Contents lists available at www.innovativejournal.in

**Asian Journal of Computer Science and Information Technology**

Journal homepage:http://www.innovativejournal.in/index.php/ajcsit

ISSN 2249 - 5126

# IMAGE COMPRESSION USING AREA SUB-DIVISION ALGORITHM RELYING ON QUADTREE

**M Anand Kumar**[1], D Demudubabu [2], Ch Mohan Babu [3]

*[1, 3]LINGAYA'S Institute of Management and Technology, Vijayawada, India.*

*[2]V.K.R & A.N.B & A.G.K College Of Engineering & Technology, Gudiwada, India.*

Contents lists available at www.innovativejournal.in

Asian Journal of Computer Science and Information Technology

Journal homepage:http://www.innovativejournal.in/index.php/ajcsit

ISSN 2249 - 5126

# IMAGE COMPRESSION USING AREA SUB-DIVISION ALGORITHM RELYING ON QUADTREE

**M Anand Kumar**[1], D Demudubabu[2], Ch Mohan Babu[3]

*[1, 3]LINGAYA'S Institute of Management and Technology, Vijayawada, India.*

*[2]V.K.R & A.N.B & A.G.K College Of Engineering & Technology, Gudiwada, India.*

**ARTICLE INFO**

**Corresponding Author:**
M Anand Kumar
LINGAYA'S Institute of Management and Technology, Vijayawada. India.

**KeyWords:** Quadtree, Image Compression, Image Processing.

**ABSTRACT**

This paper presents an image compression algorithm that has the ability to divide the original grey level image into unoverlapped blocks depending on a threshold value. The proposed algorithm is based on quadtree. It can compress and decompress the image in easy way using two stacks instead of tree. In the compression process, the proposed technique stores the information of all blocks, for instance the upper left coordinate, size, minimum, and difference values in a stack, and the divided blocks are numbered in effective way. This information will be used to decompress the image again. It was found that the algorithm provides a high compression ratio ranged between 0.12 and 0.68.

## INTRODUCTION

A compression algorithm finds redundancy in data and removes detail too minute to be detected by the human eye. The internet, digital library, multimedia publishing service, geographical information system, computer aided design and medical image archiving systems, image processing are widely used. However, since image data takes enormous storage space, how to store and retrieve an image, both economically and effectively, has a high priority in current research efforts. Therefore, the necessity of efficient data compression is increasing. A compression algorithm finds redundancy in data and removes detail too minute to be detected by the human eye. For example, in an uncompressed image file, a shade slightly different, unnoticeable by the human eye, is numerically significant, and extra storage is allocated for this unnoticeable feature.

Data compression provides two advantages: reducing storage space and transmission time by finding the humanly imperceptible differences. Data compression techniques may be applied to digital image data. Compressed files are stored more efficiently and transmitted more quickly. The degree of data compaction is expressed as the compression ratio; that is, a ratio of the original digital file size to the new compressed file size. Many techniques for data compression have been developed. For any individual compression technique applied to a given image, a fixed compression ratio may be achieved without loss of data (lossless compression). Higher compression ratios may be obtained when data loss is permitted (lossy compression). Greater levels of lossy compression generally imply inferior image quality. A major advantage of the quad tree technique for data compression is the simplicity of its approach. Unlike many other compression techniques, a quad tree algorithm can compress images relatively quickly on a personal computer. We were particularly interested in the potential identification of a threshold compression ratio at which there would be no significant loss of diagnostic accuracy.

The most widely used multimedia data are two dimensional images; hence we focus on these data. There are many compression techniques in use today; these techniques include those that use mathematical transforms such as Discrete Cosine Transform (DCT) transform, wavelet transform, fractal, and quad tree techniques.

Most of the aforementioned techniques tend to be mathematically complex, except for the quad tree algorithm. The quad trees are powerful and simple data structures for representing compressing digital images. It is possible to find applications of quad tree decomposition in many different contexts, such as the compression of sub-band coefficients in wavelet decomposition and coding of the sub-blocks data in EBCOT algorithm. The quadtree decomposition is an important tool for fractal image compression where many suitable smart variants of quadtrees are used and applied.

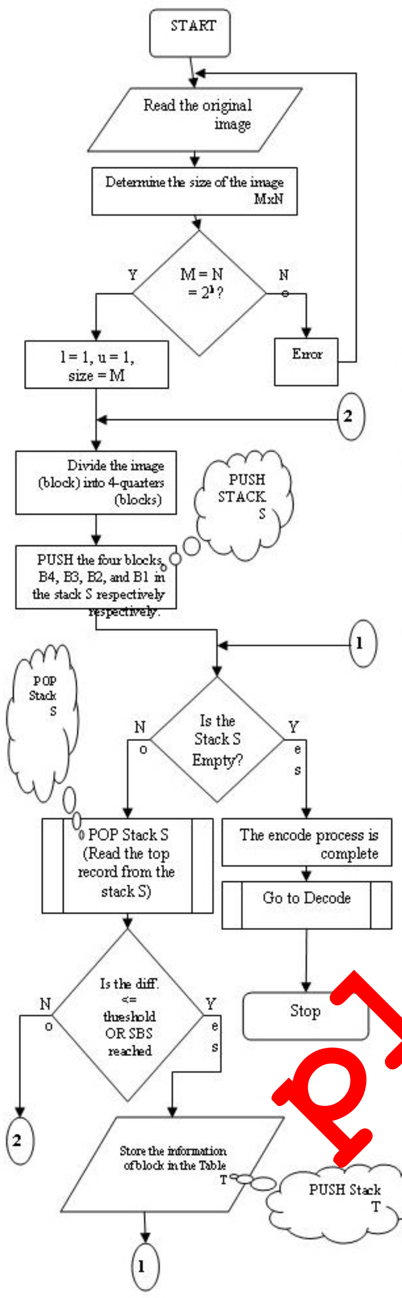
The quadtree algorithms are based on simple averages and comparisons. A quadtree is a tree-like data structure where each node either terminates on a leaf containing useful information, or branches into four sub-level quadtrees.

### Proposed Algorithm

This algorithm is illustrated in Figure. The new idea in this algorithm, two stacks are used instead of tree that used in the quadtree algorithm, they are designed to restore compressed images effectively in easy way. These stacks are called S and T. For applying the algorithm, the whole image should be square of size (2k); k is a positive integer number. At the first, divide the image into 4-quarters (blocks) as follows:

r = l + size -1;

d = u + size -1,



START
Read the original image
Determine the size of the image MxN
M = N = 2k ?
Y
N
l = 1, u = 1, size = M
Error
2
Divide the image (block) into 4-quarters (blocks)
PUSH STACK S
PUSH the four blocks, B4, B3, B2, and B1 in the stack S respectively
1
POP Stack S
Is the Stack S Empty?
No
Yes
POP Stack S (Read the top record from the stack S)
The encode process is complete
Go to Decode
Is the diff. <= threshold OR SBS reached
No
Yes
Stop
2
Store the information of block in the Table T
PUSH Stack T
1

Figure (1): The proposed algorithm diagram

Information of the block North West (NW) is:

$$\left(1,\left\lfloor\frac{u+d}{2}\right\rfloor+1\right),\left(\left\lfloor\frac{1+r}{2}\right\rfloor,d\right)$$

size = size/4, min = Minimum, diff = max-min, and encoding = 'B1', PUSH in the stack

S, information of the block South West (SW) is:

$\left(\left\lfloor\frac{1+r}{2}\right\rfloor+1,\left\lfloor\frac{u+d}{2}\right\rfloor+1\right),\left(r,d\right)$ l r 1, u d 1 , (r,d)

size = size/4, min = Minimum, diff = max - min, and encoding = 'B2', PUSH in the stack S, information of the block North East (NE) is:

$$\left(1,u\right),\left(\left\lfloor\frac{1+r}{2}\right\rfloor,\left\lfloor\frac{u+d}{2}\right\rfloor\right)$$

size = size/4, min = Minimum, diff = max-min, and encoding = 'B3', PUSH in the stack S, information of the block South East (SE) is:

$$\left(\left\lfloor\frac{1+r}{2}\right\rfloor+1,u\right),\left(r,\left\lfloor\frac{u+d}{2}\right\rfloor\right)$$

size = size/4, min = Minimum, diff = max-min, and encoding = 'B4', PUSH in the stack S. Where, (l,u) is the upper-left corner coordinates and (r,d) is bottom right corner coordinates for any block. Then, the stack S is popped and a threshold condition is applied on the popped block.

If this condition is satisfied (diff. <= Thresh. OR the supposed smallest block size (SBS) reached), the popped block is pushed in stack T, otherwise, the popped block is divided into 4-subblocks and they are pushed in the stack S. This process is continued until the encoding image is completed, then, the stack S is empty and the stack T contains the encoding image.

**Encoding Image**

In order to divide the original image, two stacks are created. The stacks consist of five fields. These fields contain encode blocks, the coordinates of each block (top east corner), block size, and the difference value between minimum and maximum of each block respectively. The original image is firstly divided into four equal blocks. The coordinates of these blocks are determined. These blocks are in the directions NE, NW, SE, and SW. They are coded B1, B2, B3, and B4 respectively and they are pushed in a stack S. Then, the stack S is popped (B1 is last element in the Stack S).

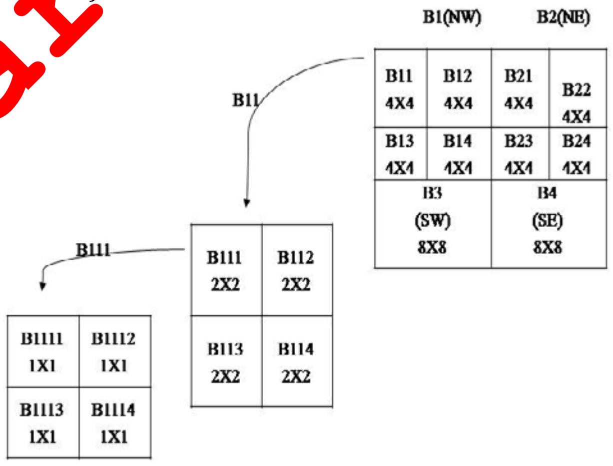


**Diagram of the segmented image**

The popped block (B1) is checked, if the difference value less than the threshold, then the required information of this block is stored in the stack T. Otherwise, the popped block is divided into four sub-blocks and pushed in the stack S. They are coded B11, B12, B13, and B14 respectively. After that, the stack S is popped (B11) is last element in the Stack S). The popped block (B11) is checked, if the difference value less than the threshold, then the required information of this block is stored in a stack T. Otherwise, the popped block is divided into four sub-blocks and pushed in the stack S. They are coded B111, B112, B113, and B114 respectively and so on.

The division process is continuous for the other blocks, if the difference value is greater than or equal to the threshold. The division process is stopped for any block, if the difference value less than the threshold or the smallest block size is reached; see Figure and Table. At the end of the process of encoding image, the stack S is empty and the stack T contains encoding image blocks. The divided blocks are numbered using the above coding technique that helps to determine these blocks correctly and easily.

Table: The stack T containing the classified information from Figure.

| Encode Block | Top-left Coordinates (x, y) | Size | Minimum | Difference |
|---|---|---|---|---|
| B4 | (9,9) | 8 | | |
| B3 | (9,1) | 8 | | |
| B24 | (3,13) | 4 | | |
| B23 | (5,9) | 4 | | |
| B22 | (1,13) | 4 | | |
| B21 | (1,9) | 4 | | |
| B14 | (5,5) | 4 | | |
| B13 | (5,1) | 4 | | |
| B12 | (1,5) | 4 | | |
| B114 | (3,3) | 2 | | |
| B113 | (1,4) | 2 | | |
| B112 | (1,3) | 2 | | |
| B1114 | (2,2) | 1 | | |
| B1113 | (2,1) | 1 | | |
| B1112 | (1,2) | 1 | | |
| B1111 | (1,1) | 1 | | |

**Example**

This example is proposed to describe the techniques of the designed algorithm. The example contains an image of size 16x16. Depending on the aforementioned details in the Section 2.1, the image is divided into 4-blocks. The information of the 4- blocks is described as follows:

Encoded blocks are B1, B2, B3, and B4,
coordinates are: (1,1), (1,9), (9,1), and (9,9),
Minimum are: 0, 25, 0, and 10,
Differences are: 102, 77, 255, and 7,
Size of all blocks is 8x8;

These blocks are pushed in the stack S. The stack S is popped (get the block B1), and the threshold condition is not satisfied for this block. Therefore, it is divided into 4-subblocks and they are pushed in the stack S. Then the stack S is popped (get the block B11), and the threshold condition is not satisfied for this block. Therefore, it is divided into 4-subblocks and they are pushed in the stack S. After that, the stack S is popped (get the block B111), and the threshold condition is not satisfied for this block. Therefore, it is divided into 4-subblocks and they are pushed in the stack S. Again, the stack S is popped (get the block B1111), and the threshold condition is satisfied for popped block. Therefore, this block is pushed in the stack T. By the same way, the following blocks, B1112, B1113, B1114, B112, B113, B114, B12, B13, and B14 are popped from the stack S and they are satisfied the threshold condition, therefore, they are pushed in the stack T respectively. Now, the stack S contains the following blocks, B2, B3, and B4. Then, the stack S is popped (get the block B2), and the threshold condition is not satisfied for this block. Therefore, it is divided into 4- sub blocks and they are pushed in the stack S. By the same way, the other processes are executed until the stack S is empty, and the stack T has the encoding image.

**Decoding image**

In order to decode image, the proposed two stacks are used as a replacement for tree, and the divided blocks are numbered in effective way to determine these blocks correctly.

This is designed to restore compressed images again in easy way. The image is decoded using the following technique:

```
1. POP the stack T
2. pixel-value = minimum + diff.
3. for i = l : l + size
4. for j = u : u + size
5. decoded-image(i, j) = pixel-value
6. end
7. end
8. if the stack T is empty then (the decoding image is restored completely)
9. else go to step1
```

| Encode Block | Top-left Coordinates | Size | Minimum | Difference |
|---|---|---|---|---|
| B4 | (9,9) | 8 | 10 | 7 |
| B34 | (13,5) | 4 | 192 | 7 |
| B33 | (13, 1) | 4 | 248 | 6 |
| B32 | (9, 5 | 4 | 93 | 7 |
| B31 | (9, 1) | 4 | 0 | 7 |
| B24 | (5, 13) | 4 | 50 | 7 |
| B234 | (7,11) | 2 | 98 | 4 |
| B2334 | (8, 10) | 1 | 94 | 0 |
| B2333 | (8, 9) | 1 | 100 | 0 |
| B2332 | (7, 10) | 1 | 96 | 0 |
| B2331 | (7, 9) | 1 | 95 | 0 |
| B232 | (5, 11) | 2 | 25 | 2 |
| B231 | (5, 9) | 2 | 25 | 5 |
| B22 | (1,13) | 4 | 70 | 7 |
| B21 | (1,9) | 4 | 60 | 7 |
| B14 | (5,5) | 4 | 70 | 7 |
| B13 | (5,1) | 4 | 80 | 5 |
| B12 | (1,5) | 4 | 95 | 7 |
| B114 | (3,3) | 2 | 45 | 5 |
| B113 | (1,4) | 2 | 30 | 7 |
| B112 | (1,3) | 2 | 18 | 6 |
| B1114 | (2,2) | 1 | 15 | 0 |
| B1113 | (2,1) | 1 | 8 | 0 |
| B1112 | (1,2) | 1 | 10 | 0 |
| B1111 | (1,1) | 1 | 0 | 0 |

The pushed blocks in the stack T (The compressed image).

## CONCLUSION

This paper presents an efficient algorithm that has the ability to compress and decompress grey level images in easy way. The proposed algorithm is based on quadtree. Two stacks are used during the process of dividing the original image into blocks depending on a threshold value. These stacks are used as an alternative of tree, and the divided blocks are numbered effectively to determine these blocks correctly. This is designed to restore compressed images again in easy way quickly. The first stack (S) contains the specified bocks information that resulted from blocks division. These blocks may be divided again according to the threshold value. The other stack (T) is used to keep the information of all blocks (to decompress the image) that satisfy the threshold condition or when the smallest block size is reached. This means that the block is not divided again. After the division process is completed, the stack S must be empty, while, the stack T must contain the all divided blocks. The compression ratio is obtained by dividing the size of the compressed output file by the size of the original image file. From the application, it was found that the compression ratio is ranged between 0.12 and 0.68. The compression ratios are dependent on the threshold values, which can be affected whether quality or compression.

## REFERENCES

[1] John Arthur Porche, "An architecture for quad-tree based image compression", (M.Sc. Thesis, North Carolina A&T State University), 2002.

[2] Yung-Kuan Chan, Chin-Chen Chang, "Bloch image retrieval based on a compressed linear quadtree", Image and Vision Computing, 22(5): 391-397, 2004

[3] Hwang-Soo Kim , Jae-Young Lee, "Image coding by fitting RBF-surfaces to subimages", Pattern Recognition Letters, 23: 1239–1251, 2002.

[4] Ethan J. Halpemn, Jeffrey H. Newhouse, E. Stephen Amis, Howard M. Levy, and Herman W. Lubetsky, "Evaluation of a Quadtree-based Compression Algorithm with Digitized Urograms", Radiology, 171: 259-263, 1989.

[5] D.J. Duh, J.H. Jeng and S.Y. Chen, "DCT based simple classification scheme for fractal image compression", Image and Vision Computing, 23(13): 1115-1121, 2005.

ICGST-GVIP, ISSN 1687-398X, Volume (8), Issue (III), October 2008

www.icgst.com

# Qualitative Image Compression Algorithm Relying on Quadtree

A.A. El-Harby and G.M. Behery
*Mansoura University, Faculty of Science, Math. Dept., New Damietta, Egypt.*
elharby@yahoo.co.uk

## Abstract

This paper presents an image compression algorithm that has the ability to divide the original grey level image into unoverlapped blocks depending on a threshold value. The proposed algorithm is based on quadtree. It can compress and decompress the image in easy way using two stacks instead of tree. In the compression process, the proposed technique stores the information of all blocks, for instance the upper left coordinate, size, minimum, and difference values in a stack, and the divided blocks are numbered in effective way. This information will be used to decompress the image again. It was found that the algorithm provides a high compression ratio ranged between 0.12 and 0.68.

***Keywords:*** *quadtree, image compression, image processing.*

## 1. Introduction

A compression algorithm finds redundancy in data and removes detail too minute to be detected by the human eye. For example, in an uncompressed image file, a shade slightly different, unnoticeable by the human eye, is numerically significant, and extra storage is allocated for this unnoticeable feature [1].
The internet, digital library, multimedia publishing service, geographical information system, computer-aided design and medical image archiving systems, image processing are widely used. However, since image data takes enormous storage space, how to store and retrieve an image, both economically and effectively, has a high priority in current research efforts [2]. Therefore, the necessity of efficient data compression is increasing. A compression algorithm finds redundancy in data and removes detail too minute to be detected by the human eye. For example, in an uncompressed image file, a shade slightly different, unnoticeable by the human eye, is numerically significant, and extra storage is allocated for this unnoticeable feature [1].
Data compression provides two advantages: reducing storage space and transmission time by finding the humanly imperceptible differences [3]. Data-compression techniques may be applied to digital image data. Compressed files are stored more efficiently and transmitted more quickly. The degree of data compaction is expressed as the compression ratio; that is, a ratio of the original digital file size to the new compressed file size. Many techniques for data compression have been developed. For any individual compression technique applied to a given image, a fixed compression ratio may be achieved without loss of data (lossless compression). Higher compression ratios may be obtained when data loss is permitted (lossy compression). Greater levels of lossy compression generally imply inferior image quality. A major advantage of the quadtree technique for data compression is the simplicity of its approach. Unlike many other compression techniques, a quadtnee algorithm can compress images relatively quickly on a personal computer. We were particularly interested in the potential identification of a threshold compression ratio at which there would be no significant loss of diagnostic accuracy [4].
The most widely used multimedia data are two-dimensional images; hence we focus on these data. There are many compression techniques in use today; these techniques include those that use mathematical transforms such as Discrete Cosine Transform (DCT) transform [5-8], wavelet transform [9-12], fractal [13-16], and quadtree techniques [17-19].
Most of the aforementioned techniques tend to be mathematically complex, except for the quadtree algorithm. The quadtrees are powerful and simple data structures for representing compressing digital images. It is possible to find applications of quadtree decomposition in many different contexts, such as the compression of sub-band coefficients in wavelet decomposition and coding of the sub-blocks data in EBCOT algorithm. The quadtree decomposition is an important tool for fractal image compression where many suitable smart variants of quadtrees are used and applied [20].
The quadtree algorithms are based on simple averages and comparisons. A quadtree is a tree-like data structure where each node either terminates on a leaf containing useful information, or branches into four sub-level quadtrees [21].