Images Representation Using Compact Quadtree Traversal Coding Technique

Haithem Kareem Abass

Al-Mansour University College

Abstract :

A quadtree structure is a spatial data structure and considered as one of the most important hierarchical techniques for representing digital images that attempt to save storage by aggregation data that have similar or identical values into blocks considered as quadtree nodes. Quadtree traversal is a pointerless technique (called implicit quadtree) which represent image in the form of preorder traversal of the quadtree nodes. It is a listing of all the node colors in the order given by the preorder traversal (depth first traversal) of the tree, which is more efficient in storage saving and fast retrieving and transmitting image. In this paper a new compact quadtree traversal coding was introduced that based on merging between a quadtree traversal technique and run length code technique that is used in image representation.

1. INTRODUCTION

The main idea of this paper is to store a digital image by using a compact form of spatial data structure called implicit quadtree [1]. This form of quadtree is based on traversing pointer-based quadtree (i.e. explicit quadtree) in preorder form to create a sequence of node colors corresponding to each block of pixels or pixel of an image. This technique provides a good storage saving and fast retrieving for an image [2] [3]. To get more storage saving and make our image representation more compact the resulted quadtree treversal corresponding to an image is coded by using run length coding technique. The final representation for an image has effectively reducing memory spaces compared with traditional quadtree we call it compact quadtree traversal coding technique. The following steps describe this new technique:

Step 1: Converting colored bitmap image (.BMP file format) to pointer based quadtree by using an efficient quadtree largest node insertion technique.

Step 2: Traversing the explicit quadtree resulted from step 1 in preorder form (depth first) to get implicit compact quadtree called quadtree traversal.

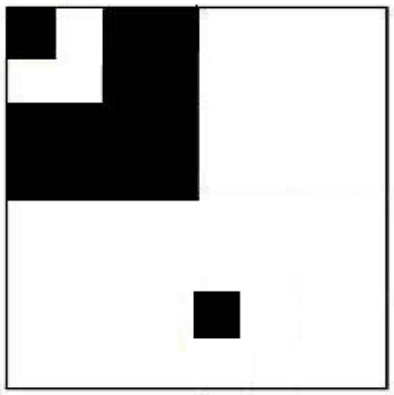
Step 3: coding the resulted quadtree traversal using run length code technique.

2. QUADTREE REPRESENTATION TECHNIQUE

The term quadtree is used in general sense to describe a class of data structures that based on the principle of recursive division of space. A region quadtree (often called quadtree) decomposes a 2^n x 2^n image that is not homogeneous into four equal quadrants. Each nonhomogeneous quadrant (block) is decomposed into subquadrants, and so on until each block is homogeneous (i.e., a block having same color). The subdivision process is represented by a tree of degree 4 [4].

In the quadtree representation of a binary image the root node corresponds to the entire image. Each child of a node represents a quadrant (labelled in order NW, NE, SW, and SE) of the region represented by that node. The leaf nodes of the tree correspond to those blocks for which no further subdivision is necessary. A leaf node is said to be a black or white depending on whether its corresponding block is entirely inside the represented region or outside the region [5][6]. All nonleaf nodes are said to be gray (i.e., blocks contain 0s and 1s). The root node is said to be at level 0 while a node level n corresponds to a

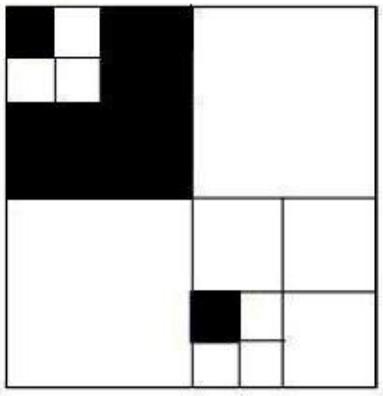
single pixel in the image. Figure 1 shows an example of an image and its corresponding quadtree.



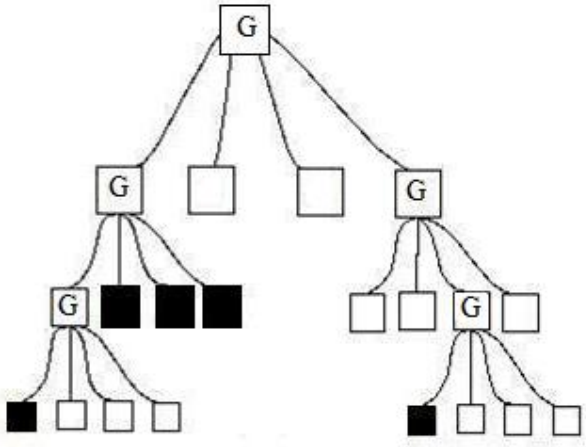
(a)

1	0	1	1	0	0	0	0
0	0	1	1	0	0	0	0
1	1	1	1	0	0	0	0
1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0

(b)



(c)



(d)

Figure 1. (a) region, (b) binary image, (c) block decomposition of region in (a),

(d) quadtree representation of the blocks in (c)

The quadtree representation is a variant on the maximal block representation. It requires the blocks to be disjoint to have standard size and standard location. The number of nodes N in a full quadtree for an image of size 2^n x 2^n that can be obtained in the worst case are considered in the following equation 1.1 [3]:

$$N=\sum_{i=0}^{n} 2^i \times 2^i \quad \text{.....1.1}$$

3. CONSTRUCTING A QUADTREE

An image usually exists as a sequential file where raw i proceeds raw i+1. Such representation is termed a raster representation, consisting of a list of the pixels ordered by raws. This representation is not practical for large images. So it is preferable to convert image to another representation that it takes less memory spaces by constructing a quadtree representation. In this paper we used a largest node insertion technique to build a pointer based quadtree (called explicit quadtree). This technique would in the worst case make a single insertion for each node in the output quadtree. Then using the number of insertion into output quadtree as a metric, this technique is optimal within a constant factor and faster that traditional technique to build a quadtree.

The optimal technique can be used to build a linear quadtree which is a pointer less representation directly from the raster representation. At any point while a quadtree is being constructed, there is a processed portion of the image and an unprocessed portion. Both the processed and the unprocessed portions of the quadtree have been assigned to the nodes. If it were possible to know the current value of all unprocessed pixels in the tree, then it would not be necessary to insert a pixel with color C for which a previous largest-node insertion has already set the containing node for that pixel to color C. The node is said to be active if at least one, but not all, pixel covered by the node has been processed. The largest node insertion process must keep track of all these active quadtree nodes. For a 2^n x 2^n image there are at most 2^n-1 nodes [7].

The following algorithm 1 for constructing a quadtree using largest node insertion technique:

Algorithm 1:

1) **For each pixel in the raster scan (bitmap file format) do the following:**
2) **If the pixel has the same color as appropriate active node, do nothing.**
3) **Otherwise,**
   - **Insert the largest possible node for which is the first (upper leftmost) pixel, and (if it is not a 1 x 1 pixel node) add it to the set of active nodes.**
4) **Remove any active nodes for which this is the last (low right) pixel.**

In this paper we modified algorithm 1 to include not only 2^n x 2^n image but also any other images of different sizes. This modification is based on expand an image to get nearest 2^n x 2^n size by adding invisible value -1 to every new pixels that make required size, for example an image of size 230 x 200 can be expanded by adding invisible and unsaved values to get image of size 256 x 256, this makes the recursive decomposition to four quadrants possible.

The additional new pixels are used only for decomposition but they are not saved and also not used to construct quadtree.

4. QUADTREE TRAVERSALS

It is important to reduce the space required by explicit quadtree by converting it to more compact form which is called quadtree traversal or implicit quadtree.

This representation is a pointer less representation that can be obtained by traversing the tree in preorder form (depth first) of the nodes. It is simply a listing of all nodes colors in the order given by the preorder traversal of the tree. If the color of each node is stored in 1 byte, then the list will require N bytes of internal storage. Where N is the total number of nodes in the tree [8].

For a binary image, the result is a string consisting of symbols 'G', 'B', 'W' corresponding to gray, black, and white nodes respectively. This depth-first expression of the quadtree can be constructed by traversing its children in order NW, NE, SW, SE. The original image can be reconstructed from depth-first expression by observing that the degree of each nonleaf node is always 4 [9]. The quadtree depth-first expression of the image in figure 1 is given by the following sequence:

G G G B W W W B B B W W G W W G B W W W W.

The following algorithm 2 presents the steps of constructing quadtree traversal representation in recursive way:

Algorithm 2:

1) If the node_color is gray then:
- **Make the current_color = gray.**
- **Traverse SW subtree.**
- **Trverse SE subtree.**
- **Trverse NW subtree.**
- **Trverse NE subtree.**

2) Else, make current_color = node_color.

5. RUN-LENGTH CODE

A compact representation of a binary image is the run-length code. In order to extract the run-length code, a binary image is scanned line by line. If a line contains a sequence of *p* equal pixels, we do not store *p* times the same pixel, but store the value of the pixel and indicate that it occurs *p* times. In this way, large uniform line segments can be stored in a very efficient way [10].

For binary images, the code can be especially efficient since we have only the two pixels values zero and one. Since a sequence of zeros is always followed by a sequence of ones, there is no need to store the pixel value. We only need to store the number of times a pixel value occurs. The following example of run-length code for gray scale image:

The line of a gray scale image:

12 12 12 20 20 20 20 25 27 25 20 20 20 20 20 20

Written in run-length code in hexadecimal numbers is

82 12 83 20 25 27 25 85 20

In this code, a sequence where the most significant bit is set (hexadecimal 80) indicates that the following number is to be repeated n − 80 + 1 times.

Run-length code is suitable for compact storage of images. It has become an integral part of several standard image formats, for example, the TGA or the TIFF formats [11].

6. COMPACT QUADTREE TRAVERSAL CODING

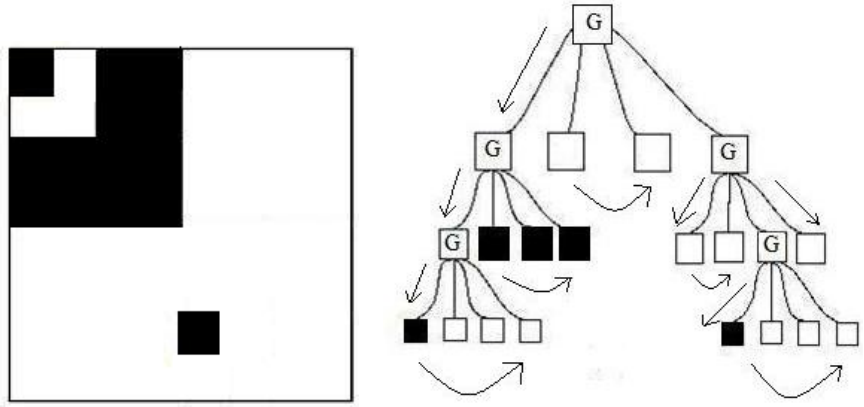
The main idea of compact quadtree traversal coding is to combine between quadtree traversal technique (depth-first expression) and run-length code to get more effective representation and much storage saving. In our representation an image (a bitmap image) is converted to another form which is pointer based quadtree (implicit quadtree). This pointer based tree can be effectively traversed in preorder form(depth-first) to get a sequence of only colors of quadtree's nodes that represent the image, this new compact representation reduce the amount of memory spacees required by image as well as spaces required by a pointer based quadtree. After getting the depth-first expression of colors for quadtree nodes, then it will be coded using run length code technique. This new technique make our representation of image more compact than qudtree traversal technique discussed in section 4 and traditional run-length code discussed in section 5, because we code the sequence of colors for quadtree nodes which they compact form but not coding the sequence of pixels of image itself like traditional technique. The following algorithm 3 shows the steps of our new representation called compact quadtree traversal coding technique:

Algorithm 3:

a. Scan the input bitmap image (.BMP file format) in order line by line.
b. Construct a pointer based quadtree of a bitmap image using optimal quadtree largest node insertion technique (algorithm 1).
c. Traverse the pointer based quadtree in preorder form recursively to get sequence of colors node or depth-first expression (algorithm 2).

d. Code the resulted sequence of colors using run length code technique.
e. Store the new image representation as .QTC file format (Quadtree Traversal Coding).

The following figure presents an example that shows the steps of compact quadtree traversal coding technique for an 8 x 8 image sample:



(a) **(b)**

G G G B W W W B B B W W G W W G B W W W W

(c)

82 G B 82 W 82 B 81 W G 81 W G B 83 W

(d)

Figure 2. (a) a bitmap image, (b) the pointer based quadtree corresponding to image in (a) , (c) quadtree traversal of the pointer based quadtree in (b), (d) run-length code for quadtree traversal in (c).

As shown form previous example in figure2, our compact quadtree traversal coding technique takes a number of symbols to represent image less than traditional qudtree traversal (depth-first expression) nodes and also less than the number of pixels for a bitmap image. So if we assume for example in figure 2 that each pixel in a bitmap image, each node in a quadtree, and each symbol in a run-length code takes 1 byte then the number of bytes for each representation are:

1) **64 bytes for a bitmap image.**
2) **21 byte for qudtree traversal nodes.**
3) **16 bytes for compact quadtree traversal code technique.**

In order to retrieve or reconstruct original image from compact quadtree traversal code the operation would be first rebuild a depth-first expression from run-length code of quadtree traversal ,second we can obtain original image directly from depth-first expression by processing this expression recursively in preorder form, without needing to reconstruction of pointer based quadtree to regain original image. This is very effective and useful to fast retrieving image.

7. EXPERIMENTAL RESULTS

A number of different 256 color and gray scale images shown in figure 3, figure 4, figure 5, and figure 6 are represented using three types of image format:

1) Bitmap image (.BMP file format).

2) Quadtree traversal image (.QTT file format).

3) Compact quadtree traversal coding image (.QTC file format).

There performances are compared in term of number of bytes used to store each image as shown in the following tables:

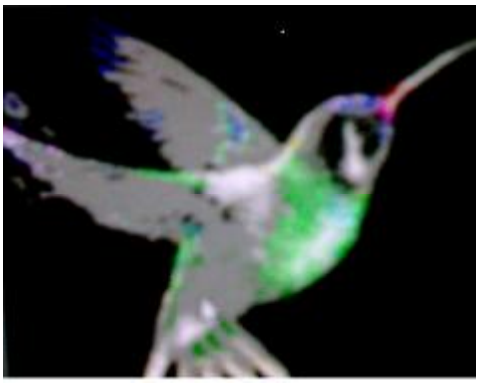


Figure 3. 256 x 232 image sample

Image Type	Number of Byte
.BMP	59392
.QTT	15155
.QTC	14261

Table 1 result of image in figure 3

Figure 4. 256 x 256 image sample

Image Type	Number of Byte
.BMP	65536
.QTT	11061
.QTC	9464

Table 2 result of image in figure 4

Figure 5. 512 x 400 gray scale image sample

Image Type	Number of Byte
.BMP	204800
.QTT	46391
.QTC	40640

Table 3 result of image in figure 5

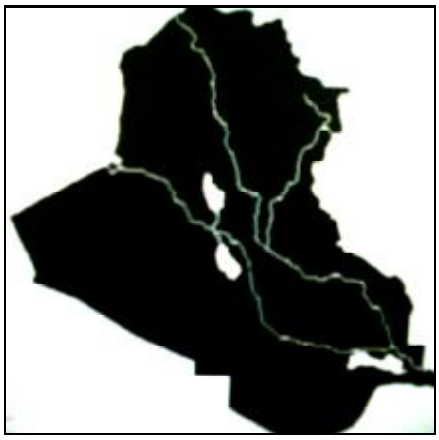


Figure 6. 486 x 472 gray scale image

Image Type	Number of Byte
.BMP	229392
.QTT	11143
.QTC	9460

Table 4 result of image in figure 6

From the above tables we can note that our compact quadtree traversal coding technique required a number of bytes less than a bitmap image and quadtree traversal representation ,so this new technique is more compact and make a good representation for an image that can be easily reconstruct from a quadtree traversal image.

Also we can note that the size of quadtree traversal image as well as the size of compact quadtree traversal coding image will not be effected (increased) when a bitmap image is scaled, as shown in the figure 7 and the result table 5, because quadtree structure depends on nodes corresponding to blocks, rather than depending on set of pixels as a bitmap image.

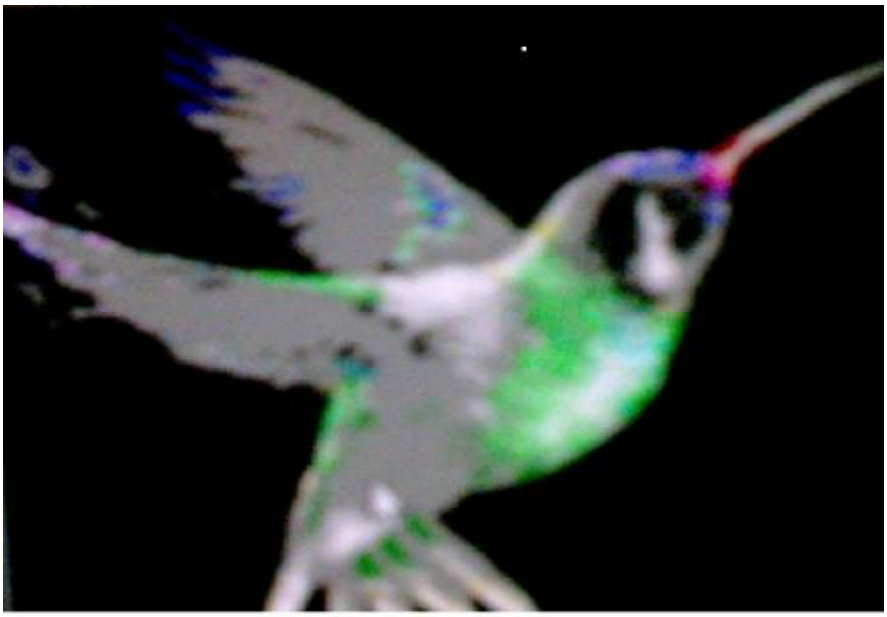


Figure 7. 512 x 464 scaled image in figure 3

Table 5 result of image in figure 7

Image Type	Number of Byte
.BMP	237568
.QTT	15155
.QTC	14261

8. CONCLUSIONS

We have proposed a simple but efficient technique called compact quadtree traversal coding to represent digital images, which is more compact representation than quadtree traversal and a bitmap representation of an image. This new representation that combines between pointer less quadree technique and run-length code is utilized from hierarchical features of quadtree which are very effective and useful in digital image representation and also utilized from the ability of run-length to make codes for sequence of colors that have identical values. Our new representation can be fast and easily reconstructed and retrieved to a bitmap image. This compact representation of an image can be applied in many applications like set theoretic operations such as union, intersection, and inverting. Also it can be used to implement linear image transformations like translation, rotation, and scaling in efficient way such us, scaling by a power of two, and rotation by a multiple of 90 degrees. For example, rotating an image using its quadtree representation by 90 degrees counter clockwise can be achieved by rearranging nodes, where all the pixels in the NW, NE, SE, and SW quadrants became in the SW, NW, NE, SE quadrants respectively.

9. REFERENCES

[1] H. Samet, "The design and analysis of spatial data structure", Addison-Wesly, 1990.

[2] H. Samet, "Applications of spatial data structures: computer graphics and image precessing, and GIS", Addison-Wesly, 1990.

[3] Y. Cohe, M. S. Landy, and M. Pavel, "Hierarchical coding for binary image", IEEE Transaction on Pattern Analysis and Machine Intelligence", vol. 7, no. 3, 1988, pp. 371-380.

[4] D. S. Scott and S. S. Lyengar, " A new data structure for efficient storing of images", Pattern Recognition Letters, vol. 3, no. 3, 1985, pp. 211-214.

[5] G. Klajnšek, B. Žalik, F. Novak, G. Papa, "A quadtree-based progressive lossless compression technique for volumetric data sets", *J. Inf. Sci. Eng.,* 2008.

[6] H. Samet and R. E. Webber, "hierarchical data structure and algorithms for computer graphics", Part 1. Fundamentals, IEEE Computer Graphics and Applications, vol. 8, no. 3, 1983, pp. 48-68.

[7] C. A. Shaffer and H. Samet, "Optimal quadtree construction algorithms", Computer Vision, Graphics, and Image Processing, vol.29, no. 5, 1985, pp 418-429.

[8] D. R. Fuhrmann, "Quadtree traversal algorithm for pointer-based and depth-first representation", IEEE Transaction on Pattern Analysis and Machine Intelligence, vol 10, no. 6, 1989, pp. 955-960.

[9] E. Kawaguchi, T, Endo, and M. Yakota, "DF-expression viewed from digital picture processing", IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 5, no. 4, 1983, pp. 373-384.

[10] Bermd Jahne, " Digitl image processing", Springer, 2005.

[11] Bermd Jahne, "Practical handbook for scientific and technical applications", Second addition, CRC Press, 2004.

تمثيل الصور بأستخدام تقنيـة ألانتقال على الشجرة الرباعية المدمجة والمرمزة

م.م. هيثم كريم عباس

كلية المنصور الجامعة

المستخلص :

تعتبر الشجرة الرباعية المهيكلة هي احدى أنواع هياكل البيانات الفضائية وتمثل واحدة من أهم طرق التمثيل الهرمي للصورة الرقمية التي تعمل على تقليل مساحة الخزن اللازمة للصورة عن طريق تجميع البيانات المتطابقة أو المتشابهة على شكل مربعات تعتبر بمثابة عقد على الشجرة الرباعية. الانتقال على الشجرة الرباعية هي احدى تقنيات الشجرة الرباعية عديمة المؤشر (تسمى بالشجرة الرباعية الضمنية) ويكون تمثيل الصورة فيها على شكل أنتقال متقدم (preorder) لعقد الاشجارالرباعية ويسمى كذلك الانتقال الاول العميق (depth- first) والذي يتم تمثيل الصورة فيه على شكل سلسلة من العقد الملونة بطريقة الانتقال المتقدم وهذه الطريقة تعتبر أكثر كفاءة في عملية تقليل حجم الذاكرة المخصص للصورة. وكذلك أسرع في عملية استرجاع وأرسال الصورة. في هذا البحث تم تقديم طريقة جديدة هي استخدام تقنية الانتقال على الشجرة الرباعية المدمجة والمشفرة, وذلك بالاعتماد على المزج بين طريقة الانتقال على الاشجار الرباعية وطريقة التمثيل الطولي للصورة المستخدم في تمثيل الصور.