GIS Data Compression Based On Embedded Approach

Phyo Phyo Wai
University of Computer Studies Mandalay
Phyophyowai84@gmail.com

ABSTRACT

Geographical Information System (GIS) is rapidly developing and then compression techniques for sensor image data are becomes an active research area in the field of GIS. The purpose of this paper is to get best solution of GIS data compression for information collection. This paper mentions the compression approaches of blockwise coding and quadtree coding. Compression techniques are very important for GIS. Raster data images are very large size. So, it needs to compress these images for save storage space and effective transition on current network.

Keywords: Compression techniques, GIS.

1. INTRODUCTION

With the development of GIS technology, the compression techniques of image data are very important. Raster image data are heavy size. So, it needs to compress.

Large amounts of data can create enormous problems in storage space and transmission time. The main reasons of data compression could be summarized as:

- Multimedia data have large data volume
- Difficulty sending real-time uncompressed data over current network

Compression data principle is to eliminate data redundancy and try to find a code with less data volume. All image compression techniques try to get rid of the inherent redundancy, which may be spatial (neighboring similarity or equal pixels), spectral (pixels in different spectral bands in a color image) or temporal (correlated images in a sequence, e.g. television).

In this paper, it presents the compression approaches. These methods are blockwise and Quadtree approach.By performing blockwise compression, each block can have its own compressed bitsream [3].

Quadtree is a class of hierarchical data structures which contains two types of nodes: non-leaf node and leaf node, or you can call them internal node and external node. Owing to its "Divide and Conquer" strategy, it is most often used to partition a two dimensional space by recursively subdividing it into four quadrants or regions [4].

This paper mentions the compression idea of block-wise and quadtree. The rest of this paper is organized as follows. Section 2 shows the related work for this paper. In section 3, we briefly describe some background theories about system architecture, raster imagery foundations, compression techniques, blockwise coding, and quadtree coding. Section 4 is about propose system. The paper is concluded in section 5.

2. RELATED WORK

The maintenance of large raster images under spatial operations is still a major performance bottleneck. For reasons of storage space, images in a collection, such as satellite pictures in geographical information systems, are maintained in compressed form [2].

Raster data structure expresses spatial objects in the form of grid cells or pixels into a matrix. It is convenient for spatial analysis and it can be easily integrated with DTM and digital remote sensed images in geographical information system (GIS). This structure is one of the effective models to represent spatial objects in GIS. When raster data structure is used, the smaller the size of pixel or grid cell is, the higher resolution of spatial data comes and the more cells with the same attribute value are resulted in, which causes a large amount of redundant information. The memory space required to store spatial data is large. This requirement causes other problems for a GIS. As there are heavy loads for both spatial data processing and communication, much research is concerned with the storage problem. A quadtree is a well-known hierarchical data structure applied to represent geometric data. It is especially useful for raster-based data processing and important for many applications, such as computer vision, computer graphics, image processing, cartography, and so on (Samet, 1990) [1].

3. THEORITICAL BACKGROUND

3.1. Raster imagery foundations

Technically speaking a raster image is a two-dimensional rectangular matrix of individual **pixels**. Each pixel corresponds (directly or indirectly) to some well defined **color**. And colors are defined as **RGB** values, i.e. as a triplet of values indicating the **Red**, **Green** and **Blue** relative intensity. Most usually an **8-bit** color depth is used, and so a **0** value corresponds to "*completely off*", i.e. black, and **255** to "*completely on*", i.e. pure red or pure green or pure blue. Any real color can be encoded using an appropriate RGB value: an 8-bit color depth allows RGB to represent **16,777,216** different colors, i.e. the so called **true color** commonly used by digital cameras, screens, color printers and other well known digital imagery equipments. This is also known as the **RGB color space** (*commonly used on color digital photography*): each pixel requires 3 bytes, i.e. *24 bits* to be represented into this color space [5].

RGB hex value	Red	Green	Blue	Color	Example
0x000000	0	0	0	Black	
0xFFFFFF	255	255	255	White	
0x808080	128	128	128	Medium Gray	
0xD0D0D0	208	208	208	Light Gray	
0xFF0000	255	0	0	Red	
0x00FF00	0	255	0	Green	
0x0000FF	0	0	255	Blue	
0xFFFF00	255	255	0	Yellow	
0x00FFFF	0	255	255	Cyan	
0xFF00FF	255	0	255	Magenta	

Figure1: Color Table

As it can easily notice from the above table, a full 256-levels gray scale can always be represented using a single channel, because any gray value always has identical values for Red, Green and Blue. And this defines the GRAYSCALE color space (commonly used on black and white digital photography): each pixel requires a single byte, i.e. 8 bits to be represented into this color space.

In the MONOCHROME color space only two colors are supported (usually, black and white): each pixel requires a single bit, i.e. one single byte can store 8 pixels.

As an alternative way, it can define a PALETTE-based color space. The palette stores a limited set of RGB values (usually, max. 256), and consequently each pixel doesn't requires any longer a full RGB value: it will simply store a palette index, thus indirectly retrieving the corresponding RGB value: consequently, each pixel requires a single byte, i.e. 8 bits to be represented into this color space.

Quite obviously, a lot of different color spaces exist: but they aren't too much widespread, so we can ignore them at all [5].

Color Space	Single Pixel Size	Notes
RGB	3 bytes 24 bits	True color 16 million colors
GRAYSCALE	1 byte 8 bits	256 levels gray scale
MONOCHROME	1 bit	Bi-level Black or white [*no half tones*]
PALETTE	1 byte 8 bits	256 colors palette

Figure2: Color Space and their single pixel size

3.2. Compression Techniques

The goal of data compression is to represent an information source as accurately as possible using the smallest storage space (=number of bits). Purpose of image compression is to remove image redundancy such that storage and/or transmission efficiency be increased.

Data compression principles are listed. Data compression:

- Is the substitution of frequently occurring data items, or symbols, with short codes that require fewer bits of storage than the original symbol.

- Saves space, but requires time to save and extract.
- Success varies with type of data.
- Works best on data with low spatial variability and limited possible values.
- Works poorly with high spatial variability data or continuous surfaces.
- Exploits inherent redundancy and irrelevancy by transforming a data file into a smaller one.

The **compression ratio** is the ratio of the two file sizes. e.g., original image is 100MB, after compression, the new file is 10MB. Then the compression ratio is 10:1.

Two compression techniques are mostly used: **Lossless** and **Lossy** compression [8].

3.2.1. Loseless compression

A **lossless compression** algorithm eliminates only redundant information, so that one can recover the data exactly upon decompression of the file. Lossless data compression is compression without any loss of data quality. The decompressed file is an exact replica of the original one. Lossless compression is used when it is important that the original and the decompressed data be identical. It is done by re-writing the data in a more space efficient way, removing all kinds of repetitions (compression ratio 2:1). Some image file formats, notably PNG, use only lossless compression, while those like TIFF may use either lossless or lossy methods.

Examples of LOSSLESS METHODS are:

- Run-length coding
- Huffman coding
- Lempel-Ziv-Welsh (LZW) method [8].

3.2.2. Lossy compression

A **lossy compression** method is one where compressing data and then decompressing it retrieves data that may well be different from the original, but is "close enough" to be useful in some way. The algorithm eliminates irrelevant information as well, and permits only an approximate reconstruction of the original file. Lossy compression is also done by re-writing the data in a more space efficient way, but more than that: less important details of the image are manipulated or even removed so that higher compression rates are achieved. Lossy compression is dangerously attractive because it can provide compression ratios of 100:1 to 200:1, depending on the type of information being compressed. But the cost is loss of data. The advantage of lossy methods over lossless methods is that in some cases a lossy method can produce a much smaller compressed file than any known lossless method, while still meeting the requirements of the application.

Examples of LOSSY METHODS are:

- PCM
- JPEG
- MPEG [8]

3.3 Blockwise coding

This method is a generalization of run-length encoding to two dimensions. Instead of sequences of 0s or 1s, square blocks are counted. For each square the position, the size and, the contents of the pixels are stored [8].

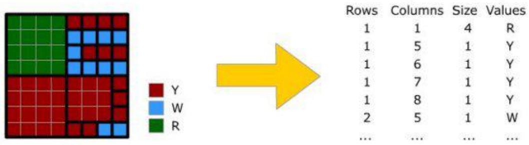

Figure3: Blockwise coding

Blockwise compression or reconstruction is to enable a random access to a compressed bitstream and also regulate the memory usage and the amount of transmitted data [3].

Wherein the prediction value for a given pixel of the block is

L, if the given pixel is in a first row of the block

U, if the given pixel is in a first column of the block F, otherwise, where

F = min(U.L) if S ≥ max(U,L)

F = max(U.L) if S

F = U + L - S otherwise, and where

U is the pixel value of a upper neighbor pixel of the given pixel,

L is the pixel value of a left neighbor pixel of the given pixel, and

S is the pixel value of a left and upper neighbor pixel of the given pixel.

An apparatus for compressing a block of pixels, comprising: means for calculating a prediction value for each pixel of a group of pixels of the block, wherein the prediction value for a given pixel is calculated dependent on pixel values of at least one neighboring pixel of the given pixel, means for calculating the difference between the pixel value and the prediction value for each pixel of the group, means for variable length coding the calculated differences, wherein a same set of coding parameters of the variable length coding is used for coding each difference of the block [6].

3.4 Quandtree coding

The quad tree compression technique is the most common compression method applied to raster data. Quad tree coding stores the information by subdividing a square region into quadrants, each of which may be further subdivided in squares until the contents of the cells have the same values [6].

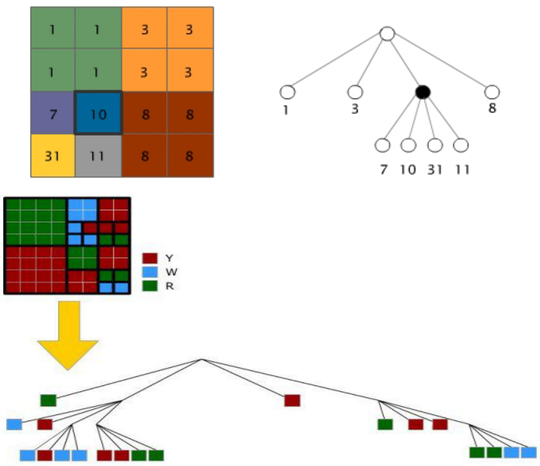

Figure4: Quadtree Coding

According to Same, the types of quadtrees can be classified by following three principles:

1) the type of data that they are used to represent,
2) the principle guiding the decomposition process, and
3) the resolution (variable or not) [7].

Currently, quadtrees can be classified according to the type of data they represent, including areas, points, lines and curves. More generally, quadtrees have been categorized by whether the shape of the tree is independent of the order data is processed. Some common types of quadtrees in this kind of classification are: region quadtree, point quadtree, and edge quadtree.

4. PROPOSE SYSTEM

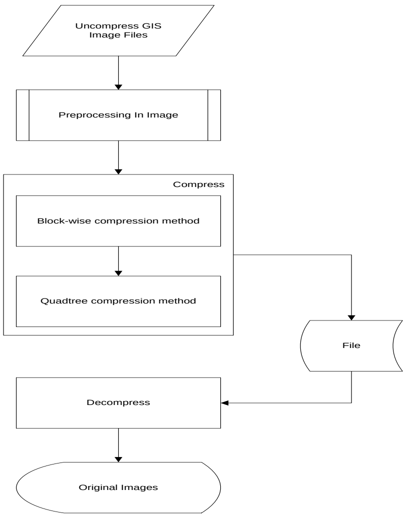

Figure5: Propose System Design

In this propose system, the input data may be GIS image data especially like raster data. And then it takes the action of preprocessing such as removing noise data, image resizing according to the need of this system. After these steps, the compression operation is start. In this step, the blockwise coding method divides the image into blocks, such as n x n block size, according to the bitstream of the image. When the block size is large, the quadtree coding method will performs in this block. The blockwise coding method has neighboring boundary pixel problem and the quadtree coding method has multi-tree level problem. By embedding quadtree coding method in blockwise coding, we will solve these problems, effective the processing time and get high accuracy images results. In next step, it saves the compress image file. And then decompress the image file and show the image like originally.

5. CONCLUSION

The compression of image is save the storage space and it is effective the sending data over current network.

This paper presents the propose system of GIS image compression using block wise coding method that embedded with quad tree method. This system is more suitable and more compress than other compression idea.

REFERENCES

[1] Sheng Yehua, Tang Hong, Zhao Xiaohu, “Linear Quadtree Encoding Of Raster Data And Its Spatial Analysis Approaches”

[2] Renato Pajarola, Peter Widmayer ,"An Image Compression Method for Spatial Search"

[3] JinKook Kim, Jong BeomRa,"A real-time terrain visualization algorithm using wavelet-based compression", Published online: 4 March 2004 Springer-Verlag 2004

[4] Xiang Yin, "Quadtree Representation & Compression of Spatial Data"

[5] www.gaia-gis.it/spatialite/rasterlite-man.pdf

[6] Clair Cho, Eike Grimpe, Yin-Chun Blue Lan, "Method and apparatus for blockwise compression and decompression for digital video stream decoding"

[7] Xiang Yin, "Quadtree Representation & Compression of Spatial Data"

[8] Claudia Dolci, Dante Salvini, Michael Schrattner, Robert Weibel, "Structures for Data Compression"