International Journal of
Geo-Information

MDPI

Article

HexTile: A Hexagonal DGGS-Based Map Tile Algorithm for Visualizing Big Remote Sensing Data in Spark

Xiaochuang Yao [1,2,*], Guojiang Yu [1], Guoqing Li [3], Shuai Yan [1], Long Zhao [3] and Dehai Zhu [1,2,4]

1 College of Land Science and Technology, China Agricultural University, Beijing 100193, China
2 Key Laboratory for Agricultural Land Quality Monitoring and Control, Ministry of Natural Resources, Beijing 100193, China
3 Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing 100094, China
4 Laboratory of Remote Sensing for Agri-Hazards, Ministry of Agriculture and Rural Affairs, Beijing 100193, China
* Correspondence: yxc@cau.edu.cn; Tel.: +86-010-62-737-554

Abstract: The advent of the era of big remote sensing data has transformed traditional data management and analysis models, among which visualization analysis has gradually become an effective method, and map tiles for remote sensing data have always played an important role. However, in high-latitude regions, especially in polar regions, the deformation caused by map projection still exists, which lowers the accuracy of global or large-scale visual analysis, as well as the execution efficiency of big data. To solve the above problems, this paper proposes an algorithm called HexTile, which uses a hexagonal discrete global grid system (DGGS) model to effectively avoid problems caused by map projection and ensure global consistency. At the same time, the algorithm was implemented based on the Spark platform, which also has advantages in efficiency. Based on the DGGS model, hierarchical hexagon map tile construction and a visualization algorithm were designed, including hexagonal slicing, merging, and stitching. The above algorithms were parallelized in Spark to improve the big data execution efficiency. Experiments were carried out with Landsat-8, and the results show that the HexTile algorithm can not only guarantee the quality of global data, but also give full play to the advantages of the cluster in terms of efficiency. Additionally, the visualization was conducted with Cesium and OpenLayers to validate the integration and completeness of hexagon tiles. The scheme proposed in this paper could provide a reference for spatiotemporal big data visualization technology.

Keywords: HexTile; DGGS; big remote sensing data; visualization; Spark

check for updates

Citation: Yao, X.; Yu, G.; Li, G.; Yan, S.; Zhao, L.; Zhu, D. HexTile: A Hexagonal DGGS-Based Map Tile Algorithm for Visualizing Big Remote Sensing Data in Spark. *ISPRS Int. J. Geo-Inf.* **2023**, *12*, 89. https://doi.org/10.3390/ijgi12030089

Academic Editor: Wolfgang Kainz

Received: 20 November 2022
Revised: 20 February 2023
Accepted: 21 February 2023
Published: 23 February 2023

1. Introduction

Big remote sensing data presents multi-dimensional expression on space and time scales, and contains complex and abstract information. As an effective method for spatiotemporal data analysis, visualization can intuitively express the meaning of big data. The map tile is a widely used visualization solution for remote sensing images, which are subdivided into pieces of tiles organized as a pyramid model for display [1]. However, most of the existing map tiles are constructed based on a plane, which cannot avoid a series of problems caused by map projections, such as area or length deformation, and missing high latitude data. At the same time, for different regions, the diversity of map projection types and the complexity of algorithms have further exacerbated the difficulty of multi-source data fusion in the era of big data, which also makes it impossible for global-scale or large-scale scientific research to be carried out within a unified space-time framework. The concept of discrete global grid systems (DGGS) provides a reference for solving the above problems [2]. A DGGS is a special case of a spatial reference system that uses tessellations rather than lattice points to encode location. After decades of development, discrete global grid systems have been applied to the organization, management, analysis, visualization, and other aspects of spatiotemporal data [3] as a digital Earth [4]. A specification for describing DGGSs was incorporated into OGC in 2014 and into ISO 19170–1:2021 in 2001. These developments will further promote the industrialization of DGGSs [5]. A discrete global grid system is a group of cells built on a spherical surface based on the geographic coordinate system, with great potential as a reference framework for digital representations of Earth [2]. A number of different types of DGGSs have been proposed, including tessellations of triangles, diamonds [6], and hexagons [7], which have their own advantages and disadvantages for different applications. A pole-oriented DGGS has been proposed with the polar semi-hexagon grids and rectangular grids [8]. Hexagon grids are different from the others, having more consistent adjacency, superior angle resolution, and higher coverage. Furthermore, hexagon cells share an edge with their six neighbors, and a hexagon grid maintains the same distance between the central hexagon cell and the neighbor cell [9], which makes hexagonal DGGSs popular as the basis for global dynamic simulations, such as atmospheric convection, Earth magnetic field changes, etc. These studies mainly focus on the internal structure and material/energy movement of space objects, and generally adopt the field object model for modeling, that is, the location, shape and distribution of space objects are approximated through the division unit of the sphere space grid, and the attribute information of space objects is organized according to the grid unit, so as to establish the simulation model of the movement and change process of space objects. The consistency and adjacency of hexagons have good spatial structure properties, which are suitable for spatial dynamic modeling. Notably, a grid aperture is defined to depict the ratio of the areas at two adjacent levels (e.g., level *k* and level *k*-1)—for instance, hexagon grids have three types of models according to the aperture (aperture 3, 4, and 7).

Table 1 summarizes the existing open-source DGGS coding schemes, and many of them have available DGGS model source code online [10]. The Open Geospatial Consortium (OGC) also established a group for DGGS standardization in 2014 to discuss the implementation of DGGS to extend their usability and interoperability. Two kinds of DGGS, H3 and rHEALPix, are widely discussed in the DGGS specification manuscript of OGC, and are integrated in GeoServer. H3 is an aperture 7 hexagonal DGGS built on an icosahedron. Hierarchical Equal Area isoLatitude Pixelation (HEALPix) [11] produces a subdivision of a spherical surface where each pixel confers the same surface area as each other pixel. This model was developed to process and analyze data of cosmic microwave background experiments [12]. In the experiment in this study, we used H3 as the basic DGGS for tile slicing. On the one hand, H3 is widely used and discussed in OGC reports, and it has rich language bindings. On the other hand, the hexagonal shape has the advantages of adjacency and topological consistency.

Table 1. The existing open-source schema of DGGS.

DGGS	Base Polyhedron	Shape of Polygon	URL
H3	Icosahedron	Hexagon (aperture 7)	https://github.com/uber/h3 (accessed on 20 September 2021)
OpenEAGGR	Icosahedron	Triangle (aperture 4) Hexagon (aperture 3)	https://github.com/riskaware-ltd/open-eaggr (accessed on 20 September 2021)
DGGRID	Icosahedron	Triangle/diamond/Hexagon (aperture 3/4/mixed)	https://github.com/sahrk/DGGRID (accessed on 21 September 2021)
HEALPix	Rhombic-dodecahedron milliarcseconds	Curvilinear quadrilaterals	https://healpix.sourceforge.io/index.php (accessed on 21 September 2021)
rHEALPix	Cube	Square grid	http://atlas.gge.unb.ca/rHEALPix (accessed on 21 September 2021)
Geogrid	Icosahedron	Hexagon (aperture 3)	https://github.com/giscience/geogrid (accessed on 20 September 2021)

In the field of remote sensing, relevant research has been carried out on hexagonal DGGS-based data modeling [13], hexagon accuracy evaluation [14], data fusion [15], and applications [16]. HexASCII [17], which is similar to the ESRI ACSII raster format, was proposed as an intermediate file to store DGGS data, and it defines specific rules to record the position of DGGS cells. For examples, the cell values should be delimited by spaces. No carriage returns are necessary at the end of each row in the raster, and so on. However, data stored in HexASCII files still need to be converted to a vector data format such as KML to be visualized in GIS software. Robertson [15] developed applications for data analysis and visualization based on DGGS using grid cells to express the shape of vector data (e.g., points, polylines, and polygons). As for raster data, resampling methods have been adopted to extract pixel values from the DGGS cells at the closest resolution. Ma [14] compared the accuracy of three resampling methods using a planar aperture 4 hexagon grid system and pointed out that the bilinear resampling method had the highest accuracy. The above works are an exploration and attempt to use the advantages of DGGS to process and solve remote sensing data. For the visualization of big remote sensing data, the traditional map service engine has played a huge role. However, the existing web map services are not supported for hexagonal DGGS tiles. This paper proposes a hexagonal DGGS-based remote sensing big data visualization algorithm, HexTile, which focuses on the remote sensing data with the "scene" as the unit. The slicing of multi-scale hexagon tiles on the server side and the stitching on the client side are designed to meet the global visualization needs of remote sensing data.

In order to further improve the execution efficiency of HexTile, its parallel transformation was carried out based on the Spark cloud computing platform. Many studies have proposed distributed and cloud environment-based frameworks for massive geospatial data processing and visualization. Distributed computing frameworks, e.g., SpatialHadoop [18], HadoopGIS [19], and GeoSpark [20], are designed for geospatial data management to leverage the high performance advantage of Hadoop MapReduce and Spark, providing spatial operation and basic spatial models for processing. HadoopViz [21] and GeoSparkViz [22] were developed based on SpatialHadoop and GeoSpark, respectively, for geospatial data visualization. HadoopViz is compatible with SpatialHadoop, and the two are responsible for visualization and data processing, respectively [18]. GeoSparkViz is a scalable spatial data visualization framework with massive data rasterization, pixel aggregation, color rendering and other parallel processing capabilities, and its efficiency can be increased by up to five times [22]. GeoTrellis is also an open-source high-performance geographic data framework developed from Spark, which is capable of processing and visualizing massive amounts of geospatial data. These frameworks are designed for big geospatial data processing, with which the efficiency of tile generation has been greatly improved. Large-scale tile data storage and retrieval also represent a significant challenge, and distributed databases such as HDFS, HBase, and Ceph have been used to solve this problem [23,24]. In total, distributed computation and parallel algorithms have been proven to provide great improvement for massive geospatial data processing. The trinity of big remote sensing data, cloud computing, and discrete global grid systems can provide a more suitable solution [25].

The proposed HexTile algorithm considers the unified data framework of DGGS. In this study, the characteristics of Spark cloud computing were combined to design and implement the map tile visualization algorithm to maximize the mining and utilization of remote sensing data values. Firstly, we designed the slicing algorithm based on a hexagon grid to generate hexagon map tiles. Secondly, we adopted the strategy of stitching child tiles into parent tiles to avoid repeatedly reading remote sensing images. The HexTile algorithm was implemented in Spark to achieve high performance. Finally, several experiments were carried out to measure the performance of our algorithms, and the generated hexagon tiles are displayed on a web map with Cesium and OpenLayers to validate tile completeness.

The remainder of this paper is organized as follows: Section 2 provides an overview of our algorithms. The implemented environment and experimental results are outlined in Section 3. Sections 4 and 5 are the discussion and conclusion.

2. Materials and Methods

As shown in Figure 1, the HexTile algorithm proposed in this paper mainly includes three steps: (1) A tile slicing algorithm of remote sensing images based on hexagonal DGGS. This step is mainly to slice the remote sensing image data into hexagonal grids. (2) Hexagon tile merging of boundary fragments. For remote sensing image data with “scene” as the unit, irregular fragments are unified merging to form a hexagon tile. (3) Stitching of tiles of different levels. This is mainly based on basic-level tiles to generate parent hexagonal tiles of higher level, which is a many-to-one process.

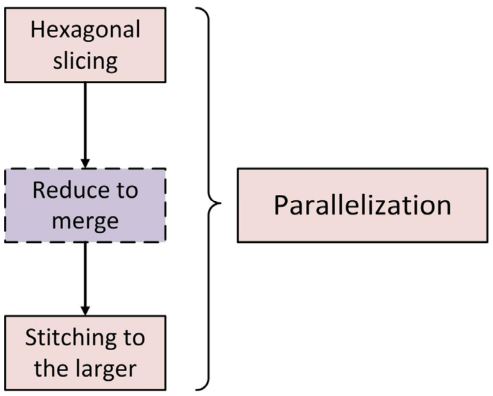

Figure 1. Main three steps of the algorithm.

2.1. Hexagonal Slicing for Remote Sensing Data

As shown in Figure 2, the basic flow of hexagon slicing algorithm for remote sensing image data is described. Taking the K level in DGGS as an example, for a remote sensing image, the hexagonal grid covering the space range in the K level is firstly calculated according to the calculated image range. Then, the envelope rectangle range of the hexagonal grid is calculated, according to the envelope rectangle range cut, and the corresponding grid part of each hexagonal grid in the image is extracted.

The position relationship between the hexagonal grid and a remote sensing image can be divided into two kinds. In the first, the grid is located at the boundary of the image and intersects with the image part. In this case, the solution is to expand the clipped grid space according to the space range of the hexagonal grid and the ratio of width and height. As shown in Formula (1), we can inversely calculate the offset of the intersecting image relative to the hexagonal grid by taking the upper left coordinate as the starting point according to the spatial range and image resolution of the hexagonal grid and the intersecting image. According to the offset of the two, the image of the intersecting part is drawn to the corresponding position in the hexagonal grid canvas, so as to maintain its relative position in the hexagonal grid.

For the second position relationship, the hexagon grid is completely located inside the image, and there is no stretching problem caused by partial intersection of tiles during visualization, so a complete hexagon tile can be directly generated.

Algorithm 1 shows the pseudocode of hexagonal slicing for remote sensing data.

$$\begin{cases} X_{offset} = \dfrac{abs\left(X_{min}^{cropped} - X_{min}^{hexagon}\right)}{resolution} \\ Y_{offset} = \dfrac{abs\left(Y_{max}^{cropped} - Y_{max}^{hexagon}\right)}{resolution} \end{cases} \tag{1}$$

where $\left\{X_{offset}, Y_{offset}\right\}$ is the offset of the cropped raster in the hexagon cell, $\left\{X_{min}^{cropped}, Y_{max}^{cropped}\right\}$ is the minimum longitude and maximum latitude of the cropped raster, $\left\{X_{min}^{hexagon}, Y_{max}^{hexagon}\right\}$ is the minimum longitude and maximum latitude of the hexagon cell, and *resolution* is the resolution of the remote sensing imagery.

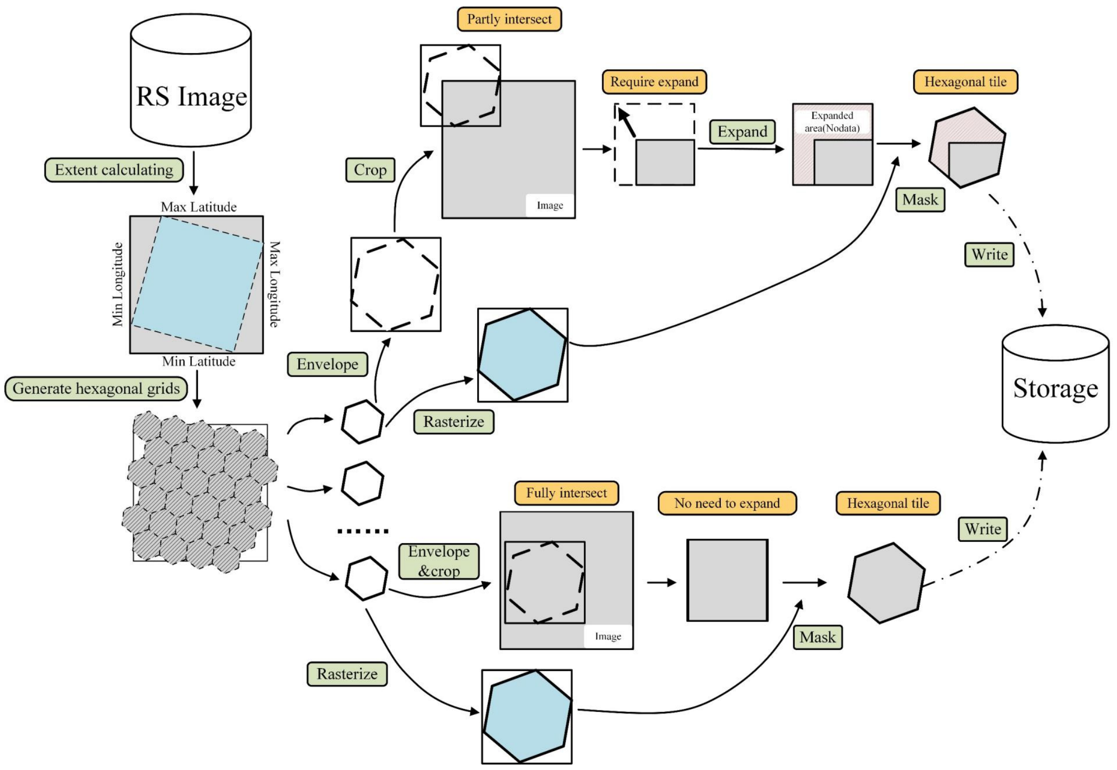

Figure 2. Illustration of slicing method based on a hexagon grid.

Algorithm 1: The pseudocode of remote sensing data hexagonal slicing

```
Input: Remote Sensing Image
Output: hexagon Tiles
1. Calculate the extent of image.
2. Generate the list of hexagons covering the image according to the extent.
3. Loop for each hexagon.
for each hexagon in hexagon list do
    3.1 Calculate the extent of hexagon cells and crop the image with the extent.
    3.2 Calculate the offset (Offsetx, Offsety) between the extent of cropped raster and hexagon raster.
end
4. Merge the cropped raster images that from the same hexagon cells.
5. Mask the raster with hexagon cell and render into hexagon tile.
```

2.2. Hexagonal Merging for Boundary Fragments

Since the remote sensing image takes "scene" as the unit, which is inconsistent with the tile range of hexagonal DGGS, a large number of trivial fragments are generated in the generation process of the hexagon tile, which are post-processed, specifically via the combination of hexagon tiles, to ensure the integrity of map tiles.

As shown in Figure 3, the merger of hexagonal tiles is relatively simple, mainly involving the merging of fragments located on the same tile into a complete hexagonal tile. This process merges tiles as a unit, making it ideal for parallelization.

2.3. Hexagonal Stitching Based on Hierarchy

Figure 4 shows the hexagon tile stitching algorithm. This algorithm is mainly based on the hierarchy of the DGGS to generate hexagonal tiles of a higher level, so as to avoid building multi-level map tiles from the original image and save the overall time of tile construction. In general, we took the map tiles constructed for the first time as the base layer, that is, the lowest layer, and then produced the tiles of the next level according to the parent-child relationship of the DGGS hierarchy.

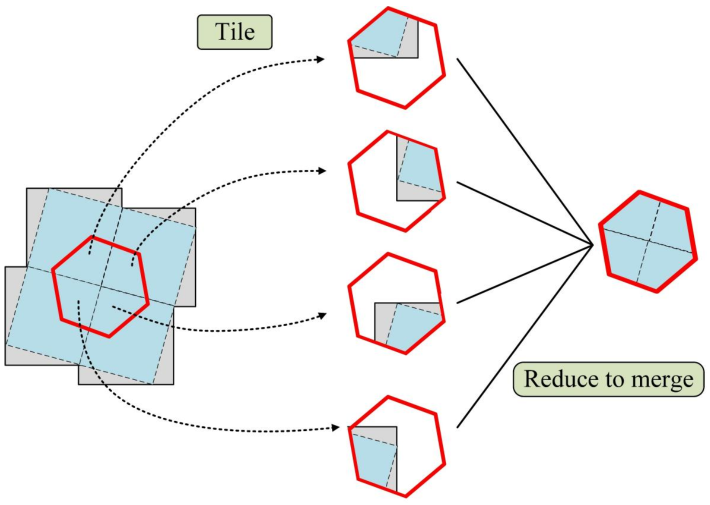

Figure 3. The diagram of merging the tiles intersecting with the same hexagon cell into one tile.

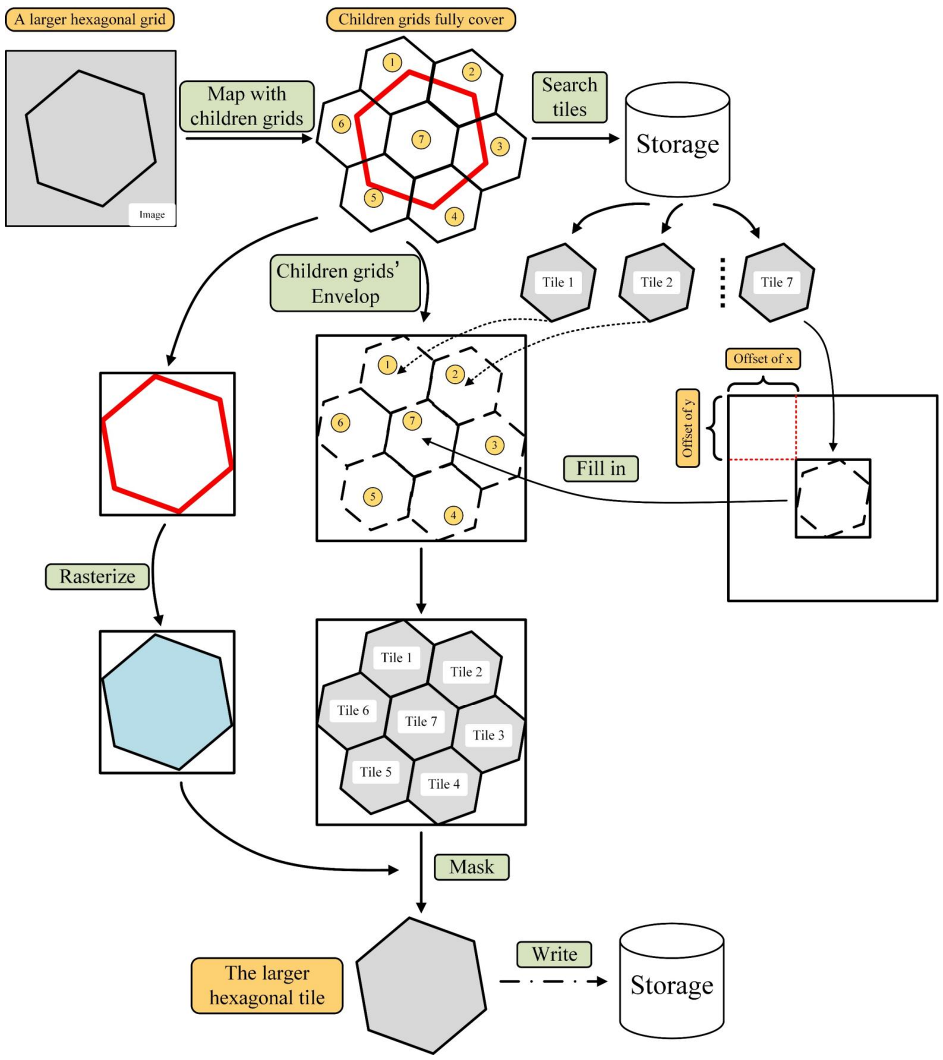

Figure 4. Illustration of stitching hexagon tiles.

The key to generating parent tiles by stitching child tiles is to determine the relationship between the parent and child hierarchical grids. Here, we used H3, so the corresponding relationship is 1:7, or a multiple of 7. Assuming that the level k has been built, the level k-1 is grid h_p. Firstly, all the sub grids covering the h_p are calculated, namely $h_c = \{h^i{}_c, i \in N^+\}$. Then, the coordinate offset of each sub-grid $h^i{}_c$ in the range of h_c, namely, X_{offset} and Y_{offset}, are calculated to determine the relative position of the child grid, as shown in Formula (2). Finally, the corresponding tiles of each child grid are filled into the corresponding position of the canvas to realize hexagonal stitching based on the DGGS hierarchy.

$$\begin{cases} X_{offset} = int \frac{(X_{min} - X^s_{min})}{resolution} \\ Y_{offset} = int \frac{(Y^s_{max} - Y_{max})}{resolution} \end{cases} \tag{2}$$

where X_{min} and Y_{max} are the coordinates of the upper left corner of the envelope rectangle of the parent grid or a single child grid, respectively. $X^s{}_{min}$ and $Y^s{}_{max}$ are the coordinates of the upper left corner of the envelope rectangle of the subgrid set, *resolution* is the real surface resolution represented by the original resolution of the image or the unit pixel in the parent tile after tile combination.

Algorithm 2 shows the pseudocode of hexagonal stitching based on the DGGS hierarchy.

Algorithm 2: The pseudocode of hexagonal stitching.

```
Input: hexagon grids at level k-1 and hexagon tiles at level k
Output: hexagon tiles at level k-1
Calculate the child hexagon grids that fully cover the parent grid in level k.
Map parent grids with the corresponding child grids.
for each child grid do
2.1 Read the corresponding tile.
2.2 Calculate the extent of child cells and its offset in the parent cell.
end
4. Read the corresponding child tiles and stitch the tiles that have the same parent cell into a new raster.
5. Mask the stitched tiles with corresponding parent hexagon cells and render parent hexagon tiles.
```

2.4. HexTile Parallelization in Spark

With the help of the MapReduce programming model and Spark rich operator, as well as GeoTrellis Raster, Tile, and other serialized objects and related functions and methods, the parallel construction of hexagonal DGGS tiles was realized. Figure 5 shows the entire flow chart of the HexTile algorithm in Spark.

(1) Slicing. Hexagon cells that cover the extent of each image at the basic level were calculated to match the key-value pair <*hexagon, image*>. Then, each image was cropped with the corresponding hexagon cells, and the offset of the cropped raster in the hexagon cell was computed. The two steps are executed in the function.

(2) Merging. Additionally, one hexagon cell might partly intersect with several images, at most four, so the next step was to merge these raster images into one raster, which was implemented in the function. Finally, the raster images were masked with the hexagon cell and rendered into hexagon tiles in parallel.

(3) Stitching. In the stitching part, child cells that fully cover their parent cells were firstly acquired, and the key-value pair was matched. Then the pair was flattened to a one-to-one pair using the flat Map function. In the next step, the spatial offset of each child cell in the parent cell was calculated to stitch these corresponding tiles in the correct position. The stitching process was operated in the function. Finally, each stitched tile was masked and rendered into hexagon tiles.

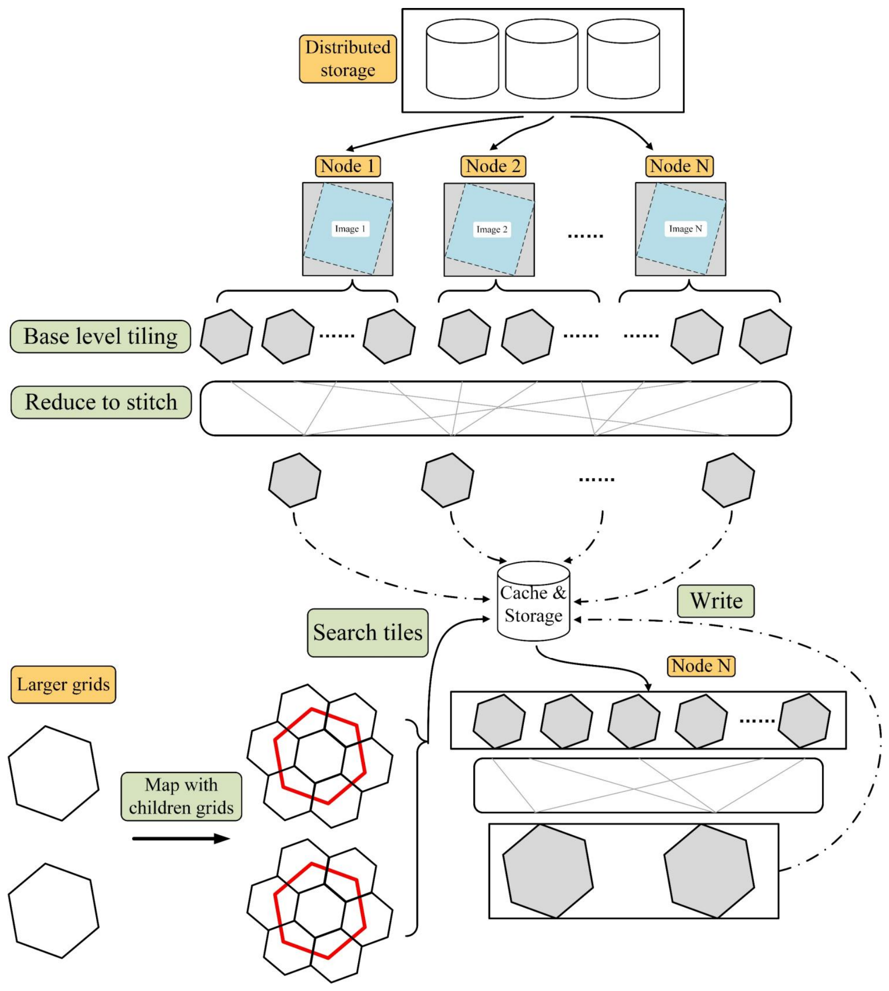

Figure 5. The flow chart of the HexTile algorithm in Spark.

3. Results

3.1. Experiment Environment and Datasets

Experiments were designed to measure the stability and scalability of our algorithms. The experiments were run in a cluster environment that was composed of one master node and three workers. The master node was run on Ubuntu 18.04 with an 8-core Intel CPU @2.1GHz and 16 GB of memory. The three workers were run on Ubuntu 16.04 with a 4-core Intel CPU @2.1GHz and 32 GB of memory. We also applied solid-state drives to workers as temporary caches for Spark. The code was written in Scala, and the programming environments were JDK-1.8, Scala SDK version 2.12.14, Hadoop-3.3.0, and Spark-2.4.7.

The image dataset we used was Landsat-8 data L1T, downloaded from the website of the United States Geological Survey (USGS). We performed some preprocesses that converted the projection coordination to the geographic coordinate system (WGS84, EPSG:4326) and combined bands of blue, green, and red into one GeoTiff file. The spatial extent of the data is all over China, approximately 93 GB in size and comprising 553 scenes. The dataset was stored in the Hadoop Distributed File System (HDFS). The HexTile algorithm was developed based on GeoTrellis. The hexagon discrete global grid system we used in the experiment was H3, developed by Uber.

3.2. Performance of HexTile Algorithm

The first experiment was carried out to estimate the stability and performance of the parallel algorithm from one to three nodes for slicing at level 4. In the second experiment, images were sliced directly from levels 6 to 2. In the third experiment, we selected a different data volume for direct slicing (Table 2) at level 4. The number of hexagon tiles sliced from the datasets is shown below (Table 3).

Table 2. Data volume and the corresponding number of hexagon tiles at level 4.

Data Volume	Number of GeoTiffs	Number of Hexagon Tiles
1 GB	6	158
10 GB	60	1270
20 GB	120	2292
50 GB	360	5870
93 GB	553	8210

Table 3. The number of hexagon tiles sliced from 553 images in different levels.

Level	Number of Hexagon Tiles
L2	168
L3	1183
L4	8210
L5	57,444
L6	402,122

We performed a pre-experiment to adjust the configuration of Spark and chose the best settings for the run-time test. The performance was distinctly improved with the increase in the number of nodes and different levels (Figure 6). We tested the algorithm under the conditions of a single machine environment, and the code was run in the same worker node. It is obvious that the parallel algorithm accelerates the processing of slicing, and the efficiency is enhanced by 3.7 to 10 times. At the same time, for the hexagonal DGGS model, the higher the level, the more time is consumed. This comparison indicates the advantages of parallel processing and the stability of our algorithm.

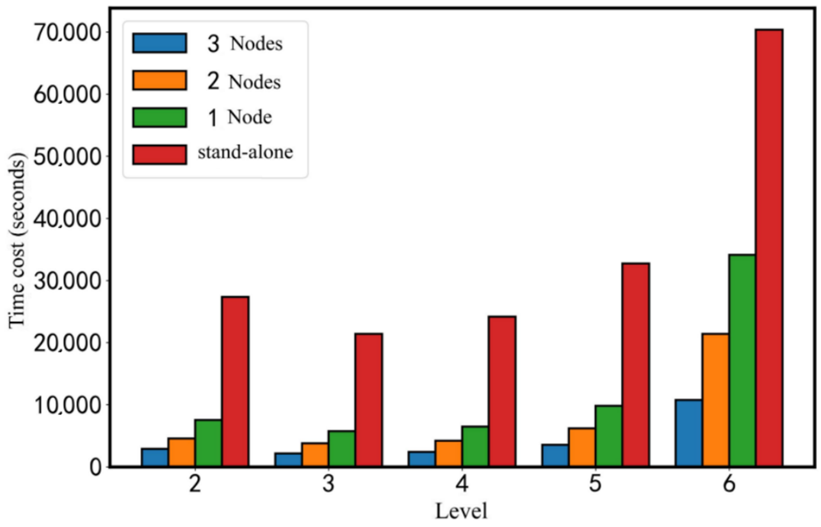

Figure 6. Time-consumption comparison of tile construction with different numbers of nodes and different levels.

The time consumption for each tile generation decreases as the level number increases, even if the number of hexagon tiles is more than 400,000 at level 6 (Figure 7). However, we found that the efficiency of tile generation at level 3 performs better than at level 2. The explanation for this contradiction is that the extent of the hexagon cell at level 2 is much larger than the scene of a Landsat-8 image, and hence a number of images are stitched together to generate a larger raster tile instead of slicing. This experiment shows that the slicing algorithm is suitable for the generation of a large number of tiles.

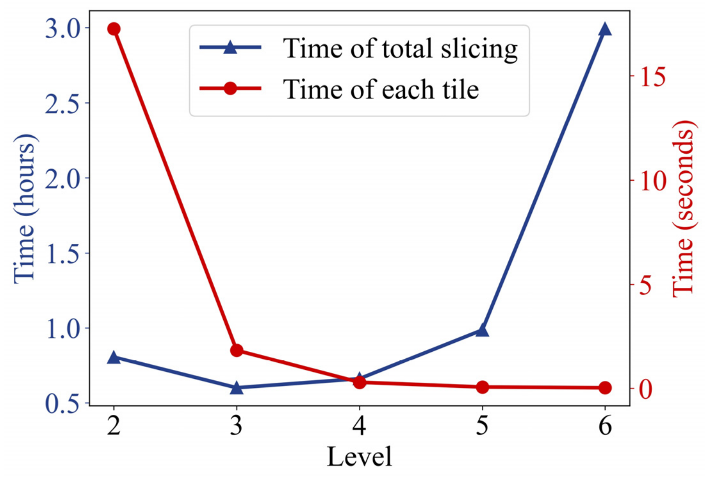

Figure 7. The performance of directly slicing all images at different levels.

We also tested slicing efficiency for different scales of data volume. The result shows that the efficiency of tile generation declines as the data volume grows (Figure 8). Although the total time consumption of slicing for 1 GB of data is less than 2 min, it took more time to generate each hexagon tile than slicing for 10 GB of data. For the slicing of a massive number of images, the total time consumption increases immensely, but it is still acceptable for the generation time of each tile.

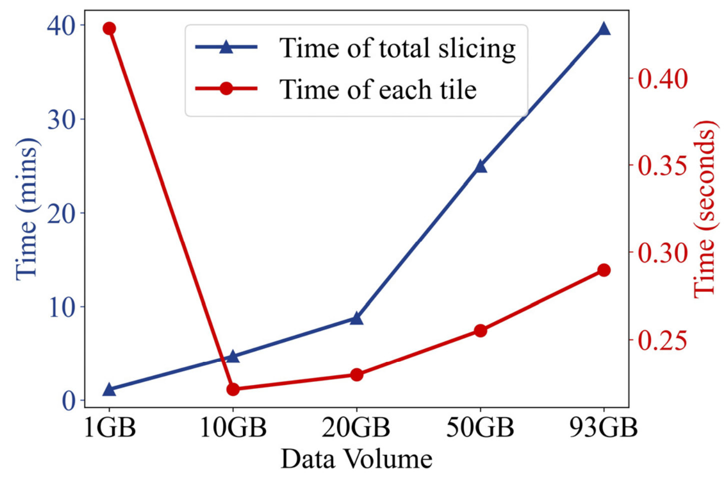

Figure 8. The performance of slicing in different data volumes.

We also designed an experiment to compare the efficiency of traditional direct slicing and the stitching algorithm. The propose of stitching tiles is to reduce the disk read times; however, it would take a large amount of time for the machine to search for certain tiles before stitching. Thus, we tested the time cost of randomly generating hexagon tiles. The comparison results are shown in Figure 9. It only requires about 0.6 s on average to stitch a single tile, showing that the efficiency performance for stitching is at least four times faster than traditional direct slicing. The time cost for each tile is higher than the time consumption in other experiments, which might result from the random sampling time because we recorded the whole running time instead of separate parts. This comparative experiment indicates that it saves a great amount of time through stitching, which can increase efficiency by up to 22 times.

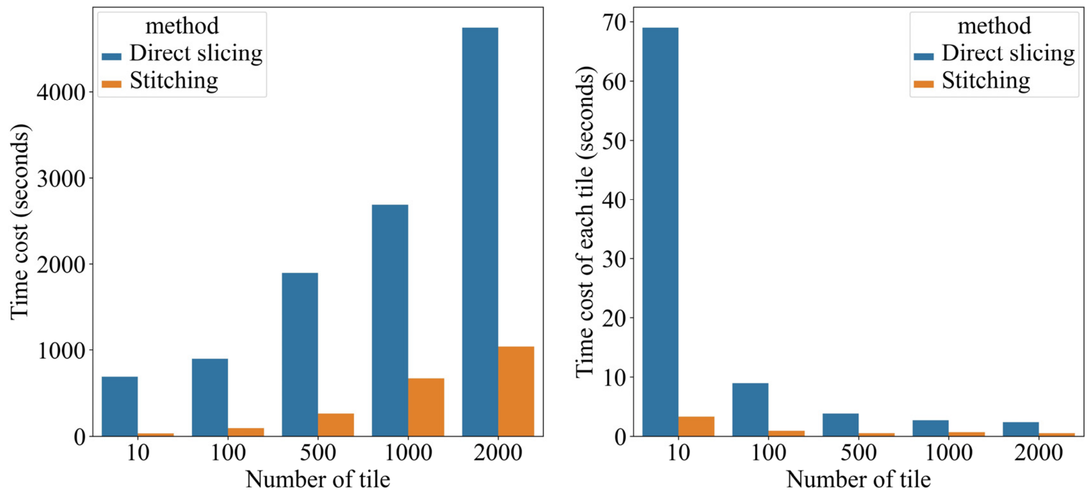

Figure 9. Efficiency comparison between direct slicing and stitching.

3.3. Accuracy Evaluation of Hexagonal Tiles

For the evaluation of the quality of the hexagon grid, two indicators were selected in this study: deviation index of spectral information (*DISI*) and mean root mean square error (*RMSE*) [14]. *DISI* compares the information loss pixel by pixel from a global perspective, while *RMSE* evaluates the geometric features of the image. Manually selected sample points with obvious features in the image were utilized, such as the boundary between land and water and the boundary between a natural surface and artificial surface. A total of 30 sample points (Figure 10) were selected for the calculation of the different study areas a and b. The calculation formulas of the two indexes are (3) and (4).

$$DISI = \frac{1}{Col \cdot Row} \sum_{i=1}^{Col} \sum_{j=1}^{Row} \frac{|W_{i,j} - S_{i,j}|}{S_{i,j}} \tag{3}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (W_i - S_i)^2}{n}} \tag{4}$$

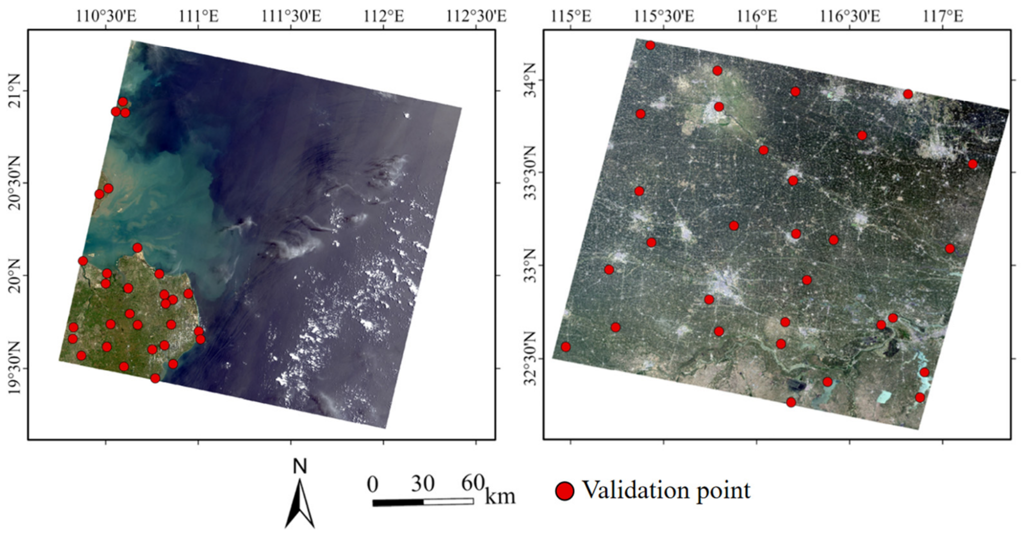

Figure 10. Distribution of sampling points for precision evaluation.

Among them, *Row* and *Col* are the number of rows and columns of the image; *W* and *S* represent the tile-stitched image and the original image, respectively; and *n* is the selected sample point. The smaller the values of the two indices, the higher the precision.

The evaluation index calculates the average value of the accuracy of the three bands of R, G, and B as the result, and the evaluation index calculation results are shown in Table 4 for the different study areas a and b. The accuracy of the tiles obtained by using the slicing method is better than that generated by merging. The larger the number of levels of tile construction, the higher the accuracy of the tile. At the sixth level, the complete data restored from the tiles generated by the slices is very close to the original data, and the error of pixel loss is also minimized. However, the tiles constructed by tile merging have large errors in pixel loss and position offset.

Table 4. Quality accuracy evaluation of hexagon tiles.

Study Areas	**Indices**	*DISI*			*RMSE*		
	Levels	**4**	**5**	**6**	**4**	**5**	**6**
(a)	Slice	0.0347	0.0299	0.0157	2.0324	2.6402	0
	Merge	1.1422	1.2268	1.1238	6.8613	6.3547	5.8274
(b)	Slice	0.0849	0.0228	0.0101	2.0717	1.8140	0
	Merge	1.2670	1.1699	1.0892	7.2619	6.6685	3.9679

The tiles obtained by using the slicing method can better restore the remote sensing image data, so combining the hexagon tiles when calculating with hexagonal DGGS data can guarantee better calculation accuracy. However, the tiles constructed by the merging method have low accuracy and are not suitable for computing, but they are suitable for map visualization.

3.4. Visualization with WebGIS

Map tiles are mainly used for efficient visualization of spatial data. In terms of visualization engine, Cesium and OpenLayers, which are commonly used as carriers, were selected in this study to load and display hexagonal tiles. It is worth mentioning here that the original Cesium and OpenLayers class libraries do not support the display of hexagonal map tiles. Therefore, we designed the mapping relationship between map level, zooming level, and hexagonal discrete grid level to quickly browse the map tiles.

Figure 11 shows Landsat-8 and LAI images in multiple levels visualized with Cesium. The position of the hexagon tile in the map is consistent with the actual spatial position, which also verifies the reliability of the parallel slice. In addition, the multi-level hexagon tile also shows details at different scales.

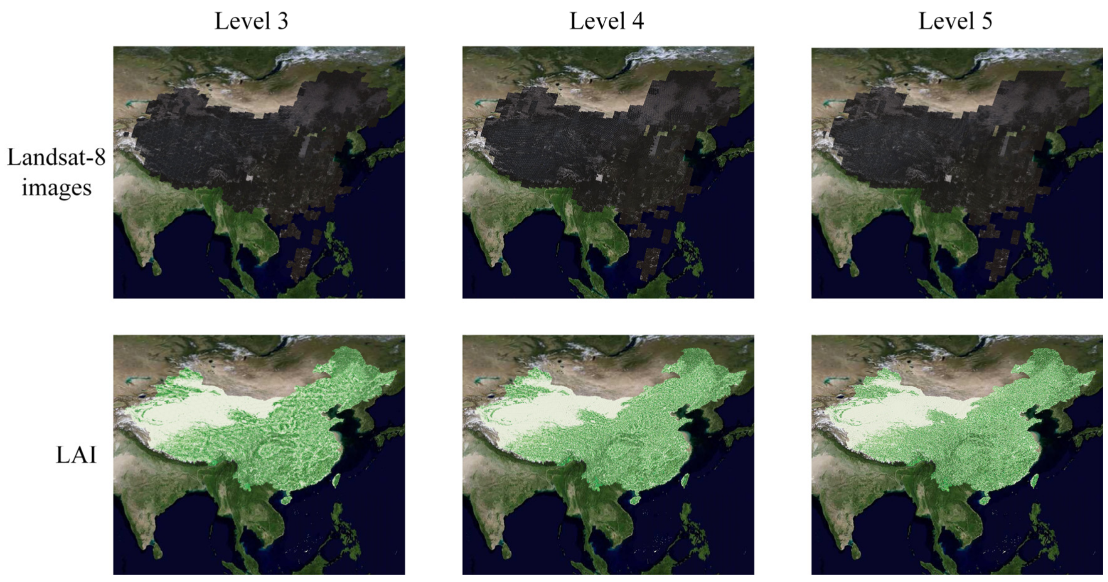

Figure 11. Hexagon tile visualization on the web map.

We also compared hexagon tiles with traditional map tiles loaded through Web Map Service and Web Map Tile Service in GeoServer, and the result is shown in Figure 12. Greenland is chosen as the study area here.

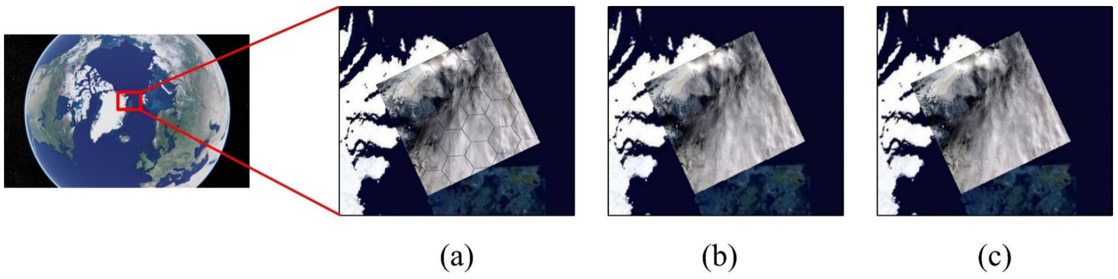

Figure 12. Comparison of hexagon tiles and traditional map tiles for visualization: (**a**) Hexagon tiles generated by HexTile. (**b**) Image loaded through WMS. (**c**) Map tiles loaded through WMTS.

The visualization shows the completeness of the hexagon tiles generated by our hexagonal DGGS-based slicing algorithms. In terms of the hexagon tiles generated by the stitching method, we compared hexagon tiles generated by traditional direct slicing and the tiles generated by the stitching (Figure 13) in OpenLayers. From the perspective of the visualization effect, the suture algorithm proposed in this paper is not only more efficient, but also has no loss in data display quality.

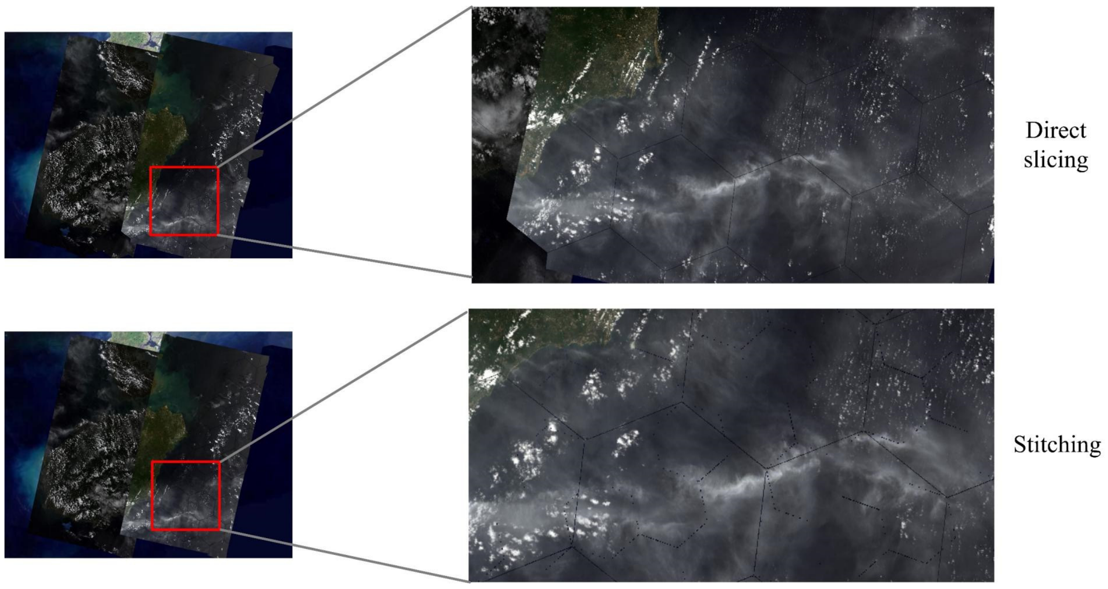

Figure 13. Comparison of different tile generation methods: direct slicing and stitching.

4. Discussion

In the era of big data, visualization has become an effective data analysis method, through which the distribution and change of data can be expressed well. However, how to effectively and truly display the data of globalization has always been a hot research topic. Based on the advantages of the hexagonal discrete global grid system (DGGS), this paper focuses on the construction of algorithms on the server side and the client side and the parallelization of algorithms.

The hexagonal global discrete grid system has unique advantages in spatiotemporal data organization, calculation, and analysis, which have been recognized by previous studies. The visualization of large-scale remote sensing data is still a problem worth studying. As we know, the types and quantities of remote sensing data are constantly increasing and globally oriented, and the globalization and large-scale applications based on remote sensing data are increasingly rich. Therefore, the research content of this paper combines the global discrete grid system with remote sensing big data, and through data slicing, merging, stitching and other algorithms, a map tile technology based on hexagonal DGGS was realized, which effectively solves various deformation problems caused by the projection of traditional map tiles in high latitudes and polar regions. This also brings great benefits for further data calculation and data fusion, because globally homogeneous tiles can also effectively solve problems such as calculation hotspots and global unified models.

In order to solve the efficiency problem caused by large-scale remote sensing data, the Spark cloud computing platform was adopted in the algorithm construction in this study. Spark technology has been proven to have significant advantages in terms of big data computing efficiency, and this view is also verified by a series of experiments in this study. During the construction of the Spark platform, the efficiency of hexagonal DGGS tiles is dozens of times that of the traditional single-machine algorithm. In addition to the above experiments, this study also combined the Cesium and OpenLayers class libraries to visualize the tile data in the front end and evaluate the accuracy of the data, which achieved good results.

5. Conclusions

In this paper, a HexTile algorithm is proposed, which is mainly based on the hexagonal DGGS model and Spark cloud computing platform to solve the visualization problems caused by big remote sensing data. The HexTile algorithm mainly realizes hexagon slicing, fragment merging and multilevel tile suturing of large-scale remote sensing data. In terms of algorithm verification, a large number of experiments were conducted to compare the efficiency of algorithms and verify the quality of data visualization, and the loading and display of hexagonal tiles were integrated with Cesium and OpenLayers. The results show that the HexTile algorithm proposed in this paper can not only improve the implementation of big remote sensing data visualization efficiently but also maintain the quality of data visualization well. At the same time, the algorithm can be effectively integrated with existing data engines.

However, the HexTile algorithm has room for improvement, such as in conjunction with spatial indexing, and the uncertainty of the boundary. In addition, it is also a solution to develop a visualization engine suitable for hexagonal DGGS from the bottom, but at present, this is still challenging.

Author Contributions: Conceptualization, Xiaochuang Yao, Guoqing Li and Dehai Zhu; methodology, Xiaochuang Yao and Guojiang Yu; software, Guojiang Yu; validation, Guojiang Yu and Long Zhao; formal analysis, Guojiang Yu; investigation, Xiaochuang Yao; resources, Xiaochuang Yao and Shuai Yan; data curation, Guojiang Yu and Shuai Yan; writing—original draft preparation, Guojiang Yu; writing—review and editing, Xiaochuang Yao; visualization, Guojiang Yu; supervision, Xiaochuang Yao, Guoqing Li and Dehai Zhu; project administration, Xiaochuang Yao; funding acquisition, Xiaochuang Yao. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Department of Science and Technology of Henan Province through grant 201400210100 and the National Key R&D Program of China through grant 2019YFE0127000. This work was also supported by the National Supercomputing Center in Zhengzhou.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset used in this paper are supported by the United States Geological Survey (USGS) from https://earthexplorer.usgs.gov/ (accessed on 10 September 2021).

Acknowledgments: We are grateful for the comments and contributions of the anonymous reviewers and the members of the editorial team.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Guo, N.; Xiong, W.; Wu, Q.; Jing, N. An Efficient Tile-Pyramids Building Method for Fast Visualization of Massive Geospatial Raster Datasets. *Adv. Electr. Comput. Eng.* **2016**, *16*, 3–8. [CrossRef]
2. Sahr, K.; White, D.; Kimerling, A.J. Geodesic Discrete Global Grid Systems. *Cartogr. Geogr. Inf. Sci.* **2003**, *30*, 121–134. [CrossRef]
3. Rawson, A.; Sabeur, Z.; Brito, M. Geospatial Data Analysis for Global Maritime Risk Assessment Using the Discrete Global Grid System. In Proceedings of the 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS, Brussels, Belgium, 11–16 July 2021; pp. 3904–3907.
4. Hojati, M.; Robertson, C.; Roberts, S.; Chaudhuri, C. GIScience research challenges for realizing discrete global grid systems as a Digital Earth. *Big Earth Data* **2022**, *6*, 358–379. [CrossRef]
5. Yao, X.; Li, G. Big spatial vector data management: A review. *Big Earth Data* **2018**, *2*, 108–129. [CrossRef]
6. Zhao, X.S.; Bai, J.J. Hierarchical model of global discrete grids based on diamonds. *J. China Univ. Min. Technol.* **2007**, *36*, 397.
7. Zhao, L.; Li, G.; Yao, X.; Ma, Y.; Cao, Q. An optimized hexagonal quadtree encoding and operation scheme for icosahedral hexagonal discrete global grid systems. *Int. J. Digit. Earth* **2022**, *15*, 975–1000. [CrossRef]
8. Zhou, M.; Chen, J.; Gong, J. A pole-oriented discrete global grid system: Quaternary quadrangle mesh. *Comput. Geosci.* **2013**, *61*, 133–143. [CrossRef]
9. Li, M.; McGrath, H.; Stefanakis, E. Geovisualization of Hydrological Flow in Hexagonal Grid Systems. *Geographies* **2022**, *2*, 227–244. [CrossRef]
10. Li, M.; Stefanakis, E. Geospatial Operations of Discrete Global Grid Systems—A Comparison with Traditional GIS. *J. Geovis. Spat. Anal.* **2020**, *4*, 26. [CrossRef]
11. Górski, K.M.; Hivon, E.; Banday, A.J.; Wandelt, B.D.; Hansen, F.K.; Reinecke, M.; Bartelmann, M. HEALPix: A Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere. *Astrophys. J.* **2005**, *622*, 759. [CrossRef]
12. Gibb, R.G. The rHEALPix Discrete Global Grid System. *IOP Conf. Ser. Earth Environ. Sci.* **2016**, *34*, 012012. [CrossRef]
13. Zhao, X.; Wang, L.; Wang, H.; Li, Y. Modeling methods and basic problems of discrete global grids. *Geogr. Geo-Inf. Sci.* **2012**, *28*, 29–34.
14. Ma, Y.; Li, G.; Yao, X.; Cao, Q.; Zhao, L.; Wang, S.; Zhang, L. A Precision Evaluation Index System for Remote Sensing Data Sampling Based on Hexagonal Discrete Grids. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 194. [CrossRef]
15. Robertson, C.; Chaudhuri, C.; Hojati, M.; Roberts, S.A. An integrated environmental analytics system (IDEAS) based on a DGGS. *ISPRS J. Photogramm. Remote Sens.* **2020**, *162*, 214–228. [CrossRef]
16. Yan, S.; Yao, X.; Zhu, D.; Liu, D.; Zhang, L.; Yu, G.; Gao, B.; Yang, J.; Yun, W. Large-scale crop mapping from multi-source optical satellite imageries using machine learning with discrete grids. *Int. J. Appl. Earth Obs. Geoinf.* **2021**, *103*, 102485. [CrossRef]
17. de Sousa, L.M.; Leitão, J.P. HexASCII: A file format for cartographical hexagonal rasters. *Trans. GIS* **2018**, *22*, 217–232. [CrossRef]
18. Eldawy, A.; Mokbel, M.F. SpatialHadoop: A MapReduce framework for spatial data. In Proceedings of the 2015 IEEE 31st International Conference on Data Engineering, Seoul, Republic of Korea, 13–17 April 2015; pp. 1352–1363.
19. Aji, A.; Wang, F.; Vo, H.; Lee, R.; Liu, Q.; Zhang, X.; Saltz, J. Hadoop-GIS: A High Performance Spatial Data Warehousing System over MapReduce. *Proc. VLDB Endow.* **2013**, *6*, 1009–1020. [CrossRef]
20. Jia, Y.; Wu, J.; Sarwat, M. GeoSpark: A cluster computing framework for processing large-scale spatial data. In Proceedings of the 23rd SIGSPATIAL International Conference, Seattle, WA, USA, 3–6 November 2015; pp. 1–4.
21. Eldawy, A.; Mokbel, M.F.; Jonathan, C. HadoopViz: A MapReduce framework for extensible visualization of big spatial data. In Proceedings of the 2016 IEEE 32nd International Conference on Data Engineering (ICDE), Helsinki, Finland, 16–20 May 2016; pp. 601–612.
22. Yu, J.; Zhang, Z.; Sarwat, M. GeoSparkViz: A Scalable Geospatial Data Visualization Framework in the Apache Spark Ecosystem. In Proceedings of the 30th International Conference on Scientific and Statistical Database Management, Bozen-Bolzano, Italy, 9–11 July 2018; pp. 1–12.
23. Wan, L.; Huang, Z.; Peng, X. An Effective NoSQL-Based Vector Map Tile Management Approach. *ISPRS Int. J. Geo-Inf.* **2016**, *5*, 215. [CrossRef]
24. Tang, X.; Yao, X.; Liu, D.; Zhao, L.; Li, L.; Zhu, D.; Li, G. A Ceph-based storage strategy for big gridded remote sensing data. *Big Earth Data* **2022**, *6*, 323–339. [CrossRef]
25. Yao, X.; Li, G.; Xia, J.; Ben, J.; Cao, Q.; Zhao, L.; Ma, Y.; Zhang, L.; Zhu, D. Enabling the Big Earth Observation Data via Cloud Computing and DGGS: Opportunities and Challenges. *Remote Sens.* **2019**, *12*, 62. [CrossRef]