IMAGE RETRIEVAL USING EFFICIENT DATA STRUCTURES

A thesis submitted in partial fulfillment of the requirements for the award of the degree of

B.Tech

in

Electronics and Communication Engineering

By

ATHIRA B.NAIR (108112017)

ELECTRONICS AND COMMUNICATION ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY
TIRUCHIRAPPALLI-620015

MAY 2016

BONAFIDE CERTIFICATE

This is to certify that the project titled **IMAGE RETRIEVAL USING EFFICIENT DATA STRUCTURES** is a bonafide record of the work done by

ATHIRA B.NAIR (108112017)

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Electronics and Communication Engineering** of the **NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI**, during the year 2015-2016.

Dr B. NITHYA
Guide

Dr D. SRIRAM KUMAR
Head of the Department

Project Viva-voce held on _____

Internal Examiner

External Examiner

ABSTRACT

This project focuses on using efficient data structures for region-based image retrieval. The main aim of this project is to find a specific image region from an image database that is most similar to the query image region. Each image is automatically split into four quadrants and segmented into regions using Expectation-Maximisation method. Each region is associated with a colour descriptor such as colour histogram. Querying is based on the attributes of a particular region rather than the description of the entire image. The descriptors for each region are stored in a tree data structure, along with coordinates of the polygon bounding the region. The output of the program is the image region that roughly corresponds to the query image region cropped by the user. Experimental results show the efficiency of this method and the performance of this tree data structure with respect to other varying parameters.

Keywords: Region-based image retrieval; Segmentation; Tree; Similarity measure.

ACKNOWLEDGEMENTS

I wish to express my deep sense of gratitude and indebtedness to my Project guide Dr B. Nithya for her guidance, constructive criticism and valuable advice throughout the course of this project.

I also extend my sincere thanks to my project coordinator Dr R. Malmathanraj for his encouragement and suggestions throughout.

I am also thankful to my Head of Department Dr D. Shriram Kumar for his immense support and guidance. Finally I wish to thank all the faculty of the Electronics and Communication Engineering Department for their invaluable teaching.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

An image retrieval system (IRS) is a computer system for searching and retrieving images from a database of digital images. Searching interested images based on visual properties or contents of images is a challenging problem and it has received much attention from researchers in the last 20 years.

1.1 IMAGE RETRIEVAL SYSTEM

Most traditional and common methods of image retrieval utilize some method of adding meta-data such as captioning', keywords, or descriptions to the images so that retrieval can be performed over the annotation words. On the downside, it brings too heavy workload and it still remains subjective and uncertain. Manual image annotation is time-consuming, laborious and expensive; to address this, there has been a large amount of research done on automatic image annotation. This method is known as Annotation based image-retrieval which has not been too successful due to the shortcomings in this method.

Another method for image retrieval is based on the content of the image. The visual properties used to compare the images may be colour, texture, shape, etc. extracted from the image. IBM's QBIC is one of the earliest systems to use CBIR successfully. The semantic gap between low-level features and high-level semantic understanding of images is often hard to bridge. In order to solve this semantic gap problem, one of the most popular approaches in recent years is to change the focus from the global content description of images into the local content description by regions (region-based image retrieval) or even the objects in images (object-based image retrieval).[1]

Current image retrieval systems do not perform well in both accuracy and speed. A key reason for the poor quality of query results is that the systems do not look for meaningful image regions corresponding to objects. RBIR is an image retrieval approach which focuses on contents from regions of images, not the content from the entire image. In RBIR systems, the image is first segmented into multiple regions and extracts a set of local features based on which the comparison is done. A similarity measure is employed to determine the best match for the query image. [2]

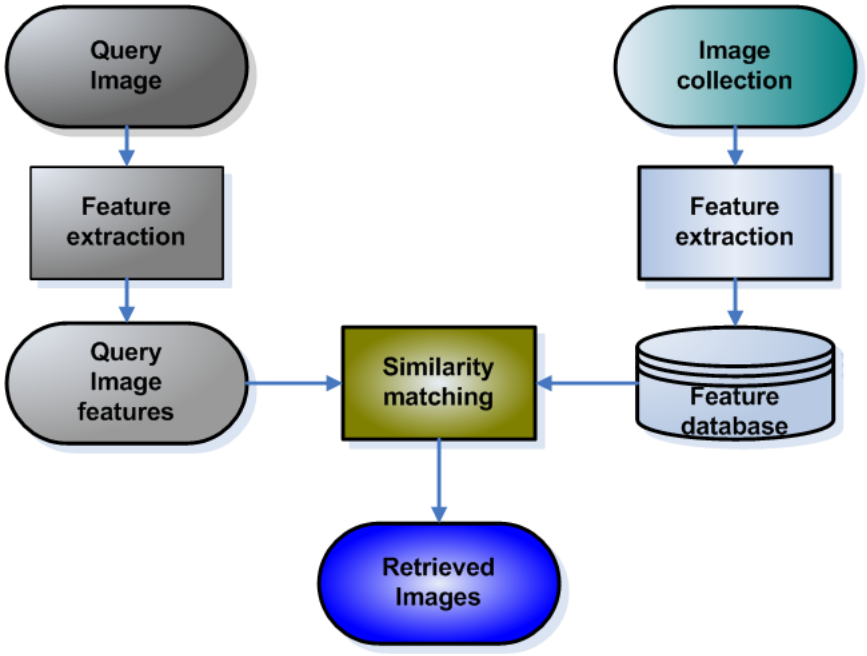

Figure 1.1: Image retrieval system

1.2 APPLICATIONS

Our ultimate goal as computer scientists is to build intelligent machines capable of image management the way humans are, and it is now time for the community to move aggressively into making it a real-world technology. We sense a paradigm shift in the goals of the next-generation researchers in image retrieval. The need of the hour is to establish how this technology can reach out to the common man in the same way text retrieval techniques have.

The applications of image retrieval are wide-spread in multiple areas. Reverse image search is an application that is based on these techniques for identifying images similar to the query image. These systems are particularly helpful when the origin of the image and its contents are not known to the user and cannot be annotated as in traditional annotation based retrieval systems. Image retrieval also has applications in facial recognition i.e. when the user has to find the name of a person from their picture by matching their facial features with the images in the database. It is also used in gesture recognition. It is a technology that can be employed in many useful gadgets, for example, it can be used to help the blind with navigation through their gestures alone. Using image retrieval, certain gestures can be matched to gestures already stored in the database that correspond to certain presaved commands. The commercial systems developed include:

- IBM's QBIC
- Virage's VIR Image Engine
- Excalibur's Image RetrievalWare
- VisualSEEk and WebSEEk
- Netra
- MARS
- Vhoto
- Pixolution
- Blobworld

The main inspiration for this project is the Blobworld project, a new framework for image retrieval based on segmenting each image into "blobs" which generally correspond to objects or parts of objects.

1.2.1 Medical Applications

The number of digitally produced medical images is rising strongly. Imaging systems and image archives have often been described as an important economic and clinical factor in the hospital environment. Medical images have often been used for retrieval systems and the medical domain is often cited as one of the principal application domains for content-based access technologies in terms of potential impact. Still, there has rarely been an evaluation of the performance and the description of the clinical use of systems is even rarer. It is useful for matching a query image to a collection of images of a specific type, for example finger prints or X-ray images of a specific organs.

- The CervigramFinder system was developed to study the uterine cervix cancer. It is a computer assisted framework where on in an image are computed and, using similarity measures, similar images are retrieved from a database.
- The Spine Pathology and Image Retrieval System (SPIRS) is a web-based hybrid retrieval system, working with both image visual features and text-based information. It allows the user to extract spine x-rays images from a database by providing a sketch/image of the vertebral outline. The retrieval process is based in an active contours algorithm for shape discrimination.

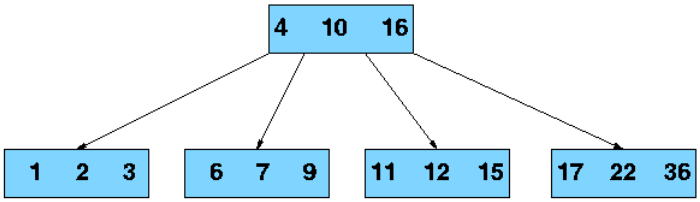

Figure 1.2: B-tree

- The Image Retrieval for Medical Applications (IRMA) system is a generic web-based x-ray retrieval system. It allows the user to extract images from a database given an x-ray image query. Local features and similarity measures are used to compute the nearest images. The SPIRS and IRMA systems were merged to form the SPIRS-IRMA system, with the functionalities of both.

One of the important application domains in medical imaging is the magnetic resonance images (MRI) of brain. The MRIs of multiple conditions in the brain can be matched to the specific disease that affects the brain. CT brain scans the search for medical tumours by their properties after segmentation have been described. The tumour in a brain may have a different MRI from that of a brain of an epileptic person. Image retrieval can match the corresponding images to identify the medical condition under observation.

1.3 DATA STRUCTURES USED IN IRS

To reduce the time an index takes to locate data items, we need to minimize the number of data accesses it uses during the search. This can be accomplished using a tree-based index that stores the data elements in its leaves. Beginning at the root, we can traverse a single path down to one of its leaves to locate a particular data point. Thus, the number of steps a tree uses for searching is proportional to its height. Because of this, much of the research into multidimensional access methods has been focused on developing tree-based indexes. [3]

Many of the trees used in image processing are designed to store geometric shapes. These trees store the shapes using their Minimum Bounding Region (MBR). This refers to the smallest region, usually rectangular, that encloses the entire shape.

The B-tree is a self-balancing search tree structure. It is a generalization of a binary search tree in that a node can have more than two children. Figure 1.2 depicts a sample B-tree.

Unfortunately, the B-tree is inappropriate for multidimensional data. Researchers would like an indexing structure that is as effective for image data as a B-tree is for traditional data. Consequently, most of the research in this area has been directed to modifying the B-tree so it can effectively index multidimensional data.

In these trees, each node corresponds to a specific region of the data space, say R. The children of the nodes correspond to subregions of R. The problem with trees like the K-D-B tree is that the splitting of regions is performed only along a single dimension. Trees like the R-tree, R* tree, R+ tree are more versatile in that they can store information about the Minimum Bounding Region (MBR).

The R-tree, or Rectangle Tree, is one of the more popular data structures for indexing spatial data. Also, like the B-tree, the R-tree is balanced and has two variants, the R* tree and R+ tree. The R+-tree differs from the R-tree in that the MBRs are not permitted to overlap. When MBRs are permitted to overlap, the search algorithm must traverse multiple paths of the tree. If they do not overlap, however, a shape can be located using only one path of the tree. The R*-tree uses a concept called Forced Reinsert, which tries to prevent splits by deleting, then reinserting elements of a full node. [4] Figure 1.3 shows a sample R-tree.

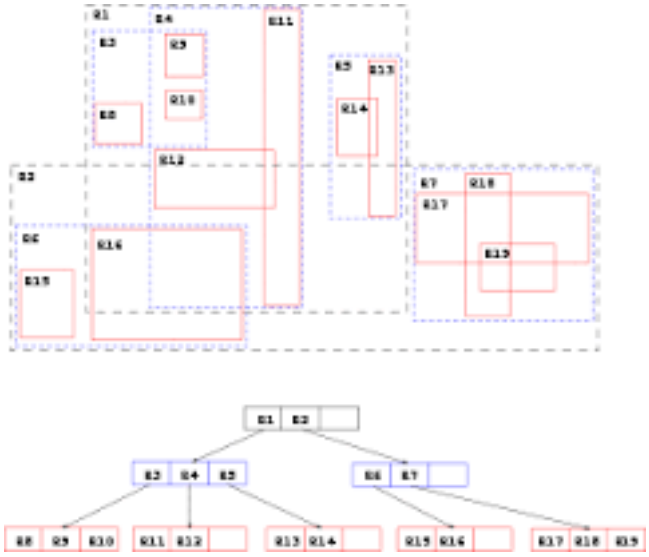

Figure 1.3: R-tree

The Polyhedral or P-trees use a different approach for their MBRs. Instead of using a rectangular bounding region, the P-tree uses a more general shape such as a polygon. The X-tree is another variant which is so named because when the splitting algorithm cannot a split without overlap, it creates a new type of node called a supernode.

Another data structure that has been used in image manipulation but different from the B-tree is the quadtree. It is a tree data structure in which each internal node has exactly

four children. Quadtrees are most often used to partition a two-dimensional space by recursively subdividing it into four quadrants or regions. [**?**] The regions may be square or rectangular, or may have arbitrary shapes. Figure 1.4 shows a sample quad tree.

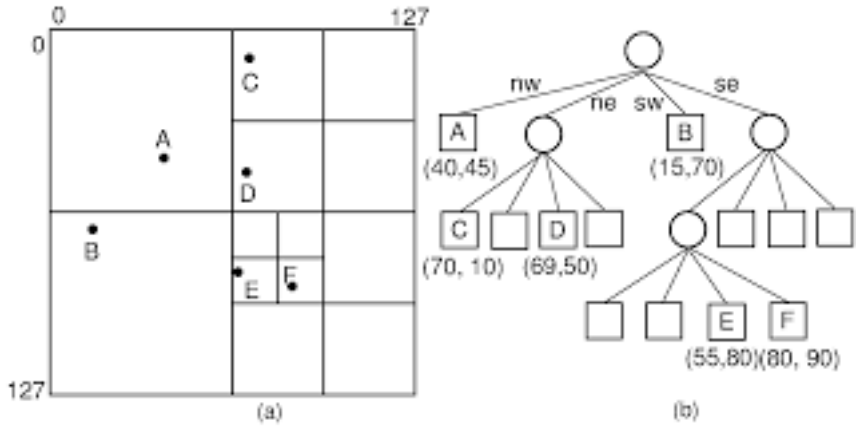

Figure 1.4: Quad tree

Quad picture tree is a picture tree with the original image as the start region and progressively divides each region into four (square) sub-regions with an equal number of pixels. It can be used to guide the search for regions with uniform gray-levels for gray-level images by developing a measure of uniformity (homogeneity) to test the regions as candidates to be split.

1.3.1 Drawbacks

The main disadvantage of quadtrees is that it is almost impossible to compare two images that differ only in rotation or translation. This is because the quad tree representation of such images will be so totally different. The problems start to occur when the image is rotated. This rotation will completely change the quadtree of the image. Once the quadtree has changed it becomes a difficult task to compare it to an earlier version quadtree. Therefore quadtrees have poor shape analysis and pattern recognition.

Although an R tree is height balanced, the branching factor of each node is not the same. This has several drawbacks. This means that the nodes are not fully occupied, causing the structure to go deeper. This leads to imbalance in terms of number of objects in the nodes. Since rectangles are duplicated, an R tree can be larger than other data structures built on same data set. Construction and maintenance of R trees is more complex than the construction and maintenance of other trees.

Though so many variants of data structures of data structures have been used, all of them have noticeable drawbacks as mentioned, that hinder the development of an effi-

cient retrieval system. In the system used in this project, a combination of the quad tree and R-tree is used in order to achieve accurate results. This leads to a less complex construction than an R-tree such that its implementation does lead to high overhead. At the same time, it ensures that the quadtree decomposition is utilised to effectively partition the image. Sometimes image segmentation does not give completely reliable results and hence partitions may be haphazard. By splitting the image into quadrants for the initial child nodes of the tree, missing out on any region can be avoided. Therefore, the need to overcome these drawbacks and create a new efficient structure is the motivation for this project.

CHAPTER 2

LITERATURE REVIEW

2.1 TEXT-BASED IRS

The image retrieval that is based on artificial notes labels images by using text firstly, in fact it has already changed image retrieval into traditional keywords retrieval. There are two problems remain in this method. One, it brings too heavy workload. Two, it still remains subjectivity and uncertainty. Because the image retrieval that is based on artificial notes still remains insufficient, further study that adapts vision image features has come up and become the main study. The major trait of this method is image feature extraction, whether the retrieval is good or not depends on the accuracy of the features extraction.

2.2 CONTENT-BASED IRS

One of the reasons for writing this section is that CBIR, as a field, has grown tremendously after the year 2000 in terms of the people involved and the papers published. Lateral growth has also occurred in terms of the associated research questions addressed, spanning various fields.

Many current image retrieval systems perform retrieval based primarily on low-level image features, including IBM's Query by Image Content or QBIC, Photobook, Virage, VisualSEEk, Candid and Chabot. Lipson et al. retrieve images based on spatial and photometric relationships within and across simple image regions. Little or no segmentation is done; the regions are derived from low-resolution images. [5]

2.2.1 CBIR in Medical Imaging

Research in content-based image retrieval (CBIR) today is an extremely active discipline. There are already review articles containing references to a large number of systems and description of the technology implemented. A more recent review reports a tremendous growth in publications on this topic. Applications of CBIR systems to medical domains already exist, although most of the systems currently available are based on radiological images. Most of the work in dermatology has focused on skin cancer detection. Different techniques for segmentation, feature extraction and classification have been reported by several authors. Concerning segmentation, Celebi et

al. presented a systematic overview of recent border detection methods: clustering followed by active contours are the most popular. Numerous features have been extracted from skin images, including shape, colour, texture and border properties. Classification methods range from discriminant analysis to neural networks and support vector machines. These methods are mainly developed for images acquired by epiluminescence microscopy (ELM or dermoscopy) and they focus on melanoma, which is actually a rather rare, but quite dangerous, condition whereas other skin cancers are much more common.

There are few CBIR systems in dermatology. Chung et al. created a skin cancer database. Users can query the database by feature attribute values (shape and texture), or by synthesised image colours. It does not include a query-by-example method, as do most common CBIR systems. The report concentrates on the description of the web-based browsing and data mining. However, nothing is said about database details (number, lesion types, acquisition technique), nor about the performance of the retrieval system. Celebi et al. developed a system for retrieving skin lesion images based on shape similarity. The novelty of that system is the incorporation of human perception models in the similarity function. Results on 184 skin lesion images show significant agreement between computer assessment and human perception. However, they only focus on silhouette shape similarity and do not include many features (colour and texture) described in other papers by the same authors. Rahman et al. presented a CBIR system for dermatoscopic images. Their approach include image processing, segmentation, feature extraction (colour and textures) and similarity matching. Experiments on 358 images of pigmented skin lesions from three categories (benign, dysplastic nevi and melanoma) are performed. A quantitative evaluation based on the precision curve shows the effectiveness of their system to retrieve visually similar lesions. Dorileo et al. presented a CBIR system for wound images (necrotic tissue, fibrin, granulation and mixed tissue). [6]

Dermatology atlases containing a large number of images are available on line. However, their searching tool only allows query by the name of the lesion. On the other hand, the possibility of retrieving images based on visual similarity would greatly benefit both the non-expert users and the doctors. As already pointed out, there is a need for CBIR as a decision support tool for dermatologists in the form of a display of relevant past cases, along with proven pathology and other suitable information. CBIR could be used to present cases that are not only similar in diagnosis, but also similar in appearance and cases with visual similarity but different diagnoses. Hence, it would be useful as a training tool for medical students and researchers to browse and search large collection of disease related illustrations using their visual attributes.

2.3 BLOBWORLD

Blobworld is a system for region-based retrieval image which is the main motivation for this project. Each image is automatically segmented into regions or ”blobs” with associated color and texture descriptors. Querying is based on the attributes of one or two regions of interest, rather than a description of the entire image. In order to make large-scale retrieval feasible, it indexes the blob descriptions using a tree. Because indexing in the high-dimensional feature space is computationally prohibitive, it uses a lower-rank approximation to the high-dimensional distance.

The Blobworld representation is related to the notion of photographic or artistic scene composition. Blobworld is distinct from color-layout matching as in QBIC in that it is designed to find objects or parts of objects; each image is treated as an ensemble of a few blobs representing image regions which are roughly homogeneous with respect to color and texture. A blob is described by its color distribution and mean texture descriptors.[7]

2.4 FEATURE EXTRACTION

Feature selection is applied in order to select a subset of texture features from all the feature extracted. Most systems perform feature extraction as a preprocessing step, obtaining global image features like colour histogram or local descriptors like shape and texture.

2.4.1 Blobworld Descriptors

Colour and Texture Descriptors

Each pixel is assigned a vector consisting of color, texture, and position features. The three color features are the coordinates in the L*a*b* color space; features are smoothed to avoid over segmentation arising from local color variations due to texture. The three texture features are contrast, anisotropy, and polarity, extracted at a scale which is selected automatically. The position features are simply the coordinates of the pixel; including the position generally decreases over segmentation and leads to smoother regions. The distribution of pixels in this 8-D space using mixtures of two to five Gaussians. We use the Expectation-Maximization algorithm to fit the mixture of Gaussians model to the data. To choose the number of Gaussians that best suits the natural number of groups present in the image, the Minimum Description Length principle is applied. Once a model is selected, spatial grouping of connected pixels belonging to the same

color texture cluster is performed.

The color histogram over the pixels in each region are stored. The histogram is based on bins with width 20 in each dimension of L*a*b* space. This spacing yields ve bins in the L* dimension and ten bins in each of the a* and b* dimensions, for a total of 500 bins. [8]

Texture is a well-researched property of image regions, and many texture descriptors have been proposed, including multi-orientation filter banks and the second-moment matrix. While color is a point property, texture is a local neighbourhood property. It does not make sense to talk about the texture of zebra stripes at a particular pixel without specifying a neighbourhood around that pixel. In order for a texture descriptor to be useful, it must provide an adequate description of the underlying texture parameters and it must be computed in a neighbourhood which is appropriate to the local structure being described. The first requirement could be met to an arbitrary degree of satisfaction by using multi-orientation filter banks such as steerable filters. The second requirement, which may be thought of as the problem of scale selection, does not enjoy the same level of attention in the literature. This is unfortunate, since texture descriptors computed at the wrong scale only confuse the issue.

Combining Colour and Texture

The colour/texture descriptor for a given pixel consists of six values: three for colour and three for texture. The three colour components are the colour-cone coordinates found after spatial averaging using a Gaussian at the selected scale. The three texture components are ac, pc, and c, computed at the selected scale; the anisotropy and polarity are each modulated by the contrast in analogy to the construction of the colour-cone coordinates. In effect, a given textured patch in an image first has its texture properties extracted and is then replaced by a smooth patch of averaged color. In this manner, the color and texture properties in a given region are decoupled; for example, a zebra is a gray horse plus stripes. Note that in this formulation of the colour/texture descriptor, orientation and selected scale do not appear in the feature vector; as a result, grouping can occur across variations in scale and orientation.

Spatial Descriptors

Shape is a key attribute of segmented image regions, and its efficient and robust representation plays an important role in retrieval. The geometric descriptors of the blob are simply the centroid c and scatter matrix S of the blob region; the centroid provides a notion of position, while the scatter matrix provides an elementary shape description. In the querying process, centroid separations are expressed using Euclidean distance. The determination of the distance between scatter matrices is based on the three quantities, These three quantities represent approximate area, eccentricity, and orientation.

CHAPTER 3

IMAGE PROCESSING ALGORITHMS

3.1 IMAGE SEGMENTATION

Image segmentation is the process of partitioning a digital image into multiple segments. The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyse. Image segmentation is typically used to locate objects and boundaries in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics. Image segmentation is the first and foremost step in image retrieval since in order to find clusters in the images, segmentation has to be done.

The role of segmentation is crucial in most tasks requiring image analysis. The success or failure of the task is often a direct consequence of the success or failure of segmentation. However, a reliable and accurate segmentation of an image is, in general, very difficult to achieve by purely automatic means. Application of segmentation include industrial inspection, Optical character recognition (OCR), tracking of objects in a sequence of images, classification of terrains visible in satellite images., detection and measurement of bone, tissue, etc., in medical images. [9]

The conditions for a good segmentation are:

- All pixels must be assigned to regions.
- Each pixel must belong to a single region only.
- Each region must be uniform.
- Any merged pair of adjacent regions must be non-uniform.
- Each region must be a connected set of pixels.

A great variety of segmentation methods has been proposed in the past decades. Image segmentation algorithms generally are based on one of two basic properties of intensity values: discontinuity and similarity.

- Discontinuity based approach: Partition an image based on abrupt changes in intensity.
- Similarity based approach: Partition an image based on regions that are similar according to a set of predefined criteria.
  - Thresholding
  - Region growing
  - Region splitting and merging

Each of the pixels in a region are similar with respect to some characteristic or computed property, such as color, intensity, or texture. Perfect image segmentation, that is each pixel is assigned to the correct object segment is a goal that cannot usually be achieved. Indeed, because of the way a digital image is acquired, this may be impossible, since a pixel may straddle the real boundary of objects such that it partially belongs to two (or even more) objects.

Methods that assign a segment probability distribution to each pixel are called probabilistic. This class of methods is theoretically more accurate, and applications where a probabilistic approach is the only approach accurate enough for specific object measurements can easily be named. The success of image analysis depends on reliability of segmentation, but an accurate partitioning of an image is generally a very challenging problem. Many segmentation techniques such as thresholding method, adaptive K-means clustering, watershed algorithm, histogram-based methods, edge detection and region growing models have been used in previous methods.

3.1.1 Expectation maximization algorithm

The Expectation-Maximization (EM) algorithm is used to determine the maximum likelihood parameters of a mixture of K Gaussians inside the 6-D feature space. The EM algorithm is used for finding maximum likelihood parameter estimates when there is missing or incomplete data. In our case, the missing data is the region to which the points in the feature space belong. The values are estimated to fill in for the incomplete data (the E-Step), compute the maximum likelihood parameter estimates using this data (the M-Step), and repeat until a suitable stopping criterion is reached. Upon convergence, the Gaussian mixture parameters can be inspected to determine what color/texture properties are represented by each component of the mixture. Some examples of groups that can form include:

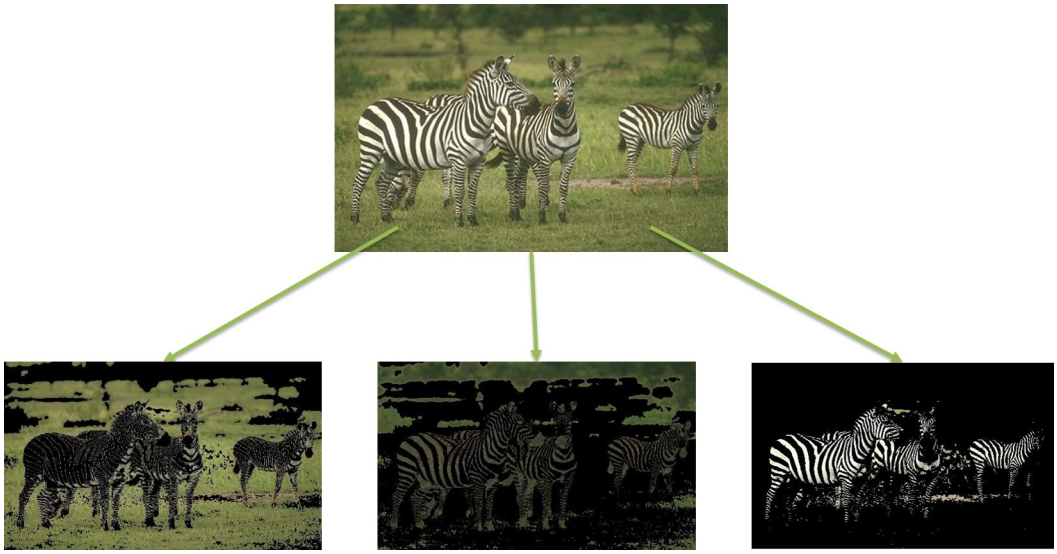

Figure 3.1: Sample image and its 3 segment masks

- bright, bluish, and textureless regions (e.g., sky)
- anisotropic and non-polar regions (e.g., zebra hide)
- green weak-isotropic texture (e.g., grass)

A sample image segmented into 3 regions has been shown in Figure 3.1 with its masks alongside.

The parameter that can be varied to get different results is the variable K, the number of mixture components. Ideally that value of K is chosen that best suits the natural number of groups present in the image. Once a model is selected, the next step is to perform spatial grouping of those pixels belonging to the same color/texture cluster. First a K-level image is produced which encodes pixel-cluster memberships by replacing each pixel with the label of the cluster for which it attains the highest likelihood.[10]

EM is frequently used for data clustering in machine learning and computer vision. In natural language processing, two prominent instances of the algorithm are the Baum-Welch algorithm and the inside-outside algorithm for unsupervised induction of probabilistic context-free grammars. In psychometrics, EM is almost indispensable for estimating item parameters and latent abilities of item response theory models. With the ability to deal with missing data and observe unidentified variables, EM is becoming a useful tool to price and manage risk of a portfolio. It is also widely used in medical image reconstruction, especially in positron emission tomography and single photon emission computed tomography. In structural engineering, the Structural Identification using Expectation Maximization algorithm is an output-only method for identifying natural vibration properties of a structural system using sensor data.

3.2 MINIMUM BOUNDING REGION

In geometry, the minimum or smallest bounding or enclosing box for a point set (S) in N dimensions is the box with the smallest measure (area, volume, or hypervolume in higher dimensions) within which all the points lie. The minimum bounding box of a point set is the same as the minimum bounding box of its convex hull, a fact which may be used heuristically to speed up computation.

The minimum bounding box was first introduced as an approximation tool for spatial indexing by Guttman. His idea was to have each index record in a leaf node of an R-Tree be identified by the smallest enclosing rectangle that spatially contains the n-dimensional data object and a pointer to the file containing the actual representation of the object. It has since become one of the most frequently used approximation techniques for spatial indexes. The minimum bounding box is one of the most popular approximation methods in spatial access methods. The reason for its popularity is its simple representation. It only requires two points to represent a minimum bounding box, whereas the object the minimum bounding box represents may be many orders of magnitude more complex. Many spatial data structures have been developed to take advantage of minimum bounding box approximations. The most common are the R-Tree and its variations.

3.2.1 Convex hull

In mathematics, the convex hull or convex envelope of a set X of points in the Euclidean plane or Euclidean space is the smallest convex set that contains X. For instance, when X is a bounded subset of the plane, the convex hull may be visualized as the shape enclosed by a rubber band stretched around X.

In digital image processing, the bounding box is merely the coordinates of the rectangular border that fully encloses a digital image when it is placed over a page, a canvas, a screen or other similar background. Figure 3.2 shows a sample image with a MBR as a rectangle surrounding the zebras in the image. Figure 3.3 shows the sample with a minimum bounding region as a general polygon or a convex hull.

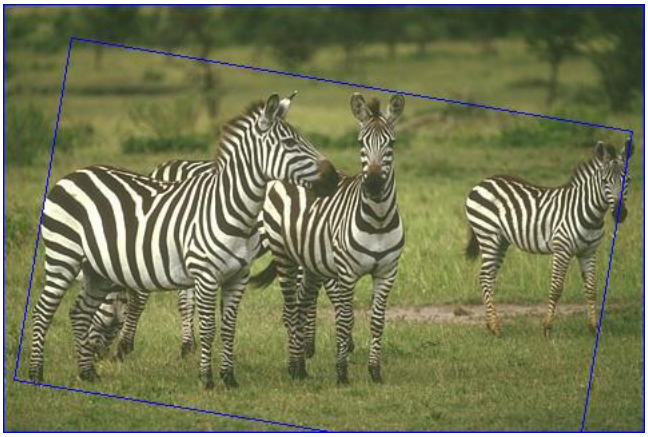

Figure 3.2: Minimum bounding rectangle of an image region

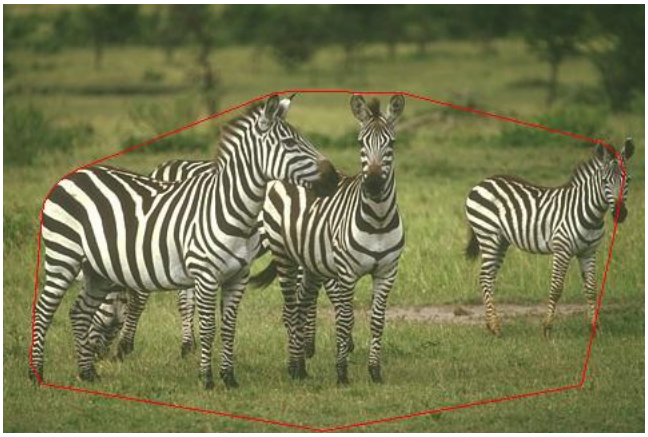

Figure 3.3: Minimum bounding polygon of an image region

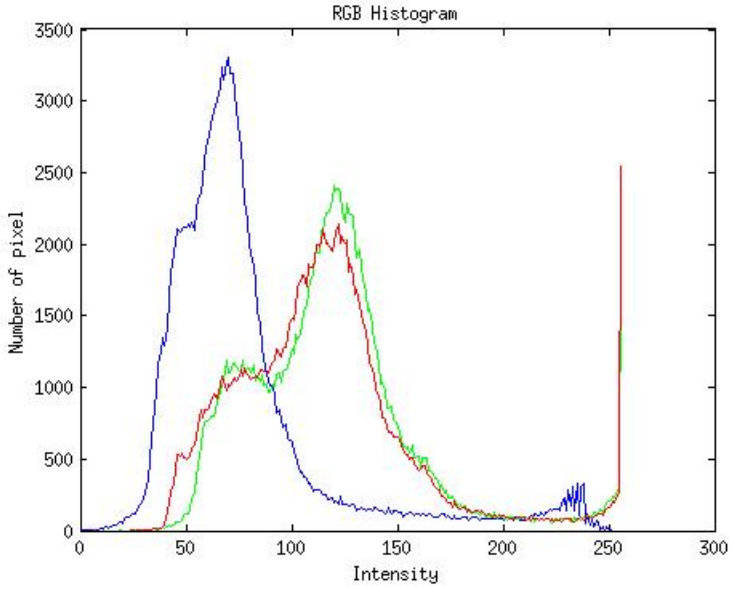

Figure 3.4: RGB Histogram

3.3 DESCRIPTOR

A descriptor is a value that defines a region based on its properties like colour, texture and shape. For describing multimedia contents, low level features are usually used since they can be generated efficiently. Among the low level image features, color is the most expressive and effective in visual content description.

Histograms are frequently used tools in natural language processing and various computer vision tasks, including image retrieval, image classification, shape matching, and object recognition, to represent texture and color features or to characterize rich information in local/global regions of objects. In particular, a histogram in the statistics is the frequency distribution of a set of specific measurements over discrete intervals. For many computer vision tasks, each object of interest can be presented as a histogram by using visual descriptors. As a result, the resulting histogram obtains some merits of the descriptors, for example, rotation-invariant, scale-invariant, and translation-invariant. These make it an excellent representation method for performing classification and recognition of objects. When the histogram representations are adopted, the choice of histogram distance metric has a great impact on the classification performance or recognition accuracy of the specific task.

A color histogram is a representation of the distribution of colors in an image. Colour is one of the most reliable visual features that are also easier to apply in image retrieval systems. Colour is independent of image size and orientation, because, it is robust to background complication. Colour histogram is the most common method for extracting the color features of coloured images. Colour histograms are widely used for RBIR systems in the image retrieval area. It is one of the most common methods for predicting the features of an image. The bins for each colour are kept the same in the histogram. This leads to n bins in each colour leading to a total of n^3 bins. A sample colour histogram is displayed in Figure 3.4 which displays the frequency for different intensities of the colours.

3.4 SIMILARITY MEASURE

A distance measure is then used to measure the (dis-)similarity between images based on the descriptions of their color distributions in order to quickly find relevant images. The development and investigation of statistical methods for robust representations of such distributions, the construction of distance measures between them and their applications in efficient retrieval, browsing, and structuring of images is very crucial for the project.

There are many established methods to compute the distance between two histograms, examples being quadratic distance, euclidean distance, hamming distance, Mahalanobis distance and so on. The chi squared distance method is used as the primary similarity measure. The chi squared distance d(x,y) is, as you already know, a distance between two histograms $x = [x_1, .., x_n] and y = [y_1, ..., y_n]$ having n bins both. Moreover, both histograms are normalized, i.e. their entries sum up to one. The distance measure d is usually defined (although alternative definitions exist) as

$$d(x, y) = \frac{1}{2}\Sigma\frac{(x_i - y_i)^2}{x_i + y_i}$$

It is often used in computer vision to compute distances between some bag-of-visual-word representations of images. The name of the distance is derived from Pearson's chi squared test statistic for comparing discrete probability distributions (i.e histograms). However, unlike the test statistic, d(x,y) is symmetric with respect to x and y, which is often useful in practice, e.g., when you want to construct a kernel out of the histogram distances.

Although metric learning about Mahalanobis distance has been widely studied, metric learning for chi-squared distance is largely unexplored. Unlike Mahalanobis distance, chi-squared distance is a non-linear metric and its general form requires the learned linear transformation to be simplex-preserving. Therefore, the existing linear metric learning algorithms cannot naturally apply to chi-squared distance.

CHAPTER 4

PROPOSED SOLUTION

4.1 QUERYING

Anyone who has used a search engine, text-based or otherwise, is familiar with the reality of unwanted matches. Often in the case of text searches this results from the use of ambiguous keywords. Unfortunately, with image queries it is not always so clear why things go wrong. Unlike with text searches, in which the user can see the features (words) in a document, none of the current content-based image retrieval systems allows the user to see exactly what the system is looking for in response to a similarity-based query. Simply allowing the user to submit an arbitrary image (or sketch) and set some abstract knobs without knowing how they relate to the input image in particular implies a degree of complexity that searching algorithms do not have. As a result, a query for a bear can return just about any object under the sun if the query is not based on image regions, the segmentation routine fails to find the bear in the submitted image, or the submitted image contains other distinctive objects. Without realizing that the input image was not properly processed, the user can only wonder what went wrong. In order to help the user formulate effective queries and understand their results, as well as to minimize disappointment due to overly optimistic expectations of the system, the system should display its representation of the submitted image and the returned images.

This is the reason why region-based image retrieval systems came into being. They give the user the flexibility of selecting the region, allowing them more precision to filter their results. The program focuses on acquiring the image region closest to the query.

Query by example is a query technique that involves providing the image retrieval system with an example image that it will then base its search upon. The underlying search algorithms may vary depending on the application, but result images should all share common elements with the provided example. Options for providing example images to the system include a pre-existing image that may be supplied by the user or chosen from a random set, or the user draws a rough approximation of the image they are looking for, for example with blobs of color or general shapes. This query technique removes the difficulties that can arise when trying to describe images with words.

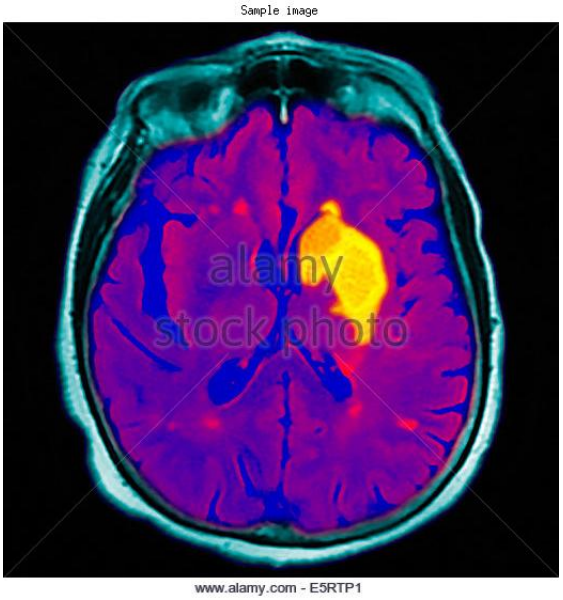

Figure 4.1: Query Image

In the program, the query brain MRI image is cropped to get the specific region to be queried, say a tumour or an abnormality in the brain. The user can also select a non-tumour region. The program first lets the user select the query region in the image using the Matlab crop tool. Afterwards, the RGB histogram of the cropped selection is calculated as the initial step. The images to be compared with are retrieved from memory after query is given as input.

Figure 4.1 shows the image sample used for query and Figure 4.2 shows the cropped region of the image.

Figure 4.2: Cropped query region

4.2 DATA STRUCTURE

Brain MRI images are used for experiment purposes in order to emphasize its use in medical imaging applications. For this project, a new tree was developed in order to accommodate the various differences in brain MRI images. Figure 4.3 shows the sample image used for the database.

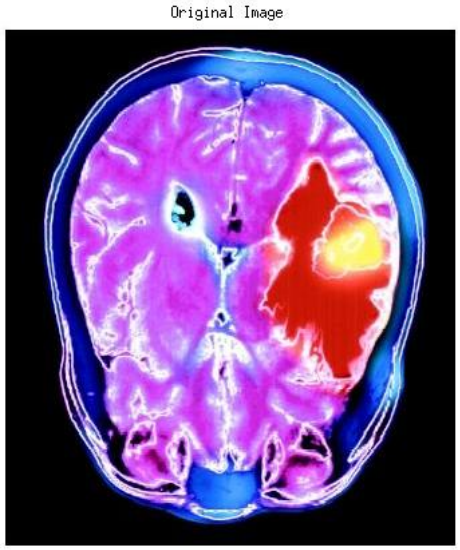

Figure 4.3: Original Image

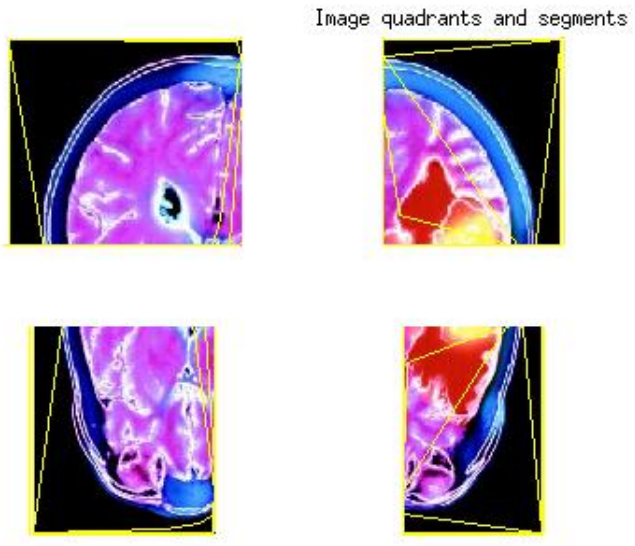

Figure 4.4: Quadrants of image and its segments highlighted

The image in the database is first split into four quadrants, similar to a quad tree. This gives rise to the first level i.e. the root (the image) consists of four children. Instead of splitting into four recursively in each level, we instead store each region from the

segmentation in its child nodes. The image segmentation through EM method leads to K segments, which can be defined by the user. The number of regions under each quadrant depend upon K, and hence the number of leaf nodes amount to 4 x K. Each node in the first level contains a list of the minimum bounding regions (MBRs) of its children and pointers to those children in the form of coordinates. It also contains information on the histogram of that region, which it uses to compare with other regions for similarity measure. MBR of a node is a polygon that minimally encloses all the children of that node. Leaf nodes contain all the data points that enclose the MBRs. Figure 4.4 shows the sample image with four quadrants and each of the segments in these quadrants highlighted in yellow.

The hybrid of a quad-tree and R-tree leads to better results. The R-tree is not the most optimal tree for search and the quad tree is bulky memory-wise. This tree data structure compromises on the recursive nature of quad tree and region splitting of R-tree to obtain optimum search time and efficient results.

However unlike R+-trees, the data structure hasn't taken overlap into account. There may be regions with heavy overlap which may lead to degradation in results. Unlike R-trees, the tree stores MBRs as polygons or convex hulls in order to get a better contour of the match. This leads to higher memory utilisation than it would if rectangles were used.

Index speed degrades as the dimensionality of the data indexed increases. Because index trees are secondary storage structures, each node and leaf in an index tree must fit on a single disk page. Shorter paths from root to leaf in an index tree lead to fewer disk accesses to reach the leaves, and thus faster index retrievals. Node fanout, the number of data entries that can fit in a node, dictates index tree height. Smaller data entries allow greater fanout and faster index retrievals. Higher dimensional data requires larger data entries and thus lower fanout. At sufficiently high dimensions fanout becomes so low that query speed using the index is worse than simply scanning the entire database. To avoid this, only trees of depth 2 are used in order to store entries unlike in quad trees where it's recursive. Hence like R-trees, this data structure utilises only two levels of a tree to store information.

The tree defined in Matlab such that a constructor is called each time a tree has to be created. It is included with inbuilt methods to add, remove or modify elements and perform simple operations using tree elements. It also has the added functionality of being able to traverse the tree depth-wise or breadth-wise using an iterator function. The tree class has a very small memory overhead, by construction, and is rather memory efficient. [11]

The first way is to iterate depth-first. That is, when a node is reached, all of its children and sub-children are parsed through before parsing its siblings. So traversal is done from the root through the first branch, going until the leaf of that branch is reached. Then a level is further upped and the same process is continued until all the children have been traversed. This is the method used in our tree data structure, as each quadrant is parsed its children are parsed immediately to parse through all the regions within the quadrant. Using just a pointer to the root, the tree can be traversed through depth-first and hence it is very fast and memory-efficient. Figure 4.5 shows the tree structure using an image from the database.

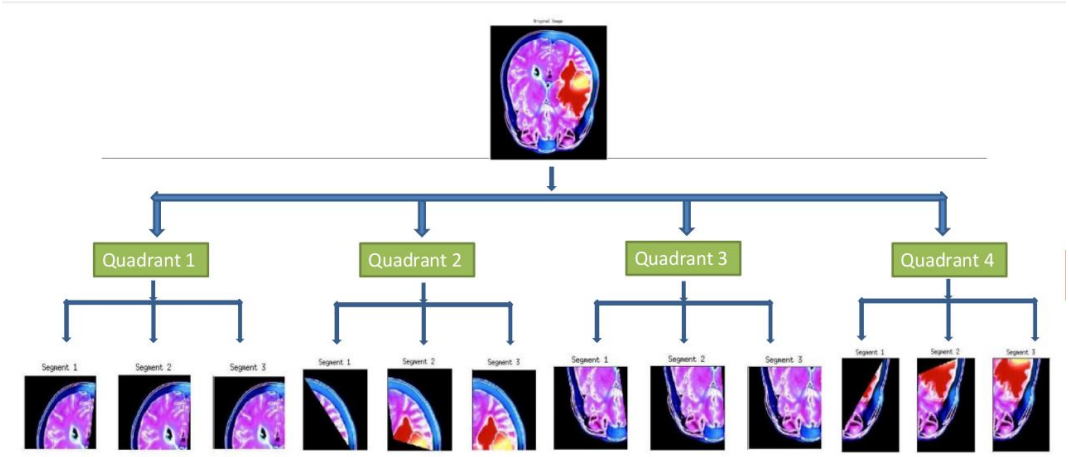

Figure 4.5: Tree structure

Figure 4.6 shows the tree structure as displayed on Matlab with leaf nodes having information regarding the region stored in a cell. The child nodes of the root have the respective quadrant images stored in them. As the tree is traversed, the quadrant region can be fetched and the MBR coordinates and colour histogram descriptor can be obtained by traversing one level deeper.

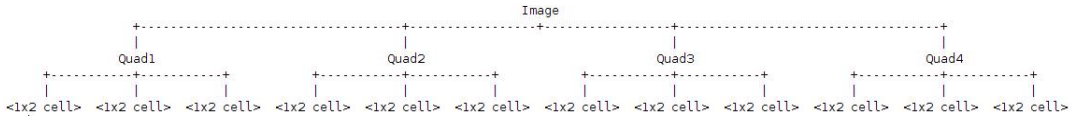

Figure 4.6: Tree in Matlab

CHAPTER 5

RESULTS AND DISCUSSIONS

The program is run by specifying different parameters during run time to analyse how the time taken to run the program varies with the change in the variables. Time analysis is done using Matlab and its internal timer. The program does not take into the account the time taken by the user to crop the image, as it is subject to variation depending on external circumstances. The program code is in alignment with the step-by-step approach to the image retrieval solution.

First, the query image is cropped by the user. This part of the execution is not taken into consideration for time analysis. The timer is started from the time the histogram of the query region is being calculated. Overhead times to read the image and process it are also taken into consideration while loading images to database i.e. they are being read during runtime.

The segmentation, being the most complicated part of the process, takes up considerable part of the execution time. This suggests that as the value of K increases, runtime should also be affected proportionately. The segmentation process therefore will have a significant role to play in the results obtained, as shown in Figure 5.1.

It is observed that size of the images in the database has considerable effect on the total runtime. As the dimensionality of the image increases, the runtime increases which is a trade-off to consider in systems where latency is an important criteria. To avoid this, images may be compressed or scaled suitably before the program execution by the user. We have not included the feature as embedded in the program, due to the fact that the user may want to retain the extra dimensionality without compromise on the quality. It is quite possible that brain MR images may have subtle differences in the tumour region that may be blurred or pixelated if the image is compressed.

It is also observed that as the size of the database increases, the run time of the program also increases. The time taken for each image to segment and be stored in the tree nodes occupies most of the runtime. As more images have to be segmented, it takes longer for the program to find the appropriate result. There is a quality time trade off. As the index returns more images, the final query results will get better, but the query will take longer.

Profile Summary

Generated 03-May-2016 11:53:08 using cpu time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
code2	1	52.153 s	0.364 s	
EMSeg	1	40.841 s	10.863 s	
EMSeg>distribution	400178	23.845 s	23.845 s	
EMSeg>histogram	1	6.122 s	5.683 s	
imcrop	9	4.795 s	0.021 s	
imcrop>parseInputs	9	4.773 s	0.086 s	
rgbhist_fast	13	4.163 s	4.163 s	
imcrop>interactiveCrop	1	3.307 s	0.086 s	
imcropRect>imcropRect.imcropRect	1	2.911 s	0.000 s	
imrect>imrect.imrect	1	2.868 s	0.000 s	

Figure 5.1: Factors affecting run time

5.1 PERFORMANCE

In order to have a fair picture of how each parameter significantly affects the performance of the system, the environment is considered to have only one image in the database, amounting to 4 x K regions. In our environment, we have used 3 segments as the norm for EM segmentation. Therefore the standard for the image retrieval system is 12 regions. The sample image in the database and the query image are the ones shown in the previous chapter. The variables under consideration in our experiments were size of the sample image and the value of K i.e. the number of segments.

Table 5.1 shows the results obtained for the program when the number of segments was varied. The time taken to run the program was duly recorded as number of segments were increased. These results were obtained for both cases where tumour regions and non-tumour regions were selected. Figure 5.2 show the graph for the variation of runtime vs number of segments.

Table 5.1: Table for runtime vs number of segments for image of size 29kB

Number of segments	2	3	4	5	6
Tumour region	13.0090	16.2303	21.1704	25.9023	29.6796
Non-tumour region	11.9902	18.0261	22.5562	26.2556	30.0437

The results indicate that not much difference is shown if the region selection has been varied and the tumour has not been captured. The stark difference is captured in the variation of number of segments. As K increases, understandably the time to compute

increases. The segmentation algorithm is more complex if the number of regions to segment into increases, hence the execution time is also shown to increase almost linearly, for both tumour and non-tumour selections. The graph has been plotted for an image of size 29 kB i.e. it has been kept constant.

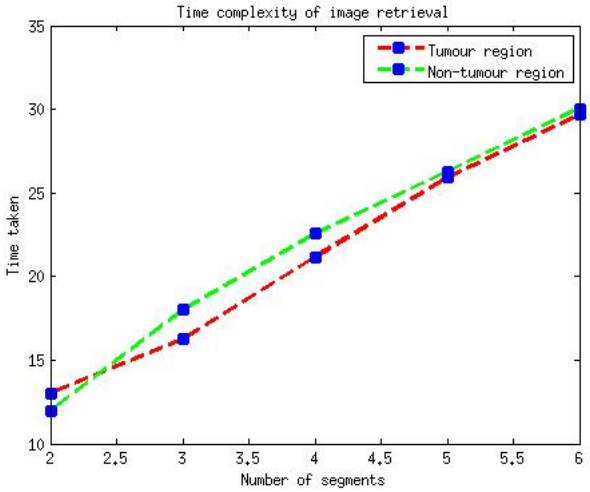

Figure 5.2: Graph of number of segments vs runtime

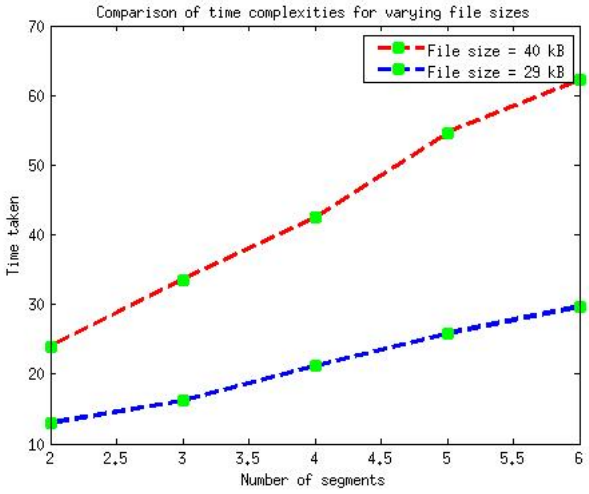

Figure 5.3: Graph of number of segments vs runtime for varying image sizes

The next result was obtained by changing K, the number of segments, as well and varying the image size as the x-axis parameter. Two images of size 29 kB and 40 kB were used in the database for the experiment. As predicted, with the size of the image the dimensionality of the image increases. This leads to a considerable increase in time which is non-linear to the increase in size.

Table 5.2 shows the variation of runtime with number of segments for the two image sizes. Figure 5.3 plots the graph for both the images for comparison.

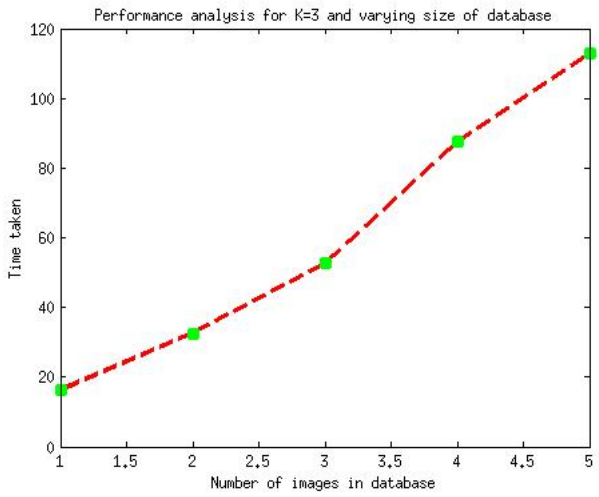

Figure 5.4: Graph of number of images vs runtime

Table 5.2: Table for runtime vs number of segments for image of size 40 kB

Number of segments	2	3	4	5	6
Tumour region	23.9771	33.5614	42.4109	54.6040	62.2217
Non-tumour region	23.4808	33.2340	43.5257	56.1291	67.2334

The results have been compiled for a database containing 12 regions. As for Table 5.2, the number of regions in the database varies from 8 to 24. As tumour images are highly specific, the search results are narrowed down. In order to speed up computation for trials in the experiment, the most basic number of images have been used. The code can be modified for a varying number of images in the database by adding another image tree to the database. The pointers to the tree are stored in a cell 't'. Therefore the size of this cell can be varied without any upper limit restriction, although this is not advisable because of increasing runtime.

Table 5.3: Table for runtime vs number of images for K=3

Number of images	1	2	3	4	5
Time taken	16.2303	32.5573	52.5519	87.4352	112.9952

The experiment was done for varying number of images in the database. The degradation in the performance of the system is observed by analysing the time taken to run the code. It is observed that the runtime increases starkly after the number of images are increased beyond the point. This only confirms the predictions that for larger databases, it is more time-consuming. Table 5.3 shows the variation of runtime with increase in number of images in database, for a constant value K = 3. Figure 5.4 plots the graph for the given values.

5.2 LIMITATIONS

Though the program has achieved a set target, it has various limitations that have to be conquered before it can be implemented on a large scale. The program is yet to be run on a large scale i.e. it is yet to be tested with a large enough database that is often used in image retrieval systems on the internet. Though it works for a small scale medical imaging application, it needs multiple trials to be expanded for larger projects.

Another drawback of the project is that it does not accomplish enough feature extraction to take into account texture and shape properties of the region. Though colour histograms continue to be a fault-tolerant property for small scale operations, it may not suffice for a larger database where the nuances may not be easily distinguishable. In case of images with very minute variations as in case of very small tumours in the brain, there may be a need to analyse the shape of the tumour along with other variations. These are yet to be implemented in the project and it still works on a very basic level. It also relies on colour properties alone for segmentation which make it unilateral for region splitting.

It also does not capitalize on more distance measures and classifiers that can be incorporated, owing to the fact that there are a myriad of techniques available to compare histograms. It is highly likely that different classifiers may yield better results depending upon the quality of the database. Results testing the accuracy of these (dis)similarity measures are yet to be compared and drawn from experiments.

5.3 FUTURE SCOPE

The results can further be improved upon by adding more images to the database. Having more images in the database would result in a wider range for comparison, thereby increasing accuracy. It may also increase runtime considerably. A balance between the number of images and runtime has to be found to maintain an optimum number in the database. Further classifiers can be tested out for similarity measures as there are a number of histogram techniques available for computer vision. The system can be extended beyond brain MR images to other medical images and can be put to use in various applications.

5.4 CONCLUSION

A new method is proposed which uses Expectation-Maximization on color histogram to provide an image segmentation, as well as a new image representation which uses this segmentation and its associated descriptor to represent image regions explicitly. A new tree structure is proposed that efficiently splits the image into quadrants and then stores the segments for a more memory-efficient storage. Experiments were conducted to study the performance of this system with varying parameters to understand its efficacy. A query mechanism is demonstrated that retrieves images and help guide user queries through the use of an efficient data structure.

REFERENCES

[1] Remco C. Veltkamp, Mirela Tanase, and Danielle Sen. Features in content-based image retrieval systems: A survey.

[2] Wei Huang, Yan Gao, and Kap Luk Chan. A review of region-based image retrieval. In *Journal of Signal Processing Systems*, volume 59, pages 143–161, 2010.

[3] Leonard Brown. Tree-based indexes for image data.

[4] Megha Mihir Shah and Pratap Singh Patwal. Multi-dimensional image indexing with r+-tree. volume 3, 2014.

[5] Mussarrat Yasmin, Muhammad Sharif, and Sajjad Mohsin. Use of low level features for content based image retrieval: Survey. In *Research Journal of Recent Sciences*, volume 2, pages 65–75, 2013.

[6] Lucia Ballerini1, Xiang Lil, Robert B. Fisher, and Jonathan Rees. A query-by-example content-based image retrieval system of non-melanoma skin lesions. In *Medical Content-Based Retrieval for Clinical Decision Support*, pages 31–38. Springer Berlin Heidelberg, 2010.

[7] Chad Carson, Megan Thomas, Serge Belongie, Joseph M. Hellerstein, and Jitendra Malik. Blobworld: A system for region-based image indexing and retrieval. In *Visual and Information Systems*, pages 509–517. Springer Berlin Heidelberg, 1999.

[8] Amanbir Sandhu and Aarthi Kochar. Content based image retrieval using texture, color and shape for image analysis. In *International Journal of Computers and Technology*, volume 3, 2012.

[9] M V Sudhamani and C R Venugopal. Image: Region-based image retrieval toolbox. 2007.

[10] Chad Carson, Serge Belongie, Jitendra Malik, and Hayit Greenspan. Blobworld: image segmentation using expectation-maximization and its application to image querying. volume 24, pages 1026–1038, 2002.

[11] Jean-Yves Tinevez. Tree data structure as a matlab class.

APPENDIX A

PROGRAM CODE

```
%query image as input for cropping
sampleim = imread('./mri/10.jpg');
samplereg = imcrop(sampleim);

tic % start timing

sample = rgbhist_fast(samplereg,4); % calculate histogram of query region

imgno=2; %number of images in database
t = cell(imgno,1); % pointers to tree


%first image in database
img = imread('./mri/2.jpg');
k=3; %number of segments

[mask,mu,u,v]=EMSeg(img,k); %EM segmentation function

%split into quadrants and cropped regions as masks

rect1 = [0,0,size(img,1)/2,size(img,2)/2];
rect2 = [size(img,1)/2,0,size(img,1),size(img,2)/2];
rect3 = [0,size(img,2)/2,size(img,1)/2,size(img,2)];
rect4 = [size(img,1)/2,size(img,2)/2,size(img,1),size(img,2)];

i1 = imcrop(img, rect1);
i2 = imcrop(img, rect2);
i3 = imcrop(img, rect3);
i4 = imcrop(img, rect4);

m1 = imcrop(mask, rect1);
m2 = imcrop(mask, rect2);
m3 = imcrop(mask, rect3);
m4 = imcrop(mask, rect4);



t{1} =tree(img); %construct tree
t{1} =t{1}.addnode(1,i1); %add child nodes
t{1} =t{1}.addnode(1,i2);
t{1} =t{1}.addnode(1,i3);
t{1} =t{1}.addnode(1,i4);


%m1 - for first quadrant
```

```
row = cell(k,1);
col = cell(k,1);
convex = cell(k,1);
coord = cell(k,1); %coordinates
h = cell(k,1); %histogram


%find convex hull and histogram of that region

for i=1:k
    [row{i},col{i}] = find(m1==i);
    convex{i} = convhull(row{i},col{i});
    coord{i} = [col{i}(convex{i}), row{i}(convex{i})];
    x = roipoly(i1,coord{i}(:,1),coord{i}(:,2));
    x = uint8(x);
    x(:,:,2)=x(:,:,1);
    x(:,:,3)=x(:,:,1);
    z = x.*i1;
    h{i} = rgbhist_fast(z,4);
end

%storing in tree as child of quads
for i=1:k
    t{1} =t{1}.addnode(2,{h{i}, coord{i}});%for MBRs
end


%m2
row = cell(k,1);
col = cell(k,1);
convex = cell(k,1);
coord = cell(k,1); %coordinates
h = cell(k,1); %histogram


for i=1:k
    [row{i},col{i}] = find(m2==i);
    convex{i} = convhull(row{i},col{i});
    coord{i} = [col{i}(convex{i}), row{i}(convex{i})];
    x = roipoly(i2,coord{i}(:,1),coord{i}(:,2));
    x = uint8(x);
    x(:,:,2)=x(:,:,1);
    x(:,:,3)=x(:,:,1);
    z = x.*i2;
    h{i} = rgbhist_fast(z,4);
end

%storing in tree as child of quads
for i=1:k
    t{1} =t{1}.addnode(3,{h{i}, coord{i}});%for MBRs
end



%m3
row = cell(k,1);
col = cell(k,1);
convex = cell(k,1);
```

```
coord = cell(k,1); %coordinates
h = cell(k,1); %histogram


for i=1:k
    [row{i},col{i}] = find(m3==i);
    convex{i} = convhull(row{i},col{i});
    coord{i} = [col{i}(convex{i}), row{i}(convex{i})];
    x = roipoly(i3,coord{i}(:,1),coord{i}(:,2));
    x = uint8(x);
    x(:,:,2)=x(:,:,1);
    x(:,:,3)=x(:,:,1);
    z = x.*i3;
    h{i} = rgbhist_fast(z,4);
end

%storing in tree as child of quads
for i=1:k
    t{1} =t{1}.addnode(4,{h{i}, coord{i}});%for MBRs
end


%m4
row = cell(k,1);
col = cell(k,1);
convex = cell(k,1);
coord = cell(k,1); %coordinates
h = cell(k,1); %histogram


for i=1:k
    [row{i},col{i}] = find(m4==i);
    convex{i} = convhull(row{i},col{i});
    coord{i} = [col{i}(convex{i}), row{i}(convex{i})];
    x = roipoly(i4,coord{i}(:,1),coord{i}(:,2));
    x = uint8(x);
    x(:,:,2)=x(:,:,1);
    x(:,:,3)=x(:,:,1);
    z = x.*i4;
    h{i} = rgbhist_fast(z,4);
end

%storing in tree as child of quads
for i=1:k
    t{1} =t{1}.addnode(5,{h{i}, coord{i}});%for MBRs
end

time1= toc; %time until this point

disp(t{1}.tostring);  %display first tree
%%

%second image in database
img = imread('./mri/7.jpg');

[mask,mu,u,v]=EMSeg(img,k);

rect1 = [0,0,size(img,1)/2,size(img,2)/2];
```

```
rect2 = [size(img,1)/2,0,size(img,1),size(img,2)/2];
rect3 = [0,size(img,2)/2,size(img,1)/2,size(img,2)];
rect4 = [size(img,1)/2,size(img,2)/2,size(img,1),size(img,2)];

i1 = imcrop(img, rect1);
i2 = imcrop(img, rect2);
i3 = imcrop(img, rect3);
i4 = imcrop(img, rect4);

m1 = imcrop(mask, rect1);
m2 = imcrop(mask, rect2);
m3 = imcrop(mask, rect3);
m4 = imcrop(mask, rect4);


t{2} =tree(img);

t{2} =t{2}.addnode(1,i1);
t{2} =t{2}.addnode(1,i2);
t{2} =t{2}.addnode(1,i3);
t{2} =t{2}.addnode(1,i4);


%m1
row = cell(k,1);
col = cell(k,1);
convex = cell(k,1);
coord = cell(k,1); %coordinates
h = cell(k,1); %histogram


%find convex hull and histogram of that region

for i=1:k
    [row{i},col{i}] = find(m1==i);
    convex{i} = convhull(row{i},col{i});
    coord{i} = [col{i}(convex{i}), row{i}(convex{i})];
    x = roipoly(i1,coord{i}(:,1),coord{i}(:,2));
    x = uint8(x);
    x(:,:,2)=x(:,:,1);
    x(:,:,3)=x(:,:,1);
    z = x.*i1;
    h{i} = rgbhist_fast(z,4);
end

%storing in tree as child of quads
for i=1:k
    t{2} =t{2}.addnode(2,{h{i}, coord{i}});%for MBRs
end


%m2
row = cell(k,1);
col = cell(k,1);
convex = cell(k,1);
coord = cell(k,1); %coordinates
h = cell(k,1); %histogram
```

```
for i=1:k
    [row{i},col{i}] = find(m2==i);
    convex{i} = convhull(row{i},col{i});
    coord{i} = [col{i}(convex{i}), row{i}(convex{i})];
    x = roipoly(i2,coord{i}(:,1),coord{i}(:,2));
    x = uint8(x);
    x(:,:,2)=x(:,:,1);
    x(:,:,3)=x(:,:,1);
    z = x.*i2;
    h{i} = rgbhist_fast(z,4);
end

%storing in tree as child of quads
for i=1:k
    t{2} =t{2}.addnode(3,{h{i}, coord{i}});%for MBRs
end



%m3
row = cell(k,1);
col = cell(k,1);
convex = cell(k,1);
coord = cell(k,1); %coordinates
h = cell(k,1); %histogram


for i=1:k
    [row{i},col{i}] = find(m3==i);
    convex{i} = convhull(row{i},col{i});
    coord{i} = [col{i}(convex{i}), row{i}(convex{i})];
    x = roipoly(i3,coord{i}(:,1),coord{i}(:,2));
    x = uint8(x);
    x(:,:,2)=x(:,:,1);
    x(:,:,3)=x(:,:,1);
    z = x.*i3;
    h{i} = rgbhist_fast(z,4);
end

%storing in tree as child of quads
for i=1:k
    t{2} =t{2}.addnode(4,{h{i}, coord{i}});%for MBRs
end


%m4
row = cell(k,1);
col = cell(k,1);
convex = cell(k,1);
coord = cell(k,1); %coordinates
h = cell(k,1); %histogram


for i=1:k
    [row{i},col{i}] = find(m4==i);
    convex{i} = convhull(row{i},col{i});
    coord{i} = [col{i}(convex{i}), row{i}(convex{i})];
```

```
    x = roipoly(i4,coord{i}(:,1),coord{i}(:,2));
    x = uint8(x);
    x(:,:,2)=x(:,:,1);
    x(:,:,3)=x(:,:,1);
    z = x.*i4;
    h{i} = rgbhist_fast(z,4);
end

%storing in tree as child of quads
for i=1:k
    t{2} =t{2}.addnode(5,{h{i}, coord{i}});%for MBRs
end

time2 = toc; %time until this point
%%

%compute chi square distance for all histograms and query region histogram in leaf
alldist = zeros(1,3);
for l = 1:imgno
    iterator = t{l}.depthfirstiterator;
    for j = iterator
        if t{l}.isleaf(j)==1
            currenth = t{l}.get(j);
            currenth = currenth{1};
            a = chi_square_statistics(currenth',sample');
            alldist = [alldist;a j l];
        end
    end
end

alldist = alldist(2:end,:);

[mdis,mno] = min(alldist); %min distance value and quadrant number
mno = mno(1);
mi = alldist(mno,3); %image number
mno = alldist(mno,2);
match = t{mi}.get(mno); %match minimum distance with the leaf node pointer
finalpart = t{mi}.get(t{mi}.getparent(mno)); %final quadrant


%crop only region needed and display image
y = roipoly(finalpart,match{2}(:,1),match{2}(:,2));
y = uint8(y);
y(:,:,2)=y(:,:,1);
y(:,:,3)=y(:,:,1);
finalimg = y.*finalpart;

time3 = toc; %total time to run code

imshow(finalimg); %display final image
```