Medical Image Retrieval From Distributed Environment

M.S.Anbarasi
Department of Information Technology
Ponicherry Engineering College
Puducherry, India.

K.M.Mehata
Department of Computer Science & Engg.
College of Engg, Anna University
Chennai, India

S.Sandhya
Department of Information Technology
Ponicherry Engineering College
Puducherry, India.

V.Suganya
Department of Information Technology
Ponicherry Engineering College
Puducherry, India.

***Abstract*— Image retrieval from distributed database is one of the challenging tasks in recent researches. Unlike text retrieval through SQL, MYSQL, etc, image retrieval is not an easy task because of its storage space, color, shape and texture factor. We propose a technique of retrieving images from a distributed database environment by giving a region of an image as an input query. By applying segmentation techniques while retrieval, the efficiency of retrieval is enhanced. This can be applied in medical field to locate tumors and other pathologies, measure tissue volumes,computer-guided surgery,diagnosis,treatment planning and study of anatomical structure**

Keywords- Segmentation, edge detection, filters, feature representation, feature organization

I. INTRODUCTION

Image-based image retrieval, a technique which uses visual contents to search images from large scale image databases according to users' interests, has been an active and fast advancing research area. During the past decade, remarkable progress has been made in both theoretical research and system development. However, there remain many challenging research problems that continue to attract researchers from multiple disciplines. Early work on image retrieval can be traced back to the late 1970s. In 1979, a conference on Database Techniques for Pictorial Applications was held in Florence. Since then, the application potential of image database management techniques has attracted the attention of researchers. Text-based image retrieval uses traditional database techniques to manage images. Through text descriptions, images can be organized by topical or semantic hierarchies to facilitate easy navigation and browsing based on standard Boolean queries. It is difficult for the traditional text-based methods to support a variety of task-dependent queries. The difficulties faced by text-based retrieval became more and more severe. The efficient management of the rapidly expanding visual information became an urgent problem.

In 1992, the National Science Foundation of the United States organized a workshop on visual information .It was widely recognized that a more efficient and intuitive way to represent and index visual information would be based on properties that are inherent in the images themselves. Since then, research on content-based image retrieval has developed rapidly . Since 1997, the number of research publications on the techniques of visual information extraction, organization, indexing, user query and interaction, and database management has increased enormously. Similarly, a large number of academic and commercial retrieval systems have been developed by universities, government organizations, companies, and hospitals. Only Content-based image retrieval uses the visual contents of an image such as color, shape, texture, and spatial layout to represent and index the image. To retrieve images, users provide the retrieval system with example images or sketched figures. Recent retrieval systems have incorporated users' relevance feedback to modify the retrieval process in order to generate perceptually and semantically more meaningful retrieval results.

We now propose image retrieval system by giving segment of an image as input, thereby improving the enhancement of content-based image retrieval system. Here we apply in the case of retrieving a image from a distributed environment. Images are distributed in many nodes. A region of image is given as input and all similar images matching with the given region are retrieved with enhanced efficiency.

II. TERMINOLOGIES

Techniques covered in EIRDD literature for stock images do not meet retrieval reliability necessary for biomedical images. In this section, we present challenges, open problems, and lessons learned for each task in the EIRDD trail during our development and use of the research prototype system.

Segmentation:

Fully automated segmentation/feature extraction from medical images is a very challenging problem with no method directly applicable as found in the literature. Its quality is affected by three important factors; viz., technique, image quality, and image size/resolution. Hence, computer assisted segmentation is usually a more promising approach. We have faced significant problems with poor image quality in the digitized spine x-ray images where segmentation methods often confuse tissue and vertebra boundaries. We discovered that techniques that have been known to work well for smaller images often do not scale well when presented with higher resolution images. Poor segmentation results were observed on application of fairly robust color and texture segmentation techniques developed for 192x128 pixels Blob world images to the 2399x1637 pixel uterine cervix images from the NCI dataset.

Feature Representation

The dimensionality of the represented feature is highly correlated with the quality and inversely with efficiency of retrieval. Additionally, developers of a EIRDD system must ensure that the representation technique faithfully captures the image content/semantics. We learned this lesson during performance evaluation on retrieval quality of vertebrae with anterior osteophytes. The method selected for representing vertebra boundaries was Fourier descriptors following initial reduction in dimension using polygon approximation. It was found that while the method did fairly well (70% retrieval precision) on complete shape queries, the users were unable to focus on specific regions of the pathology. Intuitive modifications to the query shape were not comparably represented resulting in low quality retrieval. This led us to pursue research into *partial* or *incomplete* shape matching.

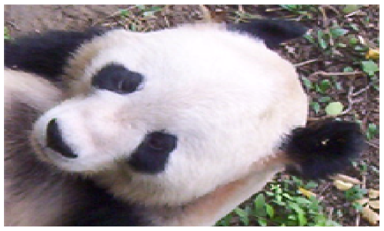
In a recent evaluation of retrieval performance improvement through the use of partial shape matching, it was observed that for pathological shapes the retrieval relevance improved by a factor of 2.2. That is, the average retrieval performance improved from 3.815 relevant matches with a standard deviation of 1.66 in the top 10 retrieved shapes to on average 8.125 relevant matches with a standard deviation of 0.25.

Feature organization:

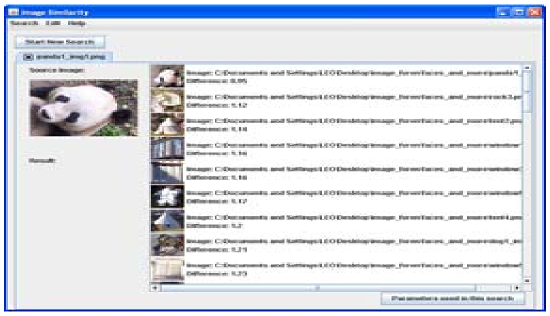
Our initial evaluation of hierarchical clustering showed that there is a significant performance improvement over linear searches. In a set of experiments on a 1298 shape database, on average only 176 leaf nodes needed to be accessed for a 13 nearest neighbor (most similar shapes) query. In scaling the range to 40 nearest neighbors, it needed to access an average of only 315.25 nodes. This is a significant improvement over 1298 accesses required in a linear search. It is, however, optimized for only one feature. Additionally, it uses shape descriptions with a fixed number of boundary points since it requires the similarity measure to be a metric. The Procuresses distance that we use as a similarity measure is a metric, but operates only on boundaries with fixed number of points. Fixed number of points in low dimension can affect image feature detail and consequently adversely affect retrieval quality.

III. DESCRIPTION

Fig 1.1 shows the flow diagram of how image is retrieved in our system. In the fig 1.2, the first step is selecting source image. The selected source image is searched with other target images and all the similar images are displayed. The comparison is done pixel by pixel. This is done to get very accurate search results. Even the size of the pixel for search can be changed.



This is the source image which is to be compared with the other images and the similar images will be displayed as shown below.



Result

Fig 1.2 Example of Image Similarity Search

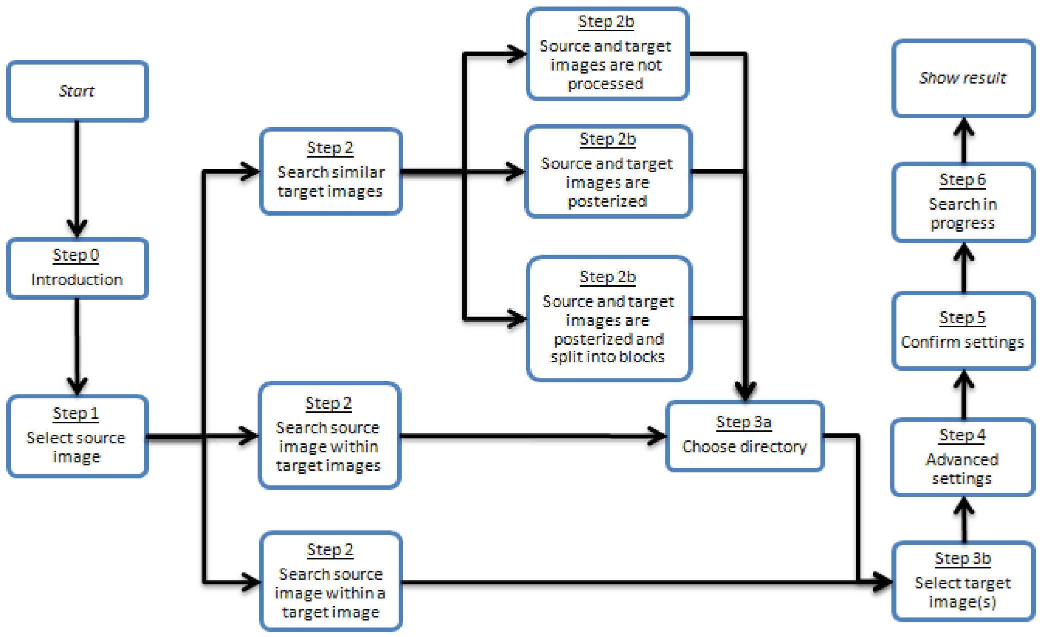

Start
Step 0
Introduction
Step 1
Select source image
Step 2
Search similar target images
Step 2
Search source image within target images
Step 2
Search source image within a target image
Step 2b
Source and target images are not processed
Step 2b
Source and target images are posterized
Step 2b
Source and target images are posterized and split into blocks
Step 3a
Choose directory
Step 3b
Select target image(s)
Step 4
Advanced settings
Step 5
Confirm settings
Step 6
Search in progress
Show result

Fig 1.1 Flow diagram of EIRDD system

IV **TECHNIQUES APPLIED**

Edge detection:

Edge detection is a problem of fundamental importance in image analysis. In typical images, edges characterize object boundaries and are therefore useful for segmentation, registration, and identification of objects in a scene. In this section, the construction, characteristics, and performance of a number of gradient and zero-crossing edge operators will be presented.

An edge is a jump in intensity. The cross section of an edge has the shape of a ramp. An ideal edge is a discontinuity (*i.e.*, a ramp with an infinite slope). The first derivative assumes a local maximum at an edge. For a continuous image $f(x, y)$, where *x* and *y* are the row and column coordinates respectively, we typically consider the two directional derivatives $\partial_x f(x, y)$ and $\partial_y f(x, y)$. Of particular interest in edge detection are two functions that can be expressed in terms of these directional derivatives: the gradient magnitude and the gradient orientation. The gradient magnitude is defined as $|\nabla f(x, y)| = \sqrt{(\partial_x f(x, y))^2 + (\partial_y f(x, y))^2}$

, and the gradient orientation is given by

$$\angle \nabla f(x, y) = \mathrm{ArcTan}(\partial_y f(x, y) / \partial_x f(x, y)).$$

Local maxima of the gradient magnitude identify edges in $f(x, y)$. When the first derivative achieves a maximum, the second derivative is zero. For this reason, an alternative edge-detection strategy is to locate zeros of the second derivatives of $f(x, y)$. The differential operator used in these so-called zero-crossing edge detectors is the Laplacian

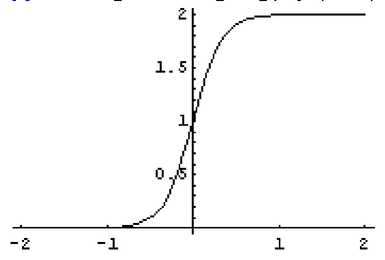
$$\Delta f(x, y) = \partial_{\{x,2\}} f(x, y) + \partial_{\{y,2\}} f(x, y).$$

This loads the package.

```
In[1]:=  << ImageProcessing`
```

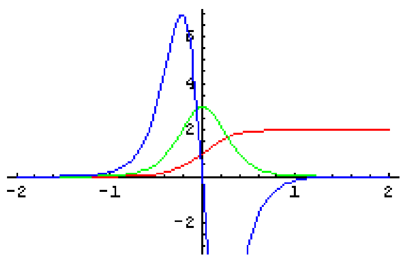
The following demonstration illustrates the principles of edge detection. We model a one-dimensional edge with a simple smooth function, for example, $f(t) = \mathrm{Tanh}(\alpha t)$. As the parameter alpha increases, the slope of the edge gets steeper. Here we use $\alpha = 3$.

```
In[2]:= Plot[1 + Tanh[3 t], {t, -2, 2}, PlotRange → All];
```



This evaluates and plots the first and second derivatives of our "edge" function.

```
Plot[Evaluate[{1 + Tanh[3 t], ∂_t Tanh[3 t], ∂_{t,2} Tanh[3 t]}], {t, -2, 2},
 PlotStyle → {Hue[0], Hue[1/3], Hue[2/3]}];
```



As expected, the result shows that maxima of the first derivative (green) and zero crossings of the second derivative (blue) locate the center of our model edge. It can also be shown that the zero crossings are independent of the steepness of the transition, while the gradient magnitude is directly related to the edge slope. In practice, finite difference approximations of first-order directional derivatives are used. These are represented by a pair of masks, say h_x and h_y. Formally these are linear-phase FIR filters. A convolution of the image with h_x and h_y gives two directional derivative images g_x and g_y respectively. The gradient image is traditionally calculated as $\nabla = \sqrt{g_x^2 + g_y^2}$, or alternatively using $\nabla = |g_x| + |g_y|$ [1]. A pixel location is declared an edge location if the value of nabla (at point *x*, *y*) exceeds some threshold. The location of all edge points constitutes an edge map. The selection of a threshold value is an important design decision that depends on a number of factors, such as image brightness, contrast, level of noise, and even edge direction. Typically, the threshold is selected following an analysis of the gradient image histogram. It is sometimes useful to calculate edge-direction information.

This is given by $\phi = \text{ArcTan}\left[-\frac{g_x}{g_y}\right]$.

Smoothing and Differentiating Filters

LaplacianFilter[α]	returns a 3 × 3 linear phase FIR filter approximation of the Laplacian, a second derivative operator. The default value of α is α -> 2.
GaussianFilter[n_1, n_2, ...]	returns a Gaussian smoothing filter of dimensions $n_1 \times n_2 \times \ldots$ with standard deviation $\frac{n_i - 1}{6\sqrt{2}}$ at level *i*.
LoGFilter[n_1, n_2, ...]	returns a Laplacian-of-Gaussian edge operator of dimensions $n_1 \times n_2 \times \ldots$ with standard deviation $\frac{n_i - 1}{6\sqrt{2}}$ at level *i*.
ZeroCrossing[*img*]	returns a binary image that shows the position of the zero-crossings in *img*.

VI LAPLACIAN BASED EDGE DETECTION

The edge points of an image can be detected by finding the zero crossings of the second derivative of the image intensity. The idea is illustrated for a 1D signal in Figure 5.3.1. However, calculating 2nd derivative is very sensitive to noise. This noise should be filtered out before edge detection. To achieve this, Laplacian of Gaussian is used.

This method combines Gaussian filtering with the Laplacian for edge detection.

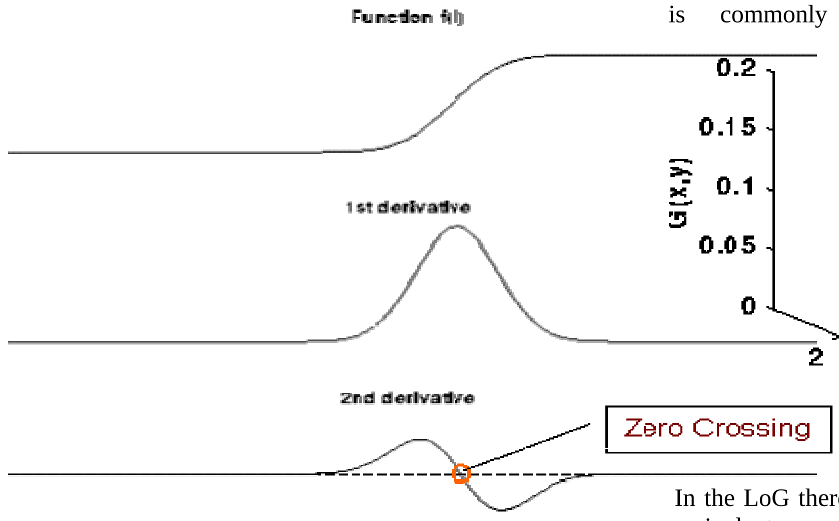


Figure 5.3.1 1st and 2nd derivative of an edge illustrated in one dimension.
The first graph represents an edge in 1D.

In Laplacian of Gaussian edge detection there are mainly three steps: Filtering Enhancement and detection.

Gaussian filter is used for smoothing and the second derivative of which is used for the enhancement step. The detection criterion is the presence of a zero crossing in the second derivative with the corresponding large peak in the first derivative.

In this approach, firstly noise is reduced by convoluting the image with a Gaussian filter. Isolated noise points and small structures are filtered out. With smoothing; however; edges are spread. Those pixels, that have locally maximum gradient, are considered as edges by the edge detector in which zero crossings of the second derivative are used. To avoid detection of insignificant edges, only the zero crossings whose corresponding first derivative is above some threshold, are selected as edge point. The edge direction is obtained using the direction in which zero crossing occurs.

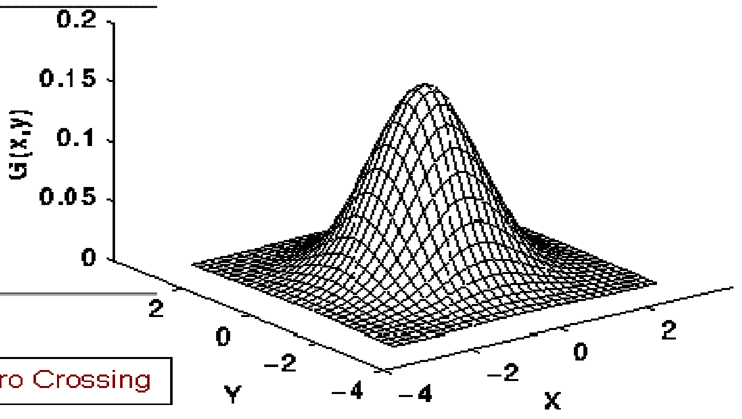
The output of the Laplacian of Gaussian (LoG) operator; h(x,y); is obtained by the convolution operation:

$$\begin{aligned} h(x,y) &= \Delta^2[g(x,y) * f(x,y)] \\ &= [\Delta^2 g(x,y)] * f(x,y) \end{aligned}$$

where

$$\Delta^2 g(x,y) = \left(\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right)^{-(x^2+y^2)/2\sigma^2}$$

is commonly called the Mexican hat operator.



In the LoG there are two methods which are mathematically equivalent:

1) Convolve the image with a gaussian smoothing filter and compute the Laplacian of the result,

2) Convolve the image with the linear filter that is the Laplacian of the Gaussian filter.

This is also the case in the LoG. Smoothing (filtering) is performed with a Gaussian filter, enhancement is done by transforming edges into zero crossings and detection is done by detecting the zero crossings.

V Filters

Filters are useful in applications such as games where images need to be generated on the fly, or where it's quicker to generate them rather than downloading them. For instance, it's quicker to download one image and rotate it several times than to download several separate images.

ChannelMixFilter

This filter mixes the RGB channels of an image. For each channel, you specify how much of the other two channels to mix in.

DiffusionFilter

This filter converts an image to a specified number of colours using Floyd-Steinberg error diffusion dithering. It generally produces better results than ordered dithering, with less distinct "patterning". You can choose straight or serpentine dithering and the number of color levels to dither to.

Parameters:

- int[] matrix - The dither matrix
- int levels - The number of levels to dither to
- boolean colorDither - true for color dithering, false for monochrome
- boolean serpentine - true to use a serpentine pattern which reduces "cascading" effects

DitherFilter

It's often necessary to reduce the number of colours used in an image, usually in order to display iton a device which has colour limitations.This filter dithers an image to a specified number of colours. You supply the number of colors and the dither matrix. There are built-in matrices for common patterns such as 4x4 magic square and ordered dither etc.

Here's the simplest 2x2 dither matrix:

```
0 2
3 1
```

Parameters:

- int[] matrix - The dither matrix
- int levels - The number of levels to dither to

ExposureFilter

This filter simulates changing the exposure of an image.

GrayFilter

This filter produces a 'grayed-out' version of an image. It's useful for producing disabled icons for buttons and simply works by averaging each pixel with white. There are no parameters to this filter.

GrayscaleFilter

This filter converts an image to a grayscale image. To do this it finds the brightness of each pixel and sets the red, green and blue of the output to the brightness value. But what is the brightness? The simplest answer might be that it is the average of the RGB components, but that neglects the way in which the human eye works. The eye is much more sensitive to green and red than it is to blue, and so we need to take less acount of the blue and more account of the green. The weighting used by GrayscaleFilter is: luma = 77R + 151G + 28B

HSBAdjustFilter

This filter adds or subtracts a given amount from each of the hue, saturation and brightness channels of an image.You can use it to tint an image or to make the colours more saturated for example.

Parameters:

- float hFactor - Amount to shift hue
- float sFactor - Amount to shift saturation
- float bFactor - Amount to shift brightness

InvertAlphaFilter

This filter will invert the alpha channel of an image. This is not normally terribly useful, but has its uses on accasion. There are no parameters to this filter.

MapColorsFilter

A filter which replaces one color by another in an image. This is frankly, not often useful, but has its occasional uses when dealing with GIF transparency and the like.

MaskFilter

This filter simply applies a 32-bit mask to each ARGB pixel in an image. You can use it for things like masking out individual channels in an image or posterizing.

PosterizeFilter

This filter posterizes an image by quantizing each channel to a specified number of levels. This is generally an ugly way to reduce colours, but is often used as a special effect.

QuantizeFilter

QuantizeFilter quantizes an image to a specified number of colors using an octtree algorithm. You can specify the number of color and whether to use Floyd-Steinberg error diffusion dithering or not. This is a very useful filter to use when you want to convert an image to use a colormap, for example, when you need to write an image out as a GIF file.

Parameters:

- int levels - The number of levels to dither to
- boolean dither - true to use dithering to imrpove the quality
- boolean serpentine - true to use a serpentine pattern which reduces "cascading" effects

THRESHOLDFILTER

This filter converts an color image black and white by changing all pixels lighter than a threshold to white and all pixel darker than the threshold to black.

TRITONEFILTER

This filter creates a tritone version of an image. You supply three colors for the shadows, midtones and highlights and the filter interpolates between the colors according to the brightness of each pixel.

VI OVERVIEW OF OUR METHOD

Fig 1.3 shows the overview of the proposed method. Our system works in two phases. In the learning phase, discrete wavelet transform is applied to each image in a database, and texture features for pixels are computed, then, the image is segmented into regions hierarchically according to the computed features (the left part of Fig 1.3). Each segmented regions is indexed by the average of the features over the pixels in the region (the middle part of Fig 1.3). Although we treat only monochromatic images in this paper, we think that it is possible to expand our method into a color-image retrieval system by using a wavelet based method for colored textures. In the retrieval phase, a user specifies a region in a query image (the right part of Fig 1.3). Wavelet transform is applied to the query image, and texture feature for the specified region is computed. Then images whose features are close to the feature of the specified region in the index space are retrieved.

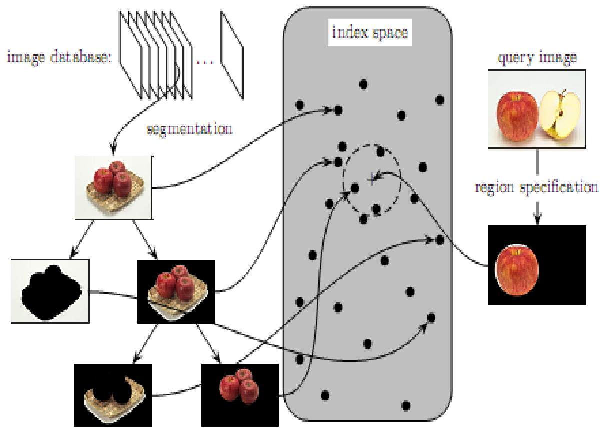


Fig 1.3 Overview of image retrieval system

Indexing

By the texture segmentation described in the previous section, each image in a folder is segmented into regions, and a hierarchy of the regions is constructed as shown in the left part of Fig 1.3 The hierarchy of regions can be represented by a binary tree, in which a node with two child nodes represents a region obtained by merging the two regions correspond to the two child nodes, and the root node represent the whole image. For each region, the system assigns an index which I over all pixel in the region. When we use level n wavelet decompositions indices are (3n + 1) dimensional vectors. In general, wavelet coefficients obtained by decomposing an image are not invariant with respect to translation rotation, and scaling of the image. Most previous content based image retrieval techniques which use wavelet transform suffer from the lack of invariance in wavelet coefficients. For region-based image retrieval, this problem will be much worse. However, the method will work only for very simple image in which only a few objects exist. In our method, we use a texture feature, which will be preserved well with respect to image translation or rotation. So, we expect that the problem will be alleviated.

Retrieval Phase

In the retrieval phase, a user selects a query image and specifies a region in the image using GUI (Graphical User Interface). Then, a wavelet decomposition of the whole query image is computed and a feature vector is computed by averaging feature vectors over all pixels in the region. This averaged feature vector is a query key. The system simply outputs specified number of images which have regions whose indices are closest to the feature vector (query key) in the Euclidean distance. The images are ordered so that the image which has the closest region comes first. Even if an image has some regions which are close to the query key, the image appears only once in the output. The user can check the regions which match the query key in the output images. This is an advantage of region-based image retrieval. For a practical image retrieval system, it should not take long time to process a query, even when a large number of images are stored in the database. Many techniques to boost the speed of index search proposed in the literature of database. In our method, SR-tree is used, to shorten the time for index search, in which an index space is split into hierarchical regions to restrict the space actually inspected.

Image retrieval from distributed environment

The state-of-the-art approach for the creation and re- trieval of image databases is based on the comparison of a priori defined features, which can be directly derived from the image raw data when updating the database. The extracted features can be combined and weighted in different ways and describe characteristic content properties. At query time the user creates a sample sketch or loads an image, which is subsequently processed in the same manner as the images stored in the database. The similarity degree of a query image and the target images is then determined by calculation of a multidimensional distance between the corresponding features using adapted and weighted versions of well-known metrics and functions. Acceptable system response times are achieved, because no further processing of the image raw data is necessary during the retrieval process. The straightforward integration in existing database systems is a further advantage of this approach. Extraction of simple features results in a disadvantageous reduction of the image content. Important details like objects, topological information etc. are not sufficiently considered in the retrieval process making a precise detail search difficult. Furthermore, it is not clear, whether the known, relatively simple features can be correctly combined for the retrieval of all kinds of images. Therefore, image retrieval with dynamically extracted features is necessary. Dynamical retrieval is the process of analysis, extraction, and description of any manually selected image elements, which are subsequently compared to all image sections in the database. Other regions of the query image and the object background are not regarded, so that a detail search can be performed.

VII CONCLUSIONS

This paper presents our experiences, techniques adopted, and lessons learned in continuing research towards enabling Image Retrieval by giving a region of image as input from distributed databases. It also presents the various techniques adopted in retrieving the image efficiently without loss of image parameters.

REFERENCES

[1] Tao Xie, and Xiao Qin, "Security-Aware Resource Allocation for Real-Time Parallel Jobs on Homogeneous and Heterogeneous Clusters", *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 5, may 2008 pp 682-697.

[2] Sameer K. Antani, Thomas M. Deserno, L. Rodney Long, Mark O.Guld, Leif Neve1, George R. Thoma, "Interfacing Global and Local CBIR Systems for Medical Image Retrieval".

[3] MVSudhamani,CR,Venugopal,"MULTIDIMENSIONAL INDEXING STRUCTURES FOR ONTENT-BASED IMAGE RETRIEVAL: A SURVEY", *International Journal of Innovative Computing, Information and Control ICIC International °c 2008 ISSN 1349-4198* Volume 4, Number 4, April 2008.

[4] Remco C. Veltkamp, Mirela Tanase" Content-Based Image Retrieval Systems: A Survey" Technical Report UU-CS-2000-34, October 2000.

[5] www.Image Processing Fundamentals – Segmentation.htm

[6] Thang, Tu-Jih "Advance in image and video segment" USA Idea group published 2006

[7] A. Hoover, V.Kouznetsover and H.Holdborum,"Locating blood vessels in retrieval images by piecewise threshold probing of matched filter response", *IFFT Tran, Medical image* Vol1b PP(2003-2007)

[8] Gonzalez, R. C., and Woods, R. E. *Digital Image Processing* (Reading, MA: Addison-Wesley, 1992).

[9] Marr, D., and Hildreth, E. "Theory of Edge Detection," *Proceedings of the Royal Society London* 207 (1980) 187-217.

[10] Haralick, R. M., and Shapiro, L. G. *Computer and Robot Vision*, vol.1 (Reading, MA: Addison-Wesley, 1992).