CONTENT BASED IMAGE RETRIEVAL FOR IDENTIFICATION OF PLANTS USING COLOR, TEXTURE AND SHAPE FEATURES

by

Hanife Kebapcı

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University

August 2009

Content Based Image Retrieval for Identification of Plants
Using Color, Texture and Shape Features

APPROVED BY

Assoc.Prof. Berrin Yanıkoğlu .....

(Thesis Supervisor)

Assist.Prof. Gözde Ünal .....
(Thesis Co-Supervisor)

Assist.Prof. Hakan Erdoğan .....

Assist.Prof. Hüsnü Yenigün .....

Dr. Devrim Ünay .....

DATE OF APPROVAL: .....

Content Based Image Retrieval for Identification of Plants Using Color, Texture and Shape Features

Hanife Kebapcı
Comp.Sci.&Eng., Master's Thesis, 2009
Thesis Supervisor: Berrin Yanıkoğlu

Keywords: Image Retrieval, Feature Extraction, Color Features, Gabor Wavelets, Contour-based Shape Features

Abstract

In this thesis, an application of content-based image retrieval is proposed for plant identification, along with a preliminary implementation. The system takes a plant image as input and finds the matching plant from a plant image database and is intended to provide users a simple method to locate information about their plants. With a larger database, the system might be used by biologists, as an easy way to access to plant databases.

Max-flow min-cut technique is used as the image segmentation method to separate the plant from the background of the image, so as to extract the general structure of the plant. Various color, texture and shape features extracted from the segmented plant region are used in matching images to the database. Color and texture analysis are based on commonly used features, namely color histograms in different color spaces, color co-occurrence matrices and Gabor texture maps. As for shape, we introduce some new descriptors to capture the outer contour characteristics of a plant. While color is very useful in many CBIR problems, in this particular problem, it introduces some challenges as well, since many plants just differ in the particular hue of the green color. As for shape and texture analysis, the difficulty stems from the fact that the plant is composed of many leaves, resulting in a complex and variable outer contour and texture. For texture analysis, we tried to capture leaf-level information using smaller shape regions or patches. Patch size is designed to contain a leaf structure approximately.

Results show that for 54% of the queries, the correct plant image is retrieved among the top-15 results, using our database of 380 plants from 78 different plant types. Moreover, the tests are also performed on a clean database in which all the plant images have smooth shape descriptors and are among the 380 images. The test results obtained using this clean database increased the top-15 retrieval probability to 68%.

Bitki Tanımaya Yönelik Renk, Doku ve Şekil Sistemlerini Kullanan İçerik Tabanlı Görüntü Bulma Sistemi

Hanife Kebapcı
Bilg.Bil.&Müh., Yüksek Lisans Tezi, 2009
Thesis Supervisor: Berrin Yanıkoğlu

Keywords: Görüntü Erişimi, Öznitelik Çıkarma, Renk Öznitelikleri, Gabor Dalgacıkları, Çevrit-tabanlı Şekil Öznitelikleri

Özet

Bu tez çalışmasında, bitki tanımaya yönelik bir İçerik Tabanlı Görüntü Bulma sistemi önerildi ve bu sistemin başlangıç uygulaması geliştirildi. Kullanıcılara kendi bitkileri hakkında çeşitli bilgiler sunmayı amaçlayan sistem, oluşturulmuş bitki veritabanı içerisinden, kullanıcıdan aldığı bitki resmiyle eşleşen bitkiyi bulur. Hazırlanan sistem, daha geniş ve kapsamlı bir veritabanıyla birlikte biyologlar tarafından bitki veritabanlarına daha kolay erişim sağlamak için de kullanılabilir.

Görüntüdeki bitkiyi görüntünün arkaplanından ayırmak, böylece bitkinin yapısını çıkarabilmek için görüntü bölütleme yöntemi olarak maksimum-akı minimum-kesik yöntemi kullanıldı. Çeşitli renk, doku ve şekil öznitelikleri, girdi olarak alınan görüntüleri veritabanındakilerle eşleştirmede kullanılmak üzere, bölütlenmiş bitki bölgesinden çıkarıldı. Renk ve doku analizi için kullanılan yöntemler, bilinen ve çoğunlukla tercih edilen özniteliklere dayanıyor. Bu öznitelikler: renk histogramları, renk eş görülme matrisleri ve Gabor doku haritalarıdır. Şekil öznitelikleri için ise bitkinin dış hatlarını ve özelliklerini ifade edebilecek bazı yeni şekil açıklayıcıları sunuldu. Renk, görüntü bulma sistemlerinde çok etkili bir öznitelik olmasına rağmen, bu problemde çoğu bitkilerin birbirinden yalnızca yeşilin tonlarıyla farklılık göstermesi renk analizi konusunda bir zorluk olarak görüldü. Şekil ve doku analizindeki zorluk ise, bitkilerin pek çok yapraktan oluşması ve bu sebeple bitki doku ve dış hat görüntüsünün karmaşık ve değişken olmasıdır. Doku analizinde yaprak seviyesindeki

görüntü bilgisini yakalayabilmek için küçük şekil parçaları ve yamalar kullanıldı. Her bir yamanın bitkinin yaprağını ifade edebilecek büyüklükte olması amaçlandı.

78 farklı bitki tipinden 380 görüntü barındıran bitki veritabanımızda yaptığımız testler sonucu, doğru bitki %54 olasılıkla eşleşen ilk 15 bitki arasında yer aldı. Bunun yanında, sadece bu veritabanındaki şekil öznitellikleri iyi olan görüntülerin olduğu 132 görüntüden oluşan veritabanından alınan sonuçlar, doğru bitkinin ilk 15 bitki arasında olma olasılığını %68'e çıkarmıştır.

to my family

Acknowledgements

I wish to express my gratitude to,

Berrin Yanıkoğlu, for her valuable advices, infinite patience and support,

Gözde Ünal, for her confidence and support,

all my jury members Hakan Erdoğan, Hüsnü Yenigün and Devrim Ünay for reading and commenting on this thesis,

Berkay Kaya and Burak Karaboğa for providing an initial GUI application that is used in this thesis,

our project team: Arif, Burak and Ercan for their effort to collect our plant image database,

my sister Çiğdem and my friends for their valuable comments, and helps that facilitated my writing,

last, but not the least, to my family, for their enormous encouragement and patience, for without them, this work would not have been possible.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

Due to the rapid improvement in technology -especially related to the Internet- and spread in usage of digital cameras, the number of images in digital platforms has increased tremendously in the last decades. Websites devoted to images also increase in number everyday; e-newspapers, digital image libraries, photo sharing websites, personal web albums are some examples showing the prevalent usage of digital images today. Frequent and common use of digital images and the sheer number of images have brought the need for efficient indexing, classifying and searching algorithms. The earliest image search applications used the text on websites and in the image filenames, to extend text search capabilities for image searching. Since the performance of text-based image retrieval depends on the existence and relevance of text, this approach is often insufficient in finding desired images.

Image *annotation* or *tagging* is also used to help image retrieval systems. This method is still widely used in photo sharing systems such as Flickr; digital art sharing websites such as deviantart; social networks such as facebook and by Google. All of these applications have huge amounts of digital images and manage them in some form of tagging. Some of these systems offer web-based games to encourage image annotation, for instance the Google Image Labeler [1] is played by two parties where

[1] http://images.google.com/imagelabeler/

each person tries to label the same image appropriately at the same time. These annotation systems are based on manual tagging which is very slow with respect to the increase in the number of images. Studies conducted to encounter that problem brought a new subject to agenda: *automatic annotation*, which is in a different context than text-based approaches.

The problem with the aforementioned methods is that they do not use the visual information of images. While Image Retrieval (IR) refers to the general problem of searching and retrieving images, *Content based image retrieval* (CBIR) is the problem of retrieving relevant images based on their content. CBIR offers efficient search and retrieval of images based on their content. Two important query categories can be distinguished: i) query by example and ii) semantic retrieval using a description of the search concept (e.g. find images containing bicycles). Query by example is often executed by comparing images with respect to low level features obtained from the whole image, such as color, texture or shape features. Semantic retrieval on the other hand requires higher level understanding of the image contents which requires a more local approach. For instance, local features such as scale-invariant feature transform (SIFT) descriptors can be used in locating objects within complex scenes. These two broad categories can be further subdivided. For instance the query by example can be done by providing a sketch or a template, instead of an image. Similarly, the semantic retrieval can be made in different levels of abstraction of the query concept (e.g. bicycle) [4].

Research on CBIR has shown its first significant results with feature-based systems in early 1990s [5–7]. Commonly used features can be grouped as color, texture, shape, and location features. Examining images based on color is one of the most widely used technique, partly due to its simplicity. Color matching between two images can be done simply by using a color histogram over the whole image or over a fixed region of the image (e.g. find sky in the top half of images). More complex color features may involve looking into spatial relationship of multiple colors or looking at the color histograms in automatically segmented regions of the

image. Other widely used features can be grouped as texture features and shape features. Texture can be described as spatial patterns formed by color or grayscale variations that are often uniform over a region. Texture analysis and matching can be done using various techniques such as Gabor filter which are linear image filters in the form of a wavelet convolving a Gaussian and a harmonic function, and local binary patterns (LBP) which describes the texture in terms of small pixel intensity groups and their relative position statistics by focusing on a local neighborhood in the image. Finally, shape measures may be used to find a particular shape in the queried images. Shape measures often use segmentation and edge detection, as they refer to objects within the image. Recent research on CBIR moved more towards semantic analysis of content (e.g. [8]) from low level features. Also, in order to improve usability, relevance feedback was later developed to give the system feedback from the user. Recent survey articles summarize the latest research activities in the field [8–10].

While there are some plant images in the commonly used image retrieval databases (e.g. the Corel database) [2], we are not aware of a CBIR system geared specifically towards house plant retrieval. However, there are some related work in the areas of plant classification and identification that are developed for botanical or agricultural needs. In systems geared towards botanical applications [11–18], clean leaf images are used to identify unknown plant varieties, using features obtained from the leaf contour. Yahiaoui et.al. proposed an image retrieval system for identifying plant genes by using contour-based shape features in [11]. The extracted shape descriptors in this study include the length histogram of contour segments in different directions. Another work on plant image retrieval ([13, 14]) focused on the leaf image retrieval problem using features such as centroid-contour distance (CCD) curve, eccentricity, and angle code histograms (ACH). These features are extracted from the leaf edge contour after some preprocessing (e.g. scale normalization). In some recent work ([12, 15]), the retrieval algorithm is supported with machine learning

[2]Besides Corel database, Caltech vision group has a *Leaves* database containing 186 images of 3 species only http://www.vision.caltech.edu/html-files/archive.html

techniques. In [12], plant leaves are classified based on their texture features: LBP, and Gabor wavelets are used together. Local texture features of plant leaves are extracted with the LBP operator using the Gabor filtered image. Then extracted texture features (spatial histograms) are fed to support vector machine (SVM) classifier. The study in [15], combined color and texture features (i.e. color moments and wavelet transform) after a preprocessing task which normalized the rotation of leaves (all looking to the same direction). SVM classifier is trained with extracted color and texture features, then used to recognize plants.

Systems geared towards agricultural applications include detecting weeds in the field [19], detecting position of specific plants [20], and deciding whether or not a plant is damaged by a specified illness [21] are frequent applications in this area. In [19] color and shape information is used to detect weeds in the field. Sena's work [21] on identifying damaged maize plants proposes a segmentation step to be used first. Leaf segmentation is done by thresholding the monochrome images that are converted from RGB using a transformation called the normalized excess green index ($2g - r - b$, where g, r, and b are corresponding RGB color channels [22]) to distinguish weeds from soil regions. In [20], position of a maize plant on the field is located by finding the center of the plant by intersecting the detected main vein lines of leaves. Vein lines in turn are detected by using reflectance difference of veins and leaves.

The aim of aforementioned agricultural image retrieval applications typically is to detect position of a plant or an illness on the plant which is different from our intention. While the botanical image retrieval systems stated above use various descriptors extracted only from the plant leaf, in contrast, in our system, we use the overall plant information.

In this thesis, we present a CBIR system for identifying a plant. The system works by matching a query image to all the plant images in the database, after background segmentation. The system can be used as a web service by people who

may want to obtain information about their house plants. The future web service may be designed to work by receiving a sample plant image from the user, then using the image as the query to search images from the same plant type. The system works by receiving a sample plant image from the user, then using the image as the query to search images from the same plant type. At the end of the search process, images of most similar plants are provided on the user interface. Ideally, in identification problems one would like to retrieve only the searched plant; however, since this is often not possible, the top-N images are returned to the user. In order to have more realistic and user-friendly system, top-15 plants are provided to the user which can easily fit into the applications' browsing area. In the envisioned application, the user will then browse the 1-page returned images and pick the correct plant, which in turn will bring information about that plant. The plant information might be collected from trusted botanic resources.

As humans perceive and identify plants by high level features (e.g. one can say: chlorophytum comosum (spider plant) has long leaves with green and white stripes), image retrieval systems intend to acquire high level features by referring to the low level features extracted from the images. In order to extract the general structure of the plant, max-flow min-cut technique, which is a fast and satisfying method, is used as the image segmentation method to separate the plant from the background of the image. Common feature extraction methods are used for color and texture analysis. However, a new shape descriptor is proposed in this work which represents the outer shape of the plant region. In order to provide the sufficiency of image descriptors, various color, texture and shape features extracted from the segmented plant region are used in matching images to the database. In other words, color information of the plant is complemented with the texture and the shape information. For example, the stripes on the leaves of the spider plant are captured by Gabor texture analysis, while white and green color values of spider plant leaves are represented in color histograms. The plant body structure which is characterized with the name *spider*, is represented by the structure and spread of leaves with our shape-based features.

Contributions of this thesis can be summarized as follows:

- development of a plant identification problem as a particular application of CBIR,
- using the segmented plant region for feature extraction that removes the noise effect created by the background, which increases the quality of the extracted features,
- evaluating different features for their effect on overall system performance,
- proposing some new shape descriptors that provide the outer contour characteristics of a plant.

The outline of this thesis is as follows. The segmentation algorithm used for separating the background is Boykov and Kolmogorov's graph-based segmentation technique which is explained in Chapter 2, along with our implementation specifications. Chapter 3 gives an overview of various color, texture and shape features which are used in this thesis. In addition to basic features such as color histograms, we present our new shape features which are extracted from the overall contour of a plant. Besides general information on these features, our approaches are expanded by also considering the problems we have encountered. How image features are used to compute a similarity between two images is explained in Chapter 4. In Chapter 5, our plant image database is introduced. The database consists of 530 images, of which 380 of them are manually segmented by our project workers . Chapter 6 presents the results of evaluations of various feature combinations. The success rates, average minimum correct retrieval ranks are provided for different test methods. Finally, we conclude with a discussion of this work and its success in Chapter 7, along with future work ideas.

Chapter 2

IMAGE SEGMENTATION

Image segmentation is the task of subdividing an image into its constituent regions or objects [23]. Segmentation is necessary to separate the foreground from background and is used in wide variety of recognition and retrieval problems, such as optical character recognition (OCR) or medical imaging [24]. Figure 2.1 shows results of segmentation showing the object boundaries.

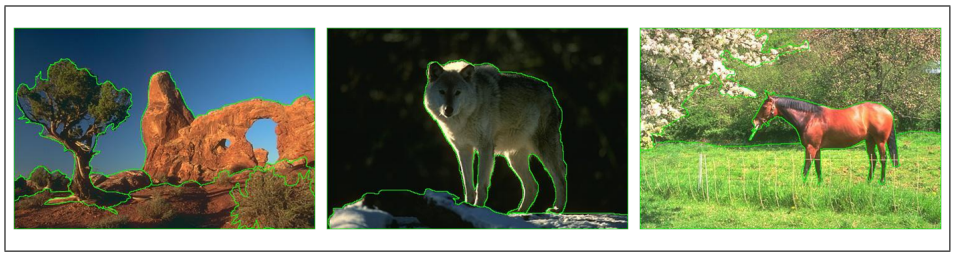

Figure 2.1: Segmented image examples from [1]

This is done in order to extract more significant low-level information (color, texture), as well as to extract the object contour which is used in calculating the shape features. As mentioned before, segmentation is an important part of our proposed system since background regions affect the quality of the extracted features. Both the images in our database and the query plant images (may) contain background

information. With a segmentation preprocess, only characteristic information of the plant is used in matching.

While different in their approaches, all segmentation algorithms use low-level information such as color, texture or intensity changes. There are two contrasting approaches to segmentation. In the first one, regions showing high variations, such as edges, are detected and used in locating object/region boundaries. In the second approach, similar or homogenic regions are expanded, combining similar pixels to form larger segments. Edge linking, edge following, thresholding are examples that use discontinuity, while region growing, region merging are characteristic examples that use similarity of pixels.

We mentioned that color and texture information are the most common visual information used in image segmentation, but the way they are used changes. In terms of the segmentation methodology, the techniques can be grouped as: i) Histogram-based ii) Clustering-based, iii) Region growing, iv) Split-and-merge, v) Morphological and vi) Graph-based. Histogram-based approaches use intensity distribution of pixels in order to find regions of uniform histogram characteristics. Clustering-based approaches feed the pixel intensity values to a clustering algorithm and produce region clusters on the image. Blobworld [25] is the most famous implementation of this method. In region growing, homogenic pixels are connected to form a segment and growing is stopped when irrelevancy reaches a specified limit. Graph-based segmentation techniques represent the image as an arc-weighted directed graph where pixels are graph nodes and pixel intensities are edges of the graph. Segmentation is completed by labelling all graph nodes as one of the two classes: background and foreground. The segmentation method we use (max-flow min-cut method) is a special method of graph-based image segmentation as explained in Section 2.1.

2.1 Max-Flow Min-Cut Method

Consider a directed graph $G(V, E)$ where V indicates the vertices and E indicates the edges between the vertices. The *cut* operation splits the graph nodes into two disjoint sets S and T. Capacity of a cut is defined as:

$$c(S,T) = \sum_{u \in S, v \in T | (u,v) \in E} c(u,v)$$

where u and v indicate the edges in S and T respectively, and $c(u, v)$ denotes the capacity between u and v.

The max-flow min-cut algorithm considers an image as a finite graph in which pixels form the nodes or vertices of the graph and neighboring pixels are connected with an edge. The intensity difference between two neighboring pixels u and v determine the edge weight, or $c(u, v)$.

The algorithm requires seed plant and background pixels (sink and source respectively) to be specified. The selected seeds form the initial values of the sets S and T. The max-flow min-cut segmentation algorithm splits the graph into two disjoint sets S (source) and T (sink) minimizing a cost functional. The output corresponds to a binary labelling of the image with foreground and background regions. The functional is based on two values: i) a spatial smoothness term which measures the cost of assigning the same label (e.g. foreground or background) to adjacent pixels, and ii) an observed data term that measures the cost of assigning a label to each pixel. The graph cut algorithm maximizes the *flow* between the source and sink nodes or equivalently finds a cut through the graph which minimizes the total cost of the graph edges on the cut as explained. This graph cut technique is derived from Megner's theorem [26], which proves that maximum amount of flow in a graph (or network) equals the capacity of the minimum cut in that graph.

Max-flow min-cut graph cut technique is one of the most preferred segmentation

approach in vision problems. An important feature of this segmentation method is the fact that minimizing energy functions is easy and efficient. In terms of implementation, there are various approaches for solving the max-flow min-cut problems on directed weighted graphs such as the *augmenting path* proposed by Ford-Fulkerson, the *push-relabel* method and a new method proposed by Boykov and Kolmogorov [2] which is a modified version of the augmenting path method. In this thesis, Boykov and Kolmogorov's technique is used, as it is the most efficient, hence the most preferred method today.

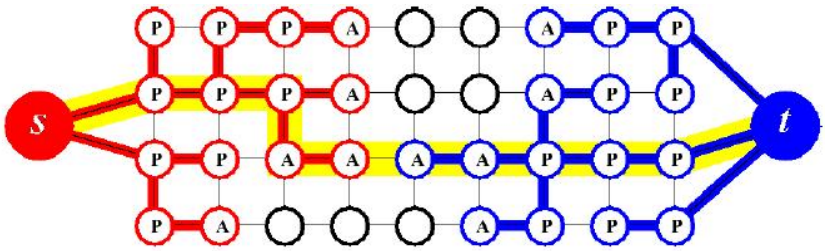

Figure 2.2: Example search tree of algorithm given in [2] at the end of growth stage.

Figure 2.2, which is adapted from the Boykov and Kolmogorov's paper [2], is used to explain the algorithm. The nodes s and t (source and target nodes respectively) are the roots of S and T trees which are grown. Red or blue colored nodes indicate that they are elements of either trees. The unlabeled nodes are the free nodes that can be labeled by either sets. The active nodes (expressed by A) are in the active growth stage in which the neighboring nodes will be visited. The nodes labeled as P are the passive nodes that are labeled in either trees and will not grow. The free nodes will be labelled iteratively through active nodes, so S and T trees will be grown till an $s-t$ path occurs. This step is named as *growth stage* which is followed by *augmentation stage*. Path found in the growth stage is augmented. Since maximum flow is tried to be achieved, some edges in the path may become orphans and trees may be divided into pieces. In order to reconstruct the tree structures of S and T correctly, a third step is designed that is named as *adoption stage*. One of the

important reasons of why this new method is better than standard ones is that a dynamic tree algorithm is designed which grows in two directions (from both source and target). Currently, max-flow min-cut method gives the best performance for vision problems.

Figure 2.3: Segmentation examples from our database: The input image, the seed (sink and source) map and the segmented image result are shown. (Sink seed regions are displayed as red, source seed regions are displayed as white.)

Figure 2.3 shows sample segmentation results on two plant images from our database. Source and target seed points are marked by drawing closed regions by clicking region edges. Currently the seed and background selection is carried out manually, using a MATLAB GUI program we have implemented. In this system, defining 5 seed regions on the average requires to select 15 points that define the closed regions. As a future work, we aim to develop automatic or semi-automatic approaches for the segmentation or the selection process.

The output of the segmentation is an image where the non-plant regions are marked as black pixels (RGB(0,0,0)), to be discarded in feature extraction step, while the plant region retains the original image pixels. Prior to segmentation, the black pixels in the image is assigned with the color value RGB(1,1,1), so as to allow

for this efficient in-place segmentation. Note that this modification is harmless since it is a small change and occurs in all images to be compared.

2.2 Challenges

Figure 2.4: Noisy segmentation examples from our database. The main difficulty of the first plant image is, the plant region on the background of the main plant, while the difficulty in second image is, close representations of leaf and rock regions in gray-scale.

Noisy image characteristics present one of the major challenges to the plant image segmentation problem. Challenging cases are due to a textured background or a continuous plant region. In order to prevent this, we can define some constraints for input (query) images from users. Secondly, the implemented method of Boykov and Kolmogorov uses gray-scale color information and 256 intensity values is inadequate in several cases. For instance, the plant and the background regions may have close intensities such as in a dark-leaf-plant standing in front of a dark wall, or

even various real colors having the same intensity in gray-scale. Using RGB or HSI color models and measuring the energy values between pixels according to 3D color information (24 bits rather than 8 bits) might increase the accuracy of segmentation. Although this modification will not make the segmentation of green plants within a green region easier (i.e. Figure 2.4 a,b,c), it is expected to improve the quality of segmentation in other cases such as the example shown in Figure 2.4 d,e,f. In conclusion, for further study, a similar graph cut method might be implemented in 3D data using RGB or HSI channel values of each pixel rather than gray-level. Having RGB or HSI values will increase the data three times, hence increase the accuracy.

Chapter 3

FEATURE EXTRACTION

In the system we developed, images are analyzed using various color, texture, and shape features. Color, is an important feature in all CBIR applications and the same applies for plant image retrieval problem as well. However, the use of color in plant retrieval is more complicated compared to most other CBIR applications, since most plants have green tones as their main color.Furthermore, the color of the flowering problems also poses a challenge: a flowering plant should be matched despite differences in flower colors. For instance given an orchid of a certain color, one ideally should find its exact match from the database, as well as other orchid plants with different flower colors. The texture information due to colors and veins of the plants, is also important in plant identification. In the current system, we experimented with different Gabor wavelets in order to extract texture information. Third important core feature for the plant images is shape-based features. The outer contour of the plant is extracted using a contour tracing algorithm, starting from the segmented image. Using this extracted contour, several features are extracted about the shape of the plant and its leaves. This section details the feature extraction process.

3.1 Color Analysis Techniques

Color is the most important, common, and primary feature of an image. That is why the earliest image retrieval studies used color as distinguishing comparing feature between images [10,27,28]. Certain objects or scenes have particular colors: e.g. sky is blue, grass is green, or lemon is yellow. If the problem is to distinguish whether an image has sky as part of it, blue values of the color histograms give information about this existence. On the other hand, other entities, buildings, cars, or flowers may also have the blue color. Another complication is that images do not consist of one object and one color, but of many elements usually. Even the image is a photograph of the seaside, it may additionally have trees, sand, rocks, animals, or people. The seaside images in Figure 3.1 have different sub-elements, hence they consist of different colors beside blue.

The color histogram shows the color spectrum of the image, or the distribution (in terms of frequency) of various colors [23]. To compute the histogram, we first decide on the number of bins to represent the colors. A higher number of bins represent the distribution in a higher color resolution, but a lower number of bins is more robust to small color variations. Color sensitivity of color histograms varies and they are called as n-bin color histogram if the color map is quantized to n distinct regions. Histograms are typically normalized by the total number of pixels in the image, so as to represent the color distribution as a percentage of the number of pixels in the image.

Figure 3.1: Various images having similar content but different color distributions.

Another point of consideration is the representation in different of color formats. Black & White images are represented by binary representation, while gray images are represented by 8 bit representation (256 different colors) and color images are represented by 24 or 32 bits of information. RGB, the current standard format for computers and TV screens, generate colors by combining red, green and blue lights. Another color format, CMYK, is designed for printing purposes. In addition, there exist other color spaces that are modelled for specific purposes. Hence, each color model has some usage advantages. Likewise, normalized-RGB (nRGB) and HSI color models are examples that are often used in CBIR problems.

3.1.1 RGB Color Space

Many systems such as cameras, televisions, monitors use the RGB color space where the letters R, G, B stand for Red, Green, Blue color channels. By using separate red, green, and blue channels as light sources, color display systems on other colors represent by mixing a weighted combination of these three components. Figure 3.2 depicts the 3D RGB color space where black and white points are also marked.

While commonly used, the RGB color space has some well-known shortcomings (e.g. sensitivity to illumination changes); in fact, different color spaces may be suitable in different applications. Alternative color spaces include the normalized RGB (nRGB) and the HSI color spaces. Both color spaces are often used in order to obtain robustness against illumination differences. Because of this property, both color models are appropriate for CBIR studies and are often preferred to the RGB model. The nRGB color model is a derivation of the RGB model in which each channel value is normalized with the total intensity of all channels. The normalization process discards different illumination conditions. In the HSI (Hue Saturation Intensity) color model, which is also called as HSL (Hue Saturation Luminance) or HLS, luminance of color is represented separately from the chromaticity. Another color space that is similar to HSI is YIQ which is originally designed for TV

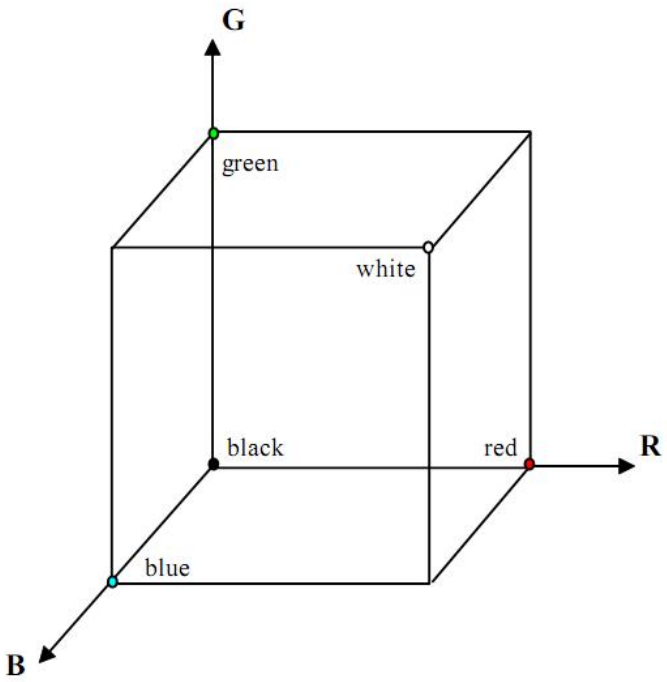

Figure 3.2: 3D RGB cube: illustrating RGB color space. Any color can be represented as a point in the color cube by (R, G, B). For example, red is (255, 0, 0), green is (0,255,0), and blue is (0, 0, 255).

broadcasting, but also used in CBIR [28]. Due to the separation of the intensity component(Y), the YIQ color space is also robust to illumination variations.

3.1.2 nRGB Color Model

The nRGB color model is a derivation of RGB model in which each channel value is normalized with the total value of three channels (R,G,B). The normalization process effectively normalizes for different illumination conditions. The colors are represented by three normalized color values (nR, nG, nB), which indicate the red, green, and blue color ratio in a specific pixel. The normalization computation for red and green channels are formulated as: $nR = R/(R+G+B)$ and $nG = R/(R+G+B)$. The efficiency of this color space is due to its robustness to the illumination changes. Humans are robust to these changes, perceiving for instance a red object similarly in

difficult illumination conditions. For the color spaces, the image of the same object or location taken in different illumination conditions correspond to widely different RGB values while they show similar normalized channel values. For example; both RGB(150,150,150) and RGB(75,75,75) colors are gray with different brightness levels. On the other hand colors are represented as nRGB(0.33,0.33,0.33) which indicate their equality as the same color. Note on the other hand that converting an RGB image into nRGB removes the effect of any intensity variations, thus it is preferred in CBIR problems.

3.1.3 HSI Color Space

HSI stands for Hue, Saturation and Intensity, which is also called HLS and HSL (for Hue, Lumination & Saturation). The motivation of the HSI color-system is to imitate the human perception better than the RGB model. Similar to the RGB and nRGB color models, color is represented by three channels in the HSI color space as well. However, in the HSI color-system, colors are not combinations of three colors, but the juncture of three different visual factors such as color, density, and intensity. Namely, in the HSI color space, color is represented using its Hue, Saturation and Intensity values. The important novelty is that brightness factor of light is considered apart from the color itself. For instance, while dark blue and light blue colors have different R, G, and B values in RGB space, both have the same hue values in HSI space. The saturation value depicts the density of the color. Therefore, having same hue and intensity values, the different tones of blue can be represented by changing the saturation. This condition indicates the closeness of HSI space to human perception. We humans perceive and also name the shades of green as light green, pale green, green, dark green i.e., as the same green color with different amounts of saturation. As mentioned above, by separating intensity, the false effect of different light sources and angles is discarded.

3.1.4 Color Feature Extraction

Color is an important feature in all CBIR applications and the same applies for plant image retrieval problem. However, the use of color in plant retrieval is in a way more complicated compared to other CBIR applications, since most plants have green as their main color with subtle differences. Furthermore, flowering plants should be successfully matched despite differences in flower colors. For instance given an orchid of a certain color, ideally one should find its exact match from the database, but also other orchid plants with different flower colors.

As in many other studies [28–31], we used color histograms and color co-occurrence matrices to assess the similarity between two images. If the occurrences of colors or color pairs in two images are close, the images will be matched as similar in terms of their color distributions. Three different color spaces are used to produce color histograms; namely RGB, normalized RGB (nRGB), and HSI. In order to obtain a histogram robust to normal variations in plant images, the 24-bit RGB information is quantized into a 9-bit representation (for a total of 512 bits, using 3 bits for each color channel), before calculating the RGB color histogram. For the nRGB representation, one of the channels can be deduced from the normalized value of the other two (nR+nG+nB=1); therefore we compute the nRGB color histogram using only the values of two normalized channels, which affords more bins (for a total of 256 bins, using 4-bit for each of the nR and nG values). In the HSI space, the 360 different hue values which indicate the color are quantized to 10, 30 or 90 bins. Intensity value is intentionally discarded, while saturation is not used for simplicity. Prior to histogram matching, we smooth the computed histograms by taking weighted averages of the consecutive bin values, so as to obtain some robustness against quantization problems.

As an extension of the color histogram, a color co-occurrence matrix gives information about the color distribution in neighboring image pixels. Although color co-occurrence is generally mentioned as a texture analysis method, it primarily indi-

cates the distribution and sequentiality of color pairs. We use a 30x30 co-occurrence matrix computed from the HSI color space, where $C[i][j]$ stores the number of neighboring image pixels having the hue values i and j. We generate the co-occurrence matrix using three different methods: i) considering only four neighboring pixels (i.e. top, bottom, right, and left neighbors); ii) considering all eight neighboring pixels and iii) using 8-neighbors but ignoring the diagonal elements of the co-occurrence matrix. Diagonal elements store the number of neighboring pixels having the same quantized color and dominate the matching process since they correspond to large uniform color regions in the image. This last method aims to capture color change information, rather then uniform areas.

3.1.5 Challenges

The primary challenge we have encountered in color analysis is caused indirectly by the insufficient segmentation results. When background region is not cleaned up smoothly, these regions effect and bias the generated color histograms. Additionally in hue histograms, we have encountered the undefined saturation and meaningless hue values. Meaningless hue values are obtained in two cases; i) singularity problem causes zero saturation and undefined hue, ii) very dark and very bright points have saturation values of 0 and 1, respectively, while their hue values vary widely. To avoid undefined hue and saturation values, the system may be enhanced with additional controls on singularity points, as well as very dark or bright points. For instance RGB, or intensity values may be used as a color feature in such cases as proposed in [32]. In fact, we implemented a modification to ignore pixels with undefined or problematic values, but this attempt was not very successful, partly due to ignoring the white areas inside the plants. Another study has evaluated the success of different color spaces and transformations on skin detection [33] with similar results, concluding that removing illumination information may reduce performance, a finding in line with our experience.

3.2 Texture Analysis Techniques

Texture is another low-level property of images. The structure of the image, actually structure or surface [34] of the object/region in the image can be expressed as image's texture. Texture patterns on the image are important as a characteristic of the object or region and that is the property aimed to be extracted. As in real life, real objects have different visual patterns; for instance grass and stone appearances are distinct. Other typical texture examples are shown in Figure 3.3. Although these examples are ideal and specifically aim to express the various texture types, every object has some texture information even though it has plain surface. Similarly, objects may display different texture characteristics in different areas.

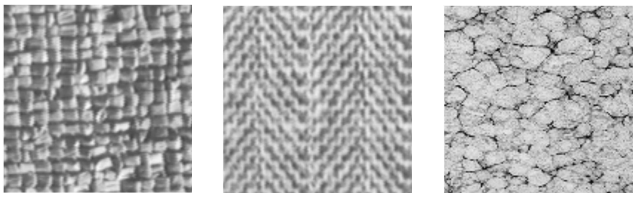

Figure 3.3: Various texture samples taken from the Brodatz collection

Texture of a plant leaf is one of the most important distinguishing feature in plant images. It may be due to having many veins in different directions or parallel lines of different colors. In addition to the single leaf texture, a global texture information is extracted in this thesis, since the whole picture of plants are used rather than only a single leaf. This includes the frequency of leaves, their orientation and curvature.

In the general CBIR research, various approaches to retrieve texture features are used both in spatial and frequency domain. The simultaneous auto-regressive (SAR) model, gray-level co-occurrence (GLC) matrices, Markov random field (MRF), pyramid-structured wavelet transform (PWT), tree-structured wavelet transform (TWT) and Wold decomposition are some examples for spatial-domain methods [35–37].

In addition, one of the most preferred methods for texture analysis is the Gabor wavelets [35, 38, 39]. The following section explains Gabor wavelets in detail.

3.2.1 Gabor Wavelets

Gabor wavelets in different scales and orientations are suitable for texture analysis, since texture depends on scale.

It is easy to understand Gabor wavelets starting from 1D Gabor filters. Gabor filter is generated by convolving a Gaussian curve and a sinusoid function. Figure 3.4 nicely illustrates this concept. Gabor filter consists of two parts, real and imaginary as depicted in Figure 3.4c and d respectively. While the real part indicates the Gabor filter generated by *cosine* function, the imaginary part carries the Gabor filter by *sine* function. This double design can be understood in Equation 3.3.

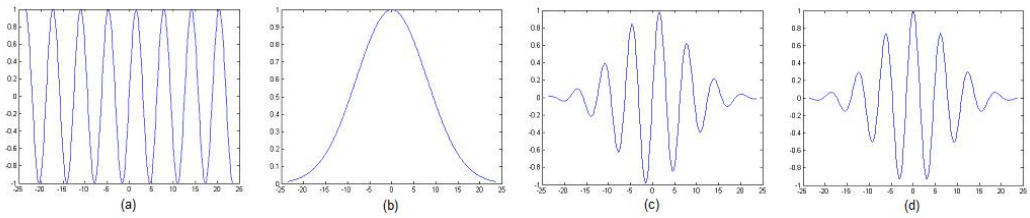

Figure 3.4: 1D composition of a Gabor filter a) A sinusoid b) A Gaussian c) Resulting Gabor filter(real part) d) Resulting Gabor filter(imaginary part)

Since images are 2-dimensional, 2D Gabor filters are used in image recognition and retrieval. A 2D filter window is slided on the image to measure the local responses. High-response means that the texture of that region is aligned with that filter. A 2D Gabor filter is not very different than in 1D: the Gaussian is a 2D Gaussian kernel as shown in Figure 3.5a and sinusoid function is a sinusoidal curve repeated in the second dimension (see Figure 3.5b). The composition of a 2D Gabor

filter is depicted in Figure 3.5c. While sinusoid function helps to retrieve the texture pattern in the image, Gaussian kernel smooths the filter to adjust the effect of points according to their distance to the center (i.e. points that are corresponding to the outer regions of the filter will have less effect on the total response).

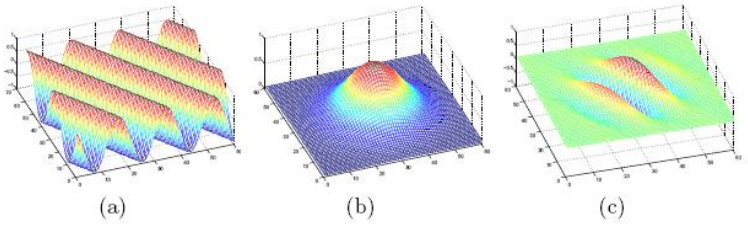

Figure 3.5: 2D composition of a Gabor filter (taken from [3]) a) A sinusoid b) A Gaussian c) Resulting 2D Gabor filter(wavelet)

The response of a Gabor filter on an image $I(x, y)$ is the convolution of the image $(I(x, y))$ and the Gabor filter. The convolution is expressed below:

$$R_{mn}(x, y) = \sum_s \sum_t I(x - s, y - t) g_{mn}(s, t)$$

where $g(s, t)$ denotes the Gabor function, s and t are variables that are corresponding to Gabor filter window's size and m, n are scale and orientation variables of the Gabor wavelet function.

The mathematical basis of Gabor functions is based on wavelets. 2D Gabor functions are expressed with the following equation in several papers [35–38] -with a few different notations only- as:

$$g(x, y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{1}{2}(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2})} e^{2\pi i W x} \tag{3.1}$$

The first term; $\frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{1}{2}(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2})}$ is the 2D Gaussian function with different stan-

dard deviations in both dimensions and the rest of the Gabor function is the complex sinusoid. The complex sinusoidal function can be seen if the Gabor function is transformed using Euler's formula which is:

$$e^{2\pi i\theta} = \cos 2\pi\theta + i \sin 2\pi\theta \tag{3.2}$$

then the Gabor function can be written as:

$$g(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{1}{2}(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2})}(\cos 2\pi W x + i \sin 2\pi W x) \tag{3.3}$$

Here, W denotes the window size of the filter [38], while the standard deviations of Gaussian kernel in x and y dimension are expressed by σ_x and σ_y respectively. The Gabor function given above is defined as mother wavelet function (i.e. in [36–38]), from which Gabor wavelets with various parameters are generated. A Gabor wavelet is generated by:

$$g_{mn}(x,y) = a^{-m} g(x', y'), \;\; a > 1 \tag{3.4}$$

where

$$x' = a^{-m}(x\cos\theta + y\sin\theta)$$

$$y' = a^{-m}(-x\sin\theta + y\cos\theta)$$

$$a^{-m} = \left(\frac{U_l}{U_h}\right)^{\frac{-m}{S-1}}$$

The parameters, m and n specify the dilation (scale) and orientation of the wavelet. The angle, θ, is defined by the parameter n which has values between 0 to $K-1$ where K is the total number of orientations used. In other words, $\theta_n = n\pi/K$ and $n = 0...K-1$. Likewise, a^{-m} determines the scale of the wavelet in which U_l and U_h denote the minimum and maximum filter sizes respectively and $m = 0...S-1$

where S is the number of scales.

3.2.2 Texture Feature Extraction

In this thesis, Gabor filters in different orientations are used to detect textures in different directions, while the use of different scales aims to detect textures in different scales. The Gabor function we used is given below:

$$g(x, y, f, u, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x'^2+y'^2}{\sigma^2}} (\cos(\frac{2\pi x'}{\lambda}) + i \sin(\frac{2\pi x'}{\lambda})) \tag{3.5}$$

where

$$x' = x\cos(\theta) + y\sin(\theta)$$

$$y' = -x\sin(\theta) + y\cos(\theta)$$

Here x and y indicate the coordinate on the Gabor wavelet. Equation 3.5 is a special case of Equation 3.3 with $\sigma_x = \sigma_y$. Hence, we denoted the standard deviation of Gaussian kernel with only one variable σ. Besides, we have taken a^{-m} as 1 and we did not specify the dilation of wavelet with W. Instead we used $1/\lambda = f/C$ which expresses the spatial frequency of the sinusoid proportional to the filter window size in x-dimension: C. It should be emphasized that σ is related with dilation of the sinusoid and changes with the frequency, f (number of sinusoid peaks on the filter exactly). The last parameter, u, indicates the chosen orientation number that finds the angle θ by $\theta = u\pi/4$. We multiply u with $\pi/4$, since unit orientation difference is taken as $\pi/4$. u can take values between 0 and $K-1$ where K is the number of orientations.

In this thesis, Gabor filters in four different orientations and scales are used (K=4, S=4). With 4 scales (k_{1-4}) and 4 orientations (θ_{1-4}), a total of 16 Gabor wavelets are applied to each image, resulting in 16 different Gabor response images.

Figure 3.6 offers response images of a sample plant image in K=1 (f=3 in a 40x40 filter), S=0...3, where texture patterns and their orientations are evident. We use the mean (μ_i) and standard deviation (σ_i) of these maps in comparing the texture differences between two images.

When comparing the texture similarity of two images, often the comparison is done using the Gabor responses in all scales. This is called the *default* texture feature. An alternative is to use the most dominant scale for each image. This is called the *max-scale* texture feature and is meant to deal with scale difference across images of the same plant. We introduced a third approach which is called *patch-based*, to provide rotational invariance on a leaf level, as explained in Section 3.2.3. In addition, two other methods are proposed which produce new response maps using the Gabor response images. *Maxima over scales* method selects the dominant response of a pixel among all scales and generates a new response map by performing this selection for all pixels. The final method (*sum of orientations*), provides a new image as well. However, the new pixel values are computed by summing the response values of corresponding pixels in all orientations, to provide rotation invariance.

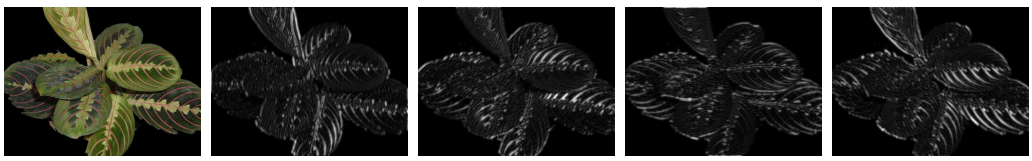

Figure 3.6: The original image and four different maps show the texture energy in different orientations (0, 45, 90, 135 from vertical, from left to right). Note that the texture in different leaves of this plant are captured in different orientations.

Rotation Invariance

When a uniformly textured object (e.g. straw or fabric) is rotated, its Gabor response within the same scale but in different orientations are circularly shifted. For instance, when an object with a dominant texture along the x-axis (0 degree) rotates at 45 degrees, the response of the rotated image is dominant on the 45-degree Gabor response. Hence, if we represent the feature vector starting with the angle having the maximum response (a canonical representation) and in increasing angular order, we can match the corresponding maps. In the given example, the initial texture feature vector:

$$\{(\mu_0, \sigma_0), (\mu_{45}, \sigma_{45}), (\mu_{90}, \sigma_{90}), (\mu_{135}, \sigma_{135})\}$$

would be matched to the circularly shifted feature vector of the rotated image by 45 degrees:

$$\{(\mu_{45}, \sigma_{45}), (\mu_{90}, \sigma_{90}), (\mu_{135}, \sigma_{135}), (\mu_0, \sigma_0)\}$$

3.2.3 Patch-Based Approach

The situation is more complex in plant images than in general CBIR problems. Even if the texture in a plant little varies across the leaves of the plant, the fact that the leaves are often oriented in different directions makes the above method inapplicable (see Fig. 3.6 for an example). For this problem, the ultimate solution is to go down to the leaf level and compare the texture responses of individual leaves. This thesis attempts to approximate this approach by using a patch-based approach where we obtain uniformly distributed patches on the image and rotate each of them to a canonical orientation (in angular order, starting with the most dominant response) by rotating each patch individually. While this attempt is not

guaranteed to provide full (leaf level) rotation invariance, the experimental results show that it does help with the texture analysis. We have implemented a patch-based method considering Gabor response images in one scale only. Hence, there are 4×2 patch-based texture features with $K = 4$ and $S = 1$. Feature extraction for the patch-based method is performed as follows. First, the plant region bounding box is detected since it is an efficient size measure than the image size. Then, corresponding Gabor response image is divided into 20×20 distinct patches. Mean intensity and standard deviation values (μ, σ) are calculated for all patches. At the end, the mean and standard deviation of these 400 patches is computed. Hence, for each response image one μ and one σ values are produced. With four different orientations, we reach 8 feature values.

3.2.4 Challenges

The main challenge we have encountered is image resolution which causes texture to appear in different Gabor responses. Although we have produced Gabor response images in $S = 4$ different scales, since they are compared one-by-one on the same scale, the amount of plant detail that belongs to the sub-image on the same size with Gabor filter window. Figure 3.7 depicts this effect on a cymbidium orchid image which was originally in 1280×1024 pixels size. The comparison is done with a subsample of the original (larger) image in size 600×480. While Gabor response map of the original image has more details, the response of the smaller one lacks details since some texture patterns are missing. Hence, comparing images by disregarding their image resolutions is not very sufficient. A simpler solution may be to use response images in scales that are adaptive to image size.

Although image size seems a problematic issue, our proposed texture analysis methods are two alternative solutions. Max-scale and patch-based methods overcome both orientation and scale variance problems. The max-scale approach assumes that the highest response of gabor filter is obtained from the most fitting

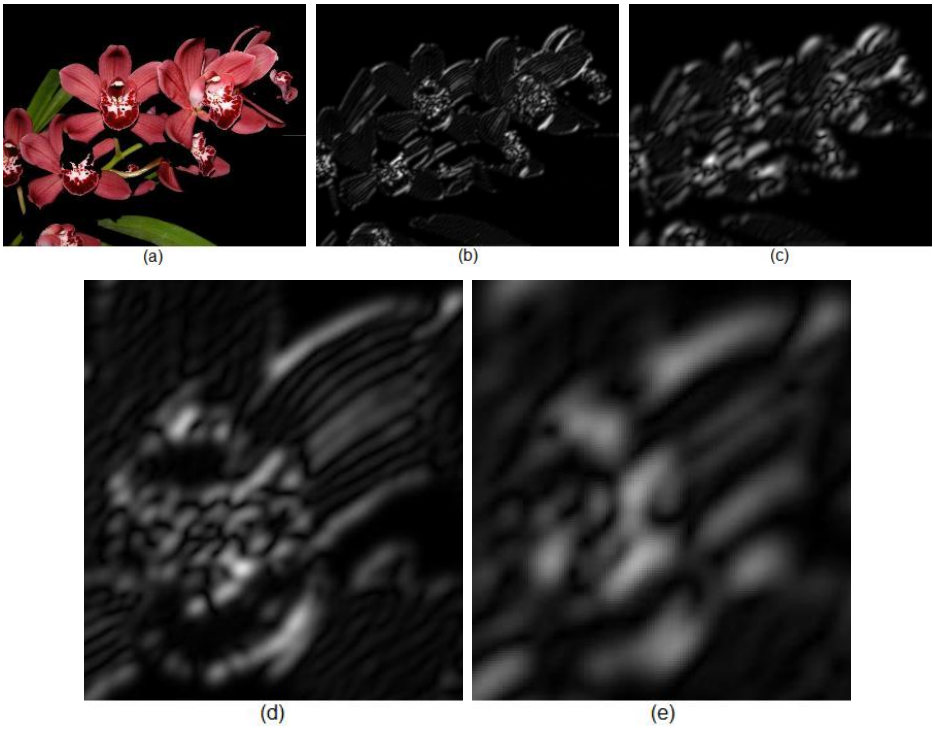

Figure 3.7: The effect of image resolution to the Gabor response images and retrieved texture patterns. a) Plant image b) Gabor response image of the image in size 1280x1024 c) Gabor response image of the image in size 600x480 d) Detailed view of b focused on some pattern e) Detailed view of c focused on the same pattern with d.

filter scale, because it retrieves most of the patterns. In the max-scale texture analysis technique, we have used texture feature values (σ and μ) of the scale that has given the highest-response (as the only texture feature). In patch-based approach, images are partitioned into fix number of patches. This approach overcomes the image resolution variety problem stated above. Although the same plant is shot with a 1280×800 and a 600×375 pixels resolution, since both images will be separated into an equal number of patches, the patches have identical patterns in both images. Hence, as expected, the texture similarity of both images are very high in patch-based method. Moreover, this method overcomes the rotation variance problem of leaves as well, by rotation all patches to a canonical orientation.

The common assumption of photographs taken for the purpose of identifying an object is that such images are to clearly indicate the general structure and/or outline of the object. Therefore, we can expect the input plant images show the plants from a distance where all parts of the plants are seen. Most of our plant image database contains this kind of photos, but there are few close-ups also. If a constraint is defined to regulate the plant position in the image, scale invariance problems will only depend on image size and can be easily handled.

3.3 Shape Analysis Techniques

Shape information is probably the best distinguishing characteristic of a plant, hence shape features are used in image retrieval systems frequently. In shape-based CBIR, two basic approaches exist: region-based and boundary-based (contour-based) [40]. Region-based systems typically use moment descriptors [41] that include geometrical moments, Zernike moments and Legendre moments [40]. Boundary-based systems use the contour of the objects and usually give better results on images that are distinguishable according to their shape outlines. Fourier descriptors [40,41], curvature scale space [18,42] are some commonly used contour-based methods for shape feature extraction.

3.3.1 Contour-Based Shape Analysis

Contour-based shape analysis techniques are built on the fact that images express more descriptive shape information on their outer boundary compared to their internal content. In this aspect, when dealing with such images, the boundary of the object or region is the most important shape descriptor for this type of shape-based image analysis. There are several steps to analyze the shape of a region using its contour. The initial step is extraction and quantification of the contour. The second step is evaluating this contour to extract the feature values describing the contour. Different contour-based shape analysis methods mostly vary in the second step, while they commonly use chain code descriptors for the contours.

A common representation of the image contour is the *chain code*. Chain code is the representation of the contour by a series of enumerated direction codes which are in the interval of [1-8] [1] and depicted in Figure 3.8.

In such a contour system, if you know the starting point position and the chain

[1]The most common case of chain codes has 8 directions. The other alternative is using 4 directions. Pixels have a neighbor at each eight directions which means there are eight possible directions to move from a pixel.

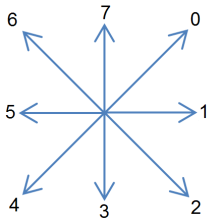

Figure 3.8: 8-directions that are used in the chain code and their corresponding enumerators

code, you can follow the changes in the direction and draw the contour of the object. This method also makes it possible to measure direction changes on the object boundary very easily that we can observe smoothness and roughness of the object shape. Therefore, we have preferred to use a contour-based shape analysis method.

3.3.2 Contour Tracing

For plant image retrieval, the outline of a plant image is considered as an appropriate shape descriptor, since plant leaves are recognizable on outer regions of the plant (see Figure 3.9).

In this thesis, the shape feature of the plant images are extracted with a contour-based approach. In order to retrieve shape information of the plants, plant boundaries need to be recognized and quantized as image contours. How we have traced the plant boundaries is explained in this section. The pseudo-code of the algorithm is given in Appendix B.1.

Most of the contour tracing algorithms are based on following the boundary of the objects, the way bugs find their ways around objects. In this analogy, when the bug follows the contour from the outside, it tries to advance and keep the object on its left, turning clockwise when necessary. There is also an alternative where the contour is followed from the inside. Since, images are segmented in our system, they

Figure 3.9: An example for contour tracing. Left: Original segmented image where background pixel value of 0 while foreground is non-zero, Center: traced contour of the image, Right: Detailed view of the contour focused on the marked region where concave and convex points are marked as red and blue.

have background value of 0. Hence, the background is easily detected when a black pixel (intensity = 0) is found.

For contour tracing, we have used the first method (tracing from outside) which is expected to more robustly draw the plant contour. Although using segmented plant images would be also sufficient as input, we have used their segmentation maps, thinking that using the PNG image would be more efficient [2].

To find the first edge point, we have proposed a different procedure rather than starting from the (0,0) point and continuing to find a plant-region point. In our algorithm, the starting points are specified in the beginning by approaching the image from four different directions: east, north, west, and south.

After finding an initial point, the plant is started to be encircled by approxi-

[2]We have stored the segmentation maps in the PNG format, while segmented images are in the JPG format. Since JPG is a compressed image format, usually it causes slight changes in pixel intensities which effect our plant region outline, which is why PNG is preferred for segmented maps.

mating that it has a roughly circular shape. For example, if the bug starts to trace from the first point on the left, you know that the bug has to move right from either above or below pixels. Since directions are altered counter-clockwise in our approach and the first direction is east (1 in Figure 3.8), then the next direction becomes northeast (0). If the point on the northeast is an edge point, then the bug starts to round the boundary from the top. In the next iteration, the bug attempts to turn left again trying to move the above point (in direction 7). However if that point is not an edge (plant) point, then the bug is forced to turn right. Since the bug will turn right when it sees non-plant regions, it follows the path attached to the object boundary and also moves on the possible left points. At the end, the bug completes the path when returning to the starting position. Additionally, special conditions exist such as attempting to move to a point out of the image; we change the direction of the bug in such cases. All these controls and direction changes can be seen in the above contour tracing algorithm given in Appendix B.1.

3.3.3 Interest Point Detection from Contours

The interest points in contour-based shape analysis are sharp points of the contour. Since the contour is stored/quantized as chain code, then sharp points can be easily detected by measuring direction changes in the chain coded-contour. In our problem, sharp points are expected to be the tip and base points of leaves or small leaf structures (juts) which can be detected by directional changes (see Figure 3.9,3.13 for an example).

In our system, the interest points are specified during the main contour tracing operation. While tracing a new contour point, the direction change around that point is numerically measured. To decrease the effect of noise on the contour, as a smoothing factor, we have considered direction difference between two n-pixel contour segments rather than only two pixels. The variable n is a parameter called as trace *run steps* and had two values: 5 and 10 in our experiments. The sharpness

is measured as the difference of the average directions of these two contour segments. Then if this measured direction change (sharpness) is above a predefined limit, the corresponding breakpoint is labelled as a sharp point. Algorithm that is used to detect sharp points depending on their sharpness is provided in Appendix B.2.

The sharpness limit is defined as acute angles. Since the 8-direction chain code is used (see direction enumerators in Figure 3.8), 90 degrees numerically corresponds to a direction difference (sharpness measure) of 2.

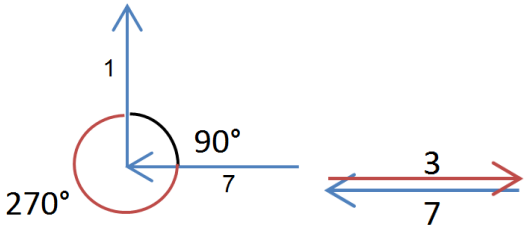

Figure 3.10: Illustration of direction change in the contour. Left: Example for wide-acute angle comparison. Direction change from 1 to 7 gives sharpness measure 2 (abs(1-7)=6≡2). Right: The highest sharpness is obtained when an opposite direction is followed another (abs(3-7)=4)

Consider direction change from 7 to 1 illustrated in Figure 3.10. Here the difference is +6 which corresponds to 270 degrees of difference, but the real angular difference is 90 degrees. In order to solve this problem and use the acute angle defined by two directions, we use:

```
diff = abs(dir_1-dir_2);
if (diff>4)
diff=8-diff;
```

At the end of the contour tracing process, a list of interest points is produced. The next step is labelling these sharp points as convex or concave. Figure 3.11 is explains the heuristic under this labelling task. Concave points are placed in the

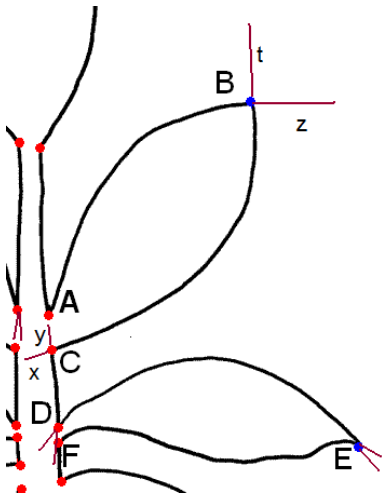

Figure 3.11: Illustration of the heuristic that labels the sharp points as concave or convex

inner parts of the shape and if contour approaching on that point is elongated on this direction, this extended partition will continue on the inner part of the shape (see point C in Figure 3.11). The reverse is true for the convex points. If the contour approaching to B is continued, the lines like z or t will be drawn depending on the tracing direction you are following.

In our system, an imaginary line is drawn on the same direction of the sharp point and the first non-contour point of the line is checked if it belongs to the inner or outer part of the region. Since the segmentation maps of the images are used as input images for shape analysis, it is easy to decide if the points are in or out of the plant region. Please refer to Appendix B.3 for the pseudo-code of the convex/concave point differentiation algorithm. At the end of detecting convex and concave points, the feature values are measured. That step is explained in Section 3.3.4.

3.3.4 Extracted Features/Shape Descriptors

For plant image retrieval, the outline of a plant image is considered as appropriate shape descriptor, since plant leaves are recognizable on the outer regions of plant.

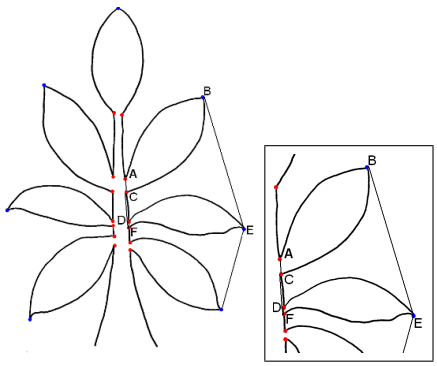

Figure 3.12: Illustration of the sharp-based feature measures on a plant contour

Therefore, features extracted from the overall plant contour are used in this thesis. The contour of the plant is extracted by tracing the segmented plant image and represented as a chain code in 8 directions (i.e. one enumerator for each 45 degrees). Then, six features are extracted from this contour:

1. Number of Concave Points (e.g.A,C,D,F)
2. Number of Convex Points (e.g.B,E)
3. Leaf Arc Length ($\widehat{ABC}$)
4. Normalized Leaf Arc Length ($n\widehat{ABC}$)
5. Leaf Base Distance ($\overline{AC}$)
6. Leaf Tip Distance ($\overline{BE}$)

In order to extract these features, we first extract sharp (edge) points by analyzing direction changes on the contour. These points are labeled as convex or concave depending on their position and direction (or curvature of the contour). All these measures are depicted in Figure 3.12. Given concave and convex points, the corresponding leaf arc length is measured as the arc distance between two consecutive concave points ($\widehat{ABC}$). The leaf base distance ($\overline{AC}$) is measured as the straight

line distance between them. Similarly, leaf tip distance is the distance between two consecutive convex points ($\overline{BE}$).

For robustness, we obtain the median value of these features from the whole plant region. Assume that n distinct leaf structures have been found, then n different leaf tip distances are measured. The feature value is computed with the following operation:

$$median(l_0, l_1, ..., l_{n-1})$$

Extracted shape features that are listed above can be grouped to three, according to the information they represent. Number of concave and convex points indicate the sharp point distribution when they are proportioned to the contour length. While leaf tip distance represents leaf distribution, the remaining three features (namely leaf arc length, leaf base distance, and normalized leaf arc length) provide information about the leaf structure.

For this thesis, we designed shape features that describe the overall contour of the plant. Local features such as scale-invariant feature transform (SIFT) features have been successfully used in many recognition and retrieval problems. SIFT method locates key points or interest points in the images and matches the descriptors that are collected at these interest points in order to match two images. Since a plant image consists of a collection of leaves which are not very distinguishing object parts, our initial attempts in using the SIFT features for this problem have not been very successful. In the future, we intend to use the SIFT features to identify interest points that indicate leaf boundaries, by modifying the standard SIFT algorithm to be more suitable for our problem. Since, SIFT is a local descriptor, leaf boundaries in the inner plant regions would also be captured.

3.3.5 Challenges

As also mentioned in early sections, accuracy of image segmentation and smoothness of segmented image part is the most important factor directly affecting the success of shape-based image analysis. Erroneous, jaggy regions in the contour is a very common problem which leads to faulty feature extraction. Here is an example for an inaccurate segmentation and its corresponding noisy contour that shows the plant as if it had very small frequent leaves, while in reality the plant has a few long and smooth leaves (see Figure 3.13).

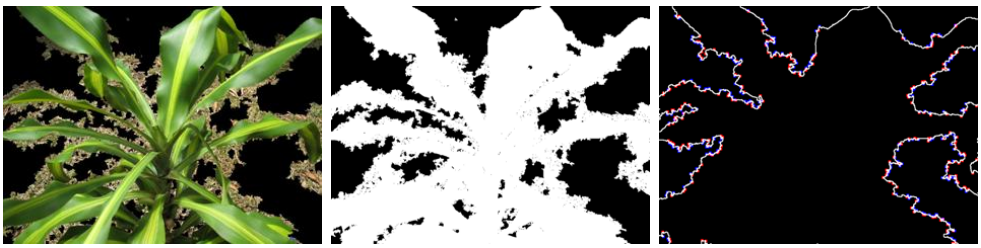

Figure 3.13: Example for a jaggy (noisy) plant contour caused by insufficient segmentation. Left: Segmented image, Middle: Segmentation map to see the segmentation faults, Right: Traced contour of the image with convex and concave points marked as blue and red, respectively

In such a problematic case, since our system considers the direction changes on the contour, many points are labelled as sharp points. However, most are fake and would not be labelled as sharp if segmentation had been more robust. In conclusion, the fake sharp points are also used in evaluation of shape features such as normalized leaf arc length. While leaf base distance and leaf tip distance measures decrease, number of concave and convex points increase drastically. To apply a smoothing filter or using longer run steps may be a solution. However, applying a filter will also bring some extra image processing such as edge detection, line following. Tracing the contour with larger run steps may decrease the noise effect. However detecting if an image needs this adjustment is another issue.

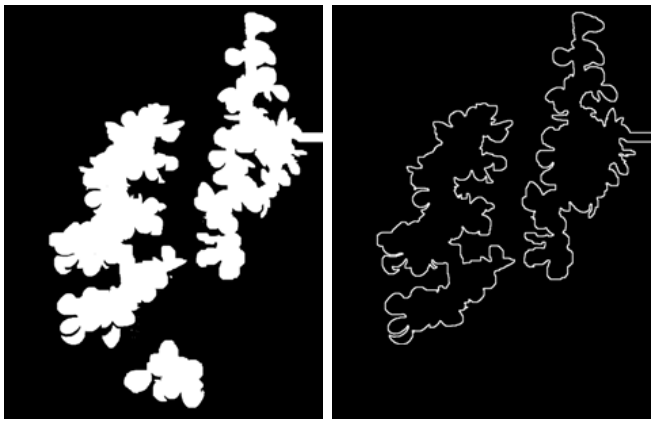

Figure 3.14: Example of a segmented plant image with three separate plant regions and its corresponding plant contour. By a small modification on contour tracing algorithm, two regions are retrieved and added to the contour.

Although our contour tracing algorithm is generally satisfactory, some conditions can cause contour tracing to be hard or incomplete. If a segmented map consists of several plant region segments as in Figure 3.14, since the algorithm designed to trace only one closed area, it is not possible to catch all these partitions. To overcome this problem, we have designed the system which starts tracing from four different directions. So, it is possible to retrieve four different contour segments. However, if these contour segments are not meaningful or long enough (if they are smaller plant parts spread to the outer parts of the image, while the main part is in the middle), the actual shape information will not be extracted. Note that in our current system contour parts shorter than $4 \times run\ steps$ are discarded and not added to the resulting contour.

Chapter 4

MATCHING CRITERIA

At the end of the feature extraction stage, there are several features extracted from the query images such as color, texture and shape features. The corresponding features are also precomputed for the images in the database. Identification techniques based on similarity matching depend on distance between feature vectors of images. Therefore, the success of identification systems strongly depends on how the features are accurately extracted and how the feature vectors are compared. Feature extraction step was explained in Chapter 2 and this chapter explains the matching step using the extracted features.

Given a query image, the database images are matched to the query image according to their similarity. After the similarity of all images in the database are measured, a similarity ranking is generated with respect to these values. As a result, the image at the top of the ranking list is the top-match among the database. The similarity and dissimilarity of two images are complementary scores: if they are 60% similar, this also means they are 40% dissimilar. Since measuring the dissimilarity of images is easier, to understand that is the common procedure followed in recognition/identification problems. The simplest method is to measure the Manhattan distance (also called as L_1-distance) between feature vectors. The

Manhattan distance of two vectors: p and q is computed as:

$$||p-q||_1 = \sum_{i=1}^{n} |p_i - q_i|\,,\ p = (p_1, p_2, ..., p_n), \quad q = (q_1, q_2, ..., q_n)$$

In our system, given a query image, the dissimilarity between the query image and each image in the database is assessed according to the used feature(s). Histograms and co-occurrence matrices in different color spaces (RGB, nRGB and HSI), Gabor texture features and shape features such as frequency of sharp points, structure of leaves (leaf boundary length, leaf width, leaf length/width ratio) and distinct leaf distance comprise the individual features, while some combinations of these features are also tested.

Dissimilarity scores using different individual features are given in the following equations. The RGB color dissimilarity score of two images Q (query) and I (database image) is calculated using the L_1 norm of the difference between the histograms of the two images, according to the formula:

$$\delta_{RGB}(Q, I) = \sum_{i=1}^{512} |h_Q(i) - h_I(i)|$$

Here each histograms has 512 bins and $h_Q(i)$ is the value of i^{th} bin of Q's histogram.

The other color features, namely nRGB and hue histograms and the HSI based color co-occurrence matrix are matched similarly. In those cases, the summation is done over 256, 30/90 and 900 (30x30) elements, respectively.

Prior to the hue histogram matching, as a different calculation method, we smooth the computed histograms by taking weighted averages of the consecutive bin values, so as to obtain some robustness against quantization problems. Smoothed 90-bin hue histogram matching is computed by the following formula:

$$\delta_{HueSmooth}(Q,I) = \frac{1}{4}\sum_{i=1}^{90}\left|h'_Q(i) - h'_I(i)\right|$$

where $\frac{1}{4}$ is a normalization factor and $h_Q(i)$ is the smoothed value of i^{th} bin of Q's histogram:

$$h'_Q(i) = h_Q(i-1) + 2 \times h_Q(i) + h_Q(i+1)$$

For the modified color co-occurrence matrix, the distance of non-diagonal elements are taken into account as:

$$\delta_{Co-oc}(Q,I) = \sum_{i=1}^{30}\sum_{j=1}^{30}\left|cc_Q(i) - cc_I(i)\right|, i \neq j$$

where the condition $i \neq j$ eliminates the diagonal values.

The (default) texture dissimilarity of two images is calculated by summing the dissimilarity of each of the $s \times u$ Gabor response features as displayed in the following two equations. Here s denotes the scale, u denotes the orientation and $\delta_{su}(Q,I)$ denotes the dissimilarity in the given scale and orientation (s,u):

$$\delta_{Gabor}(Q,I) = \sum_{s}\sum_{u}\delta_{su}(Q,I)\ ,\ s = 1\ldots4\ ,\ u = 1\ldots4 \tag{4.1}$$

where

$$\delta_{su}(Q,I) = \sqrt{(\mu_{su}(Q) - \mu_{su}(I))^2 + (\sigma_{su}(Q) - \sigma_{su}(I))^2}.$$

We also compare texture features of two images using two other approaches. One is called the Max Scale method, where the texture of two images are matched in the dominant scale, as:

$$\delta_{Gabor}(Q,I) = \max_{s}\sum_{u}\delta_{su}(Q,I)\ ,\ s = 1\ldots4\ ,\ u = 1\ldots4$$

The dissimilarity of the patch-based texture method is the same as the default, while the computation of the Gabor response images differ. As stated above, there are $s \times u$ default texture features for each image in the default method. However, the patch-based method uses the Gabor response images in one scale only (since the fixed number of patches resolve the resolution and scale problems to some extent), hence there are u patch-based texture features and Equation 4.1 is computed with $s = 1$.

Besides color and texture-based dissimilarity scores, shape-based dissimilarity is measured as L_1-distance of shape feature vectors of two images, Q and I, as:

$$\delta_{Shape}(Q, I) = \frac{1}{n} |F_Q - F_I| = \frac{1}{n} \sum_{i=1}^{n} w_i \left| f_{i_Q} - f_{i_I} \right| \ , \ n : num.of features \ in \ F$$

where F_Q can be expanded as: $F_Q = f_{1_Q}, \ldots, f_{n_Q}$. Although shape-based dissimilarity is treated as one score while combining with color or texture methods, we have also made tests by separating it to three shape feature types such as frequency of sharp points, structure of leaves and leaf distribution.

Sharp point counts is calculated as L_1-distance of number of concave and convex points:

$$\delta_{FreqSharp}(Q, I) = |conc_Q - conc_I| + |conv_Q - conv_I|$$

The dissimilarity of leaf structures is computed according to three different feature values; leaf base distance (d_{base}), leaf arc length (d_{ArcLen}) and normalized leaf arc length ($d_{nArcLen} = \frac{d_{ArcLen}}{d_{base}}$)

$$\delta_{Leaf}(Q, I) = \left| d_{base_Q} - d_{base_I} \right| + \left| d_{ArcLen_Q} - d_{ArcLen_I} \right| + \left| d_{nArcLen_Q} - d_{nArcLen_I} \right|$$

Finally the dissimilarity of leaf tip distances (d_{tip_Q}) is computed as:

$$\delta_{LeafDistr}(Q, I) = \left| d_{tip_Q} - d_{tip_I} \right|$$

Contrary to texture-based features, features in different scales are not used together in shape-based dissimilarity scores. Rather, the appropriate scale is chosen according to the image size and those features are used in computation.

In addition to these similarity measures involving single features, we also experimented with combination methods, as done in various other studies [43, 44]. In this case, the overall dissimilarity score is the average of the individual scores.

$$\delta_{Comb}(Q, I) = \frac{1}{n} \sum_{i=0}^{n} \delta_{m_i}(Q, I) \ , \ n : num.of features$$

where n is number of available feature analysis methods, m_i is the i^{th} method and i indicates the index for the i^{th} method. The outcome of various method combinations are presented in Section 6.

Chapter 5

PLANT DATABASE

Currently, we have 380 plant images from 78 different plant types in our database, but the data collection is ongoing, with the aim of extending the variety to a minimum of 100 different house plants as a future work. The number of images for each plant type varies from type to type; while, there are only 3 images of a plant type, another plant has 14 different images. The average number of images per plant type is 4.87. Images are mainly retrieved from the internet, but also collected by taking pictures of available house plants. All the images are segmented to remove the background. The created house plant image database is publicly available [1].

In the database the plant images are named following a standard notation: a number prefix indicating the plant type, the Latin name, an optional sample number and a postfix as "-segmImg" to indicate that the image is segmented. For example; 52-kentia2-segmImg.jpg is a segmented plant image file where 52 denotes the plant type (kentia), 2 is the sample number among kentia plants, and "segmImg" shows that the image is segmented. A list of plant types and number of images present at each plant type is also available in Appendix A.

A general gallery view of our implemented system is depicted in Figure 5.1, displaying original images currently in the database.

[1] See *http://students.sabanciuniv.edu/hkebapci/SUPlants/*

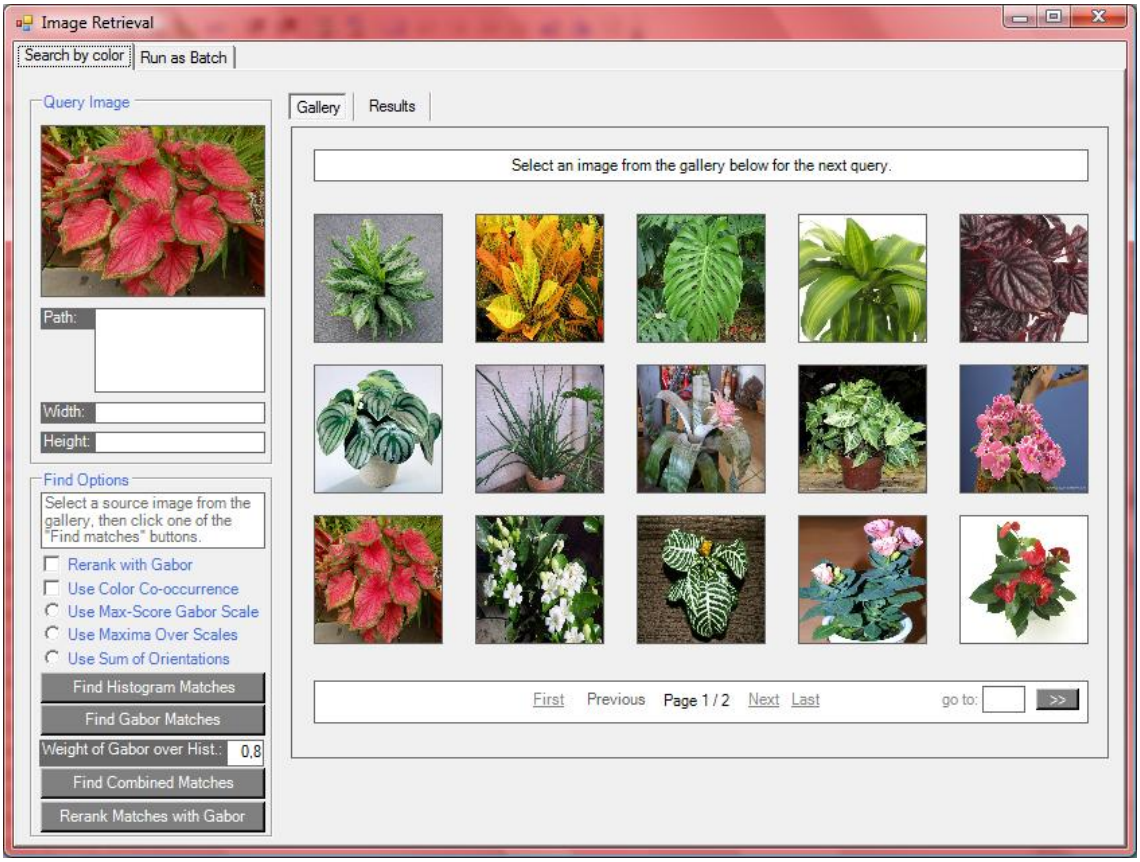

Figure 5.1: Some of the plant images are displayed in the gallery page of the implemented system to show the variety of the plants.

Besides the plant image database collection, we use MySql database table to index the images with precomputed features, which are used in matching the database images with a query image. The precomputed information for each plant image is stored in a table including filename, width, height, rgb histogram with 512 bins, hue histogram with 90 bins, color co-occurrence matrix of 30×30, 32 gabor features (consisting of 16 μ and 16 σ values), and six shape features indicating sharpness of the plant contour, leaf structure, and leaf distribution. In the beginning, each image is indexed and inserted to the database. During this data insertion process, all color and texture features are calculated and stored. At the end of insertion, all features of every image are accessible directly with an sql query. When a query image is

uploaded to the system, the query image will also be indexed and compared with existing images according to their stored feature values.

Chapter 6

EXPERIMENTAL RESULTS

The performance of the system is evaluated by running tests over our plant image database that consists of 380 plant images from 78 distinct plant types. About 1/3 of the database consists of clean images showing a clear plant outer contour and removed background. Each test is applied as *one versus the rest* test (also called as *leave-one-out cross-validation*) and run 380 times querying each one of the 380 plant images for the full database. Unless otherwise indicated, the full database is only used in the tests. The clean database is used in shape related tests which require a clear outer boundary for the plants.

The main metric used in assessing performance is the portion of query images that return a correct image in the top-10 and top-15 returned images. We assume that a user can easily and quickly identify the correct image among 10-15 returned images. In addition to the top-10 and top-15 results, the average minimum rank value indicating the rank order of the best matching image of the correct plant is also presented. All three feature classes (color, texture, and shape) are tested with all possible parameters and retrieval methods that we proposed. In addition to these individual tests, several combined methods are run in tests as well. In conclusion, test results showed that the performance of the system is increased with these combined methods that consider several classes of features. The following

sections present the test results with comments on the performance of the retrieval methods.

6.1 Results Using Color Features

Among distinct color feature analysis approaches, the RGB histogram provides the best top-10 and top-15 rates of 42% and 50%, respectively. It is followed by the color co-occurrence matrix and the nRGB histogram methods with 48% and 46% as top-15 rates. However, as shown in Table 6.1, the accuracy difference among these color analysis methods is not very significant. Even though, due to the illumination factor, the RGB color histogram is expected to be less efficient than the nRGB and hue color histograms, it performed better. The reason might be the lack of illumination variations of the database images or possibly the fact that illumiation difference somehow helped in identification performance. The best top-15 retrieval rate of the Hue histogram methods is 44%, while the remaining Hue histogram methods are around 43%. The reason for the lower performance of the Hue histogram can be the existence of white flower regions in the plant image and quantification of these pixels in unstable hue values. White and black pixels have various, unstable hue values in the HSI color space, since they are lack of chromaticity (see Section 3.1.4).

In addition, little modifications applied on nRGB and color co-occurrence methods improved the performance of these method. The modified color co-occurrence method, which ignores diagonal entries in the co-occurrence matrix as mentioned in Section 3.1.4, outperforms the conventional color co-occurrence method (48% vs 39%). In order to represent each color channel with 2^3 values as in the RGB histogram, rather than 2^4, 256 bins of nRGB histogram is shrinked to 64 bins. The 64-bin nRGB color histogram performed better than 256-bin with top-15 rates of 46% and 43% respectively.

Table 6.1: Color Analysis Results

Method	Top-10%	Top-15%	Avg.Min.Rank
RGB Histogram	**0.42**	**0.50**	**36.9**
nRGB Histogram	0.35	0.43	38.4
nRGB Histogram(64-bin)	**0.37**	**0.46**	**39.4**
Hue Histogram(10-bin)	0.32	0.43	42.9
Hue Histogram(30-bin)	0.34	0.42	42.3
Hue Histogram(90-bin)	0.32	0.43	42.2
Hue Histogram(30-bin smooth)	0.32	0.44	41.2
Hue Histogram(90-bin smooth)	0.33	0.41	42.1
Color co-occurrence	0.31	0.39	48.4
Color co-occurrence(off-diag)	**0.39**	**0.48**	**38.8**

6.2 Results Using Texture Features

We compared the results of i) the default approach where two images are compared in all scales, ii) only the maximum scale (the one with the highest energy) is used in the comparison, iii) patch-based comparison where individual patches are rotated before comparing as in default. These three methods are explained in Section 3.2.2. The *default* and *patch-based* approaches are the best performing texture analysis methods according to their accuracy rates. Both methods achieved 30% at top-10 and 36% at top-15 retrieval accuracy. Although the top-N rates of both methods are the same, *patch-based* approach retrieves the correct plants in higher ranks; at rank 50 rather than 53.2. With a 28% retrieval rate at top-15, *Max scale* comparison method is less successful than the previous two methods.

Table 6.2: Texture Analysis Results

Method	Top-10%	Top-15%	Avg.Min.Rank
Default (All scales & orientations)	**0.27**	**0.36**	**53.2**
Max Scale	0.19	0.28	61.8
Patch-based	**0.27**	**0.36**	**50.0**

Although the performance of the texture-based approaches are not satisfactory by themselves, as shown in Table 6.2, they provide better results when combined with a color-based approach (see Table 6.5). Among the different texture matching methods, it seems that there may be a slight advantage of using *patch based* and *default* methods, over the *max scale*, as indicated in Table 6.5. In addition to three outstanding texture methods presented in Table 6.2, two different texture methods are also tested (see Section 3.2.2 for explanation). *Sum of orientations* method is not sufficient in terms of the retrieval rate it obtains. The other method, *maxima over scales*, performes better than the *max scale* method in individual tests (i.e. 33% in top-15 rate). However, since *maxima over scales* method is not sufficient in combined approaches, we do not present its results in either tests.

6.3 Results Using Shape Features

As explained in Section 3.3.4, six different shape features are extracted characterising sharp point distribution, leaf structure and leaf distribution. Individual Top-10% and Top-15% retrieval rates of these six features are presented in Table 6.3. Number of convex and concave points are used together, since they are closely related. In addition, they turn out to be the best individual shape features with a 28% retrieval rate at top-15. Remaining individual shape features obtained 21% and 22% top-15 accuracy rates. Hence, the number of convex and concave points feature is obviusly more effective than the other shape features. Referring to Section 3.3.4 again, we have stated that leaf structure is represented with features: leaf arc length, leaf base distance, and normalized leaf arc length (i.e. measures indicate the leaf boundary length, leaf width, and leaf boundary length normalized with its width respectively). The combined method joining these three features achieved a 24% probability at top-15 with a slight improvement with respect to the methods using these features individually. Similarly, using the number of concave and convex point features jointly with other shape features improved the accuracy as well with 30% and 31%

top-15 return rate. Since different shape features support each other and combining them constitute a more comprehensive shape descriptor, we also measured all shape features together. This method achieved 26% top-10, and 31% top-15 rate in which the top-10 accuracy is slightly lower than the best texture method (See Table 6.2). However, the existing database images are not very suitable to be used in shape analysis, since many of them do not have smooth shape contours by various reasons. For example, only 333 of the 380 images have plant region contours, others are mostly close-up images. Moreover, plant images with bad segmentation have noisy contours. Therefore, to measure the success of shape descriptors sufficiently, the shape features are tested on a clean database of 132 images from 32 plant types. The clean database is a segment of the 380 image-database in which each plant type has at least three images with smooth shape contours.

Table 6.3: Shape Analysis Results(Full Database)

Ref.	Method	Top-10%%	Top-15%%	Avg.Min.Rank
1	**Num. of Convex-Concave Pts.**	**0.22**	**0.28**	**75.9**
2	Leaf arc length	0.15	0.21	86.9
3	Leaf base dist.	0.18	0.22	86.7
4	Leaf tip dist.	0.18	0.22	86.9
5	Norm. leaf arc length	0.15	0.21	88.1
6	Leaf Structure(2 + 3 + 5)	0.18	0.24	85.0
7	1 + 2 + 3 + 5	0.25	0.31	79.6
8	1 + 4	0.22	0.30	81.1
9	**All (1-5)**	**0.26**	**0.31**	**79.9**

The shape analysis results on clean database is given in Table 6.4. The maximum top-15 accuracy rate is achieved by number of concave and convex points among the individual shape features, as was the case in Table 6.3. In contrast with the results on 380-image database, leaf base distance and leaf tip distance features are efficient and their accuracy rates are very close to the best method's with 38% and 39% top-15 probabilities respectively. While leaf structure features (leaf arc length + normalized leaf arc length + leaf base distance) did not achieve a satisfactory result,

using these features together with number of concave and convex point features increase the top-15 accuracy from 38% to 47%. As on the 380-image full database, combining all shape features provided the best accuracy rate on clean database as well. The top-10 and top-15 probabilities are 40% and 49% respectively.

Table 6.4: Shape Analysis Results (Clean Database)

Ref.	Method	Top-10%%	Top-15%%	Avg.Min.Rank
1	**Num. of Convex-Concave Pts.**	**0.30**	**0.41**	**41.3**
2	Leaf arc length	0.19	0.31	50.2
3	Leaf base dist.	0.33	0.38	43.8
4	Leaf tip dist.	0.31	0.39	47.1
5	Norm. leaf arc length	0.21	0.28	53.5
6	Leaf Structure(2 + 3 + 5)	0.31	0.38	45.2
7	1 + 2 + 3 + 5	0.41	0.47	38.7
8	1 + 4	0.34	0.44	40.4
9	**All (1-5)**	**0.40**	**0.49**	**39.0**

6.4 Results of Combined Techniques

As the combined methods take into account color, texture and shape-based features, they enhance the overall information and improve the result, hence correct retrieval probability as expected. Table 6.5 indicates the results of the tests combining outstanding color and texture methods.

RGB was the best individual color feature (see Table 6.1) and its combination with different texture methods performed better than combinations with the nRGB and the hue histograms. The combination of RGB and texture methods improved the RGB results from 50% to 55%. As for texture features, even though results of the combined tests using different three texture methods (i.e. accuracy of RGB+default, RGB+maxscale, or RGB+patch-based) are very close, patch-based is typically the most accurate one. Although patch-based texture analysis technique is not very sat-

Table 6.5: Color + Texture Analysis Results

Color	Texture	Top-10%	Top-15%	Avg.Min.Rank
RGB	**Default**	**0.45**	**0.55**	30.9
	Max Scale	0.41	0.53	32.6
	Patch Based	**0.48**	**0.54**	31.6
nRGB	Default	0.40	0.50	31.9
	Max Scale	0.41	0.50	33.6
	Patch Based	0.44	0.52	30.6
HSI(90 bins)	Default	0.39	0.46	35.4
	Max Scale	0.39	0.47	37.4
	Patch Based	0.42	0.51	34.6

isfactory individually, it has increased the accuracy when it is combined with a color method. The combination of patch-based and nRGB features shows an improved result as 52%, while individual patch-based and nRGB methods had achieved 36% and 43% respectively at top-15.

The third image feature family, shape-based features, is also combined with several color and texture features to expand and improve our similarity analysis method. The most powerful shape analysis method is found to be using the full set of features together (see Table 6.3 and 6.4). Therefore, we have chosen this combined shape method as the shape analysis technique while combining shape with color or texture features. Results of shape+color analysis is presented in Table 6.6. The best two methods are using the RGB and nRGB histograms which are followed by several hue histograms and the color co-occurrence matrix at last. The hue histograms perform close to each other, around 45% top-15 probability. Similar to the individual color analysis results, the RGB histogram outperforms the nRGB histogram with a slight difference when the color histograms are combined with the shape features. The top-15 rate of *RGB+shape* and *nRGB+shape* analysis are 51% and 49% respectively. On the other hand, combining shape and texture features did not obtain a satisfactory result with respect to the other combined methods. The highest top-15 accuracy rate is achieved by using patch-based texture and all shape features as 39%.

Table 6.6: Shape (full set) + Color Analysis Results

Color Method	Top-10%	Top-15%	Avg.Min.Rank
RGB	**0.43**	**0.51**	**43.8**
nRGB(64-bin)	**0.39**	**0.49**	**43.7**
nRGB	**0.39**	**0.46**	**43.7**
HSI(10-bin)	0.37	0.45	46.5
HSI(90-bin)	0.37	0.44	45.7
HSI(30-bin smooth)	0.36	0.46	44.6
Color co-occurrence	0.31	0.40	52.5

The next step was to combine features from all three classes: color, texture, and shape. In these tests, shape features are included to the methods that are presented in Table 6.5, and the results are given in Table 6.7. The most successful method is to combine RGB, patch-based, and shape features. This best method results 46% and 54% as top-10 and top-15 accuracy rates. Other methods containing the RGB histogram features, follow this best method with 53% top-15 probability. Shape+nRGB+texture methods come after shape+RGB+texture methods. For instance, shape+nRGB+patch-based method achieved 50% rate at top-15. Methods using the hue histogram are in the lowest order in terms of top-15 rates. For example, shape+hue+patch-based method results 49% probability at top-15. According to the average minimum ranking results, shape+nRGB+patch-based method is the best method giving an average rank of 37.3. Furthermore, although the nRGB histogram features seems worse than the RGB histogram according to the top-15 accuracy rates, the nRGB histogram method results better in terms of the average minimum rank metric. The comparison between combined methods including different texture methods shows that patch-based texture analysis method is the best one and followed by default texture method (see Table 6.5).

Although it seems that using all three feature classes (see Table 6.5) gained no improvement with respect to the color+texture methods, the results are closely related with the database image quality. As indicated in Section 6.3, shape descriptors of most of the plant images are not suitable for shape analysis. In order to express

Table 6.7: Shape + Color + Texture Analysis Results

Shape M.	Color Method	Texture Method	Top-10%	Top-15%	Avg.Min.Rank
All	**RGB**	Default	0.46	0.53	39.7
		Max Scale	0.43	0.53	41.2
		Patch Based	**0.46**	**0.54**	**38.5**
	nRGB	Default	0.42	0.50	38.9
		Max Scale	0.40	0.48	40.5
		Patch Based	0.41	0.50	37.3
	HSI(30-bin smooth)	Default	0.40	0.49	39.6
		Max Scale	0.40	0.48	41.3
		Patch Based	0.42	0.49	39.7

the shape descriptors' quality, new combined tests including shape features are performed on clean database. The test results are provided in Table 6.8. In order to make the comparison easy, two test results are presented at each line: the first one is of the original method without shape, the second one is of the new method with shape (i.e. original method combined with shape features). As depicted in the table, each color and texture method accuracy improved by combining shape features with. Both top-10 and top-15 accuracy rates increased. The highest top-15 rates are achieved by three color histograms: RGB with 61%, nRGB with 60%, and Hue histogram with 60%. The best method after the color histograms is patch-based texture analysis method. With 54% probability this method retrieves the correct plant image at top-15 results.

Finally, to measure the gain of combining all three classes of image features (i.e. color, texture, and shape), we need to test the analysis methods on quality plant images (in terms of image descriptors they have). Therefore, the latest combined tests are run on our 132-image clean database. The test results given in Table 6.9 are the highest accuracy rates among all results and they indicate that database quality has important role on accuracy of the system. While almost each color, texture, and shape feature combinations are tested, only outstanding methods are displayed in the table. The only shape feature used contains all individual shape features and uses scale invariant comparison technique. Scale invariant technique which matches

Table 6.8: Contribution of Shape Features (Clean Database)

	Without Shape		With Shape	
Method	Top-10%	Top-15%	Top-10%	Top-15%
RGB Histogram	**0.50**	**0.58**	**0.55**	**0.61**
nRGB Histogram	0.41	0.46	0.46	0.53
nRGB Histogram(64 bin)	**0.43**	**0.50**	**0.50**	**0.60**
Hue Histogram(10 bin)	**0.40**	**0.53**	**0.50**	**0.60**
Hue Histogram(90 bin)	0.36	0.50	0.50	0.58
Hue Histogram(30 bin smooth)	0.35	0.49	0.47	0.58
Color co-occurrence	0.37	0.43	0.38	0.50
Color co-occurrence(off-diag)	0.39	0.48	0.42	0.50
Default	0.31	0.40	0.40	0.49
Max Scale	0.22	0.34	0.34	0.43
Patch Based	0.38	0.47	0.44	0.54

the image shape features according to the image sizes, performed better than the default one-to-one match. With respect to the top-15 accuracy rates, the best performing method is *shape+RGB+patch-based method*. The highest top-15 rate of 68% is attained by this method. The second best performance is achieved with 65% top-15 rate by *shape+nRGB+default method*. On the other hand, with respect to the top-10 accuracy metric, the best performing method is *shape+nRGB+default texture method* with 54%. However, each feature combination perform similarly for top-10 performance: for instance, *shape+nRGB+patch-based*, *shape+RGB+patch-based*, and *shape+RGB+default* give 53% top-10 rate. Moreover, according to the average minimum ranking the best two methods use shape and patch-based texture features together with the RGB or nRGB histograms. Average rank values are 22.9 and 23.6 respectively which are not very far than the top-15 goal. Consequently, combining different image features improves the system performance rather than using individual features. On the other hand, the significance of the image descriptors (in other meaning quality of the images) is very important for accuracy of the system as well.

The results of CBIR studies are normally evaluated based on *precision* and *recall*

Table 6.9: Shape + Color + Texture Analysis Results (Clean Database)

Shape	Color	Texture	Top-10%	Top-15%	Avg.Min.Rank
All (Scale Inv)	**RGB**	Default	0.52	0.62	24.2
		Max Scale	0.53	0.63	24.5
		Patch Based	**0.53**	**0.68**	**22.9**
	nRGB	Default	0.54	0.65	24.0
		Max Scale	0.51	0.61	25.5
		Patch Based	0.53	0.59	23.6
	HSI(10)	Default	0.51	0.62	27.5
		Max Scale	0.49	0.57	28.5
		Patch Based	0.49	0.57	25.2

rates. Precision is defined to be the ratio of the relevant images in all of the returned images. Recall is defined to be the ratio of retrieved relevant images to all relevant images. These metrics are not very suitable in identification problems where higher precision values are desired, but it is sufficient to have only one relevant image among all the returned images. For identification problems, top-N results are often reported. Furthermore, we also plot the probability of having a relevant result in the top-N, for changing N values.

Figure 6.1 displays the top-N accuracy of each color, texture, and shape method. It is seen that color and shape methods performs similar among themselves and all color methods are better than texture and shape methods. Patch-based approach is more accurate than the default texture method especially between N=20 and N=120. For $N < 40$ the full shape feature method including all the shape features outperforms the number of concave and convex method. Among color methods, nRGB seems to be the best in general; its accuracy is the largest after N=50 but for smaller N values, RGB and color co-occurrence perform better. In Figure 6.2, outstanding individual and combined methods are compared on the clean database. Accuracy curves of two combined methods (RGB+patch+shape and nRGB+default+shape) are indistinguishable and outperform the separate texture and shape methods. Only the RGB histogram among the individual methods has higher accuracy curve. The third combined method, RGB+shape, seems worse after N=10, however it is the

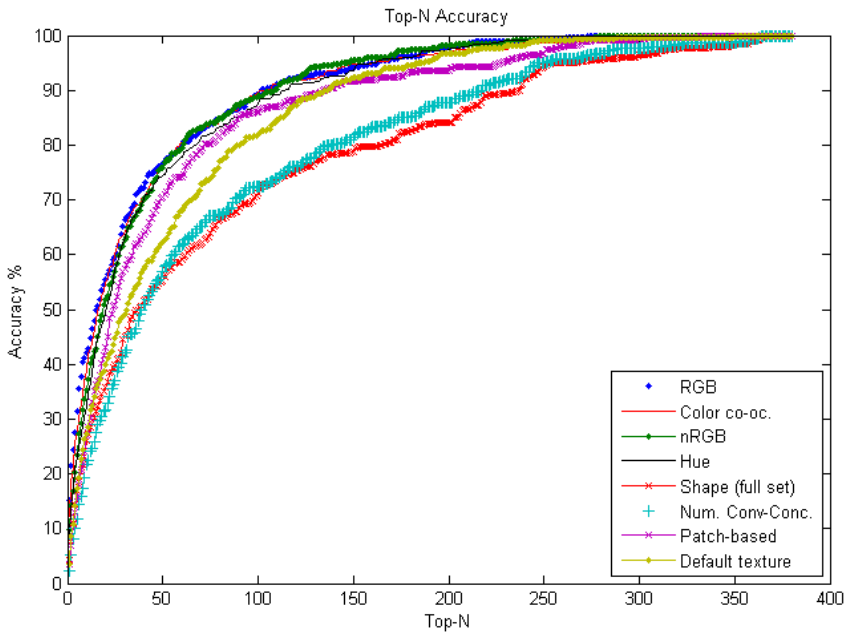

Figure 6.1: Accuracy graph for each color and texture method

best among all for N<10. Considering best methods in Figure 6.1 and 6.2, 50%, 60%, 70% and 90% accuracy rates are achieved on N=8, 12, 15 and 100, respectively.

While a direct comparison is not meaningful, the fact that many CBIR problems report precision values around $50-60\%$ [8,9,43,45], can give an idea of the difficulty of the CBIR problem. Shape-based retrieval in botanical collections [11], which is the closest study to our problem, reports precision rates as 0.92 at top-5 and 0.88 at top-10 retrieval ranks. These results are obtained on the public Swedish tree leaves database. The database consists of 1125 isolated leaves (single leaves on a plain background) from 15 different Swedish trees species. We can think that the current problem is at least as difficult, since identifying the plant based on its leaf shape is a subset of our problem.

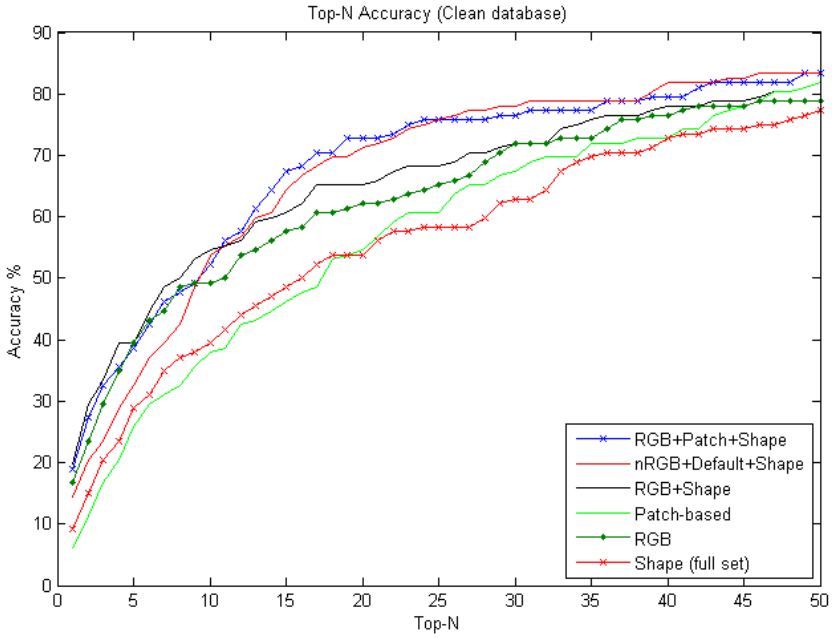

Figure 6.2: Accuracy graph for outstanding color, texture and combined methods

Chapter 7

CONCLUSION

In this thesis, we present a content-based image retrieval system, combining various content-based image retrieval approaches, with a segmentation preprocessing step. Separating the plant from the background using the max-flow min-cut segmentation technique has given us an opportunity to focus solely on the plant, which increased the consistency of the retrieved global features. Furthermore, combining different color, texture, and shape features extracted from the plant region enhances the accuracy of the system. Common techniques are used in color and texture feature extraction steps. Specifically, global color features of plant regions are extracted in the form of color histograms and color co-occurrence matrices. Similarly, global texture features are extracted by Gabor filters which are commonly used in texture analysis. Shape-based features are extracted using the plant contour which is obtained by tracing the outer plant boundary. These shape features are sharpness of the plant contour, structure of plant leaves (leaf arc length, normalized leaf arc length, and leaf base distance), and distribution of leaves in the plant region.

One of the contributions of the thesis is a modification of the default texture analysis, where we focus on the local features obtained from (sub)leaf regions. The proposed patch-based approach represents an image with $m \times n$ patches in which each patch is roughly expected to correspond to a plant leaf or leaf structure. The

novelty of this technique is that, rotating the approximated leaf patches to a canonical direction provides the effect of rotating the plant leaves to the same orientation. Consequently, by rotating these patches to a canonical direction, we obtain some invariance towards the rotation of the individual leaves. The contribution on shape descriptors is two new shape features: i) sharpness of the plant contour, ii) distribution of the plant leaves. Sharpness of the plant contour signifies the count of sharp leaf tips and thorn-like structures. Leaf structures are recognized and their structural information is obtained around these sharp points. The distribution of the plant leaves is measured by the distance between sharp points.

The evaluation of the system using the top-N identification rate shows the best top-15 performance of 68% on the clean database and 54% on the full database using all the features. Since, the full database with 380 images contains some low-quality images that are useless for shape analysis, plant images with smooth shape contours are selected among this full database and the clean database has emerged. The best accuracy within individual color methods is obtained using the RGB histogram, with a 50% top-15 rate. Although, individual texture analysis methods do not give sufficient results, combining a texture method with color methods increase the accuracy of the color method drastically. Using color, texture, and shape features together has improved the system performance as well. For example, integrating the patch-based texture features to the *shape+RGB* test has increased the top-10 rate from 61% to 68% on the clean database (see Table 6.9). Similarly, the best result on the full database (54% top-15 retrieval rate) is achieved with a combined method -of color, texture, and shape analysis techniques- as well.

Our studies demonstrated that the method we proposed has promising results in the plant retrieval problem, however the work is still preliminary. Further studies might extend and improve the proposed system by integrating additional features such as sharpness histogram, color distribution with spatial information, and so on. The size of the database and the number of plant types in it are intended to be increased. In addition, the system might be finalized by integrating the implemen-

tation as a web service.

The contributions of this thesis are:

- development of a plant identification problem as a particular application of CBIR,
- using the segmented plant region for feature extraction to remove the background and increase the quality of the extracted features,
- evaluating different features for their effect on overall system performance,
- proposing new texture matching technique to provide invariance and some new shape descriptors that provide the outer contour characteristics of a plant.

7.1 Future Work

Insufficiency of segmentation is one of the primary challenges in image retrieval systems that use segmented images. Erroneous segmentation causes the feature extraction to be inaccurate. While, global color and texture features that we have used are not sensitive to segmentation quality, shape features are too sensitive, since they only depend on the outline contour of the plant region. The effect of bad segmentation to the color features is; to store color histogram data of some background pixels involved in the data of plant pixels. If the amount of the background that is labeled as foreground is small proportional to the plant region, the negative effects can be avoided. The most resistant feature to erroneous segmentation is texture-based features. The background has no effect on extracted texture features, if it does not contain a significant texture. In contrast, shape features are useless if the outline of the plant could not be revealed.

Setting constraints for plant images might be an alternative solution for segmentation problems which is easy to apply. For example, the outer contour of the query

plant should be visible which is a must to trace the outer boundary of the plant. Further, it is better if the images do not include complex background that may intermix with the plant; this might be another constraint for the input images. As a result, the max-flow min-cut segmentation algorithm can smoothly segment a plant which is on a plain (one color, no texture) background. Furthermore, segmentation algorithm may be improved by exploiting the domain information, such as the green color or the fact that we expect to see the outer contour of plant. Using the RGB color intensities rather than the gray-scale might be a solution in which segmentation algorithm distinguishes the green plant regions from non-green background very easily.

Another challenge encountered in this CBIR system is the scale or resolution variance. The plant images in the database may have different resolutions. To match a violet image in 300×400 pixels with another violet image in 1200×1600 pixels is an issue. In texture analysis, the max scale method is proposed to solve this problem. Similarly a scale invariant comparison technique is implemented for shape analysis. Shape features are extracted in three different scales, and the suitable scale is selected with respect to the image size for each plant image. However, scale invariance is not an issue in color analysis, since histogram values are normalized with the number of pixels.

A further development on the shape-based analysis might be generating sharpness histograms. To use a histogram indicating the distribution of the sharpness (direction changes) on the shape contour would be an improvement, instead of using only two numbers; number of convex and concave points. Notice that we express sharpness (direction changes) with the values in the interval [0-4]. A possible set up is: reserving one bin for each 0.5 or 0.25 sharpness measure, so generating a 8-bin or 16-bin histogram. Although this new approach is more sensitive to roughness on the surface, with sufficient shape descriptors it presents more information about the shape of the image. Besides sharp points, smoother ones even straight contour regions and their quantity will be retrieved and measured as well.

Using the fact that main color of plants is green and flowers have different various colors, plants can be classified as *flower-plants* and *plants without flower*. This distinction may increase the accuracy by simplifying the problem and even decrease the query time. Further, spatial color distribution might be added as a new feature which helps to measure the distribution of plant flowers for example. If the flowers (non-green plant regions) of a plant are spread out, spatial distribution of the non-green pixels would have higher values than the plant with a single large flower.

Appendix A

The list of plant types in our plant database

A total of 108 plant types are indexed, while only 78 of them have segmented images in our database. While all these 108 indexed types are listed below, the number of segmented images in 78 plant types are also indicated.

1-agalonema commutatum 4
2-aglaonema modestum 4
3-agalonema silverqueen 4
4-Fatsia Japonica
5-oncidium orchid
6-aloe 4
7-anthurium 7
9-asplenium nidus 4
10-chlorophytum comosum 3
11-codiaeum 7
12-diffenbachia tropicsnow 3
13-dracaena marg 3
14-monstera deliciosa 3
15-guzmania 4
16-dracaena janet 8
17-hedera helix 3
18-marantha leuconeura 6
19-pachypodium 3
20-philoddomesticum (Philodendron) 4
21-setcreasea 5
22-peperomia argyreia 6
23-peperomia clusiifolia
24-sansevieria cylindrica 3
25-tradescantia zebrina 11
26-chamaedora elegans 3

27-aechmea 3
28-syngonium 3
29-hypoestes sanguinolenta 4
30-tradescantia spathacea (rhoeo)
31-african violet 13
32-portulacaria afra 7
33-sansevieria trifasciata 6
34-caladium 4
35-bamboo 4
36-calatheabella 3
37-bougainville aglabra 3
38-dieffenbachia camille
39-aspidistra 4
40-schlumbergera (christmas cactus)
41-Sedum rubrotinctum   4
42-Cyclamen 6
43-Narcissus 3
44-Hyacinth 8
45-Tagetes petula 3
46-Sedumreflexum
47-Fritillaria whittallii
48-euphorbia milli 5
49-begonia 7
50-rhododendron 4
51-pelargonium zonale 5
52-kentia 4
53-Phalaenopsis (orchid) 3
54-cycas revoluta 3
55-schefflera arboricola 12
56-Murraya paniculata 4
57-Miltoniopsis 3
58-ficus elastica 6
59-Mimosa pudica
60-aphelandra squarrosa 5
61-Eustoma grandiflorum 3
62-Hippeastrum 3
63-Saxifraga stolonifera 3
64-Epiphyllum 5
65-Amaryllis 3
66-Crassula_ovata
67-Epipremnum aureum
68-Ficus benjamina 3
69-Nephrolepis exaltata
70-Echinopsis subdenudata
78-Proboscidia louisianica 3

79-spathiphyllum maunaloa 5
80-Cactaceae 6
81-Bonsai 7
82-Opuntia 8
83-sinningia speciosa 4
84-kalanchoe blossfeldiana 6
85-Crocus
86-cyclamen
87-Zygocactus 7
88-Amorphophallus 9
89-Achimenes 4
90-Aeschynanthus 4
91-dionaea 5
92-Echeveria 7
93-Gardenia 4
94-Hoya 4
95-Balsaminaceae (impatients) 5
96-Nerium(oleander)
99-streptocarpus
100-musa species
101-areca 4
102-cupressus
103-cymbidium orchid 5
104-dianthus barbatus 5
105-tulip 9
106-buxus sempervirens 3
107-ficus ginseng
108-yucca 3

Appendix B

Pseudo-codes of the contour tracing and related algorithms.

B.1 The pseudo-code of the contour tracing algorithm.

```
# runsteps: # of tracing run steps used for sharpness measure
::::
start(x,y) := FindStartingPoint();
dir := 1; // starting direction is east
while(counter < limit)
p_next(x,y) := MoveTo(p_current(x,y),dir);
counter++;
if(p_next(x,y) == start(x,y))
break;
end
if(p_next.y >= height) //p_next exceeds bottom limit of the image
dir := 5;
elseif(p_next.x >= width) //p_next exceeds right
dir := 3;
elseif(p_next.x < 0) //p_next exceeds left limit
dir := 7;
elseif(p_next.y < 0) //p_next exceeds top limit
if(p_next is parsed before)
dir := (dir+7) % 8; //turn left
else
dir := (dir+1) % 8; //turn right
end
else //p_next is in the image region
next := image[p_next(x,y)];
if(next == 0) //not edge point
dir := (dir+1) % 8;
else
p_current := p_next; //extend path
p_current.dir := dir;
```

```
Contour[contourIndex] := p_current;
index := contourIndex % runsteps;
array[index] := dir;
sharpness = MeasureSharpness(array, index);
if (sharpness >= 2 and <= 4) //if sharp enough
SharpPointsArray[sharpEdgeIndex] = Contour[contourIndex-runsteps]; //the breakpoint
sharpEdgeIndex++;
end
contourIndex++;
dir := (dir+7) % 8; //turn left
end
end
end
```

B.2 The pseudo-code of the labelling algorithm for given sharp points

```
for each s in SharpPointsArray
dir = s.dir; //direction of s
p(x,y) = MoveTo(s(x,y),dir);  //p is the first point on the imaginary line
while (p(x,y) on contour)
p(x,y) = MoveTo(p(x,y),dir);  //continue straight on the imaginary line
end
if(p(x,y) is in the region)
s(x,y).label = "‘concave"’;
else
s(x,y).label = "‘convex"’;
end
advance s;
end
```

B.3 The pseudo-code of the convex/concave point differentiation algorithm

Algorithm that is used to detect sharp points:

```
# Contour: the contour list
# i: index of p (number of points currently in the list)
# RecentSharpnessArray: stores last n sharpness measure
# n: also called as trace run steps (runsteps)
for each new contour point p
Contour[i] = p;
```

```
RecentSharpnessArray[i%n] = p.dir;
sharpness = MeasureSharpness(RecentSharpnessArray, i%n);
if (sharpness >= 2 and <= 4) //if sharp enough
SharpPointsArray[j] = Contour[i-(n/2)]; //the breakpoint is (n/2) points before p
end
end
```

Function; MeasureSharpness, that is called from the above code:

```
# index: keeps rounding index of the array
# array: stores last n sharpness measure
# n: also called as trace run steps (runsteps)
float MeasureSharpness(array, index)
for i=0...n/2
avgDir1 += array(index+i);
end
for i=n/2...n
avgDir2 += array(index+i);
end
diff = |avgDir1 - avgDir2|;
if (diff>4)
diff=8-diff;
return diff;
end function
```

Bibliography

[1] S. Aksoy, "Image segmentation," April 2007. [Online]. Available: http://www.cs.bilkent.edu.tr/ saksoy/courses/cs484-Spring2007/slides/cs484_segmentation.pps

[2] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 9, pp. 1124–1137, Sept. 2004.

[3] V. Prasad and J. Domke, "Gabor filter visualization," 2005. [Online]. Available: http://www.cs.umd.edu/class/spring2005/cmsc838s/assignment-projects/gabor-filter-visualization/report.pdf

[4] J. P. Eakins, "Towards intelligent image retrieval," *Pattern Recognition*, vol. 35, no. 1, pp. 3 – 14, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/B6V14-4477X94-2/2/d2ecb0153a72765e3bc38c0eb42dadab

[5] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1992.

[6] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by image and video content: the qbic system," *Computer*, vol. 28, no. 9, pp. 23–32, Sep 1995.

[7] J. R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. C. Jain, and C. F. Shu, "Virage image search engine: an open framework for image management," I. K. Sethi and R. C. Jain, Eds., vol. 2670, no. 1. SPIE, 1996, pp. 76–87.

[8] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain, "Content-based multimedia information retrieval: State of the art and challenges," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 2, no. 1, pp. 1–19, 2006.

[9] R. C. Veltkamp and M. Tanase, "Content-based image retrieval systems: A survey," 2000. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.7344

[10] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Trans. Pattern Anal. Mach. Intell.*, no. 12, pp. 1349–1380, December 2000.

[11] I. Yahiaoui, N. Hervé, and N. Boujemaa, "Shape-based image retrieval in botanical collections," in *PCM*, 2006, pp. 357–364.

[12] F.-Y. Lin, C.-H. Zheng, X.-F. Wang, and Q.-K. Man, "Multiple classification of plant leaves based on gabor transform and lbp operator," in *ICIC (3)*, ser. Communications in Computer and Information Science, D.-S. Huang, D. C. W. II, D. S. Levine, and K.-H. Jo, Eds., vol. 15. Springer, 2008, pp. 432–439.

[13] Z. Wang, Z. Chi, and D. Feng, "Shape based leaf image retrieval," *Vision, Image and Signal Processing, IEE Proceedings* -, vol. 150, no. 1, pp. 34–43, Feb 2003.

[14] Z. Wang, Z. Chi, D. Feng, and Q. Wang, "Leaf image retrieval with shape features," in *VISUAL '00: Proceedings of the 4th International Conference on Advances in Visual Information Systems*. London, UK: Springer-Verlag, 2000, pp. 477–487.

[15] Q.-K. Man, C.-H. Zheng, X.-F. Wang, and F.-Y. Lin, "Recognition of plant leaves using support vector machine," in *ICIC (3)*, ser. Communications in Computer and Information Science, D.-S. Huang, D. C. W. II, D. S. Levine, and K.-H. Jo, Eds., vol. 15. Springer, 2008, pp. 192–199.

[16] D. M. Woebbecke, G. E. Meyer, K. V. Bargen, and D. A. Mortensen, "Plant species identification, size, and enumeration using machine vision techniques on near-binary images," J. A. DeShazer and G. E. Meyer, Eds., vol. 1836, no. 1. SPIE, 1993, pp. 208–219. [Online]. Available: http://link.aip.org/link/?PSI/1836/208/1

[17] S. Yonekawa, N. Sakai, and O. Kitani, "Identification of idealized leaf types using simple dimensionless shape factors by image analysis," *Transactions of the ASAE*, vol. 39, pp. 1525–2533, 1996.

[18] S. Abbasi, F. Mokhtarian, and J. Kittler, "Reliable classification of chrysanthemum leaves through curvature scale space," in *SCALE-SPACE '97: Proceedings of the First International Conference on Scale-Space Theory in Computer Vision*. London, UK: Springer-Verlag, 1997, pp. 284–295.

[19] A. J. Prez, F. Lpez, J. V. Benlloch, and S. Christensen, "Colour and shape analysis techniques for weed detection in cereal fields," *Computers and Electronics in Agriculture*, vol. 25, no. 3, pp. 197 – 212, 2000.

[20] J. Jia and G. Krutz, "Location of the maize plant with machine vision," *Journal of Agricultural Engineering Research*, vol. 52, pp. 169 – 181, 1992.

[21] D. G. S. Jr, F. A. C. Pinto, D. M. Queiroz, and P. A. Viana, "Fall armyworm damaged maize plant identification using digital images," *Biosystems Engineering*, vol. 85, no. 4, pp. 449 – 454, 2003.

[22] K. V. B. D. M. Woebbecke, G. E. Meyer and D. A. Mortensen, "Color indices for weed identification under various soil, residue and lighting conditions," *Transactions of the ASAE*, vol. 38, no. 1, p. 256269, 1995.

[23] R. C. Gonzalez and R. E. Woods, *Digital Image Processing.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001.

[24] L. Grady, Y. Sun, and J. Williams, "Interactive graph-based segmentation methods in cardiovascular imaging," in *Handbook of Mathematical Models in Computer Vision*, N. Paragios, Y. Chen, and O. Faugeras, Eds. Springer, 2006, pp. 453–469.

[25] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik, "Blobworld: a system for region-based image indexing and retrieval (long version)," EECS Department, University of California, Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/CSD-99-1041, 1999. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/1999/5567.html

[26] S. C. Locke, "Menger's theorems and max-flow-min-cut," July 1996. [Online]. Available: http://www.math.fau.edu/locke/Menger.htm

[27] C. C. Venters and D. M. Cooper, "A review of content-based image retrieval systems," Manchester Visualization Centre, Manchester Computing, University of Manchester, Manchester, UK, Tech. Rep., June 2000. [Online]. Available: http://www.jisc.ac.uk/publications/documents/contentreviewfinalreport.aspx

[28] H. T. Tico M and K. P., "A method of color histogram creation for image retrieval," vol. 2, June 2000, pp. 157–160.

[29] B. Zarit, B. Super, and F. Quek, "Comparison of five color models in skin pixel classification," 1999, pp. 58–63.

[30] S. Sural, G. Qian, and S. Pramanik, "Segmentation and histogram generation using the hsv color space for image retrieval," vol. 2, 2002, pp. II–589–II–592 vol.2.

[31] S.-O. Shim and T.-S. Choi, "Image indexing by modified color co-occurrence matrix," vol. 3, Sept. 2003, pp. III–493–6 vol.2.

[32] A. L. K N. Plataniotis, A N. Venetsanopoulos, *Color image processing and applications*. Springer, 2000.

[33] M. C. Shin, K. I. Chang, and L. V. Tsap, "Does colorspace transformation make any difference on skin detection?" in *WACV '02: Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision*. Washington, DC, USA: IEEE Computer Society, 2002, p. 275.

[34] M. Šonka, V. Hlaváč, and R. D. Boyle, *Image Processing, Analysis and Machine Vision*, 1st ed. London, UK: Chapman and Hall, 1993.

[35] W. Ma and B. Manjunath, "Texture features and learning similarity," Jun 1996, pp. 425–430.

[36] J. Han and K.-K. Ma, "Rotation-invariant and scale-invariant gabor features for texture image retrieval," *Image and Vision Computing*, vol. 25, no. 9, pp. 1474 – 1481, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/B6V09-4MNYYW5-3/2/60f49ea02e8e3d0c9d30da338aa5e18c

[37] M. I. Dengsheng Zhang, Aylwin Wong and G. Lu, "Content based image retrieval using gabor texture features," December 2000, p. 392395.

[38] B. Manjunath and W. Ma, "Texture features for browsing and retrieval of image data," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 18, no. 8, pp. 837–842, Aug 1996.

[39] C. Liu and H. Wechsler, "A gabor feature classifier for face recognition," vol. 2, 2001, pp. 270–275 vol.2.

[40] A. El-ghazal, O. A. Basir, and S. Belkasim, "Shape-based image retrieval using pair-wise candidate co-ranking." in *ICIAR*, ser. Lecture Notes in Computer Science, M. S. Kamel and A. C. Campilho, Eds., vol. 4633. Springer, 2007, pp. 650–661.

[41] J. S. Park and T.-Y. Kim, "Shape-based image retrieval using invariant features." in *PCM (2)*, 2004, pp. 146–153.

[42] S. Abbasi, F. Mokhtarian, and J. Kittler, "Curvature scale space image in shape similarity retrieval," *Multimedia Syst.*, vol. 7, no. 6, pp. 467–476, 1999.

[43] Q. Iqbal and J. K. Aggarwal, "Combining structure, color and texture for image retrireval: A performance evaluation," vol. 2, August 2002, pp. 438–443.

[44] M. Arevalillo-Herrez, J. Domingo, and F. J. Ferri, "Combining similarity measures in content-based image retrieval," *Pattern Recognition Letters*, vol. 29, no. 16, pp. 2174 – 2181, 2008.

[45] Y. M. Wong, S. Hoi, and M. Lyu, "An empirical study on large-scale content-based image retrieval," July 2007, pp. 2206–2209.

[46] D.-S. Huang, D. C. W. II, D. S. Levine, and K.-H. Jo, Eds., *Advanced Intelligent Computing Theories and Applications. With Aspects of Contemporary Intelligent Computing Techniques, 4th International Conference on Intelligent Computing, ICIC 2008, Shanghai, China, September 15-18, 2008, Proceedings*, ser. Communications in Computer and Information Science, vol. 15. Springer, 2008.