An Efficient Tile-Pyramids Building Method for Fast Visualization of Massive Geospatial Raster Datasets

Ning GUO, Wei XIONG*, Qiuyun WU, Ning JING
College of Electronic Science and Engineering, National University of Defense Technology, 410073, Changsha, China
*xiongwei@nudt.edu.cn

***Abstract*—Building tile-pyramids is an effective way for publishing and accessing the map visualization service of large-scale geospatial data in the web. But it is a time-consuming task in Geographic Information System (GIS) to build tile-pyramids using traditional methods. In this article, an adaptive multilevel tiles generation method is proposed, which first builds grid index for the geospatial raster dataset, and then generates tiles according to different hierarchy level numbers in the tile-pyramid. With the optimized map rendering engine implemented, a parallel tiles pyramid generation method for large-scale geospatial raster dataset is integrated into a high performance GIS platform. Proved by experiments, the new method shows acceptable applicability, stability and scalability besides its high efficiency.**

***Index Terms*—geographic information systems, indexing, parallel algorithms, spatial resolution, tiles.**

I. Introduction

With the development of the technology of obtaining geospatial data, many organizations and research teams are able to get high resolution geographic images. High resolution image datasets are geographically interfaced and are often stored in the disk file system in the format of raster, while the dataset's large-scale volume places restrictions on the efficient organizing and indexing of itself in traditional database, which becomes the bottle-neck of efficient image visualization. Fast browsing of massive raster dataset has become an urgent demand in the web geographic applications.

Currently, visualization of large-scale geospatial raster dataset costs long time in data pre-processing, which consist of coordinate transforming, image mosaicking, image-pyramid and tile-pyramid building. The long pre-processing time is too insufferable for the decision-making department. For instance, a relevant department gets thousands of raster images whose data size reach up to hundreds of GB in the resolution of 0.2 meter of a county using unmanned aerial vehicle. If taking the traditional methods, it will cost several days to mosaic them to a single geo-tiff image and build an image-pyramid. After that, the visualization of the area's map image can be possible to realize. To be more specific, if we use the popular software ArcGIS, it will take a week or even more to process the image data. Besides, it will be an awful experience with intolerable time-delay when users zoom or drag the image in the screen using the huge well-built image-pyramid, especially when the operations are highly concurrent. To solve the problem, researchers invented the tile technology, which caches a tile-pyramid on the basis of image-pyramid. When visit the map, we only need to call the relevant tiled image instead of the whole image, which increases the performance of map service greatly. There are many Geo-data services take advantages of tile technology to accelerate concurrent map service, including ArcGIS, MapGIS, Google Map and Baidu Map etc.

But for massive raster datasets, the traditional methods of building tile-pyramids are too inefficient to satisfy the demand of increasing size of geo-dataset. As a result, how to provide tile service rapidly for massive raster dataset becomes an urgent problem to be solved technically. This paper proposed an adaptive multilevel method for generating tiles for massive raster dataset, which can dynamically switch rendering tragedy according to the viewport display level. Apart from that, parallel ideas are implemented in data processing progress. Proved by experiments, this method can build tile-pyramids for massive raster datasets at a satisfying speed. By this method, the data production to service publishing's cycle is shortened greatly.

II. Related Works

Research on visualization and rapid tile generation of large-scale data mainly focuses on the storage management of massive image dataset and optimization measures of the traditional tile generation method.

In literature [1], X. Wang et al. proposed a construction and index method of tile-pyramid based on tile and hierarchy of the terrain data, realizing the real-time rendering of multi-resolution map; Wang also concerned of the problem encountered in the process of tiling the vector data of variety map projection type; In [2], Y. Wang et al. extended an open source application called "TileStache" based on OGR geographic data models, realizing the fast tiles generation for vector geographic data; J. Li et al. took advantages of parallel programming model Map-Reduce and used dynamic data dividing mechanism and space adjacent based uploading mechanism to optimize slice algorithm. And they proposed a real-time slicing method for geo-images [3]; Y. Zhao et al. proposed a dynamic projection algorithm which using a polynomial to do numerical projections for every tiles and a fast-clipping algorithm based on scan line

This work was supported in part by the National Natural Science Foundation of China under Grants 41271403 and Grant 41471321.

Digital Object Identifier 10.4316/AECE.2016.04001

filling algorithm, finally realized efficient one-time automatic slicing under multi-constraint condition [4]; Z. Du proposed an approach to using multispectral images to store point cloud data and create point cloud pyramid to improve rendering efficiency of different LOD (level of details) [5]; X. Yin studied large scale terrain data's management and visualization based on the CUDA GPU parallel computing environment [6]; C. Dai et al. proposed a terrain data organization method based on tile-pyramid and linear quad tree index. They also optimized the retrieved strategy of the selected area which can quickly find the target tiles [7].

Previous research focused mainly on a single-file raster data, which did not take specific consideration of some datasets containing large number of image files. However, a typical progress of building tile-pyramid for datasets is formed gradually and applied widely. The typical process including image mosaicking, image-pyramids building and tile slicing which is shown in Figure 1.

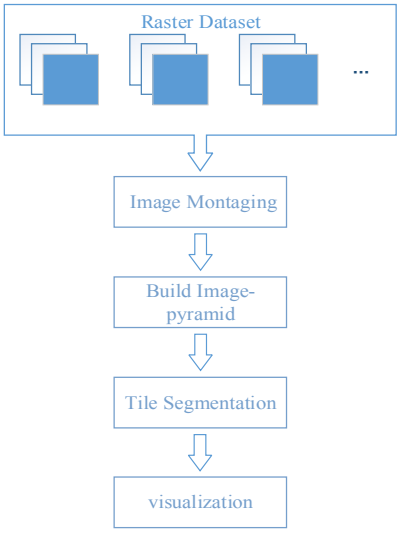

Figure 1. General progress of dataset visualization

Schematic of image mosaic is as shown in Figure 2.

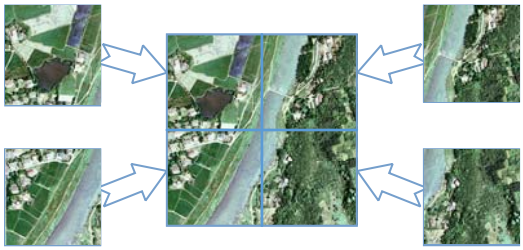

Figure 2. Sketch map of image mosaic

Image-pyramid is a set of deferent resolution of image generated from the original image by certain sampling rules. It is a typical form of layered data structure designed for organizing multi-resolution raster data and it can significantly reduce image rendering time. So image-pyramid became a widely-used data structure in GIS. Many literatures have introduced the generation and maintenance theories of it. Some classical methods involve quad-trees triangulation or meshes and other adaptive optimization measures with the purpose of accelerating the rendering efficiency of multiple level-of-detail (LOD) [8-11]. The logical structure of the pyramid is shown in Figure 3.

Each layer of the pyramid corresponds to a resolution of the raster data, and when users request the map from their point of view, the new display resolution and geographic range of the current viewport will be calculated. Then according to the matched resolution of the image-pyramid, the appropriate maps will be displayed to the users [12].

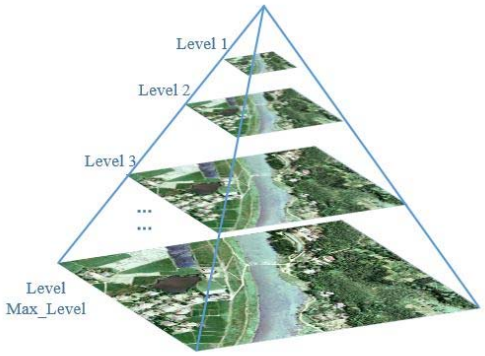

Figure 3. Sketch map of image-pyramid structure

When the data size reaches up to hundreds of GB scale, establishment of the pyramid would be an extremely slow process. For example, using the ArcGIS software to build a 137GB 3-band raster image dataset's pyramid will take more than 6 hours. We have proposed a MPI-based algorithm to accelerate the process [13], which can greatly improve the performance of pyramid building for single-file raster data. However, it is still a slow process while facing the large number of raster data files.

Tile-pyramid is a similar multi-resolution data structure model like image pyramid. The only difference is that each layer in the tile-pyramid is formed by slicing the image at the same level in the image-pyramid. Each tile has the same size of 256×256 pixels and corresponds to a same geographic range [9], [13]. Substantially, it is equivalent to a grid index whose grid size is adaptive to the level. When you select a specific geographical area, the location of target image data can be obtained by some simple calculations according to the tile division step's length [14]. Figure 4 shows the structure of a tile-pyramid.

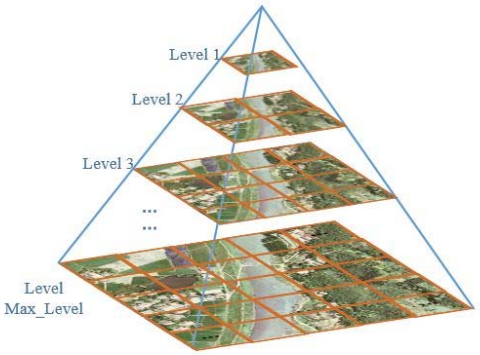

Figure 4. Sketch map of tile-pyramid structure

Judging from the research status, existing GIS require image mosaic and pyramid construction pretreatment process in order to generate a complete tile-pyramid to provide map tile services. The overall time cost of visualization of datasets is far from satisfactory. And in the course of data processing, parallel programming is rarely adopted. Based on the above cogitation, we propose an adaptive Multilevel tiling method of a datasets.

III. MULTILEVEL ADAPTIVE TILE METHOD OF GENERATING

Mapnik is an open source Python/C++ map rendering engine. Its functions are to pack geographic objects such as maps, layer, data source, characteristics of geometry to a defined XML-style file, and rendering it to a bitmap to provide web map services (WMS). Mapnik not only supports multiple platforms and data formats, but also adapts well to multithread environments so it is ideal to provide GIS services for Web applications.

This article proposed a new layered drawing method based on the existing technology solution which changes drawing strategies in real time according to the zoom level selected by users. If the display level is high, it uses a low-level view of sampling data to build tiles. If the display level is low, it uses the index to query image files in the viewport. After that, dynamic XML-style strings required by Mapnik are generated in real time, so that the rendering engine can segment images to tiles for visualization.

A. Overview of the new method

The map display level refers to the viewport zoom level. Different levels show corresponding view scale of map. The smaller the level number is, the wider the viewport displays and calls the higher level tiles in pyramid, so the display resolution is lower. The bigger the level number is, the narrower the viewport displays and calls the lower level tiles in pyramid, so the display resolution is higher. Generally, the highest resolution that the viewport can display equals to the resolution of the source data.

This paper presents an adaptive tiling method which could switch rendering tragedy dynamically according to the display level. GIS usually has 20 display levels, and we choose a middle threshold view level denoted as level N where the intersection number of image files equal to 30.

If level number is bigger than N, that is the lower part of the pyramid: we call the open source slice tool Mapnik with a new way to provide it with the XML style sheet. According to each tiles' MBR (Minimum Boundary Rectangular), retrieve its space cover range's intersection with grid image files. And depend on the change of query window, XML style strings are generated dynamically to be provided to Mapnik for tile slicing.

If level number is smaller than N, that is the higher part of the pyramid: we generate higher levels of tiles using interval sampling tiles data in level N hierarchically. After that a complete tile-pyramid is built and stored in the cache. When you zoom in or out to view a specific level, just call the appropriate levels of sliced data and it will be quickly displayed.

In traversing the intersect data to build index, querying the data files as well as resampling the lower level tiles to generate the higher level tiles' process, parallel programming ideas are used to maximize the use of multi-core hardware environments, which greatly improve the efficiency of data processing.

In this way, we skip the steps of image mosaic and image-pyramid building. Instead, we generate the tile-pyramid directly, and effectively speed up the process of visualization of datasets. The overall process is shown in Figure 5.

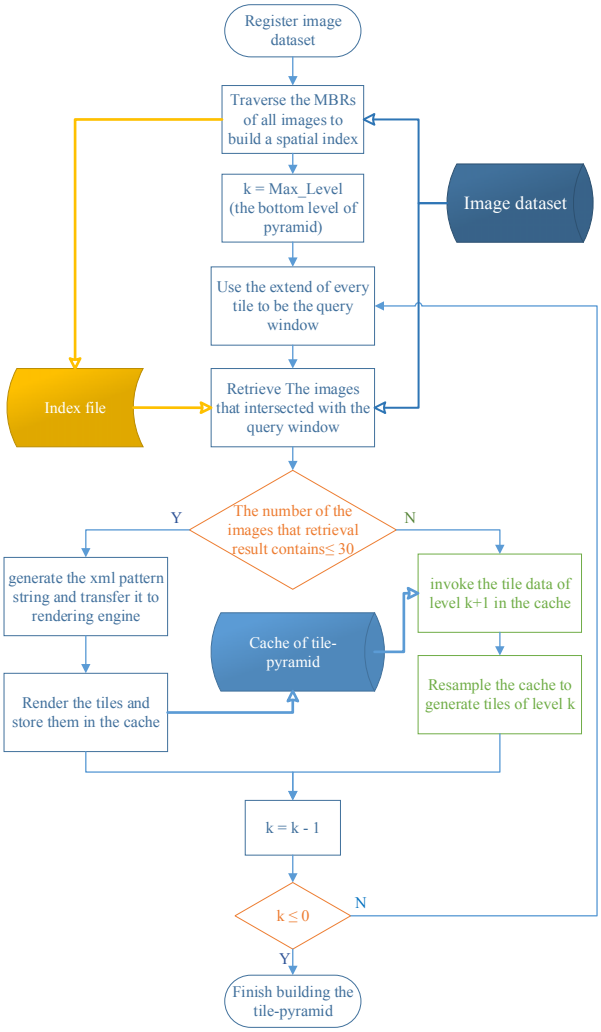

Figure 5. Overall process of the new method

B. Massive raster dataset grid index

Because most of raster data are simple bounded, grid index is sufficient for querying the raster image that is spatially continuous. The index's data structure is as shown in Figure 6 and Figure 7

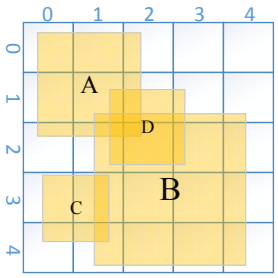

Figure 6. Establishment grid index for the dataset

The detailed grid index's process is described as follows:

1. Traverse the dataset using open-source Geospatial Data Abstraction Library (GDAL) [15], select the appropriate grid size according to each single TIFF image's MBR. We set the size to be a half of the minimum edge length of all MBRs.

2. Established grid index storage structure, and compute the intersect situation between TIFF images and each grid according to their MBR. Then register the images to their intersected grids. Each grid cell maintains a list of images. The registered object's data structure can be designed

specifically. For example, our design of the index object includes TIFF image's storage path in the file system and their geographic coverage range, namely MBR, which could be used for follow-up fine filter operation. Other image property information can be added to the index structure in light of users' requirements. The index file is stored in the file system to be called by rendering engine.

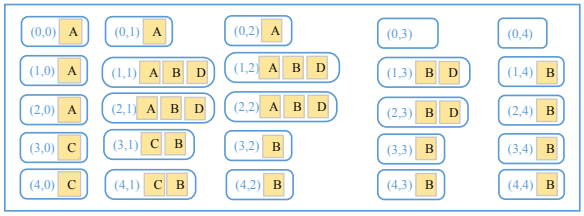

Figure 7. Data structure of the grid index

The following is a simple example of querying progress.

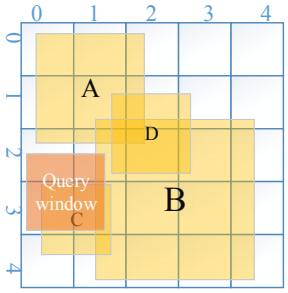

Figure 8. Using the MBR of a tile as query window

As Figure 8 shows, presume the dark grey box to be the coverage range of a random tile and transfer its MBR to the query function integrated in the rendering engine, so that we can retrieve the intersected grid's ID with the pre-built index file. In this example, the grid ID query result is (2,0), (2,1), (3,0), and (3,1); After that, we pop out all the objects contained in these grids as Figure 9 shows and remove the repeated items.

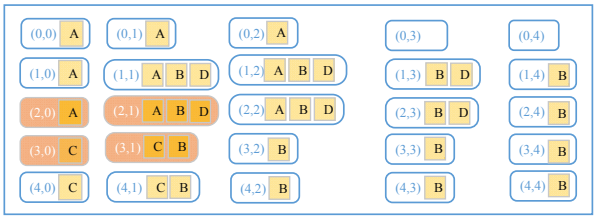

Figure 9. Query result

After we get the rough query results of A, B, C, and D, it is necessary to do refined filter by their MBR's spatial relation with query window's. Eventually we get the final query results: B and C. Their full path will be passed to a special function to generate XML-style string provided to the Mapnik rendering engine. After this tile is drawn, traverse all tiles hierarchically follow the procedure above, a tile-pyramid will be built completely.

Because the actual raster dataset may contain thousands of image files, using the tiling strategy described above is suitable for high levels' tiles, which cover smaller spatial scale so that we can quickly retrieve the few intersected images in the dataset, and the XML-style string generated is also short which greatly reduce the burden of the rendering engine for a quick visualization purposes.

As we can see, the query result by the grid index before refined filter is redundant because the objects in the grids that achieved are not always intersected with the query window. And if we reduce the grid size to improve retrieval precision, the generated index file volume will increase, causing unnecessary reading time cost and retrieving burden, which affect the efficiency of the visualization in the end. To tackle the problem, refined filter of the middle result is necessary to minimize the size of the result set which will be transferred to Mapnik. And to be more adaptive, other types of spatial index will be introduced to the solution in our future work, which can also decrease the retrieval redundancy without the need of an additional filter to the result.

C. Low level tile generation method

For the lower levels in tile-pyramid, the pre-built index is essential to the tiles generation. From the bottom level to threshold level, we use each tile's MBR to be the query window and retrieve the intersected images to render the tile image. Since the pre-set file number threshold is 30, if the query window's spacial range intersects less than 30 image files (assuming the progress comes to level N pyramids when the file number threshold is reached), in that level number below N, we use the original image data to generate tiles data in the way described in Chapter 2.2: Reading index objects' full path as well as MBR information and generating XML style string to pass to Mapnik rendering engine. Then Mapnik reads image data to slice the tiles within the viewport according to the style string and save them in the tile-pyramid cache.

D. High-level parallel tiling method

Cause the tiles in high level covers larger spatial range, they intersect with large quantity of images in the dataset. It is insufferable to render the tiles using the index method above. That is the main reason why existing GISs take so much time to build a tile-pyramid. We apply a new method called resampling to do the time-consuming job: based on the tiles data in level N, resample to generate tile data in level 1 to N-1 layer-by-layer. Specific sampling procedures are as showed in Figure 10. When generating the k ($1 \leqslant k \leqslant N-1$) level tiles data, use the k+1 level tiles data as sampling source. In k+1 level tiles data, make every four adjacent tiles to be a unit as a tile package. In each tile package, one line is taken from each two lines and one column is taken from each two columns to generate tiles in the higher level. In other words, it produces a 256×256 pixels tile in the k level sourcing from a 512×512 pixels tile package in the k+1 level. Because the low-level view displays larger areas, the sampling tile's resolution does not affect the view effect.

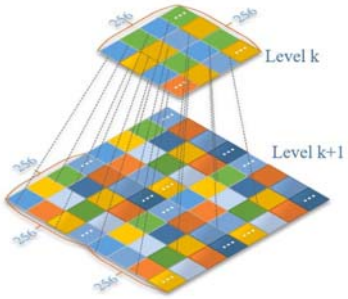

Figure 10. Resampling to generate higher level tile

IV. EXPERIMENT

We implement the new tile-pyramid building method based on a high performance GIS platform named HiGIS [16]. The performance of this method is compared with ArcGIS software.

Experimental data: Wuling district in Hunan province's high-resolution raster image dataset.

Size of experimental dataset: 10-1000 TIFF raster files.

Single image size: 79.322MB, 5201x5201 pixels.

Experimental environment: Super-micro high performance servers.

TABLE I. HARDWARE ENVIRONMENT

Environment	HiGIS	ArcGIS
CPU	64 cores	64 cores
RAM	512 GB	512 GB
OS	CentOS 6.5	Windows 7
File System	GPFS	NTFS

To evaluate the new method, we first adopted the traditional visualization process like ArcGIS does, in other words, first mosaic dataset to one image, then build an image-pyramid, finally cut it to a tile-pyramid. We found that when process GB-sized raster dataset containing hundreds TIFF files. ArcGIS's processing capacity limit has been reached. For example, mosaicking a 3000 TIFF-file dataset to a single image will cost hundreds of hours due to the hardware calculation resources are not fully used. As a result, it cannot finish building a tile-pyramid for such dataset in foreseeable time, so eventually we set the dataset's upper limit size to be 1000 TIFF-file.

As the most advanced GIS software in the industry, ArcGIS also provides a mosaic raster dataset way to display a regional image dataset except traditional tile-pyramid building method. Users can take advantages of the Mosaic Dataset tool in ArcGIS's ToolBox assembly. This tool does not mosaic images in physical, but realizes virtual mosaic in memory according to each image's geographic range. After that, the image-pyramid and tile-pyramid can be established for providing tile service for map visualization.

The following two tables show the experimental platform HiGIS's and ArcGIS's experimental results and Figure 11 is the line chart of the comparison.

Line chart's abscissa axis refers to datasets' scale, which is represented by the number of image files that the dataset contains; the axis of ordinate refers to the time cost of building a whole tile-pyramid for the corresponding dataset. As can be seen from the chart, when the raster datasets are in small scale, the efficiency of the new method and ArcGIS differs within an order of magnitude. But when a dataset contains up to hundreds or even thousands of images, new method can finish the building task as fast as more than 20 times of ArcGIS does. In addition, with the datasets' scale grows, this method takes a linear upward trend time cost, other than ArcGIS's phenomenon of exponential growth. We could safely draw the conclusion that our new method manages to improve the efficiency of building tile-pyramids for large-scale raster dataset greatly. And it also has a good running stability.

Besides, the indexes we built occupy tiny of memory space, comparing to the gigantic size of raster datasets. The indexs'sizes vary from 1kB to 1MB when the datasets' scales range from 10 to 1000 tiff files. And we can see that the timecost of building the index is negligibly small.

TABLE II. STATISTICS OF TILE-PYRAMID BUILDING TIMECOST

Dataset scale	Index building/s	Generate the tiles in level 15-20/s	Generate the tiles in level 1-14 by resampling/s	Total timecost/s
10	0.037	402	0.189	402.226
20	0.045	478	0.218	478.263
50	0.054	605	0.859	605.913
100	0.071	957	1.737	958.808
200	0.116	2286	0.824	2286094
500	0.939	4227	1.071	4229.01
1000	3.221	14250	3.942	14257.163

TABLE III. STATISTICS OF TILE-PYRAMID BUILDING TIMECOST USING MOSAIC DATASET TOOL IN ARCGIS

Dataset scale	Build image-pyramid/s	Create thumbnail/s	Build tile-pyramid/s	Total timecost/s
10	22.66	14.14	2021.94	2058.74
20	42.79	29.96	1768	1840.75
50	103.1	46.98	11354	11504.08
100	200.1	136	22489	22825.1
200	423.16	237	47103	47763.16
500	1160.16	615	123101	124876.16
1000	2451.27	1320	265139	268910.27

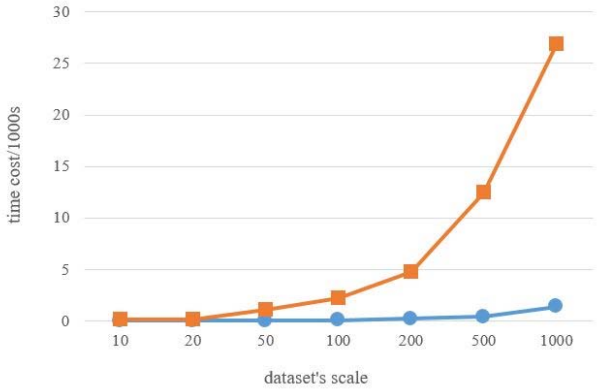

Figure 11. Time cost Statistics and comparison

Fig.12 and Fig.13 show the visualization effects of HiGIS based on our new method and the mosaic dataset tool of ArcGIS. But as we drag the map or do zoom in or out operation, the user experiences differ from each other to a large extent. HiGIS respond quickly and displays smooth effect of visualization, while ArcGIS can hardly display the result to its users within tolerable delay time.

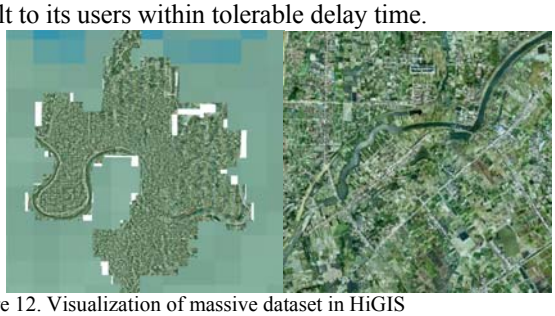

Figure 12. Visualization of massive dataset in HiGIS

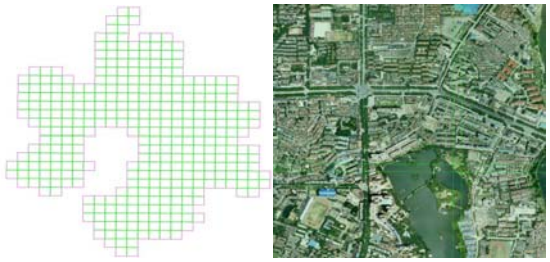

Figure 13. Visualization of mosaic dataset in ArcGIS

V. Conclusion

With experimental validation, the Multilevel adaptive method of building tile-pyramids for massive raster dataset shows dozen times of efficiency improvement comparing with ArcGIS, the most common-used GIS software in the geographic information industry. And it is also capable of supporting the tiling process for much larger size of dataset. In our practical application, the method has completed many tiling tasks which far beyond ArcGIS's capability. With the high performance of the methods, we get closer to the target of "visualizing upon obtaining" for geographic image data, which promotes time-sensitive decision and application based on fast visualization [17].

In our future work, further enhancement to the universality of the method will be under thought. For large scale spatial data in different formats, we can make use of the multilevel adaptation strategy similar to the new method [18]. Besides, more efficient and diverse spatial indexes, like Quad-tree [19], r-tree and Geohash, could be applied in the method to improve query precision in order to reduce the retrieval redundancy, which will further alleviate rendering pressure. If possible, distributed spatial index structure could be designed to take advantages of the distributed hardware environment [20-21].

Acknowledgment

Thanks to the hardware maintainers and software developers in DBRG, especially to Xing Jin, a young skilled front-end web programmer. Thanks to senior laboratory Anran Yang, an experienced, talented PhD, who gave me plenty valuable advices on algorithm development.

References

[1] X. Wang, F Zhang and L Zhang, "Tile-pyramid Construction and Organization Based on Terrain Data," Mapping and Geospatial Information, vol. 35, no. 6, pp. 49-51, Jun. 2012. doi: 10.3969-/j.issn.1672-5867.2012.06.014.

[2] Y. Wang, Y. Pu, L. David and X. Song. "Tile Generation of Multi-Source Projection Vector Data Based on TileStache," Geo-information World, vol. 22, no. 1, pp. 77-81, Jan. 2015. doi: 10.3969/j.issn.1672-1586.2015.01.020.

[3] J. Li, B. Gan, L. Meng, W. Zhang and H. Duan, "Fast Section of Sequential Remote Sensing Image Cache in the Cloud Environment," Journal of Information Sciences, Wuhan University, vol. 40, no. 2, pp. 243-248, Feb. 2015. doi: 10.13203/j. wugis20130079.

[4] Y. Zhao and N. Wang, "A Quick Tile Caching Generating Method Based on Dynamic Projection and Scan-Line Cropping," Geomatics Science and Technology, vol. 34, no. 41, pp. 34-41, April 2015. doi: 10.12677/GST.2015.32006.

[5] Z. Du and Q. Li. "A New Method of Storage and Visualization for Massive Point Cloud Dataset," In Proc. CIPA Symposium, Kyoto, Japan, Oct. 2009.

[6] N. Kang, Q. Xu, Y. Zhou and C. Lan, "A Graphic Hardware-based Algorithm for Visualization of Massive Terrain Dataset," Journal of System Simulation, vol. 19, no. 17, pp. 61-64, Sept. 2007. doi: 10.16182/j.cnki.joss.2007.17.039.

[7] C. Dai, Y. Zhang, X. Deng and Z. Geng, "Fast Rendering of Massive Textured Terrain Data," In proc. ASPRS Annual Conference, Reno, Nevada, May 2006.

[8] R. Pajarola, "Large Scale Terrain Visualization Using the Restricted Quadtree Triangulation," In Proc. IEEE Visualization Conference, IEEE, pp. 19-26, Oct. 1998. doi:10.1109/VISUAL.1998.745280.

[9] L. Hwa, M. Duchaineau and K. I. Joy, "Real-time Optimal Adaptation for Planetary Geometry and Texture: 4-8 Tile Hierarchies," IEEE Transactions on Visualization and Computer Graphics, vol. 11, no. 4, pp. 355-368, June 2005, doi: 10.1109/-TVCG.2005.65.

[10] R. Westerteiger, A. Gerndt, B. Hamann, "Spherical terrain rendering using the hierarchical HEALPix grid," In Proc. IRTG, Kaiserslautern, Germany, pp. 13-23, Oct. 2011. doi: 10.4230/OASIcs.VLUDS.-2011.13

[11] P. Lindstrom and V. Pascucci, "Terrain simplification simplified: A general framework for view-dependent out-of-core visualization," IEEE Transaction on Visualization and Computer Graphics, vol. 8, no. 3, pp. 239-254, July-Sept. 2002. doi:10.1109/TVCG.2002.1021577.

[12] F. Losasso and H. Hoppe, "Geometry clipmaps: terrain rendering using nested regular grids," ACM Transactions on Graphics, vol. 23, no. 3, pp. 766-773, 2004. doi: 10.1145/1015706.1015799.

[13] G. He, W. Xiong, L. Chen, "A MPI-based parallel pyramid building algorithm for large-scale remote sensing images," In Proc. Geoinformatics, 2015 23rd International Conference on. IEEE, pp. 1-4, 2015. doi: 10.1109/GEOINFORMATICS.2015.7378567.

[14] A. Liu, Q. Du, D. Zhang, Z. Cai and H. Li, "Organization and Indexing Mechanism for Global Tile Map Data Under Embedded Environment," Geomatics and Information Science of Wuhan University, vol. 40, no. 4, April 2015, doi: 10.13203/j.whugis-20140415.

[15] C. Qin, L. Zhan and A. Zhu, "How to Apply the Geospatial Data Abstraction Library (GDAL) Properly to Parallel Geospatial Raster I/O," Transactions in GIS, vol. 18, no. 6, pp. 950-957, 2014, doi: 10.1111/tgis.12068.

[16] W. Xiong, L. Chen, "HiGIS: An Open Framework for High Performance Geographic Information System," Advances in Electrical and Computer Engineering, vol.15, no. 3, pp. 123-132, 2015. doi: 10.4316/AECE.2015.03018.

[17] R. Barton, "Modern Algorithms for Real-Time Terrain Visualization on Commodity Hardware," In Proc. Geoinformatics FCE CTU, 2010. doi: 10.14311/gi.5.1.

[18] M. Duchaineau, M. Wolinsky, D. Sigeti, M. Miller, M. Mineev-Weinstein and C. Aldrich, "ROAMing Terrain: Real-time Optimally Adapting Meshes," In Proc. IEEE Visualization, pp. 81–88, Oct. 1997. doi: 10.1109/VISUAL.1997.663860.

[19] R. Pajarola, M. Antonijuan and R. Lario, "QuadTIN: Quadtree Based Triangulated Irregular Networks," In Proc. VIS'02: Proceedings of the Conference on Visualization, pp.395–402, Nov. 2002. doi: 10.1109/VISUAL.2002.1183800.

[20] M. Clasen and H. Hege, "Terrain Rendering Using Spherical Clipmaps," In Proc. Joint Eurographics - IEEE VGTC Symposium on Visualization, pp.91-98. May 2006. doi: 10.2312/VisSym/EuroVis06-/091-098.

[21] S. Rusinkiewicz and M. Levoy, "QSplat: A Multiresolution Point Rendering System for Large Meshes," In Proc. International Conference on Computer Graphics and Interactive Techniques, pp. 343-352, 2000, doi: 10.1145/344779.344940.