PARTIAL IMAGE RETRIEVAL SYSTEM USING SUB TREE MATCHING

P. ANANDHAKUMAR, M. BOOPATHI RAJA and A.KANNAN.
Department of Computer Science and Engineering, Anna University,
Chennai – 600 025, India.

Abstract:

A novel approach is proposed for quick and accurate partial image retrieval from a large number of images based on tree structure of images. Retrieving the partial similar images generally requires a huge amount of storage space for indexes due to the large number of portions of images. The proposed method constructs the ternary tree, which greatly reduces the storage space for database images and fast retrieval. The computation of tree structure is not a computationally intensive task and tree comparison won't take much time to find the best match image. The system is capable of accepting any size of query image decided by user and retrieves the whole images from the database. The retrieval time is of the order of seconds ranging from 42 milliseconds to 10 seconds and translation independent.

Keywords: Ternary Tree, Partial Image Retrieval, CBIR, Sub Tree Matching, Image Databases.

1.0 Introduction

Now a day, digital images are available in a huge number and these images must be easily retrieved as per the users query image. Hence techniques for quick image retrieval are strongly needed. A common approach is based on key words. However, providing a keyword for each image is a difficult task for computers and time-consuming for humans. So, a best image retrieval algorithm should retrieve images as per the users query image exactly and effectively. In some situations, whole image is not available with the user and the user is having only the partial images. In those situations, query image is a partial one and the image retrieval algorithm should retrieve the whole image.

To retrieve the exact or similar image for the partial query image, the database images should be represented in a manner, which is effective for finding the whole images for the partial queries. The representation method should be less computationally intensive, it should occupy the less memory and it should take less time to find the exact or similar images from the database. The size of the query image will vary and the user decides it, so the system should be able to handle varying sizes of query image. To consider these issues ternary tree structure is developed for database images. It takes less computation time, less storage space, less searching time and varying sizes of query image.

The proposed method avoids the calculation of rough shapes and positions as queries, which is used in the VisualSEEK, calculation of quantitative features such as colors, shapes, and textures, and region based approaches. However, the existing methods (VisualSEEK, QBIC, Region Based approaches) are not always accurate in identifying objects, and therefore most of them use domain-specific constraints to improve the accuracy. Here, we consider a domain-independent approach, which constructs tree structures for retrieving the whole image for partial query image.

The remainder of this paper is organized as follows; section 2 gives a survey of related works. Section 3 gives the details about image representation using ternary trees. Section 4 explains the details about the whole image retrieval for partial

image queries. Section 5 discusses the results and Section 6 gives the conclusion.

2. Literature Survey

Acceleration of similarity-based partial image retrieval using multistage vector quantization proposed by Akisato Kimura, Takahito Kawanishi and Kunio Kashino for partial image retrieval from a huge number of images based on a predefined distance measure. Their method utilizes vector quantization (VQ) on multiple layers, namely color, block, and feature layers. But, in our work a single tree is constructed for an image, which takes less computation than the calculation of multiple layers. [1].

The Hex-Splines [2] proposed by Dimitri Van De Ville, et al describe a new family of bivariate, non-separable splines, called hex-splines, especially designed for hexagonal lattices. In our system, we use hexagonal lattices for image representations.

Symmetric Region Growing approach [3] proposed by Shu-Yen Wan and William E. Higgins describes the Symmetric Region Growing in which region growing has been focused primarily on the design of feature measures and on growing and merging criteria. These methods have an inherent dependence on the order in which the points and regions are examined. In our system, symmetric region growing is achieved using hexagons of uniform size, which grows in all directions.

Akisato Kimura, Takahito Kawanishi and Kunio Kashino propose a new framework [4] for quick and accurate partial image retrieval from a huge number of images based on a predefined distance measure. Their method extracts portions from each database image at a constant spacing, while it extracts all possible portions from a query image. But, our system constructs a tree structure for each image and it is used for retrieving the whole image for partial queries fastly.

The technique "Hexagonal Fast Fourier Transform with Rectangular Output" by J.C.Ehrhardt [5] explains that hexagonal sampling is the most efficient sampling pattern for a two dimensional circularly band limited function. In our system, we use a ternary tree for indexing images in image databases.

Fabio Dell'Acqua and Paolo Gamba [6] describe the application of a simplified shape analysis technique based on a modal representation of the object shape, which is useful for improving the efficiency and effectiveness of shape-driven searches in image databases.

Injong Rhee, et al proposes the technique "Quad tree - Structured Variable-Size Block-Matching Motion Estimation with Minimal Error" [7]. This paper explains two efficient quad tree-based algorithms for variable-size block matching motion estimation. The first algorithm employs an efficient dynamic programming technique utilizing the special structure of a quad tree. The second algorithm adopts a heuristic way to select variable-sized square blocks. It relies more on local motion information than on global error optimization. The details about quad tree is well explained [7] which leads to the development of Ternary tree in our work.

"Hexagonal Image Sampling" proposed by R.C. Staunton [8] explains about the use of regular hexagonal sampling systems in robot vision applications.

"Edge Detection in a Hexagonal-Image Processing Framework" approach is proposed by. L. Middleton and J. Sivaswamy [9]. This paper explains about edge detection in the context of hexagonally sampled images. The result shows that the computational requirement for hexagonal processing is less than that for square sampled images. It also discusses about edge detection using hexagonal sampling.

R.C. Staunton proposes the technique called "One Pass Parallel Hexagonal Thinning Algorithm" [10]. The author presents the comparison between two fully parallel thinning algorithms designed for images sampled on square and hexagonal grids. The comparison of square and hexagonal sampling is useful for our

Ternary tree construction. Comparing with all the works provided in the literature, our ternary tree representation is different in many aspects like reconstruction of image from tree, no overlapping of regions and constant node size. This tree structure helps in achieving fast retrieval of images from image databases, Object tracking, Object hiding.

3.0 Image Representation Using Ternary Tree

In this section, representing an image using ternary trees, retrieving an exact image from database using ternary trees has been explained. Fig.1. shows block diagram of image representation using ternary tree. It takes an image as input and outputs ternary tree. In image preprocessing phase, image is smoothened using low pass filter. Then the image is discretized into hexagonal tiles in the next phase. For these, hexagonal tiles, average value are found using the R, G, and B values of the pixels. These values are transformed into array structure. For this array structure ternary tree is constructed and it is given as the output.

To smooth the image, low pass filter is applied; the value of the center pixel is given by,

'e'= (a+b+c+d+e+f+g+h+i) / 9 → (3.1)

Where, the variables a, b, c, d, e, f, g, h and i are pixel values of 3 x 3 image.

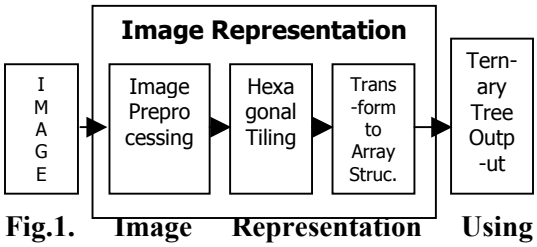

Fig.1. Image Representation Using Ternary Tree

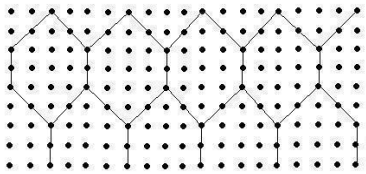

Fig.2. Hexagonal Lattice of single Hexagon

3.1 Hexagonal Sampling

In hexagonal sampling, the input image is tiled into minimum size of hexagons. To lay a hexagon,1,3,5,5,5,3,1 pixel pattern is used i.e., for the hexagon 1 in the first row, first row's third pixel, second row's first, second and third pixel, third row's first, second, third, fourth and fifth pixel, fourth row's first, second, third, fourth and fifth pixel, fifth row's first, second, third, fourth and fifth pixel, sixth row's first, second and third pixel, seventh row's third pixel are taken. For a sample image of 19x9 the hexagonal lattice is given in Fig.2. Where 1 is first pixel, 2 is second pixel and so on.

Number of hexagons in the image can be found using the following formula.

Number of hexagons in the row order,

x = (((Image height/ (Hexagon height+1))*2)-1) → (3.2)

Number of hexagons in the column order,

y= ((Image width/ (Hexagon width-1))-1) → (3.3)

Total number of hexagons in the image=x*y → (3.4)

3.2 Ternary Tree Formation

For each and every image in the database Ternary tree is formed. From the average values, which are stored in the array structure, the algorithm builds the Ternary tree. This is shown in Fig.3. Algorithm first forms the two-dimensional array (shown in Fig.4.). For example, let the value be 25.To find the number of rows and columns, square root of the value is found. (i.e. 5 for this example). So the array size will be 5x5. If the value is 120, then the square root is 10.For the remaining 20 nodes 2 rows are added. So the array size is 12x10. In some cases for the non-squared number, some array elements may by kept vacant. In those situations, they will be marked as null nodes. The first element of the array is taken as root node and its adjacent and connected three nodes are taken as child nodes. The algorithm prints the root and child nodes recursively as shown in the Fig.3.

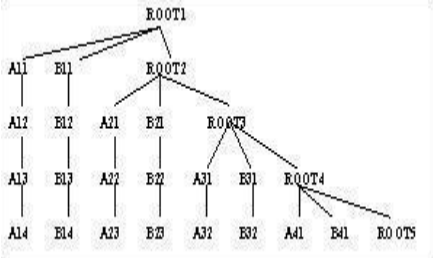

Fig.3. Ternary Tree

Root 1	A11	A12	A13	A14
B11	Root 2	A21	A22	A23
B12	B21	Root 3	A31	A32
B13	B22	B31	Root 4	A41
B14	B23	B32	B41	Root 5

Fig.4. Array Structure

3.3 Ternary Tree Construction Algorithm

Input: Image from the database.
Output: Ternary tree.
Step 1: *Read the image from database and grab the pixel values of the input image.*
Step 2: *Smooth the image using low pass filter as given in the equation (3.1).*
Step 3: *Tile hexagons in 1,3,5,5,5,3,1 patterns, as shown in the Fig.2.*
Step 4: *Find the(R, G and B) average value for each hexagon.*
Step 5: *Transfer these values to array structure.*
Step 6a: *Take the first element of the array as root and its connected three nodes are printed as child nodes.*
Step 6b: *Print the child nodes recursively until all elements in the array is printed recursively.*

4.0 Whole Image Retrieval From Database

The database images are represented by ternary trees and stored in the database. When the query image is given as input, the ternary tree is constructed for that image. To retrieve the whole image for the partial query image, the system compares the nodes of the query image and database images. The database image is tiled as shown in Fig.2. Let the partial query image starts at pixel (3,3). For this query image, hexagonal tiling is similarly applied. The first hexagon in the database image starts at x position 1 and ends at x position 5, y position 1 and ends at y position 7. For query image, starts at x position 3 and ends at x position 8, y position 3 and ends at y position 9. The mismatch starts here, to avoid this thresholding is applied.

To retrieve the whole image, partial image's ternary tree is compared with the database image's ternary tree. In comparison process, the query image's ternary tree is considered as a subtree. This sub tree is moved across database image's tree and number of matching nodes are found. If the number of matching nodes are with in the threshold then that image is retrieved and displayed to the user. So the system is also able to find the similar images. This proposed system is able to find the exact and similar images from the database for the partial query image. If the query image's tree, doesn't match with any images in the

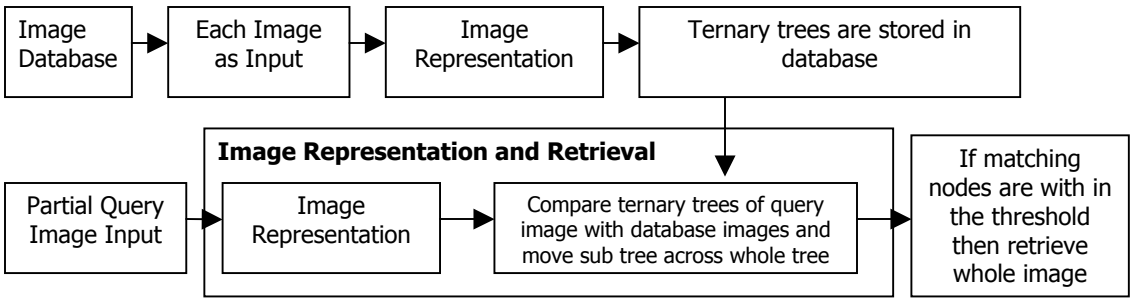

Fig.5. Whole Image Retrieval for partial image queries

database then system displays that "image not found".

4.1 Whole Image Retrieval Algorithm For Partial Image Queries

Input: Partial query image.
Output: Whole images from the database.
Step 1: *Get the image from image database.*
Step 2: *Using ternary tree construction algorithm, construct the tree for each image in the database.*
Step 3: *Store this Ternary tree in database.*
Step 4: *Get the query image from user.*
Step 5: *Using ternary tree building algorithm, construct the tree for the query image.*
Step 6: *Compare the query image's ternary tree with the database image's ternary tree.*
Step 7: *Move the query images ternary tree across database image ternary tree.*
Step 8: *Count the number of nodes that match with the database images.*
Step 9: *If the numbers of nodes matching are with in the threshold then that image is retrieved and displayed to the user. Otherwise output as "image not found".*
Step 10: *Repeat the process for all images in the database.*

5 Results and Discussions

The entire system is implemented in Java and 100 images are considered for testing the system. The size of the image is 352x288. These images are stored in Oracle database. The whole image is retrieved as per the user's partial query image. Compared to other systems [1][4], our system is capable of accepting any size of query image and retrieves the exact and similar images from the database. The retrieval time is of the order of milli seconds ranging from 10 milliseconds to 9 seconds, when it is executed in 3Ghz Pentium IV Processor with 1GB RAM. Results are shown in Fig.6. where above image is the partial query image and below image is the retrieved image.

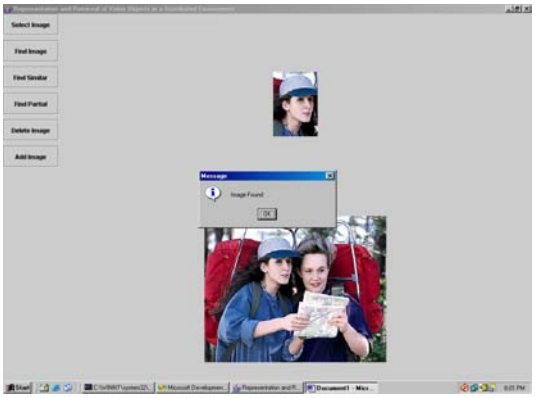

Fig.6. Partial Image Retrieval

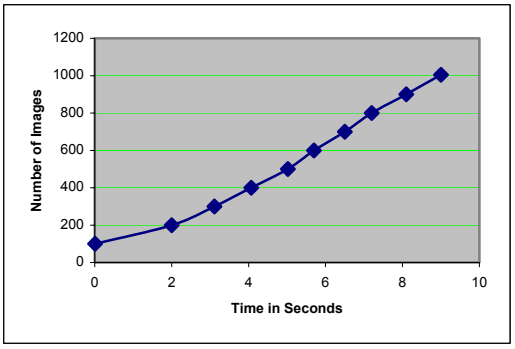

Fig.7. Graph between Number of images Vs Retrieval time in Seconds

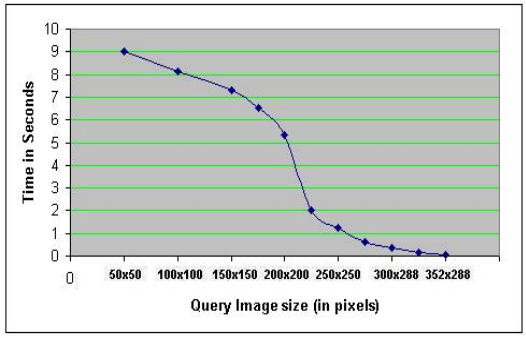

Fig.8. Graph between Retrieval time in Seconds Vs Query image size in pixels

Fig.7. shows the graph between number of images in the database and retrieval time in seconds. This graph shows that the retrieval time increases with the number of images. Using multithreading techniques for searching the tree will reduce the retrieval time. Fig.8. shows the graph between retrieval time in seconds and size of partial query image in pixels. This graph shows that the retrieval time decreases with increase in query image size.

6 Conclusion

In this work, tree structure based retrieval of whole image for the partial query image is designed and implemented. The system is capable of handling the partial image of any size as it's input and identifies the correct image independent of translation. A Ternary tree based data structure is used, which consumes less storage space and fast retrieval of whole images from the database. Further works in this direction could be the inclusion of hexagonal trees for effective storage and retrieval of images and reducing the retrieval time.

Sensitivity analysis for similar image retrieval using partial image queries:

Sensitivity analysis is carried out with 1008 images and 120 partial queries where, 70 queries for image available case and 50 for non-available case. The system responded well for all queries, since false positive as well as false negative decisions are nil. From our observations, we provide the results for sensitivity analysis in Table.1.

Table 1 Sensitivity table

Parameter	Actual observation	Observation in Percentage
True positive (1,1) – Perfect match	70	100%
True negative (0.0) – Image not found – if not available	50	100%
False positive (0,1) – Mismatch	0	0%
False negative (1,0) – Mismatch	0	0%

References

[1] Akisato Kimura, Takahito Kawanishi and Kunio Kashino, " Acceleration of similarity-based partial image retrieval using multistage vector quantization", Proceedings of the 17th International Conference on Pattern Recognition, 2004 (ICPR 2004), Volume: 2, August.23-26, 2004, Pages: 993 – 996.

[2] Dimitri Van de Mille, Thierry Blu and Michael Unser, "Hex Splines: A Novel Spline Family for Hexagonal Lattices", *IEEE Transactions on Image Processing,* Vol.13. No.6. June 2004, pp. 758-774.

[3] Shu Yen Van and William E.Higgins, "Symmetric Region Growing", *IEEE Transactions on Image processing,* Vol.12, No.9, September 2003, pp. 1007-1015.

[4] Akisato Kimura, Takahito Kawanishi and Kunio Kashino, " Similarity-Based Partial Image Retrieval Guaranteeing Same Accuracy as Exhaustive Matching", *Proceedings of International Conference on Multimedia and Expo. (ICME2004)* June 2004.

[5] J.C.Ehrhardt, "Hexagonal Fast Fourier Transform with Rectangular Output", *IEEE Transactions on Image processing,* Vol.41, Mar 1993, pp. 1469-1472.

[6] Fabio Dell Aquca and Paulo Gamba, "Simplified Modal Analysis and Search for Reliable Shape Retrieval", *IEEE Transactions on Circuits and Systems for Video Technology,* Vol.8, No.5, September 1998. pp. 656-666.

[7] Injong Rhee, Graham R. Martin, S. Muthukrishnan, and Roger A. Packwood, "Quad tree-Structured Variable-Size Block-Matching Motion Estimation with Minimal Error", *IEEE Transactions on Circuits and Systems for Video Technology,* Vol.10, No.1, February 2000, pp. 42-50.

[8] Richard C. Staunton, "Hexagonal Image Sampling: A Practical Proposition" *Proceeding of SPIE*, 1989, Vol.1008, pp.23-27.

[9] L. Middleton and J. Sivaswamy, "Edge Detection in a Hexagonal-Image Processing Framework," *Image and Vision Computing*, Dec. 2001, Vol. 19, No. 14, pp. 1071–1081.

[10] R.C.Staunton, "One Pass Parallel Hexagonal Thinning Algorithm", *IEE proceedings of Visual Image Signal Processing*, February 2001, Vol.148, No. 1, pp 45-53.

[11] V.S.Subrahmanian, "*Principles of Multimedia Database Systems*", Morgan Kaufman Publishers, 2001.