A Fast Large-Scale Path Planning Method on Lunar DEM Using Distributed Tile Pyramid Strategy

Zhonghua Hong, *Member, IEEE*, Bin Tu, Xiaohua Tong, *Senior Member, IEEE*, Haiyan Pan, Ruyan Zhou, Yun Zhang, Yanling Han, Jing Wang, Shuhu Yang, and Zhenling Ma

***Abstract*—In lunar exploration missions, path planning for lunar rovers using digital elevation models (DEMs) is currently a hot topic in academic research. However, research on path planning using large-scale DEMs has rarely been discussed, owing to the low time efficiency of existing algorithms. Therefore, in this article, we propose a fast path-planning method using a distributed tile pyramid strategy and an improved A* algorithm. The proposed method consists of three main steps. First, the tile pyramid is generated for the large lunar DEM and stored in Hadoop distributed file system. Second, a distributed path-planning strategy based on tile pyramid (DPPS-TP) is used to accelerate path-planning tasks on large-scale lunar DEMs using Spark and Hadoop. Finally, an improved A* algorithm was proposed to improve the speed of the path-planning task in each tile. The method was tested using lunar DEM images. Experimental results demonstrate that: in a single-machine serial strategy using source DEM generated by the Chang'e-2 CCD stereo camera, the proposed A* algorithm for open list and closed list with random access feature (OC-RA-A* algorithm) is 3.59 times faster than the traditional A* algorithm in long-distance path planning tasks and compared to the distributed parallel computation strategy using source DEM generated by the Chang'e-2 CCD stereo camera, the proposed DPPS-TP based on tile pyramid DEM is 113.66 times faster in the long-range path planning task.**

***Index Terms*—A star, distributed computing, hadoop distributed file system (HDFS), path planning, spark, tile-pyramid.**

I. Introduction

LUNAR exploration using a lunar rover is the first step in human space exploration, and the path planning problem for lunar rovers has been an important focus of research in lunar exploration projects [1].

The core idea of path planning is to determine an optimal path from the current starting position to the goal position in an unknown environment. Path planning algorithms are generally divided into two categories: global path planning and local path planning [2], [3].

Global path-planning research uses global terrain and obstacle information to model and calculate optimal paths, whereas local path-planning research usually models the environmental data in the process of data collection and then measures as many effective paths as possible based on the model. Global path-planning algorithms include classical algorithms such as Dijsktra [4], Floyd [5], A* [6], RRT [7], and intelligent algorithms such as the ant colony algorithm [8] and genetic algorithm [9]. Local path-planning algorithms mainly include artificial potential field-based and neural-network-based methods. Among the above-mentioned methods, Dijsktra and Floyd can calculate the optimal path, but require more time and a large amount of memory as they need to traverse and store all the points [10], [11]. The A* algorithm adopts a heuristic search technique based on the Dijsktra algorithm, which accelerates the path search speed [12]. The RRT algorithm has a fast path-finding speed; however, the path is usually neither optimal nor smooth [13]. The ant colony and genetic algorithms show strong performance in path planning; however, they rely heavily on the parameter setting, and the convergence speed is slow. Gan et al. [14] proposed an improved RRT algorithm for fast tree construction to reduce the time spent of path planning. Liu et al. [15] combined pheromone diffusion and geometric local optimization to propose an improved ant colony algorithm for solving the problem of slow convergence. Bounini et al. [16] proposed a novel potential field method for robot navigation, and Qu et al. [17] proposed a modified pulse-coupled neural network model for real-time path planning of mobile robots in dynamic environments.

With the rapid development of modern mapping technology and the upgrading of sensor hardware, the generated digital elevation models (DEMs) have become increasingly accurate, and the data volume has become larger; therefore, pathfinding calculations based on DEMs are becoming increasingly time-consuming. To solve the above-mentioned problems, Hong et al. proposed an improved A* algorithm using closed list with random access data structure (C-RA-A* algorithm) [18]. Compared with the traditional A* algorithm (Trad-A* algorithm), the efficiency of path planning on DEM generation was significantly improved. However, when the data volume of DEMs is excessively large, the applicability of this algorithm is limited by the memory size of the server. Compared with a single computer, distributed storage [e.g., Hadoop distributed file

Manuscript received 1 September 2022; revised 6 November 2022 and 29 November 2022; accepted 30 November 2022. Date of publication 5 December 2022; date of current version 15 December 2022. This work was supported in part by the National Key R&D Program of China under Grant 2018YFB0505400 and in part by the National Natural Science Foundation of China under Grant 41871325. *(Corresponding author: Xiaohua Tong.)*

Zhonghua Hong, Bin Tu, Haiyan Pan, Ruyan Zhou, Yun Zhang, Yanling Han, Jing Wang, Shuhu Yang, and Zhenling Ma are with the College of Information Technology, Shanghai Ocean University, Shanghai 201306, China (e-mail: zhhong@shou.edu.cn; m200711469@st.shou.edu.cn; hy-pan@shou.edu.cn; ryzhou@shou.edu.cn; y-zhang@shou.edu.cn; ylhan@shou.edu.cn; wangjing@shou.edu.cn; shyang@shou.edu.cn; zlma@shou.edu.cn).

Xiaohua Tong is with the College of Surveying and Geo-Informatics, Tongji University, Shanghai 200092, China (e-mail: xhtong@tongji.edu.cn).

Digital Object Identifier 10.1109/JSTARS.2022.3226527

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

system (HDFS), HBase] and computational technologies (e.g., MapReduce, Spark) use the storage and computational resources of clusters and show tremendous advantages when data increases dramatically. Therefore, they are extensively used in the storage [19], [20], [21], calculation [22], [23], segmentation [24], and path planning of massive remote sensing data. Wang et al. used the MapReduce-based distributed parallel Dijkstra algorithm to solve the shortest path problem. The MapReduce-based distributed Dijkstra algorithm has significant advantages over the traditional Dijkstra algorithm for large-scale path planning [25]. However, for frequent iterative computation, MapReduce needs to spend a considerable amount of time on the disk IO of intermediate data. Alazzam et al. proposed a path-planning algorithm for A* on Spark, and the results showed that the Spark-based A* algorithm has a significant effect on large-scale graph theoretic data [26]. However, their algorithm cannot efficiently use the neighborhood property of DEM grids to obtain neighbour nodes.

According to the above analysis, significant progress has been made, and a number of algorithms have been proposed to improve the efficiency of path planning. However, some problems still need to be explored. First, most existing path-planning algorithms focus on small areas. Relatively few studies have been conducted on path planning based on large-scale data. Second, the efficiency of existing algorithms needs to be improved, particularly when applied on a large scale, and the processing of nodes using big data platforms is mutually irrelevant. Finally, most research focuses on road and traffic data and DEMs of the Earth, and few studies on path planning based on lunar DEMs have been conducted. Therefore, the objective of this article is to propose a fast large-scale path-planning method based on lunar DEMs. The proposed method adopts a distributed tile-pyramid strategy to improve the path-planning efficiency on a large-scale lunar DEM. In addition, an improved A* algorithm was proposed by modifying the data structure, which can enhance the efficiency of sub-path planning on tiles.

The main contributions of this article can be summarized as follows.

1) This article presents a fast large-scale path planning method based on lunar DEM images. The time efficiency of the path search is significantly improved, especially in long distance path planning task.
2) We propose a distributed tile-pyramid strategy for performing large-scale path-planning tasks. This strategy calculates the nodes of path planning from coarse to fine according to the characteristics of the tile pyramid and uses the idea of divide-and-conquer to accelerate the process by distributing the data across the cluster.
3) The data structure of the A* algorithm is improved, thereby accelerating the sub-path-planning task.
4) The method of this article provides support for the fast search of long-distance three-dimensional (3-D) paths over thousands of kilometers. The proposed method is not only applicable to the path planning of lunar rover, but also suitable for path planning task on Mars and Earth.

The remainder of this article is organized as follows. Section II presents the details of the methodology. Section III describes the dataset and experimental design. The experimental results and analyses are presented in Section IV. The influence of factors with different cut tile sizes or computational parallelisms is discussed in Section V. Finally, Section VI concludes the article.

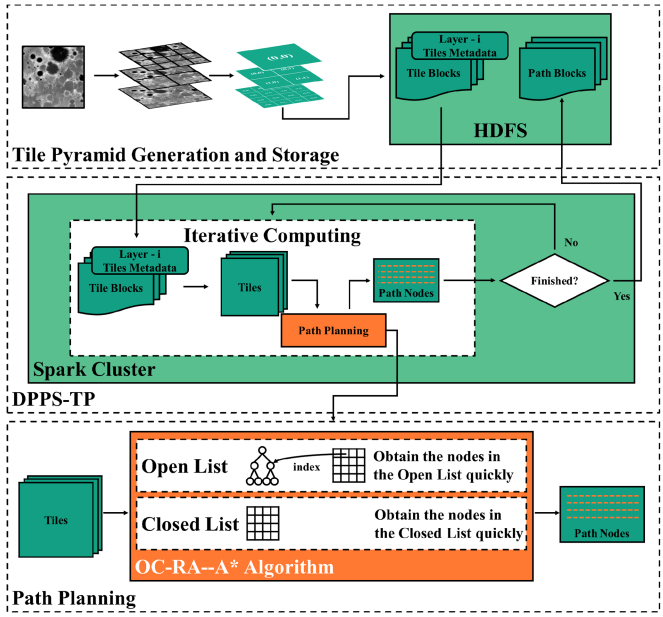


Fig. 1. Overall flowchart of path planning for DEMs using distributed tile pyramid.

II. Methods

This article proposes a fast large-scale path-planning method for lunar DEMs using a distributed tile-pyramid strategy. As shown in Fig. 1, the proposed method comprises three components. First, a tile pyramid was generated for a large lunar DEM and stored in HDFS. Second, a distributed path-planning strategy based on a tile pyramid (DPPS-TP) was used to accelerate path-planning tasks on large-scale lunar DEMs using Spark and Hadoop. Finally, an improved A* algorithm (OC-RA-A* algorithm) was proposed to improve the speed of the path-planning task in each tile.

A. Pyramid Generation and Storage

The tile pyramid was constructed by up-sampling with a 2:1 ratio for the source DEM data to generate the pyramid and cut each layer of the pyramid into rectangular tiles of the same size [27]. As shown in Fig. 2, the source DEM (size 2048×2048px) is the bottom layer (layer 0) of the tile pyramid, and its tile pyramid consists of three layers when the tile size is 512×512px, of which the size of layer 1 is 1024×1024px and that of layer 2 is 512×512px. The tiles of each layer have corresponding tile row and column numbers, which are used to quickly locate the tiles on the tile pyramid.

HDFS was used in this article to store the tile pyramid of the DEM. Because HDFS has high-throughput data access and a data redundancy mechanism, it is well suited for large-scale remote sensing data applications. The HDFS cluster is a typical master-slave operation mode that controls and manages the

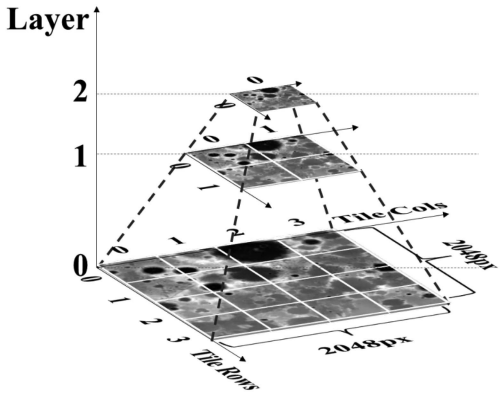


Fig. 2. DEM tile-pyramid structure.

distributed storage of the cluster through the NameNode and DataNode nodes. The NameNode node is used to store the metadata of tile byte blocks. The function of the metadata is to locate the storage location and order of the tile byte blocks in each DataNode node, which is used to store and redundantly generate the data of the tile pyramid. The NameNode node communicates with the DataNode node through a network [28]. Fig. 3 shows how a DEM with a pixel size of 2048×2048 is converted into a tile pyramid with a tile size of 512×512 and stored in the HDFS. First, the tile pyramid is generated from the source DEM with a tile size of 512×512px; subsequently, the tiles of each layer are filled using a Z-order curve [29] or Hilbert curve [29], [30]. Finally, the tile data and pyramid layer information are serialized and stored in HDFS.

B. Principle and Implementation of DPPS-TP

This article proposed DPPS-TP and implemented it using Spark. Using the pyramid model properties, the set of starting and ending points of each tile was calculated from coarse to fine, and the sub-path planning task on each tile was accelerated by Spark using the divide-and-conquer idea.

To reduce the iteration time, a layer-hopping process was included in the tile layer selection from the upper tile layer to the lower tile layer. As shown in Fig. 4, coarse-grained path planning is first performed from the topmost tile layer (n = 4), and the set of starting and ending points of $n = 2$ tile layers is inferred. Subsequently, distributed path planning is performed for the n-2 tile layers until the path planning result of the bottom tile layer is calculated.

This article used Spark to implement DPPS-TP. Apache Spark is a fast, general-purpose computational engine designed for large-scale data processing. It uses resilient distributed datasets (RDDs) and directed acyclic graphs (DAGs) to ensure that the Spark tasks run quickly and correctly in a distributed environment. RDDs are the most fundamental data processing model in Spark and represent a resilient, immutable, partitionable, parallel computable in-memory set. DAGs are a set of combinations of vertices and edges, where vertices are used to represent RDDs and edges are used to represent the operational relationships between RDDs. DAGs in Spark are essential for ensuring that distributed computations can perform tasks sequentially [22], [31]. Research shows that the main reason Spark runs faster than MapReduce is that Spark reduces unnecessary disk IO operations by building DAGs to improve task execution efficiency; however, as MapReduce operations are independent of each other, the results produced by each MapReduce operation are written to the disk [32], [33].

As shown in Fig. 5, in a master-slave mode cluster, Spark divides the path-planning task into multiple subtasks and assigns them to the slave nodes for execution. The slave nodes fetch the tile data from HDFS, execute the subtasks assigned by the master nodes, and finally write the results to HDFS.

In this article, DPPS-TP was implemented using Spark, which is used to construct distributed tile datasets and distribute sub-path planning tasks to clusters for execution. As shown in Fig. 6, the main steps for implementing DPPS-TP using Spark are as follows:

1) Read the topmost pyramid tiles from HDFS and construct tile data RDDs in memory.
2) Input the starting and ending points of the topmost pyramid.
3) Filter the tile RDDs containing the start and end points based on the set of start and end points.
4) Perform the distributed path planning subtask for each tile from the filtered tile RDDs to obtain the tile path RDDs for this layer.
5) Determine whether this layer is the bottom tile pyramid; if it is not the bottom layer, execute step 6; if it is the bottom layer, output the tile path RDDs to the HDFS.
6) Use tile path RDDs of this layer pyramid to deduce the set of starting and ending points in the next tile pyramid layer that needs to perform local path planning.
7) Read the tiles of the next layer pyramid from HDFS and construct tile data RDDs in memory.
8) Input the results of step 6 and step 7 into step 3 and continue down from step 3.

It should be noted that since the starting and ending points of each tile are determined by the high abstraction and low resolution layers, there may be points between tiles that will be fail to pass. These impassable points will be corrected by iterating over the global path points at the end of the experimental program. This step has little impact on the final path planning task. More detail analysis please refer to the results section.

DPPS-TP reduces the time spent performing path planning tasks on large-scale DEMs by moving from coarse-grained tile path-planning tasks to fine-grained distributed tile path planning tasks in an iterative manner.

In the Spark implementation of DPPS-TP, Driver is responsible for broadcasting the set of starting and ending points of each tile and controlling the number of iterations, and workers is responsible for the sub-path planning task of each tile.

As shown in Fig. 7, first, driver and workers construct the tile RDDs and tile metadata of the topmost pyramid ($i = n$) from HDFS, and driver broadcasts the start and end points of the path planning to workers. Subsequently, workers filter the tile RDDs that do not participate in the path planning and perform sub-path planning for the remaining tiles according to the set of

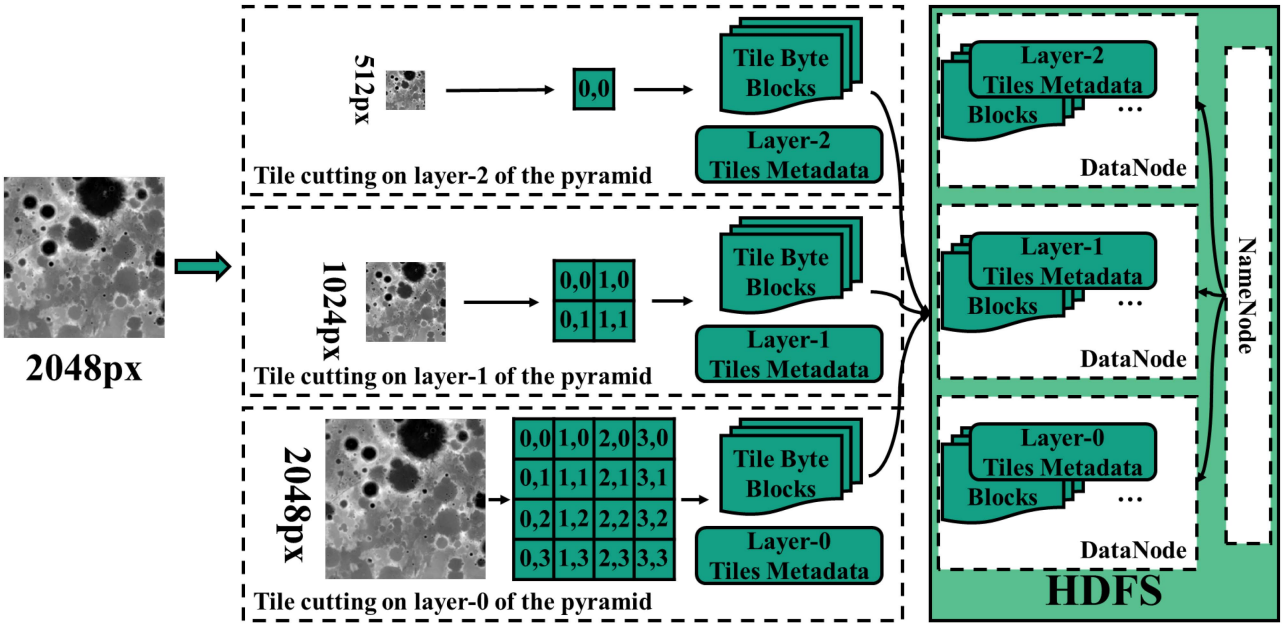


Fig. 3. DEM is distributed to generate a tile pyramid and stored in the HDFS.

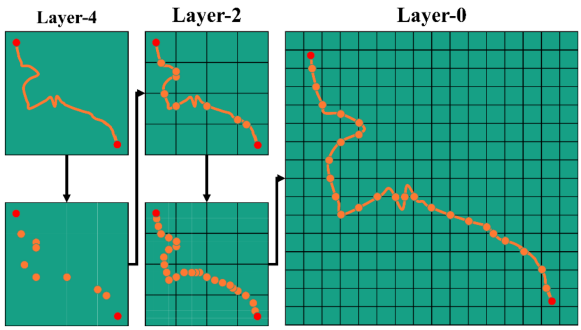


Fig. 4. Principle of distributed path planning strategy based on tile pyramid.

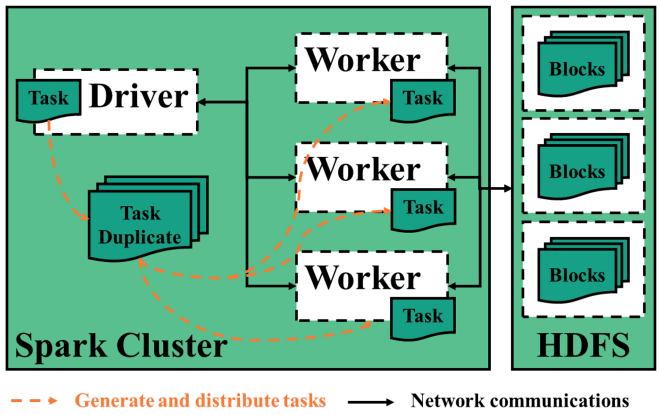


Fig. 5. Spark cluster architecture and task distribution.

start and end points. Finally, DPPS-TP determines whether this layer is the bottom layer ($i = 0$) based on the tile metadata. If yes, the obtained path node RDDs are saved to HDFS, if not, iterative operations are performed. In the iterative operation, driver collects the path node RDDs obtained from workers and calculates the set of starting and ending points of each tile of layer i-2. And then, the Spark cluster sends a read request for the tile pyramid at level i-2 to HDFS and constructs the tile RDD and tile metadata of the tile pyramid at level i-2 from HDFS and finally performs distributed path planning of layer i-2.

C. Improved A Algorithm Using Open List and Closed List With Random Access Data Structure (OC-RA-A* Algorithm)*

The C-RA-A* algorithm uses a minimum heap to implement the open list and a 2-D matrix to implement the closed list. Because the time complexity of the minimum heap to find elements is $O(n)$, the C-RA-A* algorithm spends more time determining and searching the neighbouring nodes of the current node when the amount of data in the open list is large.

Therefore, we made some improvements based on the C-RA-A* algorithm for the closed list and open list data structures. As shown in Fig. 8, in the OC-RA-A* algorithm, the data structure of the open list is implemented using the minimum heap and 2-D matrix, and the data structure of closed list was implemented using a 2-D matrix. Owing to the random-access property of the two-dimensional array structure, the OC-RA-A* algorithm stores the references of the nodes in the open list in memory and improves the search speed of the nodes in the open list. Compared with the C-RA-A* algorithm, the time complexity of the OC-RA-A* algorithm in determining whether the adjacent nodes of the current node are in the open list is reduced from $O(n)$ to $O(1)$.

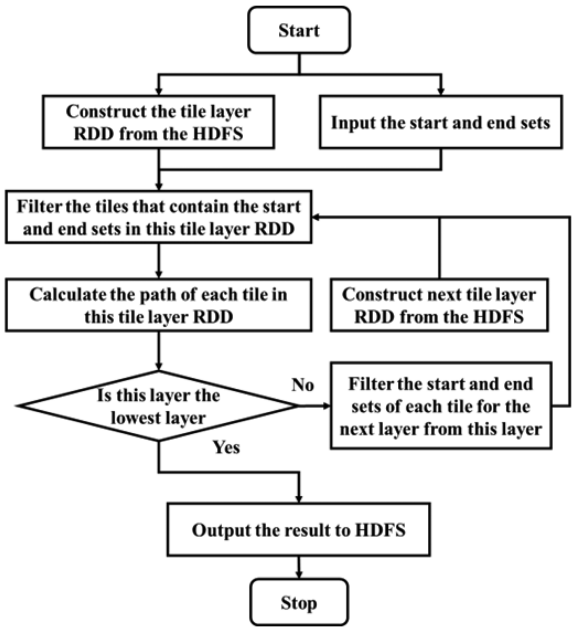


Fig. 6. Flowchart of spark implementing distributed path planning strategy based on tile pyramid.

III. Data and Experiments

A. Data

The 20-m resolution lunar DEM data generated by the Chang'e-2 CCD stereo camera were used in this experiment. The area selected for the experiment is located from 77 N to 87 N, 158 E to 158 W. The total image size of the selected area was 32768×32768 pixels, and the total area was 429496.7296 km^2, as shown in Fig. 9.

B. Experimental Design

Experiments were performed on a single local computer with an AMD Ryzen 7 processor with 2.90 GHz speed and 32G random access memory and on a distributed cluster of three virtual machines with 96G RAM and 48 cores each. In addition, a tile pyramid was generated from the DEM data and stored in HDFS before the experiment.

To ensure the effectiveness of path planning, the tracked lunar vehicle was used as a reference, and a feasible slope threshold of 20° was set for whether to pass that was applied to the experiments of the DEM grid path planning task [34]. As shown in Fig. 10, in the experiments, the projected area of the vehicle is assumed to be a grid cell, and a 3 × 3 window is formed based on the grid cell (i, j) where the vehicle is located, and the eight surrounding grid cells (the corresponding elevation of (i, j) is $Z_{i,j}$). The slopes in the vertical/horizontal and oblique directions were calculated using (1) [35], where CellSize is the size of each grid

$$\text{Slope} = \begin{cases} \tan^{-1}\frac{|Z_{i,\ j}-Z_{x,\ y}|}{\text{CellSize}}, & |x-i|+|y-j|=1 \\ \tan^{-1}\frac{|Z_{i,\ j}-Z_{x,\ y}|}{\text{CellSize}*1.414}, & |x-i|=1 \text{ and } |y-j|=1 \end{cases}. \tag{1}$$

The heuristic function of the A*algorithm used in this experiment is given by

$$F\ (P) = G\,(P) + H\,(P)\,. \tag{2}$$

In (2), $G(P)$ is the actual distance cost from node P to the starting point and $H(P)$ is the estimated distance cost from node P to the end point. In this article, $G(P)$ was obtained by calculating the Euclidean distance, as shown in (3), and $H(P)$ was obtained by calculating the Manhattan distance, as shown in (4). In (3) and (4), *G*(*P*') represents the actual distance from the starting point to P'; $G(P)$ represents the actual distance from the starting point through P' to point P; P^{end} represents the position of the end point; $x,\ y,\ z$ represent the horizontal and vertical positions of the nodes and the corresponding elevation values, respectively,

$$G\ (P) = G\,(P') + \sqrt{(P_x - P'_x)^2 + \left(P_y - P'_y\right)^2 + (P_z - P'_z)^2} \tag{3}$$

$$H\ (P) = \left|P_{\text{x}} - P_{\text{x}}^{\text{end}}\right| + \left|P_{\text{y}} - P_{\text{y}}^{\text{end}}\right|. \tag{4}$$

In this article, four experiments were designed to verify and explore the efficiency of the DPPS-TP and OC-RA-A* algorithms. As given in Table I, these experiments selected the starting and ending points at different distances. Except for the experiments on A* path planning based on long paths with different degrees of parallelism in DPPS-TP, four sets of points were selected for the remaining experiments to simulate the starting and ending points of the long, medium, medium, and short paths.

C. Evaluation Indicators

Time cost and path accuracy were used to measure the performance of the proposed strategy. The time cost is defined as the total running time for path planning, where the time cost for distributed computing is divided into the cluster start-up time and parallel computing time. The total time was calculated using

$$T\ = T_{\text{Start}}\ + T_{\text{Compute}}. \tag{5}$$

In addition, the average deviation between the planned path and the reference path is calculated to evaluate the accuracy of the path planning. The reference path is defined as the shortest passable path from the starting point to the ending point by using the traditional A* algorithm. The planned path represents the path obtained from the experimental calculation.

The average deviation is calculated as the mean offset between the points of the reference path and the planned path, as shown in (6). Where (x_i, y_i), (xg_i, yg_i) denotes the pixel horizontal and vertical coordinates of the planned path and the reference path, respectively, n denotes the number of point pairs, and CellSize denotes the size of each grid. Totally, 90% of the points in the

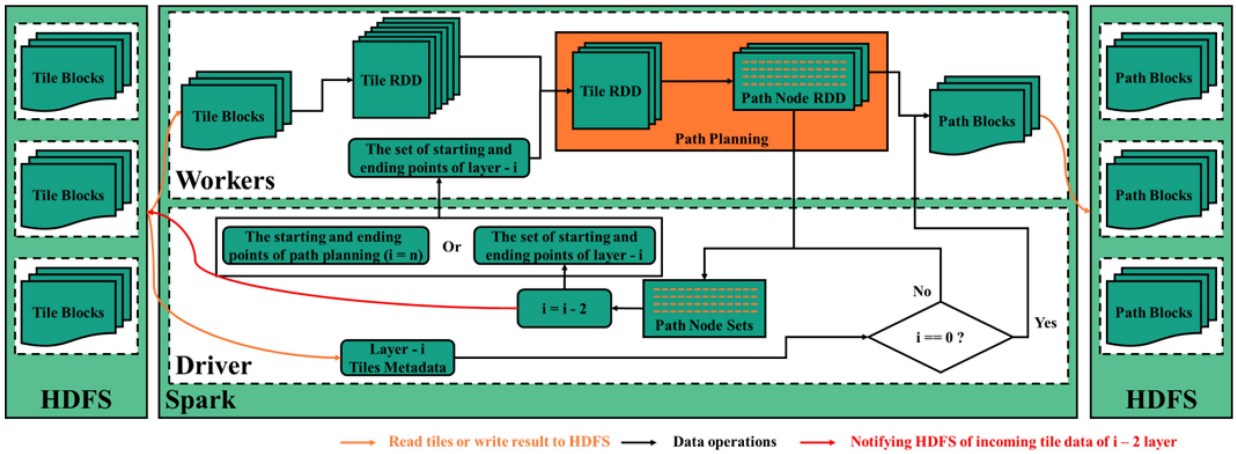


Fig. 7. Spark implementation of distributed path planning strategy based on tile pyramid.

TABLE I
SELECTION OF STARTING AND ENDING POINTS OF DIFFERENT EXPERIMENTS ON DEMS

Experiment	Long Path Start Point-End Point (px, py)-(px, py)	Middle-Long Path Start Point-End Point (px, py)-(px, py)	Median Path Start Point-End Point (px, py)-(px, py)	Short Path Start Point-End Point (px, py)-(px, py)
A* path planning based on single machine and DPPS-TP	(160, 272)-(32416, 32608)	(3072, 3168)-(23968, 23936)	(8176, 8512)-(16848, 17664)	(12224, 12128)-(14016, 14320)
A* path planning based on distributed cluster and DPPS-TP	(16368, 16512)-(17856, 17984)	(16384, 16368)-(16944, 16992)	(16416, 16400)-(16688, 16672)	(17024, 17056)-(17200, 17216)
A* path planning based on DPPS-TP for different tile sizes	(128, 256)-(32384, 32640)	(3072, 3200)-(24000, 23936)	(8192, 8512)-(16832, 17664)	(12224, 12160)-(14016, 14336)
A* path planning based on DPPS-TP with different degree of parallelism in long path	(160, 272)-(32416, 32608)	-	-	-

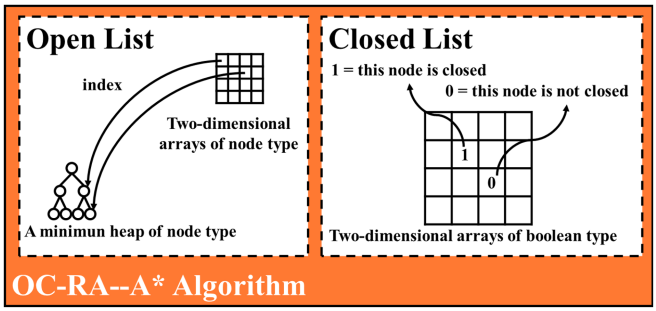


Fig. 8. Data structures of open list and closed list in OC-RA-A* algorithm.

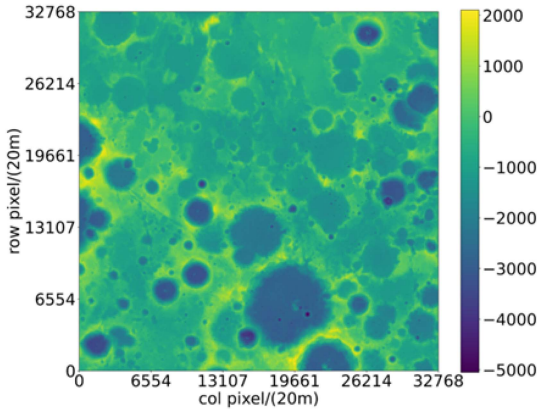


Fig. 9. Lunar DEM data with 20-m resolution.

path are involved

$$\text{Deviation} = \frac{\text{CellSize}}{n} \sum_{i=0}^{n} \sqrt{(x_i - xg_i)^2 + (y_i - yg_i)^2}. \tag{6}$$

IV. RESULTS

A. Results of Time Cost Comparison of Different Methods

The method in this article consists of two parts: the OC-RA-A* algorithm for sub-path planning on each tile and DPPS-TP for parallel processing of the tiles. In order to verify the efficiency of the OC-RA-A* algorithm for path planning at different distances on tiles, this experiment uses a single machine serial computing strategy of the source DEM to mimic the path planning task of different A* algorithms on tiles.

In order to verify the efficiency of the improved A* algorithm in the proposed method, the proposed OC-RA-A* algorithm is compared with the C-RA-A* and Trad-A* algorithms. As given in Table II, the time cost of the Trad-A* algorithm is 3.59 times higher than that of the OC-RA-A* algorithm and the time cost

TABLE II
TIME COST OF USING DIFFERENT A* ALGORITHMS IN SINGLE-MACHINE SERIAL COMPUTING AND DISTRIBUTED PATH-PLANNING STRATEGY BASED ON TILE PYRAMID

Calculative strategy	Path	Using OC-RA-A* Algorithm in Tile			Using C-RA-A* Algorithm in Tile			Using Trad-A* Algorithm in Tile		
		Start-up Time (s)	Running Time (s)	Sum Time (s)	Start-up Time (s)	Running Time (s)	Sum Time (s)	Start-up Time (s)	Running Time (s)	Sum Time (s)
Single-machine serial computing strategy	Long	0	21.685	21.685	0	64.342	64.342	0	77.918	77.918
	Middle-Long	0	18.831	18.831	0	28.969	28.969	0	28.770	28.770
	Median	0	16.177	16.177	0	14.121	14.121	0	15.353	15.353
	Short	0	6.780	6.780	0	3.056	3.056	0	2.566	2.566
DPPS-TP	Long	1.181	12.553	13.734	1.166	12.690	13.856	1.156	16.784	17.940
	Middle-Long	1.163	11.816	12.979	1.167	11.925	13.092	1.178	15.747	16.925
	Median	1.162	11.215	12.377	1.180	11.529	12.709	1.174	14.089	15.263
	Short	1.156	11.041	12.197	1.171	10.956	12.127	1.206	11.609	12.815

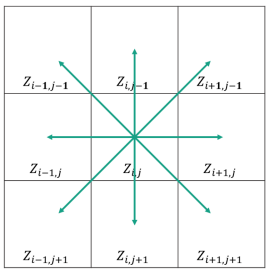


Fig. 10. Central 3 × 3 window grid.

of the C-RA-A* algorithm is 2.96 times higher than that of the OC-RA-A* algorithm when a single machine serial computing strategy is used for the long distance path planning task of the source DEM. It is important to note that the scale of data and the computational environment used in this experiment is different from that used by Hong [18], and that slope maps were not pre-generated for the DEM in this experiment.

To further validate the efficiency of using DPPS-TP, the proposed DPPS-TP is compared with a single machine serial computing strategy at the source DEM. As shown in Fig. 11 and Table II, the time cost using the single machine serial strategy at the source DEM is 1.57 times higher than that of DPPS-TP when using the OC-RA-A* algorithm for long distance path planning tasks. Meanwhile, as shown in Fig. 12 and Table III, the time cost of using a distributed parallel strategy for the long-range path planning task using source DEM is 113 times higher than that of DPPS-TP in a distributed computing environment.

In this experiment, there are several results that need to be explained why. As shown in Fig. 11 and Table II, the proposed method is only applicable to long-distance path planning tasks. For shorter distance path planning tasks, the time taken by DPPS-TP for cluster start-up and RDD transformation would be greater than the computation time for path planning. In addition, in the experiments with DPPS-TP, the cluster start-up times are all controlled between 1 and 1.3 s as the cluster start-up time is only related to the cluster configuration. The speed impact of DPPS-TP with different pathfinding distances is smaller due to

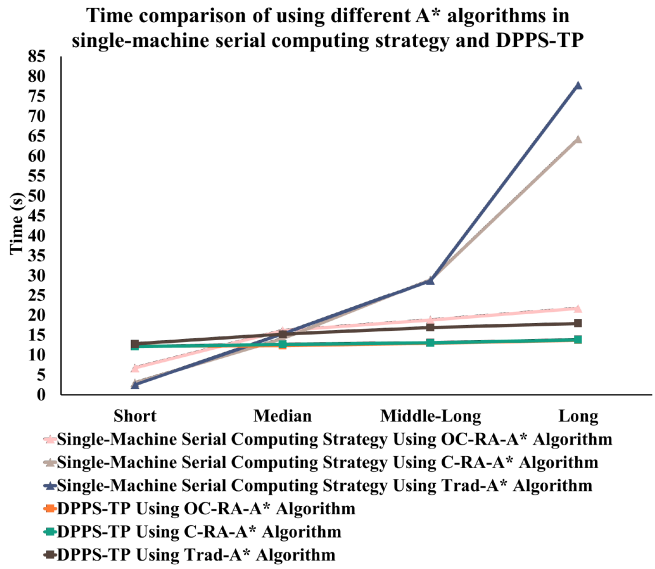


Fig. 11. Time comparison of using different A* algorithms in single-machine serial computing strategy and distributed path-planning strategy based on tile pyramid.

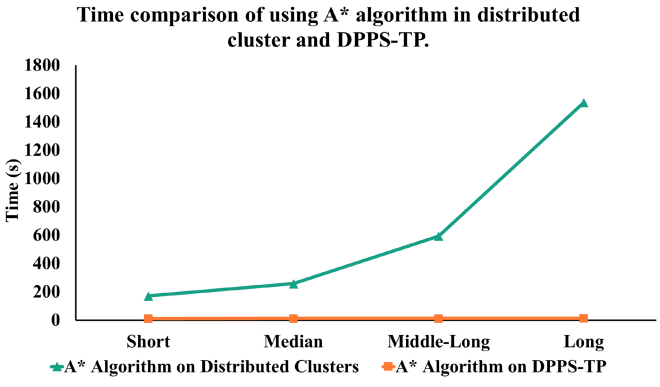


Fig. 12. Time comparison of using A* algorithm in distributed cluster and distributed path-planning strategy based on tile pyramid.

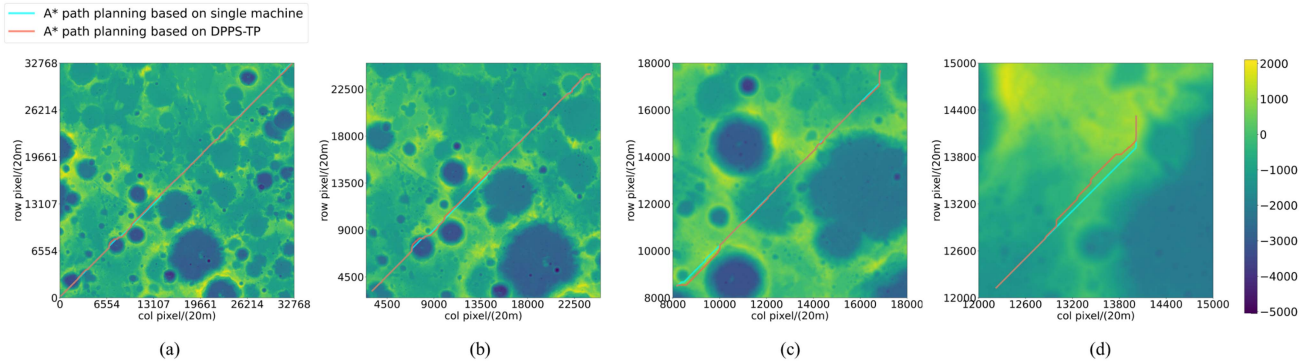


Fig. 13. A* path results based on single machine and distributed path-planning strategy based on tile pyramid. (a) Result of the long path. (b) Result of the middle-long path. (c) Result of the median path. (d) Result of the short path.

TABLE III
TIME COST OF USING DIFFERENT A* ALGORITHMS IN SINGLE-MACHINE SERIAL COMPUTING AND DISTRIBUTED PATH-PLANNING STRATEGY BASED ON TILE PYRAMID

calculative strategy	Path	Start-up Time (s)	Running Time (s)	Sum Time (s)
Distributed parallel computing strategy	Long	1.158	1535.477	1536.635
	Middle-Long	1.148	592.842	593.990
	Median	1.188	256.526	257.714
	Short	1.144	169.893	171.037
DPPS-TP	Long	1.159	12.360	13.519
	Middle-Long	1.141	11.478	12.619
	Median	1.145	11.352	12.497
	Short	1.125	10.794	11.919

the faster sub-path planning on the tiles and the fact that the main time consumption of DPPS-TP is spent on the operation of the RDD operator. As shown in Fig. 12 and Table III, the reason for the long time taken for distributed path planning using clusters only for the DEM is that it transforms the grid data of the DEM into discrete point data resulting in a dramatic increase in the number of nodes involved in the search in the distributed calculation.

B. Results of Accuracy Comparison of Different Methods

To verify the accuracy of the large-scale path planning, the proposed DPPS-TP was compared with a single-computer serial computing strategy and a distributed parallel computing-only strategy.

Figs. 13 and 14 shows the path planning results based on single machine and distributed path-planning strategy, respectively. As can be seen from the figures, the path planning results obtained by different strategies are very similar. The pathfinding distances using DPPS-TP are a little longer than those using the single machine serial strategy. This is because, in the tile pyramid, the seek nodes in the upper layer cannot accurately derive the optimal starting and ending points of each tile in the lower layer.

TABLE IV
ACCURACY OF USING A* ALGORITHM IN DIFFERENT STRATEGIES

Calculative Strategy	Path	Distance (km)	Deviation (km)
Single-machine serial computing strategy	Long	934.607135	-
	Middle-Long	606.374140	-
	Median	260.678733	-
	Short	59.152627	-
DPPS-TP	Long	961.075599	1.147319
	Middle-Long	627.226420	1.164554
	Median	268.628296	0.804097
	Short	60.243263	0.858481

TABLE V
ACCURACY OF A* ALGORITHM USING DIFFERENT STRATEGIES IN DISTRIBUTED ENVIRONMENT

Calculative Strategy	Path	Distance (km)	Deviation (km)
Distributed parallel computing strategy	Long	42.630680	-
	Middle-Long	17.280029	-
	Median	7.793226	-
	Short	4.881606	-
DPPS-TP	Long	44.272451	0.417963
	Middle-Long	17.477859	0.050926
	Median	7.920988	0.013872
	Short	4.892102	0.014968

Tables IV and V is the accuracy comparison of single machine and distributed path-planning strategy. The search paths from the single-machine serial computing strategy and the distributed parallel computing strategy are the same as the reference path because there is no mapping process of different resolution tiles. Therefore, the deviation is zero. The search path using DPPS-TP

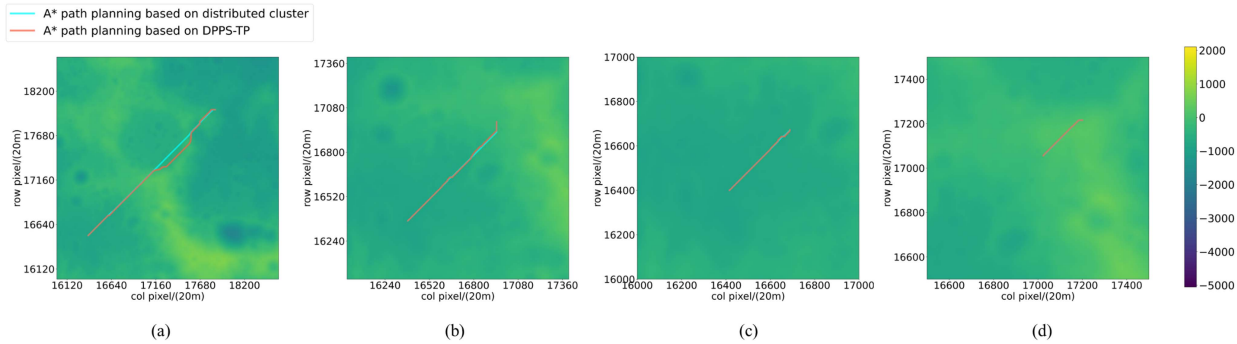


Fig. 14. A* path results based on distributed cluster and distributed path-planning strategy based on tile pyramid. (a) Result of the long path. (b) Result of the middle-long path. (c) Result of the median path. (d) Result of the short path.

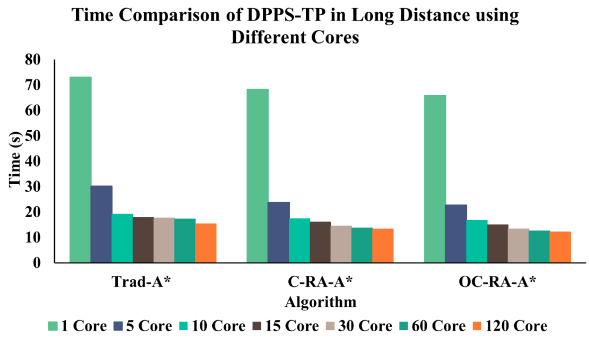


Fig. 15. Time comparison of long path planning with different parallel cores in distributed path-planning strategy based on tile pyramid.

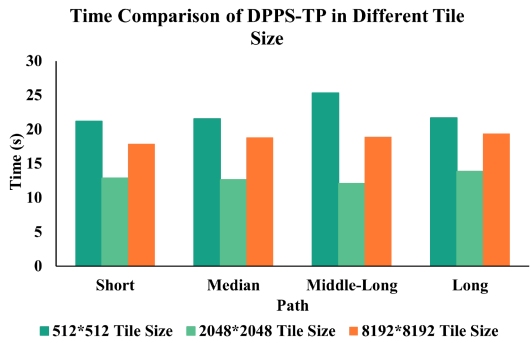


Fig. 16. Time comparison of path planning using different tile sizes in distributed path-planning strategy based on tile pyramid.

has a small deviation relative to the reference path since the starting and ending points of the local path planning on the tiles are not optimal. As can be seen from the tables, the deviation is smaller than 1 km for most of the circumstance. Even for long distance path planning, e.g., the planned distance is greater than 600 km, the deviation is less than 1.2 km. However, the time efficiency is greatly improved. The experimental result indicates the effectiveness of the proposed strategy.

V. Discussion

A. Influence of Using Different Numbers of Parallel Cores

To explore the influence of different numbers of parallel cores on the time cost of path planning using DPPS-TP, seven parallel degrees were selected for the long-distance pathfinding task.

The results shown in Fig. 15 and Table VI indicate that the parallelism of the distributed computation affects the program running time in path planning over long distances, and the lower the number of parallel cores, the higher the time overhead of DPPS-TP.

B. Influence of Constructing Different Tile Pyramids

To explore the influence of different tile pyramids on the efficiency of path planning using DPPS-TP, this experiment was performed using three different tile pyramids for the pathfinding task.

TABLE VI
Time Cost of Long Path Planning With Different Parallel Cores in Distributed Path-Planning Strategy Based on Tile Pyramid

Algorithm	Parallel Degree	Start-up Time (s)	Running Time (s)	Sum Time (s)
Trad-A*	1	1.158	71.918	73.076
	5	1.204	29.005	30.209
	10	1.159	18.071	19.230
	15	1.156	16.784	17.940
	30	1.155	16.550	17.705
	60	1.163	16.200	17.363
	120	1.153	14.284	15.437
C-RA-A*	1	1.170	67.086	68.256
	5	1.174	22.572	23.746
	10	1.165	16.291	17.456
	15	1.156	14.946	16.102
	30	1.211	13.226	14.437
	60	1.166	12.690	13.856
	120	1.154	12.230	13.384
OC-RA-A*	1	1.164	64.750	65.914
	5	1.192	21.554	22.746
	10	1.166	15.575	16.741
	15	1.175	13.861	15.036
	30	1.171	12.289	13.460
	60	1.142	11.494	12.636
	120	1.156	11.041	12.197

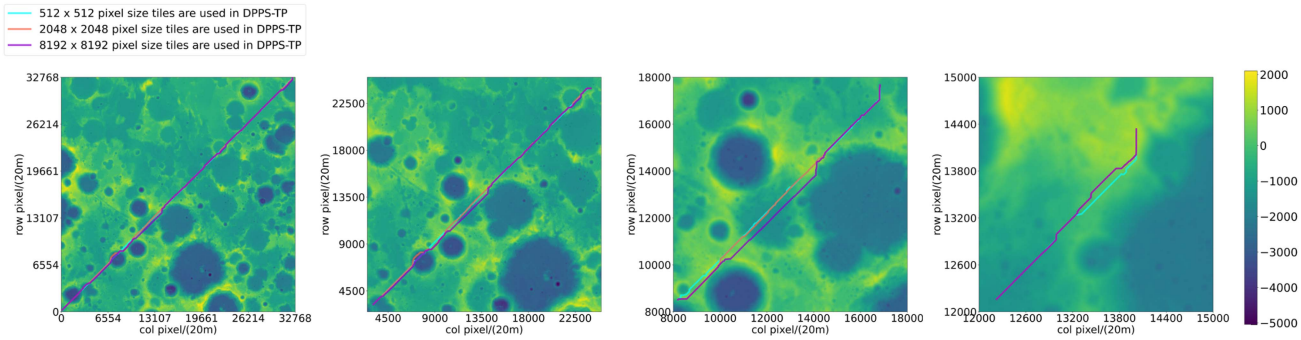


Fig. 17. A* path results based on distributed path-planning strategy based on tile pyramid in 512×512, 2048×2048 and 8192×8192 size tiles. (a) Result of the Long path. (b) Result of the middle-long path. (c) Result of the median path. (d) Result of the short path.

TABLE VII
TIME COST OF PATH PLANNING USING DIFFERENT TILE SIZES IN DISTRIBUTED PATH-PLANNING STRATEGY BASED ON TILE PYRAMID

Tile Pyramid	Path	Start-up Time (s)	Running Time (s)	Sum Time (s)
512×512×7	Long	1.162	20.509	21.671
	Middle-Long	1.127	24.222	25.349
	Median	1.118	20.407	21.525
	Short	1.134	20.040	21.174
2048×2048×5	Long	1.140	12.686	13.826
	Middle-Long	1.141	10.933	12.074
	Median	1.145	11.496	12.641
	Short	1.141	11.743	12.884
8192×8192×3	Long	1.132	18.229	19.361
	Middle-Long	1.133	17.726	18.859
	Median	1.135	17.675	18.810
	Short	1.154	16.705	17.859

TABLE VIII
ACCURACY OF PATH PLANNING USING DIFFERENT TILE SIZES IN DISTRIBUTED PATH-PLANNING STRATEGY BASED ON TILE PYRAMID

Tile Pyramid	Path	Distance (km)	Deviation (km)
512×512×7	Long	975.580250	1.861580
	Middle-Long	637.130711	7.181821
	Median	271.945805	7.737289
	Short	60.684336	0.182732
2048×2048×5	Long	961.959524	2.205383
	Middle-Long	627.944786	7.390781
	Median	267.870082	8.338281
	Short	60.051832	0.550399
8192×8192×3	Long	961.086947	1.841209
	Middle-Long	626.400827	5.388804
	Median	268.921170	11.068751
	Short	60.028377	0.550832

DEM data were used to construct tile pyramids with tile sizes of 512×512, 2048×2048, and 8192×8192. The number of pyramid layers is seven for a 512×512 tile size, five for a tile size of 2048×2048, and three for a tile size of 8192×8192. The results shown in Fig. 16 and Table VII indicate that storing tile pyramids of different sizes also affected the efficiency of DPPS-TP.

The experiments were run from the topmost level of tile pyramids until the bottom level yielded the results. The least time overhead was spent for DPPS-TP using a tile size of 2048×2048, followed by the tile pyramid using a tile size of 8192×8192. The time expense of DPPS-TP with a tile pyramid of tile size 512×512 was larger than that with a tile pyramid of tile size 2048×2048. This is because a tile pyramid of tile size 512×512 requires processing of a larger number of tiles. The time expense of DPPS-TP using a tile pyramid of tile size 8192 × 8192 is greater than that of DPPS-TP using a tile pyramid of tile size 2048 × 2048 because it takes longer to process local sub-paths using tiles of size 8192 × 8192.

As shown in Fig. 17 and Table VIII, in terms of the pathfinding distances, using a tile pyramid with a tile size of 8192×8192 is better than using tile pyramids with tile sizes of 2048×2048 and 512×512. In particular, using a tile pyramid with a tile size of 512 × 512 for DPPS-TP yielded the worst pathfinding distances. This is because with the increasing number of tiles participating in DPPS-TP, the uncertainty of connection points between adjacent tiles will also increase, which may lead to long pathfinding distance.

VI. CONCLUSION

In this article, we propose a fast path planning method using a distributed tile pyramid strategy to solve the problem of the low time efficiency of existing algorithms in large-scale path planning tasks. In addition, an A* algorithm supporting random

access data structures is proposed to reduce the time cost of path planning. In experiments with a single machine serial computing strategy using lunar source DEM data, by virtue of the properties of the random-access data structure, the OC-RA-A* algorithm of this article is 3.59 times faster than the conventional A* algorithm in long-distance path planning tasks. However, the OC-RA-A* algorithm requires high memory capacity and the single-machine serial computing strategy is limited by machine memory, so the distributed path planning strategy based on tile pyramids is further proposed. By divide-and-conquer and coarse to fine grained conversion, the proposed distributed path planning strategy based on tile pyramids is 1.57 times faster than the single machine serial computing strategy using lunar source DEM data. In addition, by retaining the neighborhood information of each pixel, the proposed distributed path planning strategy based on tile pyramids is 113.66 times faster than the distributed parallel computation strategy using the lunar source DEM.

ACKNOWLEDGMENT

We thank China National Space Administration for providing the Chang'e-2 CCD stereo camera DEM-20m data that made this article possible. This data set is processed and produced by "Ground Research and Application System(GRAS)of China's Lunar and Planetary Exploration Program, provided by China National Space Administration (http://moon.bao.ac.cn)."

REFERENCES

[1] J. H. Bai and Y.-J. Oh, "Global path planning of lunar rover under static and dynamic constraints," *Int. J. Aeronaut. Space Sci.*, vol. 21, no. 4, pp. 1105–1113, Apr. 2020.

[2] M. N. Ab Wahab, S. Nefti-Meziani, and A. Atyabi, "A comparative review on mobile robot path planning: Classical or meta-heuristic methods?," *Annu. Rev. Control*, vol. 50, pp. 233–252, Oct. 2020.

[3] H.-Y. Zhang, W.-M. Lin, and A.-X. Chen, "Path planning for the mobile robot: A review," *Symmetry*, vol. 10, no. 10, Oct. 2018, Art. no. 450.

[4] M. Luo, X. Hou, and J. Yang, "Surface optimal path planning using an extended Dijkstra algorithm," *IEEE Access*, vol. 8, pp. 147827–147838, 2020, doi: 10.1109/ACCESS.2020.3015976.

[5] V. Sakharov, S. Chernyi, S. Saburov, and A. Chertkov, "Automatization search for the shortest routes in the transport network using the Floyd-Warshell algorithm," *Transp. Res. Procedia*, vol. 54, pp. 1–11, Feb. 2021.

[6] C. Liu, Q. Mao, X. Chu, and S. Xie, "An improved a-star algorithm considering water current, traffic separation and berthing for vessel path planning," *Appl. Sci.*, vol. 9, no. 6, Mar. 2019, Art. no. 1057.

[7] K. Wei and B. Ren, "A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved RRT algorithm," *Sensors,* vol. 18, no. 2, Feb. 2018, Art. no. 571.

[8] Q. Luo, H. Wang, Y. Zheng, and J. He, "Research on path planning of mobile robot based on improved ant colony algorithm," *Neural Comput. Appl.*, vol. 32, no. 6, pp. 1555–1566, Apr. 2020.

[9] M. Nazarahari, E. Khanmirza, and S. Doostie, "Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm," *Expert Syst. Appl.*, vol. 115, pp. 106–120, Aug. 2019.

[10] T. S. Dermawan, "Comparison of Dijkstra dan Floyd-Warshall algorithm to determine the best route of train," *Int. J. Inform. Develop.*, vol. 7, no. 2, pp. 54–58, Jan. 2019.

[11] L. Marlina, A. Suyitno, and M. Mashuri, "Penerapan algoritma Dijkstra dan Floyd-Warshall untuk Menentukan rute Terpendek Tempat Wisata di Batang," *Unnes J. Math.*, vol. 6, no. 1, pp. 36–47, Oct. 2017.

[12] G. Tang, C. Tang, C. Claramunt, X. Hu, and P. Zhou, "Geometric a-star algorithm: An improved a-star algorithm for AGV path planning in a port environment," *IEEE Access,* vol. 9, pp. 59196–59210, 2021, doi: 10.1109/ACCESS.2021.3070054.

[13] L. da Silva Costa and F. Tonidandel, "DVG+ A* and RRT path-planners: A comparison in a highly dynamic environment," *J. Intell. Robot. Syst.*, vol. 101, no. 3, pp. 1–20, Mar. 2021.

[14] Y. Gan, B. Zhang, C. Ke, X. Zhu, W. He, and T. Ihara, "Research on robot motion planning based on RRT algorithm with nonholonomic constraints," *Neural Process. Lett.*, vol. 53, no. 4, pp. 3011–3029, May 2021.

[15] J. Liu, J. Yang, H. Liu, X. Tian, and M. Gao, "An improved ant colony algorithm for robot path planning," *Soft Comput.*, vol. 21, no. 19, pp. 5829–5839, May 2017.

[16] F. Bounini, D. Gingras, H. Pollart, and D. Gruyer, "Modified artificial potential field method for online path planning applications," in *Proc. IEEE Intell. Veh. Symp.*, 2017, pp. 180–185, doi: 10.1109/IVS.2017.7995717.

[17] H. Qu, S. X. Yang, A. R. Willms, and Z. Yi, "Real-time robot path planning based on a modified pulse-coupled neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 11, pp. 1724–1739, Nov. 2009, doi: 10.1109/TNN.2009.2029858.

[18] Z. Hong et al., "Improved a-star algorithm for long-distance off-road path planning using terrain data map," *ISPRS Int. J. Geo-Inf.*, vol. 10, no. 11, Nov. 2021, Art. no. 785.

[19] L. Li, W. Jing, and N. Wang, "An improved distributed storage model of remote sensing images based on the HDFS and pyramid structure," *Int. J. Comput. Appl. Technol.*, vol. 59, no. 2, pp. 142–151, Mar. 2019.

[20] J. Wu, J. Xiong, H. Dai, Y. Wang, and C. Xu, "MIX-RS: A multi-indexing system based on HDFS for remote sensing data storage," *Tsinghua Sci. Technol.*, vol. 27, no. 6, pp. 881–893, Jun. 2022.

[21] X. Zhou, X. Wang, Y. Zhou, Q. Lin, J. Zhao, and X. Meng, "RSIMS: Large-scale heterogeneous remote sensing images management system," *Remote Sens.*, vol. 13, no. 9, May 2021, Art. no. 1815.

[22] I. Chebbi, N. Mellouli, I. R. Farah, and M. Lamolle, "Big remote sensing image classification based on deep learning extraction features and distributed spark frameworks," *Big Data Cogn. Comput.*, vol. 5, no. 2, p. 21, May 2021, doi: 10.3390/bdcc5020021.

[23] J. Zhang, Z. Ye, and K. Zheng, "A parallel computing approach to spatial neighboring analysis of large amounts of terrain data using spark," *Sensors,* vol. 21, no. 2, Jan. 2021, Art. no. 365.

[24] F. Chen, N. Wang, B. Yu, Y. Qin, and L. Wang, "A strategy of parallel seed-based image segmentation algorithms for handling massive image tiles over the spark platform," *Remote Sens.*, vol. 13, no. 10, May 2021, Art. no. 1969.

[25] W. Yanli and W. Dou, "A parallel algorithm of path planning for DEM terrain data," in *Proc. IEEE 17th Int. Symp. Distrib. Comput. Appl. Bus. Eng. Sci.*, 2018, pp. 22–25.

[26] H. Alazzam, O. AbuAlghanam, and A. Sharieh, "Best path in mountain environment based on parallel A* algorithm and apache spark," *J. Supercomput.*, vol. 78, no. 4, pp. 5075–5094, Sep. 2022.

[27] X. Guan, C. Xie, L. Han, Y. Zeng, D. Shen, and W. Xing, "MAP-Vis: A distributed spatio-temporal big data visualization framework based on a multi-dimensional aggregation pyramid model," *Appl. Sci.*, vol. 10, no. 2, Jan. 2020, Art. no. 598.

[28] M. R. Ghazi and D. Gangodkar, "Hadoop, MapReduce and HDFS: A developers perspective," *Procedia Comput. Sci.*, vol. 48, pp. 45–50, Apr. 2015.

[29] G. Schrack and L. Stocco, "Generation of spatial orders and space-filling curves," *IEEE Trans. Image Process.,* vol. 24, no. 6, pp. 1791–1800, Jun. 2015, doi: 10.1109/TIP.2015.2409571.

[30] Y. Hajjaji, W. Boulila, and I. R. Farah, "An improved tile-based scalable distributed management model of massive high-resolution satellite images," *Procedia Comput. Sci.*, vol. 192, pp. 2931–2942, Sep. 2021.

[31] M. Duan, K. Li, Z. Tang, G. Xiao, and K. Li, "Selection and replacement algorithms for memory performance improvement in spark," *Concurrency Comput., Pract. Experience*, vol. 28, no. 8, pp. 2473–2486, Aug. 2016.

[32] J. Shi et al., "Clash of the titans: MapReduce vs. spark for large scale data analytics," *Proc. VLDB Endowment*, vol. 8, no. 13, pp. 2110–2121, Sep. 2015.

[33] E. Gianniti, A. M. Rizzi, E. Barbierato, M. Gribaudo, and D. Ardagna, "Fluid petri nets for the performance evaluation of MapReduce and spark applications," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 44, no. 4, pp. 23–36, May 2017.

[34] S. Wakabayashi, H. Sato, and S.-I. Nishida, "Design and mobility evaluation of tracked lunar vehicle," *J. Terramechanics*, vol. 46, no. 3, pp. 105–114, Sep. 2009.

[35] X. Shi et al., "An experiment using a circular neighborhood to calculate slope gradient from a DEM," *Photogramm. Eng. Remote Sens.*, vol. 73, no. 2, pp. 143–154, Apr. 2007.

Zhonghua Hong (Member, IEEE) received the Ph.D. degrees in GIS from Tongji University, Shanghai, China, in 2014.

He has been an Associate Professor with the College of Information Technology, Shanghai Ocean University, since 2019. His research interests including satellite/aerial photogrammetry, high-speed videogrammetric, planetary mapping, three-dimensional emergency mapping, GNSS-R, deep learning and processing of geospatial big data.

Bin Tu received the B.S. degree in software engineering from Hubei University of Technology Engineering and Technology College, Hubei, China, in 2019. He is currently working toward the M.S. degree in electronic information with Shanghai Ocean University, Shanghai, China.

His research interests include remote sensing big data, Distributed Storage & Computing.

Xiaohua Tong (Senior Member, IEEE) received the Ph.D. degree in geographic information system from Tongji University, Shanghai, China, in 1999.

From 2001 to 2003, he was a Postdoctoral Researcher with the State Key Laboratory of Information Engineering in Surveying, Mapping, and Remote Sensing, Wuhan University, Wuhan, China. He was a Research Fellow with Hong Kong Polytechnic University, Hong Kong, in 2006. From 2008 to 2009, he was a Visiting Scholar with the University of California, Santa Barbara, CA, USA. His research interests include trust in spatial data, photogrammetry and remote sensing, and image processing for high-resolution satellite images.

Haiyan Pan received the Ph.D. degree in surveying and mapping from Tongji University, Shanghai, China, in 2020.

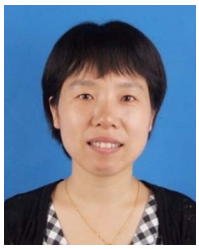
She is currently a Lecturer with the College of Information Technology, Shanghai Ocean University, Shanghai, China. Her research interests include multispectral/hyperspectral image classification, multitemporal remote sensing data analysis, and change detection.

Ruyan Zhou received the Ph.D. degree in agricultural bioenvironment and energy engineering from Henan Agricultural University, Zhengzhou, China, in 2007.

She is currently an Associate Professor with the College of Information Technology, Shanghai Ocean University, Shanghai, China. Her research interests include photogrammetry and deep learning.

Yun Zhang received the Ph.D. degree in applied marine environmental studies from Tokyo University of Maritime Science and Technology, Tokyo, Japan, in 2008.

Since 2011, he has been a Professor with the College of Information and Technology, Shanghai Ocean University, Shanghai, China. His research interests include the study of navigation system reflection signal technique and its maritime application.

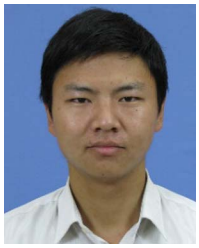


Yanling Han received the B.E. degree in mechanical design and manufacturing and the M.E. degree in mechanical automation from Sichuan University, Sichuan, China, and the Ph.D. degree in engineering and control theory from Shanghai University, Shanghai, China.

She is currently a Professor with the Shanghai Ocean University, Shanghai, China. Her research interests include the study of ocean remote sensing, flexible system modeling, and deep learning.

Jing Wang received the Ph.D. degree in biomedical engineering from the Department of Biomedical Engineering of Shanghai Jiaotong University, Shanghai, China, in 2014.

Since 2015, she has been a Lecturer with the College of Information Technology, Shanghai Ocean University. Her research interests include computer vision and medical image processing.



Shuhu Yang received the Ph.D. degree in physics of physics from School of Physics, Nanjing University, Nanjing, China, in 2012.

Since 2012, he has been a Lecturer with the College of Information Technology, Shanghai Ocean University. His research interests include hyperspectral remote sensing, evolution of the Antarctic ice sheet, and the use of navigational satellite reflections.

Zhenling Ma received the B.E. and M.S.E. degrees in photogrammetry and remote sensing from Central South University, Changsha, China, in 2008 and 2011, respectively, and the Ph.D. degree in geomatics from Wuhan University, Wuhan, China, in 2015.

Since 2018, she has been a Lecturer with the College of Information Technology, Shanghai Ocean University, Shanghai, China. Her research interests include georeferencing for satellite images and underwater photogrammetric engineering.