Extensions of Receding Horizon Task Assignment for Area Coverage in Dynamic Environments

YOUKYUNG HONG
Electronics and Telecommunications Research Institute, Daejeon, South Korea

SUNGGOO JUNG
Electronics and Telecommunications Research Institute, Daejeon, South Korea
NASA Jet Propulsion Laboratory, Pasadena, CA USA

SUSEONG KIM
JIHUN CHA
Electronics and Telecommunications Research Institute, Daejeon, South Korea

This article presents a task reassignment strategy that can flexibly handle changes in dynamic environments. Most existing studies have only considered static environments where the mission is predetermined and unexpected events do not occur. However, in practice, the environment in which unmanned aerial vehicles perform their mission is complex and includes various pop-up events. Therefore, task reassignment is essential through reoptimization to adjust the previous plan, which is no longer optimal whenever dynamic events occur. The importance of this article is that the modified receding horizon task assignment algorithm is applied for accurate and efficient task reassignment. In addition, this article aims to provide a complete system architecture from operator input to motor commands for autonomous area coverage missions. Reliable performance validation is also performed through various simulations and a challenging outdoor flight experiment with real hardware implementation.

Manuscript received 27 April 2022; revised 9 September 2022; accepted 6 November 2022. Date of publication 21 November 2022; date of current version 9 June 2023.

DOI. No. 10.1109/TAES.2022.3222139

Refereeing of this contribution was handled by Gokhan Inalhan.

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korean government (MSIT) under Grant 2022-0-00021, and in part by the Development of Core Technologies for Autonomous Searching Drones.

Authors' addresses: Youkyung Hong, Suseong Kim, and Jihun Cha are with the Electronics and Telecommunications Research Institute, Daejeon 34129, South Korea, E-mail: (youkyh1@etri.re.kr; suseongkim@etri.re.kr; jihun@etri.re.kr); Sunggoo Jung was with ETRI, Daejeon, 34129, South Korea. He is now with NASA Jet Propulsion Laboratory, Pasadena, CA 91109, USA, E-mail: (sunggoo@etri.re.kr). *(Corresponding author: Sunggoo Jung.)*

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

I. INTRODUCTION

Unmanned aerial vehicles (UAVs) are expected to gradually replace most human labor, especially in repetitive and dangerous tasks, by minimizing human intervention and maximizing the autonomy of UAVs. UAV operations can be classified into several groups [1]: 1) area coverage missions where UAVs cover a certain area of interest with equipped sensors [2], [3]; 2) search missions where UAVs find latent targets [4]; 3) routing missions where UAVs visit a set of waypoints [5], [6]. Among them, area coverage missions can be applied to various applications such as building inspection, precision agriculture, aerial surveying, and natural resource management. Therefore, this study focuses on autonomous area coverage missions performed by multiple UAVs.

For autonomous area coverage missions, several technologies are required: 1) a human operator enters parameters, such as mission area and the number of UAVs, to define the mission; 2) a list of waypoints is generated to fill the given mission area; 3) the generated waypoints are optimally assigned to multiple UAVs; 4) each UAV accurately flies along with the assigned waypoints. Although all these technologies are essential, this study investigates how to generate and assign waypoints between multiple UAVs. The general descriptions of waypoint generation and task assignment technologies are as follows. First, to fill the area of interest with each UAV's footprint, the general strategy is to generate geometric patterns [7] such as the back-and-forth movement [4], [8], spiral pattern [9], and grid-based method [10]. Among them, the back-and-forth movement has been experimentally [11] and theoretically [1] proven to be the most energy-efficient. The area of interest can be transformed into a list of waypoints by applying the geometric pattern. Second, the task assignment problem can be designed to optimally assign the waypoints to UAVs based on the multiple traveling salesman problem (TSP). A considerable amount of literature has been published on TSP, ranging from mixed integer linear programming (MILP) obtaining exact solutions [8] to heuristics determining approximate solutions [5], [12]. Both strategies for finding optimal and suboptimal solutions are worthwhile. In addition, it is impossible to determine which strategy is better, and different applications require different strategies.

Most studies on autonomous area coverage missions that generate waypoints and optimally assign the waypoints between multiple UAVs have two common limitations. First, various UAV task assignment studies considered static environments where the mission is fixed in advance, and unexpected events do not occur [2], [3], [6]. In practice, however,

uncertainty comes from various sources such as weather conditions, communication availability, time-sensitive task, changes in the number of UAVs, and human factors. In static environments, task assignment is initially performed only once and does not change. On the other hand, in dynamic environments, additional strategies are required to cope with the above uncertainties. Therefore, this study performs a reoptimization that adjusts the previous plan, which is no longer optimal whenever two dynamic events occur. The first dynamic event is adding a new mission area during a mission. The second dynamic event is a change in the number of UAVs due to UAV failure or additional UAV participation. Second, previous research assumed that all UAVs participate in a mission with a fully charged battery and the mission duration is short enough not to consider recharging the battery of UAVs [13]. However, in practice, UAVs have different battery capacities depending on the battery type and the charge state. Also, a single battery cannot perform long-term or persistent missions. Therefore, in this study, UAVs with sufficient battery power are assigned more waypoints, and UAVs with insufficient battery power are assigned fewer waypoints. This study considers a situation in which a UAV returns to the base to recharge the battery when it can no longer perform its mission due to low battery status.

Furthermore, this study proposes a new task reassignment methodology based on receding horizon task assignment (RHTA), one of the heuristic optimization algorithms, to determine a good suboptimal solution within a reasonable computation time. As mentioned earlier, methodologies for obtaining optimal or suboptimal solutions depend on the application. As mentioned previously, optimization methodologies for determining optimal or suboptimal solutions depend on the application. For example, UAVs waiting on the ground before starting a mission can consume significant computation time to obtain the exact optimal solution. On the other hand, in the case of task reassignment, it is questionable whether it is necessary to find an exact optimal solution for an extended time from the point of view of UAVs hovering in the air. Therefore, in this study, the modified RHTA algorithm is proposed to balance the efficiency (i.e., the computational time) and the accuracy (i.e., the solution's optimality).

In summary, the contributions of this study are threefold.

1) This study presents a complete system architecture from operator input (to roughly define the mission) to motor commands (to move the UAV) to perform area coverage missions in a dynamic environment. Therefore, this study attempts to define the function of each subsystem and the shared information between each subsystem.
2) The modified RHTA algorithm is developed for fast and accurate reactions to deal with newly generated dynamic events. Especially, the modified RHTA algorithm enables the back-and-forth movement of UAVs and considers the maximum travel distance according to the remaining battery power.
3) Reliable performance validation is performed in this study, ranging from various simulations to the challenging outdoor flight experiment with the complete hexacopter testbeds. Note that this is the first study to conduct the flight experiment: 1) modifying the original mission in the field; 2) reading the battery level of the flying UAV in real-time; 3) replacing the low battery; 4) reintroducing the recharged UAV.

The rest of this article is organized as follows. Section II provides a literature review of task assignment methods for the autonomous mission of multiple UAVs. Section III explains the assumptions considered in this study and introduces the entire system's architecture. Section IV details the modified RHTA algorithm to optimally assign waypoints for the given area coverage problem. Section V presents various simulation results to verify the performance of the proposed system. Section VI provides flight experiment results in which two hexacopters perform the area coverage mission in a dynamic environment. Finally, Section VII concludes this article.

II. RELATED WORK

This section presents three discussions of recent developments related to this study. The first discussion is about previous studies considering unexpected events and uncertainties in the fields of task assignment for the autonomous mission of multiple UAVs. Unexpected events and uncertainties can be divided into three categories: 1) loss or addition of UAVs; 2) creation or deletion of new tasks over time; 3) fluctuations in task duration. Methodologies can be divided into two approaches: replanning when new information is obtained (i.e., postprocessing) and robust optimization assuming uncertainty as some standard distribution function (i.e., preprocessing). Amori et al. [14] addressed the task reassignment problem by applying swarm intelligence strategies in military operations where UAV losses or onboard sensor failure may occur. In their dynamic scenario, the number of UAVs was changed by randomly removing some UAVs until a single UAV remained. Tang et al. [15] proposed task reassignment based on fuzzy C-means clustering and ant colony algorithm to cope with three types of unexpected events. Unexpected events included the following: 1) when a new target is discovered; 2) when a target is removed due to a sudden change in weather; 3) when an initial assignment needs to be adjusted. On the other hand, Chen et al. [5] modified the two-part wolf pack search algorithm to consider the failure rates of UAVs in TSP. The failure rate of UAVs was considered a random variable with an approximate range rather than an exact probability distribution. Liu et al. [16] introduced robust task assignment based on the Markov decision process in the stochastic environment where the duration of task execution is uncertain due to load capacity. The task duration distribution was also assumed to follow the normal distribution, while the actual distribution is unknown and can be obtained from the analysis of historical data. In summary, previous studies dealing with uncertainty through

robust optimization do not consider the actual distribution, so approximation errors are unavoidable. Therefore, this study adopts a replanning strategy instead of robust optimization. However, unlike previous research, this study can respond to both types of dynamic events: the changes in the number of UAVs and the changes in tasks.

Second, several attempts have been made to consider the battery status of UAVs in the fields of task assignment for the autonomous mission of multiple UAVs. These efforts can be divided into two methodologies. The first approach is to accurately model battery consumption. Schacht et al. [17] estimated the flight endurance related to the battery's energy and power capacity to calculate the UAV's remaining flight time. Valenti et al. [18] presented a support vector regression model to estimate UAV's remaining flight time for multiple UAV mission systems. However, it is difficult to determine accurate battery consumption because it is affected by various factors such as weather conditions, battery life, and flight speed. On the other hand, the second approach is to consider the battery consumption of UAVs simply instead of determining the inaccurate battery consumption model. Ramasamy et al. [19] solved the routing problem of multiple fuel-constrained UAVs by recharging from a single unmanned ground vehicle to overcome the UAV's limited battery capacity. In their problem, it was considered that the battery level of UAV decreases proportional to the traveled distance. Hartuv et al. [20] considered a persistent surveillance mission where multiple UAVs must change their battery at a fixed set of refueling stations. In their mission, it was assumed that each UAV was initially fully charged, and then the battery level decreased linearly with time. Similarly, this study assumes that the battery consumption is proportional to the Euclidean distance between two waypoints. In contrast to the previous works, however, the current battery status (i.e., the remaining battery power) is obtained in real-time from the onboard sensor in this study. Therefore, even if the battery is not fully charged at the beginning or the battery is partially exhausted during the mission, the mission can be performed by limiting the travel distance of the UAV according to the remaining battery status.

Lastly, few studies have utilized the RHTA algorithm in the fields of task assignment for the autonomous mission of multiple UAVs. Alighanbari [21] proposed the RHTA algorithm for fast replanning to determine the sequence of waypoints each UAV should visit when one of the UAVs suddenly fails. Rodrigues et al. [22] extended Alighanbaris's study by excluding invalid waypoint combinations to reduce the computation time of the RHTA algorithm. However, these two previous works could be applied to routing missions rather than area coverage missions as in this study. Valenti et al. [18] also extended the RHTA algorithm to determine when a UAV should return to base for refueling before its fuel is exhausted. Song et al. [12] implemented the RHTA algorithm to find suboptimal solutions rapidly and compared it to the performance of MILP for the problem of providing simultaneous UAV escort service to multiple customers. Nevertheless, previous works did not discuss the reassignment of remaining missions due to the exclusion of low-battery UAVs or reintroducing of recharged UAVs. Although this study adopts the RHTA algorithm for the same purpose as the previous works, this is the first study to consider the task reassignment problem for the area coverage mission in dynamic environments.

III. PROBLEM STATEMENT AND SYSTEM ARCHITECTURE

As explained in the introduction, this study focuses on the task reassignment problem for the area coverage mission using multiple UAVs and provides the entire mission system configuration. The following assumptions are considered in this study. First, this study considers the centralized task assignment scheme where the assignment results performed on ground control station (GCS) are unilaterally transferred to UAVs. In the centralized task assignment scheme, the UAVs can not communicate with each other, and there is no requirement to achieve consensus among UAVs, unlike the distributed task assignment scheme. Second, this study assumes an ideal communication situation. Therefore, the presence of communication loss or communication delay is not considered. Third, it is assumed that the mission area given by the operator is convex. Based on previous literature [8], [9], this assumption is required to facilitate the generation of waypoints for the back-and-forth movement. The waypoint generation procedure was explained in detail in the previous work [13], so the duplicated description is omitted in this study. Fourth, this study envisions performing the area coverage mission on flat terrain without obstacles the operator is unaware of in advance, such as other UAVs and air balloons. Lastly, the generated waypoints for the area coverage mission only need to be visited once by each UAV.

Fig. 1 shows the hardware and software architecture of the entire system from operator inputs through GCS to motor inputs for each UAV. As shown in Fig. 1, the main hardware components are multiple UAV platforms and one laptop computer. There are five subsystems in terms of the software—GCS for task definition and task monitoring, task assignment, task management, path planning, and flight control subsystems. The GCS and task assignment subsystems work on the laptop computer. The task management subsystem, the path planning subsystem, and the flight control subsystem operate on an onboard computer mounted on each UAV platform. Note that the entire system works on the robot operating system (ROS) framework [23].

The functions of the five subsystems can be summarized as follows. First, the GCS subsystem based on QGroundControl [24] concretizes the mission from operator inputs and shows the mission's progress to the operator for monitoring. Second, the task assignment subsystem initially determines the optimal list of waypoints to each UAV by considering the remaining battery power and quickly generates a new suboptimal list of waypoints using the modified RHTA algorithm whenever dynamic events occur during the mission. Third, the task management subsystem

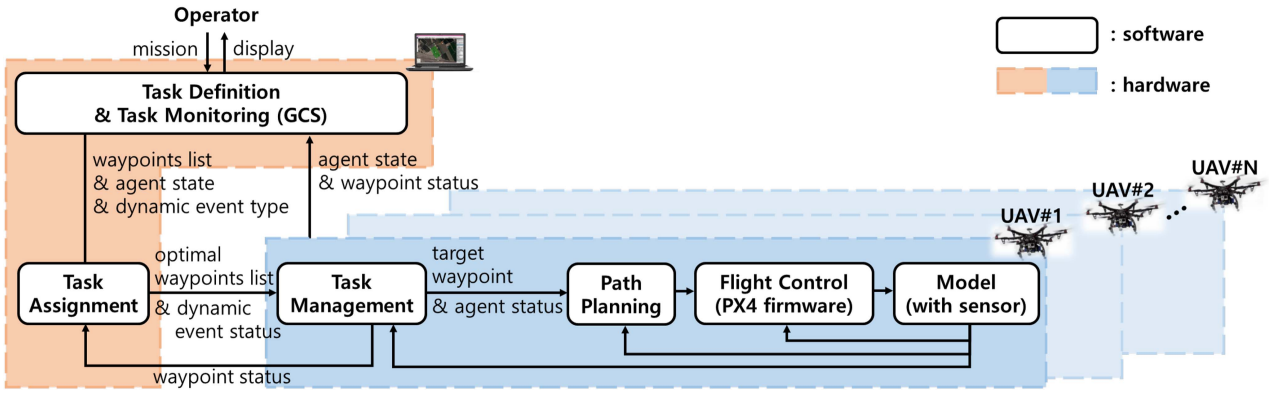


Fig. 1. Hardware and software architecture of the entire system.

identifies agent status and waypoint status. Agent status indicates whether the UAV is: 1) on the ground for take-off; 2) in-flight; 3) hovering to wait for reassignment; or 4) landing after mission completion. On the other hand, waypoint status presents whether a waypoint: 1) has already been visited; 2) should be visited now (i.e., corresponding to the target waypoint); or 3) should be visited in the future. Fourth, the path planning subsystem generates guidance commands to allow a UAV to fly toward the target waypoint at a constant flight speed. Lastly, PX4 [25], which is an open-source autopilot software, is used as the flight control subsystem to calculate motor inputs for the UAV to follow the guidance commands.

The messages exchanged between the five subsystems mentioned above can be listed. First, as the inputs of the GCS subsystem, the agent state (e.g., position, orientation, and remaining battery power) is sent from the sensors mounted on each UAV, and the waypoint status is sent from the task management subsystem for monitoring during the mission. The positions of each UAV and the waypoint in the inertial coordinate corresponding to the outputs of the GCS subsystem are transmitted to the task assignment subsystem. Dynamic event types are also sent for task reassignment: 1) zero if nothing happens; 2) one when entering additional missions; 3) two when changing the number of UAVs. Second, in the task assignment subsystem, the initial task assignment is sufficient as the inputs from the GCS subsystem. On the other hand, the task management subsystem also requires the waypoint status and agent status for task reassignment during the mission. The task assignment subsystem outputs are the optimal waypoints assigned to each UAV and sent to the task management subsystem. In particular, the dynamic event status indicating whether or not the dynamic event has been resolved is sent to the task management subsystem. Third, as inputs to the task management subsystem, the UAV's position and the remaining battery power obtained from the sensors mounted on each UAV are required to determine the waypoint and agent status, respectively. As the output of the task management subsystem, the waypoint status is transferred to the task assignment subsystem, and the target waypoint (included in the waypoint status) and the agent status are sent to the path planning subsystem. Lastly, in the path planning subsystem, the velocity command is generated based on the inputs from the task management subsystem and then sent to the flight control subsystem.

For ease of explanation, the process when the additional mission area is input by the operator is rearranged in chronological order. First, when the operator starts to modify the mission through GCS, the task management subsystem switches all UAVs to hovering. Second, the task management subsystem transmits the current waypoint status to the task assignment subsystem. Third, when the operator completes the mission change, the task assignment subsystem generates a new optimal waypoint list and sends it to the task management subsystem. Lastly, the task management subsystem changes the agent status to in-flight. The other dynamic event (i.e., changing the number of agents) can be treated similarly.

IV. MODIFIED RHTA ALGORITHM

A. RHTA Algorithm

The RHTA algorithm is one of the heuristic methods for determining the solution of an optimization problem by dividing the original problem into smaller problems and iteratively solving the smaller problems [22]. In other words, the RHTA algorithm does not consider all the tasks in the original problem, only the maximum number of tasks that can be assigned to each agent in each iteration. Iteration continues until all tasks in the original problem have been assigned. The performance of the RHTA algorithm depends on the maximum number of tasks (i.e., the size of a small problem) [21]. As the maximum number of operations increases, it is more likely to converge to the optimal solution but increase the computation time. In contrast, a smaller maximum number of tasks has the opposite effect. When the maximum number of tasks is set to one, the RHTA algorithm can be considered the greedy algorithm.

Using the toy problem represented in Fig. 2, the RHTA algorithm can be reinterpreted as follows. The toy problem has three agents and 12 waypoints. The maximum number of tasks in this toy problem is set to three. In the beginning, each agent can have ${}_{12}C_3$ combinations by choosing three waypoints in any order from 12 unassigned waypoints. For example, agent 1 can have three combinations such

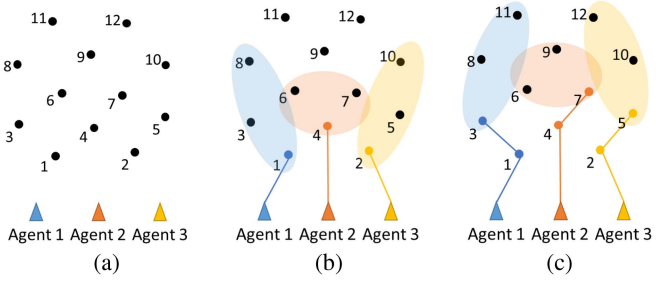


Fig. 2. Toy problem for RHTA algorithm. (a) Initial. (b) First iteration. (c) Second iteration.

as (1,3,8), (4,10,12), and (6,9,11). Agent 1 then lists all possible permutations of each combination and determines the best permutation among the listed permutations. For example, the combination (1,3,8) can have 3! permutations such as 1–3–8, 1–8–3, 3–1–8, 3–8–1, 8–1–3, and 8–3–1. The shortest permutation can be assumed to be 1–8–3 for the combination (1,3,8). Similarly, the shortest permutation for each combination can be determined. As a result, there are ${}_{12}C_3$ shortest permutations. Among ${}_{12}C_3$ shortest permutations, the best permutation is determined through optimization and assigned to each agent. Lastly, only the first waypoint of the best permutation is finally assigned to each agent, and the rest are discarded. For example, suppose the best permutations assigned for agent 1, agent 2, and agent 3 are 1–8–3, 4–7–6, and 2–10–5, respectively. Then, the waypoints assigned to agent 1, agent 2, and agent 3 are 1, 4, and 2, respectively. Repeat the same process until the number of unassigned waypoints is zero.

B. Improvement

Two modifications have been proposed compared to the typical RHTA algorithm to solve the given area coverage problem with the RHTA algorithm. First, the modified RHTA algorithm can generate a set of waypoints satisfying the back-and-forth movement by considering the remaining battery power. Second, in this study, the maximum number of tasks in the RHTA algorithm is not fixed but reduced under a particular condition. A detailed description of each modification follows:

Regarding the first improvement, in the typical RHTA algorithm, at the end of each iteration, only the first waypoint of the best waypoint permutation is assigned to the agent, and the rest are discarded. On the other hand, in the modified RHTA algorithm, waypoints are assigned in pairs to implement the back-and-forth movement. First, all waypoint combinations consisting of random pairs are found to achieve this goal. And then, some combinations that cannot generate the back-and-forth movement are removed from the original set of combinations. This elimination can reduce the amount of computation by decreasing the number of combinations to be considered in the following process. In addition, permutations exceeding the remaining battery power are excluded in the optimization problem determining the best permutation for each UAV.

Algorithm 1: Modified RHTA Algorithm.

Procedure MRHTA$N, M, \mathcal{W}, D, m, F$
  $\mathcal{W}_0 \leftarrow \mathcal{W}$, $\boldsymbol{s}^{(i)} \leftarrow \varnothing$ for $i = 1, \ldots, N$
  while $n(\mathcal{W}_0) > 0$ **do**
    if $n(\mathcal{W}_0) < m \times N$ **then**
      if $m \geq 2$ **then**
        $m \leftarrow m - 2$
      else
        $m \leftarrow 1$
      end if
    end if
    Generate a set of combination $\mathcal{C}$ satisfying the back-and-forth movement
    for each agent i **do**
      for each combination $\boldsymbol{c} \in \mathcal{C}$ **do**
        Determine the shortest permutation $\boldsymbol{p}_i$ for $\boldsymbol{c}$
      end for
    end for
    Solve the optimization problem to determine the best permutation $\boldsymbol{p}_i^*$ considering the remaining battery percent F_i
    for each agent i **do**
      if $m \geq 2$ **then**
        $\mathcal{W}_0 \leftarrow \mathcal{W}_0 / \{\boldsymbol{p}_i^*(1), \boldsymbol{p}_i^*(2)\}$,
        $\boldsymbol{s}^{(i)} \leftarrow \boldsymbol{s}^{(i)} \oplus_{end} \{\boldsymbol{p}_i^*(1), \boldsymbol{p}_i^*(2)\}$
      else
        $\mathcal{W}_0 \leftarrow \mathcal{W}_0 / \boldsymbol{p}_i^*(1)$,
        $\boldsymbol{s}^{(i)} \leftarrow \boldsymbol{s}^{(i)} \oplus_{end} \{\boldsymbol{p}_i^*(1)\}$
      end if
    end for
  end while
end procedure

Concerning the second improvement, the main limitation of the typical RHTA algorithm is that its performance depends on the maximum number of tasks. However, determining the maximum number of tasks in the RHTA algorithm is empirical and problem-dependent. In addition, the maximum number of tasks from the beginning to the end of the typical RHTA algorithm is fixed. However, an inappropriate and fixed maximum number of tasks may not determine a good suboptimal solution. Let us consider the toy problem shown in Fig. 2 again. The 12 waypoints in the toy problem can be assigned to three agents with just two iterations. If the optimal waypoint lists determined in the second iteration are 3–8–11, 7–9–6, and 5–10–12, then the final optimal waypoint lists are 1–3–8–11, 4–7–9–6, and 2–5–10–12. The RHTA algorithm does not proceed further when the number of remaining tasks becomes smaller than the maximum number of tasks multiplied by the number of agents. If the maximum number of tasks is set to four in the toy problem, the algorithm completes it in one iteration. Whether the determined suboptimal solution is sufficiently close to the exact optimal solution is questionable. Therefore, this study proposes to reduce the maximum number of tasks at the end of the typical RHTA algorithm to find a

good suboptimal solution. The maximum number of tasks gradually decreases whenever the number of unassigned waypoints is less than the maximum number of tasks multiplied by the number of agents until its value equals one.

C. Summary of Modified RHTA Algorithm

The detailed process of the modified RHTA algorithm is summarized in Algorithm 1. The modified RHTA algorithm requires six parameter inputs. These are: 1) the number of agents $N \in \mathbb{R}^1$; 2) the number of waypoints $M \in \mathbb{R}^1$; 3) the set of waypoints $\mathcal{W} \in \mathbb{R}^M$; 4) the distance matrix between waypoints $D \in \mathbb{R}^{M\times M}$; 5) the maximum number of tasks $m \in \mathbb{R}^1$; 6) the residual battery percent $F = \{F_1, \ldots, F_N\} \in \mathbb{R}^N$. The modified RHTA algorithm starts with the initialization of the unassigned waypoint set $\mathcal{W}_0$ and the optimal waypoint list assigned to each agent $\boldsymbol{s}^{(i)}$ for $i = 1, \ldots, N$. After initialization, the number of maximum tasks m is checked for each iteration. If the number of unassigned waypoints $n(\mathcal{W}_0)$ is less than the maximum number of tasks m multiplied by the number of agents N, then the number of maximum tasks m is reduced. Once the number of maximum tasks m is determined, a set of all possible combinations $\mathcal{C}$, where the back-and-forth movement can be applied, is produced. For each agent, the shortest permutation $\boldsymbol{p}_i \in \mathcal{P}$ is determined for each combination $\boldsymbol{c} \in \mathcal{C}$. Note that the number of elements in the set $\mathcal{P}$ is equal to the number of elements in the set $\mathcal{C}$ multiplied by the number of agents. After that, the optimization problem can be formulated based on MILP to determine the best permutation $\boldsymbol{p}_i^*$ for each agent. In the last process, the first pair of waypoints (i.e., $\boldsymbol{p}_i^*(1)$ and $\boldsymbol{p}_i^*(2)$) is finally assigned to each agent. Therefore, in Algorithm 1, the first and second waypoints are removed from the set $\mathcal{W}_0$. Additionally, $\boldsymbol{s}^{(i)} \oplus_{end} \{k\}$ denotes that the waypoint k is augmented at the end of agent i's optimal waypoint list $\boldsymbol{s}^{(i)}$ [26]. If the iteration is repeated until the termination condition $n(\mathcal{W}_0) = 0$ is satisfied, all waypoints are assigned to agents.

The optimization problem to determine the best permutation $\boldsymbol{p}_i^*$ in Algorithm 1 can be formulated as follows:

$$\text{Minimize} \quad d_{\max} + \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{P} W_{ij}x_{ij} \tag{1}$$

subject to

$$\sum_{i=1}^{N}\sum_{j=1}^{P} A_{ijk}x_{ij} \le 1, \quad \text{for} \quad k = 1, \ldots, M \tag{2}$$

$$\sum_{j=1}^{P} x_{ij} = \begin{cases} 1 & \text{if } \min\left(\sum_{j=1}^{P} W_{ij}\right) \le f_i \\ 0 & \text{otherwise} \end{cases}$$
$$\text{for} \quad i = 1, \ldots, N \tag{3}$$

$$\sum_{j=1}^{P} W_{ij}x_{ij} \le d_{\max}, \quad \text{for} \quad i = 1, \ldots, N \tag{4}$$

$$\sum_{j=1}^{P} W_{ij}x_{ij} = d_i, \quad \text{for} \quad i = 1, \ldots, N. \tag{5}$$

In (1)–(5), there are three decision variables. First, $x_{ij} \in \mathbb{R}^{N\cdot P}$ is a binary variable and equals one if the jth permutation of agent i is assigned and zero otherwise. Second, $d_i \in \mathbb{R}^N$ is a continuous variable representing the travel distance of each agent. Third, $d_{\max} \in \mathbb{R}^1$ is a continuous slack variable representing the longest travel distance among all agents. In other words, $d_{\max}$ is the maximum value among d_i for $i = 1, \ldots, N$. With these decision variables, the objective function, (1), is designed to minimize the maximum travel distance $d_{\max}$ [i.e., the first term in (1)] and the average travel distance of all agents ([i.e., the second term in (1))]. In (1), the constant matrix $W_{ij} \in \mathbb{R}^{N\cdot P}$ denotes the accumulated travel distance from the current position to the last waypoint in the jth permutation. The first constraint, (2), indicates that each waypoint should be visited at most once. The matrix $A_{ijk} \in \mathbb{R}^{N\cdot P\cdot M}$ can be generated based on the shortest permutation set $\mathcal{P}$. Note that A_{ijk} equals one if waypoint i is included in permutation p of agent i and zero otherwise. The second constraint (3) is required to assign only one permutation to each available agent. In (3), the constant f_i is set to the maximum travel distance the agent can travel when fully charged multiplied by the remaining battery percent F_i. Therefore, the second constraint ensures that no permutations exceeding the remaining battery power are allocated. The third constraint (4) is added to define the maximum travel distance $d_{\max}$. The last constraint (5) is necessary to determine the decision variable d_i. The upper bound of the decision variable d_i is set to f_i.

V. SIMULATION

The performance of the proposed system was validated through two different simulations. In the first simulation, the performance of the proposed RHTA algorithm was examined in terms of computation time and solution optimality when the first dynamic event that adds a mission area occurs. The second simulation demonstrated the ability of the entire system to handle the second dynamic event where the number of UAVs changes during a mission. For all simulations, the maximum number of tasks in the proposed RHTA algorithm was set to six, then gradually decreased to one. Note that the proposed RHTA algorithm was written in MATLAB (version R2020a; The MathWorks Inc., Natick, MA, USA) and used Gurobi solver [27], which is a standard optimization software package for MILP, to solve the optimization problem described in Section IV-C. All numerical computations were performed using a laptop computer with a 2.6 GHz Intel i7 CPU and 16 GB RAM running the Ubuntu operating system.

A. MATLAB Simulation

This simulation mainly focuses on comparing the performance of the proposed RHTA algorithm with three different algorithms for task reassignment. The first algorithm is the iterative MILP algorithm to find the optimal solution. The second algorithm is the typical RHTA algorithm. The third algorithm is the greedy algorithm, one of the heuristic

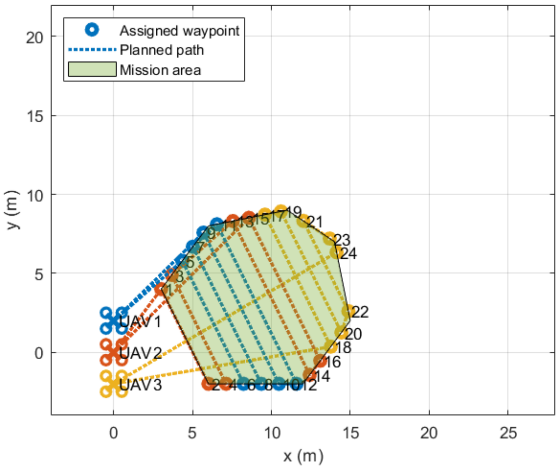


Fig. 3. Original mission and initial task assignment results for sample configuration.

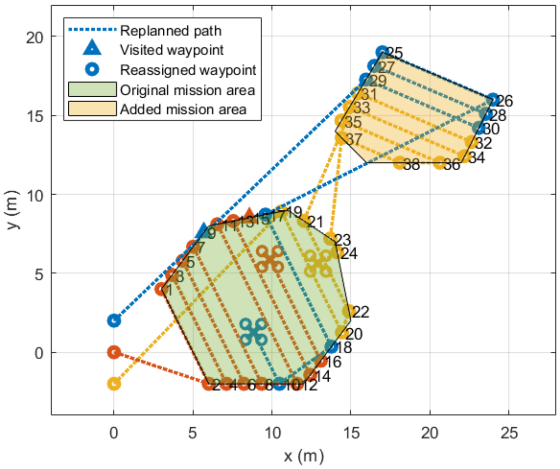


Fig. 4. Task reassignment results based on iterative MILP algorithm for sample configuration.

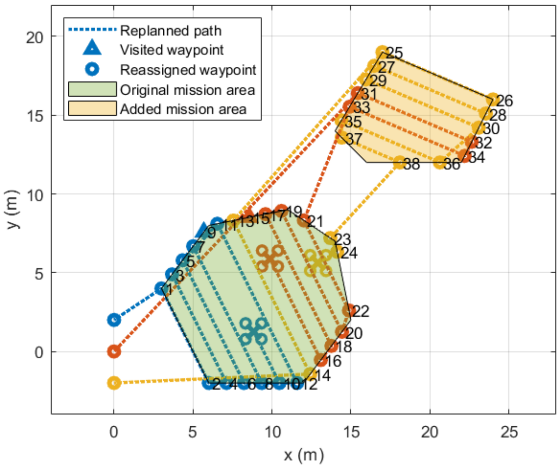


Fig. 5. Task reassignment results based on modified RHTA algorithm for sample configuration.

algorithms to find a good suboptimal solution. These algorithms are compared in terms of two performance aspects: 1) the CPU time; and 2) the value of the objective function in (1).

Three UAVs with fully charged batteries are engaged in a mission to cover a mission area given by the operator. The mission area is an arbitrary polygonal area, given at random: a triangle, a rectangle, a pentagon, a hexagon, or a heptagon. During the mission, the operator adds a new mission area. At this time, task reassignment is required to include the additional area in the mission. Three UAVs pause flight and hover in the air before solving the task reassignment problem. The UAVs continue to fly after the new optimal waypoint lists are determined through task reassignment. In order to verify the performance under various situations, 100 configurations are randomly generated by selecting two polygonal areas out of five polygonal areas and varying the dynamic event occurrence time. The dynamic event occurrence time is regarded when the travel distance becomes 2.5, 5, 7.5, 10, 12.5, and 15 m, respectively, after the start of take-off.

The simulation results for one sample configuration are as follows. Fig. 3 shows the initial task assignment result of the sample configuration. As shown in Fig. 3, 24 waypoints were generated to cover the green heptagon area. The MILP algorithm was used to solve the initial task assignment. As a result, the sequence of waypoints assigned to UAV 1, UAV 2, and UAV 3 are 9–10–12–11–7–8–6–5, 15–16–14–13–3–4–2–1, and 24–23–21–22–20–19–17–18, respectively. Figs. 4–7 show the results of task reassignment obtained by three different algorithms to consider the additional mission area. As shown in Figs. 4–7, 14 waypoints were additionally generated to cover the yellow pentagon area. In this sample configuration, the new area was added when the UAV starts flying from its initial position and has a cumulative travel distance of 15 m. At this time, UAV 3 had not yet passed the first waypoint, and UAV 1 and UAV 2 were between the first and second waypoints. In Figs. 4–7, when the dynamic event occurs, the waypoints already passed are displayed with the triangle. On the other hand, the waypoint and the path not yet passed are indicated by the circle and dotted line, respectively. Table I summarizes the task reassignment results determined by each algorithm. The objective function was determined by the iterative MILP algorithm, the proposed RHTA algorithm, the typical RHTA algorithm, and the greedy algorithm was 189.40, 193.90, 203.07, and 221.26 m, respectively. The computation time required to solve the iterative MILP algorithm, the proposed RHTA algorithm, the typical RHTA algorithm, and the greedy algorithm was 73.40, 9.94, 9.00, and 0.09 s, respectively.

Figs. 8 and 9 show the overall results for all 100 configurations. Fig. 8 shows the CPU times spent by each algorithm using Whisker plots, where the central mark is the median, the edges of the box are the 25th and 75th percentiles, and the top and bottom horizontal lines are the minimum and maximum data points. As shown in Fig. 8, the iterative MILP algorithm consumed over 40 s of CPU time for all configurations. For this reason, the iterative MILP algorithm set the time limit to 180 s of CPU time, and the best solution obtained within the time limit was reported. Fig. 9 shows the value of the objective function achieved

TABLE I
Task Reassignment Results for Sample Configuration in MATLAB Simulation

		UAV 1	UAV 2	UAV 3
MILP	Number of waypoints	9	13	14
	Sequence of waypoints	10–18–17–26–25–27–28–30–29	16–14–13–11–12–8–7–5–6–4–3–1–2	24–23–37–38–36–35–33–34–32–31–21–22–20–19
	Travel distance (m)	95.40	91.30	95.30
Modified RHTA	Number of waypoints	11	11	14
	Sequence of waypoints	10–6–5–7–8–12–11–3–4–2–1	16–18–17–19–20–22–21–33–34–32–31	24–23–38–37–35–36–30–29–27–28–26–25–13–14
	Travel distance (m)	77.42	97.18	101.78
Typical RHTA	Number of waypoints	11	9	16
	Sequence of waypoints	10–6–5–7–8–12–11–3–4–14–13	16–18–17–19–20–22–21–1–2	24–23–38–37–35–36–30–29–31–32–34–33–27–28–26–25
	Travel distance (m)	92.07	73.13	111.00
Greedy	Number of waypoints	11	11	14
	Sequence of waypoints	10–2–1–7–8–12–11–27–28–26–25	16–4–3–5–6–14–13–31–32–30–29	24–23–21–22–20–19–17–18–38–37–35–36–34–33
	Travel distance (m)	110.92	112.45	103.06

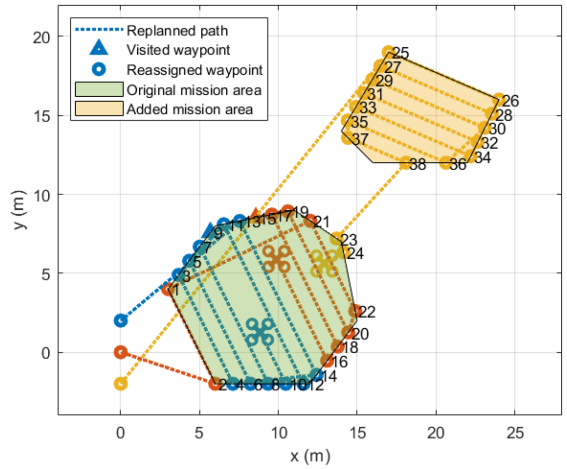


Fig. 6. Task reassignment results based on typical RHTA algorithm for sample configuration.

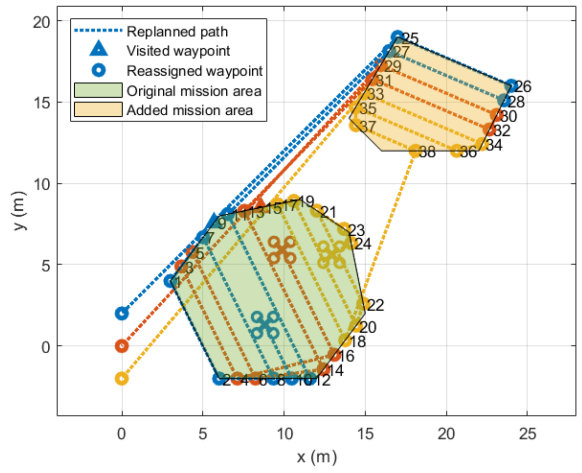


Fig. 7. Task reassignment results based on greedy algorithm for sample configuration.

by each algorithm using Whisker plots. From Fig. 9, it is apparent that the proposed RHTA algorithm can find a suboptimal solution closer to the exact optimal solution of the iterative MILP algorithm than the greedy algorithm. Therefore, it can be concluded from Figs. 8 and 9 that the proposed RHTA algorithm finds a suitable suboptimal solution within a reasonable time.

B. Gazebo Simulation

The purpose of Gazebo simulation is twofold. First, this simulation considers the dynamic event that changes the number of UAVs. In the actual flight experiment, there are cost and time constraints to preparing many UAVs. However, Gazebo simulation with PX4 Software In The Loop (SITL) can mimic several UAVs, including the dynamics and the flight controller. Second, this simulation aims to evaluate the performance of the entire system described in Fig. 1 before the actual flight experiment, unlike the previous MATLAB simulation that only verified the proposed RHTA algorithm to solve the task reassignment problem. Other settings for this simulation are as follows. First, MAVROS [28] was utilized as the interface between the path planning subsystem and the flight control subsystem. The velocity command generated in the path planning subsystem was transferred to the flight control subsystem at 40 Hz. Second, the mission altitude was set to 3 m, and the transition altitude of five UAVs was set to 5, 6, 7, 8, and 9 m, respectively. The mission altitude corresponds to the altitude of waypoints generated inside the polygonal mission area. On the other hand, the transition altitude refers to the unique altitude assigned to each UAV to avoid collisions while moving to the mission area after take-off and returning to the initial position after completing the mission (outside of the polygonal area). Detailed methods of implementing the transition altitude have been discussed in the previous paper [13] and will not be repeated here. Lastly, the UAV determines that it has reached the waypoint when it enters a circle with a radius of 1 m from the waypoint.

Gazebo simulation considers the following scenarios. Initially, five UAVs participate in the area coverage mission for this simulation. However, they are sequentially removed from the mission at random moments until only one UAV remains over time. Figs. 10–15 illustrate the simulation results. Fig. 10 shows the waypoints assigned to five UAVs to cover the mission area, corresponding to the initial task

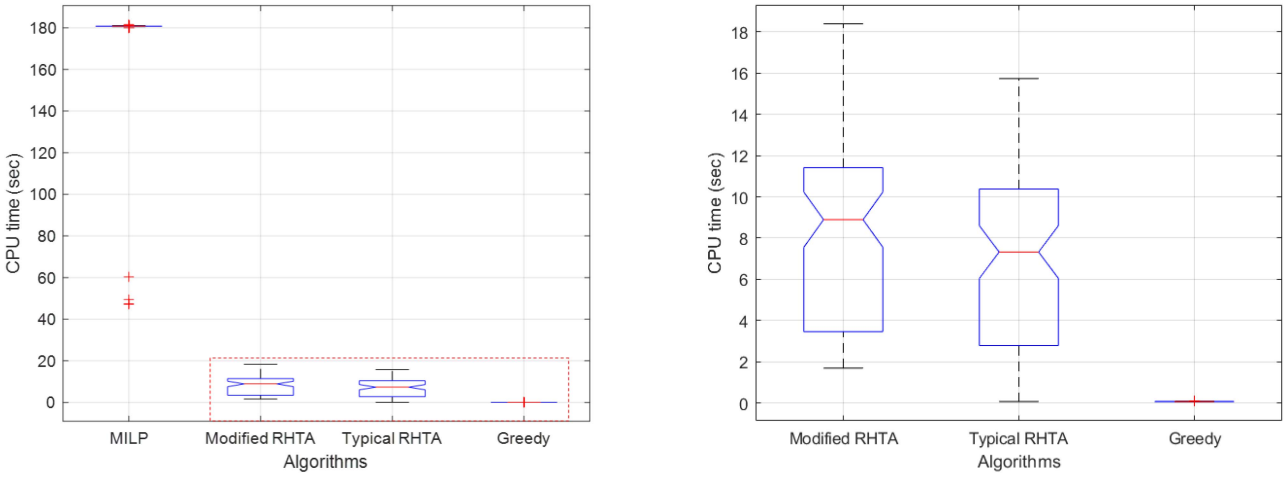


Fig. 8. CPU time for all configurations. (a) Original figure. (b) Enlarged figure.

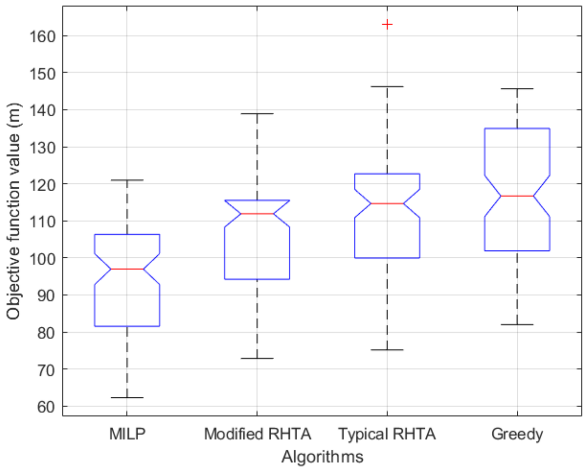


Fig. 9. Value of objective function for all configurations.

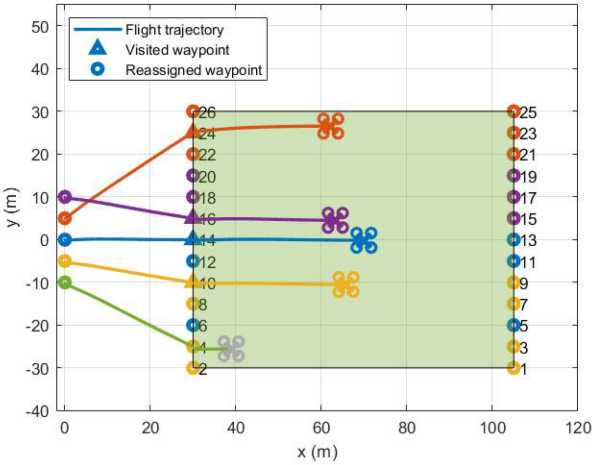


Fig. 11. Task reassignment results when the number of UAVs is reduced to four in Gazebo simulation.

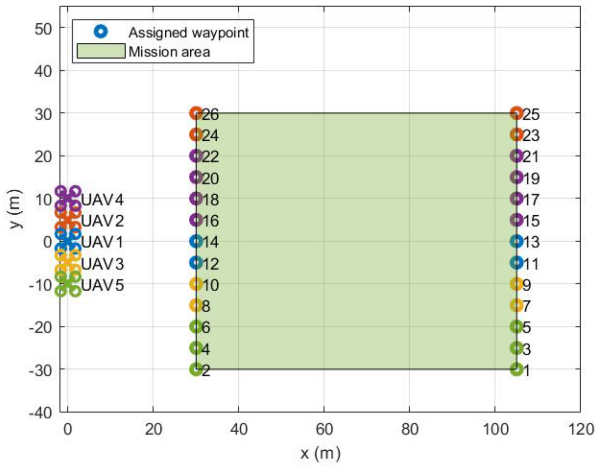


Fig. 10. Task assignment results when the number of UAVs is five in Gazebo simulation.

assignment result using the MILP algorithm. The computation time was 63.92 s. As shown in Fig. 10, the sequence of waypoints assigned to five UAVs were 14–13–11–12, 24–23–25–26, 10–9–7–8, 16–15–17–18–20–19–21–22, and 4–3–1–2–5–6, respectively. Upon receiving the assigned waypoints, five UAVs start the mission by taking off from their initial position. Fig. 11 shows the first task reassignment result when the grayed-out UAV 5 was removed from the mission 930 s after the start of the mission. When UAV 5 landed suddenly for several reasons, such as communication loss and motor fault, other UAVs performed hovering. At the same time, the task reassignment problem is solved to reassign the remaining waypoints for the remaining UAVs. In Fig. 11, the waypoint and the path already passed are displayed with a triangle and solid line, respectively, when the dynamic event occurs. However, although UAV 5 passed through waypoint 4, waypoint 4 was considered an unvisited waypoint as shown in Fig. 11. The reason is that the back-and-forth movement connecting waypoints 3

TABLE II
Task Assignment Results in Gazebo Simulation

	Number of UAVs	Algorithm	CPU time (sec)	Sequence of waypoints
Task assignment	5	MILP	63.92	UAV 1 : 14–13–11–12 UAV 2 : 24–23–25–26 UAV 3 : 10–9–7–8 UAV 4 : 16–15–17–18–20–19–21–22 UAV 5 : 4–3–1–2–5–6
1st task reassignment	4 (Excluding UAV 5)	Modified RHTA	0.71	UAV 1 : 13–11–12–6–5 UAV 2 : 23–21–22–26–25 UAV 3 : 9–7–8–4–3–1–2 UAV 4 : 15–17–18–20–19
2nd tast reassignment	3 (Excluding UAV 4)	Modified RHTA	0.34	UAV 1 : 12–18–17–1–2 UAV 2 : 22–20–19–25–26 UAV 3 : 8–6–5–3–4
3rd task reassignment	2 (Excluding UAV 3)	Modified RHTA	0.35	UAV 1 : 17–5–6–4–3–1–2 UAV 2 : 19–25–26
4th task reassignment	1 (Excluding UAV 2)	Modified RHTA	0.26	UAV 1 : 6–4–3–1–2–25–26

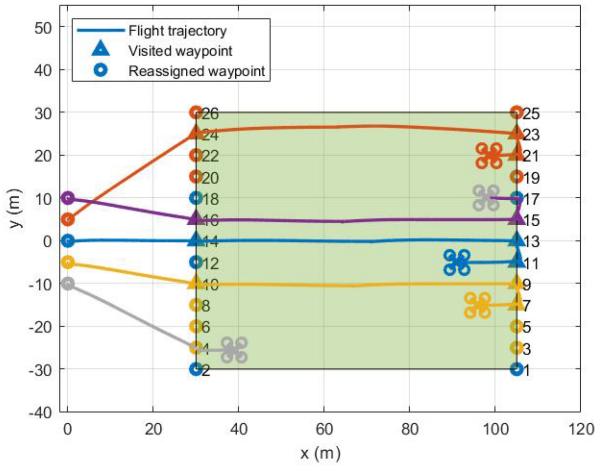


Fig. 12. Task reassignment results when the number of UAVs is reduced to three in Gazebo simulation.

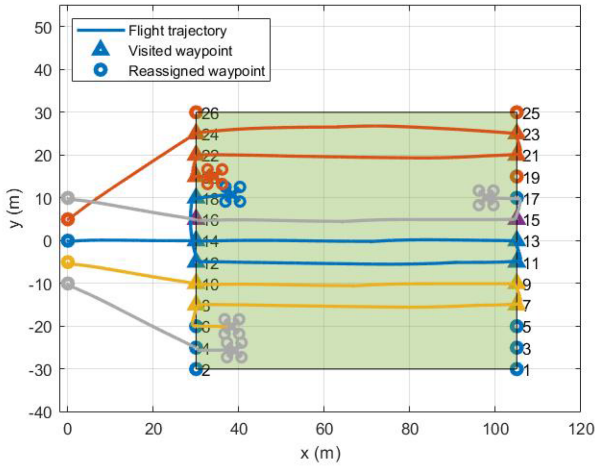


Fig. 13. Task reassignment results when the number of UAVs is reduced to two in Gazebo simulation.

and 4 cannot be completed without revisiting waypoint 4. Note that the task reassignment problem was solved using the proposed RHTA algorithm, and the spent computation time was 0.71 s. After task reassignment, the sequence of waypoints assigned to four UAVs were 13–11–12–6–5, 23–21–22–26–25, 9–7–8–4–3–1–2, and 15–17–18–20–19,

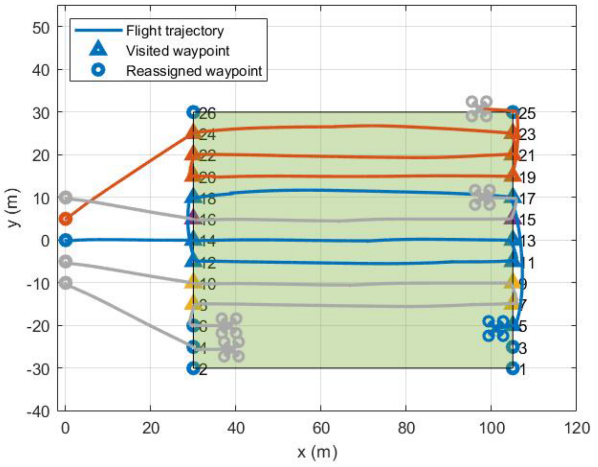


Fig. 14. Task reassignment results when the number of UAVs is reduced to one in Gazebo simulation.

respectively. When the new optimal waypoint list was sent to four UAVs, the UAVs continued flying toward their target waypoint. Figs. 12– 14 present the task reassignment results when UAV 4 is excluded 1830 s, UAV 3 is ruled out 3090 s, and UAV 2 is removed 4470 s after the mission starts, respectively. Table II summarizes the list of waypoint lists reassigned to the UAVs and the computation time required for task reassignment in each case. Fig. 15 shows the resulting flight trajectories of all UAVs in a 3-D space. In Fig. 15, the small squares represent virtual waypoints added to specify a unique transition altitude for each UAV.

VI. FLIGHT EXPERIMENT

This section describes the outdoor flight experiment. Unlike the previous simulations, the flight experiment has the following significance. First, while the simulations accounted for different dynamic events for each simulation, this flight experiment considered both types of dynamic events: adding mission areas and changing the number of UAVs during a mission. Second, in this experiment, the remaining battery percent information was read as ROS messages. Therefore, the travel distance of the UAV was limited so as not to exceed the remaining battery power. Third,

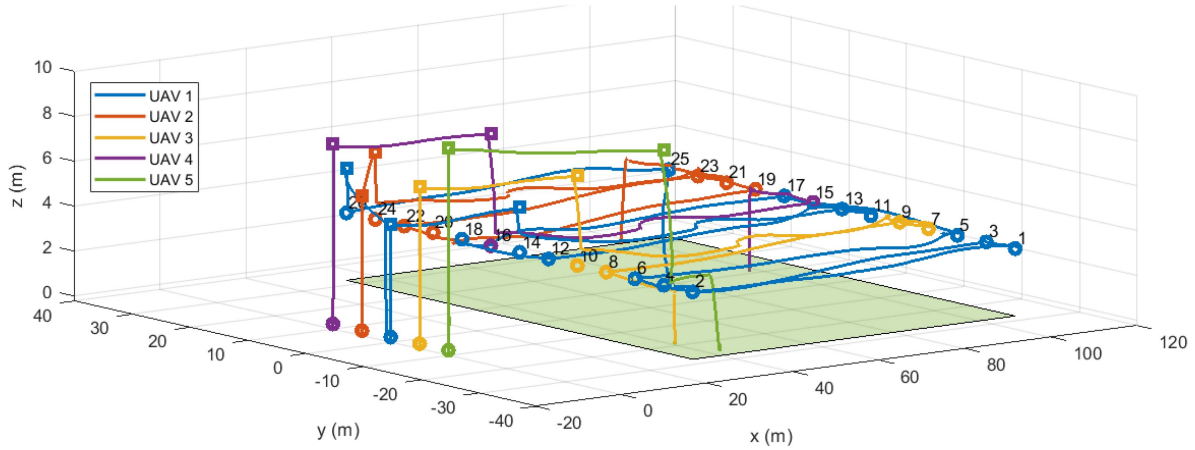


Fig. 15. Flight trajectories in Gazebo simulation.

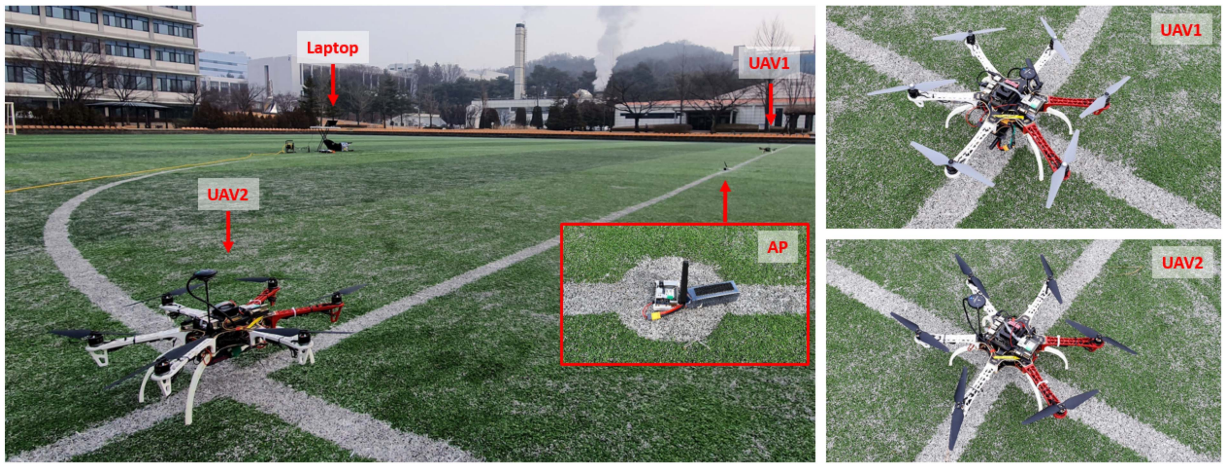


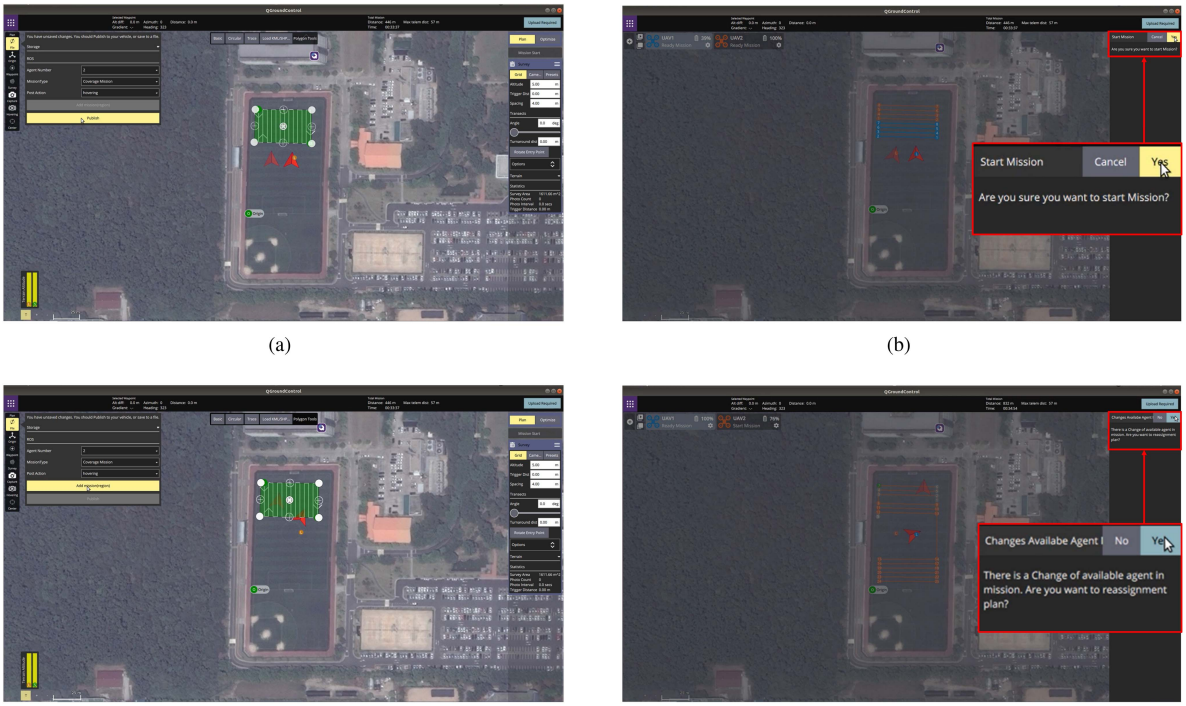
Fig. 16. Hardware configuration for flight experiment.

this flight experiment was carried out with real hardware: 1) two hexacopters; 2) one laptop computer for the GCS hardware; 3) a single Wi-Fi AP as shown in Fig. 16. Based on the DJI F550 frame, each hexacopter was equipped with HolyBro Durandal running PX4 and NVIDIA Xavier NX as a companion computer. The flight controller calculates the motor inputs and controls the hexacopter's motion by estimating the states such as position, velocity, and attitude. The companion computer fulfills the function of receiving the list of waypoints from the laptop computer, performing task management, and path planning, and transferring the desired position, velocity, yaw, and yaw rate to the flight controller. All the hexacopter hardware platform details are listed in Table III. Additionally, QGroundControl [24], which is an open-source GCS working with various vehicle types supported by PX4, has been extended to allow the human operator to define the mission by entering parameters. The most significant change is that the extended QGroundControl can publish and subscribe to ROS messages to communicate with other subsystems in the ROS environment. Other modifications to QGroundControl are detailed in the previous paper [13]. Note that the proposed RHTA algorithm works on a laptop computer, and the maximum number of tasks was set to four for the flight experiment. The laptop computer and companion computers mounted on hexacopters were connected to a single network over Wi-Fi.

TABLE III
Hexacopter Hardware Platform Specification

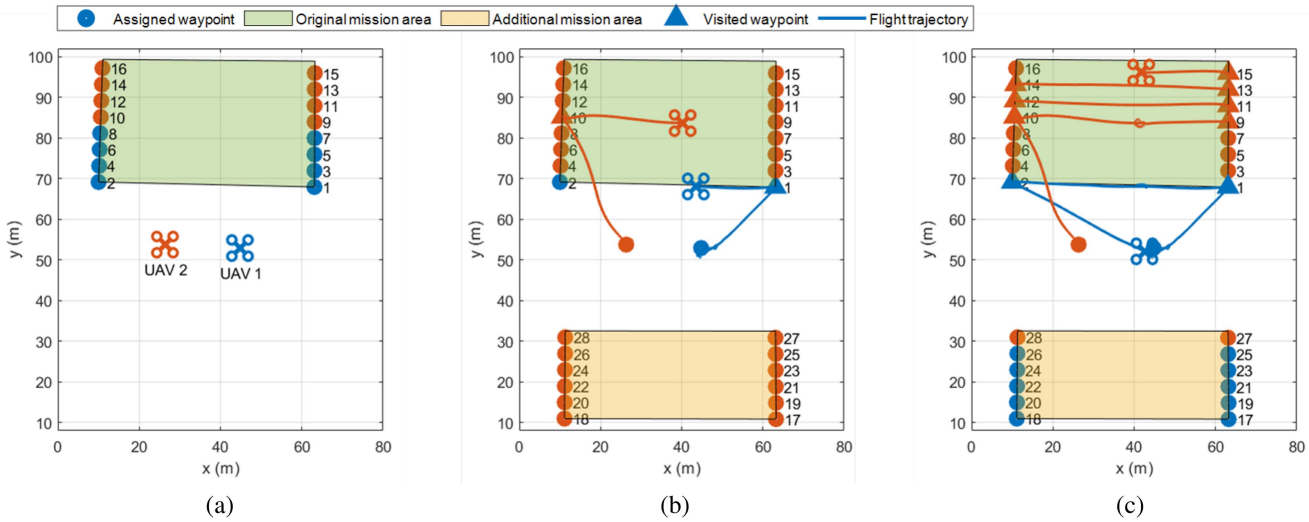
Hexacopter	Frame	DJI F550
	Motor	DJI 2312E
	ESC	DJI 430 LITE
	Propeller	DJI 9450
	Battery	Lumenier 4S 5200mAh
Sensors	GPS	ublox M8N
Computers	Flight controller	Holybro Durandal
	Companion computer	NVIDIA Xavier NX

The scenario considered in the flight experiment is as follows. First, the human operator enters: 1) the number of UAVs; 2) the polygonal areas; 3) the mission altitude at which the coverage mission will be performed; 4) the spacing between waypoints through QGroundControl. As shown in Fig. 17(a), for the flight experiment, two UAVs were given the initial mission covering the single square mission area, the mission altitude was set to 5 m, and the spacing between waypoints was set to 4 m. In addition to



(a) (b) (c) (d)

Fig. 17. Operator inputs through QGroundControl in the flight experiment. (a) Operator input to define the initial mission. (b) Operator input to start the flight. (c) Operator input to add mission area. (d) Operator input to add UAV after battery replacement.



(a) (b) (c)

Fig. 18. Task assignment and reassignment results in flight experiment. (a) Initial task assignment results. (b) First task reassignment results. (c) Second task reassignment results.

operator inputs, the initial settings considered in the flight experiment are as follows. At the start of the mission, UAV 1 has a fully charged battery of 100% while UAV 2 has a battery that is only about 50% charged. Considering the GPS positional error, the transition altitudes of two hexacopters were set to 8 and 11 m, respectively. As shown in Fig. 17(a), once the operator's input is complete, the task assignment subsystem optimally assigns waypoints to individual UAVs and displays the task assignment result on QGroundControl. As shown in Fig. 17(b), when the operator finally approves the flight according to the task assignment result, each UAV takes off and flies sequentially along with the assigned waypoints. During the mission, the operator can add the mission area at any time through QGroundControl. As shown in Fig. 17(c), once the operator clicks the "Add mission" icon, QGroundControl transmits the message to each UAV to switch to the hovering mode. Then, the operator specifies the new mission area. Once the operator reinput is complete, the task assignment subsystem sorts out the remaining waypoints based on the waypoint status sent

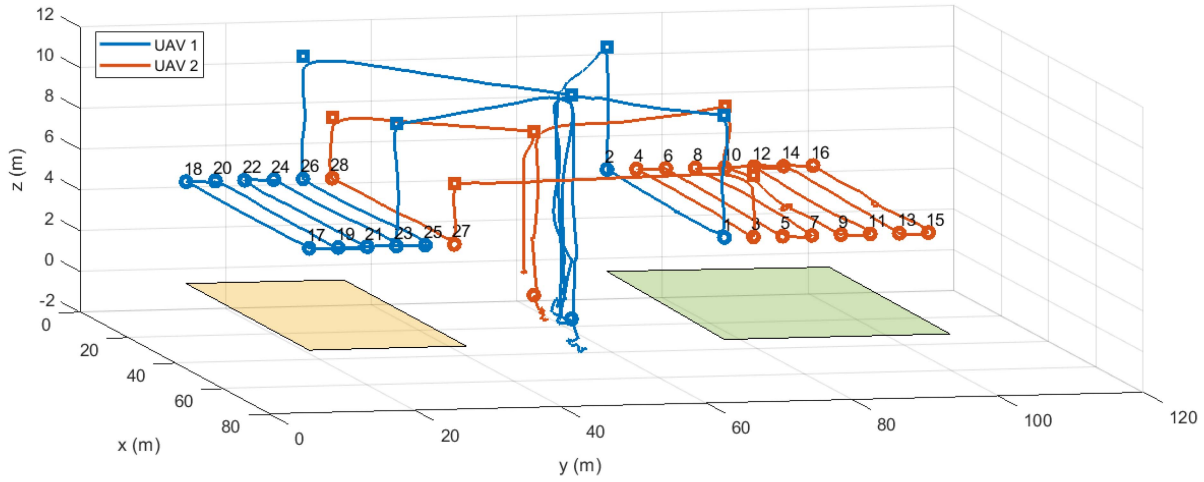
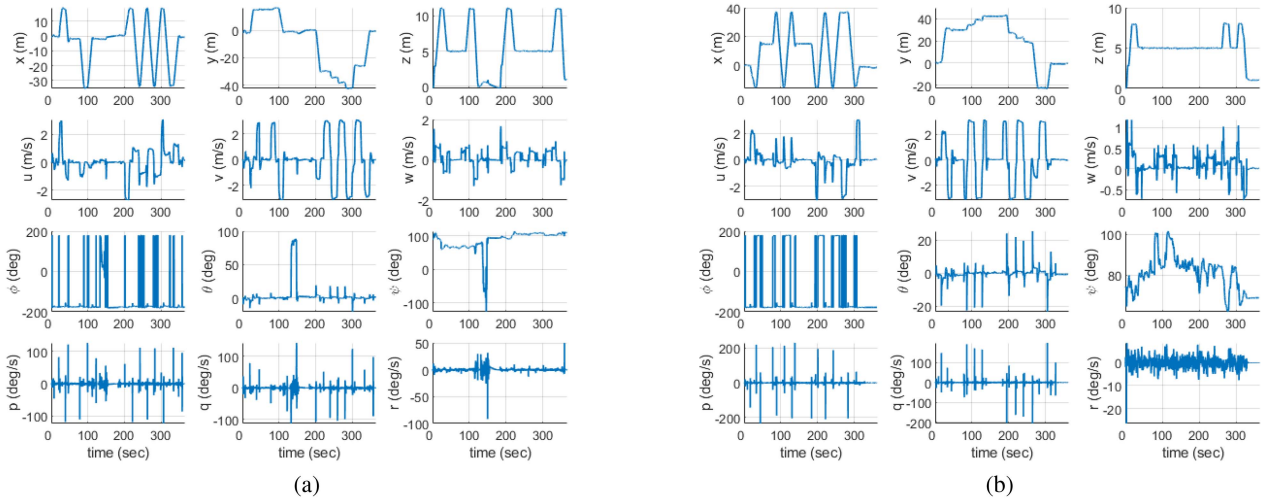


Fig. 19. Flight trajectories in flight experiment.



(a) (b)

Fig. 20. Time histories of states. (a) UAV 1. (b) UAV 2.

from each UAV. The task assignment subsystem reassigns remaining waypoints and newly added waypoints to UAVs and then transmits the first task reassignment result to each UAV. Upon receiving a new list of waypoints, each UAV ends hovering and begins flying along with the assigned waypoints. UAV 1 with only about 50% battery is assigned fewer waypoints than UAV 2 with a fully charged battery, so UAV 1 visits all given waypoints before UAV 2 then returns to the take-off position and lands. When UAV 1's battery is replaced with a fully charged battery, QGroundControl asks the operator whether or not to put UAV 1 on the mission as shown in Fig. 17(d). If the operator accepts to join UAV 1 in the mission, then UAV 2 in flight is switched to hovering mode. The task assignment subsystem solves the second task reassignment problem to relocate UAV 1 and UAV 2 to the remaining mission. Upon receiving a new list of waypoints, UAV 1 takes off, and at the same time, UAV 2 ends hovering and begins flying along with the assigned waypoints. After visiting all waypoints, each UAV returns to the initial position where it took off. And then, by landing at the initial position, UAV completes the mission.

Fig. 18(a) shows the initial task assignment results. In Fig. 18(a), the drone-shaped icons indicate the initial take-off positions. The conventional MILP algorithm initially determined this task assignment problem, and the computation time was 0.74 s. As shown in Fig. 18(a), 16 waypoints were generated to cover the green mission area. Since the maximum travel distance the UAV can travel when fully charged was set to 1500 m, the number of waypoints was equally assigned to two UAVs. The sequence of waypoints assigned to UAV 1 and UAV 2 were 1–2–4–3–5–6–8–7 and 10–9–11–12–14–13–15–16, respectively. Fig. 18(b) shows the first task reassignment results when the operator added the yellow mission area 45 s after the start of the mission. At that time, the cumulative travel distances traveled by UAV 1 and UAV 2 were 78.10 and 85.29 m, respectively. In addition, the residual battery of UAV 1 was only 7%. For this reason, as shown in Fig. 18(b), only waypoint 2 was assigned to UAV 1 while other remaining waypoints were assigned to UAV 2. Note that the proposed RHTA algorithm was used to solve this task reassignment problem. The computation time was 0.85 s. After visiting waypoint 2, UAV 1

TABLE IV
Task Assignment Results in Flight Experiment

	Dynamic event	Algorithm	CPU time (sec)	Sequence of waypoints
Task assignment	–	MILP	0.74	UAV 1 : 1–2–4–3–5–6–8–7 UAV 2 : 10–9–11–12–14–13–15–16
1st task reassignment	Add mission area	Modified RHTA	0.85	UAV 1 : 2 UAV 2 : 9–11–12–14–13–15–16–8–7–5 –6–4–3–27–28–26–25–23–24–22–21–19–20–18–17
2nd task reassignment	Add UAV	Modified RHTA	0.99	UAV 1 : 23–24–22–21–19–20–18–17–25–26 UAV 2 : 16–8–7–5–6–4–3–27–28

returned to its initial take-off position and landed. At that time, the cumulative travel distance by UAV 1 was 161.78 m. Fig. 18(c) shows the second task reassignment results when UAV 1 rejoined the mission with a fully charged battery around 135 s after the start of the mission. As shown in Fig. 18(c), residual waypoints are almost equally assigned to two UAVs. The waypoint lists assigned to UAV 1 and UAV 2 were 23–24–22–21–19–20–18–17–25–26 and 16–8–7–5–6–4–3–27–28, respectively. The computation time required to solve the second task reassignment problem using the proposed RHTA algorithm was 0.99 s. Task assignment and reassignment results are summarized in Table IV.

Fig. 19 shows the flight trajectories of UAVs recorded after the completed flight experiment. In Fig. 19, the small squares are virtual waypoints for altitude separation, set at 11 m for UAV 1 and 8 m for UAV 2. As mentioned earlier, the virtual waypoints are intended to avoid collisions between UAVs from the starting position to the polygonal mission area, when jumping between the mission areas, and back to the starting position from the mission area. Fig. 20 shows the time histories of each UAV's states, including position, velocity, attitude, and angular rate recorded in the flight experiment. The total cumulative travel distances traveled by UAV 1 and UAV 2 were 582.70 and 609.73 m, respectively. These results show that the proposed system can perform the area coverage mission with multiple UAVs even in a dynamic environment where an additional mission area is added and the number of available UAVs changes during the mission. A video attachment to this flight experiment is available at the website https://youtu.be/rBVzUXdn-xs.

VII. CONCLUSION

This article proposed a complete mission framework to perform autonomous area coverage missions with multiple UAVs under a dynamic environment. Two dynamic events were considered during the mission. The first was the addition of new mission areas by the operator, and the second was the change in the number of UAVs due to UAV failures or additional UAV participation. In order to respond flexibly to dynamic events and complete the mission, this study defined the function of the five subsystems and the information shared between each subsystem. In particular, this study attempted to develop a methodology for generating a list of waypoints to fill the given mission area and assigning the generated waypoints between UAVs optimally. Therefore, this study proposed the modified RHTA algorithm in which a pair of waypoints that can generate the back-and-forth movement is assigned by considering the remaining battery power. Additionally, in the modified RHTA algorithm, the maximum number of tasks was changed to gradually decrease until its value equals one to determine a good suboptimal solution within a reasonable computation time. This study presented two high-fidelity simulations to validate the performance of the proposed RHTA algorithm. First, the MATLAB simulation compared the performance of the proposed RHTA algorithm with two different algorithms (i.e., iterative mixed integer linear programming and greedy algorithms) for task reassignment. As a result, the proposed RHTA algorithm could achieve better solution optimality and less computation time. Second, the Gazebo simulation evaluated the overall system's ability to handle dynamic events when various UAVs stopped performing further missions at random moments. Finally, the challenging outdoor flight experiment was conducted by modifying the original mission area in the field, reading the battery status of the flying UAV in real-time, replacing the low battery, and reintroducing the recharged UAV.

This study can be expanded in the following directions in the future. First, while this study assumed that the mission area is convex to facilitate the generation of waypoints for the back-and-forth movement, further research is needed to allow operators to designate no-fly zone where coverage is unnecessary or where UAVs are not allowed to fly. Therefore, complex and irregular mission areas should be considered for a real environment. One of the methods that can be considered to generate waypoints even in such nonconvex mission areas requires an area decomposition strategy partitioning the large complex area into smaller simple areas. The proposed system can be extended and applied to more complicated and practical mission areas with these additional decomposition strategies. Second, in this study, the flight experiment was performed using only two hexacopters for whether the proposed system is sufficiently applicable even in outdoor environments through real hardware implementation. However, flight experiments with many UAVs are expensive and require a lot of time and effort to prepare many hardware platforms, but they can provide more valuable performance verification. Therefore, future research can be undertaken to perform flight experiments using three or more UAVs.

REFERENCES

[1] A. Otto, N. Agatz, J. Campbell, B. Golden, and E. Pesch, "Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey," *Networks*, vol. 72, no. 4, pp. 411–458, 2018.

[2] Y. Wang, T. Kirubarajan, R. Tharmarasa, R. Jassemi-Zargani, and N. Kashyap, "Multiperiod coverage path planning and scheduling for airborne surveillance," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 54, no. 5, pp. 2257–2273, Oct. 2018.
[3] Y. Zuo, R. Tharmarasa, R. Jassemi-Zargani, N. Kashyap, J. Thiyagalingam, and T. T. Kirubarajan, "MILP formulation for aircraft path planning in persistent surveillance," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 56, no. 5, pp. 3796–3811, Oct. 2020.
[4] J. R. Riehl, G. E. Collins, and J. P. Hespanha, "Cooperative search by UAV teams: A model predictive approach using dynamic graphs," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 47, no. 4, pp. 2637–2656, Oct. 2011.
[5] Y. Chen, D. Yang, and J. Yu, "Multi-UAV task assignment with parameter and time-sensitive uncertainties using modified two-part wolf pack search algorithm," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 54, no. 6, pp. 2853–2872, Dec. 2018.
[6] P. Maini, K. Sundar, M. Singh, S. Rathinam, and P. Sujit, "Cooperative aerial–ground vehicle route planning with fuel constraints for coverage applications," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 55, no. 6, pp. 3016–3028, Dec. 2019.
[7] T. M. Cabreira, L. B. Brisolara, and P. R. Ferreira Jr., "Survey on coverage path planning with unmanned aerial vehicles," *Drones*, vol. 3, no. 1, pp. 8–11, 2019.
[8] G. S. Avellar, G. A. Pereira, L. C. Pimenta, and P. Iscold, "Multi-UAV routing for area coverage and remote sensing with minimum time," *Sensors*, vol. 15, no. 11, pp. 27783–27803, 2015.
[9] T. M. Cabreira, C. Di Franco, P. R. Ferreira, and G. C. Buttazzo, "Energy-aware spiral coverage path planning for UAV photogrammetric applications," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3662–3668, Oct. 2018.
[10] A. V. Le, V. Prabakaran, V. Sivanantham, and R. E. Mohan, "Modified a-star algorithm for efficient coverage path planning in tetris inspired self-reconfigurable robot with integrated laser sensor," *Sensors*, vol. 18, no. 8, 2018, Art. no. 2585.
[11] J. Modares, F. Ghanei, N. Mastronarde, and K. Dantu, "UB-ANC planner: Energy efficient coverage path planning with multiple drones," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 6182–6189.
[12] B. D. Song, J. Kim, and J. R. Morrison, "Rolling horizon path planning of an autonomous system of UAVs for persistent cooperative service: MILP formulation and efficient heuristics," *J. Intell. Robot. Syst.*, vol. 84, no. 1, pp. 241–258, 2016.
[13] Y. Hong, S. Jung, S. Kim, and J. Cha, "Autonomous mission of multi-UAV for optimal area coverage," *Sensors*, vol. 21, no. 7, 2021, Art. no. 2482.
[14] J. C. Amorim, V. Alves, and E. P. de Freitas, "Assessing a swarm-GAP based solution for the task allocation problem in dynamic scenarios," *Expert Syst. Appl.*, vol. 152, 2020, Art. no. 113437.
[15] J. Tang, X. Chen, X. Zhu, and F. Zhu, "Dynamic reallocation model of multiple unmanned aerial vehicle tasks in emergent adjustment scenarios," *IEEE Trans. Aerosp. Electron. Syst.*, to be published, doi: 10.1109/TAES.2022.3195478.
[16] R. Liu, M. Seo, Y. Binbin, and A. Tsourdos, "Decentralized task allocation for multiple UAVs with task execution uncertainties," in *Proc. IEEE Int. Conf. Unmanned Aircr. Syst.*, 2020, pp. 271–278.
[17] R. Schacht-Rodríguez et al., "Mission planning strategy for multi-rotor UAV based on flight endurance estimation," in *Proc. IEEE Int. Conf. Unmanned Aircr. Syst.*, 2019, pp. 778–786.
[18] M. Valenti, B. Bethke, J. P. How, D. P. De Farias, and J. Vian, "Embedding health management into mission tasking for UAV teams," in *Proc. IEEE Amer. Control Conf.*, 2007, pp. 5777–5783.
[19] S. Ramasamy, J.-P. F. Reddinger, J. M. Dotterweich, M. A. Childers, and P. A. Bhounsule, "Cooperative route planning of multiple fuel-constrained unmanned aerial vehicles with recharging on an unmanned ground vehicle," in *Proc. IEEE Int. Conf. Unmanned Aircr. Syst.*, 2021, pp. 155–164.
[20] E. Hartuv, N. Agmon, and S. Kraus, "Spare drone optimization for persistent task performance with multiple homes," in *Proc. IEEE Int. Conf. Unmanned Aircr. Syst.*, 2020, pp. 389–397.
[21] M. Alighanbari, "Task assignment algorithms for teams of UAVs in dynamic environments," Ph.D. dissertation, Dept. Aeronaut. Astronaut. Massachusetts Inst. Technol., Cambridge, MA, USA, 2004.
[22] L. R. Rodrigues, J. P. Gomes, and J. F. Alcântara, "Embedding remaining useful life predictions into a modified receding horizon task assignment algorithm to solve task allocation problems," *J. Intell. Robot. Syst.*, vol. 90, no. 1, pp. 133–145, 2018.
[23] M. Quigley et al., "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, Kobe, Japan, 2009, vol. 3, no. 3.2, Art. no. 5.
[24] QGroundControl. [Online]. Available: http://www.qgroundcontrol.com/
[25] PX4. [Online]. Available: https://dev.px4.io/v1.9.0/en/simulation/
[26] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 912–926, Aug. 2009.
[27] GUROBI. [Online]. Available: https://www.gurobi.com/
[28] [Online]. Available: http://wiki.ros.org/mavros

Youkyung Hong received the B.S. degree in mechanical engineering from POSTECH, Pohang, South Korea, in 2010, and the M.S. and Ph.D. degrees in mechanical and aerospace engineering from Seoul National University, Seoul, South Korea, in 2012 and 2018, respectively.

She is currently a Senior Researcher with Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea. Her research interests include optimization and its application to unmanned vehicle systems.

Sunggoo Jung received the B.S. degree in mechanical and control engineering from Handong University, Pohang, South Korea, in 2014, and the Ph.D. degree in aerospace engineering from KAIST, Daejeon, South Korea, in 2020.

From 2020 to 2022, he was a Senior Researcher with Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea. He is currently a Postdoctoral Fellow with NASA Jet Propulsion Laboratory (JPL), Pasadena, CA, USA. His research interests include learning-based planning of autonomous vehicles.

Suseong Kim received the B.S. degree in mechanical engineering from Yonsei University, Seoul, South Korea, in 2010, and the Ph.D. degree in mechanical and aerospace engineering from Seoul National University, Seoul, South Korea, in 2017.

From 2017 to 2018, he was a Postdoctoral Researcher with Robotics and Perception Group, University of Zurich and ETH Zurich, Zurich, Switzerland. He is currently a Senior Researcher with Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea. His research interests include intelligent control and motion planning of autonomous aerial vehicles.

Jihun Cha received the B.S. degree from Myongji University, Yongin, South Korea, in 1993, and the M.S. and Ph.D. degrees from the Florida Institute of Technology, Melbourne, FL, USA, in 1996 and 2002, respectively, all in computer science.

He joined the Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea, in 2003 and currently serves as the Directing Manager with the Autonomous Flight Research Team. His research interests include autonomous flight and counter-UAS.