Received August 24, 2020, accepted September 5, 2020, date of publication September 28, 2020, date of current version November 11, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3027422

A Multi-Robot Coverage Path Planning Algorithm for the Environment With Multiple Land Cover Types

XIANG HUANG[1], **MIN SUN**[1,3], **HANG ZHOU**[1], **AND SHUAI LIU**[2]

[1]Institute of Remote Sensing and GIS, Peking University, Beijing 100871, China
[2]College of Engineering, Honghe University, Mengzi 661100, China
[3]Beijing Key Laboratory of Spatial Information Integration and Its Applications, Peking University, Beijing 100871, China

Corresponding author: Min Sun (sunmin@pku.edu.cn)

This work was supported in part by the Department of Sciences and Technology of the Xinjiang Production and Construction Corps, China, under Grant 2017DB005, in part by the Key Technologies Research and Development Program of China under Grant 2016YFD0300601, in part by the National Natural Science Foundation of China under Grant 41761079, and in part by the High-Performance Computing Platform of Peking University.

ABSTRACT Many scholars have proposed different single-robot coverage path planning (SCPP) and multi-robot coverage path planning (MCPP) algorithms to solve the coverage path planning (CPP) problem of robots in specific areas. However, in outdoor environments, especially in emergency search and rescue tasks, complex geographic environments reduce the task execution efficiency of robots. Existing CPP algorithms have hardly considered environmental complexity. This article proposed an MCPP algorithm considering the complex land cover types in outdoor environments to solve the related problems. The algorithm first describes the visual fields of the robots in different land cover types by constructing a hierarchical quadtree and builds the adjacent topological relations among the cells in the same and different layers in the hierarchical quadtree by defining shared neighbor direction based on Binary System. The algorithm then performs an approximately balanced task assignment to the robots considering the moving speeds in different land cover types using the azimuth trend method we proposed to ensure the convergence of the task assignment process. Finally, the algorithm improves Spanning Tree Covering (STC) algorithm to complete the CPP in the area where each robot belongs. This study used a classification image of the real outdoor environment to the verification of the algorithm. Results show that the coverage paths planned by the algorithm are reasonable and efficient and its performance has obvious advantages compare with the current mainstream MCPP algorithm.

INDEX TERMS Coverage path planning algorithm, complex geographic environments, emergency search, multi-robot CPP, multiple land cover types, terrain sub-division.

I. INTRODUCTION

The coverage path planning (CPP) problem is a task wherein a robot or robots determines a path to pass over all points of an area or volume of interest while avoiding obstacles [1]. With the continuous development of unmanned aerial vehicle and robot technology, CPP algorithms have been widely used as one of the core algorithms to assist robots or unmanned aerial vehicles in performing specific tasks, including area cleaning [2], fire monitoring [3], [4], surveillance, search and rescue tasks [5]–[7], and mapping and model reconstruction [8]–[10].

When using robots to perform large-scale emergency search and rescue tasks in outdoor environments, the use of a single robot exposes many problems, such as long time consumption and insufficient battery life. Therefore, with the development of synchronic control theory, communication technology, and intelligent hardware, utilizing coordinated multiple robots to solve the above problems has attracted increasing attention in the industry; thus, the multi-robot CPP (MCPP) problem has become a research focus [11].

In the related works on the MCPP problem, most studies have only considered unmanned aerial vehicle (UAV)

The associate editor coordinating the review of this manuscript and approving it for publication was Okyay Kaynak.

applications and ignored the effect of complex environments in the task area during CPP. Other studies have considered robot applications but had a small study area and a single cover type, such as a sweeping robot. Therefore, robots are always assumed to have a constant visual field and moving speed in the entire area in existing MCPP algorithms (The concepts of visual field and moving speed will be described in detail in Section III). Thus, after the area is divided by the approximate cell decomposition method, the cells in the entire task area have the same size.

However, the performance efficiency of robots in emergency search and rescue tasks is seriously affected due to the complexity and variation of the environmental surface, so that a robot may have different visual fields and moving speeds in different environments. For example, robots in the forest area have a considerably smaller visual field than those in the Gobi area; those in the mountain area have a remarkably slower moving speed than those in the flat area. The existing MCPP algorithms cannot provide a satisfactory solution for the aforementioned cases.

This article proposes an algorithm called MCPP-MLCT to solve MCPP problems in the environment with Multiple Land Cover Types (MLCT). The algorithm uses a hierarchical quadtree to describe the visual fields of the robots in different land cover types, with different layers in the quadtree representing different visual fields. Taking the outdoor search and rescue task as an example, the visual field of the robot and the size of corresponding cells will be small when the environment is complex. The algorithm establishes the adjacent topology relationship among cells in the same and different layers in the hierarchical quadtree by defining shared neighbor direction based on Binary System, to improve the efficiency of task assignment and path planning. We use the azimuth trend method to extend the assignment process to the unassigned area to ensure the convergence of the task assignment process, and balance the amount of task for each robot considering the variation of moving speed varies with land cover type in the process of task assignment. Finally, an improved Spanning Tree Covering (STC) method is used in the sub-area of each robot to complete the CPP.

Section II reviews the related works of the MCPP problem. Section III focuses on the MCPP-MLCT algorithm and its specific implementation. Section IV uses the real land classification data to verify the proposed algorithm and discusses the experimental results and related work of the algorithm. Section V analyzes the time complexity and shortcomings of the algorithm. Section VI summarizes the research content and provides an outlook for further work.

II. RELATED WORKS

CPP algorithms can be roughly divided into the following two types: geometry-based and grid-based, or exact cellular decomposition and approximate cellular decomposition. The general idea of the geometry-based algorithms is first dividing the area into simple sub-areas, such as non-convex, and then using a simple method (e.g., zigzag path) to complete the CPP in each sub-area. Meanwhile, grid-based algorithms divide the area into regular cells (usually squares) according to the coverage capabilities of robots or unmanned aerial vehicles, approximate these cells to points, and then connect them into continuous and non-repeated (or low-repeated) paths. The representative geometry-based algorithms include Trapezoidal decomposition algorithm [12], Boustrophedon decomposition algorithm [13], and Morse-based cellular decomposition algorithm [14]. The wavefront algorithm and the STC algorithm proposed in references [15] and [16] belong to the typical grid-based algorithms.

Cluster control is one of the key technologies in MCPP. Generally speaking, the cluster control technology can be divided into three types: centralized, decentralized, and the combination of centralized and decentralized. In recent years, many cluster control algorithms and applications based on the combination of centralized and decentralized methods have been proposed. Reference [17] proposed a framework includes a new clustering behavior that causes agents in the swarm to agree on attending a group and allocating a leader for each group, in a decentralized and local manner. The leader of each group employs a vision-based goal detection algorithm to find and acquire the goal in a cluttered environment. As soon as the leader starts moving, each member is enabled to move in the same direction by staying coordinated with the leader and maintaining the desired formation pattern. Considering the noticeable increase in the number of low-cost mobile robots readily available, Reference [18] reported one possible embodiment of such a technology—an integrated combination of hardware and software—designed to enable the assembly and the study of swarming in a range of general-purpose robotic systems. This is achieved by combining a modular and transferable software toolbox with a hardware suite composed of a collection of low-cost and off-the-shelf components. The developed technology can be ported to a relatively vast range of robotic platforms. Because cluster control is not our focus topics, we adopted the centralized method in order to reduce complexity in our works.

Compared with single-robot CPP (SCPP), MCPP can direct multiple robots to complete tasks in parallel, thus reducing time consumption and improving execution efficiency in tasks. Overall, the MCPP problem can be optimized from four different perspectives in terms of different application scenarios: (1) the minimum total cost to complete the area coverage, such as the total path length or total coverage time; (2) the minimum turn times along the path; (3) the minimum repeated coverage in the area; (4) the balanced task cost of each robot, which is generally measured by the path length or coverage time. One or more of the aforementioned optimizations must be achieved to maximize coverage efficiency.

Minimizing the total cost, such as the total path length or the total coverage time, can reduce energy consumption in the MCPP problem. An algorithm proposed in reference [2] converted the MCPP problem into the flow network through the exact cellular decomposition method. The time cost of each edge in the flow network was the sum of the coverage time of the current cell and the shift time in adjacent cells. The minimum cost path from the start node to the end node through the flow network was determined by a network search algorithm in accordance with the time cost, which could visit each node in this flow network without repetition.

Similarly, additional turns produce additional time and energy consumption. Reference [19] proposed a meta-heuristic algorithm called Harmony Search to reduce the turn times in the robot path, ensuring low energy consumption and minimum coverage time.

Repeated coverage also increases time and energy consumption. The ideal situation is to cover the entire area without repetition. Reference [20] proposed a multi-robot coverage algorithm that minimized repeated coverage. This algorithm used the Boustrophedon cellular decomposition algorithm to decompose the area into multiple simple cells and a method of task assignment to robots. Moreover, the algorithms based on STC could minimize coverage repetition [16].

The MCPP problem is a coordinated process of multiple robots. Thus, the balancing task cost of each robot could also improve the execution efficiency. Reference [21] proposed a multi-robot coordinated CPP algorithm called MSTC based on the STC algorithm, which completed the coverage by spiraling the same spanning tree using multiple robots. This algorithm completed the area coverage by multiple robots to some extent. However, a certain robot must complete almost all the coverage tasks in extreme cases due to the effects by the initial positions of robots. Approximately one year later, the author proposed an improved algorithm [22], which changed the connection method of the spanning tree according to the initial positions of multiple robots, to ensure that the task cost of each robot is as balanced as possible. However, the algorithm cannot fully balance the task cost of each robot.

A pattern-based genetic algorithm proposed in reference [23] gridded the task area according to the coverage capability of robots and considered the walking and turning times of robots. This algorithm minimized the maximum coverage time of all robots to complete the task but has high computation costs.

Reference [24] proposed a dividing areas algorithm for optimal MCPP problem (DARP) in 2017 and introduced a new optimization method for the task assignment problem. The proposed technique transformed the MCPP problem into several CPP problems, and their solutions constitute the optimal MCPP solution. This technique alleviates the original MCPP explosive combinatorial complexity and could achieve the aforementioned (1), (3), and (4) optimizations simultaneously. The DARP algorithm effectively completes the optimization of the task assignment through the aforementioned technique and provides a balanced task assignment considering the initial position of robots.

Based on this algorithm, an auction-based spanning tree coverage algorithm for motion planning of cooperative robots (A-STC) was proposed [25], in which the auction algorithm was used to complete the task assignment and CPP. Moreover, the A-STC algorithm considered the effect of the path length from the different kinds of cells linked to the spanning tree. Therefore, selecting the optimal connection method of spanning trees is necessary to generate the optimal coverage path.

However, none of the existing studies has considered the complexity of the surface of the geographical environment in the task area. Especially in tasks, such as emergency search and rescue, the efficiency of robots or unmanned aerial vehicles is seriously affected by various surface features. Therefore, considering the impact of different land cover types in outdoor environmental conditions is of considerable importance.

III. METHODS

A. DEFINATION OF MOVING SPEED AND VISUAL FIELD

As shown in Figure 1, we define the concept moving speed to describe the speed of the robot along it's moving direction, and define the concept of visual field to describe the visual range perpendicular to the moving direction.

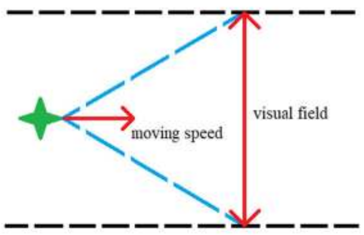


FIGURE 1. The schematic diagram of visual field and moving speed of a robot.

The value of the visual field and the moving speed are determined not only by robots' performance, but also by its environment. When a robot is in a complex environment, generally its moving speed slows down and its visual field decreases.

B. CONSTRUCTION OF HIERARCHICAL QUADTREE

In order to describe the visual fields of robots in different land cover types, we establish a hierarchical data structure called hierarchical quadtree. As shown in the Figure 2, the top layer of the hierarchical quadtree is the division of the whole task area by a set of square grids with the same size, which are called cells. Starting from this layer, by uses recursion manner, the cells of next level are obtained by dividing the current level cells into four parts.

C. OVERVIEW OF MCPP-MLCT ALGORITHM

This article proposes the algorithm called MCPP-MLCT to solve the MCPP problem in multiple land cover types (MLCT).

In our algorithm, the classification image data of the task area and the empirical parameters, including the visual fields and moving speeds of robots in different land cover types, are known. The algorithm can obtain balanced task assignments

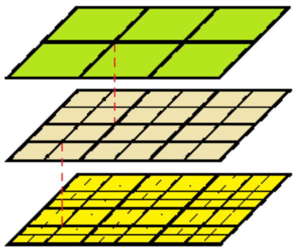


FIGURE 2. The schematic diagram of the construction of hierarchical quadtree.

and continuous coverage paths of robots, which are given the initial positions in the task area. The algorithm mainly includes the following three parts shown in Figure 3.

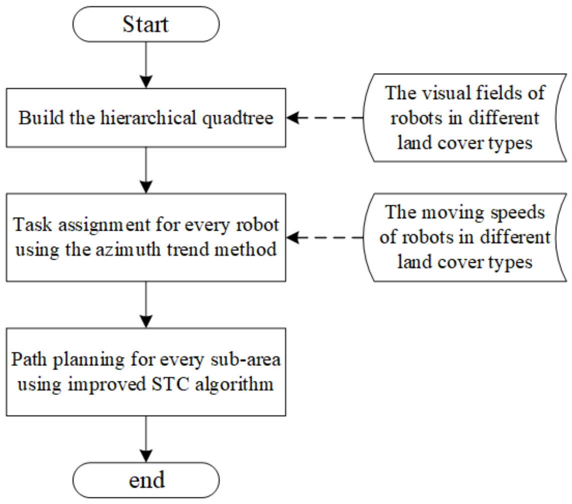

Start
Build the hierarchical quadtree
The visual fields of robots in different land cover types
Task assignment for every robot using the azimuth trend method
The moving speeds of robots in different land cover types
Path planning for every sub-area using improved STC algorithm
end

FIGURE 3. The main steps of MCPP-MLCT algorithm.

First, a hierarchical quadtree is built to describe the different visual fields of the robots in different land cover types, in which the different cell sizes in different layers represent the different visual fields of robots. The adjacent topological relationship among the cells in the same and different layers is established in the hierarchical quadtree to improve the efficiency of task assignment and path planning.

Then, task assignment is completed on the basis of the hierarchical quadtree. In this process, we propose the azimuth trend method to extend the assignment process to the unassigned area and consider the connectivity of each sub-area in the meantime to ensure the convergence of the assignment process. Moreover, the algorithm takes the ratio of the cell size to the moving speed in the corresponding cover type as the task cost of the cell to balance the task cost of each robot.

Finally, we revise the existing STC algorithm in the area of each robot and complete the CPP of each robot to adapt to the complex situation of variable cell size. The following content elaborates on the MCPP-MLCT algorithm.

D. BUILDING THE HIERARCHICAL QUADTREE

First of all, we build a hierarchical quadtree Tr based on the classification image of the task area to describe the different visual fields of the robots in different cover types. The parameter l is the number of layers in the quadtree, $C_{i,r,c}$ is the cell at the rth row and cth column in the ith layer in Tr, Cs_i is the cell size in the ith layer, $Cv_{i,r,c}$ is the land cover type of $C_{i,r,c}$, $Fd_{Cv_{i,r,c}}$ is the visual field of the robot in $Cv_{i,r,c}$, and $Sd_{i,r,c}$ is the moving speed of the robot in $C_{i,r,c}$, which is affected by $Cv_{i,r,c}$. Each cell has neighbor cells in the same and different layers. We use $Nc_{i,r,c}$ to represent the set of the neighbor cells of $C_{i,r,c}$.

1) CONSTRUCTING A HIERARCHICAL QUADTREE WITH ROBOT VISUAL FIELDS AS PARAMETERS

As previously mentioned, we consider the situation of outdoor emergency search and rescue. The visual fields and moving speeds of robots are affected by different land cover types. Thus, we use a hierarchical quadtree to express different visual fields in different land covers. Normally, a highly complex land cover corresponds to a small visual field of robots and cell size in the quadtree. By contrast, a simple land cover corresponds to a large visual field of robots and a large cell size in the quadtree.

TABLE 1. Example of the visual field approximation.

Land cover types	Original visual field (m)	Approximate visual field (m)	Cell size in the quadtree (m)
Type 1	5	5	10
Type 2	7	5	10
Type 3	21	20	40

The cell size of adjacent layers in a quadtree structure should be twice according to Figure 2. Thus, the robot visual fields in different land covers must satisfy the relationship of $2^k (k = 0, 1, 2, \ldots)$. However, the truly visual fields of robots in different land covers cannot satisfy this relationship due to the abstract factors of the complexity and diversity of the environment surface.

We approximate the original visual fields to simplify this problem. Fd_{max} and Fd_{min} are used to represent the maximum and minimum values of the approximate visual fields, respectively. Based on the minimum visual field Fd_{min}, we approximate the visual fields Fd_t in other land cover types as $Fd_t = Fd_{min}2^k$, where k is the maximum value satisfied by $Fd_{min}2^k \leq Fd_t$. Table 1 shows that visual field 5 in Type 1 is the minimum value; thus, the original visual field 7 in Type 2 is approximately $5 \times 2^0 = 5$, and the visual field 21 in Type 3 is approximately $5 \times 2^2 = 20$.

We divide the task area into a hierarchical quadtree based on the approximated visual fields. Different layers correspond to different visual fields and land covers, thereby constructing a hierarchical quadtree for the entire area.

A supplementary note is presented as follows. The final path planning of the algorithm is based on the improved STC algorithm with cell sizes twice the robot visual fields. Therefore, the cell size in each layer in the hierarchical quadtree is twice the corresponding approximate visual field, as shown in Table 1.

Based on these cell sizes, we build the hierarchical quadtree. First, calculate the number of layers by the following formula:

$$l = 1 + log_2 \frac{Fd_{max}}{Fd_{min}} \quad (1)$$

Then, the top layer of the hierarchical quadtree is constructed. The task area is then divided with twice of the maximum approximate visual field Fd_{max}. The number of cells is $m \times n$, where m and n are the minimum numbers that all cells in the top layer can completely cover the study area. If the cells exceed the task area, then we simply set their cover type to null, which will be detailed in the next paragraphs.

From the top layer down, each inferior layer is obtained by the quadtree partition of the upper layer until the cell size is equal to twice of Fd_{min}. Each layer in the quadtree corresponds to one visual field, and the visual fields in all land cover types in the task area can be found in the quadtree. However, no one-to-one correspondence exists between layers in the quadtree and the land cover types of the task area.

For example, a three-layer quadtree can be constructed on the basis of the data in Table 1, with cell sizes of 40, 20, and 10. The first layer corresponds to Type 3, the second layer has no corresponding land cover type, and the third layer corresponds to Type 1 and Type 2.

Each layer in the quadtree can fully cover the task area, but the task area must be covered only once during the search process. Therefore, only parts of the cells in the quadtree must be associated with the actual land cover types to avoid the duplication and redundancy cover by different layer cells. Thus, we divide the cells in the hierarchical quadtree into two kinds: the active cells and the auxiliary cells. The active cells have a real land cover type and all of active cells can completely cover the task area without repetition (as shown in the case example in Figure 4). The auxiliary cells are the remaining cells whose land cover types are set null. Thus, we can distinguish the two kinds of cells by the setting of the land cover types.

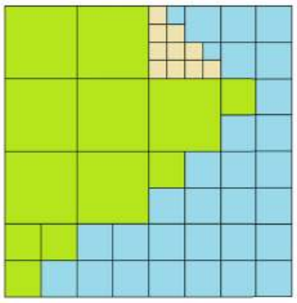


FIGURE 4. Example of all pre-cells covering the area without repetition.

The areas occupied by different land cover pixels in $C_{l,r,c}$ are separately cumulated for each cell $C_{l,r,c}$ in the bottom layer L_l in the quadtree. The largest area type is regarded as the land cover type of $C_{l,r,c}$, which is expressed with $Cv_{l,r,c}$. If two or more cover types with the largest area are available, then the most complex one is selected.

Starting from the penultimate layer, the cells are integrated from the low layer to the upper layer following these rules. The current cell $C_{i,r,c}$ has four child cells, $C_{i+1,r*2,c*2}$, $C_{i+1,r*2+1,c*2}$, $C_{i+1,r*2,c*2+1}$, and $C_{i+1,r*2+1,c*2+1}$. If the land cover types, $Cv_{i+1,r*2,c*2}$, $Cv_{i+1,r*2+1,c*2}$, $Cv_{i+1,r*2,c*2+1}$, and $Cv_{i+1,r*2+1,c*2+1}$, of the four child cells are the same but not null, moreover, the four child cell sizes are smaller than twice of the approximate visual field of robots in the corresponding land cover type, then we integrate the four child cells upward by setting the land cover type $Cv_{i,r,c}$ to its parent cell and the land cover types of the four child cells to null. If the four child cells have different cover types or the size of the four child cells is equal to twice the approximate visual field of robots in the corresponding land cover type, then the situation is maintained and processing is stopped.

The above operations are repeated until the top layer of the hierarchical quadtree is processed. Figure 4 shows that all the active cells can completely cover the task area without repetition. The details of the algorithm are shown in Algorithm 1.

Algorithm 1 Build the Hierarchical Quadtree Tr

```
1  calculate layer number $l = 1 + log_2 Fd_{max} / Fd_{min}$;
2  add the top layer $L_1$ into $P$, $Sc_1 = Fd_{max} \times 2$;
3  $i = 2$;
4  for $i \leq l$ do
5      add $L_i$ into Tr, $Sc_i = Sc_{i-1}/2$;
6  end
7  for each $C_{l,r,c}$ in $L_l$
8      $Cv_{l,r,c}$ = the land cover type with maximum area
       in $C_{l,r,c}$;
9  end
10  $i = l - 1$;
11  for $i \geq 1$ do
12      for $C_{i,r,c}$ in $L_i$ do
13          if the land cover types of 4 child cells of $C_{i,r,c}$
        are the same but not null and
        $Sc_{i+1,r*2,c*2} < Rg_{Ct_{i+1,r*2,c*2}}$ then
14              $Cv_{i,r,c} = Cv_{i+1,r*2,c*2}$;
15              $Cv_{i+1,r*2,c*2} = null$;
16              $Cv_{i+1,r*2+1,c*2} = null$;
17              $Cv_{i+1,r*2,c*2+1} = null$;
18              $Cv_{i+1,r*2+1,c*2+1} = null$;
19          end if
20      end
21      $i = i - 1$
22  end
```

2) BUILDING THE ADJACENT TOPOLOGICAL RELATIONSHIP FOR THE ACTIVE CELLS

We build the adjacent topological relationship for the active cells in the hierarchical quadtree, which are described by a set of neighbor cells for the corresponding cell. This relationship is built to reduce the complexity of the task assignment and ensure the continuity of the results by the task assignment process.

We introduce the shared neighbor direction concept to improve the efficiency of searching neighbor cells in different layers in the hierarchical quadtree. This concept could accelerate the search process by using simple binary operations, such as "AND" or "OR." Related definitions of shared neighbor direction are as follows.

In the hierarchical quadtree, each cell, except the cells in the bottom layer, has four child cells in the lower adjacent layer. Correspondingly, each cell, except the cells in the top layer, has a parent cell in the upper layer. The shared neighbor direction $Dn_{i,r,c}$ of the cell $C_{i,r,c}$ is defined to show the directions where the cell and its parent cell shared the neighbor cells.

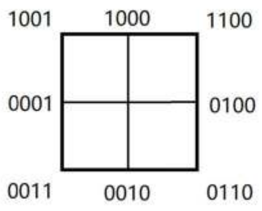


FIGURE 5. The schematic diagram of the binary representation of the neighbor directions.

As shown in Figure 5, we initially use four binary numbers to represent four neighbor directions of a cell: the left direction is 1, which is 0001 in binary; the bottom direction is 2, which is 0010 in binary; and the right and top directions are 0100 and 1000, respectively.

We assume that one child cell is $C_{i,r,c}$ and the parent cell is $C_{i-1,r/2,c/2}$ in Figure 5. When $C_{i,r,c}$ is located at the top left of $C_{i-1,r/2,c/2}$, the parent and child cells share neighbor cells on the top and left directions, which are respectively represented by 0001 and 1000 in binary. The logical operation "OR" result is 1001, which means the value of the shared neighbor direction $Dn_{i,r,c}$ is 9. When the child cell is located at other directions, the values of the shared neighbor direction are as follows: the top-right is $Dn_{i,r,c} = 4|8 = 12$, the bottom-left is $Dn_{i,r,c} = 1|2 = 3$, and the bottom-right is $Dn_{i,r,c} = 2|4 = 6$. We particularly use 0 when a cell has no shared neighbor direction with its parent cell.

During the process of searching neighbor cells, search judgment between two adjacent layers can be quickly made in accordance with the shared neighbor direction value. For example, if the value is 9, then we can immediately determine that the neighbor cells in the upper layer are the neighbor cells of the parent cell on the top and left directions. Similarly, the neighbor cells of the parent cell in the lower layer contain the neighbor cells of the child cell on the top and left directions.

For the cells of non-adjacent layers, the shared neighbor direction value can be calculated by logical operation "AND" the shared neighbor direction of the cells in the layers between the two cells. An example is given in Figure 6 to understand this relationship intuitively.

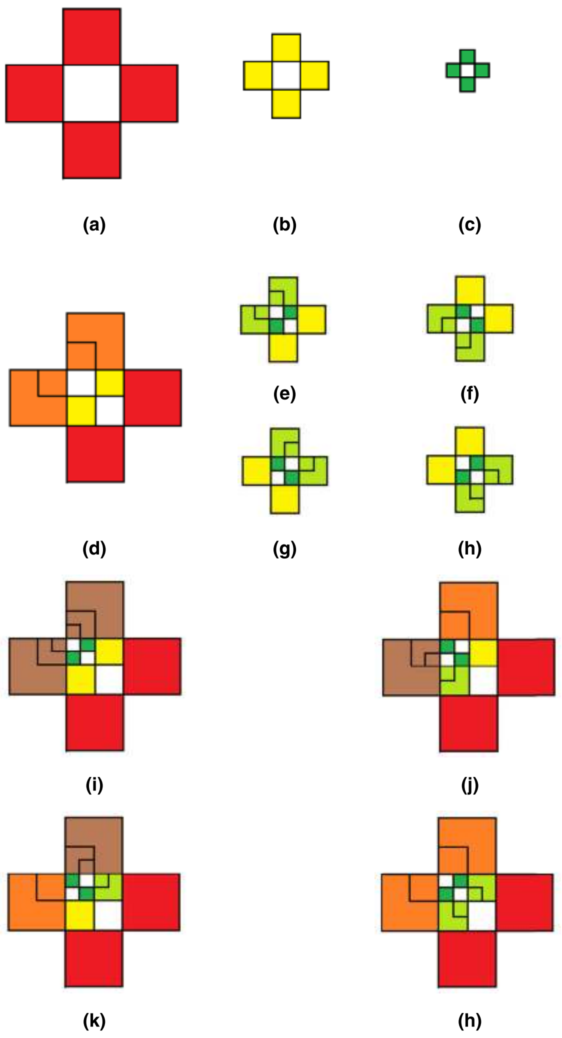


FIGURE 6. The schematic diagram of the shared neighbor cells in the different layers.

In Figure 6, the cells in three different layers are given in 6(a), 6(b), and 6(c), with the neighbor cells in the same layer respectively represented by red, yellow, and green.

When 6(b) is located at the top left of 6(a), the shared neighbor cells of 6(a) and 6(b) are represented by orange in 6(d), and the shared neighbor direction value is $0001||1000 = 1001$.

The graphs in 6(e), 6(f), 6(g), and 6(h) represent the four different conditions of shared neighbors. 6(c) is located at different positions in 6(b), and the shared neighbor direction values are 1001, 0011, 1100, and 0110.

6(i), 6(j), 6(k), and 6(h) describe the shared neighbor cells of 6(a) and 6(c) in the non-adjacent layers obtained by the overlay of 6(a), 6(b), and 6(c). The shared neighbor cells are shown in brown, and the shared neighbor direction values are $1001\&1001 = 1001$, $1001 \&0011 = 0001$, and $1001\&1100 = 1000$. 6(h) shows no shared neighbor cells because the shared neighbor direction value is $1001\&0110 = 0000$.

Based on the above definition and calculation of the shared neighbor direction, we can quickly finish building the adjacent topological relationship among the active cells in the entire hierarchical quadtree. For each active cell $C_{i,r,c}$, the search process to obtain $Nc_{i,r,c}$ (the set of neighbor cells) is divided into three parts. The specific implementation is given in Algorithm 2.

For the cells in the same layer, we must consider the land cover type of $C_{i,r-1,c}$, $C_{i,r+1,c}$, $C_{i,r,c-1}$, or $C_{i,r,c+1}$, which are the four neighbor cells of the current cell $C_{i,r,c}$, and add the neighbor cell into $Nc_{i,r,c}$ if its land cover type is not null.

Recursive search is used for searching the neighbor cells in the layers above or below the current layer.

For the cells above the current layer, we initialize the current recursive layer cell C_t and make it equal to $C_{i,r,c}(t = i)$. The shared neighbor direction Dn_{total} between the current recursive layer cell C_t and the cell $C_{i,r,c}$ is equal to $Dn_{i,r,c}$. The shared neighbor direction of C_t and the corresponding upper parent cell C_{t-1} are Dn_t. Thus, the shared neighbor direction of C_{t-1} and $C_{i,r,c}$ can be determined by the binary operation "AND" (i.e., $Dn_{total} = Dn_{total}\&Dn_t$). Consequently, the neighbor cells of $C_{i,r,c}$ in the $t-1$-th layer is the shared neighbor cells whose land cover types are not null, with C_{t-1}. Finally, the function *findUpperNeighborCells*$(Nc_{i,r,c}, C_t)$ is used to recurse upward through $C_t = C_{t-1}$ until $Dn_{total} = 0$ or reach the top layer.

For the cell below the current grid layer, we initialize the current recursive layer cell C_t and make it equal to $C_{i,r,c}(t = i)$. The shared neighbor direction Dn_{total} between the current recursive layer cell C_t and the cell $C_{i,r,c}$ is equal to $Dn_{i,r,c}$. The shared neighbor direction between C_t and the corresponding lower child cell C_{t+1} is Dn_{t+1}. Thus, the shared neighbor direction of C_{t+1} and $C_{i,r,c}$ can be determined by the binary operation "AND" as $Dn_{total} = Dn_{total}\&Dn_{t+1}$. Consequently, the neighbor cells of $C_{i,r,c}$ at the lower layer is the shared neighbor cells of C_{t+1} whose land cover types are not null. Different from the upward search, four corresponding child cells are found in the lower layer of C_t in the process of downward search. Thus, this searching process must be individually executed. The function *findLowerNeighborCells*$(Nc_{i,r,c}, C_t)$ is used to recurse downward by parameter C_t (i.e., let $C_t = C_{t+1}$) until $Dn_{total} = 0$ or the penultimate layer is reached.

Through the three abovementioned steps, the process of searching the neighbor cell set $Nc_{i,r,c}$ of the cell $C_{i,r,c}$ is completed (as shown Figure 7). All the active cells are traversed, and a complete topological adjacency relationship is established.

Algorithm 2 Build Adjacent Topological Relationship of Active Cells

```
1  $i = 1$;
2  for $i \le l$ do
3    for $C_{i,r,c}$ in $L_i$ do
4      switch location of $C_{i,r,c}$ in $C_{i-1,r,c}$ do
5        case top-left then $Dn_{i,r,c} = 1|8 = 9$;
6        case top-right then $Dn_{i,r,c} = 4|8 = 12$;
7        case bottom-left then $Dn_{i,r,c} = 1|2 = 3$;
8        case bottom-right then $Dn_{i,r,c} = 2|4 = 6$;
9      end switch
10   end
11   $i = i + 1$;
12 end
13 $i = 1$;
14 for $i \le l$ do
15   for $C_{i,r,c}$ in $L_i$ do
16     if $Cv_{i,r,c}$ is null then continue; end if
17     // neighbor cells in the same layer
18     if $Cv_{i,r-1,c}$ <> null then add $C_{i,r-1,c}$ into $Nc_{i,r,c}$; end if
19     if $Cv_{i,r+1,c}$ <> null then add $C_{i,r+1,c}$ into $Nc_{i,r,c}$; end if
20     if $Cv_{i,r,c-1}$ <> null then add $C_{i,r,c-1}$ into $Nc_{i,r,c}$; end if
21     if $Cv_{i,r,c+1}$ <> null then add $C_{i,r,c+1}$ into $Nc_{i,r,c}$; end if
22     // neighbor cells in the upper layers
23     $t = i, C_t = C_{i,r,c}$;
24     $Dn_{total} = Dn_{i,r,c}$;
25     //findUpperNeighborCells($Nc_{i,r,c}, C_t$)
26     if $t > 1$ then
27       $Dn_{total} = Dn_{total}\&Dn_t$;
28       if $Dn_{total} > 0$ then
29         find $Nc_{i,r,c}$ by $Dn_{total}$ in 4 neighbor cells of $C_{t-1}$;
30         findUpperNeighborCells($Nc_{i,r,c}, C_{t-1}$);
31       end if
32     end if
33     // neighbor cells in the lower layers
34     $t = i, C_t = C_{i,r,c}$;
35     $Dn_{total} = Dn_{i,r,c}$;
36     // findLowerNeighborCells($Nc_{i,r,c}, C_t$)
37     if $t < n$ then
38       $Dn_{total} = Dn_{total}\&Dn_{t+1}$;
39     if $Dn_{total} > 0$ then
40       find $Nc_{i,r,c}$ by $Dn_{total}$ in 4 neighbor cells of $C_{t+1}$;
41       findLowerNeighborCells($Nc_{i,r,c}, C_{t+1}$);
42     end if
43   end if
44 end
45 $i = i + 1$;
46 end
```

E. TASK ASSIGNMENT BASED ON AZIMUTH TREND METHOD

A highly effective optimization method is given in the DARP algorithm to optimize the MCPP problem. This method can decompose the MCPP problem into task assignment optimization problem and multiple SCPP problems under certain constraints and obtain an approximate optimal solution

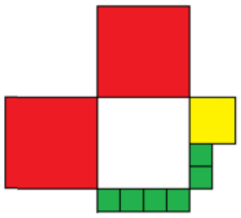


FIGURE 7. The schematic diagram of the neighbor cells in the same layer and different layers. The neighbor cells in the same layer have the same size as the center cell.

for the MCPP problem, considerably reducing optimization complexity [24].

Given a rectangular coverage area D, s represents the number of robots, R_j represents the j-th robot, Sp_j represents the start position of R_j, A_j represents the area assigned to robot R_j, St_j represents the spanning tree of A_j, P_j represents the coverage path obtained by spiral A_j, and T_j represents the coverage time of A_j. The results of the task assignment must meet the following conditions:

(1) $A_1 \cup A_2 \cup \ldots \cup A_s = D$;

(2) $A_j \cap A_k \rightarrow \emptyset, \forall j, k \epsilon\ [1, s]\, and j \neq k$;

(3) $|T_1| \approx |T_2| \approx \ldots \approx |T_s|$;

(4) $St_j is connected, \forall j \epsilon\ [1, s]$;

(5) $Sp_j \in A_j, \forall j \epsilon\ [1, s]$.

Among these constraints, the following conditions are ensured: (1) the task area can be completely covered, (2) the repeated coverage is as small as possible to reduce the energy and time consumption, (3) the task distribution is balanced, (4) the area assigned to the same robot is continuous to avoid the energy and time consumption used to move between the discontinuous areas, and (5) the robots can perform tasks immediately when turned on, avoiding the energy and time consumption from the initial position to the coverage area.

Although the cell sizes are various in our method, we could still use the five constraints to complete our optimal task assignment based on the hierarchical quadtree structure.

The task area is covered by the active cells without repetition in the hierarchical quadtree. We assign all the active cells to the robots without duplication to meet constraints (1) and (2). We take the ratio of the cell size to the moving speed of the robot in the corresponding land cover type as the coverage cost of the cell to calculate and balance the task cost of each robot to meet constraint (3). We propose the azimuth trend method to meet constraints (4) and (5). This method, extends the assignment to the active cells in the unassigned area, ensuring the convergence of the task assignment process. The specific algorithm is given as follows.

Let Ca_j represent the center of the sub-area A_j assigned to robot R_j and Na_j represent the neighbor cell set of the area as shown in Figure 8.

We initialize the area A_j of each robot R_j and add the cell C_j, where the initial position S_j of the robot is located, into area A_j. The neighbor cell set Nc_j of the cell C_j is assigned the neighbor cell set Na_j of area A_j, that is, $Na_j = Nc_j$.

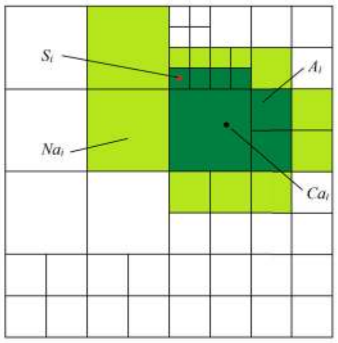


FIGURE 8. The schematic diagram of main parameters in the task assignment process.

The assignment of cells is then performed by the iteration process. The break condition of the process is that all cells are assigned to the robots, that is, the active cell set A_{un} is null. The following steps are performed in each iteration.

1) DETERMINE THE ROBOT R_t TO SELECT AN ACTIVE CELL

In each task assignment process, we try to assign a cell to the robot with the least amount of tasks, so as to achieve the goal of balanced allocation. Therefore, we define the variable Qa_j to measure the task amount of robot R_j, which can be used to describe the time taken by the robot to complete its tasks approximately. Qa_j is calculated by the following formula:

$$Qa_j = \sum_{C_{i,r,c} \in A_j} \frac{2Sc_i}{Sd_{i,r,c}} \tag{2}$$

We select the robot R_j as the robot R_t if R_j has the minimal task cost Qa_j that covers its belonging area A_j. By comparing Qa value of different robots, we assign the active cell to the robot with the minimal task cost (the minimal time to complete its task).

2) SELECT THE TO-BE-ASSIGNED CELL C_t BASED ON THE AZIMUTH TREND METHOD

Here, we propose the azimuth trend method to finish the assignment process. This method can extend the assignment process to the unassigned area and consider the connectivity of each sub-area in the meantime to ensure the convergence of the assignment process. This method will be introduced in details next.

The cells in Na_t, which is the neighbor cell set of the area A_t of robot R_t, are divided into the following two categories: the cell set $Na_{t,as}$, which has been assigned to other robots, and the cell set $Na_{t,un}$, which is not assigned to any robot. The selection rules of the to-be-assigned cell C_t are as follows:

(1) The cells in $Na_{t,un}$ take precedence over those in $Na_{t,as}$.

(2) Cell selection rules in $Na_{t,un}$.

For all cells in $Na_{t,un}$, the corresponding cost value is first calculated in accordance with the following formula. The to-be-assigned cell is then selected in cost ascending order:

$$cost_{un,i} = Dis(C_{t,un,i}, Ca_t) - \frac{\sum_{j \neq t} Dis(C_{t,un,i}, Ca_k)}{n-1} \tag{3}$$

where Ca_k denotes the center of the area A_k, $C_{t,un,i}$ denotes the cell in $Na_{t,un}$, and $Dis(C_i, Ca_j)$ denotes the distance from the center of cell C_i to Ct_k.

(3) Cell selection rules in $Na_{t,as}$.

Although this assignment process has low complexity, the boundaries of some assigned areas are easily connected before the completion of the assignment process because the initial robot position distribution does not have any regularity. For example, when the A area is connected to the B area, one cell at the boundary may be assigned from A to B. However, the area may be assigned from B to A in the next step. Therefore, the assignment process will fall into an endless iteration if no constraints are assigned. We propose the azimuth trend method to solve this problem. This method can extend the entire assignment process to the unassigned area and converge the assignment process normally.

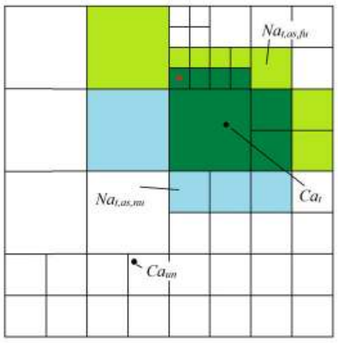


FIGURE 9. The neighbor cell distribution of azimuth trend method.

According to the relationship between the directions that the cell points to Ca_{un} and Ca_t (Figure 9), the cells in $Na_{t,as}$ are divided into $Na_{t,as,nu}$ (which represents the cells near to Ca_{un}) and $Na_{t,as,fu}$ (which represents the cells far from Ca_{un}). The cells in $Na_{t,as,nu}$ take precedence over those in $Na_{t,as,fu}$, which are to be selected in the assignment process.

However, the following two types of cells cannot be selected: those where the initial position of the corresponding robot is located and those that cause discontinuities in the corresponding area after deletion.

When several cells are in $Na_{t,as,nu}$ or $Na_{t,as,fu}$ simultaneously, the corresponding cost value of the cells is calculated in accordance with the following cost function. The to-be-assigned cell is then selected in ascending order of this cost.

$$cost_{as,i} = Dis(C_{t,as,i}, Ca_t) \tag{4}$$

where $C_{t,as,i}$ denotes a cell in $Na_{t,as,nu}$ or $Na_{t,as,fu}$.

The specific implementation of the three aforementioned rules refers to Algorithm 3.

3) ADD THE TO-BE-ASSIGNED CELL C_t INTO AREA A_t

Whether the cell has been assigned to other robots, this step must be considered in the following two cases:

If C_t is an unassigned cell, then C_t is added directly into area A_t, and the area center Ca_t, assignment Qa_t, and neighbor cell set Na_t of the area A_t are updated.

If C_t is an assigned cell, then C_t is first deleted from the original area A_o. The area center Ca_o, assignment Qa_o, and the neighbor cell set Na_o of the original area A_o are then updated. Finally, C_t is added into area A_t, and the area center Ca_t, assignment Qa_t, and neighbor cell set Na_t of the area A_t are updated.

The specific implementation of this part is shown in Algorithm 3.

Algorithm 3 Task Assignment for Robots

```
1  while $A_{un}$ <> null then
2    choose robot $R_j$ with minimum $Qa_j$ as $R_t$;
3    divide $Na_t$ into 2 parts of $Na_{t,un}$ and $Na_{t,as}$;
4    $C_t$ = null;
5    if $C_t$ == null then
6      choose $C_t$ in $Na_{t,un}$ with the minimum $cost_{un}$;
7    end if
8    if $C_t$ == null then
9      divide $Na_{t,as}$ into 2 parts of $Na_{t,as,nu}$, $Na_{t,as,fu}$;
10     choose $C_t$ in $Na_{t,as,nu}$ with the minimum $cost_{as}$;
11     if $C_t$ == null then
12       choose $C_t$ in $Na_{t,as,fu}$ with the minimum $cost_{as}$;
13     end if
14   end if
15   if $C_t \epsilon Na_{t,un}$ then
16       add $C_t$ into $A_t$;
17       update $Ca_t$, $Qa_t$, $Na_t$;
18    else if $C_t \epsilon Na_{t,as}$ then
19       delete $C_t$ from $A_o$;
20       update $Ca_o$, $Qa_o$, $Na_o$;
21       add $C_t$ into $A_t$;
22       update $Ca_t$, $Qa_t$, $Na_t$;
23    end if
24  end
```

F. SEARCHING COVERAGE PATHS

After processing the area using the above algorithm, we make a relatively balanced assignment of the area D to all robots and obtain several continuous areas. In this section, we focus on the path planning of each robot in its area.

Considering that our algorithm is based on the idea of approximate cell decomposition, we use the STC algorithm to complete the path planning for the area of each robot. However, different from the application of the original STC algorithm to the situation wherein all cells have the same size, different cell sizes are found in our algorithm. Therefore, we must revise the original STC algorithm slightly to meet our algorithm.

1) STC

The STC algorithm is a method for single-robot CPP based on approximate cell segmentation. This algorithm has been proven to have an algorithm complexity of $O(n)$, where n represents the number of cells. This algorithm can obtain the optimal solution of the shortest path in most cases.

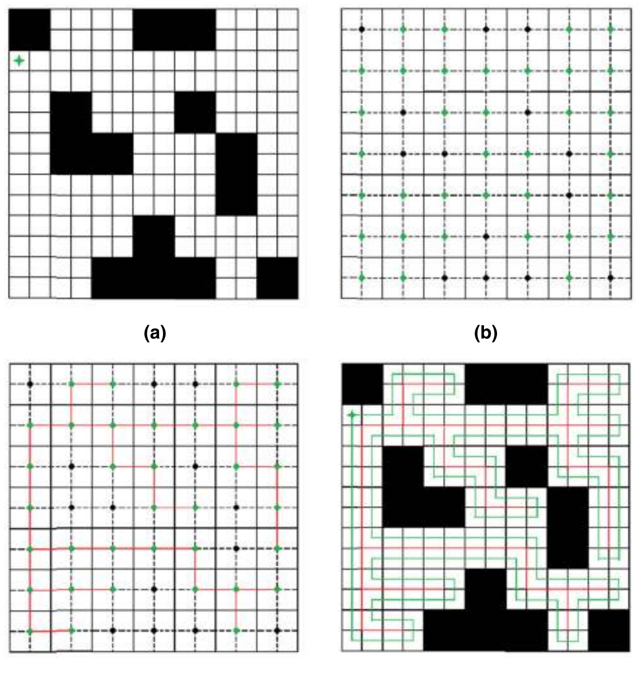


FIGURE 10. The schematic diagram of STC algorithm. (a) The area divided by cells with a certain size and the initial position; (b) The centers of big cells combined with 4 small cells; (c) The MST of these centers; (d) The coverage path got by spiraling the MST from the initial position.

As shown in Figure 10(a), the area is divided into cells with a certain size. White cells indicate accessible areas, black cells indicate inaccessible areas, such as obstacles and no-fly zones, and the green star symbol indicates the initial position of the robot. The STC algorithm is mainly divided into the following steps.

(1) Division of the area by twice the size of the original cell.

Each parent cell contains four small child cells, as shown in Figure 10(b). The center of the parent cell is regarded as the node of the spanning tree. The green dots indicate accessible nodes, and the black dots indicate inaccessible nodes that do not require connection to the spanning tree.

(2) Connection of the Minimum Spanning Tree

The Minimum Spanning Tree (MST) algorithm, such as Prim and Kruskal algorithms, are used to connect accessible nodes into an MST as shown in Figure 10(c).

(3) Search of the coverage path

Finally, starting from the initial position of the robot, the spanning tree is spiraled in certain directions, such as clockwise or counterclockwise, to obtain the coverage path as shown in Figure 10(d).

2) IMPROVED STC

We make some revisions to the original STC algorithm. These revisions are conducted to help the STC algorithm adapt a quadtree structure in our research case and complete the CPP for each robot in our algorithm.

We introduce how to obtain the optimal coverage path of the area after the hierarchical quadtree segmentation and assignment. If we have the area as shown in Figure 11(a), and the green star symbol indicates the initial position of the robot, then the algorithm is divided into the following three steps.

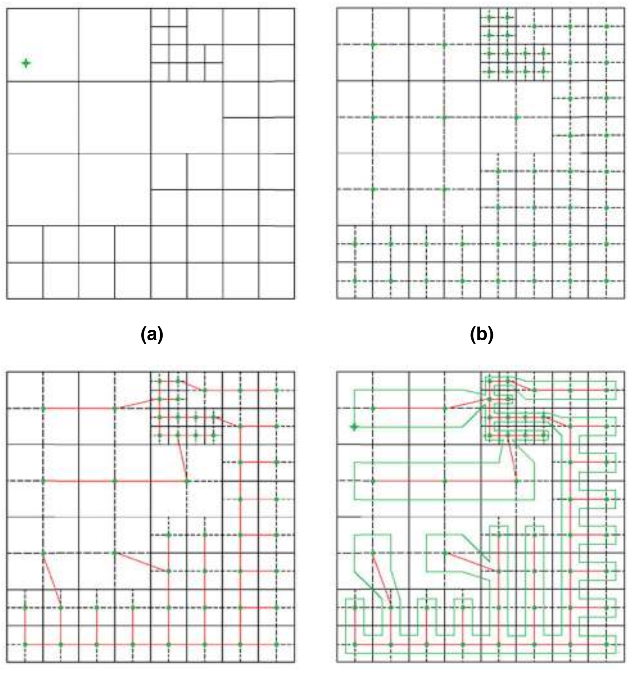


FIGURE 11. The schematic diagram of Improved STC algorithm. (a) The active cells in the area and the initial position; (b) The centers of these cells; (c) The MST of these centers; (d) The coverage path got by spiraling the MST from the initial position.

(1) Determine the spanning-tree nodes

In the original STC algorithm, four child cells can form a parent cell because of the same cell size. However, the four cells cannot be formed into a uniform parent cell in our algorithm due to the different cell sizes. When building a hierarchical quadtree, we divide the task area with twice the visual field of robots, that is, the cell in our algorithm is equipvalent to the parent cell in the original spiral STC algorithm. This division is performed to meet the requirements for the cells in the spiral STC algorithm, Therefore, the center of the cells is the node of the spanning tree, as shown in Figure 11(b).

(2) Construct the spanning tree

The connected cells must be adjacent in the STC algorithm. We can construct a spanning tree quickly because we have built the adjacency topological relationship in the previous steps and only the neighboring cells must be considered when connecting the spanning tree. When generating the spanning tree, we first consider the length of the edge, which connects the corresponding node into the spanning tree. We then select the node with the shortest edge, which is the requirement of the Prim algorithm for generating a spanning tree. When the lengths of the edges are the same, we select the access nodes clockwise to avoid the path turns frequently. A spanning tree is constructed starting from the cell where the initial position of the robot is located, as shown in Figure 11(c).

(3) Generate coverage path

In the original STC algorithm based on the same cell size, the distances from the points on the coverage path generated by spiraling the spanning tree to the spanning tree are equal (Figure 10(d)). However, the distance in our algorithm must be changed due to the different cell sizes. Therefore, path planning is slightly different. The specific process is as follows.

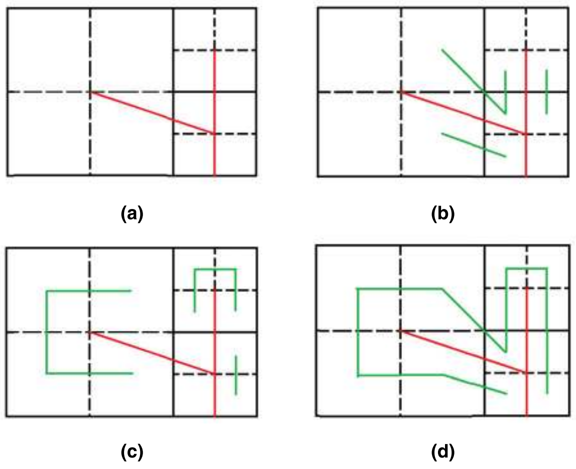


FIGURE 12. The schematic diagram of spiral path generation process. (a) The local spanning tree; (b) The coverage path between cells; (c) The coverage path inside cells; (d) The complete coverage path of the local spanning tree.

A part of the spanning tree is depicted in Figure 12(a). The Figure shows three cells, and the solid red lines represent the edges of the spanning tree.

Generation of paths between cells. We determine the azimuth relationship between the two connected cells and connect the centers of their corresponding child cells to generate paths between the cells, as shown by the green line in Figure 12(b).

Generation of paths inside cells. We determine the azimuth of the edges connected to the current cell center and connect the centers of adjacent child cells on the azimuth where no edge is connected, as shown by the green line in Figure 12(c).

Finally, the paths between and inside cells are connected to form a complete coverage path, as respectively shown in Figure 11(d) and Figure 12(d).

The proposed MCFT-MCPP algorithm completed the implementation from task assignment to path planning for the area D with multiple land cover types and several robots. We will verify our proposed algorithm based on a real classification image in the next section.

IV. EXPERIMENT AND ANALYSIS

A. VALIDATION EFFECTIVENESS

In order to verify the effectiveness of the algorithm, we choose the UAV remote sensing image of real environment, shown as Figure 13. The image range is 1200m $\times$ 1200m, and its ground resolution is about 0.83m.

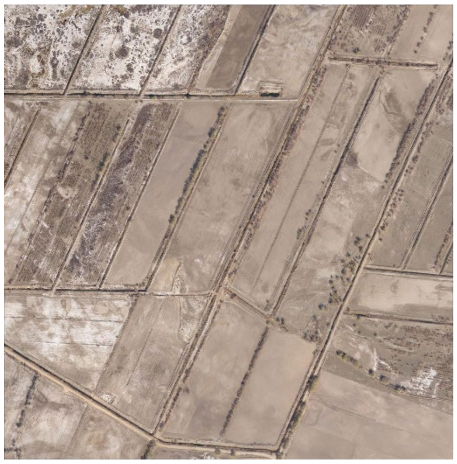


FIGURE 13. UAV remote sensing image data of experiment area.

We use machine learning algorithm to classify this image into six land cover types, including farm, road, grass, wood, bare land, and Gobi.

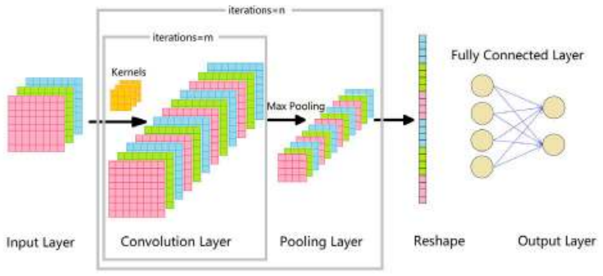


FIGURE 14. The structure of DCNN in machine learning algorithm.

The algorithm uses Deep Convolution Neural Network (DCNN) structure that was trained on sample data taken from UAV image. The structure is shown in Figure 14. We set value of iteration m and n to 2 respectively for the best performance in our case. Value 2 for m and n means that the structure has two big cycles and each of which has two convolution layers and one pooling layer.

Table 2 shows the amount of sample data we used to train, verify and test DCNN structure respectively.

TABLE 2. The details of sample data.

Sample	Sample size (pixel count)					
Using	Grass	Road	Farm	Wood	Gobi	Bare land
Train	1274	4156	2728	2750	2413	2875
Verify	424	1385	909	916	806	958
Test	426	1387	911	918	806	960
Total	2124	6928	4548	4584	4025	4793

In the training process, we set the learning rate as 0.001, batch size as 30, and epoch number as 30. The same experiment repeated 4 times, and the final network loss and accuracy are shown in the Table 3, from which we can see that the mean average loss and accuracy can reach to 0.1407 and 94.56% respectively.

TABLE 3. The loss and accuracy of the trained DCNN.

Time	Loss	Accuracy (%)
1	0.1169	96.21
2	0.1722	92.04
3	0.2272	90.58
4	0.0466	99.39
Mean	0.1407	94.56

Finally, we use the trained DCNN network to complete the classification of the experimental area, and the classification result is shown in Figure 15.

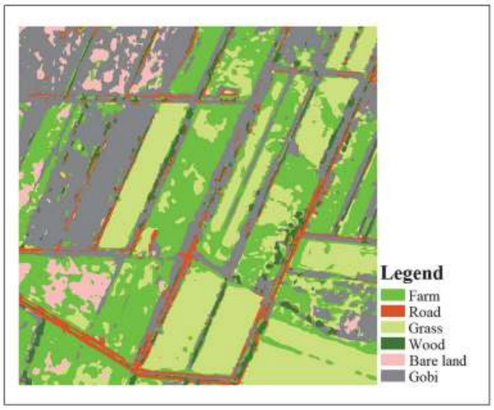


FIGURE 15. Classification image data of experiment area.

We assumed there are 10 robots which have the same performance and the random initial positions. We set the visual fields and moving speeds of these robots in these different cover types to some empirical values given in Table 4 to apply our algorithm to this area. The visual fields and moving speed values in Table 4 are not real values measured outdoors but empirical values for the simulation of the experiment. However, in actual application cases, all parameters should be collected with certain equipment.

TABLE 4. Robot configuration parameters in different land cover types.

ID	Land cover type	visual field (m)	Moving speed (m/s)
1	Farm	10	2
2	Road	30	8
3	Grass	20	3
4	Wood	5	1
5	Bare land	50	5
6	Gobi	40	4

In the experiment, we first use our algorithm to build a hierarchical quadtree the experimental area into 8091 different size cells that cover the whole entire area without repetition according to the search capabilities of robots in different land cover types, as shown in Figure 16. The adjacent topological relationship of the active cells is established to facilitate subsequent task assignments.

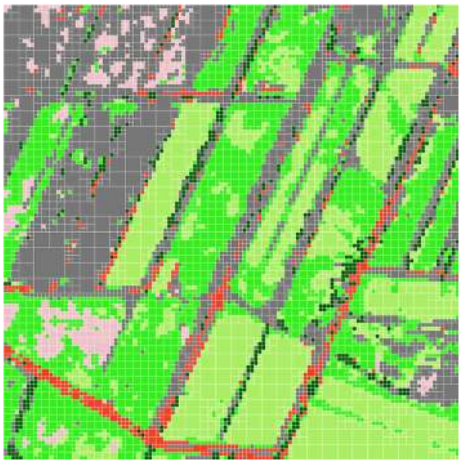


FIGURE 16. The to-be-assigned cells in the hierarchical quadtree.

After building the hierarchical quadtree and the adjacent topological relations in the task area, we create the initial positions for 10 robots in random way. Then we used our algorithm to finish the task assignment.

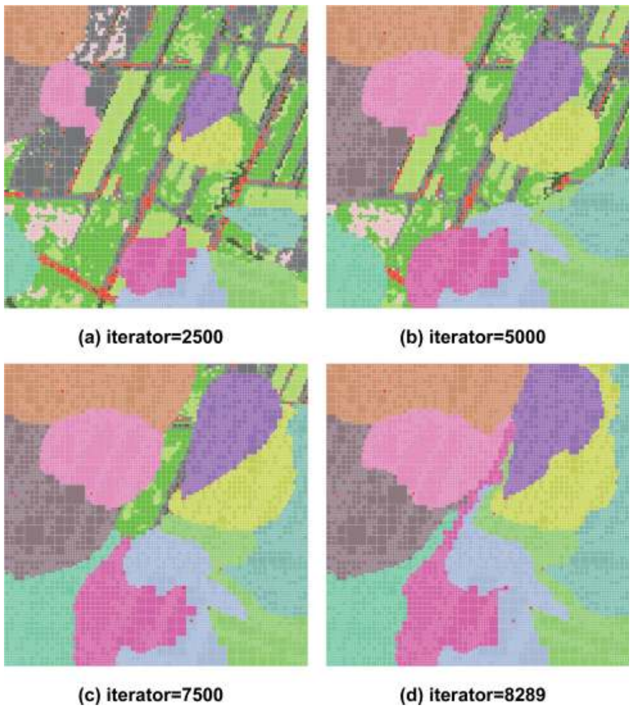


FIGURE 17. The process and results of task assignment.

As shown in Figure 17, the initial positions of each robots are indicated by red dot in initial cells. Figure 17(a), 17(b), 17(c) and 17(d) show the middle process of the task assignment with the iteration increased. A total of 8289 loops were used to complete the assignment of all

8091 active cells. It shows that the Azimuth Trend Method can guide the task assignment process to converge quickly.

We count the task cost of each robot after the task assignment. The result in Table 5 indicates that the task costs of 10 robots are close.

TABLE 5. Coverage path cost statistic for each robot.

Robot ID	Task cost (s)
1	7539.70
2	7532.37
3	7534.20
4	7535.03
5	7537.03
6	7543.20
7	7546.20
8	7534.53
9	7533.70
10	7530.53

Based on the task assignment result shown in Figure 16, we use the advanced spiral STC algorithm to plan the coverage path of each area, as shown in Figure 18. Different colors are used to distinguish different task areas of each robot. The solid red line indicates the spanning tree of each area.

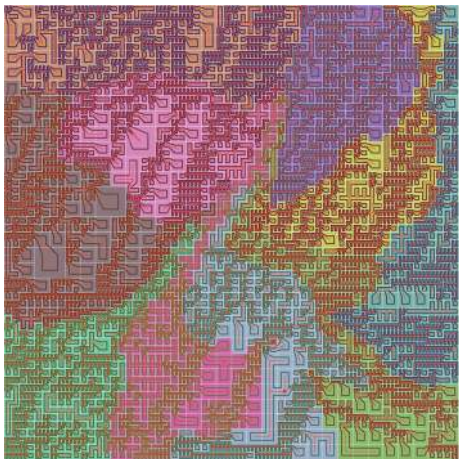


FIGURE 18. The spanning tree and coverage path of each robot in each sub-area.

We provide the visualization results in Figure 19 to show the coverage path of every robot clearly and demonstrate the land cover types in real environment. The Figure 19(a) shows that the coverage paths in different sub-areas are continuous and disjoint, and the denseness of the paths varies with the land cover type. Different from other MCPP algorithms based on approximate cell decomposition, the assignment of each robot is approximately equal. However, the size of each area may not be equal in our algorithm due to the different land cover types. This finding is consistent with the real application scenarios. At last, we overlay the coverage path onto the real environment with different land cover types.

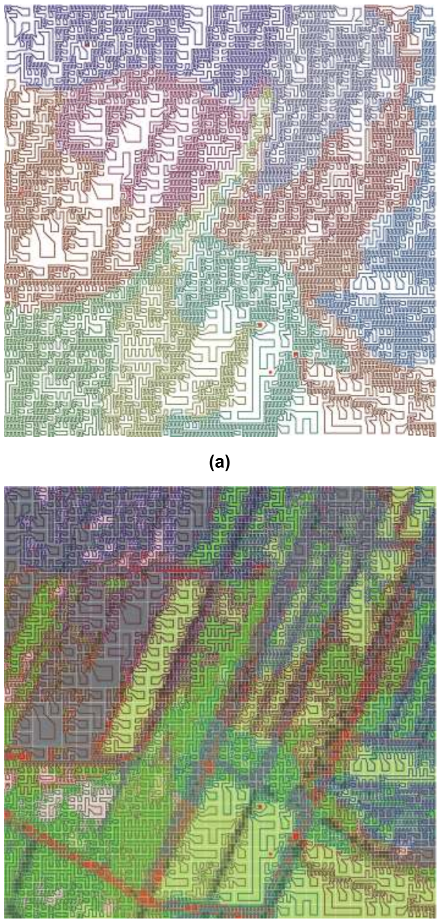


FIGURE 19. The coverage paths of every robots. (a) the coverage paths of each robot; (b) overlaying the coverage path onto the real environment with different land cover types.

Experimental results show that the proposed algorithm is effective for CPP of multiple robots in the environment with multiple cover types. First, different sizes of the active cells can describe the different visual fields of the robots in different land cover types. Second, the area assigned to each robot by the azimuth trend method is continuous and contains the initial position of the robot. The active cells could cover the entire task area without repetition and are assigned to robots with approximately balanced task cost. Third, the coverage path of each area is continuous and disjoint, and its denseness changes with the cover type.

B. ANALYSIS OF COMPLEXITY

The algorithm proposed in this article can be divided into three parts: the hierarchical quadtree establishment, the task assignment, and the path planning. We use the mean of the number of cells in all layers of the pyramid to approximate the number of active cells w and describe the complexity of the algorithm; that is,

$$w \approx \frac{\sum_{i=1}^{l} w_i}{l} \tag{5}$$

where w_i represents the number of cells in the ith layer.

Therefore, the complexities of establishing the hierarchical quadtree is approximately $O(l)$, setting the coverage type is less than $O(wl)$, and building the adjacency topology relations of the cells is approximately $O(wl^2)$. The task assignment part is highly complex to explain. However, the complexity of each iteration is generally less than $O(s+w^2)$. The algorithm complexity of the path-planning part is also approximately $O(wlgw)$. Consequently, the time complexity of the entire algorithm can be approximated as follows:

$$O(wl^2 + maxIter\left(s + w^2\right)) \tag{6}$$

where $maxIter$ is less than nm and represents the maximum iteration in the task assignment process. Moreover, with the increase of the number of cells, the computing time of our algorithm does not increase at an exponential rate. Thus, the practical application of the algorithm is guaranteed.

C. COMPARISON WITH A-STC ALGORITHM

As so far, genetic algorithm [23], DARP algorithm [24] and A-STC [25] algorithm based on auction algorithm can be considered as the representative MCPP algorithms. In the reference [25], the author has fully compared A-STC algorithm with genetic algorithm and DARP algorithm.

The results from reference [25] show that compared with A-STC algorithm, genetic algorithm cannot guarantee 100% coverage rate and spend more time. When the environment has more obstacles, the coverage area is wider, and the number of robots increases, the performance of A-STC algorithm is obviously better than that of genetic algorithm.

Compared with DARP algorithm, A-STC algorithm can obtain a more outstanding solution with shorter time in large-scale cases without obstacle. DARP algorithm has strong global optimization ability, and can obtain better completion time performance of coverage task but cost more running time. However, in some cases, DARP algorithm cannot segment the target area, and the initial position of robots would have a strong impact on the algorithm running time.

For most instances of MCPP algorithms based on equal size grids method, A-STC algorithm can find out the almost optimal solution in a short time.

In general, Genetic algorithm, DARP algorithm, A-STC algorithm and our MCPP-MLCT are all grid-based methods. But except our MCPP-MLCT algorithm that uses various grid sizes to considering multiple land types, the remaining three algorithms are all use one fixed grid size in the whole calculation process. So the efficiency of these algorithms cannot be directly compared. However, we can follow the above compared results from reference [25], so in this article, we only need to compare our MCPP-MLCT algorithm with A-STC algorithm.

In order to compare the practical application efficiency of the two algorithms, we used the following methods:

Given the classification image of an area and the visual field and moving speed parameters of robots corresponding to different land cover types. The initial positions of robots are the same in these 2 algorithms. As A-STC algorithm only use the fixed cell size, in order to avoid missing coverage in searching in complex ground surface, we assigned the minimal visual field value as the cell size to segment the task area and used the corresponding algorithm to complete the MCPP problem. For the MCPP-MLCT algorithm, we just used the method proposed in the previous sections to segment the task area and solve the MCPP problem.

From the experiment results of reference [25] and the results of our previous experiment (as shown in table V), we can get a conclusion that A-STC algorithm and MCPP-MLCT algorithm can both achieve approximate balanced task allocation, and each robot has almost same task amount, which means both algorithms can obtain ideal results.

So, for the further comparison, we need to compare the running time of the two algorithms and the repeated coverage rates of the robots.

Here we introduce the calculation of repeated coverage rate for the detailed comparison. According to the coverage ability (including visual field and moving speed) of the robot, we can calculate the real coverage area S_{real} that is really searched by the robot along the planned path. Assume that the original planned coverage area is S_{area}, then the repeated coverage rate R_{rc} is given as follows:

$$R_{rc} = \frac{S_{real}}{S_{area}} \tag{7}$$

Obviously, MCPP-MLCT can achieve no repeat coverage under ideal conditions if the visual fields of the robot in different land cover types completely satisfy the relationship of $2^k (k = 1, 2, 3, \ldots)$ and there is no over segmentation at the boundary of various types. However, A-STC algorithm cannot realize $R_{rc} = 1$ in any complex cases, as robots cannot keep same searching capabilities, which we expressed by the parameters of moving speed and visual field, in complex environment surface.

In the following sections, we took 3 experiments to compare our algorithm with A-STC algorithm. We used the same data and parameters as above experiment, the land cover types and the related parameters of robots are the same as Table 4.

1) COMPARISON EXPERIMENT I

In order to compare the search efficiency of the two algorithms, we cut four different areas with size of 600m × 600m from the original classification image, and set the robot initial positions randomly for four areas. We use the repeated coverage rate to describe the search efficiency.

As we mentioned above, we had to set cell size in 5 m for A-STC algorithm to avoid missing coverage, which is the minimal visual field of robot, which is corresponding land type 1 in table 6.

The experimental results are shown in Table 6. The MCPP-MLCT algorithm has significantly lowered the repeated coverage rate and running time compared with A-STC, the main reason is that the number of searched cells is dramatically decreased by segmentation of task area with different cell sizes.

TABLE 6. Results of the comparison experiment I.

Area	A-STC		MCPP-MLCT	
	Running time (s)	Repeated coverage rate	Running time (s)	Repeated coverage rate
Area 1	844.4	3.557	1.163	2.170
Area 2	508.2	3.208	1.140	1.949
Area 3	1348	5.323	1.392	2.923
Area 4	913.3	3.407	2.732	2.371

A-STC get quite different running times in this four areas with the same segmented cells, because the initial position of the robot would greatly impact the algorithm running time.

Compared with A-STC algorithm, MCPP-MLCT uses the azimuth trend method we proposed to accelerate convergence of task assignment process, so the algorithm can solve the situation more quickly than A-STC and DARP algorithms, that the majority of robots are closer to each other while only few robots are separate far away.

2) COMPARISON EXPERIMENT II

In order to explore the relationship between the algorithm efficiency and the number of robots, we just used one area (i.e. area 1 in above experiment), and increased the number of robots in different running processes, the results are shown in Table 7 and Figure 20. It shows that the running time of A-STC and MCPP-MLCT both increase, but the running times of MCPP-MLCT are obviously shorter and the changes are more stable.

TABLE 7. Results of the comparison experiment II.

Robot Number	A-STC	MCPP-MLCT
5	594.4	1.782
10	955.2	1.128
15	592.2	2.104
20	591.8	2.732
25	616.0	3.812
30	699.5	3.792

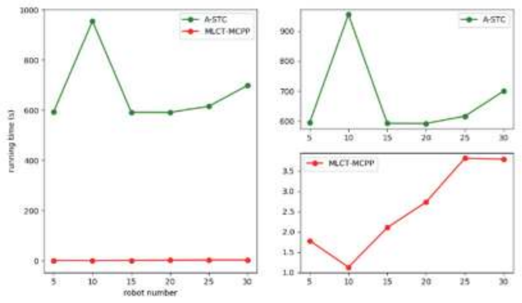


FIGURE 20. Results of the comparison experiment II. Two pictures on the right show the detail running time varieties of 2 algorithms.

3) COMPARISON EXPERIMENT III

In order to explore the relationship between the algorithm efficiency and the size of task area, we selected a series of areas whose sizes given in the Table 8, and respectively ran two algorithms for each area, then we got the results showed in Table 8 and Figure 21.

TABLE 8. Results of the comparison experiment III.

Area Size	A-STC	MCPP-MLCT
200m×200m	6.402	0.343
300m×400m	128.6	0.722
600m×600m	514.6	1.507
800m×800m	3758	3.623
1000m×1000m	10975	4.035
1200m×1200m	*	6.849

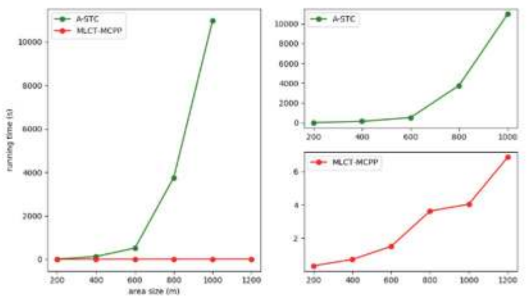


FIGURE 21. Results of the comparison experiment III. Two pictures on the right show the detail running time varieties of 2 algorithms.

The experiment shows that with the increase of task area size, the running time of the two algorithms also increase, but MCPP-MLCT shows much better merit, its running time and the increase rate are significantly lower than that of A-STC.

Above experiments show that MCPP-MLCT algorithm is quite stable with varies of task area sizes, robot numbers and initial positions, and also has a lower repeated coverage rate in the search process. The main reason is that MCPP-MLCT use multiple cell sizes for task area segmentation. Generally, outdoor environment has a complex surface with various land cover types, so MCPP-MLCT has much better merit than A-STC in practical application in outdoor environments, such as providing the basic algorithm for real-time large-area emergency search and rescue.

D. PROBLEMS

1) TASK COST REDUNDANCY INTRODUCED BY APPROXIMATE VISUAL FIELDS

In the proposed algorithm, the real visual fields of robots must be approximated to meet the cell size in the quadtree to meet the requirement that the cell size of adjacent layers in a quadtree structure should be twice. For example, Table 4 shows that robots have a wide real visual field 50 when they execute a search task in bare land. However, the visual field value must be approximated to 40 to satisfy the twice sequence cell size in the quadtree. The result decreased the search efficiency in bare land; that is, the task cost is increased or resulted in redundant task cost.

The task cost is determined by visual field and speed. Thus, if the robot speed can be adjusted in different situations, then cost redundancy can be compensated to some degree.

For example, we can increase robot movement speed in bare land if the visual field is decreased with the cell size.

2) OVER-SEGMENTATION INTRODUCED IN QUADTREE CONSTRUCTION

We use the quadtree structure in the proposed algorithm to represent the task area. Cells with different sizes are used to decompose the places with different land cover types. These places generally have irregular shapes. One land cover type place may be decomposed with several cell sizes to obtain a complete and fully covered decomposition. This approach indicates that we cannot represent one land cover type place with only one cell size (i.e., its corresponding visual field value).

The quadtree is simplified from the lowest layer to the top layer and four child cells with different land cover types are reserved to avoid the duplication and redundancy cover by different layer cells. During this process, over-segmentation is introduced in places near boundaries of different land cover types to maintain cell continuity.

The task cost in a complex place is generally larger than that in a simple place. However, the quadtree construction process must over-segment some complex places. This condition, would introduce extra task costs or decrease task efficiency to some degree.

3) OVER-SEGMENTATION INTRODUCED IN QUADTREE CONSTRUCTION

We use the quadtree structure in the proposed algorithm to represent the task area. Cells with different sizes are used to decompose the places with different land cover types. These places generally have irregular shapes. One land cover type place may be de

4) IMPACT OF ENVIRONMENTAL TOpOLOGY

Similar as DARP algorithm and A-STC algorithm, MCPP-MLCT algorithm is also suitable for topologically continuous environment, if there are some obstacles divided the task area into several separate parts, the algorithm may fail to search the whole area.

5) CAN NOT BE COMPUTED BY PARALLELl PROCESS IN MULTI-CORE SYSTEMS

The algorithm is divided into 3 parts: the establishment of hierarchical quadtree, task assignment and path planning. Only the third part can be computed by parallel processing, as in the task assignment process, each cell in the whole area had to be assigned to the robots one by one, so that the assignment cannot be completed in the parallel way.

V. CONCLUSION AND FURTHER WORK

In the outdoor complex environment, which is affected by ground cover types or terrain structures, robots or unmanned aerial vehicles cannot perform tasks with uniform efficiency. Existing MCPP algorithms cannot handle the problems in such conditions. We propose the method called MCPP-MLCT to meet the requirement for robot task assignment in the outdoor complex surface environment.

The main features of the algorithm include the building of various size cell systems without repetition based on a hierarchical quadtree structure and facilitating effective task assignment and path planning by constructing adjacent topological relationships among the cells. Several cost functions based on visual field and moving speed of the robots in different land cover types are also established. The azimuth trend method is proposed to guide the task assignment to converge efficiently. Finally, we use the improved Spiral STC algorithm to complete the path planning for each robot.

Our algorithm can effectively reduce the repeated coverage of the robots in complex environment. With the increase of search area size and the number of robots participating in the search, the performance of our algorithm has obvious advantages compared with the current mainstream MCPP algorithms. It can be used in applications of large-scale emergency rescue.

Currently, for the sake of simplicity, in the complex environment, we do not consider the existence of geographic features such as mountains and canyons, however, in actual rescue application, we may face to an environment in which many mountains and canyons exist. Therefore, in our further work, we will consider add 3D terrain model and urban buildings to the algorithm to simulate geographic features impact to robot's task assignment. In addition, we will consider using random tree replace quadtree to divide the complex environment, so as to further improve the classification of terrain features and terrain types.

REFERENCES

[1] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robot. Auto. Syst.*, vol. 61, no. 12, pp. 1258–1276, Dec. 2013.

[2] A. Janchiv, D. Batsaikhan, B. Kim, W. G. Lee, and S.-G. Lee, "Time-efficient and complete coverage path planning based on flow networks for multi-robots," *Int. J. Control, Autom. Syst.*, vol. 11, no. 2, pp. 369–376, Apr. 2013.

[3] H. X. Pham, H. M. La, D. Feil-Seifer, and M. Deans, "A distributed control framework for a team of unmanned aerial vehicles for dynamic wildfire tracking," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Vancouver, BC, Canada, Sep. 2017, pp. 1–12.

[4] R. Bailon-Ruiz, S. Lacroix, and A. Bit-Monnot, "Planning to monitor wildfires with a fleet of UAVs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Madrid, Spain, Oct. 2018, pp. 4729–4734.

[5] L. Doitsidis, S. Weiss, A. Renzaglia, M. W. Achtelik, E. Kosmatopoulos, R. Siegwart, and D. Scaramuzza, "Optimal surveillance coverage for teams of micro aerial vehicles in GPS-denied environments using onboard vision," *Auto. Robots*, vol. 33, nos. 1–2, pp. 173–188, Aug. 2012.

[6] A. Xu, C. Viriyasuthee, and I. Rekleitis, "Efficient complete coverage of a known arbitrary environment with applications to aerial operations," *Auto. Robots*, vol. 36, no. 4, pp. 365–381, Apr. 2014.

[7] H. I. A. Perez-imaz, P. A. F. Rezeck, D. G. Macharet, and M. F. M. Campos, "Multi-robot 3D coverage path planning for first responders teams," in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Fort Worth, TX, USA, Aug. 2016, pp. 1374–1379.

[8] G. Avellar, G. Pereira, L. Pimenta, and P. Iscold, "Multi-UAV routing for area coverage and remote sensing with minimum time," *Sensors*, vol. 15, no. 11, pp. 27783–27803, Nov. 2015.

[9] J. Panerati, L. Gianoli, C. Pinciroli, A. Shabah, G. Nicolescu, and G. Beltrame, "From swarms to stars: Task coverage in robot swarms with connectivity constraints," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Brisbane, QLD, Australia, May 2018, pp. 7674–7681.

[10] S. Vemprala and S. Saripalli, "Vision based collaborative path planning for micro aerial vehicles," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Brisbane, QLD, Australia, May 2018, pp. 1–7.

[11] R. Almadhoun, "A survey on multi-robot coverage path planning for model reconstruction and mapping," *Social Netw. Appl. Sci.*, vol. 1, no. 847, pp. 2523–3963, Jul. 2019.

[12] J. C. Latombe, "Exact Cell Decomposition," in *Robot Motion Planning*. Boston, MA, USA: Kluwer, 1991, pp. 200–247.

[13] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Field and Service Robotics*. 1998, pp. 203–209.

[14] H. Choset, E. Acar, A. A. Rizzi, and J. Luntz, "Exact cellular decompositions in terms of critical points of morse functions," in *Proc. Millennium Conf. IEEE Int. Conf. Robot. Automat. Symp. (ICRA)*, San Francisco, CA, USA, Apr. 2000, pp. 2270–2277.

[15] A. Zelinsky, R. A. Jarvis, J. C. Byrne, and S. Yuta, "Planning paths of complete coverage of an unstructured environment by a mobile robot," in *Proc. Int. Conf. Adv. Robot.*, vol. 13, 1993, pp. 533–538.

[16] Y. Gabriely and E. Rimon, "Spiral-STC: An on-line coverage algorithm of grid environments by a mobile robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2002, pp. 954–960.

[17] M. S. Güzel, V. B. Ajabshir, P. Nattharith, E. C. Gezer, and S. Can, "A novel framework for multi-agent systems using a decentralized strategy," *Robotica*, vol. 37, no. 4, pp. 691–707, Dec. 2018.

[18] M. Chamanbaz, D. Mateo, B. M. Zoss, G. Tokić, E. Wilhelm, R. Bouffanais, and D. K. P. Yue, "Swarm-enabling technology for multi-robot systems," *Frontiers Robot. AI*, vol. 4, no. 12, pp. 1–12, Apr. 2017.

[19] J. Valente, J. Del Cerro, A. Barrientos, and D. Sanz, "Aerial coverage optimization in precision agriculture management: A musical harmony inspired approach," *Comput. Electron. Agricult.*, vol. 99, pp. 153–159, Nov. 2013.

[20] I. Rekleitis, A. P. New, E. S. Rankin, and H. Choset, "Efficient boustrophedon multi-robot coverage: An algorithmic approach," *Ann. Math. Artif. Intell.*, vol. 52, nos. 2–4, pp. 109–142, Apr. 2008.

[21] N. Hazon and G. A. Kaminka, "Redundancy, efficiency and robustness in multi-robot coverage," in *Proc. IEEE Int. Conf. Robot. Autom.*, Barcelona, Spain, Apr. 2005, pp. 735–741.

[22] N. Agmon, N. Hazon, and G. A. Kaminka, "Constructing spanning trees for efficient multi-robot coverage," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Orlando, FL, USA, May 2006, pp. 1698–1703.

[23] M. Kapanoglu, "A pattern-based genetic algorithm for multi-robot coverage path planning minimizing completion time," *J. Intell. Manuf.*, vol. 23, no. 4, pp. 1035–1045, Aug. 2012.

[24] A. C. Kapoutsis, S. A. Chatzichristofis, and E. B. Kosmatopoulos, "DARP: Divide areas algorithm for optimal multi-robot coverage path planning," *J. Intell. Robotic Syst.*, vol. 86, nos. 3–4, pp. 663–680, Jan. 2017.

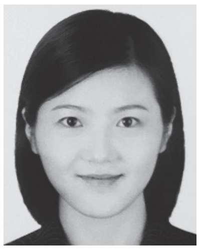
[25] G.-Q. Gao and B. Xin, "A-STC: Auction-based spanning tree coverage algorithm formotion planning of cooperative robots," *Frontiers Inf. Technol. Electron. Eng.*, vol. 20, no. 1, pp. 18–31, Jan. 2019.

XIANG HUANG was born in China, in 1995. He received the B.S. degree in GIS from Peking University, Beijing, China, in 2018, where he is currently pursuing the M.S. degree in GIS.

His research interests include UAV intelligent control and AR GIS. He and his team won the Grand Prize in the Sixth National College GIS Skill Application Competition, Nanjing, China, in 2017.

MIN SUN received the B.S., M.S., and Ph.D. degrees in survey and mapping engineering from Central South University, China, in 1992, 1995, and 2000, respectively.

From 2000 to 2002, he was a Postdoctoral Researcher with the Institute of RS&GIS, Peking University. From July 2002 to 2004, he worked as a Lecturer at the Institute of RS&GIS, Peking University, and a Visiting Scholar at Institute of Computer Science and Applied Mathematics, CAU, Kiel, Germany, from 2004 to 2005. Since July 2006, he has been an Assistant Professor with the Institute of RS&GIS, Peking University. He is the author of one book and more than 50 articles. His research interests include augment reality GIS and UAV remote sensing related image process.



HANG ZHOU was born in Henan, China, in 1997. She received the B.S. degree in GIS from Sun Yat-sen University, Guangzhou, China, in 2019. She is currently pursuing the M.S. degree in GIS with Peking University, Beijing, China.

Her project was elected as the National Training Program of Innovation and Entrepreneurship for Undergraduates, in 2018. Her research interests include UAV intelligent control and AR GIS.

SHUAI LIU received the B.S., M.S., and Ph.D. degrees in geographic information system from Central South University, Changsha, China, in 2003, 2006, and 2011, respectively. From March 2007 to May 2011, he was a Research Associate with the National Geomatics Center of China. He is currently working at the School of Engineering, Honghe University, Mengzi, China. His research interests include computer vision, 3-D modeling, and virtual augmented reality.