Plant Leaf Classification

Cem Kalyoncu

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Eastern Mediterranean University
August 2015
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Serhan Çiftçioğlu
Acting Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Doctor of Philosophy in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy in Computer Engineering.

Asst. Prof. Dr. Önsen Toygar
Supervisor

Examining Committee

1. Prof. Dr. A. Aydın Alatan
2. Prof. Dr. Hasan Demirel
3. Prof. Dr. Adnan Yazıcı
4. Asst. Prof. Dr. Önsen Toygar
5. Asst. Prof. Dr. Ahmet Ünveren

ABSTRACT

Identification of plants is an important subject that has many practical uses. In this thesis, we devoted our efforts to identify plants through images of their leaves. The reason behind this choice is that the plants are complicated organisms, therefore, it is appropriate to identify plants through their leaves.

Leaf classification is a multi-disciplinary field that requires knowledge in botany, image processing, and pattern recognition. Additionally, experimentation requires large datasets to be performed accurately. Many methods in the literature concentrate on a single descriptor to describe a leaf. However, in this thesis, we concentrate on multiple descriptors to describe the leaf from different aspects. The biggest challenge in using multiple descriptors is identifying the descriptors that complement each other without significant overlaps. Additionally, not every descriptor is meaningful for every leaf type. Therefore, a class-based prioritizing classifier is required to deal with these issues. In this study, we employ Linear Discriminant Classifier (LDC) for this task.

We propose two leaf classification methods, namely Geometric Leaf Classification (GLC) and Combination of Geometric, Texture and Color Features for Leaf Classification (GTCLC) in this thesis. These methods include new features such as Sorted LBP, application of LDC for leaf classification and several feature types combined to improve the classification accuracy. First of all, geometric features are used for the classification of plant leaves. Then, a number of features, such as geometric, shape, texture, and color features are combined to perform leaf classification. During the experiments these methods are compared with the state-of-the-art methods. According to these experiments, GTCLC outperforms all methods both in terms of accuracy and suitability.

Keywords: leaf classification, pattern recognition, feature extraction, geometric features, LBP, CIE-LCH

ÖZ

Yaprakların tanımlanması, bir çok kullanımı olan, önemli bir konudur. Biz de bu tez esnasında bitki yapraklarının tanımlanması üzerine yoğunlaştık. Yaprakları tercih etmemizdeki sebep, bitkilerin genel olarak karmaşık canlılar olmasına rağmen, yaprak görüntülerinin düzenli ve bitkiyi yeterli miktarda ifade edebilmesinden kaynaklanmaktadır.

Yaprakların tanımlanması, birden fazla bilim dalına ait bilgi gerektirmektedir; bunlar: botanik, resim işleme ve örüntü tanıma alanlarıdır. Ek olarak, deneysel çalışmaların kesin sonuçlara ulaşabilmesi için büyük boyutlu veritabanlarına ihtiyaç duyulmaktadır. Literatürdeki bir çok yöntem tek bir tip tanımlayıcı kullanmaktadır. Ancak biz bu tezde, yaprakları farklı yönleriyle tanımlayan çok sayıda tanımlayıcı kullanmaktayız. Çoklu-tanımlayıcılı sistemlerin en büyük zorluğu bir birlerine destek olurken, aynı bilgiyi tekrarlamayan tanımlayıcıların tespitidir. Ek olarak, her tanımlayıcı, tüm yaprak tipleri için anlam ifade etmemektedir. Bu sebepten dolayı, sınıf-tabanlı önceliklendirme yeteneğine sahip bir sınıflandırma sistemi kullanılmalıdır. Biz de bu çalışmada Doğrusal Ayırtaç Sınıflandırıcısı'nı (Linear Discriminant Classifier, LDC), bu sorunları çözebildiğinden dolayı kullanmaktayız.

Çalışmalarımız esanasında, Geometrik Yaprak Tanımlama (GLC) ve Geometrik, Doku ve Renk Tabanlı Yaprak Tanımlama (GTCLC) olarak iki ayrı sistem öne sürdük. Bu yöntemler, Sıralanmış Yerel İkili Örüntüler (SLBP) gibi yeni tanımlayıcılar, LDC'nin kullanımı ve farklı tipte tanımlayıcıların birlikte kullanılması gibi yenilikler içermektedir. Deneyler esnasında bu yöntemler literatürdeki yöntemlerle karşılaştırılmıştır. Bu deneylerin sonucunda, GTCLC yönteminin hem performans, hem de uyumluluk yönünden en iyi sonucu verdiği tespit edilmiştir.

Anahtar kelimeler: yaprak sınıflandırma, örüntü tanıma, öznitelik bulma, geometrik öznitelikler, Yerel İkili Örüntü, CIE-LCH

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

AC	Area Complexity
AR	Aspect Ratio
CC	Circularity
CIE	International Commission on Illumination
GLC	Geometric Leaf Classification
GTCLC	Combination of Geometric, texture, and color features
HSL	Hue, saturation, lightness color space
HSV	Hue, saturation, value color space
KNN	K-Nearest Neighbors
LBP	Local Binary Patterns
LCH	Luminance, chromacity, hue
LDC	Linear Discriminant Classifier
MBR	Minimum Bounding Rectangle
MDM	Multiscale Distance Matrix
NNC	Nearest Neighbor Classifier
PC	Perimeter Complexity
PR	Perimeter Ratio
QDC	Quadratic Discriminant Classifier
RG	Rectangularity
SLBP	Sorted Local Binary Patterns
SP	Sphericity
SLBP	Sorted Local Binary Patterns with Feature Reduction
SVM	Support Vector Machines

Chapter 1

INTRODUCTION

Plant classification is required for automated systems in medicine and food industries. Additionally, a fully functional plant classification system can help its users to identify plants in nature. Without the ability to match plants using their photographs, finding the name and related information regarding to a specific plant requires user to define plant verbally. Verbal definition of a plant is both hard and ambiguous. Therefore, automated or semi-automated recognition and/or retrieval of plants are helpful in these contexts.

Plants are complicated organisms. The visual look of a plant is chaotic and varies greatly. Worse yet, plants look similar even if they are not from the same species. However, it is possible to determine type of a plant from various factors such as its size, flowers, leaves, and density of its foliage. Easy-to-access data acquisition devices, such as digital camera, cannot determine the correct size of an object. Therefore, the size information cannot easily be obtained. Additionally, not every plant species have flowers and most flowering species only have flowers for short periods. These problems also remove flowers from plant identification. On the other hand, leaves of a plant can easily be accessed. Additionally, leaves contain great deal of information that can be extracted to classify plants. These reasons increase the use of leaves in plant recognition.

Despite its uses, leaf and plant classification was a subject that did not receive attention until 2011. With ImageCLEF plant identification campaigns, a certain level of interest is achieved and plant and leaf identification became an active topic. The success of leaf classification has increased substantially since then. However, due to the number of plant species and the variance in leaf and plant structure, there is still room for improvement.

We concentrate our efforts on leaf classification in this study. Unlike many other methods [2,3], our research focuses on multiple descriptors. Our primary aim is to describe the leaf from different aspects. In pattern recognition, it is important to keep dimensionality low, as the training phase of the classification systems are adversely affected by increasing the number of dimensions. Therefore, our secondary aim is to refine and eliminate the descriptors so that there will not be any redundant features.

In this thesis, we successfully identified descriptors that work well together. Additionally, we have developed three novel feature descriptors; margin distance, margin statistics, and Sorted Local Binary Patterns. Moreover, we analyzed and compared different classifiers to identify the best classifier that can work with multiple descriptors efficiently. These improvements improved the field significantly in terms of both accuracy and suitability to real life scenarios.

In Chapter 2, existing methods in the literature are analyzed and presented in an organized fashion. Background information on topics that are used in this thesis are summarized in Chapter 3. Additionally, terminology that is often used while describing leaf is listed. In Chapter 4, the methods that are developed during this thesis are discussed in detail. Experimental results comparing the proposed methods with the existing methods in the literature are given in Chapter 5. Finally, Chapter 6 discusses the conclusion of this thesis and possible future works.

Chapter 2

LITERATURE OVERVIEW

Leaf classification is an active research topic with many publications in the recent years [2, 4–12]. In this chapter, the developments in this field are summarized.

The applications of leaf classification in the literature can be separated into three types: identification, categorization, and retrieval. Identification involves in identifying the species that a particular leaf belongs to. There are many researches on leaf identification [2, 3, 5, 6, 8–22]. Content based image retrieval is generally used to build applications to serve requests made by humans. These systems cannot be used in automated system as the lack of the final decision mechanism. These methods are useful for leaf searches as humans can easily select the correct plant from a short list of candidates. The notable articles related to this topic includes [4, 7, 21, 23–26] Leaf categorization separates leaves depending on their properties. This separation can be useful by its own or could be used to enhance identification methods. [17, 27, 28] are some of the papers related with leaf classification.

Similar to the other classification tasks, leaf classification requires features to work. Since the leaf shape contains most of the information regarding to the type of the leaf, shape based features are more commonly used. In [2, 6, 9, 10], new shape based features for leaf classification are proposed. Additionally, [4, 7, 8, 13, 15–17, 19–27] use previously known shape-based features in their systems. There are few methods [11, 14, 18, 28] that rely on texture alone, however, texture features are combined with shape features in [4, 5, 7, 19, 25] to improve the classification accuracy.

Even though leaf margins are a part of leaf shape, it requires specific features to be used. For most global shape methods, margins are indistinguishable from noise. The systems in [3, 6, 10, 20] contain features that can effectively represent margins.

The final step in a computer vision system is the classification. Even though there are many different classification methods used in leaf classification, the most common method is KNN, with $K = 1$ in most systems. The papers that use KNN includes [2, 7, 10, 13, 14, 18–20, 22–25, 28]. However, SVM is also a popular method used in many researches such as [5, 8, 26].

Dynamic Time Warping algorithm is used in [13] to extract features. However, this method is not rotation invariant. Authors propose a dynamic algorithm to match biological sequences to achieve rotation invariance. This method calculates the distance between two samples. Using this distance metric, KNN is employed to classify leaf images. Six different plant species are used to demonstrate the performance of the system. Depending on the data set used, this method achieves up to 97% accuracy.

Authors in [23] use a modified Minimum Perimeter Polygons (MPP) to extract shape based features. These features are then used to retrieve relevant matches from the database. In this research a dynamic matching algorithm simply called as SMP is proposed for similarity calculations. The results in this paper show that Improved

MPP method outperforms other contour based methods. Additionally, SMP method reduces processing time to half of nearest neighbor search.

A method that combines wavelet transform and Gaussian interpolation is proposed in [14]. This method decomposes the grayscale leaf image using wavelet transform and adds in the lost details using Gaussian interpolation. On a database containing 20 classes, this method achieves score of 93% using nearest neighbor classifier.

A probabilistic neural network system that works on geometric features is proposed in [3]. This method extracts 12 commonly used geometric features from segmented leaf image. Principal Component Analysis (PCA) is employed to reduce the number of features from 12 to 5. These features are then used in a Probabilistic Neural Network (PNN) to perform classification. The experiments are performed on 32 kinds of plants with an accuracy of 90%. This method requires user to enter the start and the end points of the midrib. Therefore, it cannot be used for automated classification tasks.

A Content-Based Image Retrieval method is proposed for leaves in [28]. This method determines starting and ending points of the vein in the leaf to classify venation pattern before performing retrieval using shape information. The authors demonstrate that using venation classification increases the precision of the system by 25% on average. However, experimental method in this paper is not discussed in detail. Additionally, this method requires hand drawn samples to work, which increases the work necessary to adapt this system.

An extensive research on the use of geometric features is presented in [15] and [16]. In this research, 8 geometric features in addition to 7 moment invariants are used to describe leaf shape. After feature extraction, this system performs linear scaling and Wilson Editing. This method uses Moving Center Hypersphere Classifier, which is proposed specifically for leaf classification. MCH Classifier is a variant of KNN and works in a similar fashion. However, it is much faster even though its performance decreases faster, compared to KNN, with the increased number of classes.

A method that deals with leaves that have complicated backgrounds is proposed in [17]. This system uses Watershed segmentation algorithm and Zenerike's Moments for leaf classification. Authors have shown that this system can reach up to 92.6% classification accuracy using MCH Classifier. Even though the accuracy is not comparable to most methods, the segmentation algorithm that is used in this paper is quite useful.

Authors in [27] propose a method that uses shape and venation features for leaf image retrieval. Minimum Perimeter Polygons (MPP) are used to extract shape features and uses SMP method for content based image retrieval. Results are only demonstrated for venation based retrieval which divides leaves into three venation based classes. This research is a combination of the earlier works of authors that are presented in [28] and [23].

An information retrieval method using Inner Distance Shape Context (IDCS) is proposed in [24]. Even though this classification method is well-known and used before, authors demonstrated the complete system implementation, from segmentation to presenting the results, an often overlooked part, of the classification system. This detailed system allows Smithsonian Institution National Museum of Natural History botanists to catalog and search three new databases.

Another method that works on leaf textures is proposed in [18]. In this method, Gabor co-occurrences are used as features. For classification, this method uses KNN with Jeffery-divergence distance measure. The reported results display 85% accuracy on a hand selected leaf texture database containing 32 classes. Performing this algorithm on randomly selected sections of leaves produces 80% accuracy. Even though the accuracy of the system is low, this system does not use the shape of a leaf to obtain these scores, therefore, it is a useful method that can be combined with shape-based methods to achieve better classification.

An Incremental Classification method is proposed in [19]. This method uses shape and texture features. For shape features, gradient histogram of the leaf contour is employed. Similarly, texture gradient orientation histogram is used for texture representation. Both descriptors are classified using nearest neighbor classifier with Jeffrey Divergence Distance. An incremental combination strategy is employed for final decision. Authors show that the combined system achieves 81% accuracy. However, this paper does not state any methods to achieve rotation invariance as the gradient orientations depend on the leaf orientation.

In [25], a method to search a database of leaf images is proposed. This method uses shape, texture, and color features. Authors used SIFT for shape, HSV for color, and log-Gabor for texture representation. The results are demonstrated on a database with 45 classes that contains rotations of the same image. This method achieves 97% on the aforementioned database.

A novel contour-based shape descriptor which is named as Multiscale Distance Matrix (MDM) is proposed in [2]. MDM is extracted over selected points in the image contour. Originally, authors proposed to use 64 nodes. Details of this method are described in Section 3.6. This method is invariant to rotation, scaling and symmetry changes. Additionally, the authors proposed Maximum Margin Criterion (MMC) method to reduce dimensionality. Finally, Nearest Neighbor Classifier (NNC) is used for classification. Experiments that are also verified by our own research shows that MDM can effectively describe the leaf shape.

Another plant identification system called Leafsnap is proposed in [26]. This paper details a complete system from acquisition to presenting the results. Firstly, this system checks if the given image is a leaf. The second step is the preprocessing, where segmentation, background and stalk removal is performed. Then the curvature-based shape features are extracted to be used in Nearest Neighbors retrieval. The results show that this method is a substantial improvement over IDCS. However, the speed of this system is a limiting factor as it takes 5.4 seconds for a single leaf classification.

A method that uses semantic information and active contour segmentation is proposed in [20]. This method proposes polygonal model parameters used for segmentation, base and tip model and Curvature Scale Histogram (HCS) as features. Additionally, this research details polygon fitting method for segmentation, which is a crucial step in HCS. The authors demonstrated the performance of their systems over Pl@ntLeaves dataset [29].

In [21], authors propose a novel shape descriptor - Multi-level Curve Segment Measures (MLCSM) - for the purpose of leaf retrieval. MLCSM is rotation and scale invariant shape descriptor that can capture statistical characteristics such as curve bending, convexity and concavity of curve segments. This operation is performed over different curve lengths. The results demonstrated in [21] show that MLCSM has the highest performance and comparable computational complexity compared to the previous approaches.

A shape-based multiscale method for leaf classification is proposed in [9]. This method explores different representations of triangular parameters on multiple scales. Multi Probe Locality Sensitive Hashing technique is used for shape matching. This shape matching is then turned into a voting system to rank classes. Authors have demonstrated effectiveness of these methods over several databases as both classification and image retrieval method.

In [8], authors propose the use of image moments and Fourier Descriptors. Authors have optimized the weights of these features to improve accuracy of the system. This research also introduces a new database that contains woody species in Central Europe. The results presented in that paper shows that this combined method has higher or comparable performance over different databases.

A label propagation based method with novel weights and feature reduction named as Supervised Locality Projection Analysis (SLPA) is proposed in [12]. This algorithm is based on Warshall algorithm and calculates k-nearest-neighborhood-graph to be used in decision process. The researchers show that SLPA outperforms other general manifold based learning algorithms in terms of accuracy.

Authors in [22] propose two new spline embedding based methods called orthogonal locally discriminant spline embedding (OLDSE-I and OLDSE-II). These methods also employ maximum margin criterion method (MMC) which is used to reduce dimensionality.

Authors in [11] propose the use of features extracted from leaf veins and areoles to classify types of legume plants. In this research, three methods, namely support vector machines, penalized discriminant analysis and random forests methods are analyzed. The results demonstrate that random forests method reaches up to 90% accuracy.

A method that relies on leaf margins as descriptors is proposed in [10]. This method, named as Leaf Margin Sequences, extracts a map of margin peaks and pits along the leaf margin and represents them as sequences. In this research, Nearest Neighbor classifier is used with edit distances of these sequences. Authors have combined these sequences with simple geometric features to show that margin sequences descriptor has a comparable performance.

In [7], authors have proposed a plant recognition system that combines different features representing shape, texture and color of plants. Additionally, this research contains both automated and semi-automatic segmentation of plant and single leaf images. The proposed method presented in [7] has the highest score in ImageCLEF'2012 plant identification campaign in both human assisted and automatic categories.

A texture based leaf classification method is proposed in [5]. This method combines two Fourier Transformed Completed LBP descriptors, one extracted from leaf surface, the other from the edge. Even though the name suggests texture-based classification, this method actually combines shape and texture information. Authors claim that by using SVM, this method reaches to 99% score on all databases it is tried on.

Authors in [4] propose a method that contains a novel result representation. This method resembles a content based information retrieval system, however, it returns variable number of results instead of fixed number of results. This removes possibly unrelated classes from the results list. Authors use multiple descriptors with multiple classifiers combined using confidence sets.

In the literature, there are many different methods used for leaf classification. Most of these methods concentrate on shape based features, however, there are methods that use texture features as well. Diversity in classifiers is limited in leaf classification; and many methods either use KNN or SVM. Over the years, the results of leaf classification are improved significantly. Moreover, with the increasing interest in leaf classification, additional databases with more classes and samples become available.

Chapter 3

BACKGROUND INFORMATION

Leaf classification is a multidisciplinary field that requires knowledge in botany, image processing, computer vision, color models, and pattern recognition. In the following sections, required background information on these subjects are presented.

3.1 Leaf Anatomy and Morphology

Leaves in the nature are analyzed to be able to create descriptors for classification. Additionally, the way botanists identify plants are also studied. In this section, the terminology, anatomy and morphology of plant leaves are discussed. The organs of a simple leaf are displayed in Figure 3.1. The following is the names and descriptions of leaf parts:

- Midrib: is the first vein level, which is directly connected to the stem
- Blade (Lamina): leaf surface
- Tip: top section of the blade
- Base: bottom section of the leaf blade
- Margin: The edge of a leaf
- Lobe: A rounded or pointed section of a leaf

There are many different leaf shapes. These are categorized by the botanists for identification. An ideal leaf recognition system should be able to differentiate between leaf shapes. It is also important to know basic leaf shapes to understand the complex-

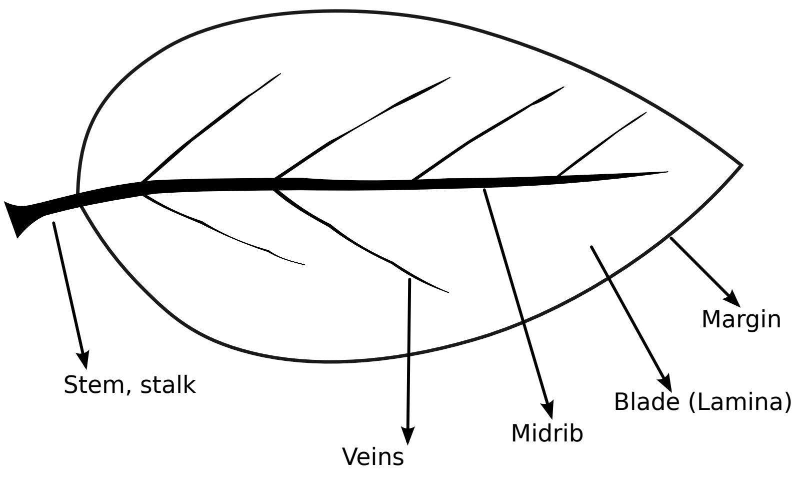


Figure 3.1: Leaf organs

ity of the problem. The following is an incomplete list of leaf shapes which are also illustrated in Figure 3.2:

a. Acicular, pinale leaf: needle shaped

b. Orbicular: circular

c. Elliptic: oval shaped

d. Rhomboid: diamond shaped

e. Acuminate: with a long point

f. Flabellate: fan shaped

g. Ovate: egg shaped, wide base

h. Palmate: with smoothly joining lobes

i. Cordate: heart shaped, growing from cleft

j. Linear: parallel margins

k. Obcordate: heart shaped, steam at point

l. Spatulate: spoon shaped

m. Cuneate: wedge shaped

n. Lanceolate: both sides are pointed

o. Obovate: egg shaped, narrow base

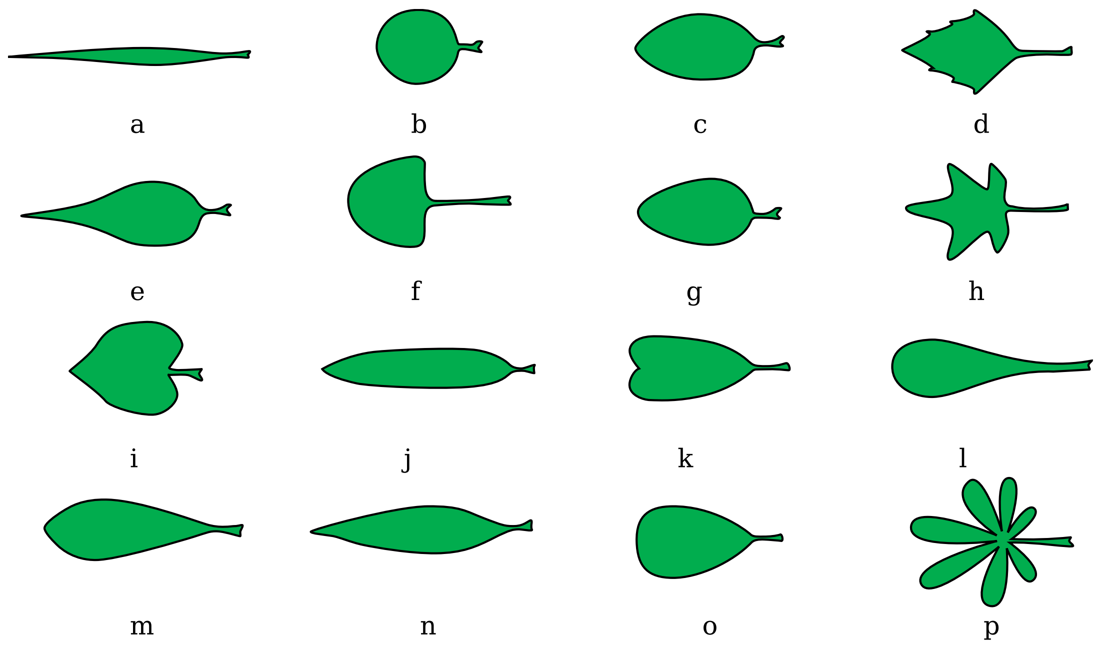


Figure 3.2: Leaf shapes

p. Digitate: lobes extending from the center

Another distinguishing feature of leaves is margins. Figure 3.3 illustrates different leaf margins. Names of these margins are listed below:

a. Entire: with no teeth

b. Serrate: teeth pointing forwards

c. Dendate: triangular teeth

d. Lobate: contains lobe-like teeth

e. Denticulate: small triangular teeth

f. Crenate: round teeth

g. Undulate: wavy

h. Serrulate: tiny serrate teeth

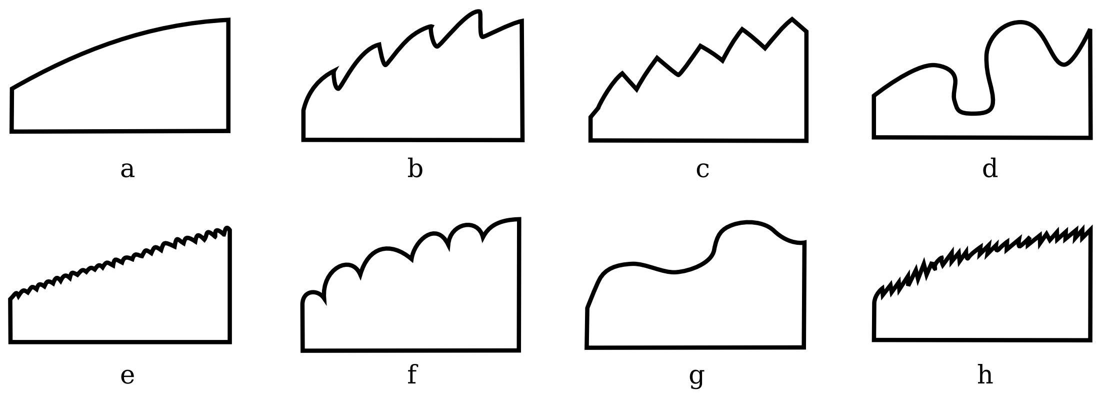


Figure 3.3: Leaf margins

Leaf texture is generally affected by the venation pattern and the visibility of its veins. The following is the list of common venation patterns which are also illustrated in Figure 3.4:

a. Arcuate: veins bending towards the tip

b. Cross-venulate: smaller veins connecting primary ones

c. Dichotomous: symmetrically branching veins

d. Pinnate: veins that are growing opposite directions along the midrib

e. Longitudinal: almost straight and intersecting veins

f. Palmate: veins that are growing out from a single point

g. Parallel: veins that grows parallel to each other

h. Reticulate: small veins forming a network

3.2 Segmentation

The iterative segmentation algorithm proposed by [1] is employed for some datasets that are discussed in the results section. This method works iteratively to find an ideal threshold for segmentation. At every step, a new threshold, which is the average of mean background and foreground intensities, is used. Initial threshold is set to the average of minimum and maximum intensity in the image. The following steps are necessary to perform this operation on image I:

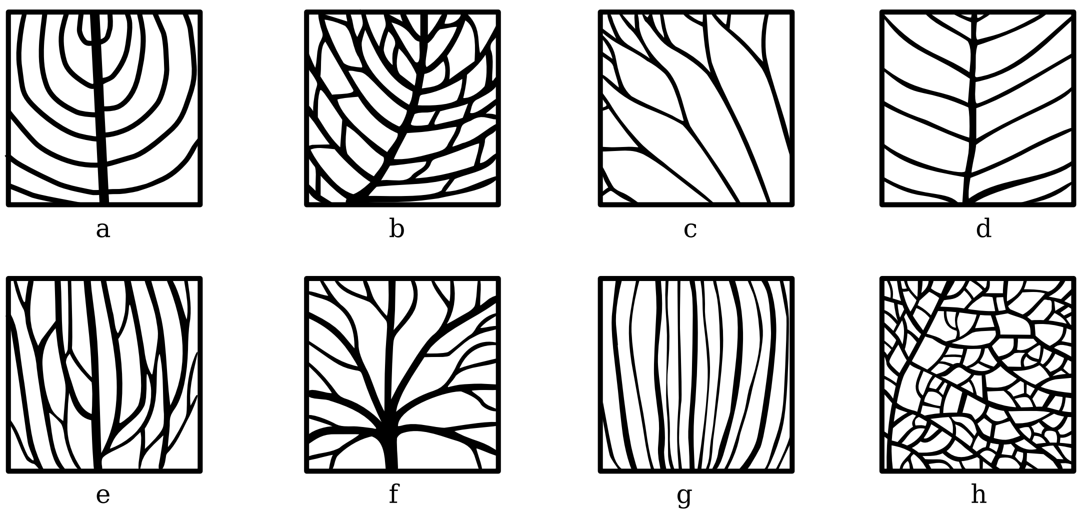


Figure 3.4: Venation patterns

Step 1. $G_{max} = \mathbf{max}\left[I_{x,y}\right], \; G_{min} = \mathbf{min}\left[I_{x,y}\right]$

Step 2. $k \leftarrow 0$

Step 3. $T_k = (G_{max} + G_{min})/2$

Step 4. Segment the image using threshold T_k to determine the object at iteration k

Step 5. $G_{fore} = \mathbb{E}\left[I_{x,y}, (x,y) \in object\right]$

Step 6. $G_{back} = \mathbb{E}\left[I_{x,y}, (x,y) \notin object\right]$

Step 7. $k \leftarrow k+1$

Step 8. $T_k = (G_{fore} + G_{back})/2$

Step 9. if $T_k \neq T_{k-1}$ then return to *Step* 4

3.3 Contour Extraction

Edges are important features. In classical approaches, edges are used as disjoint points in the image. However, it is more informative and useful to use edge points that have connections to the points coming before and after them. The term contour is used for list of points that are morphologically sequential. It is possible to apply many operations over contour points that cannot be applied to disjoint edges. For instance, derivative of contour points can be calculated. In regular images, extracting contours are not trivial, however, in a properly segmented image, contour information can be extracted using simple algorithms. In this research, we use Pavlidis' Contour Extraction [30]. This algorithm executes in linear time in respect to the edge pixels, works in 8-neighborhood, and can process single pixel thickness. We selected this algorithm over classical contour extraction algorithms like Moore's algorithm due to these advantages.

Pavlidis' Contour Extraction works over binary images to produce sequential contour data. Ideally, the starting point should be the bottom-most point of the region of interest (leaf surface in our system). The algorithm stops when the start point is reached for the second time. During its execution, this method jumps from a pixel to another one in a clockwise fashion adding them to the sequence of edges. Pavlidis' algorithm is presented in Algorithm 3.1.

3.4 Geometric Features

Geometric features are widely used in shape based descriptors. They are often combined with other descriptors, such as image moments, as they are too broad to capture details in the shapes. All geometric features are scale and rotation invariant. In this section, the frequently used geometric features are listed. These features are grouped into categories. Some geometric features defined here are not used in our proposed method. Some features are eliminated as they contain information that is available in other features.

Geometric features are calculated over segmented image. However, some of the features described here can be calculated over contours. Since contour extraction is a fast process and reduces the number of pixels to be processed to roughly the square root of the original, calculating these features using contour data should be preferred.

Algorithm 3.1: Pavlidis' Algorithm. x starts from left, y starts from bottom

```
procedure PavlidisContour(I)
    ▷ I is binary image
    $direction = north$
    for $y = 0$ to heightof($I$) do
        for $x = 0$ to widthof($I$) do
            if $I_{x,y}$ then
                $first = (x, y)$
                goto found
            end if
        end for
    end for
found:

    $location = first$
    do
        if targetvalue($I$, $location$, $direction$, frontleft) then
            $location$ = targetlocation($location$, $direction$, frontleft)
            $direction$ = rotate($direction$, left)
            push $location$ to $path$
        else if targetvalue($I$, $location$, $direction$, front) then
            $location$ = targetlocation($location$, $direction$, front)
            push $location$ to $path$
        else if targetvalue($I$, $location$, $direction$, frontright) then
            $location$ = targetlocation($location$, $direction$, frontright)
            push $location$ to $path$
        else
            $direction$ = rotate($direction$, right)
        end if
    while $location \neq first$
    return $path$
end procedure
```

3.4.1 Simple Geometric Features

Simple features are aspect ratio (*AR*), perimeter ratio (*PR*) and rectangularity (*RG*). *AR* represents how long and thin the leaf is. Generally higher *PR* denotes a complex leaf. However, thinner leaves have higher *PR*. *RG* measures how well the Minimum Bounding Rectangle (MBR) fits to the leaf. For instance a linear leaf has a higher *RG* compared to a palmate leaf. These features require MBR, Leaf Area and Leaf Perimeter. MBR is the smallest rectangle that can contain the entire leaf. Long edge of the MBR is called maximum distance (D_{max}), short edge is called orthogonal distance (D_{ort}). This rectangle can be calculated by finding furthest points in the segmented image. Leaf area is the number of points that are determined to be in the leaf surface by segmentation operation. Leaf perimeter is the number of contour points that are extracted. Using these geometric properties, simple geometric features can be calculated as follows:

$$\begin{aligned} AR &= D_{max}/D_{ort} \\ PR &= Perimeter_{leaf}/\sqrt{Area_{leaf}} \\ RG &= Area_{leaf}/Area_{MBR} \end{aligned} \tag{3.1}$$

3.4.2 Convexity Features

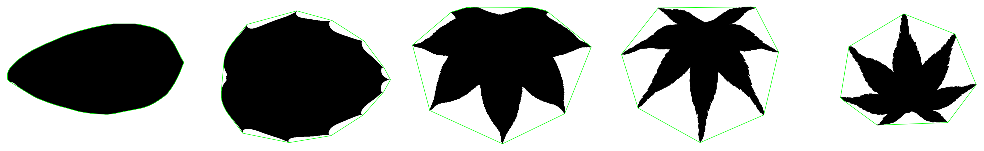
Convexity features measure the complexity of the leaf. These are Area Convexity (*AC*) and Perimeter Convexity (*PC*). The value of both of these features are 1 for a simple, fully convex leaf. As the leaf gets more complicated, these values will deviate from this point. This effect is illustrated in Figure 3.5.

Convexity features require the calculation of the convex hull of the leaf. There are different methods to calculate convex hull. The algorithm we use in this thesis is explained in Section 4.1.4. The formulas to calculate convexity parameters are as follows:

$$\begin{aligned} AC &= Area_{leaf}/Area_{convex} \\ PC &= Perimeter_{leaf}/Perimeter_{convex} \end{aligned} \tag{3.2}$$

3.4.3 Circular Features

Circular features represent circular structure of the leaf. These are Sphericity (*SP*) and Circularity (*CC*). A low *SP* shows that the leaf is circular in shape and high *CC* shows a leaf is circular. *SP* requires the radius of the largest circle (r_i) that can fit into the leaf and the radius of the smallest circle that can hold the leaf (D_{max}). The area and the



(a) a = 1.00 p = 0.92 (b) a = 0.92 p = 1.00 (c) a = 0.68 p = 1.28 (d) a = 0.47 p = 1.70 (e) a = 0.54 p = 1.68

Figure 3.5: Four different types of leaves and their area, a and perimeter, p complexities. Leaves in (d) and (e) belong to the same class.

convex hull of the leaf are required to calculate circular features.

$$\begin{aligned} SP &= r_i / D_{max} \\ CC &= Area_{leaf} / Perimeter^2_{convex} \end{aligned} \tag{3.3}$$

3.5 Moment Invariants

Moment invariants are used in object recognition. They are first proposed by [31] and later expanded and analyzed in [32,33]. They are calculated using image moments.

Image moments describe statistical information about images. Raw moments are basically weighted average of the image pixel intensities. Moment of order i, j, $(M_{i,j})$ for a regular 2-dimensional, discrete image, I, is defined as follows:

$$M_{i,j} = \sum_x \sum_y x^i y^j I(x,y) \tag{3.4}$$

Although it is meaningful for some contexts, such as in determining the centroid or area of the image, raw moments are not very useful as they are affected by translation, rotation and scale. Central moments are defined to achieve translation invariance. The basic idea is to use x and y distance from the centroid, instead of a fixed point. Central moment μ_{pq} is calculated as follows:

$$\begin{gathered} \bar{x} = \frac{M_{10}}{M_{00}}, \ \bar{y} = \frac{M_{01}}{M_{00}} \\ \mu_{pq} = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q I(x,y) \end{gathered} \tag{3.5}$$

Scale invariance can be achieved by dividing a central moment to the scaled area (M_{00}). The following equation calculates translation and scale invariant central moment η_{pq}:

$$\eta_{pq} = \frac{\mu_{pq}}{M_{00}^{(1+\frac{p+q}{2})}} \tag{3.6}$$

It is also possible to derive rotation invariant moments. In this thesis, we use 8 rotation invariant moments up to order 3. First 7 of these moments are called Hu's Moment Invariants. The equations that are used to calculate rotation invariant moments are given in (3.7).

$$\begin{aligned}
\sigma_1 &= \eta_{20} + \eta_{02} \\
\sigma_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
\sigma_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
\sigma_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
\sigma_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\
&\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
\sigma_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
&\quad + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
\sigma_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\
&\quad - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
\sigma_8 &= \eta_{11}[(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2] \\
&\quad - (\eta_{20} - \eta_{02})(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21})
\end{aligned} \tag{3.7}$$

3.6 Multiscale Distance Matrix

Multiscale Distance Matrix (MDM) is a feature descriptor proposed by [2]. It is specifically designed for leaf classification. MDM is extracted from the contour and it is scale and rotation invariant. MDM uses a number of equidistant points from the contour which are called nodes. These points can be determined from the edge pixels since the ordering is not necessary. However, in that case, the distance between the nodes might not be uniform.

MDM descriptor is a modified form of distance matrix between these nodes. The number of features generated by MDM grows quadratically with the number of nodes. The following equation can be used to calculate how many features that an n node MDM has:

$$N_{features} = n\lceil n/2 \rceil \tag{3.8}$$

Building MDM requires several steps. The first step is to create distance matrix between each node. Various distance metrics can be employed for this task. Authors in [2] recommend the use of Inner Distance proposed by [34]. However, similar to the findings in [2], we found that the distance metric does not change accuracy significantly and using Inner Distance in some cases decreases the accuracy. Since the overall improvement is negligible, we adopted Euclidean Distance in our research. The following equation can be used to construct distance matrix DM using nodes N:

$$DM_{ij} = \sqrt{({}_xN_j - {}_xN_i)^2 + ({}_yN_j - {}_yN_i)^2} \tag{3.9}$$

In the next step, distance matrix is shifted. Together with the next step, this step achieves rotation invariance by removing any location dependent data. Shifting is performed per column. Every column is shifted upwards by the zero based index of that column. This operation moves all diagonal elements, which are 0, to the first row. The following formula can be used to calculate shifted distance matrix SDM, with indices that are 0-based:

$$SDM_{ij} = DM_{(i+j \bmod n)j} \tag{3.10}$$

where n is the number of nodes. After shift operation, first and the last $\lfloor \frac{n}{2} - 1 \rfloor$ rows are discarded. The first row is discarded as it only contains zeroes. The last $\lfloor \frac{n}{2} - 1 \rfloor$ rows are the duplicates of previous rows.

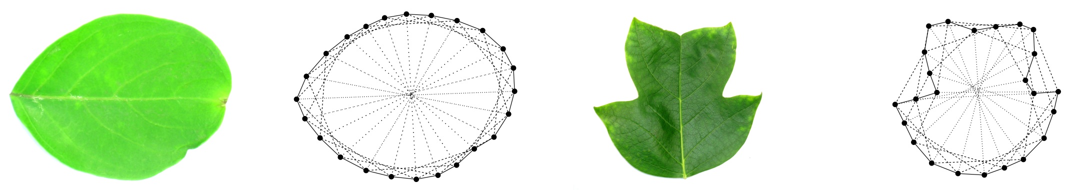


Figure 3.6: (a) and (c) are original images. (b) and (d) are on-the-image representation of the first (solid lines), fourth (dashed lines), and twelfth (dotted lines) rows of shifted distance matrix. SDMs are populated by lengths of these lines. MDM nodes are marked around the edge with discs.

After shift operation, each row of *SDM* represents a different node distance. For instance, the values on the second row are the distances between the nodes and their neighbors. Every column represents a node. Visual representation of SDM is shown in Figure 3.6. Since the order of nodes depends on the orientation of the shape, the next step is to eliminate this information. This is achieved by simply sorting every row.

Since sorted *SDM* contains distances, it is scale dependent. This descriptor is made scale invariant by dividing it to its average. In [2], different methods for scale independence are given, however, the best method both by intuitive and empirical approaches is using matrix average for scale factor. The following equation is the final step in calculating MDM descriptor:

$$MDM_{ij} = \frac{SDM_{ij}}{\mathbb{E}\left[SDM\right]} \tag{3.11}$$

where $\mathbb{E}$ is the expected value function. MDM descriptors of different leaves are presented in Figure 3.7.

3.7 Local Binary Patterns

Local Binary Patterns (LBP) is an illumination invariant transformation. LBP captures local characteristics of textures and proves itself to be a useful descriptor in Biometrics. The result of this transformation on various images is shown in Figure 3.9. As the simplest approach, LBP histogram is used as the texture descriptor. However, LBP is

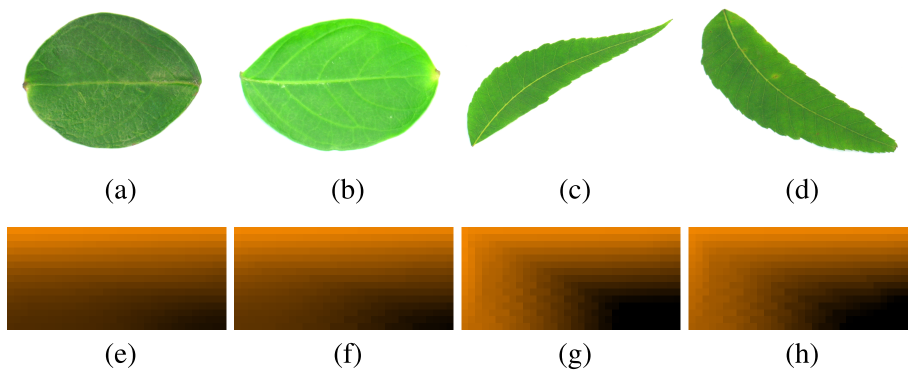


Figure 3.7: Two samples for two different leaves and their MDM representation. (MDM matrix is represented as color coded values where darkness of a cell increases with the value of that cell.)

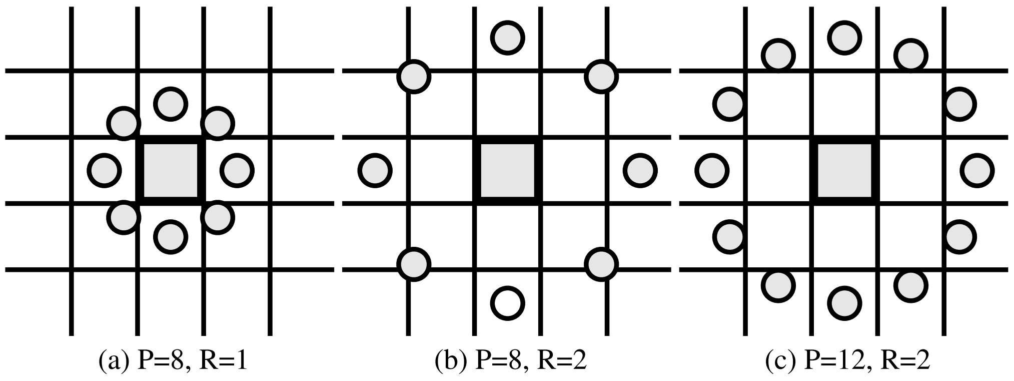


Figure 3.8: Calculation of LBP using different parameters

an orientation dependent transformation. There are many LBP variants in the literature and some of these methods are rotation invariant.

LBP transformation simply compares the intensities of the neighbors around a central point to the center. Boolean results from these comparisons are then used as a binary number where each digit is the result of a comparison. This number is then assigned as the new intensity. Every neighbor is sampled around the center. The number of neighboring points, P, determines the bit depth of the transformed image. Additionally, LBP requires another parameter, which is the distance from the center to the neighbors. This parameter is denoted as R. Since a digital image is discrete, the intensity values at the

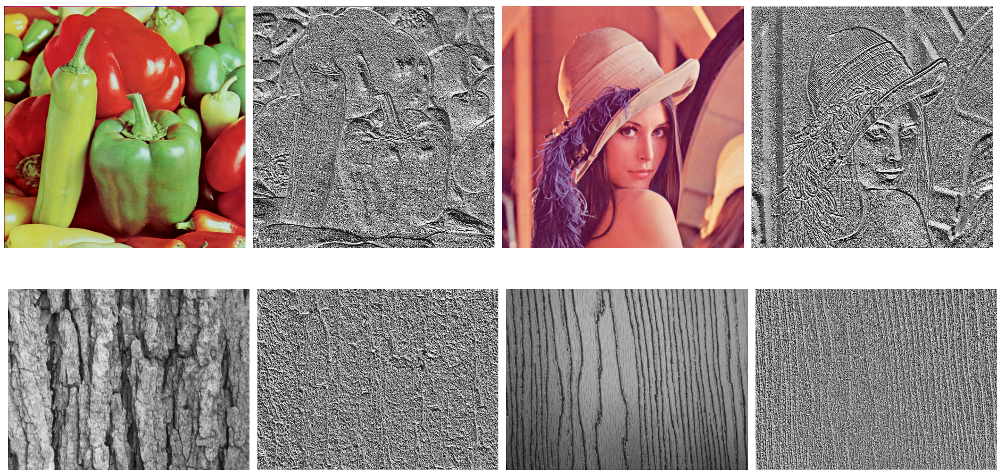


Figure 3.9: Various images and their LBP transformations.

points are determined by interpolation. Generally linear interpolation is employed for this task. Figure 3.8 illustrates the processing of different parameters. The following are the formulas that are necessary to calculate LBP transformation:

$$
\begin{aligned}
s(x) &= \begin{cases} 1, & x \geq 0 \\ 0, & \text{otherwise} \end{cases} \\
LBP_{PR} &= \sum_{p=0}^{P} s(g_c - g_p) 2^p
\end{aligned}
\tag{3.12}
$$

where g_c is the intensity of center pixel and g_p is the intensity of neighbor p. LBP transformations of sample images are displayed in Figure 3.9.

3.8 Color Models

Digital images are generally represented in Standard Red-Green-Blue (sRGB) color space. However, this color space is not very suitable for computer vision applications. It is more useful to represent color as hue, saturation and lightness. This separation allows computer to differentiate an object regarding to its color tone (hue and saturation) without being affected by the illumination, which might vary due to many factors. HSV and HSL color models separate hue, saturation, value/lightness channels. However, these color models are defined over sRGB color space (generally mistaken as RGB) and for simplicity in calculation, hue and saturation channels are not fully separated from value/lightness channels. HSV and HSL color spaces are developed for designers who can handle value shifts manually. However, in computer vision applications, this issue causes a great problem.

CIE (International Commission on Illumination) defines color spaces that are independent from devices. Most of these color spaces map the real world color more accurately than the commonly used color systems such as sRGB. The first color spaces defined were CIE 1931 RGB and XYZ color spaces. Both of these color spaces are modeled after human vision. In 1976, CIE released another color space, CIE-LAB. CIE-LAB color system covers entire color range visible to human eye and it is designed to be device independent. The components of CIE-LAB color space are lightness (L), red/green (a) and yellow/blue (b). Commonly L channel starts from 0 (black) to 100 (white). a and b channels starts from -100 to 100. If both a and b are 0, the resultant color is gray. In addition to standard representation, CIE-LAB color system can be expressed in cylindrical form which is called CIE-LCH. Similar to CIE-LAB, L channel is lightness. C is the chromacity channel which represents the saturation of the colors. H is the hue channel representing the hue of the color. Hue channel in CIE-LCH is circular and represented as an angle. Similar to HSV and HSL color spaces, 0 hue denotes red where 120 is green and 240 is blue. Unlike HSV and HSL color spaces, hue channel of CIE-LCH has a fixed lightness. This trait can be observed from Figure 3.10. It must be noted that CIE-LCH spectrum covers much larger spectrum compared to HSV and HSL color spaces. In fact, CIE-LAB color space is much larger than visible spectrum. RGB spectrum in CIE-LAB color space is displayed in Figure 3.11.

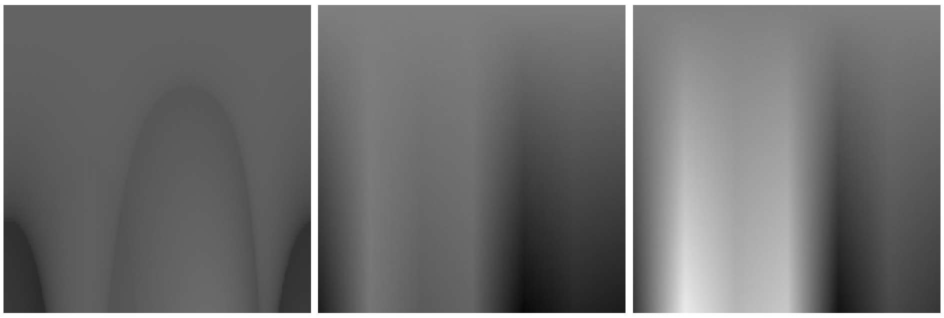

(a) (b) (c)

Figure 3.10: Perceived lightness of hue and saturation/chroma channels of (a) LCH, (b) HSV, and (c) HSL color systems.

Converting from sRGB color space to CIE-LCH color space is not trivial and requires many steps. Equation (3.13) shows the necessary calculations to convert a color represented with R, G, and B components to L, C and H components. These transformations use fixed sRGB gamma value of 2.2 for simplicity. sRGB gamma changes non-linearly from 1.0 to 2.3 with average being 2.2.

$$
\begin{aligned}
r &= R^{2.2}\ ,\ g = G^{2.2}\ ,\ b = B^{2.2} \\
X &= 0.4338r + 0.3762g + 0.1899b \\
Y &= 0.2126r + 0.7152g + 0.0722b \\
Z &= 0.0177r + 0.1095g + 0.8728b \\
x &= \sqrt[3]{X}\ ,\ y = \sqrt[3]{Y}\ ,\ z = \sqrt[3]{Z} \\
L &= y\ ,\ a = 5(x-y)\ ,\ b = 2(y-z) \\
C &= \sqrt{a^2+b^2}\ ,\ H = \frac{\arctan(\frac{b}{a})}{2\pi}
\end{aligned}
\tag{3.13}
$$

3.9 Linear Discriminant Classifier

Bayes Minimum Error Decision Rule lowers the probability of error using known priories. In a known environment, classification according to this rule results in the best possible accuracy. However, this system requires the probability distribution function for the data points. The best estimate for this task in an unknown environment is the Normal Distribution. Linear Discriminant Classifier (LDC) is one of the classifiers de-

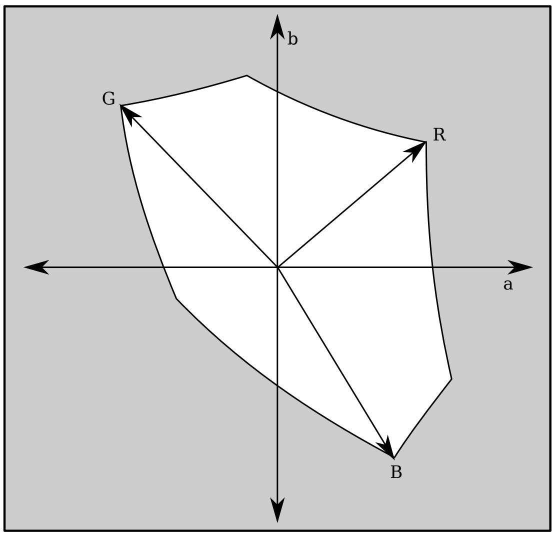


Figure 3.11: sRGB color space mapped on to CIE-LAB color space.

rived from Bayes Decision Rule using normal distribution. In a two class classification system, Bayes Decision Rule is given below:

$$\mathbf{maxarg}_i P(\omega_i|\bar{x}) \tag{3.14}$$

where P is the probability function, ω_i is the class i and $\bar{x}$ is the test sample.

Using Bayes Rule, this decision rule can be transformed into the following form:

$$\mathbf{maxarg}_i P(\bar{x}|\omega_i)P(\omega_i) \tag{3.15}$$

If we use the Multivariate Normal Distribution function which is given below

$$P(\bar{x}|\omega_i) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_i|^{\frac{1}{2}}} e^{-\frac{1}{2}(\bar{x}-\bar{\mu}_i)^T \Sigma_i^{-1}(\bar{x}-\bar{\mu}_i)} \tag{3.16}$$

where $\bar{\mu}_i$ is the expected value for class i and Σ_i is the covariance matrix of the samples of class i, the resultant decision rule is Quadratic Discriminant Classifier (QDC). It must be noted that the logarithm of the decision function is used. As logarithm is a monotonically increasing function, the decision is not affected. This transformation is performed to simplify the equation. The support for class i, g_i, using QDC can be calculated as follows:

$$\begin{aligned} g_i =& -\frac{1}{2}(\bar{x}-\bar{\mu}_i)^T\Sigma_i^{-1}(\bar{x}-\bar{\mu}_i)+(\Sigma_i^{-1}\bar{\mu}_i)^T\bar{x} \\ & -\frac{1}{2}\bar{\mu}_i{}^T\Sigma_i^{-1}\bar{\mu}_i-\frac{1}{2}ln|\Sigma_i|+lnP(\omega_i) \end{aligned} \tag{3.17}$$

LDC is closely related to Quadratic Discriminant Classifier. The main difference of LDC and QDC is that LDC assumes that the covariance matrices for all classes are the same. Although this assumption does not hold in real life scenarios, common covariance for all classes can be calculated and used as follows:

$$\Sigma = \frac{1}{N}\sum_{i=1}^{C} N_i\Sigma_i \tag{3.18}$$

where Σ is common covariance, N is the number of training samples, N_i is the number of training samples for class i, Σ_i is the covariance of class i. If the number of training samples is less than the number of dimensions, Σ will be a singular matrix.

After determining the common covariance matrix, LDC should be trained separately for each class. Training yields to a constant c and a vector $\bar{w}$ for every class. The following formula is used to calculate c and $\bar{w}$ for class i:

$$\begin{aligned} \bar{w}_i =& \Sigma^{-1}\bar{\mu}_i \\ c_i =& \ln P_i - \frac{1}{2}\bar{\mu}_i^T\Sigma^{-1}\bar{\mu}_i \end{aligned} \tag{3.19}$$

where $\bar{\mu}_i$ is the mean of class i, P_i is the probability of class i. Matrix inversion in this step will fail if the covariance matrix calculated in the previous step is singular.

While classifying a sample, support for each class is calculated separately. The class with the highest support is declared as the classification result. Support for class i is calculated as follows:

$$g_i(\bar{x}) = \bar{w}_i^T\bar{x}+c_i \tag{3.20}$$

Therefore, LDC has a lower or similar computational complexity compared to other classification methods. Additionally, LDC assigns separate weights to features for every class which allows it to ignore features that have no meaning for some classes. This trait is very important in cases where combination of multiple descriptors are used. As an improvement over QDC, LDC requires more total training samples than the number of dimensions, whereas QDC requires more training samples per class. Therefore, it is possible to use LDC where QDC cannot be used.

3.10 Technologies Used

In this thesis, we use C++ programming language and Eigen and Gorgon Libraries to implement the proposed methods and the methods that are used for comparison. These technologies are explained in the following subsection in detail.

3.10.1 C++

C++ is a multi-paradigm programming language that is designed to provide high-level functionality without sacrificing speed. There are many high level libraries, such as Eigen Library, programmed in C++ to provide additional functionality. C++ template system offers type security to typeless constructs such as vector class. Unlike generics in Java and C#, templates in C++ do not incur any cost in the run-time, making it suitable for applications that require many calculations. Additionally, silent type conversions allow library programmers to perform on-the-fly optimizations such as lazy evaluation.

C++11 standard brings more functionality to the language. For instance, lambda functions allow programmers to create functions inline, making standard library algorithms much easier to use. Automatic type deduction with the help of **auto** keyword speeds up programming tasks.

Many programmers believe that lower level languages are faster. However, in regular programming, C++ outperforms C up to %30 in some cases. There are two main reasons behind it. The first reason is that the compiler can produce faster running code when the program is more expressive. However, generally expressive programming incurs additional costs. In C++, additional functionality are all compiled into a compact form, through compiler steps such as inlining, redundant variable and code elimination and therefore, does not have significant additional cost. The second reason is the improved library functions. These functions are well-optimized and provide basic functionality that is either missing or very slow in C.

Standard Template Library (STL) is designed to provide extra functionality with minimum additional cost. Generally, the additional cost is using a little more memory but the provided extra functionality, and in some cases, speedup is significant. For instance, using an STL vector is much faster than using a C array and expanding it as necessary. The reason behind it is the allocation setup. When allocation is required, a vector object increases its size more than requested. Heuristics are employed to determine ideal increment step. Since memory allocation is a very slow process, this trait improves the speed. Only additional cost of using a vector is an extra integer defined in vector class. Additionally, using STL vector is much safer as in debugging mode, it checks the array boundaries. Additionally, STL is a header-only library. This means that all STL functions can be inlined. Inlining removes the function call by performing operations on the caller, hence, providing new possibilities for performance optimization.

3.10.2 Eigen Library

Eigen is a matrix library for C++ that supports fixed-size, variable-size and space matrices of any numeric type. Eigen uses templates and lazy evaluation to speed up operations. Lazy evaluation allows multiple matrix operations to be performed together, removing the need for temporary matrices and in some cases, removing unnecessary steps from the whole operation. Eigen Library provides operator overloads for matrix and vector operations, providing an intuitive programming for linear algebra. Additionally, Eigen Library supports explicit vectorization for different instruction sets, benefiting from the parallel execution facilities of the respective platforms. Eigen Library is well documented and tested. Every function is explained in detail and there are many examples. Finally, it has a large development team (currently 118 contributors) and a liberal license allows it to be used in any kind of project.

3.10.3 Gorgon Widgets

Gorgon Widgets is a cross-platform user interface library distributed with Gorgon Game Engine. Gorgon Widgets provides easy-to-use and functional user interface components (widgets). This library contains templated widgets such as NumberBox, so that they can be used with different data types. Additionally, this library provides support to read and write image files, which is required for computer vision applications.

Chapter 4

PROPOSED METHODS

We propose two leaf classification methods in this thesis. These methods include new approaches such as Sorted LBP, application of LDC for leaf classification and several feature types combined to improve the classification accuracy. First of all, geometric features are used for the classification of plant leaves. Then, geometric, shape, texture, and color features are combined to perform leaf classification. All the steps involved in these proposed methods, including preprocessing, feature extraction, and classification are explained in the following subsections in detail. Additionally, the order and dependency of these operations are illustrated in Figure 4.1.

4.1 Preprocessing

Preprocessing step includes segmentation, noise removal, contour smoothing, convex hull extraction, corner detection, texture mask, and stalk removal operations. These operations are described below with some examples.

4.1.1 Segmentation

Geometric features require binary images to work. Therefore, the color images that are acquired from various sources, should be segmented. Although there are many segmentation methods, we developed a simple segmentation method specifically targeted to leaf image segmentation. This method performs better compared to a more complicated and generic method described in Section 3.2 in two out of three datasets we use. The main idea behind this segmentation method is that the blue color channel

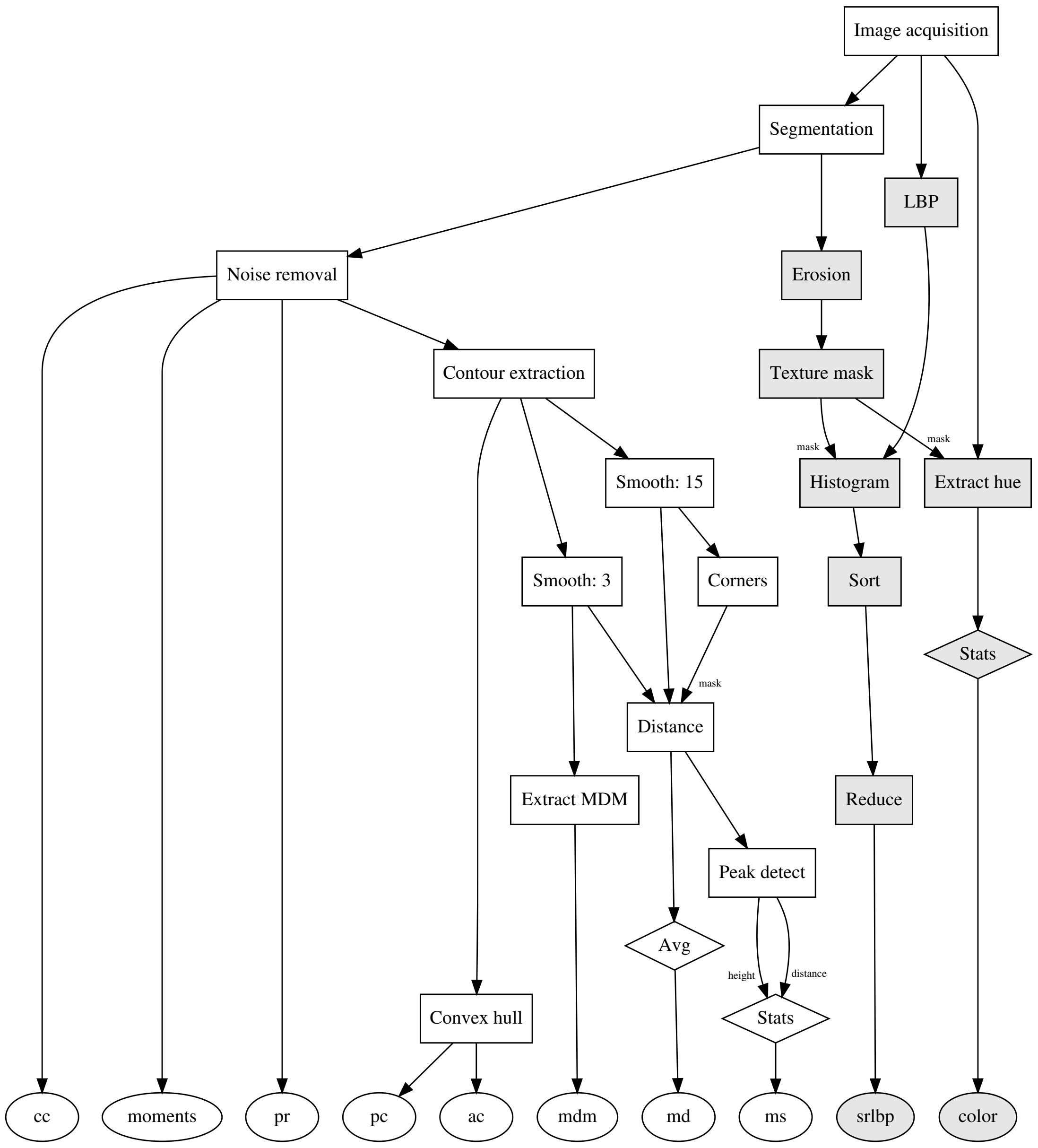


Figure 4.1: Flow chart of preprocessing and feature extraction steps. Nodes with gray backgrounds are only used in GTCLC

has the most variance in a leaf image if the background of the image is white. This is evident from Figure 4.2. Using this fact, we simply segmente the image from the mean intensity of the blue channel. Any pixel having an intensity that is lower than this threshold is assumed to be leaf surface. This segmentation method can be formulated as follows:

$$Leaf = \left\{(x,y),\ {}_{blue}I_{x,y} < \mathbb{E}\left[{}_{blue}I\right]\right\} \tag{4.1}$$

where, I is the input image and $\mathbb{E}$ is the expected value function.

The result of this segmentation method is presented in Figure 4.3. Even though this segmentation method performs admirably, there is still some noise in the result. Generally, this noise is caused by highlights or dirt particles on the acquisition device.

4.1.2 Noise Removal

It is possible to obtain a noisy segmentation due to various reasons. Geometric features are sensitive to noise. For instance, inscribed circle will be much smaller if there is a small patch within the leaf surface. We developed a simple noise removal method to fix this problem. The basic idea behind this is to iteratively analyze and correct connected components in the segmented image. Any connected component that is smaller than 20% of the entire leaf surface, is determined to be a noise patch, and its value is inverted, so that enclosing component will inherit it. Since there may be additional noise inside a noisy region, this operation is performed iteratively. In images with many noisy regions, this method might work slowly. However, it is possible to speed up this process by marking processed regions. This algorithm is illustrated in Algorithm 4.1 with the results shown in Figure 4.3.

4.1.3 Contour Smoothing

We employ contour smoothing both to reduce noise and to detect smaller changes along the leaf blade. Our smoothing operator works over a given window and performs one dimensional vector convolution. The kernel that is used consists of equal entries, causing a box smoothing. If the smoothing factor is s, the size of the smoothing kernel will be $2s+1$ and each entry in the kernel will be $\frac{1}{2s+1}$. After contour extraction, we perform a smoothing operation with a factor of 3. This is performed to cancel segmentation and quantization errors. Figure 4.4 shows the result of contour smoothing over two different contour segments. Contour smoothing with factor of 15 is used to remove the teeth of the leaf margin.

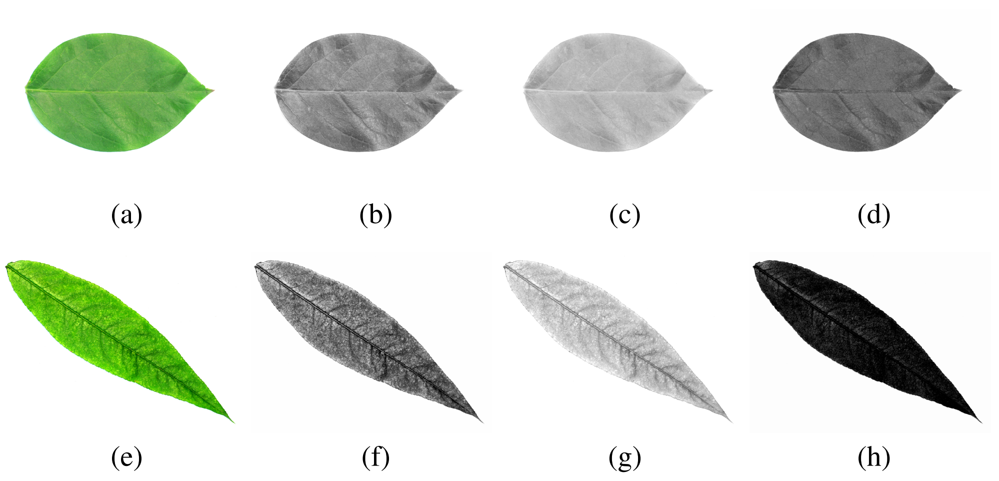


Figure 4.2: Color channels of two different leaf images. (a) and (e): all channels; (b) and (f): red channel; (c) and (g): green channel; (d) and (h): blue channel.

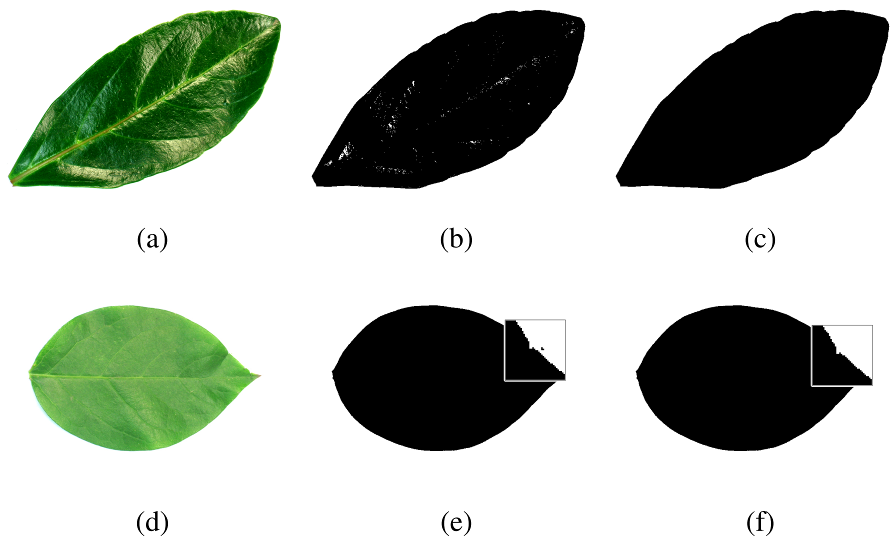


Figure 4.3: Segmentation results and noise removal. (a) and (d): original leaf; (b) and (e): segmented leaf; (c) and (f): after noise removal

Algorithm 4.1: Noise removal algorithm.

```
procedure NoiseRemove($I$, $limit$)
    ▷ $I$ is binary image
    $J$ is a new set of points
startover:
    for each pixel at $(x, y)$ do
        if $(x, y) \notin J$ then
            $open$ and $closed$ are stacks of 2D coordinates
            push $(x, y)$ to $open$ stack
            $current = I_{x,y}$

            ◇ Finding connected components
            while $open$ has items do
                $(x', y') =$ pop $open$
                push $(x', y')$ to $closed$ stack
                for $(xx, yy) \in I_{x',y'}^{4-neighborhood}$ do
                    if $(xx, yy) \notin J \wedge (xx, yy) \notin open \wedge$
                        $(xx, yy) \notin closed$ then
                        push $(xx, yy)$ to $open$ stack
                    end if
                end for
            end while

            if sizeof($closed$)$>$ $limit$ then
                append $closed$ to $J$
            else
                for $(x', y') \in closed$ do
                    $I_{x',y'} = \neg I_{x',y'}$
                end for
                goto startover
            end if
        end if
    end for
end procedure
```

4.1.4 Convex Hull

Convex hull is required for some of the geometric features. Although determining convex hull of a binary image is challenging, extracting this information from contour data is relatively simple. We developed a convex hull algorithm that works only on a fully connected contour data extracted in clockwise direction. This algorithm simply determines whether a point causes concave feature by simply analyzing its position

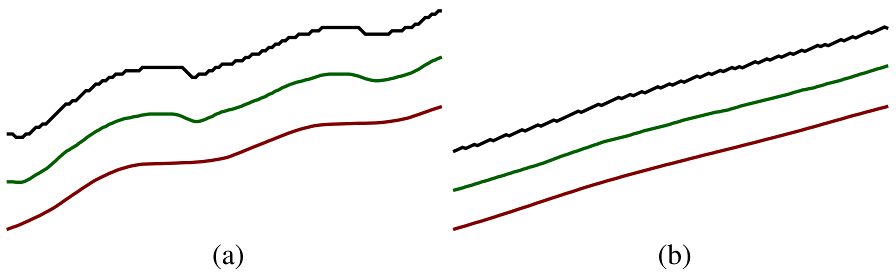


Figure 4.4: Contour smoothing operation on two leaf sections. (The lines on the top are the original contours, middle contours are smoothed with a factor of 3 and the bottom contours are smoothed with a factor of 15.)

in respect to the surrounding points. This method works in linear time and if the conditions are fully satisfied, it never fails. Algorithm 4.2 illustrates this method.

4.1.5 Corner Detection

Corner region detection is a necessary step to prevent margin features getting affected from the corners of the leaf. Corner regions are the contour segments that contain a leaf tip. Depending on the leaf type, corners may have different properties compared to the other segments of leaf blade. Therefore, these segments are ignored while calculating some of the features that are discussed in Section 4.2.1. This detection uses smooth contour information. We determined smoothing factor of 15 to be optimal in

Algorithm 4.2: Convex hull algorithm using contour information

```
procedure ConvexHull(L)
    ◇ L is the contour of the leaf
    $a = 0, b = 1, c = 2$
    while $c \not> Size_L$ do
        $m = \frac{L_{cy}-L_{ay}}{L_{cx}-L_{ax}}$
        $a = L_{cy} - mL_{cx}$
        if $mL_{bx} - L_{by} + a < 0$ then
            ◇ Removal of b causes c to point to the next contour
            remove $b$ from $L$
            $a = a-1, b = b-1, c = c-1$
        else
            $a = a+1, b = b+1, c = c+1$
        end if
    end while
end procedure
```

the datasets we use. This value depends on the size of the image, rather than the size of the leaf to be analyzed.

In order to determine corner regions, we use average arc angles of different distances. The algorithm determines the contour point that is *distance* before and *distance* after the current point to calculate an arc angle over a given distance. Using these points, the angle between the lines that passes from the previous point to the current point, and from the current point to the next point is calculated. Arc angles are averaged for the distances that start from 1 to $n/4$, where n is the number of contour points. It is possible to derive the maximum distance using heuristics, however, using a fixed ratio also results in predictable and useful corner regions. Therefore, we use the simpler technique both for efficiency and to avoid additional complexity.

The average arc angles are calculated and then a threshold is determined as follows:

$$arc_{threshold} = \frac{\min(arcs) + 3\mathbb{E}[arcs]}{4} \tag{4.2}$$

Setting threshold to this value reduces the amount of noise, while making sure to include all the meaningful corner regions. The average arc angles, their average and determined threshold are presented in Figure 4.5.

4.1.6 Texture Mask

Texture features around and outside the leaf margin are significantly different compared to the texture features inside the leaf. Additionally, the color on the edges could be slightly different because of non-perfect sampling since the edge pixels will contain color information both from leaf surface and from background. We developed a method to create a mask to only extract texture and color features that are strictly in-

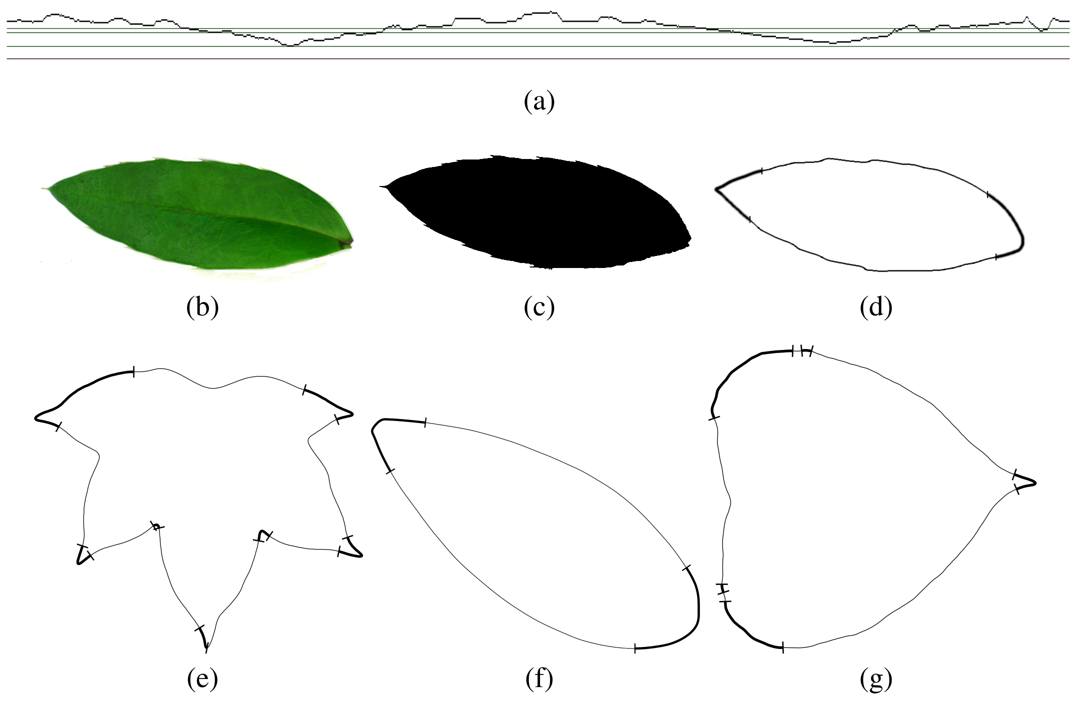


Figure 4.5: Corner region extraction of a leaf. (a) arc angles, guidelines from top to bottom: average, threshold, minimum, origin; (b) Original image; (c) Segmented image; (d) Leaf contour with corner regions marked, (e), (f), (g) additional samples showing corner regions

side the leaf surface. This mask is generated from the segmented image by applying erosion operator. The erosion operation is performed by a square kernel with size of 3. This value should be set according to the distance used in LBP operation, to ensure that LPB operation will never hit a border pixel. However, if a given leaf is extremely thin (like pinale leaves), it is possible to remove all or most of the image by the erosion operator. In these cases, instead of failing, the whole segmented image is used as the mask. Figure 4.6 compares LBP histogram around and outside the leaf margin.

4.1.7 Stalk removal

Some datasets and, in practice, result of some image acquisition techniques contain leaf stalks. Stalks may cause problems in geometric features. However, natural length of the stalk can contribute as an additional information. In some cases it might be beneficial to remove them. We developed a method to remove leaf stalks using the

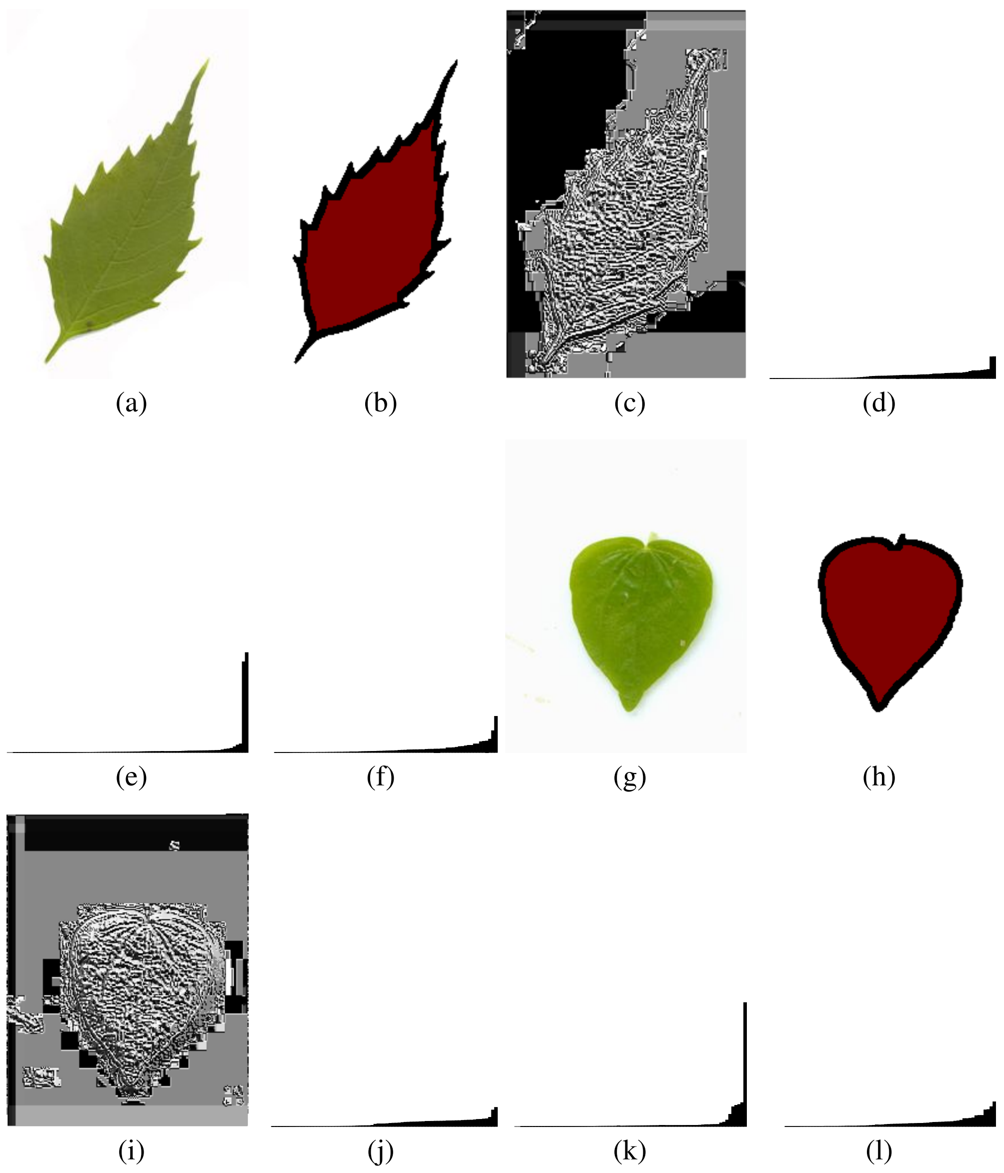


Figure 4.6: Analysis of the texture mask. (a) and (g) are original images, (b) and (h) are generated mask where the interior region is the final mask and the outer region is the area removed using erosion operator, (c) and (i) are LBP transformations, (d) and (j) are the masked histograms, (e) and (k) are the histogram outside the masked area, (f) and (l) are histograms of the border regions

smoothed contour data (with factor of 15). Using contour data instead of the whole image speeds up this procedure as contour points grow much slower than the leaf surface.

Stalk removal algorithm marks contour regions that are running parallel in opposite directions which are very close to each other. The threshold distance should be derived according to the dataset used. As an example, we use threshold distance as 4 in Leafsnap dataset. This method may produce multiple regions; in this case, only the longest of these regions is removed. Formal definition of this method is given in Algorithm 4.3 with the results shown in Figure 4.7. This algorithm assumes that arrays and lists are circular. It is possible for this algorithm to remove the entire image if the leaf is too thin. As a solution, it is possible either to ignore pinale leaves or use original contours for the feature extraction.

Algorithm 4.3: Stalk removal algorithm.

```
1: $contours$ is the array containing contour data
2: $T_{distance}$ is the input parameter
3: $T_{angle} = \pi/8$    ▷ threshold for straight line
4:
5: $detected$ is an empty list of indices
6: $pairs$ is a mapping from and index to another
7:
8: ▷ used to prevent connected points to be selected as the nearest point
9: $mindiff = 2 \times threshold$
10:
11: ▷ Find the closest point
12: for every contour point $ind$ do
13:     $current = contours[ind]$
14:
15:     $distance = T_{distance}$
16:     $minind = -1$
17:     for every contour point $ind2$ do
18:         $d = ||contours[ind2] - current||$
19:         if $d < distance$ then
20:             $distance = d$
21:             $minind = ind2$
22:         end if
23:     end for
24:
25:     if $distance \geq T_{distance}$ then
26:         continue with the next point
27:     end if
28:
29:     $other = contours[minind]$
30:
31:     ▷ Stalks are a part of the leaf surface
32:     $pivot$ is the point half way between $current$ and $other$
33:     if $pivot \notin Leaf$ then
34:         continue with the next point
35:     end if
36:
37:     ▷ Check if two points are parallel
38:     $angle_{current}$ is the angle between $contours[i-1]$ and $contours[i+1]$
39:     $angle_{other}$ is the angle between $contours[minind-1]$ and
40:         $contours[minind+1]$
41:     if $|angle_{current} - 2\pi + angle_{other}| > T_{angle}$ then
42:         continue with the next point
43:     end if
44:
45:     ▷ Successfully classified this point as a stalk point
46:     push $(current, other)$ to $pairs$
47:     push $current$ to $detected$
48: end for
```

```
if $detected$ is empty then
    No stalks to remove, exit
end if
▷ Find index difference between points
$differences$ is an empty list of numbers
for every point detected $ind$ do
    push $|detected[ind] - detected[ind-1]|$ to $differences$
end for

▷ Start from the beginning of a new group
$start = \mathbf{maxarg}\,[differences]$

▷ Find the longest uninterrupted sequence
$maxlen = 0$
$maxind = 0$
$current = 0$
$startelm = start$
for $ind$ starts from $start$ until all elements of $differences$ visited do
    ▷ End of the current sequence
    if $differences[ind] > T_{distance}$ then
        if $current > maxlen$ then
            $maxlen = current$
            $maxind = startelm$
        end if
        $startelm = ind$
        $current = 0$
    else
        $current = current + 1$
    end if
end for

▷ Remove longest sequence as the leaf stalk
$start = detected[maxind]$
$end = pairs[start]$
remove points from $contours$ beginning from $start$ to $end$
```

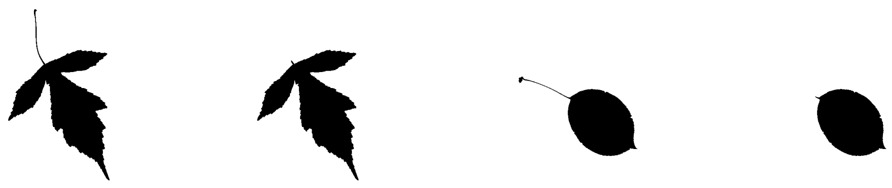


Figure 4.7: Results of stalk removal. (a) and (c) are original segmented images whereas (b) and (d) are images after stalk removal

4.2 Features

4.2.1 Margin descriptors

During our research, we examined leaves to develop additional ways of describing them. When we first started, there was a single method [3] to describe leaf margins. However, during our analysis, we found that this method was insufficient for this task. Therefore, the first feature set we introduced was aimed at describing leaf margins. Several methods used for describing margins are explained in the following four subsections.

4.2.1.1 Smooth factor

Before describing our solution, it might be beneficial to understand the margin feature proposed by [3]. This feature is called smooth factor. It is the ratio between the area of the leaf smoothed by 5x5 rectangular smoothing filter and the one smoothed by 2x2 filter. While smoothing removes margins, and therefore, creating a meaningful image, the area of the leaf is not affected. In Figure 4.8, this operation is illustrated. This example is a controlled sample, therefore, it is the worst case. In real life applications, this method might extract useful data, however, its usefulness will be limited as the method clearly does not perform the operation it intends to do.

4.2.1.2 Multiple Perimeter Ratios

Inspired from smooth factor feature, our first attempt to define leaf margin was to calculate perimeter ratio of the leaf after opening filters with different window sizes. Opening filters have an effect similar to smoothing, they remove the dents on the margin. Then, filters with different sizes are employed to capture margins of different sizes. However, this approach has several disadvantages. Firstly, filter size depends on the scale. Even though scanned leaf images have the same scale, this filter was not suitable for an all-purpose classification system independent from image acquisition

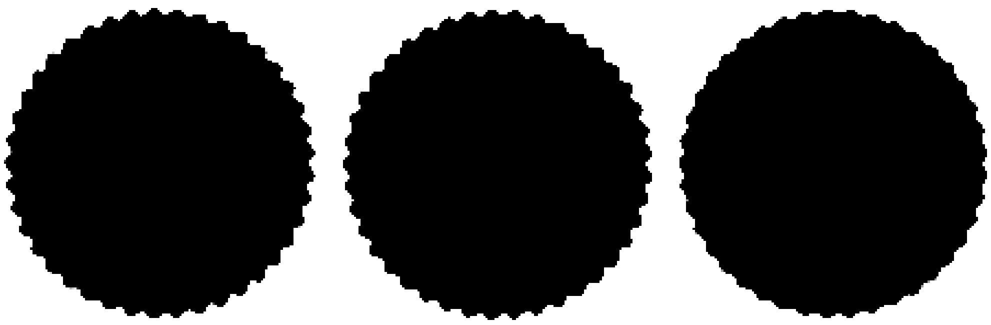

(a) A = 15110, P = 464 PR = 0.07 (b) A = 15117, P = 452 PR = 0.073 (c) A = 15125, P = 413 PR = 0.09

Figure 4.8: Area (A), perimeter (P) and perimeter ratio (PR) analysis of an artificially created margin over a circle. a) Original image, b) Smoothed by 2x2 rectangular filter, c) Smoothed by 5x5 rectangular filter

method. Another disadvantage of this method is its speed. It takes too long to perform multiple opening operations on the image. Finally, the features that are generated from this system is correlated and small scale changes might shift the values from one feature to another, causing problems with linear classifiers, such as LDC. Despite its disadvantages, this method is an improvement over smooth factor feature as evident from Figure 4.8 (on a regular circle PR is roughly 0.1 for all three cases). Since this method has many problems, we are not using this feature in our proposed systems.

4.2.1.3 Average Margin Distance

Implementing contour extraction system allows us to work with contour data. Since contour data is a time series, it can be smoothed using a one dimensional filter. Additionally, after smoothing step, it is possible to track new locations of the data points. This allows the calculation of the distance between smoothed and regular contour. This distance represents the amount of margins of a leaf. Additionally, the average of the distances extracted from the points are used to achieve scale invariance. This feature is called Average Margin Distance (MD). Margin distance feature has a single parameter and is versatile against scale changes. In the results section, we set this parameter, smoothing amount, to 15 in all simulations.

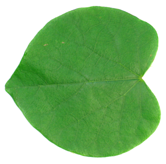
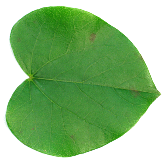
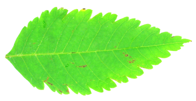
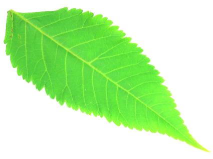
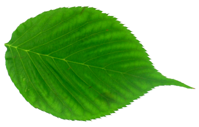
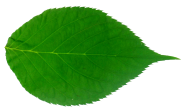
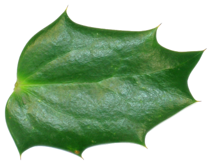
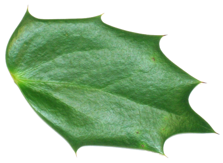
Although this method shows promise, there is a problem that prevents it from working on leaves. Smoothing causes large changes around the corners of the leaf. These changes are much larger than the changes along the leaf margin. Therefore, any variance in the leaf corners causes large amount of noise on this feature. In order to solve this issue, the corner regions detected by the method described in Section 4.1.5 are ignored while calculating this feature. Table 4.1 shows several leaf samples and their respective margin distances and statistics.

4.2.1.4 Margin Statistics

Although average margin distance is a stable and useful feature, it is possible to extract more information from the leaf margins. The second set of margin features we developed is margin statistics. These features are extracted from the margin peaks and describe the margin in more detail. For instance, a leaf type having a serrate margin and another leaf type having a crenate margin can have similar average margin distance. However, when examined, the teeth of the leaf having serrate margin would be higher to cover the same area as round and thick crenate margin.

There are four features in margin statistics. These are average peak distance (MSAD), average peak height (MSAH), peak distance variance (MSDV), and peak height variance (MSHV). When combined with margin distance, these statistics adequately describe leaf margins.

Table 4.1: Average margin distance and margin statistics of several leaves (MD: margin distance, MSAD: margin statistic: average distance, MSAH: margin statistic: average height, MSDV: margin statistic: distance variation, and MSHV: margin statistic: height variation).

Class	Image	MD	MSAD	MSAH	MSDV	MSHV
4		0.38	0.25	0.71	0.063	0.0005
4		0.37	0.50	0.39	0.25	0.0
10		1.98	0.019	2.14	0.0004	0.58
10		1.99	0.011	2.04	0.0001	0.67
22		1.86	0.012	1.92	0.0001	0.72
22		1.96	0.011	2.04	0.0001	0.54
28		0.48	0.067	0.88	0.004	0.047
28		0.49	0.053	0.92	0.003	0.034

Unlike margin distance, margin statistics cannot be extracted trivially. The first step to extract margin statistics is peak detection. Leaf margin distances that are calcu-

Algorithm 4.4: Proposed peak detection algorithm

```
procedure PEAKS(M, L)
    ◇ M is the margin distances, L is the threshold
    $V$ is an empty $M \times L$ matrix
    for $i$ = 1 to $Size_M$ do
        for $k$ = 1 to L do
            if $M_i > M_{i-k} \wedge M_i > M_{i+k}$ then
                $V_{k,i} = 0$
            else
                $V_{k,i} = 1$
            end if
        end for
    end for
    $sums_i = \sum_k V_{k,i}$
    return $indexesof_{sums=0}$
end procedure
```

lated while extracting average margin distance are used for this task. We employ a peak detection algorithm that is derived from Automatic Multiscale Peak Detection algorithm proposed in [35]. This algorithm first calculates multi-scale maxima matrix, however, instead of cutting the matrix from an automatically determined point, we set the threshold to be the half of the smoothing distance used to obtain margin distances. The final form of the algorithm is presented in Algorithm 4.4. Additionally, we remove any peaks that are less than 0.5. These very low peaks can be formed by segmentation noise around the edges and do not contain any information about margins. The margin statistics extracted from various leaves are shown in Table 4.1.

4.2.2 Texture descriptor

We developed a new variant of Local Binary Patterns for texture description called Sorted Local Binary Patterns ($LBP^s_{P,R}$). There are many LBP variants that achieve rotation invariance [36–40], however, our aim in texture description is not to describe ordinary textures. Leaf textures distinctly contain veins that are expanding from the midrib. Therefore, the texture descriptor should take this information into account. There are two modifications that are performed over the regular LBP.

After extraction of 8-bit LBP histogram with distance of 1, this method sorts the values of the histogram. This removes any direction information as it is represented by the x-axis of the histogram. We chose this method over other LBP methods that are rotation invariant as these features should represent length, frequency, and angles of the veins instead of entirety of the texture. Although, Regular LBP contains separate information for separate directions, most rotation invariant methods, such as Rotation Invariant Uniform Binary Patterns, combine different directions into a single pattern. However, this information is required to correctly distinguish between vein types. Sorting does not combine different directions into one, it simply discards the direction information related with a histogram value. Therefore, if there are three dominant patterns, there will still be three dominant patterns in the sorted histogram, but they will always be at the right side of the histogram.

$$LBP^{s}_{P,R} = \mathbf{sort}\left[LBP_{P,R}\right] \tag{4.3}$$

where P is the number of neighbors, R is the distance of neighboring samples from the center, g_c is the gray-scale intensity at the center pixel, and g_p is the intensity of neighbor p.

The second modification is the feature reduction on $LBP^{s}_{P,R}$. Since the histogram is sorted, left side of the histogram contains many similar values. These values can effectively be represented with less features as they contain much less information compared to the higher values. Our research shows that using variable averaging of these values to reduce the number of features improves the performance and the processing speed. This feature reduction method has a single parameter, σ which is the number of features that will not be averaged at the end. After this reduction operation, there will be $P - log_2\sigma + 2$ segments, where P is the neighborhood of LBP, with each having $\sigma/2$ elements. The following set of equations show how this feature reduction method works:

$$\begin{aligned} N^{\sigma}_{P,R,i} &= \mathbf{max}\left[\frac{2^P}{\sigma * 2^{\lfloor 2i/\sigma \rfloor}}\right] \\ S^{\sigma}_{P,R,i} &= \sum_{k=0}^{i-1} N^{\sigma}_{P,R,k} \\ LBP^{rs\sigma}_{P,R,i} &= \sum_{k=S^{\sigma}_{P,R,i}}^{S^{\sigma}_{P,R,i}+N^{\sigma}_{P,R,i}} \frac{LBP^{s}_{P,R,k}}{N^{\sigma}_{P,R,i}} \end{aligned} \tag{4.4}$$

where $LBP^{s}_{P,R,i}$ is the entry i (0 based) of the Sorted LBP histogram with P neighborhood and R distance. N is the number of items to be averaged for element i and S is the start of i^{th} averaging group in $LBP^{s}_{P,R}$ histogram. In addition to this formula, the algorithm for this method is presented in Algorithm 4.5.

It is possible to derive differentiation information by looking into the sorted histograms. In Table 4.2, LBP histograms of different leaves are shown. It is evident from this table that when the veins are parallel to each other, there are a few very frequent patterns (the first and the second rows), however, when the veins are not very obvious, the histogram will not have very frequent patterns. When applied to regular texture classification, $LBP^{rs\sigma}_{P,R}$ has a low performance. However, it performs comparable or better in terms of accuracy in the leaf texture classification when compared to the other LBP variants. The detailed analysis of $LBP^{s}_{P,R}$ and $LBP^{rs\sigma}_{P,R}$ with feature reduction is discussed in Chapter 5.

Algorithm 4.5: $LBP^s_{P,R}$ feature reduction.

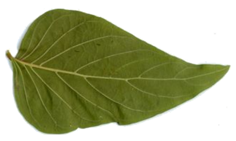
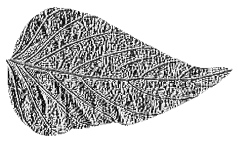
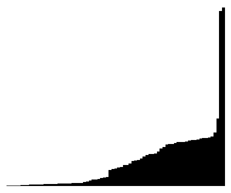
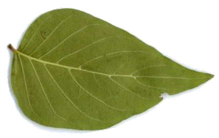
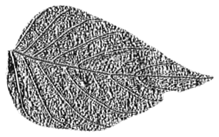
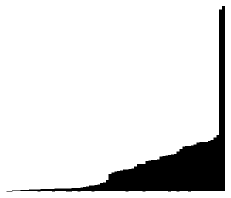
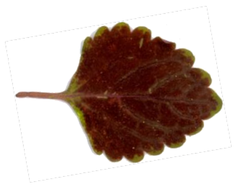
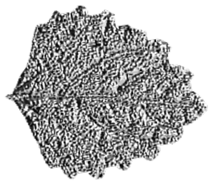
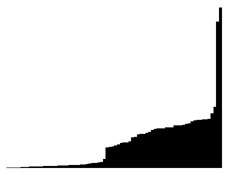
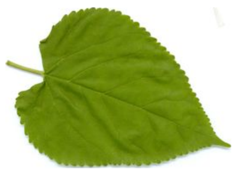
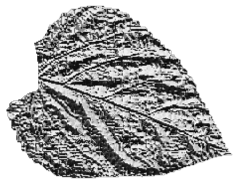
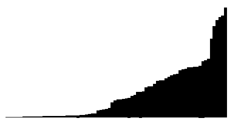
```
procedure REDUCE($SLBP$, $P$, $\sigma$)
    ◇ $P$ is the LBP neighborhood
    ◇ $\sigma$ is the number of elements that are not averaged
    $elements = \sigma/2$
    $current = elements/2$
    $ind = 0$
    $RSLBP$ is an empty vector
    while $ind < 2^P$ do
        for $i = 0$ to $elements$ do
            $sum = 0$
            for $j = 0$ to $current$ do
                $sum = sum + RSLBP_{ind}$
                $ind = ind + 1$
            end for
            push $sum/current$ to $RSLBP$
        end for
        $current = \mathbf{ceil}(current/2)$
    end while
    return $RSLBP$
end procedure
```

4.2.3 Color features

Although many leaves have green color, there are variations in the color tone. Additionally, color shifts on the leaf blade is a characteristic feature generally associated with specific plant families. Therefore, it is beneficial to include color descriptors for leaf classification. These features will not improve classification accuracy on all leaves, however, they will improve accuracy of few classes significantly. Detailed performance analysis of color descriptor is discussed in Chapter 5.

The color features we use are the mean and the deviation of the hue channel. Only the pixels that are inside the texture mask are considered while collecting statistical information. Hue channel is extracted using CIE-LCH color model. Hue is preferred over other channels as it contains less amount of noise, invariant to lighting, and more meaningful in the context of leaf classification. Hue deviation is employed to measure

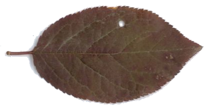
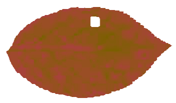
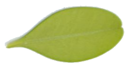
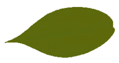
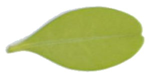
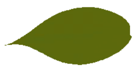
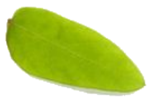
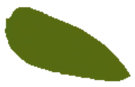
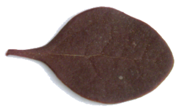
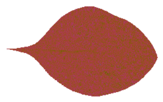
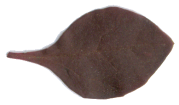
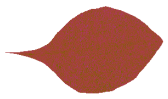
Table 4.2: Intermediate and final results from LBP feature extraction. Leaves in the first two rows belong to the same class.

Class	Original	$LBP^{s}_{8,1}$	$LBP^{rs32}_{8,1} Histogram$
192			
192			
203			
134			

color change on the leaf blade. For instance, veins on the leaves of some type of plants are colored differently compared to the rest of the leaf blade. These kind of changes can be measured using deviation. Several leaves, their hue channels, and their corresponding color features are listed in Table 4.3; it is possible to observe that these hue based features have very low variance. However, there are many leaves that have similar hue, diminishing the contribution of these features.

Hue channel is a circular value. This means that there is a discontinuity in the scale. However, this discontinuity is at pure red. The highest hue values are purple and the lowest values are orange. Since purple color leaves are rare, this discontinuity does not have any real impact on the performance of the system. Additionally, some classifiers,

Table 4.3: Color features of sample leaves.

Class	Original	Hue	Mean	Deviation
8			0.159	0.031
30			0.306	0.002
30			0.307	0.002
58			0.319	0.002
104			0.136	0.159
104			0.137	0.150

such as LDC, can easily ignore these features for a specific leaf type if the samples of this leaf type has a drastic variance in these features. Out of 220 different leaf types, this problem occurs only in a single type of leaf in the largest leaf dataset we use for experimentation.

4.3 Classifier

K-Nearest Neighbors (KNN) and Support Vector Machines (SVM) are the most popular choices for the leaf classification. We also started our initial attempts with KNN, specifically, the case where K is 1, which is called Nearest Neighbor Classifier (NNC). Despite its simplicity, this classifier outperforms many classification systems. We tried many classifiers for this task. One of the obvious choice for the task was LDC. LDC assigns different weights to features depending on the class. This allows LDC to prioritize features based on class, rather than prioritizing them globally. For instance, KNN treats all features to be of equal importance. Additionally, LDC is based on normal distribution, which often occurs in the nature.

Quadratic Discriminant Classifier (QDC) is based on normal distribution similar to LDC. However, while training LDC, rather than using different covariance matrices, a common covariance for all classes are calculated. A covariance matrix will be singular, if the number of samples it is derived from is less than the number of dimensions it has. Therefore, common covariance allows LDC to work with much smaller sample sets. This means that for QDC to work, the number of training samples should be more than the number of dimensions. However, there are no leaf datasets that have as many samples as the number of features that we use for experimentation.

4.4 Systems

During our research, we introduced various features and a classifier that has never been tried in this field. However, our most important contribution is the complete leaf classification systems we designed. For these systems, we identified combinations that work well together. First of all, we proposed a geometric leaf classification system that uses geometric features of leaves. Then, we combined texture and color features with geometric features in order to improve leaf classification accuracy. In the following subsections, these two systems are explained.

4.4.1 Geometric Leaf Classification

The first system we introduced depends on geometric features to describe leaves and named as Geometric Leaf Classification (GLC) [6]. This system is the first leaf classification system that uses LDC as classifier.

Previous studies generally focus on single descriptors [2, 9–11] or combine many features that may have overlaps [7, 15, 16]. However, for a classification system to work well, it requires enough but diverse features. In this respect, the main step is to investigate necessary features to describe a leaf. According to our analysis, a leaf shape should be described from the following aspects:

- Broad shape
- Shape
- Complexity, such as number of lobes or compound segments
- Margins

There are many geometric features to describe broad shape of the leaf. These are described in Section 3.4. Additionally, moment invariants describe the leaf shape in general adding another layer on top of classical geometric features. This combination sufficiently describes the broad shape of the leaf, however, these features cannot fully describe the leaf shape.

MDM is employed in this system for the purpose of shape description. Inclusion of MDM causes several issues. The first problem is that it contains geometric information that is already described using geometric features. We solved this issue by identifying and removing these geometric features. MDM describes all point to point distance based geometric features. Therefore, the features that are not related with MDM are the features calculated using convex hull and perimeter. These are Area Convexity, Perimeter Convexity, Circularity, and Perimeter Ratio.

The second problem related with MDM is the number of features. Authors in [2] proposed MDM using 64 or 128 nodes. An MDM with 64 nodes generates 2048 features whereas an MDM with 128 nodes generates 8192 features. Clearly, these features will dilute the effect of other features. Additionally, increased number of features might cause classifiers to fail. Therefore, we experimented with the ideal number of MDM nodes. According to these experiments, 32 node MDM combined with the other features used in this system has the highest accuracy. Additionally, we remove one extra row of MDM as this row contains duplicate values in pairs. With this modification, 32 node MDM extracts 480 features.

The third problem with MDM is leaf margins. MDM is affected by the margin types other than entire. However, this effect is not uniform. Some nodes fall on peaks of the margin, whereas some others fall on pits. Therefore, margins are a source of noise for MDM. To solve this issue, this system extracts MDM from the smooth contours where the leaf margin is mostly removed. As discussed in Chapter 5, this modification results in a slight improvement which is beneficial to some leaf types.

Since MDM and other geometric features cannot describe margins, we developed average margin distance and margin statistics for this task. These features effectively describe leaf margins, allowing leaves that differ in margin type to be distinguished easily.

The proposed Geometric Leaf Classification (GLC) system is described in detail in [6].

4.4.2 Combination of Geometric, Texture, and Color Features

The success of Geometric Leaf Classification method lead us to focus our efforts to expand our system beyond shape based classification. We developed texture and color features for this task. There are studies in the literature that use shape, texture, and color together. However, as before, our primary aim is to identify and develop features that are complementary in nature and keep dimensionality low to ensure that the data supplied to classifier is meaningful.

This system depends on our previous system, Geometric Leaf Classification. However, due to the increase in the number of features, we further reduce MDM nodes to 24. This reduces the number of features generated by MDM to 264, almost half of the previous value. This reduction increases the effect of all other descriptors.

We developed a new LBP variant for texture description. This method is specifically designed to represent the venation patterns in leaves. A sizable portion of the leaf contains similar texture information as the whole leaf. Therefore, this descriptor not only improves the accuracy by providing additional information; but also enables classification of the leaves that have problems with the segmentation and contour extraction steps.

Color is an important part of the leaf. Even though many leaves have the same color, different tones exist. Additionally, non-typical colored leaves can benefit from this extra distinguishing features. Therefore, we developed color based features into our system for this task. We use hue channel, as the other methods of describing color are either noisy (chromacity) or vary with light (luminance, red, green, blue).

This system describes a leaf from all aspects. The following is the list of feature sets and the corresponding methods used in the system:

- Broad shape: moment invariants
- Shape: MDM
- Complexity: area and perimeter convexity, circularity, and perimeter ratio
- Margins: margin distance and margin statistics
- Venation: sorted LBP
- Color: mean and deviation of hue channel

The proposed leaf classification system which is based on geometric, texture and color features, namely GTCLC, carries leaf classification into another level as the improvement over other methods are significant.

Chapter 5

RESULTS

5.1 Datasets Used

We use different datasets to demonstrate and validate methods developed. It is possible to argue the effectiveness of a system that works on a single dataset, therefore, it is crucial to test systems on multiple datasets. In this thesis, we use three different leaf datasets and an additional dataset for texture classification. Comparison of these datasets is given in Table 5.1.

Table 5.1: Comparison of datasets

Dataset	ICL	Flavia	Leafsnap
Format	JPG	JPG	JPG
License	Academic	Free	Academic
Resolution			
Mean	276x366	1600x1200	633x593
Std. deviation	97x174	0x0	39x67
Minimum	29x31	1600x1200	330x200
Background color			
Mean (red, green, blue)	252, 252, 252	254, 254, 254	225, 229, 227
Std. deviation	4.1, 3.4, 2.9	1.4, 1.3, 1.4	28, 25, 27
Classes	220	32	132
Samples	16757	1907	4375
Samples per class			
Mean	76.2	59.6	33.0
Std. deviation	94.5	6.6	4.5
Minimum	26	50	7
Algorithms			
Segmentation	Proposed	Proposed	[16]
Stalk removal	No	No	No

Intelligent Computing Laboratory Dataset (ICL) [41] is a dataset available for researchers who agree the terms set forth by Chinese Academy of Sciences. This dataset contains 220 classes with 16757 images. It is the largest leaf dataset that is publicly available. We have reduced the number of images in class 42 as it had more than 1000 images. This dataset has a lower resolution compared to other leaf datasets. Unlike many papers [2,12,22] that use ICL for experimentation, we use entirety of this dataset in our experiments. Samples from this dataset are shown in Figure 5.1.

Flavia Leaf Dataset [42] is a publicly available leaf dataset with a liberal license, allowing it to be used for any purpose. There are 32 classes with 1907 samples. Images are very high resolution and taken by a camera. The background is white with some amount of noise. Since the images are acquired using a camera, probably using flash, there are bright highlights on some leaves as seen from Figure 5.2. Due to these reasons, noise removal is a crucial preprocessing step in this dataset.

Leafsnap Dataset [26] is gathered with the intention of building a leaf library in a specific region. Therefore, there are many similar species. Additionally, the background and the illumination level of the leaves change from image to image, causing simple segmentation algorithms to fail. We have selected 4375 samples from this dataset containing 132 classes. We have avoided Pinales leaves as they are completely removed by our stalk removal algorithm. In this dataset, we employed segmentation algorithm described in Section 3.2. There are additional markers around the leaves that need to be cleaned. We have built an automated system for this task. Sample images from this dataset are shown in Figure 5.3.

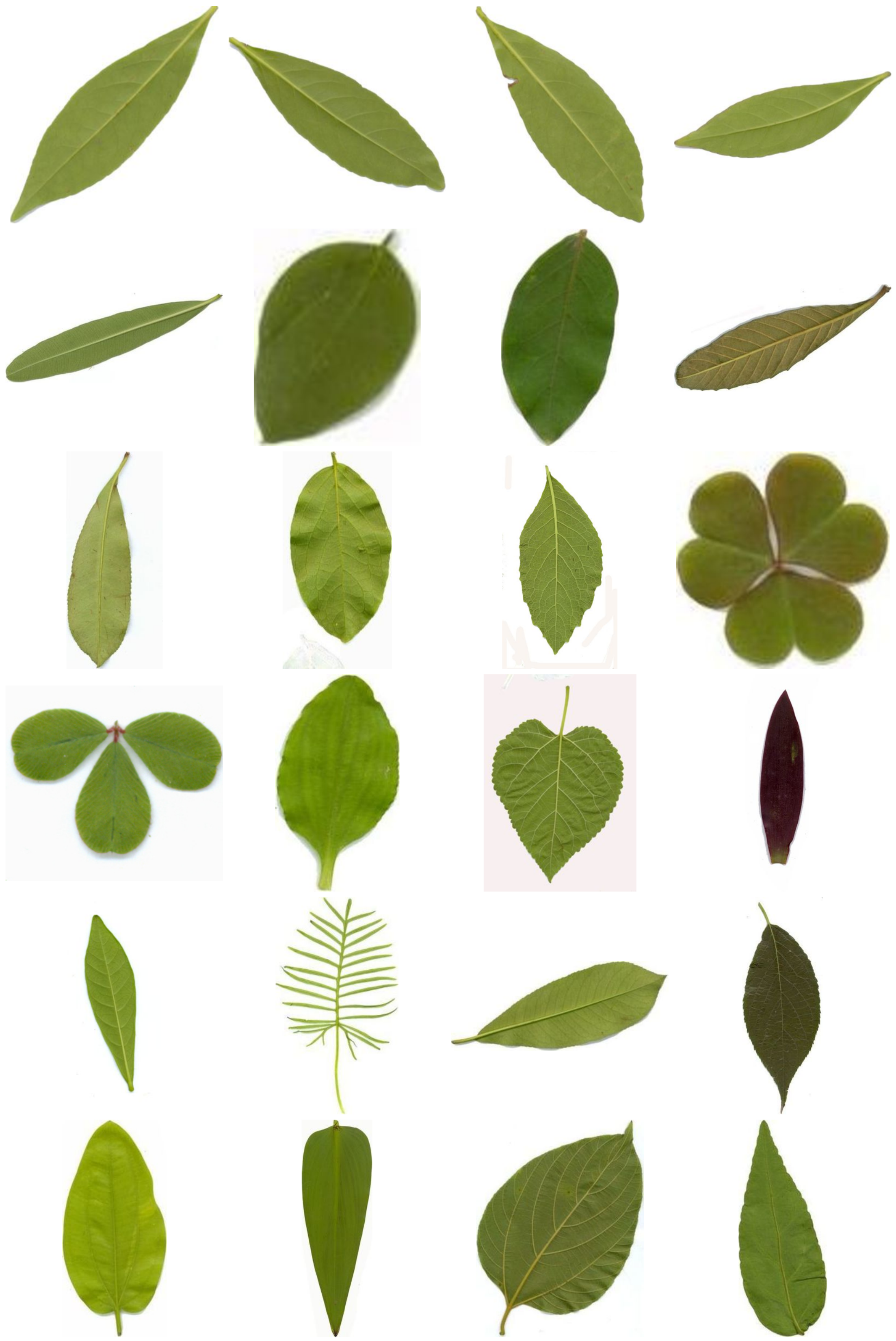


Figure 5.1: Sample images from ICL dataset. Images in the first row belong to the same class.

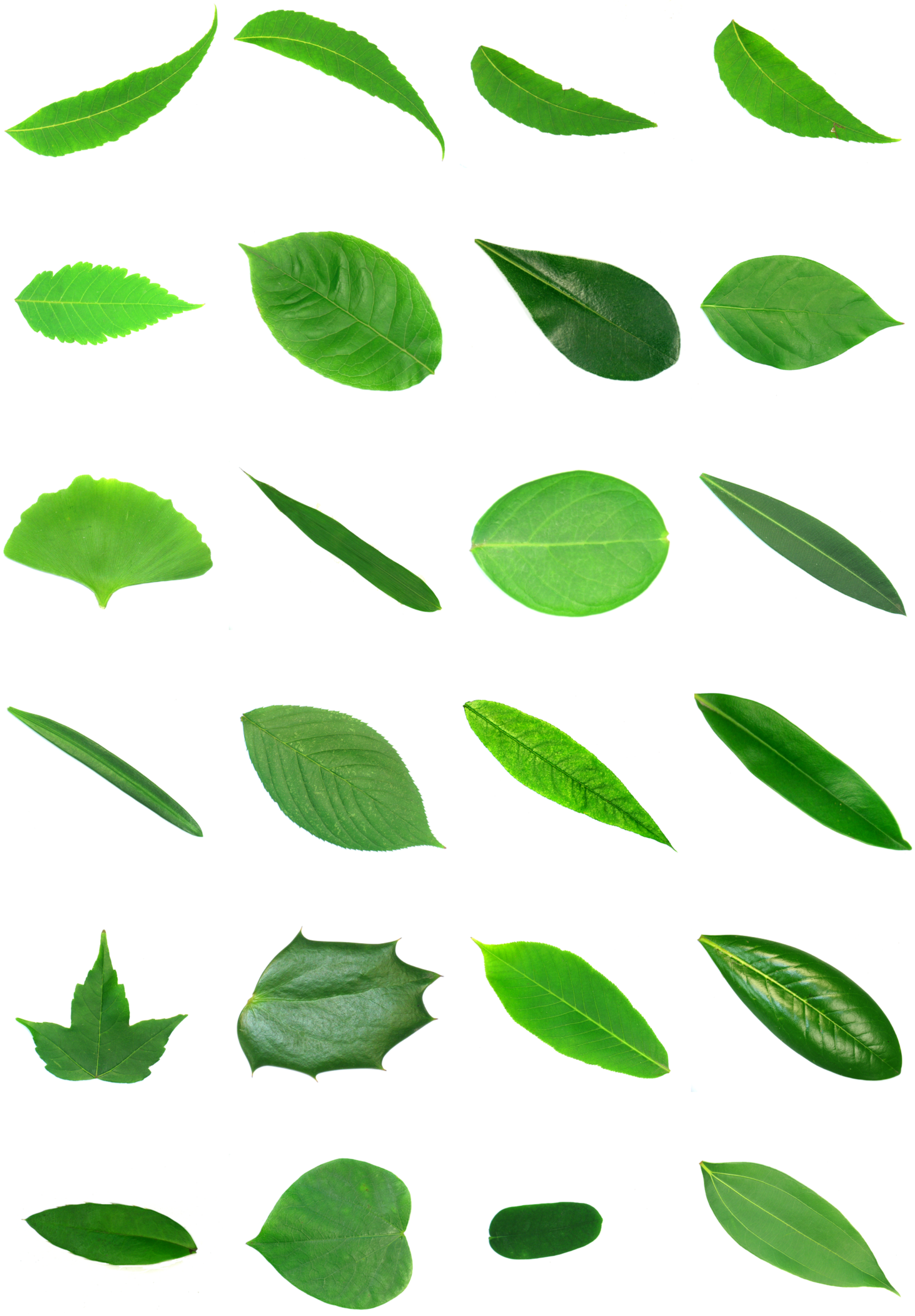


Figure 5.2: Sample images from Flavia dataset. Images in the first row belong to the same class.

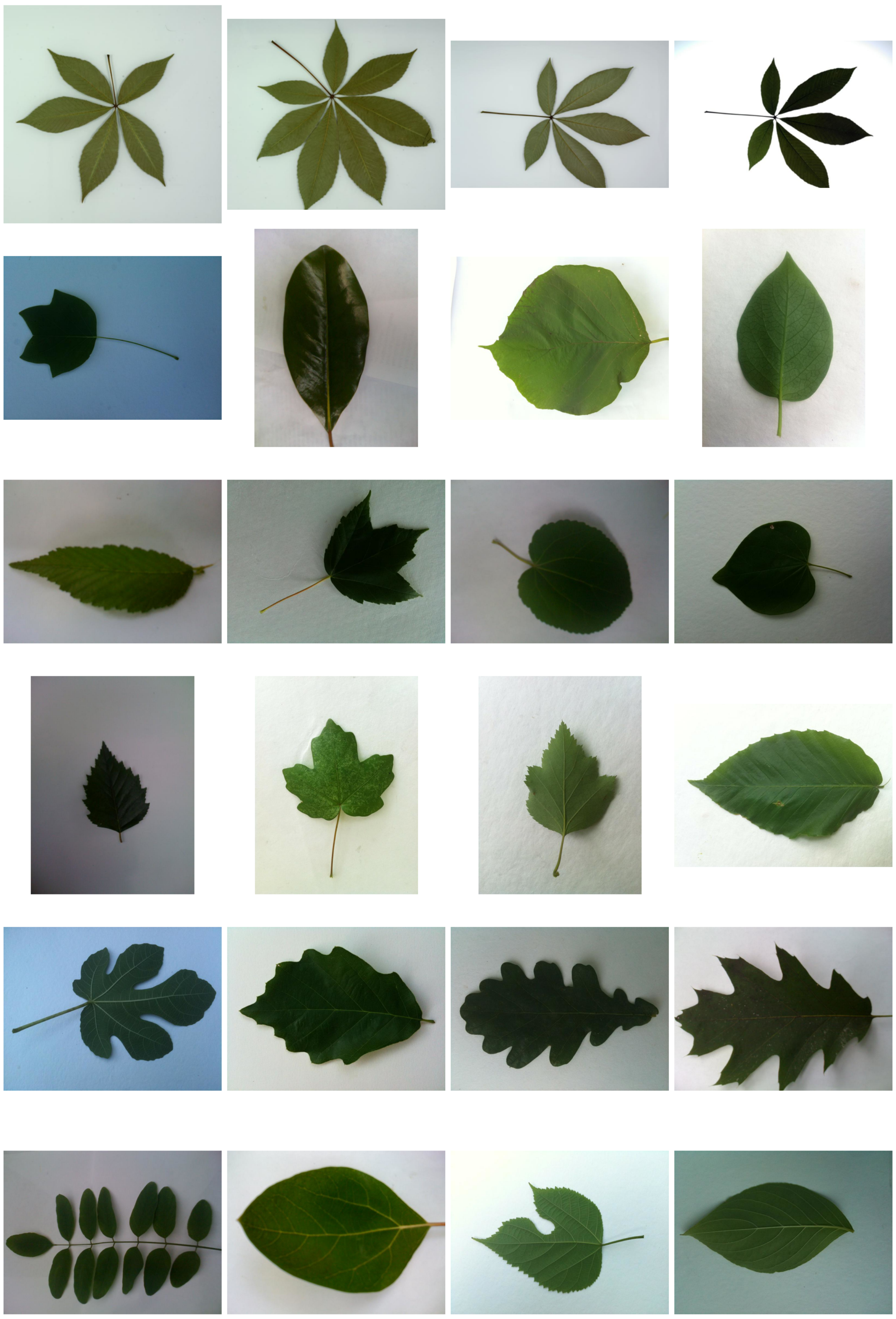


Figure 5.3: Sample images from Leafsnap dataset. Images in the first row belong to the same class.

5.2 Experimental Methodology

Unless explicitly specified, all results presented in this thesis are average of 100 simulations. All computational cost calculations are performed on a computer with Intel i5 processor and 4GB of RAM. All methods in the results section are implemented in C++. In addition to the proposed methods, we implemented other algorithms for comparison. These algorithms are also implemented in C++ and their parameters are tuned for the best results.

We performed many experiments in five categories. In the first category, different leaf classification systems are compared in various aspects. Different datasets and varying number of training samples are used in these experiments. Experiments related to preprocessing methods are presented in the second category. Additionally, the ideal size of the images are explored in this section. Different features, including the ones that are not used, are evaluated in the third category. These experiments also illustrate changes in the parameters. The fourth category compares classifiers using different feature sets and datasets. The final category explores the computational cost of every method that has been discussed in this thesis. These experiments are discussed in the following section.

5.3 Comparison of Different Systems

In this section, we compare the proposed systems with the state-of-the-art systems in the literature. These are the MDM [2], Flavia [3], and MCH [16]. Accuracy comparison of these methods are represented in Figure 5.4. Additionally, analysis of 100 simulations for every dataset is given in Table 5.2. According to these experiments, our research improved this field significantly.

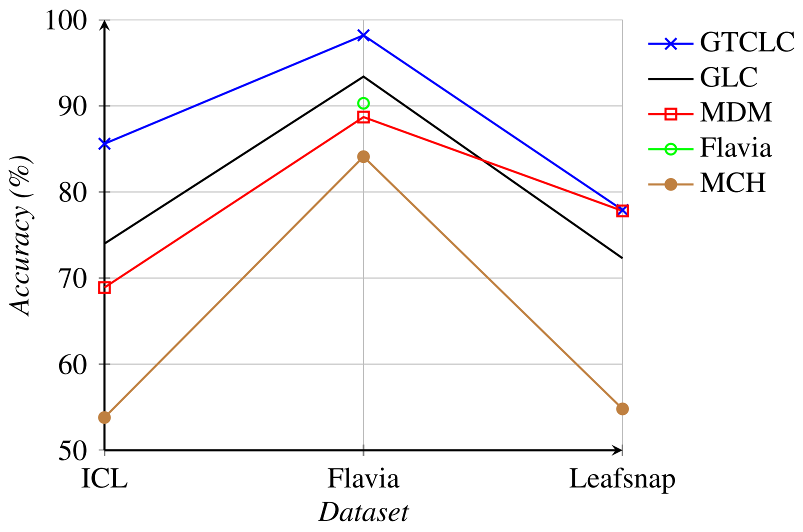


Figure 5.4: Accuracy comparison of the proposed systems with the state-of-the-art methods using different datasets.

Table 5.2: Accuracy analysis of different systems

		Accuracy			
Dataset	Method	Minimum	Maximum	Average	Std. deviation
ICL	MCH	50.9	56.7	53.8	1.19
	MDM	64.4	69.3	68.9	1.00
	GLC	72.1	76.3	74.0	0.88
	GTCLC	83.4	87.4	85.6	0.76
Flavia	MCH	78.1	88.8	83.6	2.37
	MDM	81.5	94.4	88.7	2.07
	GLC	88.2	97.8	93.4	1.91
	GTCLC	96.1	100	98.2	0.80
Leafsnap	MCH	47.5	59.6	54.8	2.45
	MDM	49.7	81.7	77.8	3.93
	GLC	68.0	77.8	72.3	2.14
	GTCLC	72.5	82.0	77.9	1.97

It is important for classification systems to work with limited amount of data. Therefore, we performed an experiment to compare the methods using different ratios for training/testing data. Results of this experiment are shown in Figure 5.5 and Table 5.3.

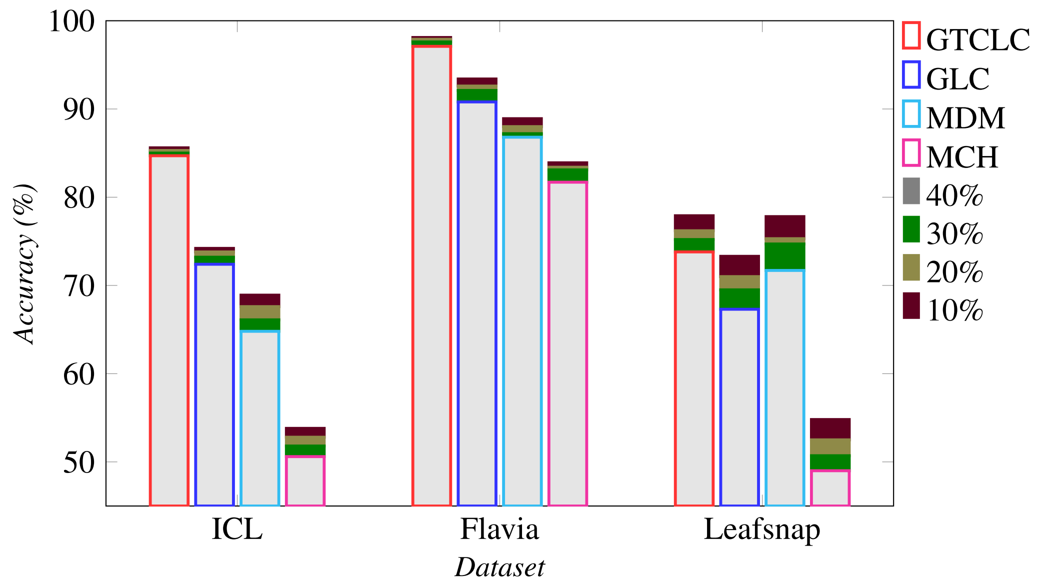


Figure 5.5: Accuracy comparison of the proposed systems with the state-of-the-art methods using different datasets with varying amount of testing samples.

Table 5.3: Accuracy table of the proposed and the state-of-the-art systems.

		Testing samples			
Dataset	Method	10%	20%	30%	40%
ICL	MCH	53.8	52.8	51.8	50.6
	MDM	66.7	67.6	66.1	64.8
	GLC	74.0	73.6	73.0	72.4
	GTCLC	85.6	85.3	85.0	84.7
Flavia	MCH	68.4	83.6	83.1	81.7
	MDM	88.7	88.0	87.2	86.8
	GLC	93.4	92.6	92.1	90.8
	GTCLC	98.2	97.8	97.6	97.1
Leafsnap	MCH	54.8	52.5	50.7	49.0
	MDM	77.8	75.3	74.7	71.7
	GLC	72.3	71.0	69.5	67.3
	GTCLC	77.9	76.2	75.2	73.8

Even though overall accuracy is important, a good classification system should be able to classify all classes with equal accuracy. In order to prove effectiveness of the systems that are proposed in this thesis, we conducted an experiment where classification accuracy of individual classes are calculated. The results of this experiment are shown

Table 5.4: Statistical information derived from per class accuracies computed using ICL dataset

	MCH	MDM	GLC	GTCLC
Average	54.0	67.3	74.1	85.5
Deviation	21.0	21.3	15.7	11.3
Lowest	3.3	8.1	26.8	32.0
Highest	99.8	100	100	100
0-20%	12	2	0	0
20-40%	43	29	7	1
40-60%	80	59	43	8
60-80%	58	56	88	40
80-100%	27	74	82	171

in Figure 5.6 with the statistical information presented in Table 5.4. Due to having multiple descriptors, GTCLC performs better in hard-to-classify classes. GTCLC results in only 9 classes being less than 60%; the number of classes having less accuracy than 60% is 50 in GLC, 90 in MDM and 135 in MCH.

In addition to the accuracy comparison of the methods that we implemented, we enlist the accuracy scores of the well-performing methods on Flavia dataset. These values are taken directly from the original papers. This comparison is shown in Table 5.5.

Finally, we present the confusion matrices of the proposed systems for Flavia dataset in Figure 5.8 and 5.9. Additionally, the leaves that are most confused are shown in Figure 5.7. These experiments signify the effect of including texture features. For instance, classes 1 and 21 are often confused in GLC. The reason behind this confusion is the variance of the shape of leaves. Class 1 has a high variance in shape, some samples are similar to class 21 and some of them are different. These different samples shift the weight matrix of class 1 away from class 21, resulting class 1 samples to be classified as 21. However, their venation patterns are different. Therefore, using texture

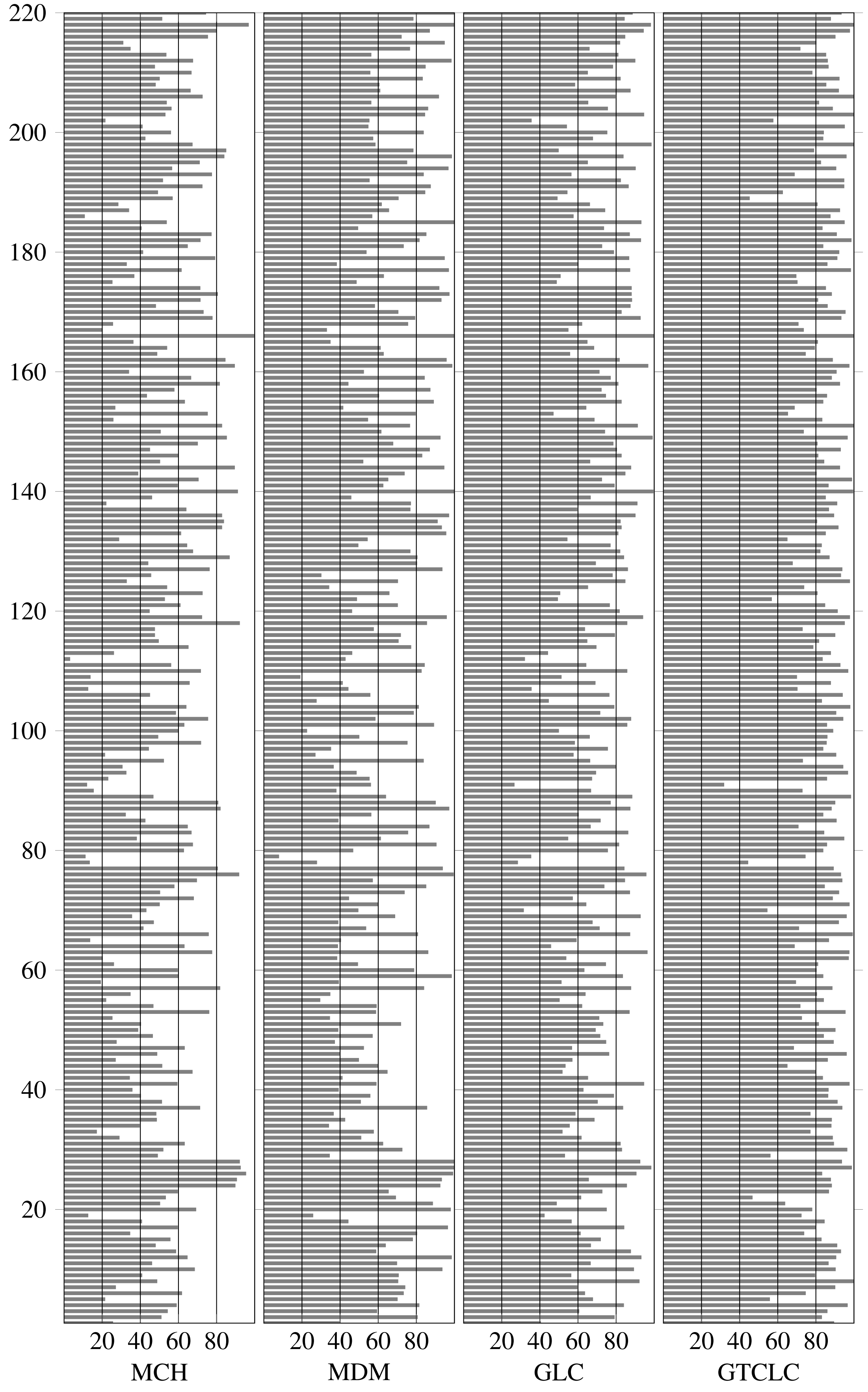


Figure 5.6: Accuracy comparison of state-of-the-art methods per class

Table 5.5: Comparison of different methods over Flavia dataset

Method	Accuracy
Flavia [3]	90
RBPNN [14]	91
MMC [17]	92
BPNN [17]	92
GLC [6]	93
1-NN [14]	93
RBFNN [43]	94
MLNN [43]	94
F-SVM [4]	94
CS1 - MAP [4]	97
GTCLC	98

information, GTCLC improves the separation of these two classes. GLC confuses these classes a total of 171 times in 100 simulations, whereas GTCLC reduces this confusion to zero.

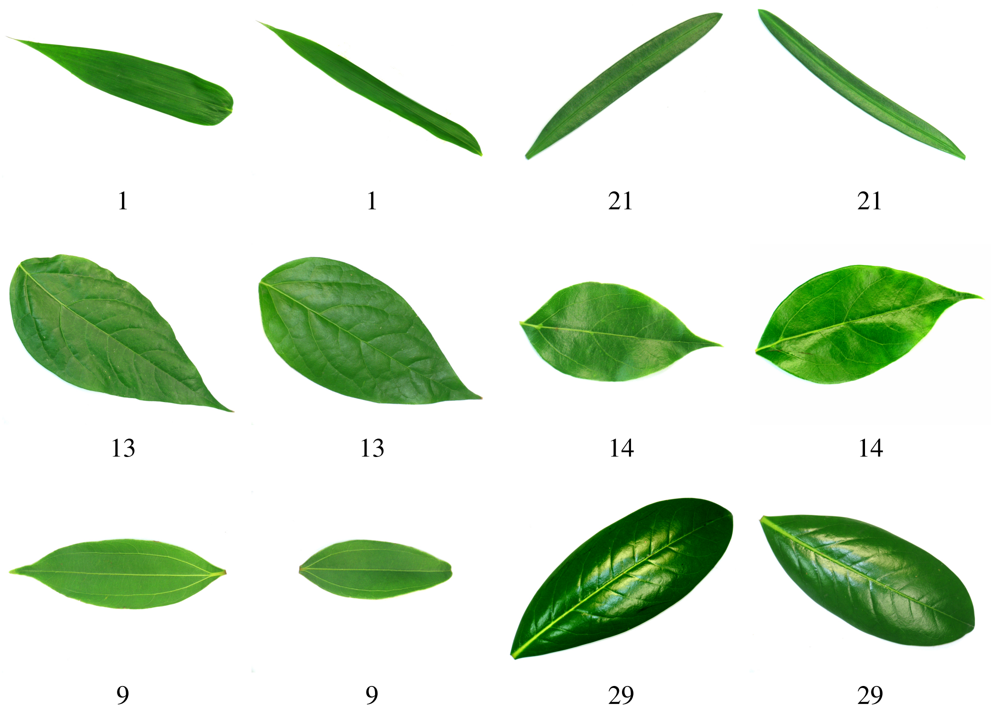


Figure 5.7: Most confused leaves and their class indices

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
483	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	597	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	6	0	0	0	0	0	0	0	0	0	0
0	0	574	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	700	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0
0	0	0	0	696	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	493	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	581	0	4	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0
0	0	0	0	0	0	0	497	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	19	0	494	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
0	0	0	0	0	0	0	0	0	499	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	500	0	0	4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	574	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	26	475	16	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5
0	0	0	0	0	0	0	0	0	0	0	0	10	565	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	21	0	0	0	0	0	0	0	0	0	0	0	591	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	487	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	700	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	589	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	600	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	591	34	0	0	0	0	16	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	566	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	493	0	0	0	0	0	0	0	0	0	0
0	0	0	0	4	0	0	0	0	0	0	0	15	8	0	1	0	8	0	0	0	0	500	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	592	0	0	0	0	0	0	0	0
10	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	498	0	0	0	0	0	0	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	475	0	0	1	0	0	0
0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	486	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	500	0	0	0	0
0	0	0	0	0	0	0	0	2	0	0	0	0	4	9	3	0	0	0	0	0	0	0	0	0	9	0	0	472	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0	600	0	0
0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	494	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	482
96.6	99.5	95.7	100.0	99.4	98.6	96.8	99.4	98.8	99.8	100.0	95.7	95.0	94.2	98.5	97.4	100.0	98.2	100.0	98.5	94.3	98.6	100.0	98.7	99.6	95.0	97.2	100.0	94.4	100.0	98.8	96.4

Figure 5.8: Transpose of confusion matrix for Flavia dataset using GTCLC. First row is the class number, last row is the accuracy of that class.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
347	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	0	0	0
0	594	7	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	26	0	0	0	0	0	0	0
0	0	578	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	9	0	0	0	0	0	0	0
0	0	0	700	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	2	0	0	4	3	0
0	0	0	0	698	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	497	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	520	0	23	0	0	1	0	0	0	9	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0
0	0	0	0	0	3	0	496	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0
0	0	0	0	0	0	64	0	446	0	0	0	0	27	6	0	0	0	0	0	0	0	0	0	9	3	0	0	50	0	0	0
0	0	0	0	0	0	0	0	0	487	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	476	0	10	0	0	0	0	12	9	0	0	0	0	3	0	0	1	13	0	9	0	0
0	0	0	0	0	0	0	0	0	0	0	556	13	7	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	4
0	0	0	0	0	0	0	0	0	0	0	25	410	78	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	3	38	452	5	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
0	0	0	0	0	0	0	0	0	0	10	1	0	0	546	1	0	0	0	0	0	0	20	0	1	0	0	0	26	0	0	0
0	0	0	0	0	0	0	0	2	0	12	14	0	0	1	450	0	0	0	0	0	0	6	0	6	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	700	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	577	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	2	0	23	17	5	0	0	0	590	0	0	0	47	0	0	0	0	29	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	596	11	0	0	0	0	48	0	0	0	0	0	0
153	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	570	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	500	0	0	0	0	0	0	0	0	0	0
0	0	0	0	2	0	0	0	1	0	0	0	6	19	36	16	0	0	0	0	0	0	427	0	0	0	0	0	0	0	0	10
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	545	1	0	0	0	0	0	0	0
0	6	15	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	43	448	0	0	0	0	0	0	3
0	0	0	0	0	0	10	0	3	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	439	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	465	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	458	0	0	0	0
0	0	0	0	0	0	6	0	19	0	0	0	0	0	1	9	0	0	0	0	0	0	0	0	0	0	0	0	424	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	8	0	0	587	0	0
0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	0	0	0	497	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	480
69.4	99.0	96.3	100.0	99.7	99.4	86.7	99.2	89.2	97.4	95.2	92.7	82.0	75.3	91.0	90.0	100.0	96.2	98.3	99.3	95.0	100.0	85.4	90.8	89.6	87.8	93.0	91.6	84.8	97.8	99.4	96.0

Figure 5.9: Transpose of confusion matrix for Flavia dataset using GLC. First row is the class number, last row is the accuracy of that class.

5.4 Preprocessing

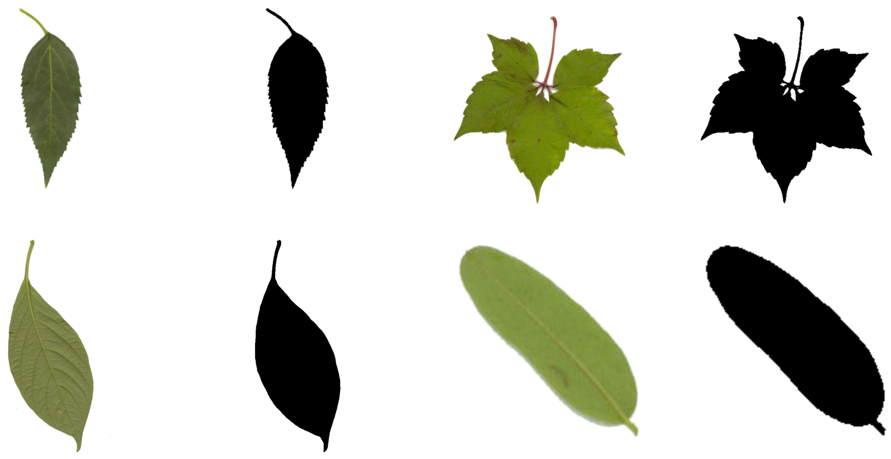
There are different preprocessing methods that we use in our systems. In this section, effects of these preprocessing steps are reported.

5.4.1 Segmentation

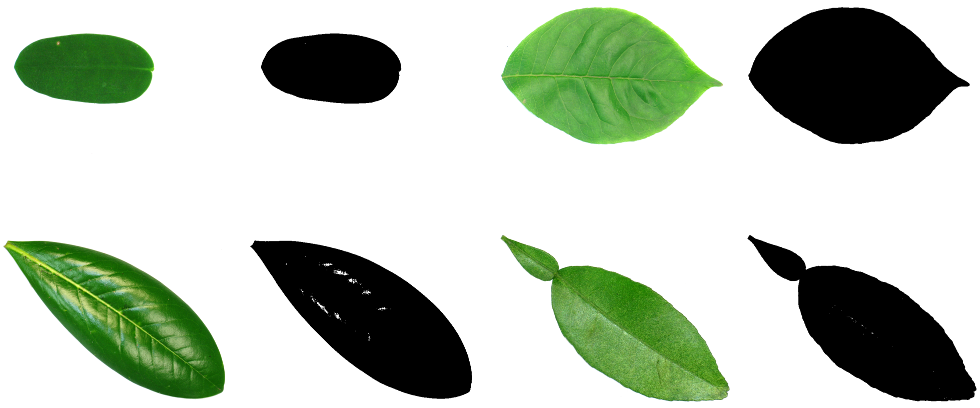
One of the most important parts of a classification system that uses geometric information is segmentation. If segmentation fails and produces incorrect results, it is not possible for the system to perform properly. This step depends on the dataset used. We use the segmentation algorithm that is described in Subsection 4.1.1 for ICL and Flavia datasets. Since the background of leaves varies in Leafsnap dataset, we use segmentation method described in [1].

The segmentation algorithm that we developed produces better results if the backgrounds have fixed color. Results of this segmentation algorithm are shown in Figure 5.10 for all datasets. The noise removal algorithm can easily deal with the noise inside and outside the leaf blade. However, noise on the margin cannot be eliminated. Therefore, this algorithm is not suitable for Leafsnap dataset where the shadows cause noise on the leaf margin.

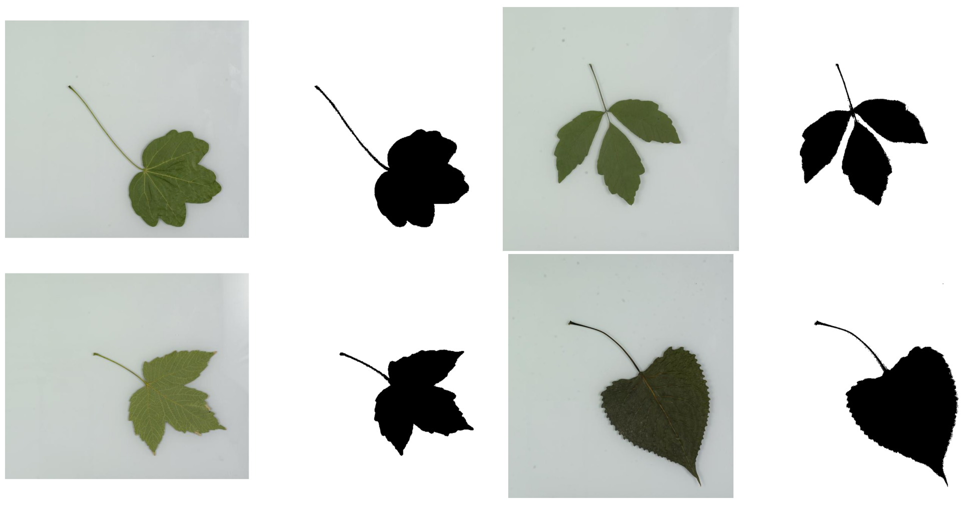
The iterative selection method [1] is more suitable for variable backgrounds as it determines threshold iteratively. However, in plain backgrounds using the segmentation method that we developed results in much better segmentation. The results of this segmentation algorithm is presented in Figure 5.11.



(a) ICL

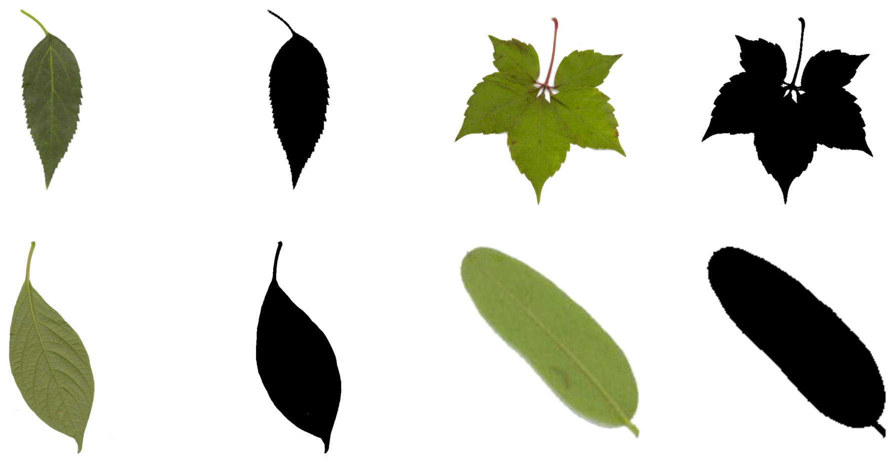


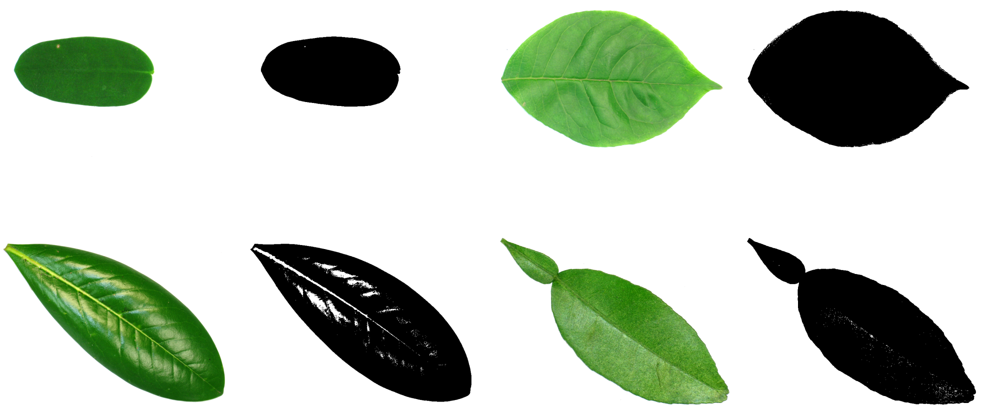
(b) Flavia



(c) Leafsnap

Figure 5.10: Results of the proposed segmentation algorithm


(a) ICL


(b) Flavia

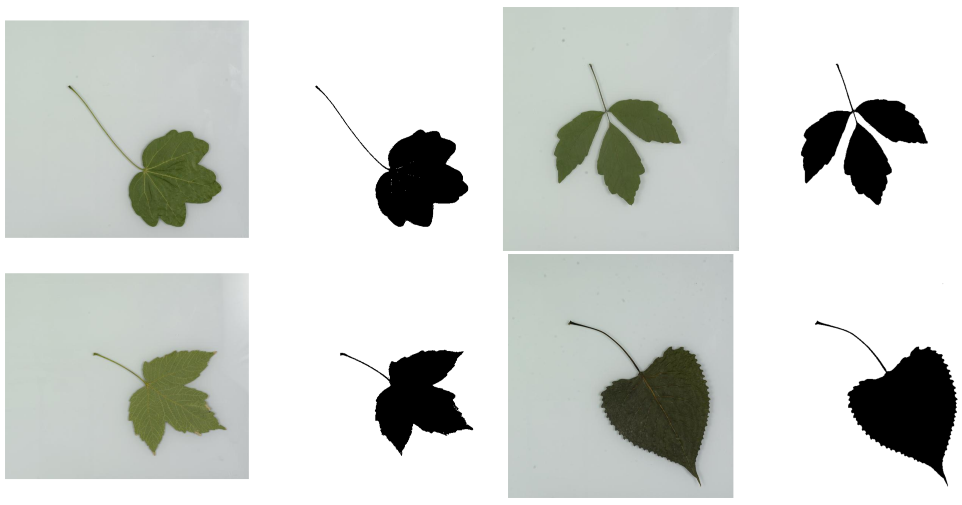

(c) Leafsnap

Figure 5.11: Results of the segmentation algorithm proposed in [1]

Table 5.6: Effect of stalk removal on classification performance

	ICL		Flavia		Leafsnap	
Method	GLC	GTCLC	GLC	GTCLC	GLC	GTCLC
Proposed	74.0	85.6	93.4	98.2	68.3	74.5
[1]	72.7	84.7	81.1	93.8	72.3	77.9

Additionally, accuracy improvement using suitable segmentation method is listed in Table 5.6. It is clear that using a suitable segmentation algorithm is one of the most important factors in these classification systems.

5.4.2 Noise removal

It is possible to get a noisy result from segmentation. This is true even if the most fitting segmentation algorithm is used for the task. This noise is caused by highlights on the leaf or dirt on the acquisition device. Additionally, Gaussian noise might be caused by poor lighting conditions. The noise in the segmentation can cause great disturbance in geometric features. It is most destructive on convexity features, inscribed and circumscribed circles. However, using noise removal removes internal features on complex leaves. The first and the second images in Figure 5.12 are examples to this situation.

In Figure 5.12 noise removal operation on segmented images are shown. Additionally, Table 5.7 shows the effect of noise removal on classification accuracy. Since MCH can exploit internal structure of the leaf, it does not benefit from the noise removal. MDM works on contour information, which generally is not affected by noise, therefore, it does not have any significant change in the result. Our proposed methods contain edge related features which are adversely affected by noise, therefore, noise removal improves their performance.

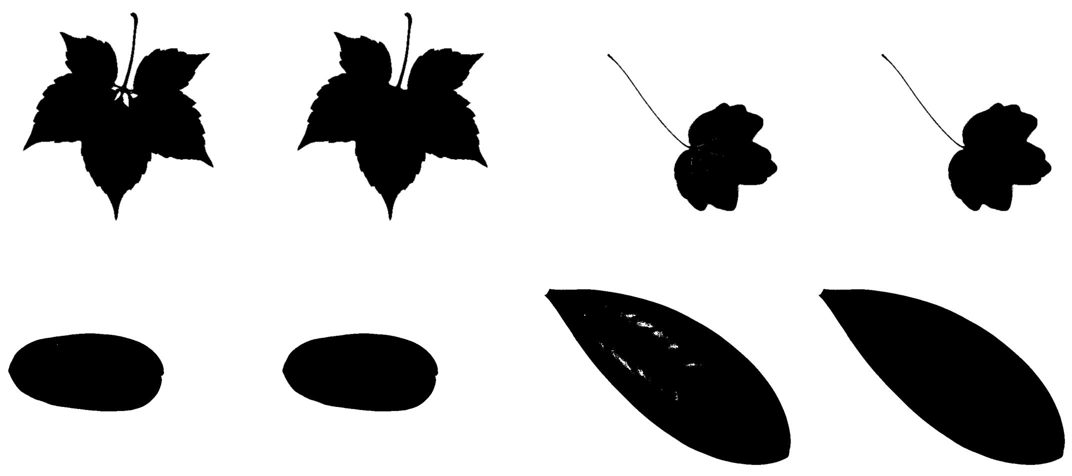


Figure 5.12: Results of noise removal operation. First image in pairs are the segmentation result, whereas other is the result of noise removal.

Table 5.7: Accuracy improvement of noise removal

	Dataset	MCH	MDM	GLC	GTCLC
With noise removal	ICL	53.8	66.7	74.0	85.6
	Flavia	68.4	88.7	93.4	98.2
	Leafsnap	54.8	77.8	72.3	77.9
Without noise removal	ICL	62.6	68.7	72.5	84.5
	Flavia	77.9	88.3	91.8	97.6
	Leafsnap	64.0	76.8	65.9	69.2

5.4.3 Stalk removal

Stalk removal is required in some scenarios where the stalks of some leaves are missing in the same class. Stalks dilute some geometric features, however, if they are intact, the length of the stalk is a useful information as different types of leaves have different stalk lengths. During our initial experiments, we included stalk removal, however, after identifying and optimizing segmentation methods, experiments show that using stalk removal actually hurts the performance. The results of these experiments are summarized in Table 5.8. Results also show that there is still room for improvement

Table 5.8: Effect of segmentation on classification performance

	ICL		Flavia		Leafsnap	
Stalk removal	GLC	GTCLC	GLC	GTCLC	GLC	GTCLC
With	72.3	84.8	92.9	97.7	70.4	76.3
Without	74.0	85.6	93.4	98.2	72.3	77.9

in our stalk removal algorithm as Flavia dataset does not contain any stalks, yet its performance is adversely affected by it.

5.5 Features

The primary aim of this thesis is to identify the features and feature sets that work well together. This aim requires experimentation with the combination of different feature sets. In this section, the results of these experiments are demonstrated. In addition to the features that we use in the proposed systems, we reported the experimental results of the features that are not used to demonstrate the reason why they are excluded. These experiments are performed on ICL dataset only. Results of these experiments are reported in Table 5.9 and Table 5.10. The first table (Table 5.9) contains performance results while using feature set alone, GTCLC and GLC with and without the respective feature set. Additionally, number of features that are in the feature set and whether it has been used in the proposed methods are reported. Table 5.10 is the accuracy report of the combination of different features.

In total there are 7 commonly used geometric features described in Section 3.4. We use 4 of these features in our systems. These are perimeter ratio, area convexity, perimeter convexity and circularity. In Table 5.9, selected geometric features represent the features we selected from these geometric features.

Table 5.9: Detailed comparison of different features

	Used in		Number of		With		Without	
Name	GLC	GTCLC	Features	Alone	GLC	GTCLC	GLC	GTCLC
Selected	yes	yes	4	26.5	74.0	85.6	72.4	84.7
All	no	no	7	37.6	74.0	85.5	74.0	85.6
Moment inv.	yes	yes	8	12.8	74.0	85.6	70.4	83.4
Margin dist.	yes	yes	1	8.8	74.0	85.6	73.3	85.1
Margin stat.	yes	yes	4	10.4	74.0	85.6	73.0	85.1
MDM-16	no	no	112	54.6	71.0	84.7	74.0	85.6
MDM-24	yes	no	264	61.6	73.7	85.6	70.4	72.7
MDM-32	no	yes	480	64.7	74.0	85.2	50.2	85.6
MDM-48	no	no	1104	65.7	73.8	70.9	74.0	85.6
MDM-64	no	no	1984	63.9	71.5	81.5	74.0	85.6
SRLBP	yes	no	80	34.1	73.9	85.6	74.0	75.1
SLBP	no	no	256	29.2	74.0	84.9	74.0	85.6
RILBP	no	no	36	27.1	74.1	76.6	74.0	85.6
ULBP	no	no	10	22.1	74.0	53.1	74.0	85.6
Color	no	yes	2	8.2	75.4	85.6	74.0	85.1

Table 5.10: Contribution of the features that are used in the proposed GTCLC system

Feature sets	Accuracy
SMDM-24	61.6%
Geometric	38.2%
Geometric + Moments	50.2%
SLBP	34.0%
SLBP + Color	37.2%
SMDM-24 + Geometric	69.1%
SMDM-24 + Geometric + Moments	73.7%
SMDM-24 + SLBP	78.7%
SMDM-24 + Geometric + Moments + Color	75.2%
SMDM-24 + SLBP + Color	79.7%
Geometric + Moments + SLBP	71.4%
Geometric + Moments + SLBP + Color	72.8%
SMDM-24 + Geometric + Moments + SLBP	84.9%
All	85.6%

5.6 Classifiers

During this thesis, we experimented with different classifiers. However, after a series of experiments, it was clear that LDC performs better than the standard classifiers we experimented on. Therefore, we use LDC for our classification systems.

This section contains experiments that compare well-known classifiers using the feature descriptors we developed. The classifiers used for comparison are Linear Discriminant Classifier (LDC), Nearest Mean Classifier (NMC), and K-Nearest Neighbors Classifier (KNN). Table 5.11 shows the results of these experiments. Classifiers that are marked with ^{1} use Euclidean Distance whereas the ones that are marked with ^{2} use Manhattan distance. In addition to these classifiers, the accuracy of selecting a random class for the classification result is also presented for reference.

5.7 Computational Cost

A classification system should be efficient and fast to be useful in real life scenarios. In some cases, the speed of computation becomes critical. For instance, a leaf classifica-

Table 5.11: Comparison of common classification methods using the proposed descriptors.

	ICL		Flavia		Leafsnap	
Classifier	GLC	GTCLC	GLC	GTCLC	GLC	GTCLC
LDC	74.0	85.6	93.4	98.2	72.3	77.9
KNN (k=1) [2]	72.1	83.8	88.6	96.2	71.9	74.9
KNN (k=3) [2]	66.3	82.5	89.3	96.1	65.9	66.6
KNN (k=1) [1]	71.6	83.5	90.6	96.0	68.2	70.5
KNN (k=3) [1]	69.6	83.0	90.2	95.8	59.4	61.8
NMC [1]	3.3	1.0	7.5	7.1	5.1	5.7
Random	0.45		3.13		0.76	

Table 5.12: Comparison of the proposed systems with the state-of-the-art algorithms in terms of computational performance on ICL dataset

Method	Features	Classifier	Total classification (includes training, seconds)	Classification (per sample, msec)
MCH	15	MCH	258	1.31
MDM	1984	kNN + MMC	1397	5.51
GLC	497	LDC	211	0.17
GTCLC	363	LDC	128	0.13

tion can be used in a factory that processes plant leaves for medical use. The decision process for eliminating unwanted leaves should work as fast as the conveyor belt not to create any bottlenecks in the production. Therefore, we performed experiments on the computational performance of our systems. The results of these experiments are displayed in Table 5.12.

5.8 Discussion on Experimental Results

We performed many experiments to demonstrate the value of the proposed systems. According to these experiments the proposed GTCLC method has the highest accuracy while the proposed method GLC has an acceptable accuracy compared to the state-of-the-art methods. However, it is clear that the accuracy alone cannot measure the value of system. Therefore, we performed more experiments for the suitability of the proposed methods. These include computational speed, per-class accuracy, statistical accuracy analysis, and confusion matrix. In all these experiments, GTCLC achieves the best results.

In addition to the experiments performed on systems, we performed experiments on the individual parts of our systems. These experiments show that the features and preprocessing steps used in the proposed methods cannot be refined further.

Chapter 6

CONCLUSION

In this thesis, we devoted our attention to leaf classification using various descriptors. We proposed two leaf classification systems. The first method, namely Geometric Leaf Classification (GLC), uses multiple geometric descriptors. The second proposed system, namely Combination of Geometric, Texture, and Color Features for Leaf Classification (GTCLC), uses geometric, texture, and color features to describe plant leaves. Linear Discriminant Classifier (LDC) is employed for classification as multiple descriptors require class-based prioritizing classifier.

We developed and identified features that work well with each other. Additionally, we eliminated the features that do not contribute to the final result. Elimination is important as the increasing number of features increases the complexity of the classification system leading to the well-known problem called curse of dimensionality. This problem causes occasional failures during training.

The features that we combined are complementary in nature, they do not have a significant overlap. These features define the leaf semantically. Every feature set used in the proposed systems have a well-defined meaning. For instance, MDM is included to define the shape of the leaf. When features that have overlapping information are combined, the overall accuracy could be adversely affected. After all, there is no additional information that can be gained to increase accuracy. For instance, selected geometric features do not perform (26.5%) as good as all geometric features (37.6%) when used alone. However, when combined with the rest of the features, all geometric features have slight negative impact compared to the selected features. On the other hand, inclusion of color features, which have a very low stand-alone accuracy of 8.2% improves the end result by 0.5%.

The research we performed for this thesis improved the state-of-the-art significantly. When we started our research, the highest scoring method was MDM, with the accuracy of 68% on ICL dataset. Our final system delivers 85.5% accuracy on the same dataset.

In addition to the systems that we introduced, we have proven that LDC is a suitable classifier for leaf classification. LDC not only outperforms other classifiers in terms of accuracy, but also in terms of computational efficiency. The only drawback of LDC is that it requires number of training samples to be more than the number of features. In real life scenarios, it is possible to increase the number of samples for the system to work. Therefore, the only problem that it has is during experiments that involves limited datasets.

Finally, we introduced new features that have specific tasks. Margin distance and margin statistics are developed to describe leaf margins. Sorted LBP is developed to define venation pattern of the leaf. These distinct and meaningful descriptors allow our systems to distinguish leaves that cannot be distinguished using generic shape descriptors.

There are still possible improvements for the proposed GTCLC system. First of all, new and better features can be employed to describe a leaf. Even though the current features result in a high accuracy system, there might be features that can result in even higher accuracy. For instance, in [10], a more complete system for margin description is proposed. This descriptor might yield a higher accuracy when combined with the rest of our system.

The methods presented here are not optimized. Their parameters are determined by logical examination. It is possible to obtain a system that results in a better classification accuracy by optimizing parameters of the methods employed. General optimization methods such as genetic algorithms could be used to optimize parameters of the system.

We believe it would be possible to reach better classification results by using classifier ensembles. Some descriptors, such as color, are not linearly separable. Therefore, performing LDC on these descriptors cannot yield the best result. However, using a non-linear classifier like QDC, requires more training samples and may cause overfitting when the entire feature space is considered. The solution to these problems can be found by using different classifiers for different feature sets and combining their results with a suitable score level fusion. It is also possible to use feature selection methods to further reduce number of features. Additionally, using a more suitable feature fusion method might improve results.

REFERENCES

[1] Riddler, T. W. & Calvard, S. (Aug 1978) . Picture thresholding using an iterative selection method. *Systems, Man and Cybernetics, IEEE Transactions on*, 8(8):630–632.

[2] Hu, R., Jia, W., Ling, H., & Huang, D. (2012) . Multiscale distance matrix for fast plant leaf recognition. *IEEE Transactions on Image Processing*, 21(11):4667–4672.

[3] Wu, S. G., Bao, F. S., Xu, E. Y., Wang, Y., Chang, Y., & Xiang, O.-L. (Dec. 2007) . A leaf recognition algorithm for plant classification using probabilistic neural network. In *IEEE International Symposium on Signal Processing and Information Technology*, pages 11–16.

[4] Rejeb Sfar, A., Boujemaa, N., & Geman, D. (2015) . Confidence sets for fine-grained categorization and plant species identification. *International Journal of Computer Vision*, 111(3):255–275.

[5] Sulc, M. & Matas, J. (2015) . Texture-based leaf identification. In *Computer Vision - ECCV 2014 Workshops*, volume 8928 of *Lecture Notes in Computer Science*, pages 185–200. Springer International Publishing.

[6] Kalyoncu, C. & Toygar, Ö. (2015) . Geometric leaf classification. *Computer Vision and Image Understanding*, 133:102 – 109.

[7] Yanikoglu, B., Aptoula, E., & Tirkaz, C. (2014) . Automatic plant identification from photographs. *Machine Vision and Applications*, 25(6):1369–1383.

[8] Novotný, P. & Suk, T. (2013) . Leaf recognition of woody species in central europe. *Biosystems Engineering*, 115(4):444 – 452.

[9] Mouine, S., Yahiaoui, I., & Verroust-Blondet, A. (2013) . A shape-based approach for leaf classification using multiscaletriangular representation. In *Proceedings of the 3rd ACM Conference on International Conference on Multimedia Retrieval*, ICMR '13, pages 127–134, New York, NY, USA. ACM.

[10] Cerutti, G., Tougne, L., Coquin, D., & Vacavant, A. (2014) . Leaf margins as sequences: A structural approach to leaf identification. *Pattern Recognition Letters*, 49(0):177 – 184.

[11] Larese, M. G., Namías, R., Craviotto, R. M., Arango, M. R., Gallo, C., & Granitto, P. M. (2014) . Automatic classification of legumes using leaf vein image features. *Pattern Recognition*, 47(1):158 – 168.

[12] Zhang, S., Lei, Y., Dong, T., & Zhang, X. (2013) . Label propagation based supervised locality projection analysis for plant leaf classification. *Pattern Recognition*, 46(7):1891 – 1897.

[13] Gandhi, A. (2002) . Content-based image retrieval: Plant species identification. Master's thesis, Oregon State University.

[14] Gu, X., Du, J., & Wang, X. (2005) . Leaf recognition based on the combination of wavelet transform and gaussian interpolation. In *Advances in Intelligent Computing*, volume 3644 of *Lecture Notes in Computer Science*, pages 253–262. Springer Berlin Heidelberg.

[15] Wang, X.-F., Du, J.-X., & Zhang, G.-J. (2005) . Recognition of leaf images based on shape features using a hypersphere classifier. In *Advances in Intelligent Computing*, volume 3644 of *Lecture Notes in Computer Science*, pages 87–96. Springer Berlin Heidelberg.

[16] Du, J.-X., Wang, X.-F., & Zhang, G.-J. (2007) . Leaf shape based plant species recognition. *Applied Mathematics and Computation*, 185(2):883–893.

[17] Wang, X.-F., Huang, D., Du, J.-X., Xu, H., & Heutte, L. (2008) . Classification of plant leaf images with complicated background. *Applied Mathematics and Computation*, 205(2):916 – 926. Special Issue on Advanced Intelligent Computing Theory and Methodology in Applied Mathematics and Computation.

[18] Cope, J. S., Remagnino, P., Barman, S., & Wilkin, P. (2010) . Plant texture classification using gabor co-occurrences. In *6th International Conference on Advances in Visual Computing (ISVC)*, pages 669–677.

[19] Beghin, T., Cope, J. S., Remagnino, P., & Barman, S. (2010) . Shape and texture based plant leaf classification. In *Advanced Concepts for Intelligent Vision*

Systems, volume 6475 of *Lecture Notes in Computer Science*, pages 345–353. Springer Berlin Heidelberg.

[20] Cerutti, G., Tougne, L., Mille, J., Vacavant, A., & Coquin, D. (2013) . Understanding leaves in natural images – a model-based approach for tree species identification. *Computer Vision and Image Understanding*, 117(10):1482 – 1501.

[21] Wang, B. & Gao, Y. (2013) . Fast and effective retrieval of plant leaf shapes. In *Computer Vision – ACCV 2012*, volume 7725 of *Lecture Notes in Computer Science*, pages 475–486. Springer Berlin Heidelberg.

[22] Lei, Y., Zou, J., Dong, T., You, Z., Yuan, Y., & Hu, Y. (2014) . Orthogonal locally discriminant spline embedding for plant leaf recognition. *Computer Vision and Image Understanding*, 119(0):116 – 126.

[23] Nam, Y. & Hwang, E. (2005) . A shape-based retrieval scheme for leaf images. In *Advances in Multimedia Information Processing - PCM 2005*, volume 3767 of *Lecture Notes in Computer Science*, pages 876–887. Springer Berlin Heidelberg.

[24] Belhumeur, P., Chen, D., Feiner, S., David W. Jacobs, W. Kress, H. Ling, I. Lopez, R. Ramamoorthi, S. Sheorey, & S. White. (2008/// 2008) . Searching the world's herbaria: A system for visual identification of plant species. In *Computer Vision–ECCV 2008*, pages 116 – 129.

[25] Bama, B. S., Valli, S. M., Raju, S., & Kumar, V. A. (Apr 2011) . Content based leaf image retrieval (cblir) using shape, color and texture features. *Indian Journal of Computer Science and Engineering*, 2(2):202–211.

[26] Kumar, N., Belhumeur, P. N., Biswas, A., Jacobs, D. W., Kress, W. J., Lopez, I. C., & Soares, J. V. B. (2012) . Leafsnap: A computer vision system for automatic plant species identification. In *Computer Vision – ECCV 2012*, Lecture Notes in Computer Science, pages 502–516. Springer Berlin Heidelberg.

[27] Nam, Y., Hwang, E., & Kim, D. (2008) . A similarity-based leaf image retrieval scheme: Joining shape and venation features. *Computer Vision and Image Understanding*, 110(2):245–259.

[28] Park, J., Hwang, E., & Nam, Y. (May 2007) . Utilizing venation features for efficient leaf image retrieval. *Journal of Systems and Software*, 81(1):71–82.

[29] Hervé, G., Bonnet, P., Joly, A., Boujemaa, N., Barthélémy, D., Molino, J., Birnbaum, P., Mouysset, E., & Picard, M. (September 2011) . The imageclef 2011 plant images classification task. In *ImageCLEF 2011*, Amsterdam, Pays-Bas.

[30] Feng, H. Y. F. & Pavlidis, T. (May 1975) . The generation of polygonal outlines of objects from gray level pictures. *Circuits and Systems, IEEE Transactions on*, 22(5):427–439.

[31] Hu, M. (February 1962) . Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187.

[32] Flusser, J. (2000) . On the independence of rotation moment invariants. *Pattern Recognition*, 33(9):1405 – 1410.

[33] Flusser, J. & Suk, T. (Dec 2006) . Rotation moment invariants for recognition of symmetric objects. *Image Processing, IEEE Transactions on*, 15(12):3784–3790.

[34] Ling, H. & Jacobs, D. W. (Feb 2007) . Shape classification using the inner-distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(2):286–299.

[35] Scholkmann, F., Boss, J., & Wolf, M. (2012) . An efficient algorithm for automatic peak detection in noisy periodic and quasi-periodic signals. *Algorithms*, 5(4):588–603.

[36] Ojala, T., Pietikainen, M., & Maenpaa, T. (Jul 2002) . Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):971–987.

[37] Guo Z., Zhang, D., & Zhang, D. (June 2010) . A completed modeling of local binary pattern operator for texture classification. *Image Processing, IEEE Transactions on*, 19(6):1657–1663.

[38] Liu, L., Zhao, L., Long, Y., Kuang, G., & Fieguth, P. (2012) . Extended local binary patterns for texture classification. *Image and Vision Computing*, 30(2):86 – 99.

[39] Guo, Z., Zhang, L., & Zhang, D. (2010) . Rotation invariant texture classification using {LBP} variance (lbpv) with global matching. *Pattern Recognition*, 43(3):706 – 719.

[40] Liu, L., Fieguth, P., Clausi, D., & Kuang, G. (2012) . Sorted random projections for robust rotation-invariant texture classification. *Pattern Recognition*, 45(6):2405 – 2418.

[41] ICL leaf image dataset. http://www.intelengine.cn/dataset/index.html.

[42] Xu, E. Y. Bao, F. S. (2009) . Flavia leaf database. http://flavia.sourceforge.net/.

[43] Du, J., Huang, D., Wang, X., & Gu, X. (2005) . Shape recognition based on radial basis probabilistic neural network and application to plant species identification. In *Advances in Neural Networks – ISNN 2005*, volume 3497 of *Lecture Notes in Computer Science*, pages 281–285. Springer Berlin Heidelberg.