

**CONTENT BASED IMAGE RETRIEVAL (CBIR)
TO FIND SIMILAR IMAGES IN A DATASET**

**MEGAN WOON TONG NING
27733327
&
JOHN GOH RENGWU
27150437**

**FIT3081 Image Processing
Implementation project
Semester 1, 2018
School of IT
Monash University Malaysia**

May, 2018

ABSTRACT

A picture speaks a thousand words; these are the words of the Chinese philosopher Confucius that asserts how images have expressiveness that are beyond the capabilities of words. In recent years, due to the explosion of digital media and the also the use of Internet, there has been an abundance of audio-visual data such as images and videos. Content-based image retrieval (CBIR) is an image indexing technique that is based on the contents of the image itself, rather than relying on keywords and labels. In this study, we implemented a CBIR system that employs color and texture feature extraction methods to accurately retrieve similar images. Color histogram and Gabor filter descriptors assisted us through the two-phase filtering process and we have managed to retrieve relevant images from various datasets via a simple MATLAB implementation with mediocre accuracy.

TABLE OF CONTENTS

INTRODUCTION	5
Introduction	5
1.1.1 Background on CBIR	6
1.1.2 Colour Feature Extraction	6
1.1.3 Texture Feature Extraction	6
1.1.3 Shape Feature Extraction	7
Aims and Objectives	7
1.2.1 Data pre-processing	8
1.2.2 Colour Feature Extraction	8
1.2.3 Texture Feature Extraction	8
1.2.4 Comparing Image Similarity and Retrieving Similar Images	9
LITERATURE REVIEW	10
Usages of Color Histogram in Literature	10
Usages of Gabor Filters in Literature	11
Other Notable Methods in Literature	11
METHODOLOGY	13
3.1 User Interface	14
3.2 Data Preprocessing and Initialization of Matrices	14
3.2 HSV-space Colour Histogram	15
3.3 Gabor Filter as Texture Feature Extraction	17
3.4 Calculating Euclidean Distance between Two Images	19
3.5 Computing Similarity Values	20
3.6 Filtering process	21
RESULTS AND DISCUSSIONS	22
4.1 Test cases and results	22
4.1.1 Test Case 1: Controlled Dataset of Images of Emu	22

	4
4.1.2 Test Case 2	23
4.1.3 Test Case 3	23
4.1.4 Test Case 4: Dataset of 296 Wild life images	24
4.1.5 Test Case 5	25
4.1.6 Test Case 6	25
4.1.7 Test Case 7	26
CONCLUSION AND RECOMMENDATIONS	29
Recommendations	29
REFERENCES	30

CHAPTER 1

INTRODUCTION

1.1 Introduction

With the current development of multimedia technology and digital devices, digital image databases are rapidly growing. These databases provide assistance towards multiple fields and sectors, namely in medical research. The immense amount of images stored in these databases has created increasing challenges for computer systems to search and retrieve relevant images swiftly and accurately. Traditionally, text-based image retrieval (TBIR) has been used to address the image retrieval problem. However, it is labour-intensive as human intervention is required to manually annotate each image with keywords. Making sure that every details of the image have been correctly indexed or tagged with the relevant keywords is a chore. Besides that, using text to identify images are not as accurate nor is it reliable when it comes to subject retrieval of images. There is also the instance that an image may have been tagged in a different language and these keywords would have to be translated into the language of the text query for matching. Lack of context in the images or keywords the user uses or inputs is also an issue in TBIR. For example, if a user were to query for the keyword ‘apple’ expecting pictures relating to the Apple technology company, but received pictures of actual apples unless they type in more details. To address TBIR’s main issues in the image retrieval problem, content-based image retrieval (CBIR) is introduced.

Content-based image retrieval is a technology that has been around for decades and have been a staple to many industries. One of the more prominent industry that uses CBIR is the medical imaging industry. Due the widespread usage of CBIR in many industries, many researchers have included this system in their research hoping that they can further improve on CBIR and the results it outputs. However, despite the advantages that CBIR have proved to have, a lot of professionals still tend to use TBIR, based on the sole reasoning that CBIR is not as accurate as they like it to be.

Since the main reason of CBIR’s existence is due to solve the shortcomings of text-based image retrieval systems, there are few things to be noted in terms of the implementation of CBIR. CBIR employs computer vision techniques to analyze the contents of each image to search or retrieve similar images from a dataset.

This project focuses on implementing a CBIR system based on the few feature extractions that the team have learned and tested in with results in order to further understand the implementation of CBIR. This report will be split into few sections. The first section will explore further on the background and the types of feature extraction methods that CBIR systems have used and still exploring on.

1.1.1 Background on CBIR

Content-based image retrieval is heavily based on the extraction and subsequent comparison of visual features of the images. These images features can be classified into three categories :

- Low-level features (human-vision related)
- Middle-level features (object related)
- High-level features (semantic related)

However since content-based image retrieval needs to be computationally efficient and simple, the majority of CBIR implementations will be focused on low-level features. Low-level features are basic, raw components of the image which usually makes up of color, texture, edges, gradient orientation and so forth. These low-level features can be further processed to detect higher level features such as objects and semantic descriptions can even be generated. Higher level features describes the visual content of an image by correlating with low-level features (Anandh, Mala & Suganya, 2016).

Low-level features are used typically in CBIR due to the fact that those are more related to the human vision as the machines needs to identify the image based on how a human perceives it. This helps with the accuracy of the output as compared with using other levels of feature. Besides that, the simplicity of the low-level features are the vital component to the reasoning of it being widely popular.

There are multiple types of low-level features that focuses on human vision, it being colour, texture and shape information of each content of the image. Each feature extractions bring their own version of advantages and disadvantages. In many cases, there are not one specific feature extraction that output the best results. In fact, according to the requirements of some CBIR, the feature extraction is chosen.

1.1.2 Colour Feature Extraction

Colour is the most common visual feature used for image retrieval. The reason for majority of its uses is due to the fact that it is robust to background complication or image size/orientation (Kaur & Sohi, n.d.). There are a number of techniques by which colour feature can be extracted out accordingly, with Color Histograms being one the most popular methods.

1.1.3 Texture Feature Extraction

Texture is an important visual feature which describes the surface property of an object and their relationship to the surrounding environment. Texture of an image is characterized by differences in brightness with high frequencies in the image spectrum. There are two categories textures are placed into, the statistical methods and the structural methods. Structural methods are commonly applied on textures that are regular where else statistical

methods, as the name refers to check on the statistical distribution of the image intensity (Srivastava, Goel & Agarwal, 2017). Common examples of texture feature descriptors include grey-level co-occurrence matrices and also local binary patterns..

1.1.3 Shape Feature Extraction

Shapes are more complicated than color and texture features. They need to go through a region or contour identification process and be segmented into meaningful objects and background before applying shape descriptors. However, it is computationally harder to perform shape comparisons, therefore, the contribution of shape descriptors in general are scarce.

Utilizing the above image retrieval techniques, we are then able to apply CBIR in many helpful ways. Firstly, law enforcement agencies who typically maintain large archives of visual evidence, including mugshots or any form of photographic evidence, can make use of CBIR to compare visual similarities between a query image with the entire database quickly. As mentioned earlier, CBIR is also heavily applicable in the medical field. Medical diagnostic techniques like radiology, histopathology and tomography has resulted in an ever increasing abundance of medical images stored by hospitals. With the integration of CBIR techniques, medical imaging systems would be able to quickly display images relating to a patient or condition. Lastly, military applications of imaging technology which includes recognition and identification of targets from radar screens and satellite images can also make use of CBIR (Sharma, 2015).

1.2 Aims and Objectives

In the research process, we have discovered and came across many reasons why content-based image retrieval is considered a superior successor to text-based image retrieval systems. The aims that we have set at the beginning of this project are considered and agreed upon. The aims are as below:

1. To implement an image retrieval system that addresses the shortcomings of text-based image retrieval systems.
2. To further understand the inner workings of how a CBIR is constructed.
3. To determine whether usage of different feature extractors affect the results output.
4. To understand the advantages and disadvantages of different feature extraction methods.

After further discussion amongst team members and further research, the team has decided upon two feature extraction methods to apply on the final implementation of the system. There are a couple of risks involved however, when it comes to implementing this project. One of it being the time limitations and the other major one is the available attention being given to this system as there are other units that the team are taking with involved several assignments as well. To ease up the progress and make sure the implementation runs

smoothly, there are a couple of objectives that the team have set up for themselves. The two feature extraction methods that the team chose involve color features and texture features.

1.2.1 Data pre-processing

Content-based image retrieval involves the processing of a large library of an available dataset. Due to the fact that a large repository of images can come in all varieties of dimensional sizes and encoding formats, preprocessing methods must be first imposed to each and every one of these images. This step is necessary to enforce a certain format standard for each of the images before feature extraction methods are applied. The pre-processing methods applied to the image are:

- Resizing the images to a predetermined resolution.
- Converting grayscale images into the same color space format (such as RGB) before further processing.

1.2.2 Colour Feature Extraction

The colour feature is the most general feature extracted from an image as it is robust, and invariant to rotations and scalings (Anandh, K.Mala, S.Suganya). The objectives for this step will be simple and straightforward.

- Implement a colour feature extractor that is based on the occurrences of each color pixel of the images.
- Have a final feature vector output that can be used as an accurate comparator for image similarity.

To satisfy the above objectives, we have decided as a team that a HSV-space color histogram would be a suitable choice. It is both time and space efficient not only due to its simple pixel occurrence-counting implementation, but also because the hue, saturation, and intensity values of each pixel are quantized before they are placed into their respective bins.

1.2.3 Texture Feature Extraction

The textural features of an image is considered a vital component in order for the system to have accurate outputs. The objectives for this extraction is as below:

- Run the images into the texture feature extraction that will identify the textures
- Ensure that the texture feature extraction take consideration of scales and orientation.
- Have an output that outlines the textures of the images.

Similarly, the subsequent output for this extraction method will be used to gauge the similarities of two images.

1.2.4 Comparing Image Similarity and Retrieving Similar Images

An effective image retrieval system would not be considered as complete without a reliable method for comparing the similarity indexes between the query image and all images in the dataset. The subsequent filtering of images that are deemed dissimilar is also vital as it is our goal to narrow down the final output results, showing only the most similar images. In our project, the objectives that we have set for this aspect include:

- Implement a precise method of calculating and measuring similarity between the query image and all images in a dataset, with the help of the extracted visual features.
- Include a filtering process that removes images that are found to be too dissimilar to a certain threshold, when compared to the query image.

By reaching these goals, a reliable image retrieval system can be actualized just by having an image as the only query input. Similar images can be retrieved, and dissimilar images will be filtered out and not be shown in the results.

CHAPTER 2

LITERATURE REVIEW

There are many pre-existing studies carried out on content based image retrieval, many of which has proven itself to be a good addition to the field of research. A lot of these article reviewed are based on colour and texture feature extractors. However, there are also other notable feature extraction methods that are also worth considering and acknowledged as it helps with the improvement of the CBIR.

2.1 Usages of Color Histogram in Literature

Color histogram is an important technique for color image database indexing and retrieval because of its efficiency and effectiveness (Swain & Ballard 1991). According to Sharma, Rawat & Singh (2011), color histogram works by counting all image pixels and keeping track of its color distribution by associating each quantized range of color value to a specific bin. Similarity of images then can be compared by comparing the obtained histogram outputs by intersecting them. The appealing aspect of this image descriptor if its simplicity and ease of computation.

Many other publications focus on color indexing techniques based on color distributions, but admittedly have limited discriminating ability because they are unable to capture local color information (Mangijao & Hemachandran 2012). Color correlogram and color coherence vector can combine the spatial correlation of color regions as well as the global distribution of local spatial correlation of colors. These methods yields a better performance than traditional color histograms in content-based image retrieval. However, expensive computations are necessary for such methods.

In a paper by Mangijao & Hemachandran (2012), in order to improve the discriminative power of color histogram indexing techniques, they have divided the image horizontally into three equal non-overlapping regions. From each of these three regions, they extracted the first three color moment distributions and stored 27 floating point numbers in the index of the image. The color distribution of the image can then be interpreted as a probability value to characterize its color moments (Stricker & Orengo 1995). The distance between the color feature vectors is then computed with Canberra distance measure. Results of the query are then displayed in decreasing similarity order. This proposed method clearly demonstrates the encoding of spatial information in the color index from the three regions of the image significantly increases the discriminating power.

Yamamoto, as cited by Singh, Sohoni & Kumar (2014), proposed a CBIR method which takes into account the spatial data of colors by utilizing multiple histograms. This system is able to capture spatial informations of colors by recursively segmenting an image

into two rectangular sub-images, dividing it into two dominant regions. This goes on until each region has a homogenous color distribution, or the region size becomes reaches a threshold value. This results in a binary tree representing the derived color distribution of the image, which facilitates the evaluation of similarity among images.

2.2 Usages of Gabor Filters in Literature

Gabor wavelets has proven itself to be a very functional texture analysis method and is widely adopted in the literature world. For instance, in a study by Zhang, Wong & Indrawan (n.d.), Gabor filter is applied in a CBIR method, with the image indexing and retrieval based on textured and natural images. Gabor texture features are found by calculating the mean and variation of the Gabor filtered image. Rotation normalization is realised by a circular shift of the feature elements so that all images possess the same dominant direction. Applying their proposed method on a image database composed of 1000 different kinds of texture, all 15 similar textures in the database are retrieved in the first 18 images, and only one image is dissimilar in the first 25 retrieved images.

In another study by Tunga, Jayadevappa & Gururaj (2017), Gabor filter is combined with color and shape feature extractors by unsupervised learning. This is meant to reduce the semantic gap between user's perception and the query. Retrieval is done in three phases: considering a database of n images, retrieved images from the first phase are given as input for the second phase. In this phase, texture feature extraction has been done using Gabor Filter and the best matched images are retrieved, and is in turn used as input for the third phase. In the third phase, shape features are extracted using the genetic algorithm. This approach works similarly to our own proposed methodology, except that we are using a two-phase filtering process instead of three. The experiments have shown good results with a precision value of almost 95%, and is attributed to Gabor feature giving good response to the texture of the image and makes it clear for genetic algorithm to retrieve similar images.

Chawki et al. (2018) proposed a new method of using Gabor filters for content-based image retrieval. Seeing the disadvantages of Gabor filters regarding the choice of values and number of frequencies and orientations, Chawki et al. proposed a method which consists extracting from the image itself with good precision the frequencies and the orientations using the 2-D ESPRIT method, which will be injected in the Gabor filters. The obtained results shows that this method is very efficient and fast compared with classical Gabor filters, with an average precision of 90%.

2.3 Other Notable Methods in Literature

Other than color and textural features, there exists other methods of feature extraction that can be employed in content-based image retrieval. Shape features are one of those many methods. Jain, Muthuganapathy & Ramani (n.d.) presented a CBIR algorithm with respect to a database consisting of engineering/computer-aided design models. The algorithm uses the shape information in an image along with its 3D information (contour-based). Retrieval of

similar images is then done with a similarity measure that combines shape and depth information. However, results show that shapes as an extracted feature in CBIR has a very high computational complexity, although it can give a very good result for a database of engineering model images.

Nandagopalan, Adiga & Deepak, as cited by Singh, Sohoni & Kumar (2014), proposed a method for image retrieval using relevance feedback (RF) method. Relevance feedback model is efficient and realistic to shorten the semantic gap between lower level visual features and higher level perceptions using three feature extraction methods specifically color, texture, and edge histogram descriptor. A support vector machine-based RF algorithm is also proposed to further advance the retrieval performance.

Although these methods are very capable and are considered good implementations of CBIR, the required work and difficulty does not fit the available time frame that we have in our disposal. Hence, we instead opted for a more simple approach towards color histograms and Gabor filters in our project.

CHAPTER 3

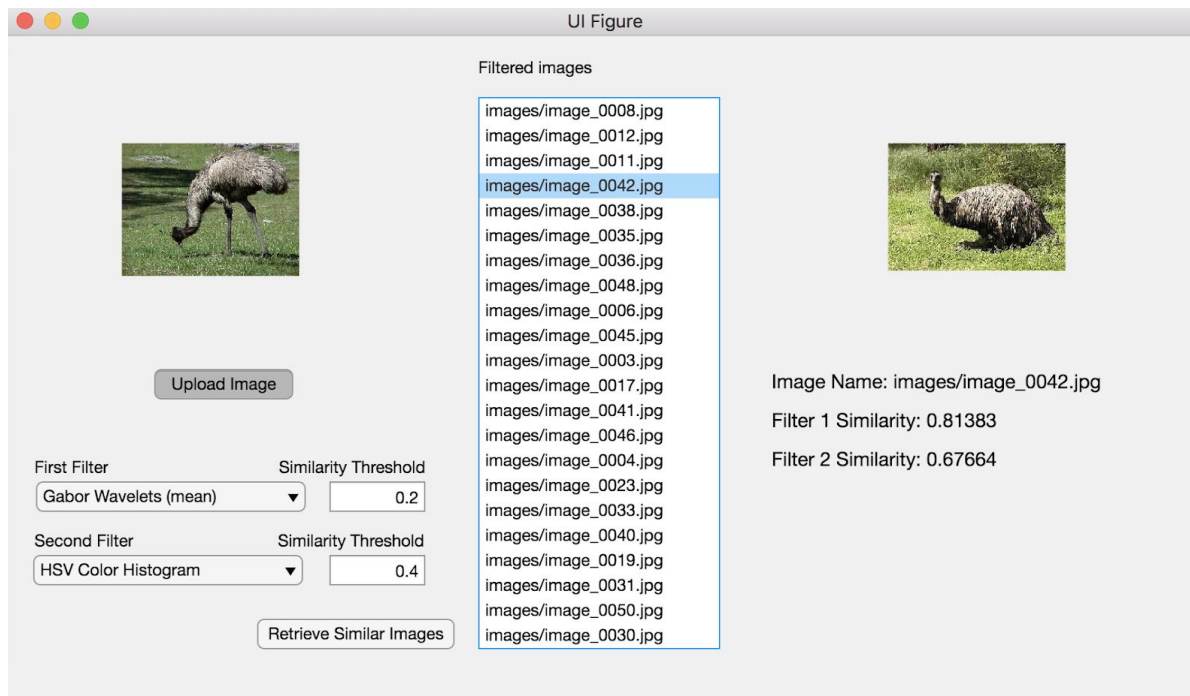
METHODOLOGY

As we are actualizing a CBIR implementation for this project, we have decided upon the following implementation flow and design.

1. **Having a database:** The team have found a database that contains a large dataset of images that is meant for image retrieval purposes. The dataset includes 10,000 low resolution web-crawled test images from the misc database in WBIIS (Li et al. n.d.).
2. **Reading in a query image as input from the user:** The team have retrieved one of the images from the large dataset and set it as a query image. This is acting as a testing at the beginning of the system.
3. **Feature Extraction:** Based on the feature extraction that the team decided, which is local colour histogram and gabor filter, extract each individual images which output would be compare later.
4. **Feature Matching:** After the features are extracted, the similar results are used to compare and then the similarity values are calculated and compared with each other.
5. **Results Checking:** Once the previous steps have been executed, the similarity values are retrieved and the similar images from the database are sorted accordingly. Dissimilar images are also filtered out accordingly based on a certain threshold requirement.

For the system that team have implemented, the basic overview of the system will be that there will be a database of images and the query image that will run through a two-pass filtering process instead of going through only one feature extraction. This ensures a much more accurate results, and also allows the user to choose the type of filtering/feature extraction they want.

3.1 User Interface



The main user interface has a simple and straightforward design. Starting on the right, the user will be able to upload a query image to retrieve similar images. Below the image upload button, the user will be able to select the desired feature descriptors to be used to filter images for first-pass filtering and second-pass filtering respectively. Two dropdown menus with the options: HSV Color Histogram, Gabor Wavelets (mean), Gabor Wavelets (std dev) are available. For each filter-pass, the user is also able to change the similarity threshold value that will be used to filter out dissimilar images below that similarity value threshold. The input box expects a decimal value ranging from 0 to 1, representing 0% to 100% similarity respectively. For instance, if the user gives a similarity threshold of 0.4 for the HSV Color Histogram filter, all of the results shown in the retrieved list of images will be **at least** 40% similar to the query image when their color histograms are compared.

No processing will take place in the background until the user clicks on the 'Retrieve Similar Images' button. When the results are retrieved, it is sorted descendingly according to the similarity values of the second filter. When a image is selected on the list of retrieved images at the center, the section on the right shows the image, the file path, and the similarity values of the two filters.

3.2 Data Preprocessing and Initialization of Matrices

After getting the large database of images for this system, there is a need to preprocess the images. This step is important to make sure every image is standardized in terms of its resolution and number of colour channels.

All images in the database are placed into a directory folder and through the *dir()* built-in MATLAB function, retrieves an array of all .jpg images in that folder. That array will be the list of images in the dataset that we will work with.

```
% Get list of all JPG files in this directory
% DIR returns as a structure array. You will need to use () and . to get
% the file names.
dirName = 'images';
oldFolder = cd(dirName);           % change dir to dirName
imagefiles = dir('*.jpg');         % get image file names in this folder
cd(oldFolder)                     % change dir back to root folder
numberOfFiles = length(imagefiles); % Number of files found
```

Before accessing the directory containing the dataset images, the uploaded query image will have its color histogram feature vector and both Gabor features (mean and standard deviation) calculated and saved to a variable for future comparisons.

Since there are three types of filtering that the user can pick for the two-pass filtering, two $3 \times n$ matrices, with n being the number of images in the dataset, are initialized with zero values. The two matrices, *euclideanDistances* and *similarityValues*, is tasked with storing the values of the calculated euclidean distances and similarity values (in comparison to the query image) respectively. Each row of both matrices stores values depending on which feature extraction method are used; row 1 stores colour histogram values, row 2 stores the mean values from the output of gabor filter, and row 3 stores the standard deviation from the output of the gabor filter. There is also another array that stores file names of all the images, and will be used to display the retrieved images later on.

Each of the images before going through feature extraction which are colour are texture feature extraction are resize to a certain dimensions. By doing so, we are making sure all the images are all relatively similar so that when the features are extracted out, it is based on the same condition. By doing so, the image information does not get lost. To resize the images, there is a simple built in function in MATLAB called *imresize* that receives two values as dimensions.

3.2 HSV-space Colour Histogram

Our project implements a colour histogram that takes account the choices of colour space and quantization level. The reason for using colour histogram is due to the fact that they have high dimensional vector and represent high colour content of an image. For the team's implementation, HSV color model is chosen, as it describes the way how humans naturally perceives and respond to colour. (Chaitanya Vijaykumar Mahamuni, Neha Balasaheb Wagh).

After getting the image information from the preprocessing state, a function called *colorHistogram()* is called from the main function. The images are passed into the function as a parameter. Before an image is passed through into the function, there is a need to check

whether the image is a RGB image, since colour models of HSV consist of 3 colour channels. Gray scale images only have one. If the image is grayscale, the image is converted into a RGB picture that consist of 3 colour channels with another function called `grs2rgb`. This function is an open-source code written by Valeriy R. Korostyshevskiy, PhD of Georgetown University Medical Center, Washington, D.C.

Once the image is rectified to be a RGB image, it gets passed into the `colorHistogram()` function and the RGB image is converted into HSV image with MATLAB built-in function `rgb2hsv()`.

Since the images will be quantized into separate bins, at the beginning, we retrieve each of the variables from the HSV model which is hue, saturation and value.

```
%Split it into three of the HSV
hueI = image(:,:,1);
saturI = image(:,:,2);
valueI = image(:,:,3);
```

Once everything is split accordingly, the V value is equalized before sorting the pixels into quantized bins. The reason for using equalization is so that we could enhance the intensity of the image and adjust it uniformly. By doing so, the images information are greatly enhanced and the comparisons will be much more accurate. The built-in function that helps with this process called `histeq()`.

When it comes to quantizing each of the HSV model in the images, we decided to use $16 \times 4 \times 4$ numbers of quantized bins instead of the classic $8 \times 3 \times 3$ as this provides a higher precision compared to the lower number of bins. We then initialize the HSV array, according to the size of it which is $16 \times 4 \times 4$.

After that is done, the maximum value pixel for each HSV component is found by using the built-in function `max()`. From the figure below, there is two for-loops which will go through the rows and the columns of the image matrix. There are three bins for each component respectively. Each time there is a value similar to the image pixel which is calculated by dividing the current image pixel with the maximum and then multiplying with the dimension set in the beginning (16 or 4). The bin count is then incremented as seen below. It is noted that instead of using 16 or 4, 15.9999 and 3.9999 are used instead. This implementation is to consider the range of the values, as for example, if there is a 0 and it is placed into bin 1, then 0.99999 would go into bin 4, however if $s = 1$ then it would go into bin 5 instead which is not what the team preallocated the histogram bins to be.

important that the waves at specific angle are chosen well. There is aspect ratio parameter that should be considered as well, it specifies the ellipticity of the support of the Gabor function. The bandwidth comes after that, this is the half response spatial frequency bandwidth of a Gabor filter. Finally, there is a parameter that pays attention to the orientation of the waves. It specifies the orientation of the normal to the parallel stripes of a Gabor function and the valid values are between 0 and 360.

The new coordinates of the pixel when applied the Gabor function is defined as :

$$g(x, y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right) + 2\pi j W x\right]$$

In our implementation of the Gabor filter, *myGabor()*, we have attempted in tweaking the values of the Gabor filter parameters in order to get a better accuracy output. However, admittedly it is still not perfect and does not necessarily yield great or accurate results all the time. Our final preset for the Gabor parameters are as below:

```
% preset gabor filter arguments
gamma = 1;      % (aspect ratio)
psi = 0.1;      % (phase)
theta = 90;     % (orientation)
bw = 2.8;       % (bandwidth)
lambda = 3.5;   % (wavelength)
pi = 180;       % (pi)
```

Before applying Gabor filter, a few steps of preprocessing methods are applied onto the image. Being a grayscale filter, of course, the first preprocessing method applied to the image would be the *rgb2gray()* built-in method. After converting it into grayscale, the image is then resized to a resolution of 270 by 480 with the *imresize()* method. This is to make sure that the Gabor calculation process would not be too computationally expensive. Then, *im2double()* is applied, and this effectively doubles the precision of the input image. Lastly, the Gabor function can be finally applied to the image:

```
% Gabor filter size 7x7 and orientation
for x=1:270
    for y=1:480
        x_theta=image_resize(x,y)*cos(theta)+image_resize(x,y)*sin(theta);
        y_theta=image_resize(x,y)*sin(theta)+image_resize(x,y)*cos(theta);
        gb(x,y) = exp(-(x_theta.^2/2*bw^2+ gamma^2*y_theta.^2/2*bw^2))*cos(2*pi/lambda*x_theta+psi);
    end
end
```



The input image and the final output of *myGabor()* function.

The output will yield a 240*480 double matrix. *mean()* and *std()* built-in functions are then applied to get the mean and standard deviation feature vectors, with both having a condensed 1*480 vector size. These feature vectors will then be used in calculating similarity with the query image using Euclidean Distance calculations.

3.4 Calculating Euclidean Distance between Two Images

Every values that have been calculated are stored into the row designated at the beginning. For example, in our implementation, row 1 is for all the values from colour histogram, row 2 is for the mean of gabor filter and row 3 is the standard deviation of gabor filter. Once the values from both feature extractions have been calculated, the feature vector matrices from query image and each image in the dataset are passed into the euclidean distance function. In the function, a simple formula is used in order to get the distance.

$$D(a, b) = \sqrt{\sum_{i=1}^n (b_i - a_i)^2}$$

From the equation above, the difference between two feature vectors are calculated and then squared. After it is squared, the summation of it is retrieved and finally the square root the whole value will output the accurate the distance between the two images. The higher the distance value is, the more dissimilar the two images should be.

```
function result = euclideanDistance(feature1, feature2)

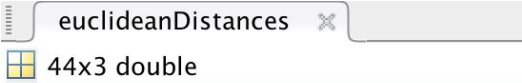
    % Euclidean distance for comparing the feature vector
    score_euclidean = sqrt(sum((feature1-feature2).^2));

    result = score_euclidean;
end
```

The values are returned into the main function. The *euclideanDistances* matrix that stores the results of Euclidean Distance calculations are updated accordingly. The further steps are explained below.

3.5 Computing Similarity Values

After computing the Euclidean Distances, we actually already have all the necessary information that we need to sort and filter out dissimilar images. However, these Euclidean Distance values are found to have no definite maximum range. To visualise this concept properly, recall that the Euclidean distance of two completely similar images would be 0, which makes sense due to the two images having no difference no matter what feature extraction methods are being used to obtain the feature vector. The two images would be considered 100% similar.



	1	2	3
2	1.1311e+05	1.4726	1.2892
3	1.0490e+05	1.7282	2.1325
4	1.0101e+05	1.7525	2.3397
5	1.1381e+05	1.7571	1.8278
6	1.1995e+05	1.7576	0.8709
7	8.6215e+04	1.7742	2.7552
8	1.0191e+05	1.7891	2.5385
9	8.8769e+04	1.8014	1.5300
10	9.9163e+04	1.8101	2.4325

A sample of calculated Euclidean distances. Each column represents results calculated from different feature extraction methods.

When the Euclidean distance of two dissimilar images are calculated, however, depending on the feature extraction method, they produce a range of values with no apparent upper limit. Two completely similar images would still yield a value of 0, which corresponds to a 100% similarity. But the problem that we have faced is: what constitutes a 0% similarity?

Aiming for a simple implementation, we have opted for using the maximum Euclidean distance value as the benchmark value for 0% similarity. This means that a similarity value of 0% would not truly mean that the two images are completely dissimilar, and all similarity values will be calculated in proportion to the most dissimilar image in the dataset.

```
% getting highest EuclideanDist value as the benchmark for 0% similarity
% a Euclidean Distance of 0 will be considered 100% similarity
maxDist = euclideanDistances(length(euclideanDistances),firstPassFilter);

% Computing similarity values for first filter pass
len = length(euclideanDistances);
for ii = 1:len
    eDist = euclideanDistances(ii,firstPassFilter);
    result = 1 - (eDist/maxDist);
    similarityValues(ii,firstPassFilter) = result;
end
```

3.6 Filtering process

There are two filtering process and each process come sequentially one after the other, like a pipelined process.

For both of the filtering process, there is a need to sort the values that the system had returned to us. This is so that the most similar images are return to the user instead. Whichever the feature extraction that the user had pick, all of the euclidean values and similarity values of each individual component is extracted based on which 3 rows the values are stored in.

At the beginning of the first pass filtering process, the euclidean distance are sorted in ascending order and the sorted indexes are saved so that the filenames can be sorted in an exactly same order. Once that is achieved, the maximum values from the euclidean distance is retrieved. The maximum would be needed to calculate the similarity values as stated above.

According to the similarity filtering values and the threshold input, the sorted file names, Euclidean distances and similarity values are spliced from the back, only retaining the images that are relevant enough.

```
% getting the detachment index to filter out dissimilar images.
% this works because all arrays are sorted descendingly by similarity values.
indexToDetach = filteringSimilarity(similarityValues, firstPassFilter, firstPassSimilarityThreshold);

% splicing arrays from the back
if indexToDetach ~= Inf
    similarityValues = similarityValues(1:indexToDetach-1,:);
    euclideanDistances = euclideanDistances(1:indexToDetach-1,:);
    fileNames = fileNames(1:indexToDetach-1,:);
end
```

The second pass filtering procedure is the same as the first. The only difference is that the arrays that had pass through the first pass sorting are used as the input now. The codes for the first pass and the second pass are the same, with just minor difference due to the different output.

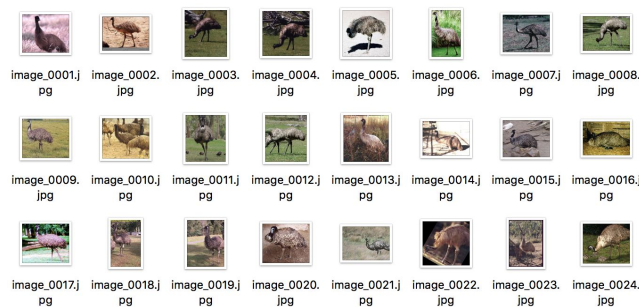
CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Test cases and results

For the output each test case, the first and second lines following the image refers to the similarity values for the first and second filters respectively. Keep in mind that the second filter applied dictates the similarity sorting order of the final output result.

4.1.1 Test Case 1: Controlled Dataset of Images of Emu



For our first test case, we have a controlled dataset of 53 images of emu.

Input query image and parameters:



First-pass filter: Gabor Filter (mean), 0.2 similarity threshold

Second-pass filter: HSV Histogram, 0.4 similarity threshold

Output (top four most similar images):



0.798	0.786	0.814	0.812	(scale of 0-1)
0.886	0.792	0.676	0.663	

As we can see, with HSV histogram as our final filter, the resulting images are visually similar in terms of color. First, emu images with dissimilar textures are filtered out with Gabor filter. This can be seen in the results where all emu possess similar feather textures. Then HSV histogram brings in the emu pictures with similar colors. Although the postures of the emu are not similar, this can be considered a successful retrieval. If postures of the emu are to be concerned, employing shape feature descriptors would yield a better result.

4.1.2 Test Case 2

Input query image and parameters:



First-pass filter: HSV Color Histogram, 0.4 similarity threshold

Second-pass filter: Gabor Filter (std dev) , 0.4 similarity threshold

Output (top four most similar images):



0.461	0.480	0.450	0.441	(scale of 0-1)
0.871	0.854	0.840	0.835	

Although color histogram is first used, the resulting output is not visually similar at all in terms of color. The sorting order has been overridden by the Gabor filter as the second filter. The retrieval can be considered a failure here due to the lack of sensitivity in our implementation of standard deviation of Gabor at certain cases. Either that, or the Gabor Filter just detects the wrong textural information.

4.1.3 Test Case 3

Input query image and parameters:



First-pass filter: Gabor Filter (mean), 0.2 similarity threshold

Second-pass filter: HSV Histogram , 0.4 similarity threshold

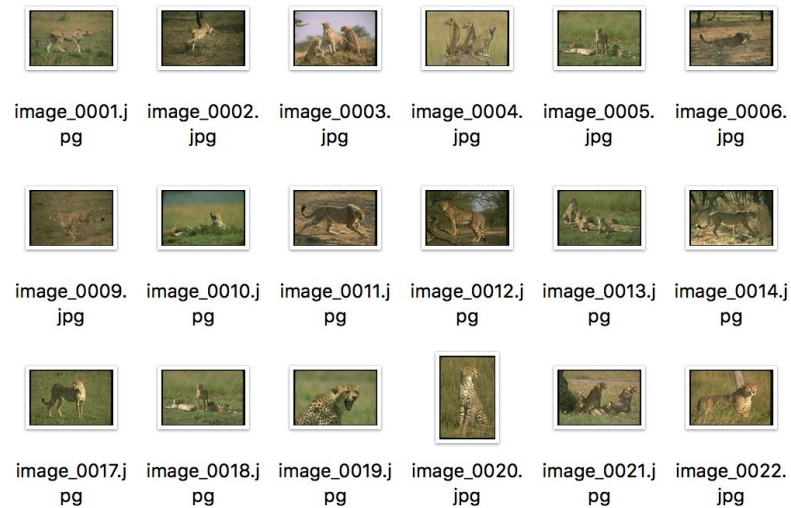
Output (top four most similar images):



0.798	0.786	0.814	0.812	(scale of 0-1)
0.886	0.792	0.676	0.663	

Apparently, using mean of Gabor filter almost guarantees results with similar textures. All four images has a high similarity value for mean of Gabor, and has grass and feather elements in it. In relation to that, all four images are also made up of a feathery creature surrounded in a grassy environment. HSV Histogram is then utilized to provide a final sorting of the images based on its color. This image retrieval case can be declared a success.

4.1.4 Test Case 4: Dataset of 296 Wild life images



This second dataset is filled with 296 images of wildlife, primarily animals like tigers, ox, leopards, cheetahs, and panthers roaming in grasslands and up in the trees. It also includes some images of flowers.

Input query image and parameters:



First-pass filter: Gabor Filter (mean), 0.4 similarity threshold

Second-pass filter: HSV Histogram, 0.5 similarity threshold

Output (top four most similar images):



0.595

0.474

(scale of 0-1)

0.986

0.505

For this case, we have set a very high threshold for both passes of filters and as a result, only 2 images are retrieved. In our implementation, Gabor filters are found to yield very unforgiving similarity values. By setting a high similarity threshold, the user is at risk at receiving very little results as they are all cut away by the filtering process. This is one of the shortcomings of our system that we have realized upon completion of our implementation. Further tweakings of the Gabor filter parameters have to be made in order to improve upon this shortcoming.

4.1.5 Test Case 5

Input query image and parameters:



First-pass filter: Gabor Filter (mean), 0.4 similarity threshold

Second-pass filter: Gabor Filter (std dev), 0.4 similarity threshold

Output (top four most similar images):



0.596	0.607	0.468	0.455	(scale of 0-1)
0.759	0.744	0.739	0.735	

Using the same input image as the previous case, we have completely disregarded color features, and went for both Gabor filters. Results have shown that only leopard images have shown up. This might be due to Gabor filter picking up the dotted texture from the leopard's skin, and using that as one of the retrieval features. Although the leopards have different postures and background colors, it is only fair because color histogram was not employed here and we could only expect a color invariant result.

4.1.6 Test Case 6

Input query image and parameters:



First-pass filter: HSV Color Histogram, 0.2 similarity threshold

Second-pass filter: Gabor Filter (std dev), 0.4 similarity threshold

Output (top four most similar images):



0.244	0.383	0.362	0.380	(scale of 0-1)
0.607	0.605	0.582	0.578	

Here, we have uploaded a random group photo of some rock band, totally irrelevant to the context of the dataset. Understandingly, the retrieval was a failure as Gabor filters were used as the second filter, and all the images in the dataset generally have rough and grassy texture, while the query image has a generally smoother texture. This failure was an expected one.

4.1.7 Test Case 7

Input query image and parameters:



First-pass filter: Gabor Filter (std dev), 0.2 similarity threshold

Second-pass filter: Gabor Filter (mean), 0.4 similarity threshold

Output (top four most similar images):



0.848

0.794



0.255

0.794



0.724

0.794



0.665

0.790

(scale of 0-1)

For this last case, the query image is a rough grassy patch of land with a bear in the middle of it. We have once again disregarded color features over here, and employed the standard deviation Gabor first before applying the mean of Gabor. Results show visually and texturally similar images, and with high similarity values too. All images are of rough grassy terrain, except result image #2 which is of mountainous terrain. Still, all images contain a bear in the center of the image. One interesting observation to make is the low similarity score for standard deviation of Gabor in image #2, which is clearly the odd one out being in the mountains. This could be an indication where standard deviation of Gabor picks up textures of background terrain more sensitively. Overall, this case is considered a successful one.

4.2 Limitations of this CBIR system

There are a couple of limitations that the system that the team implemented have that may cause inaccuracy result in the final few test cases of our system.

4.2.1 The HSV model

The reason why the HSV model is a limitation to the system is because of the advantages that HSV colour model have provided to the image processing industry. Since HSV only focuses on the Hue, Saturation and Intensity, it does not fully retrieve the colour from the image. If there are two colours in each image respectively that have similar value of saturation or intensity, however the colour scheme is not similar, it may be put into the same range as each other. This causes the similarity values to be high even though it is not when we look at it.

A similar problem that may occur is when the colour scheme is the same, however due to the intensity or saturation, it may be labelled as dissimilar. For example, from a human visual perception, we see both images as green, however due to the saturation of it, it may cause a lower similarity values.

4.2.2 Gabor Filter Limitations

One of the main reason for the relatively inaccurate results is due to the Gabor filter that the team have decided to implement. The Gabor filter that we had implemented are based on our own basic understanding of the function, since we didn't use the built-in function of gabor filter. The implementation of the Gabor Filter is simple, and does not take account a lot of the more lesser details that may impact the accuracy of the images. The other reason is due to the fact that the mean and the standard deviation was calculated with a simple algorithm, which have proven that isn't the most accurate. The mean and standard deviation are separated entity instead of together, therefore, when it comes to using gabor filter, we have two options that are not similar with each other.

From our further research after the complete implementation of the system, we realised that a better method would be to use Wavelet Transform Feature. Wavelet Transform Feature takes account of each level, the low and high pass, and extract the texture. This makes the output much more accurate.

4.2.3 Grayscale images

One of the setbacks that the team did not envisioned when implementing this system is that the grayscale image have only one colour channel, therefore many of the functions such as colour histogram does not take in the grayscale image which may cause an issue as in order to convert into a HSV model, it needs to have 3 colour channels. Therefore, the team have decided to convert the grayscale image to a RGB colour image. However, the RGB image will

be in the colour map of “HOT” therefore the image are not in the grayscale colour instead in high saturation of red and orange.

4.2.4 Mean and standard deviation

The results of mean and standard deviation for when it goes through gabor filter is very much different from each other. Therefore, we are not sure that which results are deemed the more accurate one. Sometimes the gabor filter we implemented is not consider accurate as well due to the outline of some objects being very similar. The orientations and the scales end up being very similar despite the different object in the image. For example, a deer and a bear may have similar outline as seen from the images below. Despite them being different creature, the Gabor Filter will place them into the same category.



CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Recommendations

The system that the team have implemented have flaws and by no means, meant to be taken as an addition of research in the industry. However, there is some recommendations that the team have come across that may further improve this system's accuracy.

One of the recommendation is to consider a better Gabor algorithm such as Wavelet Transform Feature , Local Binary Pattern (LBP) or Discrete Wavelet Transform. All of these algorithm takes account on more details and focuses on improving the disadvantages of Gabor Filter.

Another recommendation that could further improve the system is making it work for grayscale images. In many of the dataset that uses CBIR, there are many grayscale images such as medical images or crime-scene images used in forensic science. If the image after converting into RGB is not it's original colour scheme that it would deemed the CBIR worthless. Besides there are other ideas on the flow of the system, instead of a pipeline design, such as combining the feature vector of the output of both feature extraction. Perhaps by doing so may increase the accuracy of the outputs since the filtering will consider both of the similarity values.

5.2 Conclusion

For this implementation, we have managed to achieve our aims and objectives despite how it does not output the most accurate results. We as a team further understand how the backend of a CBIR system work and have much more respect for all the researchers working in order to better the system and the industry that uses it.

It is clear from the implementation that even a simple local colour histogram can extract the right values to compare the query image with the images in the database. However, it does not give a 100% accurate results. By adding another feature extraction to the algorithm not only helps filter the unnecessary images, in fact helps a bit with the results. However, it is noted that different feature extraction have different advantages and it is up to the team to decide which one would be more suitable in terms of which industry they are contributing to.

There are many improvement that could be made on each feature extraction, and despite the many efforts, there are still many feature extraction that haven't reach its full potential. Therefore, it is very important that researchers are fully aware and keep track on any new article that worked on CBIR.

REFERENCES

- Anandh, A., Mala, K., & Suganya, S. (2016). Content based image retrieval system based on semantic information using color, texture and shape features. In *2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16)* (pp. 1–8).
- Chawki, Y., El Asnaoui, K., Ouanan, M., & Aksasse, B. (2018). Content-Based Image Retrieval Using Gabor Filters and 2-D ESPRIT Method. In *Advanced Information Technology, Services and Systems* (pp. 95–102). Springer International Publishing.
- Jain, A., Muthuganapathy, R., & Ramani, K. (n.d.). Content-Based Image Retrieval Using Shape and Depth from an Engineering Database. In *Lecture Notes in Computer Science* (pp. 255–264).
- Kaur, M., & Sohi, N. (2016). A novel technique for content based image retrieval using color, texture and edge features. In *2016 International Conference on Communication and Electronics Systems (ICCES)* (pp. 1–7).
- Li, J., Member, Wang, J. Z., & Member. (n.d.). Automatic Linguistic Indexing of Pictures by a Statistical Modeling Approach.
- Neetu Sharma, S., Paresh Rawat, S., & Jaikaran Singh, S. (2011). Efficient CBIR Using Color Histogram Processing. *Signal & Image Processing : An International Journal*, 2(1), 94–112.
- Singh, A., Sohoni, P., & Kumar, M. (2014). A Review of Different Content Based Image Retrieval Techniques.
- Stricker, M., & Orengo, M. (1995). Similarity of color images.
- Swain, M., & Ballard, D. (1991). Color indexing. *International Journal of Computer Vision*, 7:1, 11-32.
- Tunga, S., Jayadevappa, D., & Gururaj, C. (n.d.). An Improved Technique of CBIR using Gabor Filter and Genetic Algorithm.
- Zhang, D., Wong, A., Indrawan, M., & Lu, G. (n.d.). Content-based Image Retrieval Using Gabor Texture Features.