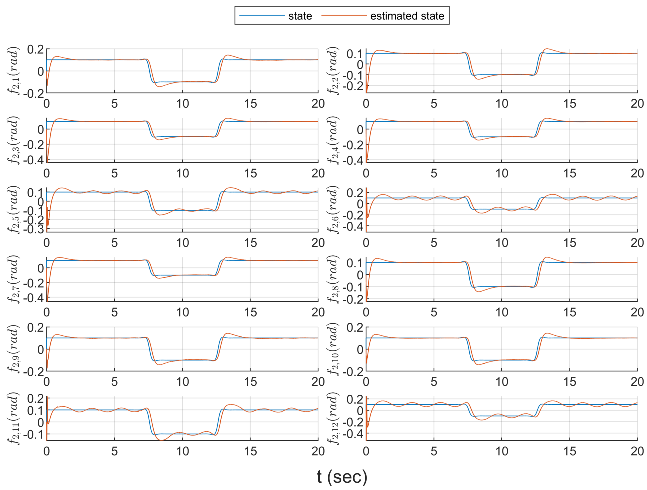

FIGURE 17: The estimation of sensor fault of the robot $\alpha_{1,2}$

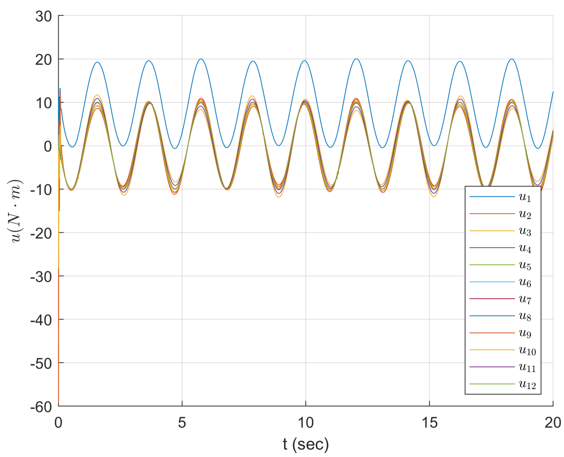

FIGURE 18: The control effort of the robot $\alpha_{1,2}$

angles immediately reach and maintain steady state under the influence of fault signals. In Figs. 12 and 16, the results show that the actuator fault, i.e., feedforward control errors and disturbances, can be effectively estimated. However, in Figs. 13 and 17, the estimation of sensor fault has an overshot phenomenon when there is a large change and returns to a steady state after about 2 seconds. In Figs. 14 and 18, the control efforts have high frequency and high amplitude at the initial instance due to the high gain characteristic of the robust control. After that, they maintain the sine wave shape to offset the estimated actuator fault value. Besides, it can be seen that the total force F of UAV remains constant against gravity in Fig. 14.

In order to show the effect of active FTC based on the proposed embed smoothing model method, a traditional PID computed torque controller without FTC is used for comparison [30]. In order to only focus on the effect of FTC, the method in [30] was revised to oberserver based, that is, the control law become $u(t) = M(r(t))(\ddot{r}(t) + u_{fb}(t)) + H(r(t), \dot{r}(t))$ where $u_{fb}(t) = [K_I, K_P, K_D][\int_0^t \hat{e}^T(\tau)d\tau, \hat{e}^T(t), \dot{\hat{e}}^T(t)]^T$. The results are shown in Fig. 19 for UAV and Fig. 20 for robot. From Fig. 19 and Fig. 20, it can be clearly seen that the influence of the acuator fault f_1 on the tracking error is revealed. The influence of the sensor fault f_2 is relatively insignificant because its value is much smaller than f_1 by our simulation setting. However, the fluctuation of f_2 with a period of 10 seconds (corresponding to the period of the smoothed square wave) can still be seen from the UAV attitude state variables x_4, x_5, x_6 in Fig. 19 and the robot joint state variables x_5, x_6, x_{11}, x_{12} in Fig. 20. Although it still stable, the tracking performance has dropped significantly compared to Fig. 11 and 15, respectively.

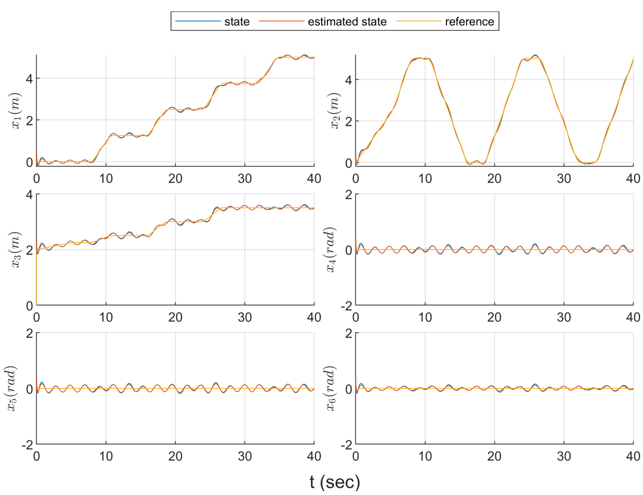

FIGURE 19: The trajectories of reference, state and estimated state of the UAV $\alpha_{1,1}$ by traditional PID computed torque controller without FTC [30]

The simulation of team tracking for $team_i, i = 1, 2, 3$ in URTS is shown in Fig. 21.

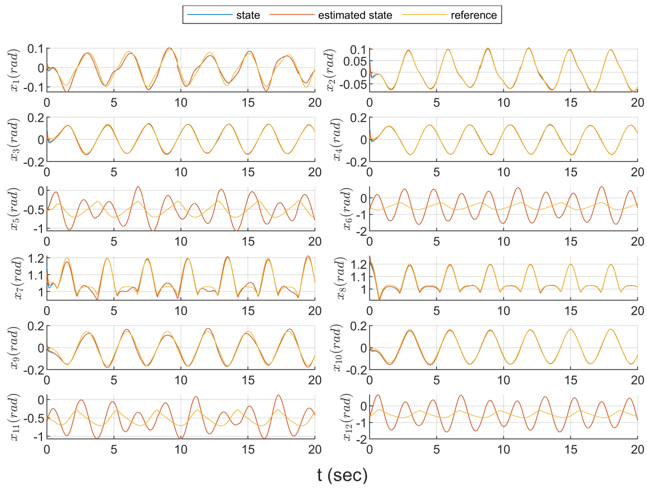

FIGURE 20: The trajectories of reference, state and estimated state of the robot $\alpha_{1,2}$ by traditional PID computed torque controller without FTC [30]

For further verification, we visualized the simulation of a hybrid team tracking in URTS and the configuration trajectory of biped robot on the online resource [31]. The results again demonstrate the effectiveness of the proposed H_∞ decentralized observer-based feedforward FTC method for agents in URTS.

VI. CONCLUSION

In this study, a system architecture of URTS is given for S&R usage. This gives a holistic view of the operational framework for URTS. By decomposing the path planning process into three subprocess, i.e., path planning, behavior layer, local motion planning, some common path planning algorithm can be applied in URTS. Next, we focus on the local motion planning of UAV flying and robot walking behavior. Besides, the bridging method between the reference path designed by local motion planning and the tracking control design is also given. By a general nonlinear agent dynamics model, the tracking control problem of UAV and robot can be analysis together. Through a feedforward strategy, the nonlinear tracking control problem with external disturbances is transformed to a regulation problem with fault signals. Then, a smoothing signal model is introduced to embed the fault signals into the state vector to avoid the corruption on the agent dynamic system. After that, a robust H_∞ decentralized observer-based feedforward FTC strategy is proposed for each agent in the hybrid URTS. To solve the robust H_∞ decentralized observer-based feedforward FTC problem, we transform it into a LMI-constrained optimization problem by a two-step design procedure, which can be effectively solved by MATLAB LMI Toolbox. A simulation example is given to illustrate more concretely how the proposed hybrid URTS architecture actually works. Finally, the effectiveness of the proposed robust H_∞ decentralized observer-based feedforward FTC method is also verified by the simulation

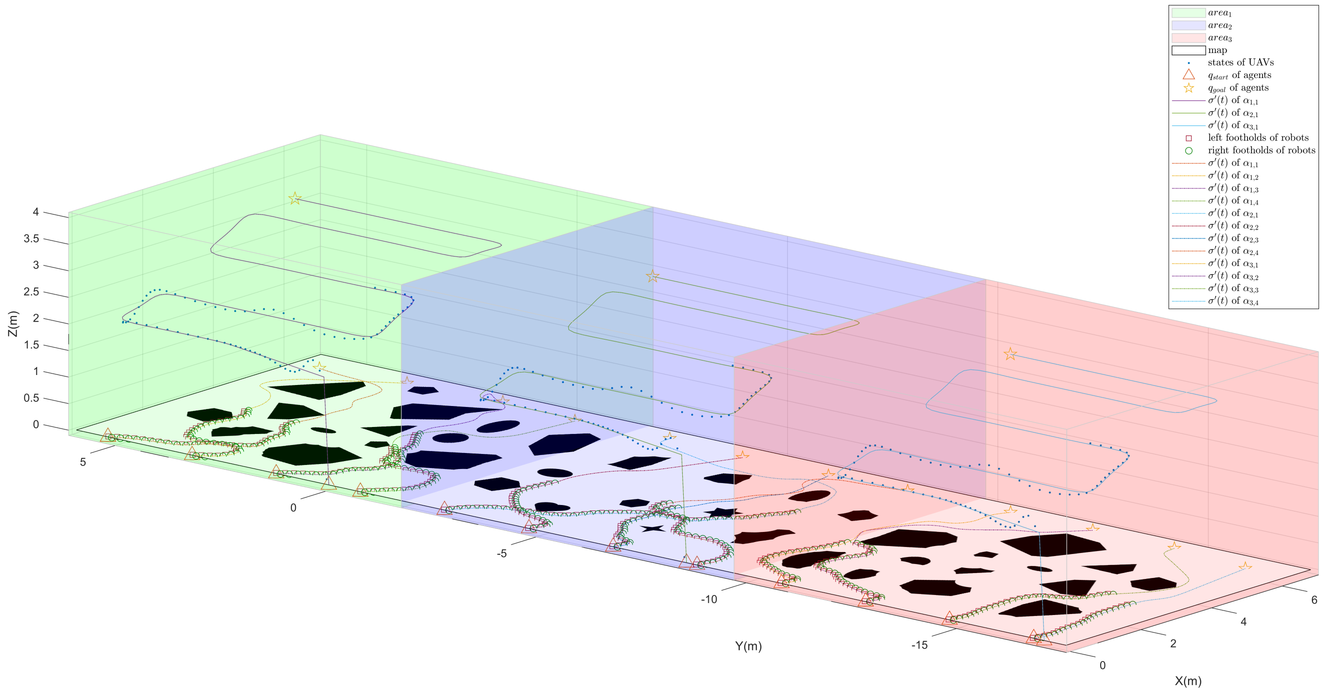

FIGURE 21: The simulation of URTS

results. In the future, we will...

REFERENCES

[1] L. Kloeker, T. Moers, L. Vater, A. Zlocki, and L. Eckstein, "Utilization and potentials of unmanned aerial vehicles (uavs) in the field of automated driving: A survey," *2021 5th International Conference on Vision, Image and Signal Processing (ICVISP)*, pp. 9–17, 2021.

[2] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," *IEEE Access*, vol. 6, pp. 28 573–28 593, 2018.

[3] F. Chen, W. Ren *et al.*, "On the control of multi-agent systems: A survey," *Foundations and Trends® in Systems and Control*, vol. 6, no. 4, pp. 339–499, 2019.

[4] W. Cheng, K. Zhang, B. Jiang, and S. X. Ding, "Fixed-time fault-tolerant formation control for heterogeneous multi-agent systems with parameter uncertainties and disturbances," *IEEE Trans. Circuits and Systems I: Regular Papers*, vol. 68, no. 5, pp. 2121–2133, 2021.

[5] G. M. Skaltsis, H.-S. Shin, and A. Tsourdos, "A survey of task allocation techniques in mas," *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 488–497, 2021.

[6] H. y. Zhang, W. m. Lin, and A. x. Chen, "Path planning for the mobile robot: A review," *Symmetry*, vol. 10, no. 10, p. 450, 2018.

[7] J. N. Yasin, S. A. S. Mohamed, M.-H. Haghbayan, J. Heikkonen, H. Tenhunen, and J. Plosila, "Unmanned aerial vehicles (uavs): Collision avoidance systems and approaches," *IEEE Access*, vol. 8, pp. 105 139–105 155, 2020.

[8] B. Arbanas, A. Ivanovic, M. Car, M. Orsag, T. Petrovic, and S. Bogdan, "Decentralized planning and control for uav–ugv cooperative teams," *Autonomous Robots*, vol. 42, no. 8, pp. 1601–1618, 2018.

[9] L. Li, Y. Miao, A. H. Qureshi, and M. C. Yip, "Mpc-mpnet: Model-predictive motion planning networks for fast, near-optimal planning under kinodynamic constraints," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4496–4503, 2021.

[10] T. Olcay and A. Özkurt, "Design and walking pattern generation of a biped robot," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 25, no. 2, pp. 761–769, 2017.

[11] Y. Sabri, N. El Kamoun, and F. Lakrami, "A survey: Centralized, decentralized, and distributed control scheme in smart grid systems," *2019 7th Mediterranean Congress of Telecommunications (CMT)*, pp. 1–11, 2019.

[12] C. Wang, J. Wang, P. Wu, and J. Gao, "Consensus problem and formation control for heterogeneous multi-agent systems with switching topologies," *Electronics*, vol. 11, no. 16, p. 2598, 2022.

[13] D. Rotondo, F. Nejjari, and V. Puig, "Passive and active ftc comparison for polytopic lpv systems," *2013 European Control Conference (ECC)*, pp. 2951–2956, 2013.

[14] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. intelligent vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[15] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," *Cooperative robots and sensor networks 2015*, pp. 31–51, 2015.

[16] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Ieee access*, vol. 2, pp. 56–77, 2014.

[17] Y. Liu and R. Bucknall, "A survey of formation control and motion planning of multiple unmanned vehicles," *Robotica*, vol. 36, no. 7, pp. 1019–1047, 2018.

[18] J. Yu and S. M. LaValle, "Multi-agent path planning and network flow," in *Algorithmic foundations of robotics X*. Springer, 2013, pp. 157–173.

[19] F. Sabatino, "Quadrotor control: modeling, nonlinearcontrol design, and simulation," 2015.

[20] B. S. Chen, Y. Y. Tsai, and M. Y. Lee. Supplementary file. hsinchu, taiwan. [Online]. Available: https://www.dropbox.com/s/qiv63y5hw7vrubp/SupplementFileRobot.pdf

[21] S. Kajita, O. Matsumoto, and M. Saigo, "Real-time 3d walking pattern generation for a biped robot with telescopic legs," *Proceedings 2001 ICRA. IEEE international conference on robotics and automation (Cat. no. 01ch37164)*, vol. 3, pp. 2299–2306, 2001.

[22] Q. Huang, K. Yokoi, S. Kajita, K. Kaneko, H. Arai, N. Koyachi, and K. Tanie, "Planning walking patterns for a biped robot," *IEEE Trans. robotics and automation*, vol. 17, no. 3, pp. 280–289, 2001.

[23] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 2, pp. 1620–1626 vol.2, 2003.

[24] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal control*. John Wiley & Sons, 2012.

[25] B. S. Chen and J. H. Lin, "Robust collaborative team formation control of hybrid teams of biped robots and wheeled vehicles under external disturbance and communication interactions," *IEEE Access*, vol. 10, pp. 77 633–77 657, 2022.

[26] J. Yang, C. Liu, M. Coombes, Y. Yan, and W.-H. Chen, "Optimal path following for small fixed-wing uavs under wind disturbances," *IEEE Trans. Control Systems Technology*, vol. 29, no. 3, pp. 996–1008, 2021.

[27] B. S. Chen, C. C. Wu, and Y. W. Chen, "Human walking gait with 11-dof humanoid robot through robust neural fuzzy networks tracking control." *International Journal of Fuzzy Systems*, vol. 15, no. 1, 2013.

[28] B. S. Chen, M. Y. Lee, W. Y. Chen, and W. Zhang, "Reverse-order multi-objective evolution algorithm for multi-objective observer-based fault-tolerant control of t-s fuzzy systems," *IEEE Access*, vol. 9, pp. 1556–1574, 2021.

[29] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear matrix inequalities in system and control theory*. SIAM, 1994.

[30] E. Rastogi and L. B. Prasad, "Comparative performance analysis of pd/pid computed torque control, filtered error approximation based control and nn control for a robot manipulator," in *2015 IEEE UP Section Conference on Electrical Computer and Electronics (UPCON)*, 2015, pp. 1–6.

[31] B. S. Chen and T. W. Hung. Hsinchu, taiwan. [Online]. Available: https://www.dropbox.com/scl/fo/5n6y25k3i8kjxsx9efxa3/h?dl=0&rlkey=x5p4stwtuf2gmch6l07etvz4b