ISPRS Journal of Photogrammetry and Remote Sensing 163 (2020) 202–213

Contents lists available at ScienceDirect

ISPRS Journal of Photogrammetry and Remote Sensing

journal homepage: www.elsevier.com/locate/isprsjprs

Global multi-scale grid integer coding and spatial indexing: A novel approach for big earth observation data

Yi Lei[a,1], Xiaochong Tong[a,*,1], Yongsheng Zhang[a], Chunping Qiu[a], Xiangyu Wu[a], Guangling Lai[a], He Li[a], Congzhou Guo[a], Yong Zhang[b]

[a] Information Engineering University, 450001 Zhengzhou, China
[b] Zhengzhou Zhonghe Jingxuan Information Technology Co., Ltd, 450001 Zhengzhou, China

ARTICLE INFO

Keywords:
Multi-scale grid
Integer Coding
Clustering property
Data management
Big earth observation data
Spatial indexing and querying

ABSTRACT

With the exponentially growing earth observation data of specific sensor-determined resolutions and update frequencies, earth observation has irreversibly arrived in the Big Data era, enabling new insights in science and engineering. With great opportunity comes great challenges regarding efficient and effective data management because earth observation data is of different scales and characterized by complexity in spatial relationships related to the real world. To overcome the challenges is crucial for, for instance, data mining, land surveying, and especially emergency mapping for disaster response. To improve the querying efficiency of big earth observation data, we proposed a novel data management approach: Global Multi-scale Grid Integer Coding and Spatial Indexing. Among our contributions are: (1) proposing Global Multi-scale Grid Integer Coding Model (GMGICM), which presents clustering property in both the scale dimension and spatial dimension, and theoretically facilitates an efficient querying; (2) deliberately applying GMGICM on multi-scale earth observation data spatial indexing, which results in one-dimensional data index, which can be queried using simple B-tree, inversion, and other one-dimensional indexes; (3) designing a strategy to assure the completeness of spatial querying, which is not well solved by existing grid-based coding models. The advantages of our proposed approach have been demonstrated with both simulated and real remote sensing data, with spatial operation 20 times as fast as Geohash and spatial querying 10 times as fast as Oracle Spatial on average. The proposed approach can be easily adapted for three or higher-dimensional earth observation data and bring potential benefit to all big earth observation data analytic projects.

1. Introduction

Earth Observation (EO) satellites currently account for over one third of the operational satellites orbiting the Earth. This features a rapidly increasing industry with various platforms and acquisition modes, e.g., from optical (multi- and hyperspectral), Lidar, and synthetic aperture radar (SAR) sensors (Klein et al.,2015; Waterfall et al., 2016). The rapidly increasing Small Satellites Constellations are also attracting more and more attention. For instance, Planet Labs (PL) has launched almost 300 cube-satellites since 2013, and more than half is still active[i].

During the last decades, advances in satellite systems have increased the volume, variety, velocity and veracity of Earth Observation (EO) data, leading to massive and ever-growing EO data archives characterized by multi-source, multi-resolution, and multi-temporal (Gilberto et al., 2016; Deren et al., 2017). Earth observation satellites have produced unprecedented peta-bytes or even exabyte of geospatial data (Qi et al., 2017). To effectively and efficiently manage these large data sets for global applications, we need, more than ever, stable and efficient solutions that are supportive to analytical tasks to retrieve useful information, and eventually foster the fifth 'V', namely value of EO big data (Cian et al., 2018).

Currently, there are mainly the following two approaches for big EO data management. This category is based on how the datasets are structured. One is satellite trajectory-based or scene-based. Examples are NASA's Earth Observation System Data and Information System (Maex, 2007) and China's Ground Data Storage and Management System for Earth Resources Satellite (Wang, 2009). This approach is

* Corresponding author.
E-mail address: txchr@aliyun.com (X. Tong).
[1] Yi Lei and Xiaochong Tong are co-first authors.
[i] https://www.ucsusa.org/nuclear-wea pons/space-weapons/satellite-database#

https://doi.org/10.1016/j.isprsjprs.2020.03.010
Received 29 September 2019; Received in revised form 14 February 2020; Accepted 16 March 2020
Available online 23 March 2020

suitable for management of high dynamic, multi-source satellite datasets and related post-processed products. However, the spatial correlation of the multi-source data in the same area is poor with the increase of the types and sources of EO data, due to a lack of uniform segmentation standards between the track strips and the scenes of EO data centers, as well as a lack of geoscience meaning of the data. In this case, it is difficult to retrieve, integrate and share EO data in the same area, which reduces the efficiency of data usage.

Another is pyramid-based image block. Examples are Google Earth and Google Map (Gibin et al., 2008) and China's national platform for common geospatial information service (Lu et al., 2012). This approach is suitable for static data with highly efficient data querying operations, and mainly used for presentation of images that cover the whole global surface. It, however, is limited for a unified management of multi-source and multi-temporal EO data in the same area. In addition, the generation of the pyramid images, will not only produce lots of data tiles, resulting in much demand of data storage space (Yin et al., 2017); but also raise up the time consumption of data preprocessing. Thus, this approach is not suitable for the management of highly dynamic EO data.

To realize effective and efficient distributed data management and spatial querying for massive EO dataset, a popular direction is to use grid coding methods within the geographic grid system, the goal of which is mainly to integrate geospatial positioning and geographic feature descriptions into the grids ranging from centimeter scale to the global scale (Tong and Ben, 2016; Raposo et al., 2019). The basic idea is to spatially relate the grid index to remote sensing metadata that is independently stored and managed with image files, and the following image retrieving and spatial querying can be performed on basis of the encoded grid index. For instance, researchers have employed grid index based on linear quadtree for global Remote Sensing data subdivision organization (Song et al.,2014), proposed a multi-level grid model for massive remote sensing data storage and management (Li, et al., 2016), studied query evaluations over multi-dimensional geospatial datasets (Malensek et al., 2017), investigated the retrieval strategy for massive remote sensing metadata based on Geohash coding (Huang and Wang, 2018), and proposed a geographic meshing and coding method based on adaptive Hilbert-Geohash (Guo et al., 2019). While promising, the following challenges remain in the grid coding-based solutions.

Firstly, the grid code does not have good clustering property in scale dimension. Geohash (Moussalli et al., 2015; Malensek et al., 2017) and GeoSOT (Lin et al., 2015; Li, et al., 2016) are both based on quad-tree grid model, which recursively divides the space into sub-space of multiple levels, and they use space filling curve and level of grids to encode multi-scale grids. This kind of coding methods is of good spatial locality within the same level of sub-space, but not across different levels, which causes bad continuity of space filling curve in multi-scale grid model, i.e., the spatial relationships of the encoded grids are missing, meanwhile, hinders the operational efficiency of the coded index.

Secondly, linear tree-based indexing does not support data with complex spatial relationships well (Jensen et al., 2006). It is suitable for managing grid codes of different scales. However, it is more appropriate for point objects at a specified grid level (Davis et al., 2018), instead of line or surface objects with various complex spatial relationships including overlaps, touches, within, and crosses. This is because it only uses the leaf node for management, not the root nodes or intermediate child nodes. This leads to its limitation of being used in structuring big EO data.

Thirdly, the spatial association between EO data and the grid is poor. It is to establish the correlation between spatial location information of the entry and the grid set. The associated grids of the existing methods are mostly single-scale, and the setting of the number of associated grids does not take account of the characteristics of remote sensing data, resulting in large redundancy of the correlated grid regions, resulting in low query accuracy (Song et al.,2014; Malensek et al., 2017).

Lastly, grid-based spatial querying is subject to incomplete-ness. This leads to that target data is missing when the query objects are areas (Jin et al., 2013). This is because the query operation only considers whether the target area includes or intersects with each candidate grid, but does not consider whether the target area is included by a candidate grid. This is due to the commonly used query algorithms: string-match-based and grid subsets query (Huang and Wang, 2018). For instance, when searching for images of Technical University of Munich, Germany, images that cover the whole city Munich tend to be omitted in the querying results.

Based on the analysis, we propose in this work a novel Global Multi-scale Grid Integer Coding Model (GMGICM) with unique clustering property in spatial dimension and scale dimension (Section 2), based on which a spatial indexing and querying approach is designed for big EO data of different scales (Section 3). The core design of our approach is to relate EO data to the proposed integer code with three reasonable restrictive conditions. In addition, we designed a strategy to assure the completeness of spatial query. Our approach will be tested with real big EO data (Section 4) after conducting simulated data-based experiments including clustering property analysis of GMGICM codes and comparison of son-grids querying between GMGICM and Geohash. Discussions and analysis about experiments are provided (Section 5) before Section 6 concludes the work and proposes further improvement.

2. Global multi-scale grid integer coding model (GMGICM)

2.1. Multi-scale grid subdividing and coding methods

The proposed Global Multi-scale Grid Integer Coding Model (GMGICM) mainly includes multi-scale grid subdividing and coding methods. The grid subdividing method of GMGICM is based on quad-tree and longitude-latitude grid subdivision (Hartmann et al., 2016), as is shown in Fig. 1. Considering that longitude-latitude grids are not symmetrical and quad-tree requires symmetry in both directions, the north–south extent (latitude direction) of the earth surface is extended from latitude 90° and latitude −90° to latitude 180° and latitude −180°, respectively, to be the same as the east–west extent. The extended parts, the blank areas shown in Fig. 1, is not existing in real-world case but still encoded by GMGICM. Take the commonly used World Geodetic System 1984 (WGS-84) as an example, the first level, level 0, consisting only one grid centered in (0°, 0°), represents the whole sphere. By partitioning the grid into four sub-grids, we can obtain the second level, level 1, consisting four grids, each of them representing 1/4 of the whole sphere. Similarly, level 2, consisting of 16 grids, represents 1/8 of the whole surface. Following this method, level 31 represents $1/8 \times 1/4^{29}$ of the whole surface, corresponding to $45°/4^{29} \times 45°/4^{29}$ area, which is 1.86 cm × 1.86 cm at the earth's Equator when the Earth radius is 6378.140 km. This 31th level is sufficient for the position accuracy of all state-of-the-art satellite images (Zhong et al., 2018).

The multi-scale grid coding method of GMGICM employs a m-bit unsigned integer to uniformly identify each grid at each level, where the integer is a GMGICM code. Specifically, we use $m = 64$ because 64-bit computer system is currently common. In this way, both X and Y directions in Fig. 1 can use 31 bits, and the left two bits is to identify the levels.

The encoding and decoding methods of GMGICM codes in 31th level (the max level of grid subdivision) are based on Z-curve, a type of widely-used space filling curves (Lawder and King, 2000) for grid coding models, such as Geohash (Guo et al., 2019) and GeoSOT (Qian et al., 2019). Z-curve, also known as Morton curve (Morton, 1996), shown in Fig. 2(a) (when the lower left corner of grid space is taken as the starting point of the Z-curve), helps to convert the high-dimensional coordinates to codes, so that the high-dimensional spatial index can be converted to a one-dimensional code index, which can improve the

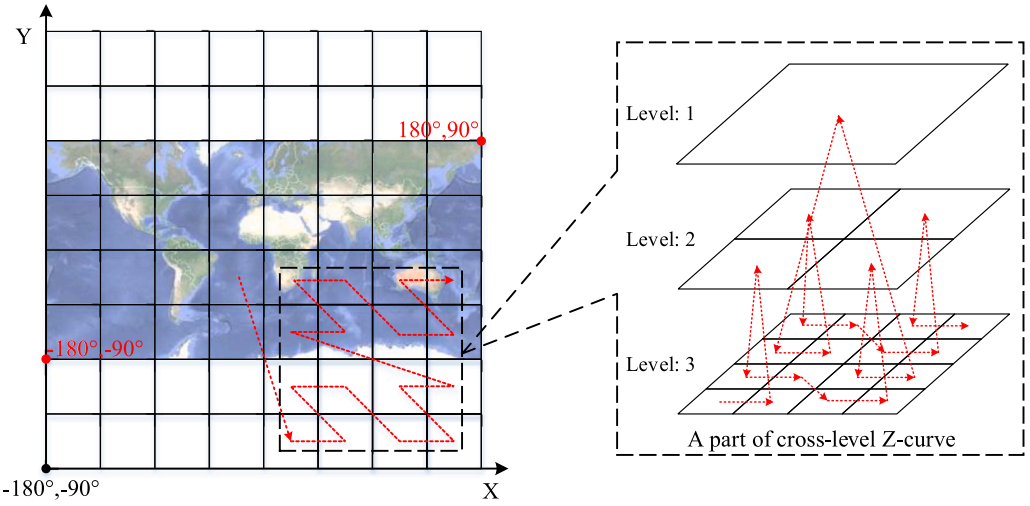

Fig. 1. Multi-scale longitude-latitude grid subdividing method and an illustration of cross-level Z-curve used in GMGICM.

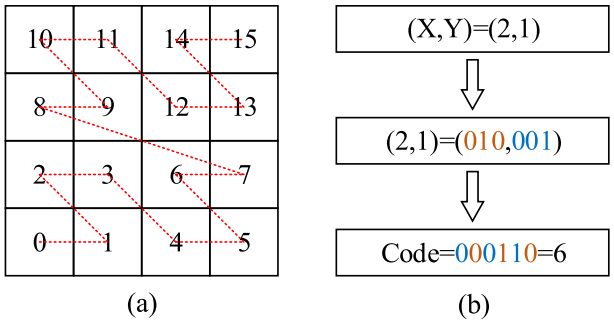

Fig. 2. Construction of Z-curve in third level and encoding process of a Z-curve code.

indexing and querying efficiency of spatial data (Li et al., 2019). As shown in Fig. 2 (b), the encoding process of a Z-curve code in third level includes: grid coordinates conversion from decimal to binary, converting the binary grid coordinates to a code using the cross-bit method (Griffiths, 1986) and the code conversion from binary to decimal. And the decoding process is the reverse of the encoding process.

Following the encoding method above, we can get the Z-curve code set $\cup_{i=0}^{i\le 2^{62}-1}\{i\}$ of 31th level, and the GMGICM code set of 31th level is $\cup_{i=0}^{i\le 2^{62}-1}\{2\bullet i\}$. Then taking the average of code values of four neighbors in 31th level, the 30th level code set can be calculated, and in the same way, code sets of all left levels can be obtained recursively. Eventually, codes of all levels form an inverted recursive quad-tree as shown in Fig. 3. Moreover, linking grids of all levels in a sequential order of their code values, we developed one special space filling curve for multi-scale grids (the dotted line in Fig. 3), defined as cross-level Z-curve. A part of cross-level Z-curve in the gird subdivision is illustrated in Fig. 1, when taking the level as the third dimension. The cross-level Z-curve has good clustering property both in scale and spatial dimension, which will be further analyzed and compared to the traditional Z-curve in Section 2.3.

Hilbert curve, another type of widely-used space filling curves, does have better clustering property than Z-curve (Faloutsos et al., 1989, Zhai et al., 2018). However, the coding efficiency of Hilbert curve is much lower than Z-curve, because codes for Z-curve are calculated using the cross-bit method (Griffiths, 1986), and codes for Hilbert curve are usually mapping transformed form Z-curve codes, or calculated using recursive algorithms (Liu and Schrack, 1996). Considering the efficiency for applications at a large-scale, therefore, GMGICM used Z-curve as an example in the paper. It should be mentioned that within the construction approach of our proposed GMGICM, the Z-curve can also be replaced by Hilbert curve to improve its clustering property in multi-scale dimensional space.

2.2. Main computations of GMGICM codes

Fig. 4 shows main computations of GMGICM codes, including Computation A: conversion between latitude and longitude coordinates and the code, Computation B: operation of the code level, Computation C: operation of neighbor grid codes, Computation D: operation of son-grid codes, Computation E: operation of the father-grid code.

Specifically, using Computation A, one coordinate (L, B) together with level N is converted to a GMGICM code; the GMGICM code can be converted to (L, B) and N by Computation B and the reverse of Computation A. The level of a code is calculated with computation B. Besides, the neighbor-gird codes, son-grid codes and father-grid code and can be obtained by Computation C, D and E, respectively. The detailed operation process of each computation will be described in the following subsections.

Computation A: Conversion between latitude and longitude coordinates and the code.

This conversion method is as shown in the formula (1). Specifically, L and B take WGS-84 as the coordinate system, whose unit is degree, and N is the level of the GMGICM code (Mc). $(x_N\cdots x_3x_2x_1)_2$ and $(y_N\cdots y_3y_2y_1)_2$ represent the binary number corresponding to the grid coordinates X and Y, respectively.

$$\begin{cases} (x_N\cdots x_3x_2x_1)_2 = (L+180)\cdot 2^N/360 \\ (y_N\cdots y_3y_2y_1)_2 = (B+180)\cdot 2^N/360 \\ Zcode = (y_Nx_N\cdots y_2x_2y_1x_1)_2 \\ Mc = (1\ll(62-N-N))-1+Zcode\ll(63-N-N) \end{cases} \quad (1)$$

Using the encoding method of a Z-curve code in Section 2.1 to calculate $Zcode$, we can get $Mc = (1\ll(62-N-N))-1+Zcode\ll(63-N-N)$, which are based on binary operations (Bylinski, 1989) that are extremely efficient.

Computation B: Operation of code level.

There are total 32 levels in GMGICM, corresponding to different integer codes and grids of different sizes in different levels. The level of GMGICM codes should be first determined before further complex operations. Given a GMGICM code Mc, its level N can be determined by:

(1) If Mc is an even number, i.e., $Mc\&1 = 0$, then $N = 31$;
(2) If Mc is an old number, then $N = n \gg 1$, where n is the number of digits to the left of 64-bit integer $Mid = (Mc-1)^{\wedge}(Mc+1)$, using bifurcation method (Tong et al., 2019) to calculate the number. Calculating Mid with XOR exploration is to determine the closest

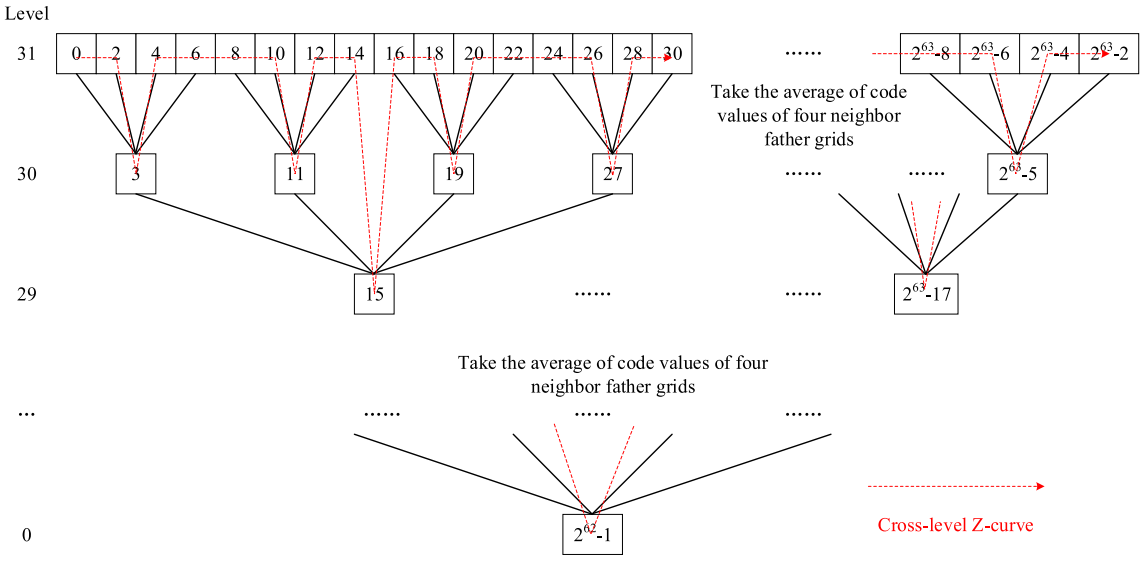

Fig. 3. Code values of all levels and inverted recursive quad-tree scheme in GMGICM.

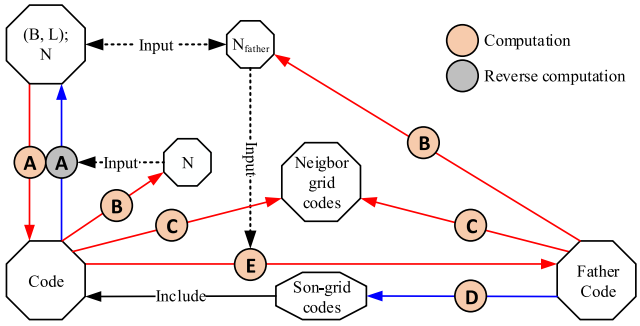

Fig. 4. Main computations of GMGICM codes and their relationships.

common father code of $(Mc - 1)$ and $(Mc + 1)$.

Computation C: Operation of neighbor grid codes.

Given a GMGICM code Mc, its eight neighbor grid codes can be determined by:

(1) Calculate the level N of Mc with the operation of code level;
(2) Using the reverse conversion between latitude and longitude coordinates and the code, the grid coordinate (X, Y) is converted from Mc;
(3) Compute the gird length of level N in grid coordinate, which is given as $It_N = 1 \ll (31 - N)$;
(4) Grid coordinates of eight neighbor grids are respectively $(X - It_N, Y + It_N)$, $(X, Y + It_N)$, $(X + It_N, Y + It_N)$, $(X + It_N, Y)$, $(X + It_N, Y - It_N)$, $(X, Y - It_N)$, $(X - It_N, Y - It_N)$, $(X - It_N, Y)$;
(5) Codes of eight neighbor girds could be converted form their gird coordinates by the formula (1).

Computation D: Operation of son-grid codes.

Son-gird codes of a GMGICM code Mc can be calculated by the formula (2), where N is the level of Mc, N_{son} is the level of son-grid codes, and $[A, B]$ is the number range of son-grid codes.

$$\begin{cases} Mc_0 = (1 \ll (62 - N - N)) - (1 \ll (62 - N_{son} - N_{son})) \\ [A, B] = [Mc - Mc_0, Mc + Mc_0] \end{cases} \quad (2)$$

Besides, the amount of son-grid codes is $4 \times (N_{son} - N)$, and the i-th of son-grid is $A + i \bullet It_{N_{son}}$, $It_{N\,son} = 1 \ll (31 - N_{son})$.

Computation E: Operation of father-grid code.

The Father-gird code FMc of a GMGICM code Mc can be calculated by the formula (3), where N is the level of Mc, N_{father} is the level of FMc.

$$\begin{cases} FMc_0 = (1 \ll (62 - N - N)) - 1 \\ \Delta FMc = Mc \ll ((N - N_{father}) \ll 1) \\ FMc = FMc_0 + \Delta FMc \end{cases} \quad (3)$$

To sum up, the coding method, mainly composed of Computation A and B shown in Fig. 4, is the key for building GMGICM, and one of important contributions in this paper. The encoding process of a GMGICM code is done by Computation A. If taking the computational complexity of a Z-curve code as $O(N)$, the computational complexity of encoding a GMGICM code is $O(N) + O(1) = O(N)$, where N is the level of the code, and $O(1)$ represents the computational complexity from *Zcode* to *Mc* in formula (1). The decoding process of a GMGICM code is done by Computation B and reverse of Computation A. Computation B adopts the bifurcation method (Tong et al., 2019) to calculate the level of a GMGICM code, so the computational complexity of Computation B is $O(\log(N))$. And the computational complexity of reverse of Computation A is equal to Computation A, $O(N)$. Therefore, the computational complexity of decoding a GMGICM code is $O(\log(N)) + O(N) = O(N)$.

2.3. Characteristics of GMGICM codes

Because of its definition and the resulting operations, our proposed GMGICM codes have the following characteristics:

Firstly, the code is a 64-bit unsigned integer, advantageous to computation of GMGICM codes, which is encoded form the longitude and latitude coordinate. Thus, it helps to convert the two-dimension spatial index to one-dimensional code index, and improve spatial indexing and query efficiency.

Secondly, each code has a unique corresponding grid. Specifically, multi-scale grids in GMGICM belong to static grids, each grid has a unique earth surface space corresponding thereto, moreover, each grid of each level has a unique integer code corresponding thereto.

Thirdly, the code with the corresponding gird has recursion. Specifically, each grid is divided into four son-grids in GMGICM, which adopts the recursive quad-tree scheme. And codes of all levels form an inverted recursive quad-tree, which is shown in Fig. 3.

Lastly, the code has good clustering property both in scale and spatial dimension. GMGICM uses the cross-level Z-curve shown in the Fig. 5(b) and (c), linking the code from all levels according to the sequence of code value, which not only makes the code of adjacent grid in same level have the proximity, but also in adjacent levels. However, the multi-scale Geohash shown in the Fig. 5(a), the wildly used coding

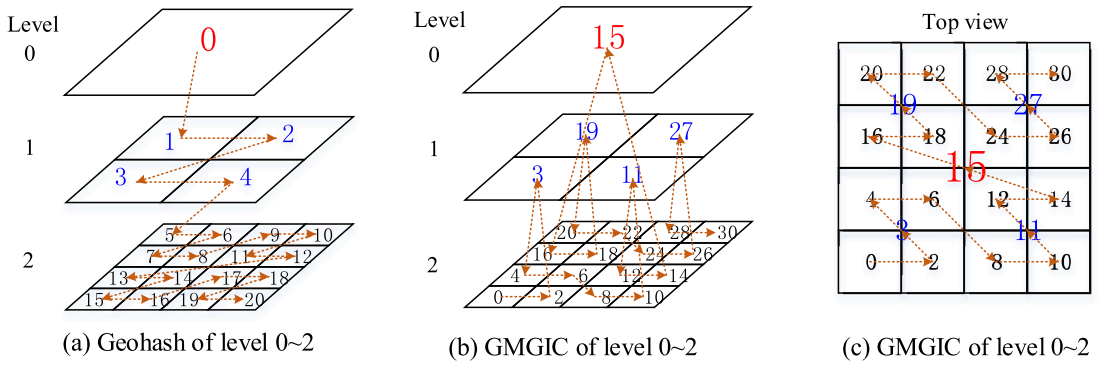

Fig. 5. Clustering comparison of GMGICM codes and multi-scale Geohash codes.

method mentioned in the introduction, uses anther type of Z-curve, the level-by-level Z-curve, linking the code level by level according to the sequence of code value, which causes the level-by-level Z-curve to have worse continuity than the cross-level Z-curve. Significantly, the better the continuity of space filling curve, the better the clustering property of code (Moon et al., 2001), and the more efficient the efficiency of computation method based on GMGICM codes.

In addition, the multi-scale grid coding method proposed in this paper theoretically supports the grid system based on various tree shapes. Considering the efficiency of coding conversion and calculation, and the applicability to EO data, this paper adopted the quad-tree based on the division of latitude and longitude space, not a more complex polyhedral grid.

3. Applying GMGICM on big earth observation data indexing and querying

In this study, big earth observation data management is mainly developing effective association methods between EO data and grids, and efficient indexing and querying for big EO data, on this basis of GMGICM. Referring to the typical EO data storage and management system (Lv et al., 2011), we adopt the mixed management method of file storage and database system (DBS), i.e., the EO data file is stored in the file storage system, in which the file can be correlated with its metadata by its path information, and the metadata of EO data, used for indexing and querying, is organized in the DBS.

At present, the development of file storage systems is very mature, such as distributed storage system (Chang et al., 2008), clustered storage system (Jy-Yong et al., 2017), etc., which is not elaborated in this manuscript, and developing new ones may not be urgent, compared with the problems and challenges for EO data management mentioned in the introduction. Therefore, the paper is mainly using the metadata to build spatial index, so as to organize, index and query EO data. The main procedure of above idea is presented in Fig. 6, including three fundamental steps: association between EO data and grids, spatial indexing of EO data, and spatial querying of EO data.

Association between EO data and grids is using grid codes to describe the spatial position information of images, which helps to identify massive multi-source data uniformly, and only adopts a finite number of grids to associate massive amount of images, the amount of which is very possible to rapidly and continuously increase further.

Based on the association results computed by the first step, the spatial index of metadata can be built with simple one-dimensional tree index, which reduces the dimension of traditional spatial index, and is beneficial to improve the efficiency of spatial indexing.

Using the spatial index, combined with our proposed computations of GMGICM codes, spatial querying can be accomplished, which can assure the integrity of query results, in addition to a high efficiency and accuracy.

3.1. Association method between EO data and grids

The method of association between EO data and grids is essential for spatial indexing method as well as EO data management. The key idea is to associate the EO data (image) with a small amount of grids whose coverage is the same or close to that of the EO data such as satellite images. In this way, the small amount of associated girds would help to improve the efficiency of spatial indexing of EO data. If the associated grids are closer to the coverage space of images, i.e., the spatial redundancy of associated grids is lower, the results of spatial querying can get more accurate. In this case, however, more associated grids might be needed. Therefore, there is a balance between the amount of associated grids and querying accuracy, for which the study will present an approach for the association between images and grids.

Since the scenes of multi-source EO data have different scales, positions and coverages, and GMGICM adopts the static grid framework, there is a case where the coverage space of EO data and its associated grids could not be completely overlap. The relationship between the

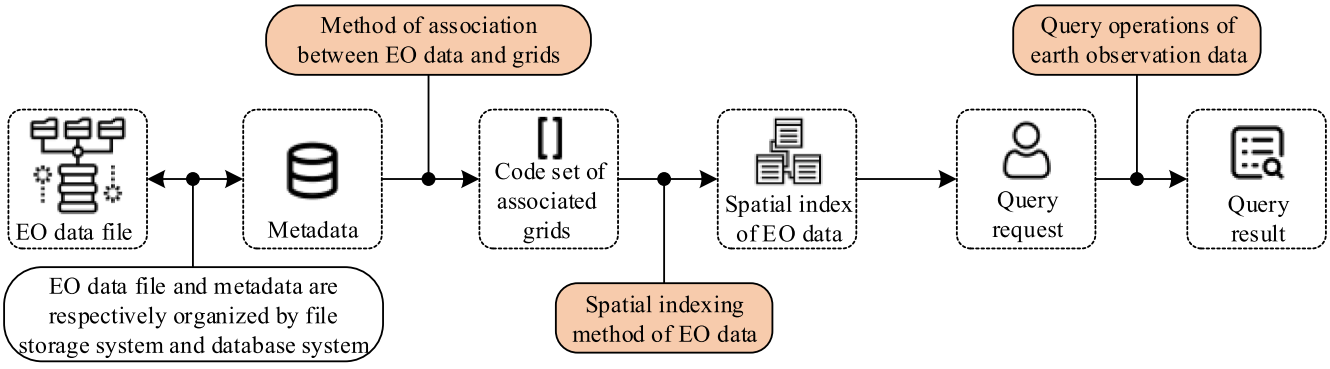

Fig. 6. Main procedure of organization, indexing and querying of EO data.

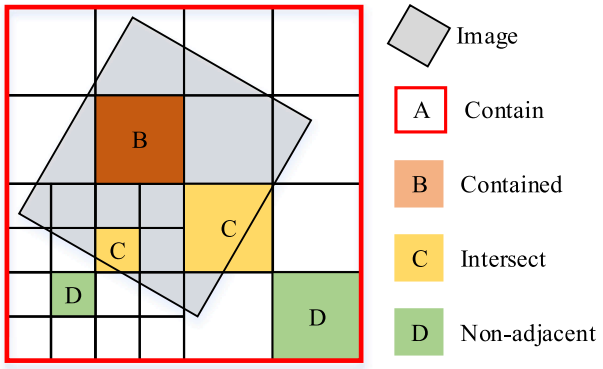

Fig. 7. Several common spatial correlations between multi-scale grids and EO data (image).

coverage space of the image and multi-scale grids can be divided into the following four cases: contain, contained, intersect, and nonadjacent, which is illustrated in Fig. 7

According to the above analysis, the spatial relationship between grids and the coverage space of EO data is checked from top to bottom level of GMGICM codes, to compute the associated grids. The codes of the associated grids should satisfy three restrictive conditions: the min level of codes is not smaller than N_{min}; the max level of codes is not bigger than N_{max}; the amount of codes is not more than S_{max}. Considering the characteristics of EO data, as well as the balance of amount and spatial redundancy of associated grids, the following strategies are adopted for above restrictive conditions. The value of N_{min} is given by formula (4), the value of N_{max} is 31, and the value of S_{max} can be determined through experiments.

$$N_{min}=\min\{31,\lceil \log_2(\frac{360}{Mid})\rceil \} \tag{4}$$

where *Mid* is the median value of image size, whose unit is degree of latitude or longitude, $\lceil\rceil$ represents rounding up the value inside. To describe the procedure of our association method, the following symbols will be used:

- $Grid_I$, a type of grids including the gird-A and grid-C shown in Fig. 7, level of which is smaller than N_{max}.
- $Grid_{II}$, a type of grids including the grid-B shown in Fig. 7 and the grid whose level $\geq N_{max}$.
- $Grid_{III}$, a type of grids including the grid-D shown in Fig. 7.
- *Region*, the vector region of coverage space of data.
- *Amount*, the total amount of $Grid_I$ and $Grid_{II}$.
- *Result*, the result of associated girds.
- *Queue*, the queue consisted of $Grid_I$ in the order of the priority, which can be determined using three rules:

(I) The smaller the level of the grid in the queue is, the higher the priority is.
(II) The more son-grids that belong to $Grid_{III}$ the grid has, the higher the priority is.
(III) The less son-grids that belong to $Grid_{II}$ the grid has, the higher the priority is.

According to the rules, the priority is given as $p = -(16N_G + 4Num_1 + Num_2)$, where N_G is the level of the grid, Num_1 is the amount of four son-grids that belong to $Grid_I$, Num_2 is the amount of four son-grids that belong to $Grid_{II}$.

Based on the above analysis, the main procedure of association method is as follows:

(1) Use grids of level N_{min} to discrete the *Region*, then divide those grids into $Grid_I$, $Grid_{II}$, and $Grid_{III}$ set with the method of polygons intersection judgement.
(2) If $Amount \geq S_{max}$ or the amount of $Grid_{II}$ is zero, the procedure is done and *Result* could be output, which is consisted of $Grid_I$ and $Grid_{II}$ set. Otherwise, turn to step (3) and push $Grid_I$ into *Queue*.
(3) Calculate the four son-grids of the grid with highest priority in *Queue*, then divide those son-grids into $Grid_I$, $Grid_{II}$, and $Grid_{III}$ set, meanwhile, update the sequence of *Queue* according to the priority of each $Grid_I$.
(4) Repeat step (2) to (3) until *Result* matches the three restrictive conditions.

Besides, the method of association between EO data and grids is also suitable for the association between polygons and grids, when the coverage space of polygons is taken as *Region*.

3.2. Spatial indexing method of EO data

Commercial database system (such as Oracle, MongoDB, etc.) is used to manage the metadata, which is the generalization and abstraction of time, location and storage path of the EO data (Hu et al., 2014). Therefore, by building the index of metadata, the related EO data could be indexed, which avoids the direct operation of the original data with huge data volume, and is beneficial to improve the efficiency of organization, management and retrieval of EO data. In addition, if the existing data management system is upgraded using this method, the management system only needs to add a few data tables, which keeps the upgrade cost low, and avoids the operation of the original data, thereby ensuring the security of the data.

Before building the spatial index of EO data, it is necessary to unify the metadata format, which is conducive to the unified storage and management of metadata, because the metadata formats of EO data of different sources are different. Based on the geographic information metadata part 2: extensions for imagery and gridded data (ISO 19115–2:2009), considering the characteristics of EO data, this study presents a kind of metadata reference format with the key fields shown in Table 1.

On the basis of the existing standard, the metadata format adds two fields. One is *ImageID*, an integer used for uniquely identifying image data. It needs to be mentioned that the retrieval efficiency of this integer type identifier is higher than that of string type identifiers. The other is *GridCodeSet*, which is calculated by the method of association between EO data and grids, and used to store of the resulting associated grids identifying spatial position of EO data.

This study mainly uses spatial position information in the metadata to build the index. Therefore, only the spatial–temporal information and path field of EO data are listed in the reference format, and other

Table 1
Reference format for EO metadata.

Field name	Field meaning	Data type
ImageID	Integer identification	NUMERIC(38,0)
GridCodeSet	Associated grid code set	NUMERIC(20,0)[]
ImageName	Image file name	VARCHAR(2 5 0)
ImagePath	File path	JSON
GetTime	Get Time of data	TimeStamp
RecordTime	Record time of data	TimeStamp
CenterLat	Center point latitude	NUMERIC(16,10)
CenterLon	Center point longitude	NUMERIC(16,10)
LeftTopLat	Upper left point latitude	NUMERIC(16,10)
LeftTopLon	Left upper point longitude	NUMERIC(16,10)
RightTopLat	Upper right point latitude	NUMERIC(16,10)
RightTopLon	Right upper point longitude	NUMERIC(16,10)
LeftBottomLat	Left lower point latitude	NUMERIC(16,10)
LeftBottomLon	Left lower point longitude	NUMERIC(16,10)
RightBottomLat	Right lower point latitude	NUMERIC(16,10)
RightBottomLon	Right lower point longitude	NUMERIC(16,10)
…	Other needed for users	…

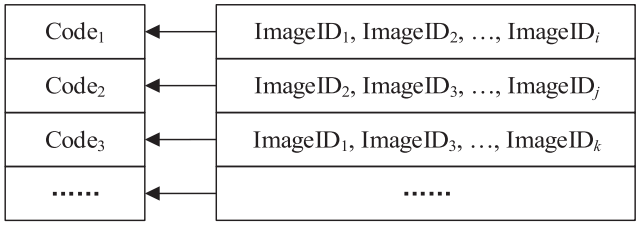

Fig. 8. Spatial correlation between grid codes and integer IDs of EO data.

fields can be selected according to the application requirements.

The steps for building a spatial index are as follows:

(1) According to the above metadata format, a metadata table should be created in the database, and the metadata of EO data can be imported into the corresponding field, and then a unified metadata table can be formed, which would be used by the database system, realizing the unified storage and management of metadata.
(2) The method of association between EO data and grids is used to calculate *GridCodeSet* field in the metadata table, thereby building the spatial association between the EO data and GMGICM codes.
(3) Based on *GridCodeSet* field, combined with one-dimensional index (such as B-tree (Bayer, 1997), B+ -tree (Zhang et al., 2009)), the spatial index for EO data could be built, which is stored in the gird code index table consisting of two fields, GMGICM codes and ID set of EO data. The mapping relationship of GMGICM codes and IDs of EO data is shown in the Fig. 8. In this mapping relationship, each GMGICM code is related to a set of IDs, which is stored in a binary format to ensure high efficiency of the combination and splitting of IDs. This can reduce the impact of one-to-many relationship between grids and IDs on query efficiency.

3.3. Spatial querying method of EO data

Based on the spatial indexing method of EO data, the spatial querying method proposed in this section transforms complex two-dimensional spatial query into a simple and efficient one-dimensional integer coded query, which is beneficial to improve the spatial query efficiency of EO data. There are four kinds of spatial relationships between images and querying areas: the two do not intersect or contain each other, the two intersect, the query area contains images, and the image contains the query area.

To overcome the problem of querying incompleteness, the implementation of the spatial query proposed in this study is shown in Fig. 9. A total of two queries for the index table are performed. The first query is mainly for the intersection of the query area and EO data and cases where the query area contains data, the second query is mainly for the case where the image contains the query area, in order not to omit the result for complete querying. The main implementation steps of the query mechanism are as follows:

(1) Input the query area boundary. The user inputs the boundary vector coordinate data of the query area, the query area may be an arbitrary polygon, and the coordinate data type is latitude and longitude coordinates, and the coordinate system is the same as the coordinate system of the EO metadata.
(2) Associate the area with grid code set S_1. According to the coordinate data input by step (1), the first grid coding result S_1 of the query region is calculated by using the method of association between EO data and grids, and the restrictive conditions are same as section 3.1, but there is no limit for S_{max}.
(3) Use S_1 to query in the index table. Using son-grid codes calculation method, the i-th code subinterval range of S_1, corresponding to $S_1(i)$, is obtained, then the code set $O_1(i)$ belonging to the subinterval range is searched in the grid code index table. Repeat the above operation until traversing S_1 to get the first query result O_1.
(4) Convert S_1 to S_2 in which level of all codes is N_{min}. Using father-grid code calculation method, all father codes of S_1 could be calculated, the level of these father codes is N_{min}. Then delete the same values in these father codes, S_2 could be got.
(5) Use S_2 to query in the index table. O_2^{temp} is calculated by traversing S_2, the method of which is same as step (3). Then use son-grid codes and father-grid code calculation method to judge whether the code in O_2^{temp} contains a son-grid code belonging to S_1, O_2 in which codes meet the judgement would be calculated.
(6) Get the ID set of images containing and contained by the area. According to codes in O_1 and O_2, the binary data corresponding to the ID set of EO data is extracted in the grid code index table, and the multi-threading technology is used to accelerate the splitting of the binary data and the deletion of the *ImageID* duplicate value, and finally get the ID set that meets the query condition.
(7) Output query results, which consist of the ID set that meet the query condition.

4. Experimental results

To comprehensively validate the proposed GMGICM, we carried out a series of experiments with both simulated and real data. Specifically, we carried out three types of experiments, including 1) clustering property analysis of GMGICM codes by comparing the continuity of our proposed cross-level Z-curve to a baseline approach, level-by-level Z-curve, 2) multi-scale code computation efficiency analysis by comparing the proposed GMGICM codes to Geohash codes, and 3) spatial indexing and querying of real and simulated EO data based on GMGICM.

4.1. Clustering property analysis of GMGICM codes

In order to compare the continuity of the cross-level Z-curve (Z_C-curve), which is developed in our proposed GMGICM, and the traditional level-by-level Z-curve (Z_S-curve), we simulated both curves with levels of 3, 4, 5, 6, 7, 8, and 9. For analysis of clustering property, two approaches were employed (Zhai et al., 2018). The first is from spatial objects to codes, where curve codes are checked to see whether they are adjacent if the corresponding spatial objects are adjacent in scope for queries. The other is from codes to spatial objects, where spatial objects are checked to see whether they are adjacent in scope for queries if the corresponding curve codes are continuous.

The experimental environment is: Windows7 64bit, Intel(R) Core

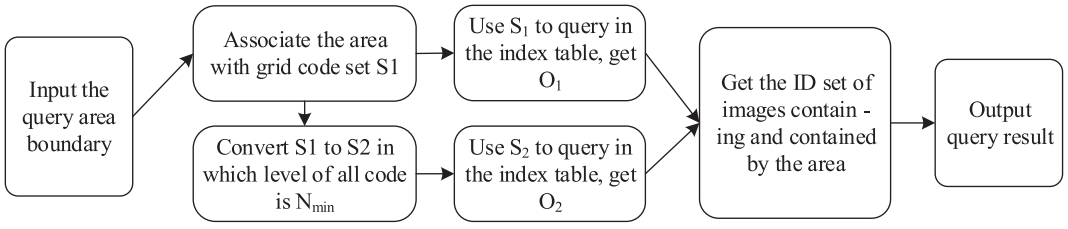

Fig. 9. Spatial querying process of EO data based on its code index table.

(TM) i7-5700HQ CPU @ 2.70 GHz, RAM 8 GB, SSD 128 GB, Visual Studio 2013, C++.

4.1.1. From spatial objects to codes

Given a grid, we count the total number of neighboring grids that are also with adjacent codes. A larger number means a better spatial continuity. The number can be calculated by:

$$Count_1 = \sum_{i=1}^{N} \sum_{d=\tau \cdot d_{min}}^{\tau \cdot d_{max}} \sum_{j=M_i-d}^{M_i+d} f_{near}(M_i, M_j) \quad (5)$$

$$f_{near}(M_i, M_j) = F_{near}(G_i, G_j) = \begin{cases} 1, adjacent. \\ 0, notadjacent. \end{cases} \quad (6)$$

where G is the set of grids, M is the corresponding set of codes, $f_{near}(M_i, M_j)$ is 1, 0 when G_i and G_j are adjacent, not adjacent, respectively. d denotes the distance threshold of codes, the chosen maximum and minimum value of which, d_{max}, d_{min}, are 20 and 1, respectively in this study, and τ is the minimum difference value of codes in the highest level. Specially, τ is 1 for Geohash codes, and 2 for GMGICM codes.

Two kinds of neighborhood, with 9 and 33-neighborhood, were being considered in this study. The 9-neighborhood includes one father-grid, four neighbor grids within the same level, and four son-grids. The 33–neighborhood includes a father-grid and its four neighbor grids within the father-grid level, four neighbor grids within the same level, and four son-grids and the 12 peripheral grids.

Based on the above approach, we compared $Count_{1_c}$ and $Count_{1_s}$ resulting from Z_C-curve and Z_S-curve coding, with the subtraction results ($Count_{1_c} - Count_{1_s}$) presented in Table 2 For better interpretation, we also showed the increase ratio defined as $100\% \times (Count_{1_c} - Count_{1_s})/Count_{1_s}$.

The experimental results in Table 2 demonstrate the increased continuity of Z_C-curve compared to Z_S-curve. Specifically, the superiority of Z_C-curve is more obvious for the 33-neighborhood than the 9-neighborhood.

4.1.2. From codes to spatial objects

By applying a sliding window on the gird in the highest level, we can count the total number of contiguous code blocks in all the codes (from all levels) that intersect with the sliding window, which is

$$Count_2 = \sum_{w_i} IntervalNum(w_i) \quad (7)$$

Take Fig. 5(a) as an example, with a 2 × 2 sliding window covering grid 11、12、17、18 on level 2, the girds that intersect with the sliding window include grid 0 from level 1, grid 2 and 4 from level 1, and 11、12、17、18 from level 2. Finally, there are five contiguous code blocks from these intersected grids: 0, 2, 4, 11 ~ 12, 17 ~ 18. Namely, $Count_2 = 5$.

In this study, we considered two sizes of sliding window, 2 × 2 and 3 × 3. Using the above approach, $Count_{2_c}$and $Count_{2_s}$, resulting from Z_C-curve and Z_S-curve coding are presented in Table 3, where we also list the decreased ration defined as$100\% \times (Count_{2_s} - Count_{2_c})/Count_{2_s}$.

Table 2
Comparing study of Z_C-curve and Z_S-curve: spatial objects to codes.

Total level	Difference of 9-neighborhood	Increase ratio of 9-neighborhood (%)	Difference of 33-neighborhood	Increase ratio of 33-neighborhood (%)
3	224	12.669	414	12.140
4	2086	28.280	5640	36.675
5	9366	31.744	25,938	42.077
6	38,486	32.610	106,998	43.393
7	154,966	32.827	431,238	43.722
8	620,886	32.881	1,728,198	43.804
9	2,484,566	32.895	6,916,038	43.825

Table 3
Comparing study of Z-Cross and Z-Step curve: codes to spatial objects.

Total level	2 × 2 window			3 × 3 window		
	Z_C-curve	Z_S-curve	Decrease ratio (%)	Z_C-curve	Z_S-curve	Decrease ratio (%)
3	25	34	26.470	18	19	5.263
4	219	278	21.223	228	292	21.917
5	1327	1618	17.985	1544	1980	22.020
6	6883	8166	15.711	8320	10,492	20.701
7	33,031	38,410	14.004	40,288	49,900	19.262
8	151,883	173,902	12.661	184,592	224,956	17.943
9	680,911	770,002	11.570	820,320	985,676	16.775

From Table 3, we can see that Z_C-curve coding is with better spatial continuity compared to Z_S-curve coding, and its superiority gets less obvious in higher level. Additionally, in higher level, the superiority is more obvious with large sliding window than with small sliding window

4.2. Comparison of son-grids querying between GMGICM and Geohash

To analyze and validate the efficiency of GMGICM in multi-scale grid code computations, comparison of son-grid querying between GMGICM and Geohash was carried out. Specially, compared to operation of neighbor grid codes or father-grid code, the operation of son-grid codes can better demonstrate the efficiency of two coding methods for multi-scale grid code computations. The experimental environment is the same as Section 4.1.

The main procedure of son-grid querying in this experiment is: given a multi-scale grid set, and a query grid, we can use the code computation method to find son-grids belonging to the query grid in the set, followed by calculating its corresponding code value. The main steps of the comparison experiment are as follows:

(1) A multi-scale grid set including n grids would be generated, the level of each grid is between 1 and 31. Then, convert the set to GMGICM code set, Set_1, and Geohash code set, Set_2, respectively.
(2) We randomly created one hundred multi-scale grids as query grids, the level of each grid is between 1 and 31.
(3) Based on computations of GMGICM codes, the son-grid code value intervals of each query grid in all levels would be calculated. Then, use dichotomy to search out the code belonging to the above intervals in Set_1. Meanwhile, the time of this step would be recorded for the comparison.
(4) Based on computation of Geohash, the time of son-grid querying would be got by adopting the same method as step (3).

We tested five different n, including 500 000、1 000 000、2 000 000、5 000 000、10 000 000, each of which is tested three times repeatedly. The averaged results over the three tests are presented in Table 4, where the ratio is between the time used by Geohash and GMGICM.

As shown in Table 4, there is a positive correlation between the required time and the number of codes n, for both kinds of codes. Furthermore, the increase ratio gets larger with increasing n. On

Table 4
Efficiency comparison of calculating son grids between GMGICM and Geohash.

n	GMGICM (s)	Geohash (s)	Ratio
500 000	1.52	28.8	18.9
1 000 000	3.44	72.32	21.0
2 000 000	8.72	196.64	22.6
5 000 000	24.32	572.64	23.5
10 000 000	49.36	1187.86	24.1

average, the GMGICM approach is 20 times as fast as Geohash.

4.3. Applications on spatial indexing and querying of big EO data

We based our experiments on Oracle database, this kind of relational database is commonly used in EO data management center (Wang, 2009; Li et al., 2016). Similar to the current cases, for the validation of the proposed coding model, we also manage multi-scale multi-source big EO data via the *meta*-data for spatial indexing and querying. To demonstrate the effectiveness, we compared the results from our approach and those from Oracle Spatial regarding to the querying accuracy and efficiency. Our approach use a B-tree index based on GMGICM Code, and R-tree index is adopted for Oracle Spatial.

Oracle Spatial has two types of indexes, quadtree and R-tree indexes (Kothuri et al., 2002). Compared with R-tree index, quadtree index is more suitable for indexing point objects. Considering that EO data has complex spatial relationships and different scales, and R-tree index can use minimum bounding rectangle to match each spatial object, so we choose R-tree index for Oracle Spatial.

The experimental environment is: Windows7 64bit, Intel(R)Xeon(R) X5650 CPU @2.67 GHz, RAM 16 GB, HDD 1 TB, Visual Studio 2013, C++, Oracle 11 g.

4.3.1. Data and experimental setup

We first retrieved two million scenes of EO metadata from websites of China Centre for Resources Satellite Data and Application (www.cresda.com), and Application and United States Geological Survey (https://lpdaac.usgs.gov/products), and the metadata is converted and stored in Oracle database according to the reference format shown in Table 1. The data is distributed across the globe (including polar regions) and includes several types of EO data such as multi-spectral, hyperspectral, radar, true/false color images. The corresponding images are of different resolutions and sizes, with the length ranging from 0° to 11° (the median is 0.389°) in order to comprehensively validate the querying effectiveness of the proposed approach. Specifically, 96.0%, 2.8%, and 1.2% of the images are with the length ranging from 0° to 1°, from 1° to 2°, and 2° to 11°, respectively. The detailed distribution of images with respect to the size (length) are presented in Fig. 10.

In addition to the real data, we also simulated 20 million scenes of EO metadata based on the 2 million scenes of the real data. Similar to the real data, simulated data is of different size, multi-source, and distributed globally. The format of the simulated data is presented in Table 1, and simulated data is also stored in Oracle database.

4.3.2. Parameter Choice: Number of grids to be associated to EO data

The choice of parameters for associating EO data with grids is of importance as it affects the description precision of the spatial location of the EO data, thus affecting the efficiency of indexing and spatial querying as well as the accuracy of spatial querying. To investigate this effect of the maximum number of grids associated with EO data, S_{max}, we designed the following experiment with two million scenes of real data:

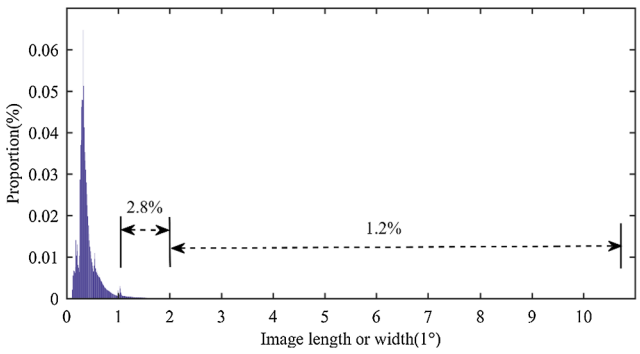

Fig. 10. Distribution of image width or length in the interval of [0, 11] degree.

(1) Building spatial index of EO metadata. We built spatial index for 2 million files of EO metadata, using both Oracle Spatial and GMGICM Code. GMGICM Code is also based on Oracle to be consistent regarding the test platform. For GMGICM Code, B-tree index is chosen as an example as it can directly use all kinds of one-dimensional index. For Oracle Spatial, the extension of B-tree index in higher dimension space, R-tree index is built.
(2) Spatial querying. For the querying areas, we randomly chose six non-overlap rectangles and polygons across the world, and the administrative area of twelve Chinese provinces (Beijing, Taiwan, Zhejiang, Shanxi, Shandong, Jiangxi, Henan, Sichuan, Hunan, Guangdong, Yunnan, Hainan), for which the examples of boundary shapes are shown in the Fig. 11. For each of these querying areas, spatial querying is carried using Oracle combined with GMGICM Code, and the querying results are *ImageID* set (as introduced in Table 1. We recorded the time used as querying efficiency, calculated the querying accuracy and omission error, which are then averaged over all 12 querying areas.

The querying efficiency (whose unit is second) is defined as the period from inputting the querying rules to outputting the querying results. The querying accuracy rate is $Num_{correct}/Num_{ouput} \times 100\%$, where $Num_{correct}$ is the number of image scenes (in querying results) that are correctly contained, intersected with, or contain the query area, Num_{ouput} is the number of all image scenes in querying results. The querying omission error is $Num_{omission}/Num_{all} \times 100\%$, where $Num_{omission}$ is the number of image scenes that should be in querying results but are omitted, Num_{all} is the number of all image scenes that should be correctly queried. Num_{all} is approximated by the number of querying results from Oracle Spatial, because using the primary filter and secondary filter (Hu et al., 2012) of Oracle Spatial, we can get a superset of correct querying results with no omission and high accuracy (more than 99.0%). In this way, we can detect the omitted scenes that exist in querying results from Oracle Spatial but not in the querying results from Oracle combined with GMGICM Code.

(3) Following the two steps described above, we repeated the spatial querying experiments six times withS_{max}= 4, 10, 15, 20, 25, 30, respectively. In each experiment, N_{min} and N_{max} are set as 10 and 31, respectively, according to the formula (4). The indexing time, average efficiency, accuracy rate and omission error of query in each experiment are presented in Table 5, in which the average efficiency, accuracy rate and omission error is the average values of the efficiency, accuracy rate and omission error for all querying areas in above experiment.

In addition, we compared the results in Table 5 with Oracle Spatial. First, based on Oracle Spatial, the above three processes were carried out, and the its indexing time, average efficiency, accuracy rate and omission error were recorded as 230.7 s, 14.9 s, 99.8%, and 0.0%, respectively. Then the ratio (including averaged accuracy ratio, indexing time ratio and averaged efficiency ratio) of results from the two approaches is visualized in Fig. 12.

From Table 5 and Fig. 12, we can see that it takes longer time for index building with the increase of N_{max} when using the GMGICM Code. It can take longer time than Oracle Spatial, as can be seen from Fig. 12. With the increase of N_{max}, querying efficiency becomes lower but still much higher than that of Oracle Spatial. With the increase of N_{max}, querying accuracy becomes higher with a decreasing speed. Omission error is zero, meaning that all results are queried without missing. Based on the comparisons, we determined a reasonable choice ofN_{max} ranges from 20 to 25, with which we can achieve similar index building efficiency, and much higher querying efficiency and querying accuracy.

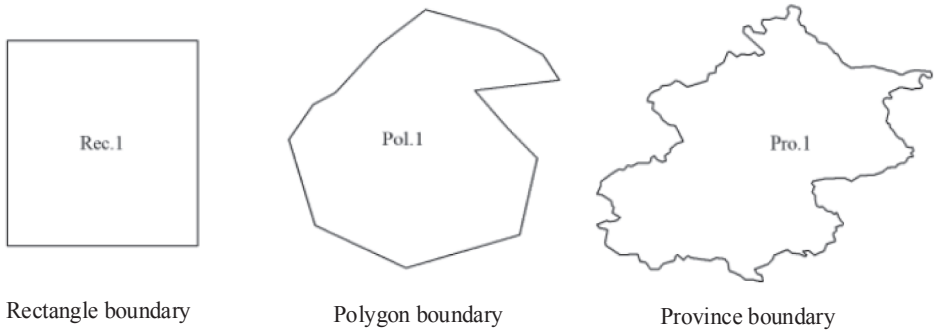

Fig. 11. Boundary shape examples of rectangle, polygon and province used as querying areas.

Table 5
Indexing time, and average querying efficiency, accuracy rate, omission error corresponding to different numbers of associated girds when using Oracle with GMGICM Code.

S_{max}	Indexing time (s)	Average querying efficiency(s)	Average accuracy rate (%)	Average omission error (%)
4	68.3	0.162	86.6	0.0
10	89.8	0.181	89.2	0.0
15	118.7	0.225	91.3	0.0
20	180.8	0.263	93.9	0.0
25	263.2	0.389	94.2	0.0
30	395.1	0.523	94.5	0.0

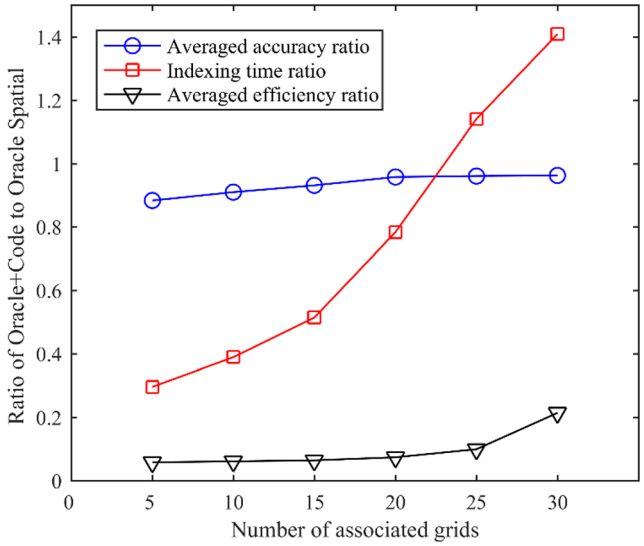

Fig. 12. Indexing time, averaged querying efficiency and accuracy comparison: ratio between Oracle with GMGICM Code and Oracle Spatial with respect to different numbers of associated girds.

4.3.3. Experimental results

To validate the applicable potential of the proposed GMGICM on the spatial indexing and querying of big earth observation data, based on 20 million of simulated data, we designed an experiment with the following steps:

(1) For the 20 million simulate data files, we built index using both Oracle with GMGICM Code and Oracle Spatial. For GMGICM Code, the parameters used are $N_{max} = 31$、$N_{min} = 10$、$S_{max} = 20$.
(2) Using the six rectangle, polygon, and twelve administrative areas (the same as Section 4.3.2) as query areas, spatial querying was carried out using two kinds of index from GMGICM Code and Oracle Spatial, respectively. We recorded the respective querying efficiency, accuracy rate and omission error for different types of querying areas.

Table 6
Efficiency, accuracy rate and omission error of big EO data querying with rectangle, polygon and administrative area boundary.

Area	Method	Average efficiency (s)	Average accuaracy rate (%)	Average omission error (%)
Rectangles	Oracle + Code	3.293	96.6	0.0
	Oracle Spatial	23.264	99.9	0.0
Pologons	Oracle + Code	3.448	96.1	0.0
	Oracle Spatial	32.173	99.8	0.0
Provinces	Oracle + Code	3.768	91.1	0.0
	Oracle Spatial	86.340	99.6	0.0

The comparing results are as presented in Table 6 and Fig. 13, which shows that the querying efficiency is much higher when using Oracle with GMGICM Code (Oracle + Code), as the querying using Oracle Spatial takes 7.1, 9.3, and 22.9 times more time. The querying accuracy ratios of two approaches are close, only 5.1% difference on average, and the omission errors are both zero.

5. Discussions

In the section, we will provide some explanations and analyzes to the posed problems in Section 1, based on the insights gained from the extensive experimental results presented in Section 4. Beyond that, this section will also discuss some further possible improvements towards a more effective and robust solution for big EO data management.

To demonstrate the good clustering property of GMGICM codes, two experiments were conducted in Section 4.1. Form Table 2, we can see that the clustering extent (measured by increased ratio) of Z_C-curve is higher than that of Z_S-curve, and it becomes larger when the level gets higher and converges to a certain value. This is because in the lower level, most of the grids are at the border position of the quadtree, which leads to no big difference between these two kinds of curves. In higher levels of Z_S-curve coding, however, the code difference between one grid and its four child-grids gets bigger, which is the same with Z_C-curve coding but with a slower increasing speed since the grid code is the average value of codes of its four child-grids. Table 3 also shows the obvious advantage of our developed Z_C-curve used in GMGICM, with better spatial continuity. This is because the unavoidable discontinuity of Z_S-curve resulting from cross-level jumping, i.e., the code based on Z_S-curve does not have clustering property in the scale dimension. The Z_C-curve adopted by GMGICM, by contrast, has good clustering property in both scale and spatial dimension. The clustering property could provide a good solution to the first challenge in Section 1, enabling higher operation efficiency. Moreover, the idea of cross-level Z-curve can be theoretically applied to Hilbert or Gray curve, to improve its clustering property.

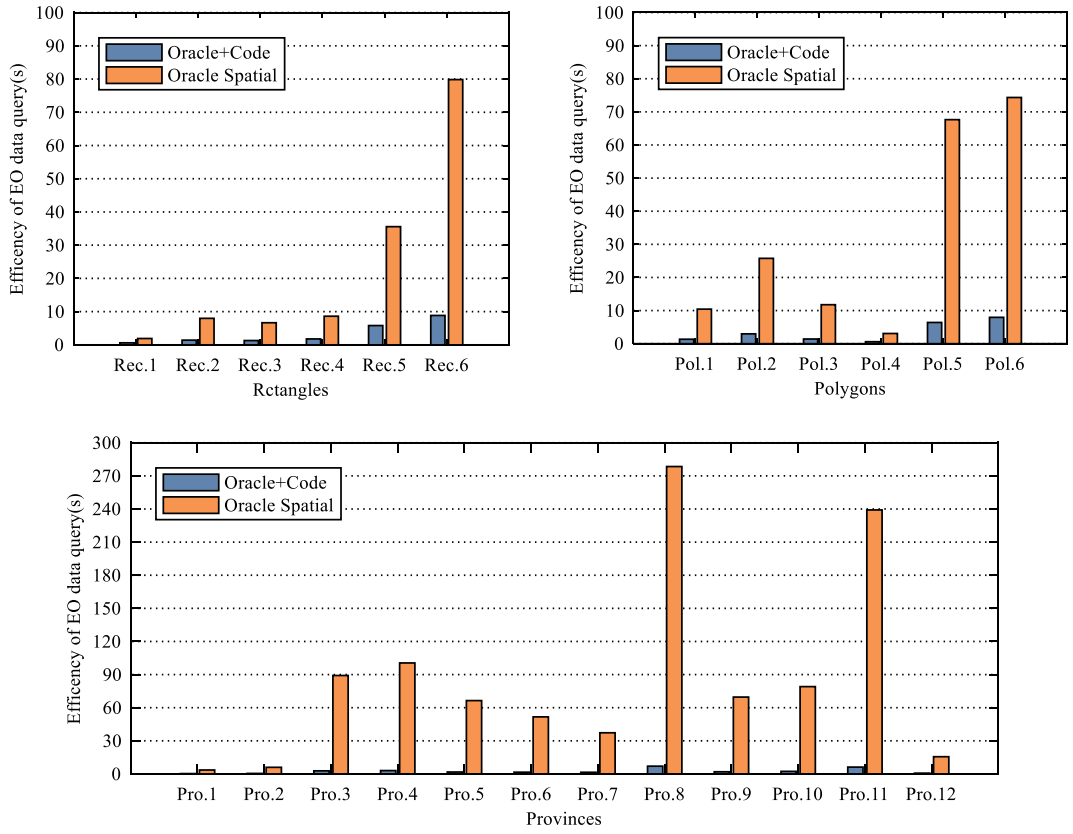

Fig. 13. Efficiency comparison of big EO data querying with rectangle, polygon and province boundary.

By comparing the operation efficiency of GMGICM codes to Geohash codes, it gets clear that the proposed GMGICM codes can be operated with a 20 times faster speed, as shown in Table 4. Furthermore, this superiority gets more obvious with increasing number of operations. This is thanks to two main reasons. The first is the integer code, a 64-bit unsigned integer as described in Section 2.3, whose calculations mainly use bit operation thus having high efficiency. The other reason is the clustering property in both spatial and scale dimension, which has been shown and analyzed by previous experiments. This will bring advantages in many applications such as multi-scale code computation and grid querying, which can be beneficial for applications such as spatial querying method of EO data (Section 3.3).

The experiments in Section 4.3 show that the proposed GMGICM can be successfully applied to the efficient management of big EO data distributed across the globe and that the inversed quad-tree adopted by GMGICM overcomes one of the challenges mentioned in Section 1 (linear tree-based indexing does not support data with complex spatial relationships well). In addition, the choice of three parameters (especially for the number of grids), used in the method of association between EO data and grids, were investigated through experiments. With reasonable number of associated grids, the indexing efficiency of EO data can be high while redundancy of the correlated grid regions can be much reduced, leading to high accuracy of grid querying.

Furthermore, from applications on big EO data and results, we can also see that the querying efficiency is much higher when using the proposed method compared to Oracle Spatial. Additionally, it is stable and not susceptible to the shape and size of querying areas. For Oracle Spatial, by contrast, the more complex the shape of the querying region, the lower the querying efficiency will get. This is because Oracle Spatial is mainly based on vector and floating-point calculations, which is much more un-efficient than grid and integer calculations. It needs to mention that the querying accuracy is a little lower when using the proposed method than Oracle Spatial. This is due to the precision loss when converting vector or region to grids, which is unavoidable and only can be reduced as low as possible. If a higher accuracy is needed, we can conduct second querying through filtering the redundancy in the querying results. Lastly, there is no omission of the querying in the proposed method, which proves that the strategy to assure the querying completeness is effective.

6. Conclusions and outlook

This paper proposed GMCICM and explored the spatial association, indexing and querying methods applicable to the organization, management and spatial retrieval of multi-source, multi-temporal and multi-scale EO data. Through theoretical analysis and experiments, the following conclusions can be drawn: (1) GMGICM codes show good clustering property in both spatial and scale dimensions, which is more advantageous than traditional methods because of improved spatial retrieval efficiency for grid system; (2) compared with commonly used Geohash, the efficiency of son-grids querying based on GMGICM is improved by about 20 times, providing important technical support for efficient grid query; (3) the method of association between EO data and grids is proved to be effective with the average accuracy of query as high as 94.6%; (4) the spatial query efficiency is about 10 times higher when using GMGICM-based Oracle than Oracle Spatial, and there is no omitted querying results from the proposed approach.

This paper provides significant solutions for existing challenges for big EO data management, and is supportive for effective and accurate services related to EO data such as data sharing. It needs to be mentioned that methods in this study have been successfully applied in one satellite data center in China, whose management system has been upgraded with this proposed low-cost, feasible and efficient solution. Future work includes further improving the querying accuracy and extending this work for spatial–temporal big data by exploiting spatial–temporal index mechanism.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by National Key R&D Program of China [Grant Number 2018YFB0505304]; National Natural Science Foundation of China [Grant Number 41671409].

References

Klein, L.J., Marianno, F.J., Albrecht, C.M., et al., 2015. PAIRS: a scalable geo-spatial data analytics platform. In: IEEE International Conference on Big Data. IEEE Computer Society, 2015.
Waterfall, A., Bennett, V., Donegan, S., et al., 2016. Big data challenges indexing large-volume, heterogeneous EO datasets for effective data discovery. Living Planet Symposium.
Câmara, Gilberto, Assis, L.F., Queiroz, G., et al., 2016. Big earth observation data analytics: matching requirements to system architectures. 5th ACM SIGSPATIAL International Work-shop on Analytics for Big Geospatial Data. ACM.
Deren, L., Mi, W., Xin, S., et al., 2017. From earth observation satellite to earth observation brain. Geomatics & Information Science of Wuhan University 42 (2), 143–149.
Qi, K., Hu, Y., Li, S., et al., 2017. An improved method for the unique code of spatial entity based on global subdivision grid. Geoscience & Remote Sensing Symposium. IEEE.
Cian, F., Marconcini, M., Ceccato, P., 2018. Normalized difference flood index for rapid flood mapping: taking advantage of EO big data. Remote Sens. Environ. 209, 712–730.
Maex, E., 2007. Earth observing system (EOS) data and information system (EOSDIS) — evolution update and future. IEEE International Geoscience & Remote Sensing Symposium.
Wang, F., 2009. Design and implementation of ground data storage and management system for earth resources satellite. Spacecraft Engineering.
Gibin, M., Singleton, A., Milton, R., et al., 2008. An exploratory cartographic visualization of London through the google maps API. Applied Spatial Analysis & Policy 1 (2), 85–97.
Lu, N., Cheng, C., Ma, H., et al., 2012. Global discrete grid systems analysis and comparison. In: Geoscience and Remote Sensing Symposium (IGARSS), 2012 IEEE International. IEEE, 2012.
Yin, M., Liang, X.Y., Duan, P.H., et al., 2017. Image super-resolution reconstruction method based on pulse coupled neural network-sparse coding and nonsubsampled pyramid transform. Journal of Optoelectronics·laser 28 (8), 918–925.
Tong, X., Ben, J., 2016. The Principles and Methods of Discrete Global Grid Systems for Geospatial Information Subdivision Organization. Surveying and Mapping Press, Beijing.
Raposo, P., Robinson, A.C., Brown, R., 2019. A virtual blobe using a discrete global grid system to illustrate the modifiable areal unit problem. Cartographica The International Journal for Geographic Information and Geovisualization 54 (1), 51–62.
Song, S., Cheng, C., Pu, G., et al., 2014. Global Remote Sensing Data Subdivision Organization Based on GeoSOT. Acta Geodaetica et Cartographica Sinica 43 (8), 869–876.
Malensek, M., Pallickara, S., Pallickara, S., 2017. Fast, Ad Hoc Query Evaluations over Multidimensional Geospatial Datasets. IEEE Trans. Cloud Comput. 5 (1), 28–42.
Li, S., Cheng, C., Tong, X., et al., 2016. A Study on Data Storage and Management for Massive Remote Sensing Data Based on Multi-level Grid Model. Acta Geodaetica et Cartographica Sinica 45 (S1), 106–114.
Moussalli, R., Srivatsa, M., Asaad, S., 2015. Fast and Flexible Conversion of Geohash Codes to and from Latitude/Longitude Coordinates. IEEE International Symposium on Field-programmable Custom Computing Machines.
Huang, K., Li, G., Wang, J., 2018. Rapid retrieval strategy for massive remote sensing metadata based on GeoHash coding[J]. Remote Sensing Letters 9 (11), 1070–1078.
Guo, N., Xiong, W., Wu, Y., et al., 2019. A geographic meshing and coding method based on adaptive Hilbert-Geohash. IEEE Access 7, 39815–39825.
Lin, W., Cheng, C., Wu, S., et al., 2015. Massive remote sensing image data management based on HBase and GeoSOT. In: Geoscience & Remote Sensing Symposium 2015.
Jensen, C.S., Tielsytye, D., Tradilauskas, N., 2006. Robust B+-tree-based indexing of moving objects. In: International Conference on Mobile Data Management. 2006.
Davis, N., Raina, G., Jagannathan, K., 2018. Taxi Demand Forecasting: A HEDGE-Based Tessellation Strategy for Improved Accuracy. IEEE Transactions on Intelligent Transportation Systems (99) 1-12.
Jin, A., Cheng, C., Song, S., et al., 2013. Regional query of area data based on geohash. Geography and Geo-Information Science 29 (5), 31–35.
Hartmann, A., Meinel, G., Hecht, R., et al., 2016. A workflow for automatic quantification of structure and dynamic of the German building stock using official spatial data. ISPRS Int. J. Geo-Inf. 5 (8), 142.
Zhong, Y., Han, X., Zhang, L., 2018. Multi-class geospatial object detection based on a position-sensitive balancing framework for high spatial resolution remote sensing imagery. ISPRS J. Photogramm. Remote Sens. 138, 281–294.
Lawder, J.K., King, P.J.H., 2000. Using space-filling curves for multi-dimensional indexing. In: British National Conference on Databases. Springer, Berlin, Heidelberg, pp. 20–35.
Qian, C., Yi, C., Cheng, C., et al., 2019. Geosot-based spatiotemporal index of massive trajectory data. ISPRS Int. J. Geo-Inf. 8 (6), 284.
Morton, G.M., 1996. A computer oriented geodetic data base and a new technique in file sequencing. IBM Germany Scientific Symposium Series 294–897.
Li, C., Wu, Z., Wu, P., et al., 2019. An Adaptive Construction Method of Hierarchical Spatio-Temporal Index for Vector Data under Peer-to-Peer Networks. ISPRS Int. J. Geo-Inf. 8 (11), 512.
Griffiths, J.G., 1986. An algorithm for displaying a class of space-filling curves. Software: Practice and Experience 16 (5), 403–411.
Faloutsos, C., Roseman, S., 1989. Fractals for secondary key retrieval. In: Proceedings of the eighth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, pp. 247-252.
Moon, B., Jagadish, H.V., Faloutsos, C., et al., 2001. Analysis of the clustering properties of the Hilbert space-filling curve. IEEE Trans. Knowl. Data Eng. 13 (1), 124–141.
Liu, X., Schrack, G., 1996. Encoding and decoding the Hilbert order. Software: Practice and Experience 26 (12), 1335–1346.
Bylinski, C., 1989. Binary operations. Journal of Formalized Mathematics 1 (198), 9.
Tong, X., Cheng, C., Wang, R., et al., 2019. An efficient integer coding index algorithm for multi-scale time information management. Data Knowl. Eng. 119, 123–138.
Lv, X., Cheng, C., Gong, et al., 2011. Review of data storage and management technologies for massive remote sensing data. Science China Technological Sciences 41 (12), 1561–1573.
Chang, Fay, Dean, et al., 2008. Bigtable: a distributed storage system for structured data. ACM Transactions on Computer Systems 26 (2), 1–26.
Jy-Yong Sohn, Choi, B., Yoon, S.W., et al., 2017. Capacity of clustered distributed storage. IEEE Transactions on Information Theory, PP (99):1-1.
Hu, C., Guan, Q., Chen, N., et al., 2014. An observation capability metadata model for EO sensor discovery in sensor web enablement environments. Remote Sensing 6 (11), 10546–10570.
Bayer, R., 1997. The universal B-tree for multidimensional indexing: general concepts. In: International Conference on Worldwide Computing & Its Applications.
Zhang, D., Baclawski, K.P., Tsotras, V.J., 2009. B+-Tree. Encyclopedia of Database Systems 288 (22), 15537–15546.
Zhai, W., Bo, Chen, Tong, X., et al., 2018. Research on continuity of multi-scale space-filling curves. Acta Scientiarum Naturalium Universitatis Pekinensis 54 (2), 331–335.
Kothuri, R.K.V., Ravada, S., Abugov, D., 2002. Quadtree and R-tree indexes in oracle spatial: a comparison using GIS data. In: Proceedings of the 2002 ACM SIGMOD international conference on Management of data, pp. 546–557.
Hu, Y., Ravada, S., Anderson, R., et al., 2012. Topological relationship query processing for complex regions in Oracle Spatial. In: Proceedings of the 20th International Conference on Advances in Geographic Information Systems, pp. 3–12.