

# Content-based Image Retrieval in Image World

Susan Laraghy 0584622

A thesis submitted in partial fulfilment  
for the degree of  
Masters of Computer Science

Leiden University

August 19, 2010

Universiteit Leiden

# Abstract

This thesis presents an interface to investigate the possibilities for a graphic user interface for an image database to be mapped onto an OpenGL surface. The content of the image was used to determine the placement within the interface and thus the images would be clustered according to their similarity. This would allow the user to browse over the images fluidly and interactively and would also provide relevance feedback. The proposed method focuses upon texture classification and combines Fractal Dimension utilising the Improved Box Counting method and Dominant Colour Diamond Segmentation in order to classify the images according to texture and colour. To determine the feasibility of the fractal method, a comparison was made to the Local Binary Pattern algorithm. These methods were subsequently tested in a usability test to ascertain the effectiveness of these methods in real world situations.

*Keywords:* image clustering, content based image retrieval, visualization image retrieval, color, fractal dimension, improved box counting, differential box counting.

# Acknowledgements

The work carried out in this thesis was done so under the supervision of Dr. Michael S. Lew. I am grateful to him for his advice, ideas, comments and for the constructive criticism I received. I would also like to thank all those who gave up so much time in order to partake in the usability tests.

# Contents

<b>List of Figures</b>	<b>iv</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Background</b>	<b>2</b>
2.1 The need for image data management . . . . .	2
2.2 Content Based Image Retrieval . . . . .	3
2.3 Previous Methods in Texture Analysis . . . . .	4
<b>3 Approach</b>	<b>6</b>
3.1 Colour Retrieval . . . . .	6
3.1.1 Diamond Segmentation . . . . .	7
3.1.2 Further Refinement . . . . .	7
3.2 Texture Retrieval . . . . .	8
3.2.1 Fractal Dimension - Box Counting . . . . .	9
3.2.2 Fractal Dimension - Differential Box Counting . . . . .	14
3.2.3 Fractal Dimension - Improved Differential Box Counting . . . . .	14
3.3 Local Binary Patterns . . . . .	16
<b>4 Implementation</b>	<b>18</b>
<b>5 Evaluation and Results</b>	<b>21</b>
5.1 Observations . . . . .	21
5.2 Case Study . . . . .	22
<b>6 Conclusion</b>	<b>28</b>
<b>References</b>	<b>29</b>

## List of Figures

1	Estimated Image growth. . . . .	2
2	Diamond Segmentation. . . . .	7
3	Image with dominant colour shown as border . . . . .	8
4	Image shown with border of adjusted dominant colour . . . . .	8
5	Julia set with a Hausdorff Dimension of 2 . . . . .	9
6	Estimation of fractal dimension. The definition by Hausdorff (a). The Box Counting approach (b). . . . .	10
7	Gaussian blur mask passed over image . . . . .	11
8	Sobel Masks . . . . .	12
9	Any edge angle within 22.5 degrees of one of the possible angles is changed to that value. . . . .	12
10	Original image used as input. . . . .	13
11	Result of Canny edge detector . . . . .	13
12	Result of Fractal Box Count. . . . .	13
13	Determination of the number of boxes by the Fractal DBC method	15
14	Box height calculation in the Fractal DBC method . . . . .	15
15	Example of Local Binary Pattern Calculation . . . . .	17
16	Graphic user interface of final implementation . . . . .	18
17	Interface shown in order to limit/increase search space . . . . .	20
18	Misclassification using Local Binary Pattern . . . . .	22
19	Image 8 . . . . .	24
20	Image 20 . . . . .	24
21	Average search time per image . . . . .	25
22	Average search time per image when performing a texture only search	27

# 1 Introduction

One of the tools that will be essential in the future for electronic publishing as well as in many other areas, is a powerful image retrieval system. Ideally, one should be able to search an image database for images that are relevant to his or her needs or that convey the desired information or mood. Most image retrieval systems associate keywords or text with each image and require the user to enter a keyword or textual description of the desired image. This text-based approach has numerous drawbacks, namely that associating keywords or text with each image is a tedious task and some image features may not be mentioned in the textual description or be difficult to describe. In an effort to overcome these problems and improve retrieval performance, researchers have focused more and more on content-based image retrieval.

Content-based image retrieval (CBIR), is a technique for retrieving images on the basis of automatically derived primitive features such as colour, texture and shape. The search will analyse the actual contents of the image rather than use the metadata such as keywords, tags, and/or descriptions associated with the image. It is hoped that a system that can filter images based on their content will provide better indexing and return more accurate results. Much research has been conducted in the past two decades into these primitive feature descriptors and their effectiveness.

This paper presents a system for a graphic user interface combining a number of recent ideas concerning colour and texture analysis that have been described in previous literature. Instead of using these feature characteristics as a means to retrieve a desired image, here they are used as a means of classifying each image so as to arrange and cluster the images within the interface. While slight variations to a straight-forward approach to colour analysis is implemented, the texture descriptors based upon Fractal Dimension demonstrate the larger advances in this area. The performance and effectiveness of the system is analysed by subjecting it to various evaluations by a typical user group.

In the Section 2, background information concerning content-based image retrieval is provided along with a brief description of previous methods. Section 3 provides information related to the methods and algorithms used in the implementation which is then discussed in Section 4. Section 5 contains information regarding the experiments and case study, along with providing the results, followed by the conclusion in Section 6.

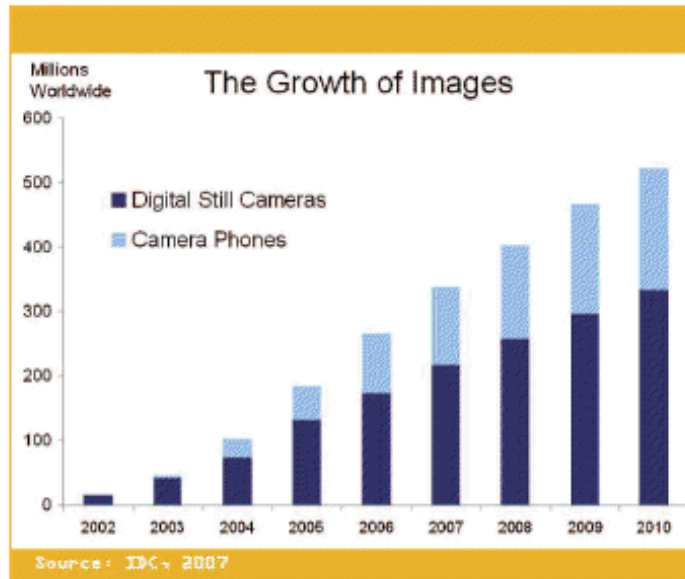


Figure 1: Estimated Image growth.

## 2 Background

### 2.1 The need for image data management

Large collections of digital images are created for use in many areas, for example commerce, government, hospitals and academia for a variety of uses such as art collections, photographic archives, retail catalogs, medical diagnosis, crime prevention, military purposes, intellectual property, architectural and engineering design, and geographical information and remote sensing systems. Many of these collections are digitised existing analogue photographs, diagrams, drawings, paintings and prints, however fuelled by the rapid growth of the World-Wide Web and the development and availability of digital cameras in the past years, interest in digital imagery has increased enormously. It is estimated that about 500 million images will be taken this year [1] (see Figure 1), and each year not only the number of images taken increase, but also the resolution of the pictures and the megabytes per image. Digital imagery has resulted in an explosion in the volume of image data required to be organised.

Often the only way of searching these collections is by browsing or by keyword indexing, however every image in the database requires a description or associated keywords that are entered by a human. This can be an extremely tedious and

expensive task and impractical on very large databases, or for images that are generated automatically such as those from surveillance cameras for example. It is also possible to miss images that use different synonyms in their descriptions or it may not capture every keyword that describes the image. The problems of image retrieval are becoming widely recognised, and the search for solutions an increasingly active area for research and development. An indication of the increased interest in the subject can be gained from the amount of scientific papers appearing on the subject over the years. During the decade from 1990–1999, a total of 974 papers appeared on the subject. This increased to 12,235 during the ensuing decade, and 912 papers have been published on the subject in the first half of 2010 alone.

Due to the limitations inherent in a metadata-based system and the problems associated with traditional methods of image indexing as previously stated, there is a growing interest techniques for retrieving images on the basis of automatically derived features such as colour, texture and shape. This is now generally referred to as Content Based Image Retrieval (CBIR).

## 2.2 Content Based Image Retrieval

The term Content-Based Image Retrieval (CBIR) seems to have originated in 1992, when it was used by T. Kato [2] to describe experiments into automatic retrieval of images from a database, based on the colours and shapes within the image. Since then, the term has been used to describe the process of retrieving desired images from a large collection on the basis of syntactical image features. The techniques, tools and algorithms that are used originate from fields such as statistics, pattern recognition, signal processing, and computer vision.

Image queries are characterised into three levels of abstraction: primitive features such as colour or shape, logical features such as the identity of objects shown, and abstract attributes such as the significance of the scenes depicted. While Content-Based Image Retrieval systems operate most effectively at the lowest of these levels, most users demand higher levels of retrieval, demanding research to diminish the gap between the effectiveness of these features and the requirements of the user.

Typically in Content-Based Image Retrieval, comparisons are made from an example image and an image from the database by using an image distance measure. An image distance measure compares the similarity of two images in various dimensions such as colour, texture, shape, among others. For example an exact match with the query based upon the considered dimensions is signified by

0. A value greater than 0 therefore indicates various degrees of similarities between the images. Search results then can be sorted based on their distance to the queried image, and the best matches are returned.

Computing distance measures based on colour similarity is most often computed by a colour histogram for each image that identifies the proportion of pixels within an image holding specific values. Current research is attempting to segment colour proportion by region and by spatial relationship among several colour regions. Examining images based on the colours is one of the most widely used techniques because it does not depend on image size or orientation. Although not the only technique in practice, colour searches will usually involve comparing colour histograms.

Texture measures look for visual patterns in images and how they are spatially defined. Texture is a difficult concept to represent and the identification of specific textures in an image is usually achieved by modelling texture as a two-dimensional grey level variation. The relative brightness of pairs of pixels is computed such that degree of contrast, regularity, coarseness and/or directionality may be estimated.

Shape does not refer to the shape of an image but to the shape of a particular region that is being sought out. Shapes will often be determined first applying segmentation or edge detection to an image or by using shape filters to identify given shapes of an image. In some cases accurate shape detection will require human intervention because methods like segmentation are very difficult to completely automate. Therefore as this was to be a fully-automated classification system, shape was not included in the feature descriptors in this system.

## **2.3 Previous Methods in Texture Analysis**

Texture analysis plays an important role in many image processing tasks and is an important fundamental problem in computer vision. There are a great number of references for texture analysis and particularly texture classification. A central point in texture analysis is the definition of good features to characterise textures, that can be useful for content-based retrieval.

One of the first most important texture classification methods in image processing is the Gabor filter [3] which is a linear filter used for edge detection. The Gabor Filters have received considerable attention because the characteristics of certain cells in the visual cortex of some mammals can be approximated by these filters, and it has been found to be particularly appropriate for texture representation and discrimination. A set of Gabor filters with different frequencies and orientations

may be helpful for extracting useful features from an image. An example of how this may be used in Content-Based Image Retrieval is in the case of Azencott et al [4], where a distance measurement between Gabor features was developed.

Local Binary Patterns (LBP) is a type of feature used for classification in computer vision. The LBP operator was first introduced as a complementary measure for local image contrast (Harwood et al. 1993, Ojala et al. 1996) [5] [6], and has since been found to be a powerful feature for texture classification. The first version of the operator worked with the eight neighbours of a pixel, using the value of the center pixel as a threshold. An LBP code for a neighbourhood was produced by multiplying the thresholded values with weights given to the corresponding pixels, and summing up the result (see Figure15). This is further explained in Section 3.3.

The LBP texture analysis operator is defined as a greyscale invariant texture measure, derived from a general definition of texture in a local neighbourhood. For each pixel in an image, a binary code is produced by thresholding its value with the value of the centre pixel. A histogram is created to collect the occurrences of different binary patterns. The basic version of the LBP operator considers only the eight neighbours of a pixel, but the definition has been extended to include all circular neighbourhoods with any number of pixels.

Through its extensions, the LBP operator has been made into a really powerful measure of image texture, showing excellent results in terms of accuracy and computational complexity in many studies. One of the most important properties of the LBP texture descriptor is its tolerance against illumination changes. Another important property is its computational simplicity, which makes it possible to analyse images in challenging real-time settings.

The LBP method has already been used in a large number of applications all over the world, including visual inspection, image retrieval, remote sensing, biomedical image analysis, motion analysis, environment modelling, and outdoor scene analysis. In recent years a good deal of attention with regards to research has been conducted in using the LBP method for face image analysis and face recognition [7–9].

In recent years more attention has been focused upon Fractal Dimension as a means for a texture descriptor to analyse images. This is further explained in Section 3.2.

## 3 Approach

Extracting features from images is always the first and most important step in a CBIR system, which has a direct effect on the system. In order to cluster images in an intuitive manner, the decision was taken to focus upon the colour and texture features of the images. These two characteristic image features were used to map the images accordingly upon the x- and y-axes respectively.

### 3.1 Colour Retrieval

Of all the features used in a CBIR system, colour is the most common. Not only is this due to the relative ease of implementation over other features and to the inherent human ability to define colours but it is also due to the fact that colour features are robust to effects such as noise, image degradation, resolution, orientation, and changes in object geometry. In the past various representation schemes such as colour histograms, colour moments, colour edge orientation, colour texture and colour correlograms [10] [11], have been proposed as a basis for various colour-based search systems. These search systems are based upon such colour systems as RGB, HSV, LAB, LUV, etc. The approach described in this paper utilises colour histograms and the HSV colour system.

The **H**(ue) **S**(aturation) **V**(alue or Lightness or Intensity) colour model corresponds most closely to the human perception of colour and is therefore a very intuitive colour model. For the purpose of plotting images along an axis within a visceral user interface, HSV was the obvious choice as only the value of hue need be utilised leaving an absolute value between 0 and 360 for each image. The disadvantage of using the HSV system is that the hue is unstable as the saturation value approaches zero due to “the non-removable singularities in the non-linear transformation, where a small perturbation of the input RGB-values can cause a large jump in the transformed values” [12].

The hue of each pixel was extracted from the RGB-values for that particular pixel using the following formula:

$$H = \arctan \left( \frac{\sqrt{3}(G - B)}{(R - G) + (R - B)} \right) \quad (1)$$

From these hue values a colour histogram was built based upon the occurrence of each hue within the image, and the dominant colour for that particular image could easily be discovered by determining the peak within the histogram.



Figure 2: Diamond Segmentation.

### 3.1.1 Diamond Segmentation

In order to further refine and improve upon this simple method, the A region based dominant colour descriptor approach detailed by XiaoFu Lee and Qian Yin [13] was also implemented.

Humans have a tendency to focus the most important object into the centre of an image, and also begin to recognise objects from this point, just as the corners of an image are subsequently ignored. To reflect the behaviour and habits of human perception, a method called Diamond Segmentation was developed. Hereby the 4 midpoints of each side of the image were connected to form a polygon. The area within this diamond covers the most important information in an image and is therefore given the weight of 0.75 whereas the information in the background (the four corners) is weighted by 0.25 giving preference to the colour of the central object(s) within the image. This process is demonstrated in Figure 2.

### 3.1.2 Further Refinement

Within a certain image it may sometimes be the case that the human eye may detect the dominant hue as one hue, but that in reality it consists of a great deal of slightly varying hues, and that the discovered dominant hue may exist as a peak of one particular but very different hue (a solid background hue for example). To accommodate for these occurrences and to eliminate these anomalies, each hue value was expressed as the mode of the neighbouring hues using the formula

$$H_i = \frac{1}{n} \sum_{j=i-n/2}^{i+n/2} x_j \quad (2)$$

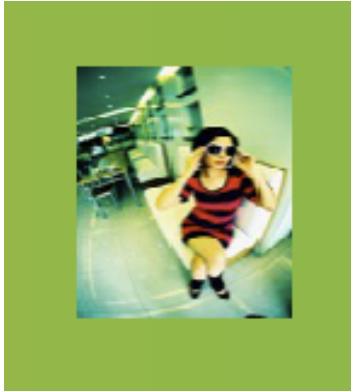


Figure 3: Image with dominant colour shown as border

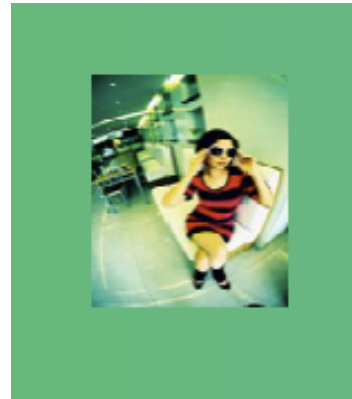


Figure 4: Image shown with border of adjusted dominant colour

where by  $n$  is a constant, and through experimentation was found to be most effective when  $n = 10$ . Figure 3 shows such an image on a background of what was calculated as the most dominant hue, however once corrected by the mode of its neighbours, it can be seen in Figure 4 depicted on the background that most fits all the ensuing shades within the image.

### 3.2 Texture Retrieval

Texture is the characteristic appearance of a surface having a tactile quality and can consist of very small elements such as sand, or of much larger elements. It may also be formed by a single surface with variations in shape, illumination, shadows, absorption and reflectance. In digital images, the characteristics of a texture can be recognised through variations in the captured intensities or colour. Differences in image pixels provide a practical means of analysing the textural properties of objects.

Fractal feature analysis in image processing has received increasing attention in recent years. Fractal theory has become widely used in the region of image processing, and much research has been devoted to this area [15–22]. The basic principle to estimate the fractal dimension is based on the concept of self-similarity and is a statistical quantity that gives an indication of how completely a fractal appears to fill a space as one zooms down to finer and finer scales.

A bounded set  $A$  is said to be self-similar if  $A$  is the union of a number ( $N_r$ ) of non-overlapping scaled copies of itself where  $r$  is the scaling factor. The fractal

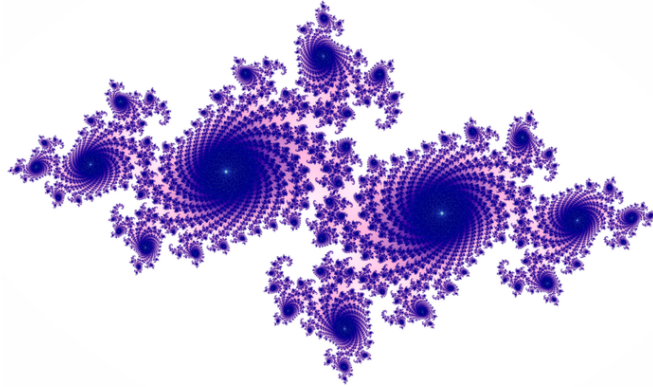


Figure 5: Julia set with a Hausdorff Dimension of 2

dimension  $D_f$  of  $A$  can be calculated by

$$D_f = \frac{\ln N(r)}{\ln(1/r)} \quad (3)$$

The differences of surface roughness between a man-made shape and a nature scene forms the basis for the fractal algorithm [25]. A scene of nature generally has a more complex surface texture and irregular shape and so has a greater fractal dimension. The converse of is therefore usually also true with the man-made shape of simpler texture and shape returning a lower fractal dimension. Fractal dimension is defined via various methods, the most important being the Rényi dimension, the Hausdorff dimension, and the packing dimension.

The Hausdorff dimension [14] of an  $n$ -dimensional vector space equals  $n$ . This means, for example the Hausdorff dimension of a point is zero, of a line is one, and of a 2-dimensional plane is two. The fractal complexity like that of a Julia set (see Figure 5) would also equate to two. The Hausdorff dimension is therefore bound by  $1 \leq HD \leq 2$  where  $HD$  stands for Hausdorff dimension.

### 3.2.1 Fractal Dimension - Box Counting

The Box Counting approach is based upon the Hausdorff Dimension and is therefore commonly known as this method [16]. Whereby the Hausdorff dimension uses the notion of (overlapping) balls covering an area of various radii, in the Box Counting method the number of boxes needed to cover a fractal (or pattern within an image) is counted (see Figure 6). This Box Counting approach is the most commonly used due its ease of implementation.

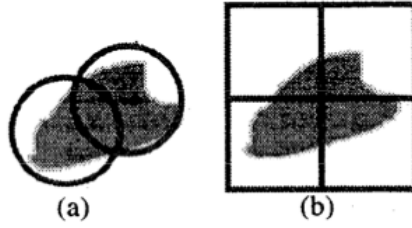


Figure 6: Estimation of fractal dimension. The definition by Hausdorff (a). The Box Counting approach (b).

According to this algorithm, an image of size  $M \times M$  is systematically partitioned into  $s \times s$  grids whereby  $1 < s \leq M/2$ . For each successive grid, the number of boxes  $N(r)$  of size  $r = s/M$  which are needed to cover the texture within the image are counted. The fractal dimension is defined as:

$$D_f = \lim_{r \rightarrow 0} \frac{\ln N(r)}{\ln(1/r)} \quad (4)$$

For the process of determining the fractal dimension according to the Box Counting method, first all colour information must be discarded from the image. The conversion of an image from RGB colour space to HSV also supplies the greyscale image by means of the Value calculation which for each pixel is simply:

$$V = \frac{R + G + B}{3} \quad (5)$$

From this greyscale image, edge detection must then be executed.

### Canny Algorithm

Edges characterise boundaries and are therefore a problem of fundamental importance in image processing. Edges in images are areas with strong intensity contrasts — a jump in intensity from one pixel to the next. Edge detecting an image significantly reduces the amount of data and filters out useless information, while preserving the important structural properties in an image.

This may be done via various methods however the Canny edge detection algorithm [26] is known to many as the optimal edge detector. Canny's aim was to develop an edge detection algorithm that would have good detection (mark as many real edges in the image as possible), good localisation (edges marked should

$$\frac{1}{159}$$

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

Figure 7: Gaussian blur mask passed over image

be as close as possible to the edge in the real image) and minimal response (a given edge in the image should only be marked once, and where possible, image noise should not create false edges). This therefore is a multi-stage algorithm to detect a wide range of edges in images.

As the Canny edge detector is susceptible to noise present on raw unprocessed image data, the first step is to remove it by applying a Gaussian filter over the entire image. This is done by passing a  $5 \times 5$  mask (as seen in Figure 7) over the whole image. Each pixel is redefined as the sum of the pixel values in its  $5 \times 5$  neighbourhood times the corresponding Gaussian weight, divided by the total weight of the whole mask. The result is a slightly blurred version of the original which is not affected to any significant degree by a single noisy pixel.

The algorithm then uses Sobel masks [27] to find the edge gradient strength and direction for each pixel. First the Sobel masks are applied to the  $3 \times 3$  pixel neighbourhood of the current pixel, in both the x and y directions as seen in Figure 8. Then the sum of each mask value times the corresponding pixel is computed as the  $Gx$  and  $Gy$  values, respectively. The edge strength is therefore computed as follows:

$$E = \sqrt{Gx^2 + Gy^2} \quad (6)$$

where  $E$  denotes the edge strength. And the edge direction is given by:

$$\theta = \arctan \frac{Gy}{Gx} \quad (7)$$

where  $\theta$  denotes edge direction. The edge direction is then approximated to one of four possible values that make up the possible directions an edge could have

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Figure 8: Sobel Masks

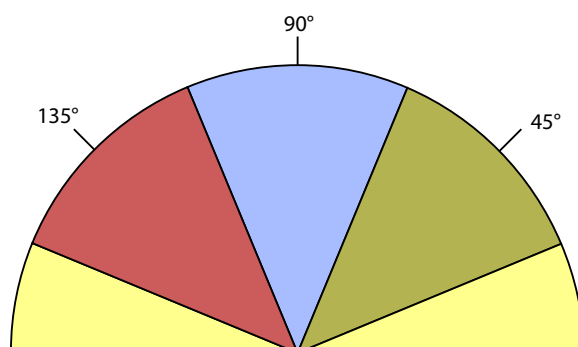


Figure 9: Any edge angle within 22.5 degrees of one of the possible angles is changed to that value.

as per Figure 9. Therefore, any edge direction falling within the yellow range (0 to 22.5 and 157.5 to 180 degrees) is set to 0 degrees. Any edge direction falling in the green range (22.5 to 67.5 degrees) is set to 45 degrees. Any edge direction falling in the blue range (67.5 to 112.5 degrees) is set to 90 degrees. And finally, any edge direction falling within the red range (112.5 to 157.5 degrees) is set to 135 degrees.

The next step is to actually trace along the edges based on the previously calculated gradient strengths and edge directions. Each pixel is cycled through using two nested for loops. If the current pixel has a gradient strength greater than the defined upper threshold, then a switch is executed. The switch is determined by the edge direction of the current pixel. It stores the row and column of the next possible pixel in that direction and then tests the edge direction and gradient strength of that pixel. If it has the same edge direction and a gradient strength greater than the lower threshold, that pixel is set to white and the next pixel along that edge is tested. In this manner any significantly sharp edge is detected and set to white while all other pixels are set to black.



Figure 10: Original image used as input.



Figure 11: Result of Canny edge detector

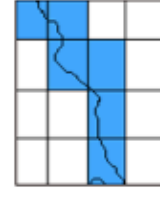


Figure 12: Result of Fractal Box Count.

The final step in the Canny edge detector algorithm is to find weak edges that are parallel to strong edges and eliminate them. This is accomplished by examining the pixels perpendicular to a particular edge pixel, and eliminating the non-maximum edges. The result is now a 1-bit image showing only the edges as seen in Figure 11, where the original input image can be viewed in Figure 10. The effect of the Canny operator is determined by three parameters — the width of the Gaussian kernel used in the smoothing phase, and the upper and lower thresholds

### Fractal Dimension - Box Counting con't

This 1-bit image is now used to calculate the fractal dimension according to the Box Counting approach. This is achieved by imagining an evenly-spaced grid overlaying the image, and counting how many boxes it requires to cover the newly found edges within the 1-bit image. The Box Counting dimension is calculated by seeing how this number changes as we make the grid finer.

As an example, in Figure 12 (a simplified version of Figure 11 for the purpose of demonstrating this algorithm), one can see that the box size is  $1/4$ , and the number of boxes through which an edge passes is 6.

This process is continued for an increasing number of squares (finer grid) and the fractal dimension is defined by the gradient of the logarithm of the number of squares occupied by the edge  $\ln N(s)$ , over the logarithm of the box size  $\ln N$ , which is found by the least square linear regression method. This then gives a quantitative analysis of the perimeter roughness of the input image.

### 3.2.2 Fractal Dimension - Differential Box Counting

The Differential Box Counting (DBC) approach was proposed by Sakar and Chaudhuri [23]. The advantage of this method is that it utilises the grey levels within the image and so the image does not need to be initially converted to a 1-bit image via the Canny edge detector algorithm. The Canny algorithm contains a number of adjustable parameters, which can affect the computation time and effectiveness of the algorithm. These adjustable parameters are the size of the gaussian filter and the threshold levels. Eliminating the Canny algorithm from this Box Counting method means in turn that it is not dependant on the optimal setting of these parameters.

As in the classical Box Counting method, an  $M \times M$  image is partitioned into a  $s \times s$  grid where the scale of each block is  $r = s$  whereby  $1 < s \leq M/2$  and  $s$  is an integer, and an estimate of  $s' = s * G/M$  whereby  $G$  is the total number of grey levels. For example,  $r = s = 3$  in Figure 13.

At the  $(i, j)$  grid, let the minimum and maximum grey levels of the image within this grid be  $k$  and  $l$  respectively. The number of boxes between the minimum and maximum grey levels at the  $(i, j)$  grid is counted by

$$n_r(i, j) = l - k + 1 \quad (8)$$

where the subscript  $r$  denotes the result using the scale  $r$ . For example in Figure 13,  $n_r(i, j) = 3 - 1 + 1 = 3$ . Considering contributions from all blocks,  $N(r)$  is counted for different values of  $r$  as

$$N(r) = \sum_{i,j} n_r(i, j) \quad (9)$$

The fractal dimension  $D_f$  can then be estimated once again using the least squares linear fit of  $\ln(N_r)$  versus  $\ln(1/r)$ .

### 3.2.3 Fractal Dimension - Improved Differential Box Counting

Fractal dimension estimates of digital image surfaces have discrepancies from their true fractal dimensions due to the limited resolution. Errors of the fractal dimension can be minimised if the appropriate approaches for box number counting are employed. Lia, Dub and Suna [24] suggest further improvements to the Differential Box Counting method by proposing three modifications so as to address this and to provide fractal dimension estimates with smaller fit errors.

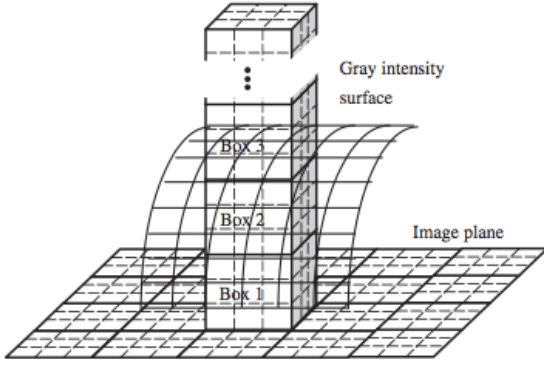


Figure 13: Determination of the number of boxes by the Fractal DBC method

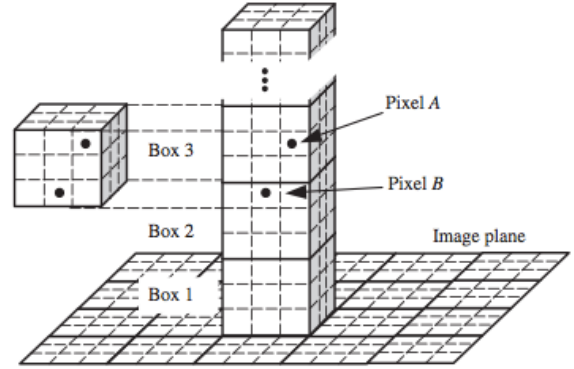


Figure 14: Box height calculation in the Fractal DBC method

The first modification was to that of the height selection. In the Differential Box Counting (DBC) method, a box height is selected by  $s' = s * G/M$  as stated in subsection 3.2.2. This means that it has a larger value when  $s$  is increased and therefore may lead to a greater computational error when counting the box numbers. Therefore the following modification is made.

Suppose the mean and the standard deviation of grey levels within the input image are denoted by  $\mu$  and  $\sigma$  respectively and that most pixels fall into the interval of grey level within  $[\mu - a\sigma, \mu + a\sigma]$ . The box height  $r'$  is selected by

$$r' = \frac{r}{1 + 2a\sigma} \quad (10)$$

where  $a$  is a positive integer and  $2a\sigma$  can indicate image roughness. In experiments conducted by Lia et al., it was found that increasing the value of  $a$  improved the results, however the FD estimates did not change obviously once it exceeded a certain value. Tests were conducted on values 1, 3, 5, 128 and 256, where the latter two values produced almost the same results.

The second modification concerned the box number calculation in order to guarantee the least number of boxes be obtained to cover every block for each specific scale. As seen in Figure 14, two pixels fall into two boxes with the size  $3 \times 3 \times 3$  but have the distance smaller than 3 in height direction. The DBC method therefore cannot acquire the least number of boxes that cover the block. In order to do so the following calculation for box number was proposed.

If the maximum and minimum grey levels of the  $(i, j)$ th block are  $l$  and  $k$ , respectively, the number of boxes that cover the block surface can be calculated as

$$n_r = \begin{cases} \lceil \frac{l-k}{r'} \rceil & l \neq k \\ 1 & l = k \end{cases} \quad (11)$$

Therefore in Figure 14 where  $r' = 3$  using this method  $n_r = 1$ , which is the exact number of boxes covering the block.

The third and final modification deals with the image intensity surface partition. An image is continuous with infinite resolution, whereas a digital image contains pixels with discretised results. The Hausdorff dimension takes this into account with overlapping balls as seen in Figure 6(a), however this feature is missing from the previous Fractal Box Counting methods. In the DBC method, there is a gap between any two spatially adjacent boxes. Therefore this indicates all boxes in the DBC method cannot completely cover the entire image when the image intensity surface is considered to be continuous.

In order to deal with this problem and to ensure that the image surface is completely covered, overlapping boxes are implemented. An image of size  $M \times M$  is partitioned into equivalent blocks of size  $s \times s$  pixels. Any two spatially adjacent blocks overlap at the boundary by one row (or column, i.e.,  $s$  pixels). Each  $s \times s$  block therefore has the scale  $r = s - 1$  which represents the maximum horizontal or vertical distance between any two pixels.

The rest of the algorithm follows the procedure for Differential Box Counting.

### 3.3 Local Binary Patterns

In order to determine the effectiveness of the Fractal Dimension method over previous and existing methods, the Local Binary Pattern method for determining texture features was also implemented so as to act as a standard on which to compare results. As previously mentioned this method has been found to be a powerful feature for texture classification and somewhat the standard from which to compare new methods.

The Local Binary Pattern texture descriptor is popular not only due to its discriminative power but also for its computational simplicity and ease of implementation. Like the Differential and Improved Box Counting methods, its is also calculated upon the greyscale version of the image.

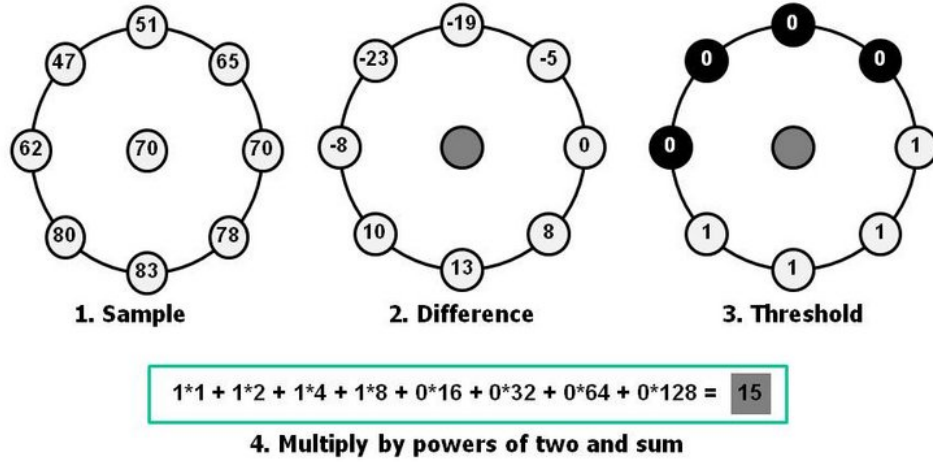


Figure 15: Example of Local Binary Pattern Calculation

In its simplest form the Local Binary Pattern feature vector, is created by comparing the pixel to each of its 8 immediate neighbours (left-top, top-middle, right-top, right-middle). The value at the centre pixel is thresholded by the neighbouring pixels  $P_i$  by

$$P'_i = \begin{cases} 0 & \text{if } P_i < P_0 \\ 1 & \text{otherwise} \end{cases} \quad (12)$$

The result of this thresholding is considered a binary number. By following the pixels along a circle, i.e. clockwise or counter-clockwise, the direction of which is irrelevant providing it remains consistent for each pixel within the image, and for all images, the discovered number is converted to a binary number so as to calculate the local binary pattern for that pixel.

$$LBP_{P,R} = \sum_{i=0}^7 P'_i 2^{i-1} \quad P_i = \begin{cases} 0 & \text{if } P_i < P_0 \\ 1 & \text{otherwise} \end{cases} \quad (13)$$

An example of this process can be seen in Figure 15.

This process is repeated for each pixel within the image to produce a histogram, of the frequency of each “number” occurring (i.e., each combination of which pixels

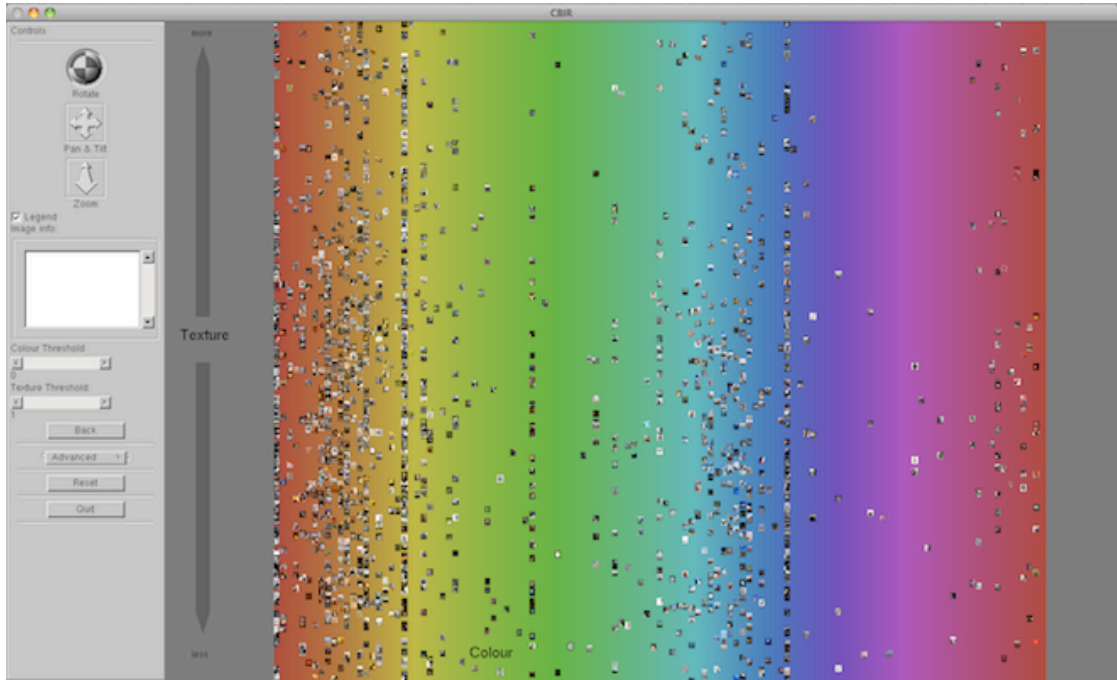


Figure 16: Graphic user interface of final implementation

are smaller and which are greater than the centre). This gives the feature vector for the window.

From this histogram, the mean, mode and median was calculated and stored in the preprocessing text file for input to the program upon startup. Through experimentation, the best results within the graphic user interface regarding image placement were achieved by using the mean.

## 4 Implementation

To facilitate a prompt start up of the program, input images were pre-processed so as to determine the relevant colour and texture values for each image according to the various algorithms as previously mentioned in Section 3. As it was the intention that the interface would work as an image catalogue, the images were subsequently resized to thumbnails of  $128 \times 128$  pixels so as to allow a greater number of images to be loaded into the program. Preprocessing was implemented in C++ using the libjpeg library to input the images and the results of the ensuing calculations were written to a text file.

The visualisation prototype was implemented in the OpenGL [32] environment in the C++ programming language. In conjunction with OpenGL, the GLUT [33] library was used for window management. The GLUI [33] library is a user interface library which provides controls such as buttons, checkboxes, radio buttons, and spinners to OpenGL applications. In particular version 2.35 of GLUI was utilised due to its addition of features such as lists and sliders. The GLUT and GLUI libraries were used due to their portability.

The text file produced during preprocessing containing the relevant information is read into the program upon startup, and each image is plotted into the 3D environment according to colour on the x-axis and texture on the y-axis. The third dimension (z-axis) was used for zooming in and out only. This produced an interface that allowed the user to move around and to interactively browse throughout the entire image collection.

Navigation is not only controlled by the GLUI controls via the control panel on the left (see Figure 16) but also via the mouse and scroll wheel to ensure an intuitive browsing environment. A colour legend was also implemented into the background so as to give a point of reference once zoomed in. This feature may be disabled at will via the control panel.

Humans tend to distinguish 11 colour categories [28], which can be divided into 6 primary colours (black, white, red, green, blue, yellow) and 5 secondary colours (grey, brown, purple, pink and orange). According to Helander et al. [29], the maximum number of colours that can be recognised on an absolute basis varies, but with training viewers can accurately recognise as many as 50 when the colours vary by not only hue but by lightness and saturation as well. In the dominant foreground colour method devised by Krishnan et al. [30], 25 predefined colours define the basis. These methods however are based upon exact user recognition of particular colours as opposed to the more relative method this paper describes where upon a user needs to search for a given image by means of browsing. This method allows for more leniency in the amount of colours.

In the implementation described in this paper, the dominant hue of each picture irrespective of the chosen colour retrieval method is defined by an integer from 0 to 360 (hue values). Given the information in the preceding paragraph and through initial tests with the prototype, it was obvious that this was too many hues giving too many options, with the difference between many hues not visibly apparent. For this reason, 120 colour groups (i.e. with hue values 0–2, 3–5, 6–8 etc) were made and these were used as the basis with which to plot the x-axis. The proposed method is not dependant on this, and given the number of images in the collection, this variable may be adjusted to suit.

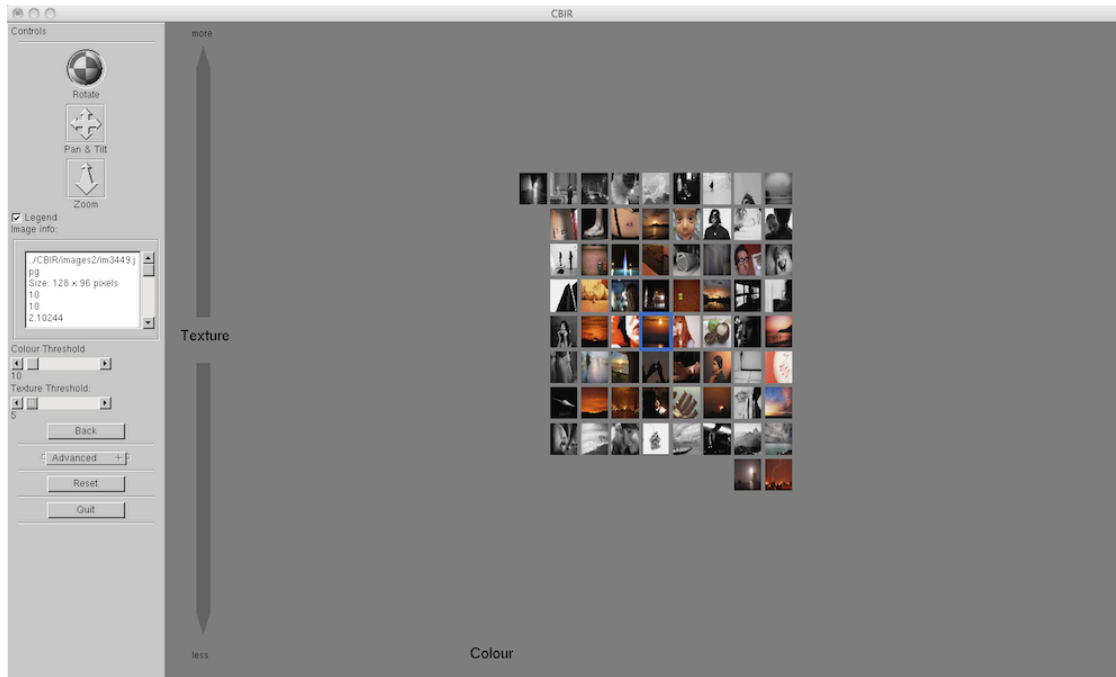


Figure 17: Interface shown in order to limit/increase search space

All methods of texture characterisation utilised in this prototype are expressed as floating-point numbers, and due to the nature of Fractal Dimension lie with a maximum value of 2 from each other. Using trial and error this scale was increased by a value 120 to give a pleasing spacing to the images within the interface. Again given the number of images in the collection, this variable may be adjusted to suit.

It is possible that two images contain the same dominant hue and (almost) the same texture as one or multiple images, therefore a second interface was implemented to offer an alternative approach and to possibly accelerate the search for a particular image. Once an image is located with a similar colour and texture, the second interface can be instigated by a double click. The selected image is then centred within the interface surrounded by a blue border (so as to distinguish it from other images) and the colour background legend is disabled (see Figure 17). In a circular pattern around the selected image appear all images with the same dominant hue and texture, ensuring no images overlap. In the user control panel on the left appear threshold sliders for both colour and texture. Adjusting the value on these sliders increases and decreases the search space in the x- and y-axis (colour and texture respectively) accordingly.

## 5 Evaluation and Results

All experiments and tests were conducted using images from the MIRFLICKR-25000 collection [34] available through the Leiden Institute of Advanced Computer Science. This collection contains 25000 images downloaded from the social photography site Flickr through its public API. This library gave a good indication of a general consumer user image collection with typically lots of images containing skin tone (such as faces), and with very few images with purple as a dominant colour which seemed the norm amongst other typical user image collections.

Due to the limitations of the 256Mb VRAM on the available testing computer, a fluid responsive program was achieved when the the total number of images entered as input to the program was limited to 2500. The MIRFLICKR-25000 collection was therefore divided into 10 sub-collections to be used for implementational testing, experiments and user tests.

### 5.1 Observations

Initial comparisons were made between the colour algorithms mentioned in Section 3.1. All three implemented colour algorithms gave good results and pleasing performance. The Diamond Segmentation approach appeared to achieve better results than the straightforward histogram approach bringing attention to the foreground objects as expected. The further refinement upon the Diamond Segmentation method amended a mere 1% of the values as opposed to the straightforward approach, and of these only 0.1% were moved more than a difference of 10 hue values (3 columns in the graphic interface). In these cases, this sometimes resulted in a better classification, and yet in other cases the reverse is true. Further investigation is needed to determine the overall effectiveness of this method.

Furthermore comparisons were made between the texture algorithms. It was quite apparent that the all Fractal Dimensions out-performed the Local Binary Pattern texture descriptor. Although the LBP approach did show some graduation from simple to complex textures, it was not always obvious and often flawed. For example, most sunset photos were all clustered together under the Fractal Dimension approach, whereas they were somewhat dispersed in the LBP approach (see Figure 18) making searching for such images somewhat cumbersome. The Fractal Dimension approaches on the other hand generally gave a distinct graduation from the most simple images at the bottom to the most complex at the top albeit not perfect.



Box Counting approach, and the other the Local Binary Pattern. During the user evaluations, observations were made as to how the user interacted with the program and their preferred search method. The sub-collection of the MIRFLICKR-25000 collection [34] containing images 2501–5000 was used for this test, and 20 images were randomly chosen previous to testing. Albeit random, these images were ensured to represent certain image categories, such as a cityscape, face, sunset, etc. Each person was asked to find the same 20 images, and was timed in doing so. These images ranged from the simple and more obvious to the more challenging. The users could use either interface or a combination of both. A maximum limit of 6 minutes to find the image was set, as it was noted that most people started to forfeit between the 5 and 6 minute mark. On average the images took between 2 and 3 minutes to find, with the whole test taking just under 50 minutes.

Everyone to some degree displayed a learning curve (some more than others) and having found the first image most settled into the task. Various search methods were observed - some remained predominantly in the first interface preferring to zoom in and browse, while on the other hand, some preferred to go to a general region amongst an appropriate colour where they would expect to find the desired image, continue onto the second interface and progress to adjust the colour and texture search spaces accordingly. The graphic designers and the texture engineer showed most inherit understanding of texture and would become frustrated when the image was not where expected. However due to their common practise of searching for images this was soon overcome. Those testing the Improved Box Counting approach were enthusiastic about the interface.

Certain search methods however stood out for their individuality and unexpectedness. One user systematically searched from bottom to top through all colours regardless of the colour or texture of the required image. Although this seems like a time-wasting exercise, the user's times were the most consistent across the test and managed to find images that many others could not due to their unexpected or obscure dominant colour. One other user was exceedingly slow at the start but during the course of the test sped up, due to the fact that the user was attempting to memorise the positions of the images, in the hope they would be requested later in the test.

As stated, one child of 9 years of age was included in the test group to observe how a child would handle the interface. Despite spending well over 5 minutes on the first image, all subsequent images were exceedingly fast - one as fast as 8 seconds. The child's method was to scarcely zoom so that many minute images filled the interface, sit very close to the screen and scan over a great deal of the screen until found. This method to a lesser degree was also observed amongst some adults who were also quick and perceptive.



Figure 19: Image 8



Figure 20: Image 20

Most users however stopped considering texture after a few images and seemed to concentrate purely on colour. Those most successful in the test were those who were not only perceptive and strongly visually aware, but also were open to other ideas finding the images and did not impose strict mental guidelines on where it should be found.

The colour feature predominated the test, and often the success or failure of finding an object was based on perceiving the correct colour. As mentioned by Forsyth and Ponse [31]: “It is surprisingly difficult to predict what colors a human will see in a complex scene; this is one of the many difficulties that make it hard to produce really good color reproduction systems.” This became evident during testing with some people making seemingly obscure choices for starting points for searches. For most images, the correct colour range was immediately chosen, however there were cases when a user would choose a bright yellow or red as a starting point when this colour, although obvious to the human eye, was a small part of the image.

One particular image (as seen in Figure 19), proved decidedly difficult with only 4 users finding it within the time limit. These users deemed the overall colour to be indigo and found it quickly (in about 1 minute) due to its lack of neighbouring images within the image arrangement. The rest however recognised a lot of skin and remained searching within the skin tones. If they exhausted that search they would continue onto the blue due to the bright blue wig worn by someone in the background.

When dealing with colour some people actually looked at the colour in the photograph, whereas some people recognised the object and sought the image according to the real-world colour. Image 20 (Figure 20) is such an example. Some users upon consideration deemed the photograph to be amongst the flesh-tones due to all the skin, whereas the rest recognised the lighting on the subject along with the colour in the jacket and explored the magenta hues where the image lay. Very few people further observed the thick black borders on the top and bottom

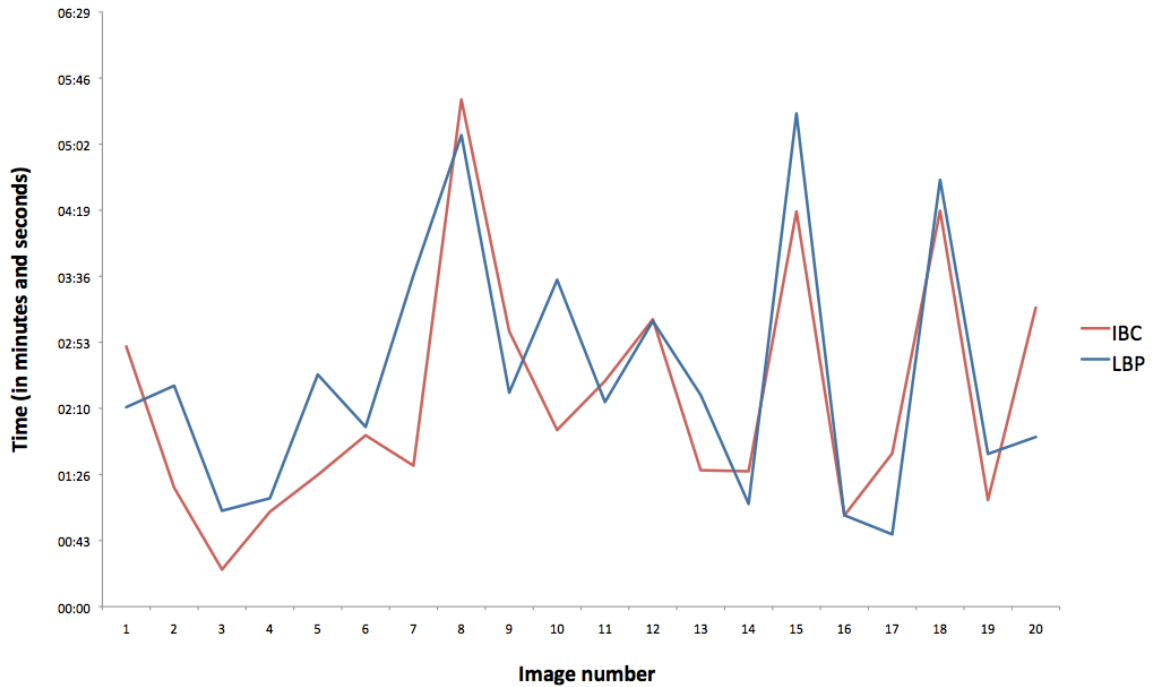


Figure 21: Average search time per image

of the image - those who did managed to find the image quickly. The success of the LBP method over the IBC approach for this image was purely down to the user guessing (or not) the correct colour classification, as under both algorithms they happen to be in relatively similar areas.

Some users also appeared more sensitive to colour and would continue to browse up and down with the same colour range in order to find the image - even if they had looked a few times. The ability to widen the search did not manifest itself as an alternative plan.

Overall though, it appeared that a lot of it came down to perception and some user's increased visual awareness. Colour played by far the most important part in the users search method, and often was the only characteristic used within the search. On the whole though, the IBC outperformed the LBP by a margin of 2.5 minutes for the overall test and on average users found the image 15 seconds faster using the IBC (see Figure 21). On the images that were clearly misplaced in the Local Binary Pattern algorithm (images 2, 5 and 7), the average search time was 45 seconds longer than for that of the Improved Box Counting method. The relatively poor performance of the IBC method in the 10th image appears to be due in part to the overcrowding of images in that section of the interface as

opposed to its position in the LBP method. Most users appeared to tire of the task after 10 images and from this point there is very little difference between the two algorithms.

Due to the obvious effect that colour considerably contributed to the search method of the user, a second evaluation test was arranged to test purely the effect of the Improved Box Counting texture descriptor as opposed to that of the Local Binary Pattern. Using a new and smaller sub-collection of images (this time 1-1000), these images were converted to greyscale after the preprocessing step and the colour legend in the background was disabled. Despite having calculated the colour information during preprocessing, this was used predominately as a means of dispersing the images over the x-axis. Random values would have been another method. The users were not given any information as to the arrangement of the images upon the x-axis.

The second interface was also disabled for this test so as to tighten control on the test variables. The selected images for the user test therefore were not randomly chosen but carefully selected to be certain that no image was concealed or inhibited from view. Due to the fatigue problem in the first test, only 10 images were requested this time, and the user group was much smaller (16 users - each group comprising of 8 people), however this time concentrated on those who showed an affinity towards working with texture or would do so for their profession.

Overall the users struggled with the lack of colour information and the test became much more difficult as a result, hence the reduction in the total number of images used in this test. Without the colour background legend, there were no immediate reference points making it more difficult for the user. Images at the extremes of texture (displaying very little, or a tremendous amount) were relatively simple for a user to understand and those images were quickly sought, whereas the users had difficulties understanding the texture of images that lay somewhere between the limits.

Despite colour being removed from this test and users being instructed to find the images based purely on texture, users nonetheless used contrast or attempted to find a scale within the greyscales as an extra help tool to find the required image. One user imagined some sort of ordering based upon three tones of greyscale, and another attempted to find some sort of ordering within the contrast levels. Finding the image purely based on texture proved to be difficult for all despite their professional background, and any excuse or added assistance by any other possible means was attempted. Not one user realised that the ordering on the x-axis was based upon the true colours of the image, and used that information accordingly to find the required image. Many users would also steer their search

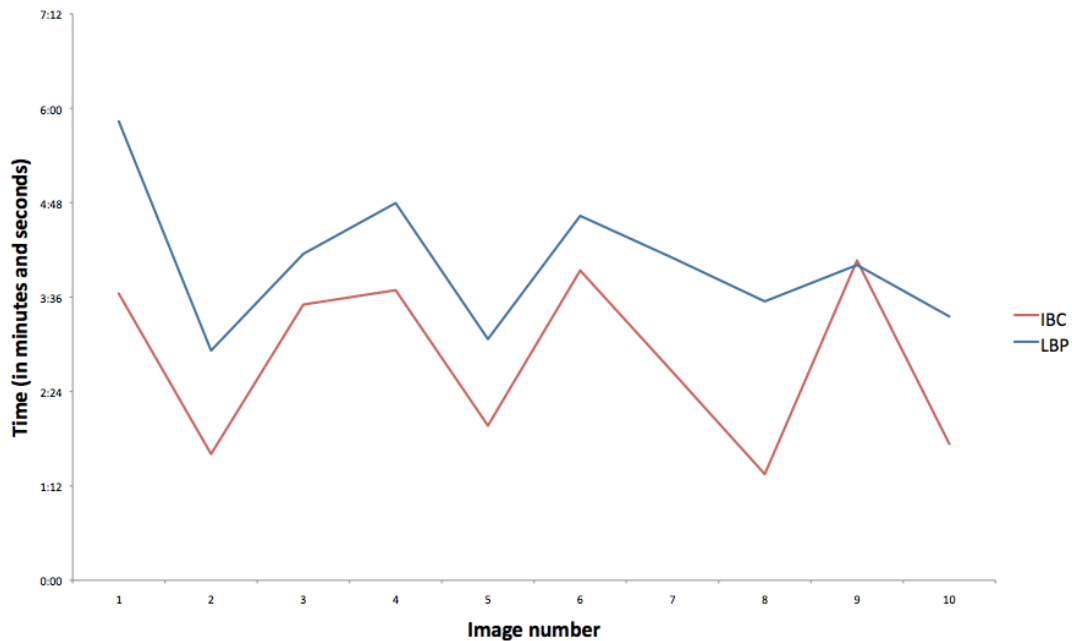


Figure 22: Average search time per image when performing a texture only search

towards the busiest part of interface and continue to return to that section as they felt they had most chance of finding the image within the section of the interface.

Despite the difficulties imposed upon restricting the user information to texture, and despite the attempt to use added information like that of contrast, from the results shown in Figure 22 it is apparent that the Fractal method of Improved Box Counting out-performs the Local Binary Pattern across the board, by an average of 1 minute 12 seconds per image sought, and users utilising this method completed the full task 30% faster. Users given the LBP method were often wasting time searching where the image should logically appear only to have to resort to a full browse through the images in order to find it.

Interestingly image 4 recorded almost exact times regardless of the method, and this appears to be due to the fact that there was comparatively little difference in the position of this image regardless of the algorithm used, and that it was positioned on an outlying edge in a sparsely populated section of the interface.

## 6 Conclusion

The aim of this project was to produce a fluid graphical user interface for an image database based purely on content of the image, and this system received positive feedback from those evaluating it and especially from those such as graphic designers who search for images on a regular basis. Using the OpenGL system to map the images produced an intuitive interface, and by combining existing techniques as a mean to cluster the images gave a pleasing result.

The Diamond Segmentation method as mentioned in Section 3.1.1 refined the dominant colour histogram approach, and through observation this method gave a very satisfying solution to the prospect of sorting images based upon colour. An attempt to further refine this method by eliminating the anomalies produced by a solid colour that dominates the slight variations in hue in what appears to the human eye as the dominant colour, produced mixed results working better with some images than others in a small percentage of the image database. Further investigation is needed to determine the overall effectiveness of this method.

Concerning the texture descriptors, the Fractal Dimension approach worked exceedingly well, and in particular that of the Improved Box Counting which was an advancement upon the Differential Box Counting method, which had already improved upon the original Fractal Box Counting approach. Pure observation already showed superiority over the Local Binary Pattern approach and this was further proven in user evaluation, where users were generally considerably faster in searching when the images were arranged using this approach, proving the effectiveness of the system.

From childhood we are taught to understand, recognise and distinguish between colours, and although we are also taught to recognise and feel the differences in texture, its impact on our lives is much less important, obvious and inherent than that of colour. This may account for the preference of colour over texture as a contributing factor in a user searching for an image. The user instinctively understands colour, and even though there appeared varying sensitivity to colour amongst the user groups, very few users managed to maintain the consideration of texture as a feature descriptor for any considerable length of time. Despite an interface based on colour and texture that works well on an analytical level, with many users suggesting they would prefer to use colour and contrast as opposing axes the question, remains as to the viability of texture as a feature descriptor.

## References

- [1] Gantz, J.F.: *The expanding digital universe - A Forecast of Worldwide Information Growth Through 2010*, An IDC white paper, (2007)
- [2] Kato T.: *Database architecture for content-based image retrieval*, Image Storage and Retrieval Systems, SPIE, Vol. 1662 No. 1 pp. 112–123 (1992)
- [3] Gabor, D.: *Theory of Communication*, Journal of the Institution of Electrical Engineers, Vol. 93 No. 26 pp. 429–457, (1946)
- [4] Azencott, R., Wang, J.P. and Younes, L.: *Texture classification using windowed fourier filters*, IEEE Trans Pattern Analysis and Machine Intelligence, Vol. 19 No. 2, pp. 148–153 (1997)
- [5] Ojala T., Pietikinen M. and Harwood D.: *A Comparative Study of Texture Measures with Classification Based on Feature Distributions*. Pattern Recognition, Vol. 29 No. 1 pp. 51–59 (1996)
- [6] Mäenpää T.: *The local binary pattern approach to texture analysis - extensions and applications*, PhD Thesis, University of Oulu (2003)
- [7] Ahonen, T., Hadid, A. and Pietikainen, M.: *Face Description with Local Binary Patterns: Application to Face Recognition*, International Conference on Computational Intelligence and Multimedia Applications, IEEE Computer Society, Vol. 28 No. 12 pp. 2037–2041 (2006)
- [8] Zhang, G., Huang, X., Li, S., Wang, Y. and Wu, X.: *Boosting Local Binary Pattern (LBP)-Based Face Recognition*, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Vol. 3338 (2005)
- [9] Bangyou D. and Nong S.: *Local binary pattern based face recognition by estimation of facial distinctive information distribution*, Optical Engineering, SPIE, Vol. 48 No. 11 pp. 117203 (2009)
- [10] Flickner M. et al: *Query by Image and Video Content: the QBIC system*, IEEE Computer, Vol. 28, No. 9, pp. 1241–1253 (1990)
- [11] Pentland A., Picard R. W. and Sclaroff S.: *Photobook: Tools for Content-based manipulation of Image Databases*, International Journal of Computer Vision, Springer Netherlands, Vol. 18 No. 3, pp. 233–254 (1996)
- [12] Stockman M.G. and Gevers T.: *Density estimation for color images*, Journal of Electronic Imaging, SPIE, Vol. 10 No. 1 pp. 221–227 (2001)

- [13] Lee X. and Yin Q.: *Combining Color and Shape Features for Image Retrieval*, UAHCI '09: Proceedings of the 5th International Conference on Universal Access in Human-Computer Interaction. Part III, Springer-Verlag, pp. 569–576 (2009)
- [14] Hausdorff F.: *Dimension und äußeres Maß*, Mathematische Annalen Vol 79. pp. 157–179 (1919)
- [15] Barnsley M.F.: *Fractal everywhere*, 2nd edition, Academic Press Professional, New York, (1993)
- [16] Feng J., Lin W.C and Chen C.T.: *Fractional Box Counting Approach to Fractal Dimension Estimation*, Proceedings of the 13th International Conference on Pattern Recognition, Vol. 2 pp. 854 (1996)
- [17] Gangepain L. and Roques-Cames C.: *Fractal approach to two dimensional and three dimensional surface roughness*, Wear, Vol. 109 No. 1–4 pp. 119–126 (1986)
- [18] Buczkowski S., Kyriacos S., Nekka F. and Cartilier L.: *The modified Box Counting method: analysis of some characteristics parameters*, Pattern Recognition, Vol. 31 No. 4 pp. 411–418 (1998)
- [19] Chen W.S., Yuan S.Y. and Heieh C.M.: *Two algorithms to estimate fractal dimension of gray-level images*, Optical Engineering, SPIE, Vol. 42 No. 8 pp. 2452–2464 (2003)
- [20] Du G. and Yeo T.S.: *Novel multifractal estimation method and its application to remote image segmentation*, IEEE Transactions on Geoscience and Remote Sensing, Vol. 40 No. 4 pp. 980–982 (2002)
- [21] Novianto S., Suzuki Y. and Maeda J.: *Near optimum estimation of local fractal dimension for image segmentation*, Pattern Recognition Letters, Vol. 24 No. 1–3 pp. 365–374 (2003)
- [22] Xu S. and Weng Y.: *A new approach to estimate fractal dimensions of corrosion images*, Pattern Recognition Letters, Elsevier Science Inc., Vol. 27 No. 16 pp. 1942–1947 (2006)
- [23] Sarkar N. and Chaudhuri B.B.: *An efficient differential box counting approach to compute fractal dimension of image*, IEEE Transaction on System Man and Cybernet, Vol. 24, No. 1, pp.115–120 (1994)
- [24] Lia J., Du Q. and Sun C.: *An improved box-counting method for image fractal dimension estimation*, Proceedings of the 2008 International Symposium on Information Science and Engineering, Vol. 01 pp. 303–306 (2008)

- [25] Liu S: *An improved differential box-counting approach to compute fractal dimension of gray-level image*, Pattern Recognition, Vol. 42 No. 11 pp. 2460–2469 (2009)
- [26] Canny J.F.: *A Computational Approach To Edge Detection*, IEEE Trans. Pattern Analysis and Machine Intelligence, IEEE Computer Society, Vol. 8 No. 6 pp 679–698, (1986)
- [27] Sobel I., and Feldman G.: *A 3x3 isotropic gradient operator for image processing*, Presented at a talk at the Stanford Artificial Project, (1968). (unpublished)
- [28] Goldstone R.L.: *Effects of categorization on color perception*, Psychological Science, Vol. 6 No. 5 pp. 298–304 (1995)
- [29] Helander M., Landauer T.K. and Prabhu P.V.: *Handbook of human-computer interaction*, Elsevier Science Inc., pp. 605 (1997)
- [30] Krishnan N., Sheerin Banu M. and Callins Christiyana C.: *Content Based Image Retrieval using Dominant Color Identification Based on Foreground Objects*, Proceedings of the International Conference on Computational Intelligence and Multimedia Applications, IEEE Computer Society, pp. 190–194 (2007)
- [31] Forsyth D.A. and Ponce J.: *Computer Vision: A modern approach*, Prentice Hall (2002)
- [32] Neider, J. and Davis, T.: *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Release 1*, Addison-Wesley Longman Publishing Co. Inc., (1993)
- [33] Hill, Jr. F.S. and Kelley, S.M.: *Computer Graphics Using OpenGL (3rd Edition)*, Pearson Education, (2007)
- [34] Huiskes M.J., Lew M.S.: *The MIR Flickr Retrieval Evaluation*, ACM International Conference on Multimedia Information Retrieval (MIR'08), Vancouver, Canada, pp. 39–43 (2008)