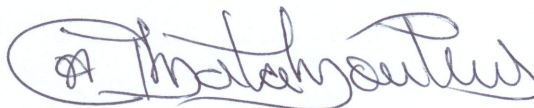


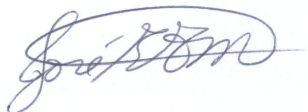
APROBACIÓN DE LA TESIS

“A Texture and Curvature Bimodal Leaf Recognition Model for Costa Rican Plant Species Identification”

TRIBUNAL EXAMINADOR



Dr. Erick Mata Montero
Profesor Asesor



PhD. José Enrique Araya Monge
Profesor Lector



Msc. Manuel Vargas Del Valle
Profesor Externo



Dr. Roberto Cortés Morales
Coordinador del Programa
de Maestría en Computación

Dicembre, 2014



TEC

Instituto Tecnológico de Costa Rica

COMPUTER ENGINEERING DEPARTMENT
MASTER IN COMPUTER SCIENCE

A Texture and Curvature Bimodal Leaf
Recognition Model for Costa Rican Plant
Species Identification

Thesis
for

Magister Scientiæ in Computer Science

AUTHOR:
Jose Carranza

ADVISOR:
Erick Mata, Ph.D.

December 2014

Abstract

In the last decade, research in Computer Vision has developed algorithms to help botanists and non-experts classify plants based on images of their leaves. Nevertheless, very few efficient tools have resulted from that research and have actually been used in the field. The most popular system to date is LeafSnap. It is considered a state-of-the art leaf recognition mobile application. It uses a multi scale curvature model of the leaf margin to classify leaf images into species. LeafSnap was applied to 184 tree species from Northeast US and achieved high levels of accuracy for that group of trees. In this document, we extend the research that led to the development of LeafSnap along several lines. First, LeafSnap's underlying algorithms are applied to a set of species from Costa Rica. Then, texture is used as an additional criteria in order to improve the level of accuracy of LeafSnap's original algorithms. Thus, the main goal of this research is to measure the level of improvement in automatic Costa Rican tree species identification achieved when texture analysis is added to the curvature model of margins of leaves. Our results confirm our hypothesis since the level of improvement reaches a 0.168 for the Costa Rican clean subset, and 0.431 for the Costa Rican noisy subset. In both cases, our results show this increment as statistically significant.

Resumen

En la última década, la investigación en Visión por Computadora ha generado algoritmos para ayudar a botánicos y personas no expertas a clasificar especies de plantas con base en las imágenes de sus hojas. Sin embargo, pocos algoritmos han resultado en herramientas eficientes que hayan sido usadas en el campo. El sistema más popular a la fecha es LeafSnap, considerado el estado del arte en aplicaciones móviles de reconocimiento de hojas. Utiliza un modelo multi escala de curvatura para clasificar imágenes de hojas en sus respectivas especies. LeafSnap fue aplicado a 184 especies de árboles del noreste de Estados Unidos, alcanzando altos niveles de exactitud para ese grupo reducido de árboles. En este documento, extendemos en varios aspectos la investigación que llevó al desarrollo de LeafSnap. Primero, los algoritmos que conforman a LeafSnap internamente, son aplicados a un grupo de especies de Costa Rica. Además la textura de las hojas es utilizada como un criterio adicional para mejorar el nivel de exactitud del modelo de curvatura de LeafSnap. Por tanto, el objetivo principal de esta investigación es medir el nivel de mejora en la identificación automática de especies de plantas de Costa Rica resultante de agregar análisis de la textura al modelo de curvatura del margen de las hojas. Los resultados obtenidos confirman nuestra hipótesis ya que el nivel de exactitud mejora hasta en un 0.168 para el subconjunto de datos limpio, y hasta en 0.431 para el subconjunto de datos con ruido. En ambos casos, nuestros resultados muestran que el incremento es estadísticamente significativo.

To my lovely wife Bárbara, for supporting me unconditionally during all this time. To my advisor, Prof. Erick Mata, for his wise guidance and for sharing all his research expertise with me. To Nelson Zamora and INBio, for all their help during the data acquisition process. To my mother, a woman who has always been a life example to me. To nature, which is the best technology we have.

Without you all, this research would not have been possible. Thank you.

Contents

1	Introduction and General Background	1
1.1	Introduction	1
1.2	Theoretical Framework	3
1.2.1	Leaf Image Acquisition	3
1.2.2	Image Enhancement	6
1.2.3	Leaf Image Segmentation	8
1.2.4	Leaf Feature Extraction	10
1.2.5	Species Classification based on Leaf Images	17
1.3	Related Work	19
1.4	Problem Description	27
1.5	Hypothesis	27
2	Objectives and Contributions	28
2.1	General Objective	28
2.2	Specific Objectives	29
2.3	Contributions	29
2.4	Scope and Limitations	30
3	Methodology	31
3.1	Introduction	31
3.2	Software	32
3.3	Leaf Recognition and Classification Process	34
3.3.1	Leaf Image Acquisition	35
3.3.2	Image Leaf Segmentation: Color Based	41
3.3.3	Image Enhancements/Post-Processing	44
3.3.4	Leaf Feature Extraction	49
3.3.5	Species Classification based on Leaf Images	56
3.4	Experiment Design	57

4	Results and Discussion	65
4.1	Introduction	65
4.2	The Flavia Experiment	66
4.3	The Accuracy of HCoS on Costa Rica And LeafSnap Incomplete Dataset Experiment	70
4.4	Experiment of Texture and Curvature Models in Costa Rica	72
4.4.1	Measuring Significance of the Accuracy Increase	76
5	Conclusions and Future Work	81
5.1	Conclusions	81
5.2	Future Work	83
	Bibliography	86
	Acronyms	92

List of Figures

1.1	Commonly defined leaf recognition phases	4
1.2	Canadian poplar	5
1.3	LeafSnap species	6
1.4	Herbarium sample	7
1.5	Basic Leaf Morphological Features	11
1.6	Venation by Hit Miss Transformation (HMT)	13
1.7	LeafSnap curvature arch length and area features with radio r	15
1.8	Local Binary Pattern (LBP)	16
1.9	Fast Retina Keypoint Descriptor (FREAK)	17
1.10	Leaf Segmentation	21
1.11	Venation Pattern Masks for Unconstrained Hit-or-Miss Transform (UHMT)	24
3.1	Leaf press	39
3.2	A <i>Robinsonella lindeniana</i> var. <i>divergens</i> sample	39
3.3	A <i>Bauhinia unguolata</i> sample	40
3.4	INBio Herbarium samples	41
3.5	Hue Saturation Value (HSV) decomposition of a leaf image	42
3.6	Segmented Samples	43
3.7	Top Hat Transformation	47
3.8	Clipping of a <i>Coccoloba floribunda</i> sample	48
3.9	<i>Croton niveus</i> contours	51
3.10	A discrete filled disk of radius=4 pixels	52
3.11	A discrete unfilled disk of radius=5 pixels	52
3.12	Area disk applied	53
3.13	Local Binary Pattern Variance (LBPV) patterns of <i>Croton draco</i> sample	55
3.14	Process of extracting LBPV	56
4.1	Individual models against <i>Flavia</i>	67
4.2	Combined models against <i>Flavia</i>	69

4.3	Histogram of Curvature over Scale (HCoS) Vs Combined Methods against Costa Rican clean dataset	74
4.4	HCoS Vs Combined Methods against Costa Rican noisy dataset	74
4.5	HCoS Vs Combined Methods against the complete Costa Rican dataset	75

List of Tables

3.1	List of species collected at the Sabana Park	37
3.2	List of species collected at the INBiopark	38
3.3	Variants of LBPV	54
3.4	Models used in the experiments	58
4.1	Individual models against Flavia	66
4.2	Combined models against Flavia	67
4.3	Comparison of obtained accuracies on Flavia	68
4.4	Accuracy of our HCoS implementation over the LeafSnap Lab subset and Costa Rican Clean subset	71
4.5	Accuracy of our HCoS implementation over the LeafSnap Field subset and Costa Rican Noisy subset	72
4.6	Increase of accuracy when combining curvature and texture	73
4.7	Proportion Test results over the Costa Rican Clean Subset	77
4.8	Proportion Test results over the Costa Rican Noisy Subset	78
4.9	Proportion Test results over the Costa Rican Complete Dataset	80

CHAPTER 1

Introduction and General Background

Each species on our planet plays a role in the healthy functioning of natural ecosystems, on which humans depend.

William H. Schlesinger

1.1 Introduction

In the last decade, research in Computer Vision has produced algorithms to help botanists and non-experts classify plants based on images of their leaves [1; 3; 4; 5; 20; 23; 30; 41; 47; 48]. However only a few studies studies have resulted in efficient systems that are used in the field. The most popular system to date is LeafSnap [24]. It is considered a state-of-the-art mobile leaf recognition application that uses an efficient multiscale curvature model to classify leaf images into species. LeafSnap was applied to 184 tree species from Northeast USA, resulting in a very high precision method for species recognition for that region, and has been downloaded by more than 1 million users [24]. LeafSnap has not been applied to identified trees from tropical countries such

as Costa Rica. The challenge of recognizing tree species in biologically rich regions is expected to be considerably bigger.

Vein analysis is an important, discriminative element for species recognition, that has been used in several studies before [10; 25; 26; 27; 29]. According to Nelson Zamora, curator of the herbarium at the National Biodiversity Institute (INBio), and one of the most renowned in Costa Rica, venation is as important as the curvature of the margin of the leaf when classifying species in Costa Rica.

To our knowledge, with exception of the work of Wijesingha and Marikar [47], herbarium specimen images have not been used for automated species identification. The collection of vascular plants at the Herbarium INBio consists of 160,068 records. The collection includes 327 families, 2,259 genera and 9,087 species. All of these plants are dried for preservation, which we think can be exploited as a dataset given that veins have a tendency to stand out once the leaf is dry [49].

This document is a thesis for a MSc degree in Computer Science to study the accuracy of a leaf recognition model based not only on the curvature of the leaf margin, but also on the venation present in its texture. This is the first attempt to create such model in Costa Rica. We must state that we would have liked to extract only venation instead of the whole texture. However, automatic venation extraction turns to be such a complex problem by itself additional research would be needed to solve it. Furthermore, the time frame given for this thesis would not allow us to satisfactorily address this problem, thus we use an approach where venation is highly important and captured, but other additional characteristics such as reflections are also captured.

The rest of this document is organized as follows: Section 1.2 describes the theoretical framework of the proposed research. It focuses on the five main phases involved in classifying plants based on leaf images. Section 1.3 presents relevant related work. Section 1.4 summarizes the problem description and Section 1.5 states the hypothesis of this research. Section 2 presents the objectives, main contributions, scope and

limitations of the proposed research. Section 3 focuses on methodological aspects and experiment descriptions, Section 4 explains the results obtained from the experiments. Finally, Section 5.1 covers the conclusions and future work.

1.2 Theoretical Framework

Computer Vision is the discipline of extracting information from images, opposite of Computer Graphics [18]. There is an overlap with Image Processing regarding basic techniques, and some authors use both terms interchangeably. The main goal of Computer Vision is to create models and extract data and information from images, while Image Processing is about applying transformations to the images, such as sharpening, contrasting, among others [18]. The following section explains the different Computer Vision techniques and studies related to leaf recognition to date.

Most authors divide the leaf recognition process in into five phases [4; 24; 30; 48]: leaf image acquisition, image enhancement, image segmentation, leaf feature extraction, and similarity search for species identification. Figure 1.1 depicts these five phases and their corresponding products.

1.2.1 Leaf Image Acquisition

Datasets

Existing datasets for leaf recognition typically use images of individual leaves on a uniformly colored background, which makes it easier to detect the leaf. For instance, the authors of LeafSnap [24] use Expectation-Maximization (EM) [12] to cluster pixels into leaf/non leaf categories, but this can only be done if the image is already prepared with a uniform background.

There are some public datasets for leaf recognition, but none has become a universally accepted standard by the Computer Vision community yet. The following are the

Leaf Image Acquisition
<ul style="list-style-type: none"> • Fresh leaf images with uniform background
Image Enhancement
<ul style="list-style-type: none"> • Images with noise and artifacts removed • Images converted to other color domains
Leaf Segmentation
<ul style="list-style-type: none"> • Leaf pixels separated from background pixels
Leaf Feature Extraction
<ul style="list-style-type: none"> • Curvature • Shape • Texture • Veins • Color • Morphological characteristics
Similarity Search / Species Classification
<ul style="list-style-type: none"> • Supervised Machine Learning Classification (kNN, PNN, SVM)

Figure 1.1: Commonly defined leaf recognition phases
Most studies agree with these phases [1; 3; 4; 5; 20; 23; 24; 30; 41; 47; 48]

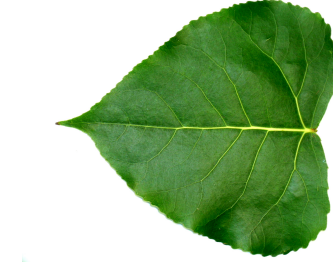


Figure 1.2: Canadian poplar
Taken from Wu et al. [48]

most widely used:

- The Swedish Dataset [44] consists of 15 species with 75 images each. Images are aligned to reduce rotation. Only one side of the leaf is captured. This image dataset offers high inter-species similarity.
- The Flavia Dataset [48] comprises 32 species with 3,621 leaf images with white backgrounds. This dataset uses fresh leaf images. Leaves were sampled from the Nanjing University campus and the Sun Yat-Sen Arboretum, Nanking, China. Figure 1.2 shows a sample of the dataset.
- ImageCLEF 2013 [16] includes 250 species of herbs and trees from France. It includes images of leaves, flowers, fruits, stem and the whole plant. It comprises both images with white background and images taken directly in the field (fresh sample leaves) with complex backgrounds, lighting and noise [16]. In previous years other similar datasets were also available [14; 15].
- The LeafSnap dataset consists of images from 184 tree species from Northeastern USA. It includes 23,916 images of fresh leaves with white backgrounds [24]. Figure 1.3 shows leaves of all the species available in the dataset.



Figure 1.3: LeafSnap species
Taken from Kumar et al. [24]

Herbaria

According to Nelson Zamora, the collection of vascular plants at the INBio herbarium consists of 160,068 records, corresponding to 327 families, 2,259 genera and 9,087 species [49]. A sample from the collection is shown in Figure 1.4.

To our knowledge, few studies have created their dataset directly from herbaria. Wijesingha and Marikar [47] used 79 images of the *Stemonoporus* genus, obtained from the National Herbarium at the Royal Botanic Garden, Sri Lanka. The images have rather low resolution (120 x 120 pixels), and the dataset is small.

1.2.2 Image Enhancement

Once images are acquired, the next phase consist of preprocessing the image to enhance important features [46]. This step includes gray scale conversion [26; 27], noise reduction and other color domain conversions [24]. The goal is to delete undesired noise and distortions that may affect the following phases [4; 21].

Hue Saturation Value (HSV) Conversion In LeafSnap [24], the image is converted to the HSV domain to obtain enhanced pixel definitions for the leaf, since it is shown better on the saturation and value domains, however hue is discarded. Wijesingha and Marikar [47] also convert the image to the saturation domain to reveal textures more clearly.



Figure 1.4: Herbarium sample
Taken from González [17]

Gray scale Conversion Converting the image to gray scale intensities is very common, since most of the feature extraction methods work better on gray scale [1; 3; 20; 23; 25; 28; 30; 34; 39; 41]. To convert a Red Green Blue (RGB) image to gray scale the following function can be used [48]:

$$gra = 0.2989 * R + 0.5870 * G + 0.1140 * B \quad (1.1)$$

Larese et al. [25] worked on gray scale because their research did not contemplated color at all, just venation.

Leaf/Non-Leaf Validation Some studies also developed mobile apps during their research, to provide users the option to capture a leaf image in real time [20; 24; 34]. By allowing the users to upload images, another problem arises which consists of detecting if the image has a leaf inside at all, to avoid further processing for invalid

images. Thus, authors have trained classifiers such as Support Vector Machines (SVM) to detect if there is a leaf inside the captured image. In LeafSnap [24] a low dimensional representation of the scene using A low dimensional representation of the scene, which does not require any form of segmentation (GIST) descriptors is used as feature set. A total of 5,972 manually labeled leaf images were used to train the SVM classifier.

1.2.3 Leaf Image Segmentation

Once the dataset is created, the images cleaned, and the noise is filtered out, the next step is to extract the leaf from the image [24]. Most studies deal with clean leaf images, with uniform backgrounds, and use color clustering techniques to extract the leaf.

Thresholding A binary image is the result of converting a normal image to a two value image (normally black and white, or in general two colors only) [26; 27]. Several studies use techniques that require binary images, especially for shape and vein extraction [25]. Equation 1.2 shows how a binary function works:

$$B(x, y) = \begin{cases} 0 & \text{if } f(x, y) \leq T \\ 255 & \text{if } f(x, y) > T \end{cases} \quad (1.2)$$

Where $B(x, y)$ and $f(x, y)$ are the intensity values of the gray scale image and the binary image, respectively, at position (x, y) , and T is the threshold value. This technique is also used by Larese et al. [25] to extract the leaf shape.

Expectation-Maximization (EM) It is an iterative algorithm used to find the maximum likelihood estimates of parameters when the model depends on unobserved latent variables [12]. It clusters the pixels into two groups: leaf pixels and non-leaf pixels [12; 24]. Applied to leaf segmentation in [24], it is used with the following mathematical

model:

$$p(x|\theta) = 1/2p(x|\mu_f, \Sigma) + 1/2p(x|\mu_b, \Sigma) \quad (1.3)$$

Where $p(x|\mu_f, \Sigma)$ and $p(x|\mu_b, \Sigma)$ are normal distributions, μ_f represents the foreground distribution (the leaf itself), and μ_b represents the background distribution. A common shared covariance matrix Σ is used. Each normal distribution has an equal weight of $1/2$ assigned.

Graph-Cut Very few studies deal with a leaf dataset that has complex backgrounds or uncontrolled conditions such as light, noise, and other objects inside the image. Several problems arise from complex background images [43], such as:

- On interactive systems such as mobile apps, the algorithms must be fast.
- Compound leaves are extremely difficult to segment due to their complex segmentation boundaries.
- Leaf images present natural variations in lighting, which creates shadows that add noise.

Soares and Jacobs [43] study how a semi-controlled light environment affects traditional clustering algorithms to extract the leaf from the image given that it produces undesired shadows. They opt to perform color clustering, like on their previous work [24], followed by a Graph-Cut (also known as Min-Cut), which finds the global optimal segmentation solution. It is based on the idea that that a graph can be partitioned in two disjoint sets by simply removing edges connecting them both. It is important to note that this method does not deal with any uncontrolled scenario, but only with semi-controlled lighting settings.

1.2.4 Leaf Feature Extraction

Once leaf images are clean and leaf pixels have been extracted, the next step is to extract a feature set of the leaf that is discriminative enough to determine its species. Most of the feature extraction techniques mentioned in literature are based on different morphological characteristics by using Computer Vision techniques [4; 24; 30; 46; 48]. Some use shapes or contour [24; 48], others textures [4; 46; 48], and even color [48] and veins [25; 26; 27].

Leaves in general present at least two advantages over flowers and fruits when it comes to automatic species classification. First of all, they are two dimensional [5]. Secondly, they are available during all year [46].

Because we use texture extraction in this research, we must note the "texture" has different meanings in Botany and Computer Vision. In Botany, texture can be defined as the physical texture of the leaf, meaning how it feels against tact [49]. In Computer Vision however, the definition of texture differs since it is the pattern of non-uniform spatial distribution of different imaging intensities [3; 46]. In this research, we use the Computer Vision definition, as we are looking for pixel patterns and relations between them.

Morphological Descriptors They are calculated from gray levels of the image after a gray scale conversion [2; 5; 20; 26; 27; 41; 48]. The following is a list of the most common (see Figure 1.5).

- Leaf Diameter: The longest distance between any two points of the leaf's margin, denoted as D .
- Physiological Length: The distance between the two terminals of the main vein named L_p . Normally it is selected manually by a human on a semi-automatic fashion.

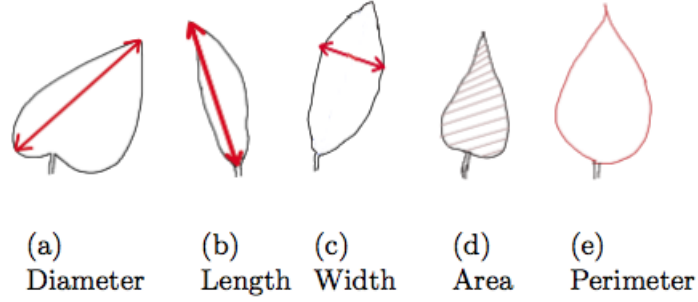


Figure 1.5: Basic Leaf Morphological Features
from Arora et al. [2]

- Physiological Width: The longest line orthogonal to the Physiological Length, denoted as W_p .
- Leaf Area: The pixel count of the leaf area A [26; 27; 48].
- Leaf Perimeter: The count P of pixels of the leaf margin.
- Aspect Ratio: L_p/W_p .
- Form Factor: The difference between the leaf and a circle, given by $4\pi A/P^2$.
- Rectangularity: The similarity between the leaf and a rectangle, given by $(L_p W_p)/A$.
- Narrow Factor: the ratio of the diameter D and length L_p , thus D/L_p .
- Circularity: The ratio involving area A of the leaf and the square of its perimeter P given by A/P^2 .
- Solidity: The ratio between A the area of the leaf and A_{ch} the area of a convex hull given by A/A_{ch}

Morphological Dilation and Erosion Both are the basic operators of mathematical morphology, typically applied to binary or gray scale images. Dilation attempts to gradually enlarge the boundaries of regions of foreground pixels, causing the foreground

areas to grow in size [18]. It uses a structuring element or kernel to determine the thickness. Erosion is the opposite of Dilation, its effect is to erode away the boundaries of the foreground pixels [18].

Morphological Opening Consists on applying erosion first to the image A given a structuring element B . Then, using the same structuring element B , dilation is applied to the result:

$$A \circ B = (A \ominus B) \oplus B \quad (1.4)$$

Where \ominus is erosion and \oplus is dilation [18].

In leaf feature extraction, morphological opening is performed on a gray scale image with flat, disk-shaped structuring element of variable radius [20; 23; 41; 48]. It is used as a feature closely related to the veins. Wu et al. [48] use disks of 1, 2, 3 and 4 radius. Remaining areas are named as A_{v1} , A_{v2} , A_{v3} and A_{v4} respectively to each radius size. Then, the authors obtain 4 additional vein features: A_{v1}/A , A_{v2}/A , A_{v3}/A and A_{v4}/A .

Hit Miss Transformation (HMT) Is a mathematical morphology operator that allows extracting all pixels that have a similar foreground and background neighbor configuration [13]. The original operator works over binary images in order to differentiate foreground from background. Larese et al. [25] used it to detect vein patterns of the leaf (see Figure 1.6). The next Equation 1.5 shows how it is calculated:

$$A \odot B = (A \ominus C) \cap (A^c \ominus D) \quad (1.5)$$

where A is the image, A^c is the complement of A , C and D are structuring elements which satisfy that $C \cap D = \emptyset$. The pair $B = (C, D)$ is also known as composite structuring element [13]. Erosion \ominus is used as the main operator.

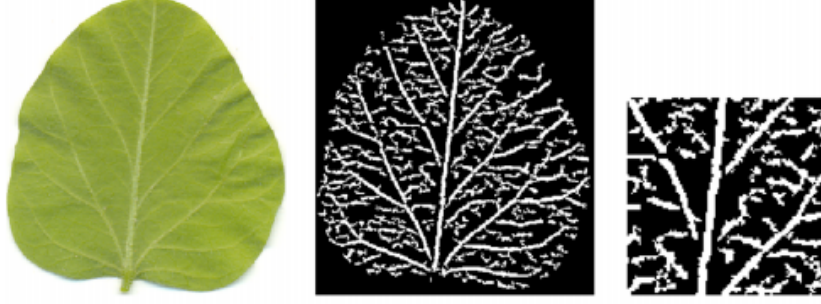


Figure 1.6: Venation by HMT
Taken from Larese et al. [25]

Color Moments Provide a measurement for similarity between images using invariant global features [20]. In case of gray scale images, let x_i be a random variable which denotes the gray levels of an image and L the number of distinct gray levels. We define the probability P of occurrence of pixels with intensity x_i , k as the number of pixels with gray level x_i , and N as the total number of pixels in the region [3; 5; 20; 23]:

$$P(x_i) = k/N \quad (1.6)$$

Then we can calculate the following gray scale moments:

- Mean: averages the gray levels of the image $\mu_0 = \sum_{i=1}^L x_i P(x_i)$
- Variance: $\mu_2 = \sum_{i=1}^L (x_i - \mu_0) P(x_i)$
- Standard Deviation: $\sigma = \sqrt{1/L \sum_{i=1}^L (P(x_i) - \mu_0)^2}$
- Skewness: $\mu_3 = \sum_{i=1}^L (x_i - \mu_0)^2 P(x_i)$

It should be noticed that this is not limited to gray scale, but images could also be generalized to any number of color channels [20].

Lacunarity Measures how fractals fill space, where patterns having bigger gaps tend to result in higher lacunarity values. It is also calculated for each gray level on the

image and used by several studies as a venation descriptor [23; 46].

Keypoints / Salient Points Salient points are defined as those which stand out from an image [32; 33]. This landmark points could be boundary points or salient points internally within the shape, like the ones from the veins. Salient points can be used to describe the leaf boundary, or to represent the spatial correlation between them and the leaf margin [32].

Filtering Image filters are useful to extract certain interesting parts of the image, such as boundaries. Some of the filters used previously on leaf feature extraction include Gabor filters [4], Laplacian [48] and Sobel filters [4]. However, a lot more filters exist in current Computer Vision literature [18].

Centroid Distance Another way to get features related to the contour is to calculate the distances from a centroid of the leaf, to each pixel of the contour [26; 27]. A centroid can be defined as:

$$C(x, y) = C(1/N \sum_{n=1}^N x_n, 1/N \sum_{n=1}^N y_n) \quad (1.7)$$

Where $C(x, y)$ is the centroid coordinate of the leaf region and N is the number of pixels of the leaf margin [26; 27]. Then the distance can be calculated by using:

$$D(i) = \sqrt{|C_x - E(i)_x|^2 + |C_y - E(i)_y|^2} \quad (1.8)$$

Where $D(i)$ is the distance between the centroid of the leaf and the i^{th} leaf contour pixel, C_x and C_y are the coordinates of the centroid, and $E(i)_x$ and $E(i)_y$ are the coordinates of the contour pixel.

Curvature The current mobile state-of-the-art application, LeafSnap [24], represents the leaf shape using multiscale curvature measures. Two type of invariant integral

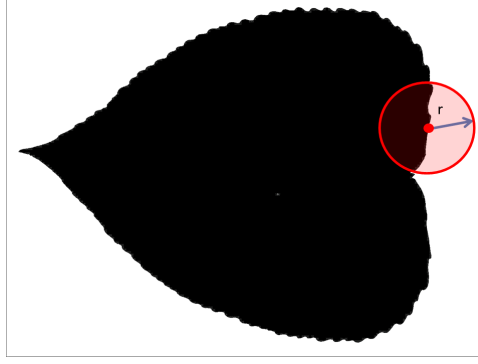


Figure 1.7: LeafSnap curvature arch length and area features with radius r . The figure shows a disk around a pixel from the margin of the leaf. 2 feature sets are taken, the internal area measure (the piece of the circle that intersects the leaf area) and the arclength (the length of the arc that intersects the leaf area). This is calculated for each pixel from the margin, for 25 different radius scales.

measures are used, the area of intersection of a disk centered at each contour point, and the arclength, which is the fraction of the disk's perimeter inside the contour. This is calculated for 25 different scales.

Figure 1.7 shows how the curvature algorithm works by going pixel by pixel calculating the area and the arclength of the intersection of a radius r circle and the leaf area, centered on pixel p . Once both feature vectors are computed, a histogram with 21 bins is calculated for each feature set. This is done for 25 different radius and concatenated together afterwards forming a HCoS [24]. This mechanism has proven promising on the LeafSnap mobile application, thanks to its speed, multiscale and rotation invariant approach.

Local Binary Pattern (LBP) They are known for their invariance to local gray scale variations, and their high descriptive power [3; 39]. The image is assumed to be formed by micro patterns such as spots, lines, flat areas and edges, and its value is calculated based on the circular neighborhood of P pixels of gray levels q_p and radius

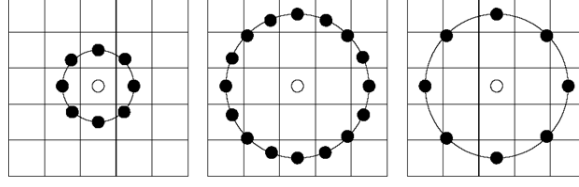


Figure 1.8: LBP

(8,1), (16, 2) and (8, 2) circular pixel neighborhoods. (8,1) means 8 pixels from the nearest neighbors are used to calculate the binary number. (16, 2) means 16 pixels from the second neighborhood level are used. (8, 2) means 8 pixels from the second neighborhood level are used. Taken from Kadir et al. [23].

r around a central pixel of gray value q (refer to Figure 1.8).

$$LBP(P, r) = \sum_{p=1}^{P-1} s(q_p - q_c) 2^p \quad (1.9)$$

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } otherwise \end{cases} \quad (1.10)$$

The $s(x)$ function shown in Equation 1.10 is known as the thresholding function which is similar to Equation 1.2 [39]. LBPV is a variant of LBP that is rotation invariant [20; 39]. For multiscale LBP some authors such as Herdiyeni [19] extract LBP descriptors at different radius scales, calculate a histogram for each radius feature set, and then concatenate those histograms together, similar to the curvature HCoS on LeafSnap [24]. This is a technique useful in complex image backgrounds, uncontrolled lighting, noise [19], and may be useful for vein extraction.

SIFT-like Algorithms Once located, keypoints need to be converted to feature vectors used to describe the image. Scale-Invariant Feature Transform (SIFT) uses a 128-dimensional vector from a grid of histograms of oriented gradient. Its high descriptive power and robustness to illumination change have ranked it as the reference keypoint descriptor for the past decade. Other similar algorithms have been created such as Speeded Up Robust Features (SURF), Fast Retina Keypoint Descriptor (FREAK),

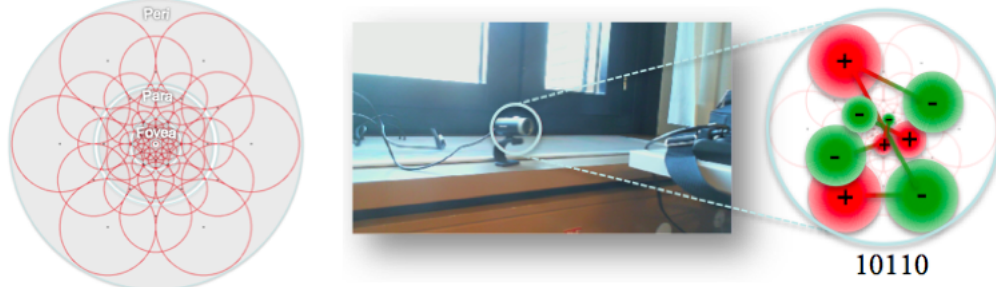


Figure 1.9: Fast Retina Keypoint Descriptor (FREAK)
FREAK descriptor of an object on a generic scene. Taken from Ortiz [37]

and Oriented FAST and Rotated BRIEF (ORB) [37]. Nguyen et al. [34] used SURF features to develop an Android application for mobile leaf recognition, similar to LeafSnap [24]. SURF features are extracted from the gray scale image of the leaf. The focus was neither the margin nor the internal veins, but the whole leaf as a texture. Figure 1.9 shows the intuition behind the FREAK descriptor.

Feature Normalization When several feature sets are used, it is necessary to normalize the feature vectors as a precaution since the feature values may vary in different ranges. In absence of feature normalization, features with larger values would have stronger influence on the cost function from the classifier [23].

1.2.5 Species Classification based on Leaf Images

The following algorithms have been used to classify species by several authors, based on leaf images.

Probabilistic Neural Network (PNN)

A PNN is derived from a Neural Network (NN) that uses a Radial Base Function (RBF) that scales a variable nonlinearly [48]. PNN is adopted because of several advantages: its training speed is several times faster than a normal NN, it can approach a Bayes optimal result under certain conditions, and it is robust to noise examples [45].

On a PNN, the network is not "trained" traditionally since node weights are just assigned. Existing weights will not be re-assigned, but new weight vectors are inserted into the weight matrices when training happens, so it can be used in real time. Since PNN are implemented using matrices, makes them very fast and useful for leaf recognition [4; 19; 20; 23; 30; 48]. The classification is done by calculating which class has the maximum probability of being the correct one.

k Nearest Neighbors (kNN)

A very common and fairly simple classifier is kNN. In leaf recognition several studies used kNN with very good results [3; 5; 24]. This algorithm measures the distance from a query item to each of the training set items, getting the k training set items that are the nearest [8]. In order to measure the distance, several metrics have been proposed, but the most common one is the Euclidean one:

$$d(x_r, x_s) = \left[\sum_{i=1}^p c_i (x_{ri} - x_{si})^2 \right]^{\frac{1}{2}} \quad (1.11)$$

Where x_r is the reference item, x_s the query item, p the feature vector and c_i is an optative weight factor used to change the weight of each variable. It is sensitive to the amount of training elements, making it slower if too many samples exist, and also particularly sensitive to the Curse of Dimensionality problem. One possible solution to both problems is the use of Locality Sensitive Hashing (LSH) where the feature set is mapped to a hash number. This has been used in leaf recognition as well [33]. Another option is to reduce dimensionality to the most representative features, using techniques like Principal Component Analysis (PCA) [41; 48].

Support Vector Machines (SVM)

A SVM classifier is a binary supervised learning algorithm that can recognize patterns to classify an item into one of two classes [6]. The items can be pictured as points

in space, mapped so the items of each class are divided by a clear gap (called hyperplane) that is as wide as possible. New items then, are predicted to belong to a class based on which side of the gap they fall on. A data point can be viewed as a p -dimensional vector, and the goal is to separate several points by using a $(p - 1)$ -dimensional-hyperplane that maximizes the margin between two classes (known as maximum-margin hyperplane). Several leaf recognition-related studies have used SVM for both leaf feature classification into species [3], but also to do a previous validation in cases where the image may not include a leaf [20; 24; 34]. The precision achieved by these studies are used by current state-of-the-art mobile systems such as LeafSnap [24].

1.3 Related Work

This section recaps previous work on leaf recognition using Computer Vision techniques. It comprises several works applying techniques for leaf feature extraction and image classification.

Shape and Margin

Mouine et al. [33] used local descriptors associated with margin sample points to create triangles based on those points. These landmark points are N points distributed uniformly around the shape, and each point p_i is represented by N_s triangles computed at different scales, with a distance $d(k)$ between the triangle points. They used 4 types of descriptors: Triangle Area Representation (TAR) is based on triangle areas, robust to noise and provides information about local concavities, Triangle Oriented Angles (TOA) describes the inner angles of the triangles, Triangle Side Lengths (TSL) the side lengths, and finally, Triangle Side Lengths and Angle (TSLA) is a combination of side lengths and angles (this last one is the most robust one) [33]. The similarity search was done using LSH and kNN, projecting the feature vectors to hash functions for faster search.

On the Flavia [48] dataset, the best obtained result was a 69.93% of precision using kNN with the TSLA feature set. On the ImageCLEF 2011 dataset [14], TSL had the best results with a 80% precision reported. They also used the Swedish leaf dataset [33; 44], where their TSLA methods achieved an impressive 96.53% of precision. As future work, the authors mentioned that adding venation may improve the precision of the model.

Mouine et al. [32] studied the leaf margin and the leaf salient points using two shape descriptors, one to describe the leaf boundary and another to represent the spatial correlation between salient points and the leaf margin. The salient points, however, are not vein points always, so the authors mention how the model could be improved by developing an specific detector of key points of the venation network and use them as salient points. The reported precision on the ImageCLEF 2011 dataset was 78.5% [14], and on the ImageCLEF 2012 dataset was 58% [15].

In LeafSnap [24] the authors created a leaf classification method based on unimodal curvature features and similarity search using kNN. This method was tested against a dataset built from North American trees, using 183 species in total. The methodology was based on 4 phases:

- Discard non-leaf images: they used a trained SVM to detect if an image contained a leaf. GIST features taken from the image were used for leaf detection, implemented using LibSVM¹, an open source SVM library from [9]. The quantity of manually labeled leaf images used to train the SVM was 5,972 in total.
- Color-Based Segmentation: since their system required images to have a uniform background, leaf segmentation worked by estimating the foreground and background color distributions, and then classifying each pixel at a time into one of those two categories. A conversion to HSV color domain was applied before using Expectation-Maximization (EM) [12] for the leaf segmentation. Refer to Figure

¹<http://www.csie.ntu.edu.tw/~cjlin/libsvm/>

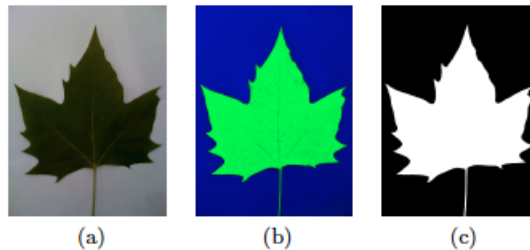


Figure 1.10: Leaf Segmentation

a-) a leaf on a uniform background, b-) the same image on the HSV color domain and c-) the result of applying EM to cluster the pixels. Taken from Kumar et al. [24].

1.10 for the color-based segmentation steps.

- **Curvature Feature Extraction:** the leaf shape was effectively represented using multiscale invariant integral curvature measures. Two measures were used, the area of intersection of a disk centered at the contour point, and the the arclength, which is a fraction of the disk's perimeter inside the contour. 25 disks with different radius were generated at each point for multiscale support. Finally a histogram was computed at each scale and then concatenated together to form a Histogram of Curvature over Scale (HCoS).
- **Similarity Search:** kNN was used over the resulting HCoS of the images.

It is important to note that the work of LeafSnap resulted on the first mobile state-of-the-art application for leaf recognition, with more than 1 million reported users in the U.S [24].

Texture

Arun et al. [3] used morphological features such as mean, variance, skewness and standard deviation of the gray levels of the image. The authors used Gray Tone Spatial Dependency Matrix (GTSDM) and Local Binary Pattern (LBP) to do classification over medicinal plants as well. The objective of the study was to obtain the best combination of features for automatic classification of medicinal plants [3]. The authors used a total

of 6 different classifiers and tested out all the possible combinations, having SVM and kNN among them. A dataset was also created by them. It is a small dataset with only 5 different species and a total of 250 images, which doesn't confirm if their method is robust for more general proposes, even when a 94.7% of precision was reported [3].

Herdiyeni and Kusmana [19] used LBP features to classify medicinal and house plants from Indonesia. They extracted LBP descriptors from different sample points and radius, calculated a histogram for each radius length feature set, and concatenate those histograms together, similar to the curvature HCoS of LeafSnap [24]. As a classifier a 4 layer PNN with a RBF was used. Their dataset consists of 30 species, 1,440 images from tropical plants, and 30 species with 10 images for each species from house plants. It is important to note the image background of the medicinal plants is uniform, but house plant images have non-uniform backgrounds. For medicinal plants the precision reported was 77% and for house plants 86.67%, revealing that using LBP for complex image backgrounds is a suitable technique.

Nguyen et al. [34] used SURF features to develop an Android application for mobile leaf recognition, similar to LeafSnap [24]. First, GIST descriptors from the gray scale image were calculated from the leaf pixels and a SVM was trained to recognize them, with a 95% of precision. This is used to detect if a picture has or doesn't have a leaf, to prevent users to upload a leaf-less image. For the species classification task, SURF features were extracted from the gray scale image of the leaf, not from the leaf itself. The feature set was reduced to histograms in order to reduce dimensionality since the resulting SURF feature vector may be too big. The precision reported was 95.94% on the Flavia dataset [48].

Venation

Lee and Hong [26]; Lee et al. [27] proposed and implemented a leaf recognition system using the major vein of the leaf. They calculated a total of 21 features includ-

ing the distance between a centroid and all the points from the contour, followed by Fast Fourier Transform (FFT), and other morphological geometric features of the leaf. The image was converted to gray scale, and then to a binary image using a threshold conversion in order to get the leaf contour. The extraction of the vein was done by performing opening operations. They obtained the gray scale image and the image of performed opening operation (erosion and dilation), and then calculated the leaf veins image by converting the difference image to binary. Once the veins were extracted, they calculated two histograms of the veins, one horizontal and one vertical. Additionally, 4 geometrical features were extracted from the leaf: aspect ratio, form factor, rectangularity, length and width. The reported precision was 97.19% on the Flavia dataset [48]. As future work, they wanted to focus on working leaf contour extraction methods for complex backgrounds.

Larese et al. [25] discarded completely color, texture, size and shape, and focused only on venation. Leaf vein segmentation was performed using Unconstrained Hit-or-Miss Transform (UHMT) over gray scale leaf images using pixel masks (see Figure 1.11). Three classes of legumes were recognized: a soybean (*Glycine max* (L) Merr), and red and white beans (*Phaseolus vulgaris*). The beans belong to the same species, presenting similar leaf shapes, but different veins. Color was discarded by converting the image to gray scale, then in order to segment the veins, UHMT was computed for different sized versions of the image intended to highlight different levels of vein detail. A central patch (Figure 1.11) was selected manually from the resulting vein image using LeafGUI² [40], since shape of the leaf was not important for this study. Based on the extracted patch, 35 additional morphological measures were computed. Using 10 independent runs of 10-fold cross validation, the reported precision of the model with SVM was 87%, over 866 RGB leaf images.

Li et al. [28] studied how to extract venation from the leaf to develop an interactive

²<http://www.leafgui.org/>

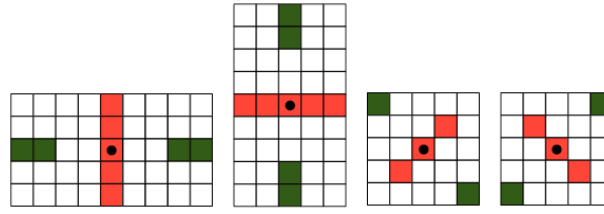


Figure 1.11: Venation Pattern Masks for Unconstrained Hit-or-Miss Transform (UHMT)

Taken from Larese et al. [25]

tool for botanists. The study used 21 kind of tree images to generate randomly 50,000 12×12 pixel patches to train an Independent Component Analysis (ICA) algorithm. Based on ICA, each image patch can be represented by a linear combination of basis patches, which are really the vein features.

Multimodal Models

Wu et al. [48] used morphological, easy to extract features with PNN to recognize plant species. This study also produced the Flavia dataset [48], mentioned in many other studies. In total, 12 feature types were extracted. First the image was converted to gray scale from RGB, then a boundary enhancement was applied by a Laplacian filter of 3×3 spatial mask in order to get the margin of the leaf. Over the extracted margin, the leaf diameter, length, width, area, perimeter, smooth factor, aspect ratio, form factor, rectangularity, narrow factor, and morphological opening were calculated. By applying PNN after reducing dimensionality with PCA, 90% precision was reported.

Beghin et al. [4] proposed a feature fusion technique using Gabor Filters for edge and texture feature extraction. Their data used fresh leaf images, not dry species. The reported precision was in the range of 56.2% and 85.93%, using 10 fold cross validation. The dataset had 9 species and 15 sample images per species. The classification method used was PNN.

Kadir et al. [23] proposed a multimodal method which uses shape, vein and color features. This is one of the few studies where color played an important role. The

system in [23] merged the following features into a single model:

- Shape features: geometric features were extracted such as slimness, roundness and dispersion. Additionally, second order Polar Fourier Transform (PFT) was used because of its invariance to scaling and rotation [23].
- Color Moments: features of the RGB were captured such as mean, standard deviation, skewness and kurtosis, for each component of the RGB [23].
- Morphological Opening was used to extract 4 vein features [48]. It was performed on the gray scale image with flat, disk-shaped structure element of variable radius [23].
- Lacunarity [23; 46] was calculated for each RGB channel of the image.

After all features were extracted, feature normalization was applied. In absence of feature normalization, features with larger values would have stronger influence on the cost function from the classifier [23]. The classifier used was PNN, reporting a 93.75% of precision on the Flavia dataset [48].

Li et al. [29] created a combined vein and curvature model from 3-dimensional models of point cloud data from 4 images of leaves. They used 3-dimensional laser capturing technology to obtain the point cloud data, and a mesh algorithm to link the points, to calculate curvature and venation.

R.D and S [41] used a multimodal system composed of 38 morphological features and a PCA approach for texture. The morphological features are captured from a gray scale version of the image, with several morphological features such as: perimeter, circularity, aspect ratio, roundness, area, rectangularity, maximum length, maximum width, morphological opening, followed by a normalization process [23]. The PCA approach had a training phase which took all the dataset pictures and put them in a matrix, where a small number of characteristic features were generated, called eigenpictures. Then, each image was represented as a linear combination of these eigenpictures. Their

reported precision on the Flavia dataset [48] for the morphological features was 91.9%, for the PCA algorithm 85.4%, and for both combined 89.2%. One of the downsides of the PCA approach is that it requires training with the complete dataset, which is not memory efficient nor scalable.

Bhardwaj et al. [5] used morphological features such as aspect ratio, leaf area, rectangularity, circularity, convexity, solidity. They also used color moments for gray scale intensities such as mean, variance, kurtosis, skewness. Classification was done using kNN on a dataset of 320 leaves of 14 different plants from different sizes, colors and shapes, achieving a recognition rate of 91.5%.

Herdiyeni and Santoni [20] worked with a combination of shape, texture and color to recognize Indonesian medicinal plants. As a classifier they used PNN with a reported precision of 72.16% over 51 medicinal species, with a total of 2,448 images. The specific features used are:

- Morphological features such as leaf diameter, length, width, area, perimeter, aspect ratio, and morphological opening [23].
- Color Moments such as mean, standard deviation and skewness were used over the different color channels.
- LBPV was used to capture the texture of the leaf with rotation invariance [20; 39].

The authors created a mobile app which runs on Android OS called Medleaf [20]. Their best precisions were achieved by using LBPV as a feature base. Morphological features don't seem to contribute a lot to the precision of their model.

It is important to note that after studying different unimodal (one feature set models) and multimodal (2 or more feature set models) studies, it is not clear if using single curvature features is good enough (given the fact that the current state-of-the-art in [24] is unimodal), or if using a combination with veins is better.

1.4 Problem Description

Herbaria keep large amounts of dry mounted samples of plants to support scientific research. As noted by Nelson Zamora, once a leaf is dry and mounted, some botanical characteristics of the leaf such as color may be lost, but vein patterns are kept intact. Veins will also have a tendency to stand out even more than normal [49].

Additionally, the current state-of-the-art mobile leaf recognition application LeafSnap [24] may not be enough to classify with high precision two species that share similar margins, especially for tropical countries. LeafSnap [24] uses curvature of the margin of the leaf only, which seems to be enough for tree species from the Northeast area of the USA.

Furthermore, Costa Rica has approximately 12,000 plant species. The species identification process is manual, slow, tedious, and error prone. There have been pioneer attempts to automate the identification process with reduced domains [4; 24; 48], however none has been tested in Costa Rica. Mega-diverse regions such as the Neotropic need tools to identify both known and unknown species in an efficient, automatic or semi-automatic way, in order to better understand and conserve the world biodiversity.

1.5 Hypothesis

1. The current multiscale curvature model used by LeafSnap will not have the same high level of accuracy when recognizing Costa Rican species.
2. The accuracy of a leaf recognition model on species from Costa Rica increases significantly by adding texture pattern analysis to LeafSnap's curvature model.

CHAPTER 2

Objectives and Contributions

We share this planet with many species. It is our responsibility to protect them, both for their sakes and our own.

Pamela A. Matson

2.1 General Objective

Compare the results of applying LeafSnap’s curvature model with a model that comprises both leaf curvature and texture features of Costa Rican tree species.

2.2 Specific Objectives

1. Design and implement an efficient, rotation invariant, multiscale texture feature extraction model for leaf images.
2. Produce a Costa Rican dataset of dry leaf images from the herbarium at INBio that can be used for Costa Rican tree species identification systems.
3. Create another dataset of fresh Costa Rican leaf images that can be used to measure the accuracy of the curvature and texture based models.
4. Compare the accuracy obtained by LeafSnap in USA using the curvature model against the accuracy obtained by the same model with Costa Rican tree species.
5. Measure the positive effect of adding texture to the curvature model for Costa Rican species identification.
6. Create a backend system with the curvature and texture algorithms for future projects such as mobile apps for Costa Rican species recognition.

2.3 Contributions

This research will produce the following:

- A document that summarizes literature review of existing Computer Vision techniques used to classify plant species based on the morphological characteristics of their leaves.
- A Costa Rican dataset with leaf pictures and plant species information oriented to leaf recognition systems, taken from the herbarium at INBio.
- Another dataset comprised of fresh leaf images taken from field trips.
- Analysis of applying the state-of-the-art curvature model on Costa Rican species.

- An efficient, rotation invariant, multiscale leaf recognition model that uses both curvature and texture features for Costa Rican species recognition.
- A study of the accuracy when applying both curvature and texture on Costa Rican species.
- A backend system containing the dataset and the leaf recognition algorithms properly coded and implemented.

2.4 Scope and Limitations

This work will be based on the LeafSnap curvature model, which is considered the mobile state-of-the-art leaf recognition. We will add texture features to the model to see if it significantly improves its accuracy. Our premise is that Costa Rica's biodiversity is more complex and a curvature-only model performance may be significantly improved by adding texture information.

This research does not include:

- Using color features for the classification of species.
- Classifying other plant species besides the ones already selected from the herbarium at INBio.
- Working with high-noise images, with complex backgrounds or with bad resolution images.
- Segmenting complex background images.
- Working with other media type different than images.
- A mobile app for leaf recognition on Costa Rican species. This research however will focus on making fast algorithms so that the model can be eventually used as a backend for a mobile app.
- Any other deliverable not explicitly mentioned on this document.

CHAPTER 3

Methodology

Living wild species are like a library of books still unread. Our heedless destruction of them is akin to burning the library without ever having read its books.

John Dingell

3.1 Introduction

This chapter covers the techniques, tools and experiment design used in this research. Manual tasks such as image digitalization of fresh captured leaves from the field were required for the reference dataset. Automatic species identification algorithms needed several software development tasks. Experimental work to compare accuracy between Costa Rican species recognition and USA species recognition was designed and executed.

This chapter is organized as follows: Section 3.2 describes the software tools. Section 3.3.1 presents how images were acquired and the different datasets used and created.

Section 3.3.2 describes how leaf segmentation was achieved using color clustering. Section 3.3.3 describes which image enhancements were applied to improve segmentation. Section 3.3.4 explains which features were extracted for both curvature and texture. Section 3.3.5 covers the techniques used to classify the images into species. Finally, Section 3.4 focuses on experiments set up.

3.2 Software

In order to implement the LeafSnap model of curvature and the LBP algorithms, we made use of several open source libraries. By using existing implementations we reduced development time and improved the implementation’s robustness. The following is a list of the key libraries used for this research:

Photoshop CS6: We used Photoshop to manually clean up leaf images. We cleaned shadows, dust, and any undesired object or noise from all images. This process was applied to a portion of the dataset, not to everything, in order to facilitate leaf segmentation. This application is not open source.

StatSolver: An statistical application was used for the experiments. StatSolver¹ implements several statistical analysis methods, and in particular, the Proportion Hypothesis Test for 2 Samples that we needed to verify if there was a significant increase of accuracy when texture was added to curvature.

R: We used the scripting language R to create better looking charts for the results chapter.

¹www.statsolver.net

Python: We choose Python² as the programming language to develop the algorithms. The main reason for its selection is that its scientific community is highly active and a significant number of scientific libraries are available. Also, Python has web frameworks that can be used easily to develop web back-ends, which is desirable in order to create a web service for Costa Rican leaf recognition.

OpenCV: Computer Vision algorithms are key to this research. The OpenCV³ framework is an open source framework with key implemented Computer Vision algorithms that we needed for this research. The Expectation-Maximization (EM) algorithm allowed to cluster the image points into leaf and non-leaf clusters. Other morphology algorithms such as opening, closing, dilation, top-hat transformations, and contour calculations were used from this library as well [7].

SciPy: SciPy⁴ is a scientific library that we used for interpolations and connected components calculations. We calculated the connected components for post-processing and to improve segmentation [22; 24].

NumPy: An n-dimensional array library was needed in order to treat images as matrices. NumPy⁵ was the key library for image manipulation. It is very fast, which allows our algorithms to run with high computational efficiency, which is part of the desired features of the proposed model of curvature and texture. NumPy allowed us to clip images and extract specific pixels in several color domains, among other functionalities [36].

Mahotas: We proposed using LBP to improve the curvature model used by LeafSnap. A fast implementation of LBP can be found in the library called Mahotas⁶. This library

²www.python.org

³www.opencv.org

⁴www.scipy.org

⁵www.numpy.org

⁶luispedro.org/software/mahotas

has a LBPV implementation that is rotation invariant, very fast, and also offers the option to specify different pixel radii and circumferences, which was used to capture different scales in our approach [11].

Scikit-learn: Because of the potential big size of the datasets, we required a highly memory efficient k Nearest Neighbors (kNN) implementation for the experiments and the classification of leaves overall. Scikit-learn⁷ is a machine learning library in Python that has an extremely fast kNN implementation that we used to get the leaf species' ranking after classification. Scikit-learn contains several distance metrics that can be used for classification (e.g., Euclidean). In addition, it allows users to add their own distance code. In our case, we added the code to calculate the Histogram Intersection distance [24; 38].

Bottle: As part of the deliverables of this research, we provide a simple web service that receives images and returns the k best matching for the provided image. The Bottle⁸ library offers easy web service development on Python, that allows stateless implementation of the service.

3.3 Leaf Recognition and Classification Process

This section describes how the leaf recognition process was created. It comprises not only curvature and the texture models, but also explains the segmentation process and post-processing needed to improve the segmentation. Feature extraction is described for both the model of curvature and the texture model. The classification process is also detailed, which distance metric was used, and how accuracy was measured.

⁷www.scikit-learn.org

⁸www.bottlepy.org

3.3.1 Leaf Image Acquisition

A key element to this research was the acquisition of leaf images. Because for Costa Rica there was no suitable dataset with uniform backgrounds that we could use, so we built our own. Additionally, in order to measure the effectiveness of our bimodal curvature plus texture approach, we also required datasets to benchmark our findings. For benchmarking we used the Flavia [48] dataset and a subset of the LeafSnap [24] dataset, that contained 3,621 and 30,866 respectively.

Flavia Image Dataset

We used the Flavia Dataset [48] to compare our approach with other leaf recognition studies. We also used it to see how accurate our implementation of LeafSnap’s model of curvature was. The dataset comprises 32 species with 3,621 leaf images with white backgrounds. This dataset uses fresh leaf images. The origin of the sampled leaves is the Nanjing University campus and the Sun Yat-Sen Arboretum in Nanking, China [48].

The LeafSnap Image Incomplete Dataset

In order to verify how our implementation of LeafSnap’s model of curvature and texture model behaved, we were granted with access to a subset of the original LeafSnap dataset⁹. As stated on their website, we quote:

"The dataset released here doesn’t exactly match that used to compute results for the paper, nor the currently running version on our servers."

The provided dataset consists of leaf images of 185 tree species of northeast USA, divided in 2 subsets:

- Lab Subset: 23,147 pressed, high quality images taken from the Smithsonian collection. There are several images per species.

⁹<http://leafsnap.com/dataset>

- Field Subset: 7,719 typical images taken from mobile phones (mostly iPhone) in outdoor environments. These images are noisy images, since they contain blur, lighting conditions, shadows, among other artifacts.

Costa Rican Image Dataset

An image dataset of leaves from Costa Rica dataset was created from scratch. To our knowledge, no other suitable Costa Rican datasets existed before. The dataset has both clean and noisy images, aimed to see how the amount of noise affects the algorithms. Also, the Costa Rican dataset was used to evaluate our hypothesis. All images were captured from mainly two places: the Sabana Park, located in San Jose, and INBiopark, located in Santo Domingo, Heredia (Tables 3.1 and 3.2 show which species were taken from each place). The dataset includes endemic species as well as species in danger of extinction.

Reference "Clean" Subset Fresh leaf images were captured during field trips to both the Sabana and INBiopark. If the leaves were not flat enough, a press was used to flatten them for 24 hours. Figure 3.1 shows the leaf press used during this process. A total of 1468 leaf images were scanned and divided in different folders for each species. The images have a white uniform background and a size of 2548x3300 pixels, scanned at 300 dpi in JPEG. Photoshop CS6 was used to clean the image background from shadows, dust particles and other undesired objects. Figure 3.2 shows a sample of a cleaned Costa Rican leaf image of this subset. The scanner technical specifications are:

- Name: HP ScanJet 300.
- Maximum 4800 dpi. Only 300 dpi were used.

Testing "Noisy" Subset A total of 2345 fresh leaf images were captured during field trips. The images were divided into one folder per species. This subset was captured

Species	Scanned	Camera
Anacardium excelsum	18	40
Ardisia revoluta	20	40
Astronium graveolens	38	40
Bauhinia purpurea	16	37
Bauhinia unguolata	20	28
Brosimum alicastrum	20	40
Calophyllum brasiliense	18	43
Cedrela odorata	20	42
Cordia eriostigma	18	36
Dalbergia retusa		34
Dipterex panamensis	20	40
Guazuma ulmifolia	16	38
Hura crepitans	18	35
Hyeronima alchorneoides	18	32
Hymenaea courbaril	38	42
Manilkara chicle	24	41
Muntingia calabura	20	41
Platymiscium parviflorum	25	33
Platymiscium pinnatum	23	44
Quercus insignis		18
Samanea saman	32	46
Sideroxylon capiri	20	35
Simarouba glauca	28	35
Swietenia macrophylla	18	42
Tabebuia impetiginosa	20	38
Tabebuia ochracea	22	44
Tabebuia ochracea CR	16	20
Tabebuia rosea	20	20
Terminalia amazonia	24	42
Terminalia oblonga	22	42
Trichilia havanensis	32	44

Table 3.1: List of species collected at the Sabana Park
Contains the number of images per species.

3.3. LEAF RECOGNITION AND CLASSIFICATION PROCESS

	Species	Scanned	Camera
	<i>Acnistus arborescens</i>	20	27
	<i>Aegiphila valerioi</i>	16	28
	<i>Annona mucosa</i>	22	33
	<i>Blackea maurafernandesiana</i>	16	26
	<i>Calycophyllum candidissimum</i>	38	52
	<i>Cestrum tomentosum</i>	28	40
	<i>Citharexylum donnell-smithii</i>	20	24
	<i>Clusia croatii</i>	20	30
	<i>Coccoloba floribunda</i>	20	30
	<i>Colubrina spinosa</i>	20	30
	<i>Cretra costaricense</i>	14	24
	<i>Croton draco</i>	18	30
	<i>Croton niveus</i>	26	37
	<i>Dendropanax arboreus</i>	22	32
	<i>Dipterix Panamensis</i>	22	21
	<i>Erythrina poeppigiana</i>	20	30
	<i>Eugenia hiraefolia</i>	20	30
	<i>Ficus cotinifolia</i>	24	34
	<i>Genipa americana</i>	16	26
	<i>Guaiaacum sanctum</i>	34	34
	<i>Heliocarpus appendiculatus</i>	24	32
	<i>Ocotea sinuata</i>	18	28
	<i>Pachira quinata</i>	30	49
	<i>Persea americana</i>	16	26
	<i>Picramnia antidesma</i>	22	30
	<i>Pimenta dioica</i>	24	34
	<i>Posoqueria latifolia</i>	18	30
	<i>Psidium guajava</i>	16	24
	<i>Quercus corrugata</i>	20	30
	<i>Robinsonella lindeniana</i> var. <i>divergens</i>	20	28
	<i>Sapium glandulosum</i>	20	30
	<i>Simarouba glauca</i>	24	34
	<i>Solanum rovirosanum</i>	22	34
	<i>Stemmadenia donnell-smithii</i>	26	30
	<i>Tabernaemontana littoralis</i>	24	32
	<i>Terminalia amazonia</i>	20	24
	<i>Urera caracasana</i>	10	18
	<i>Vernonia patens</i>	14	22
	<i>Zygia longifolia</i>	20	40

Table 3.2: List of species collected at the INBiopark
Contains the number of images per species.

3.3. LEAF RECOGNITION AND CLASSIFICATION PROCESS



Figure 3.1: Leaf press
borrowed from INBio

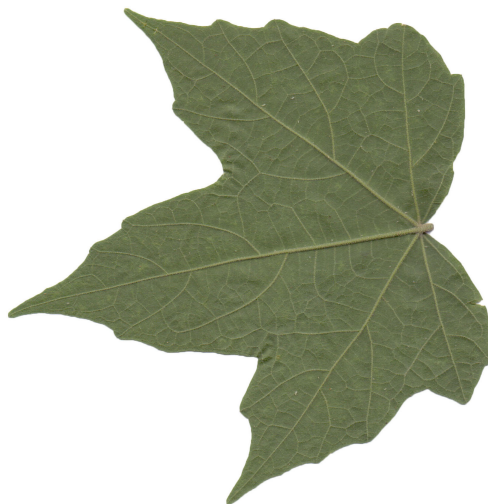


Figure 3.2: A *Robinsonella lindeniana* var. *divergens* sample
scanned from a leaf sample from INBiopark, using a HP ScanJet 300 scanner, then
cleaned using Photoshop CS6.



Figure 3.3: A *Bauhinia unguolata* sample taken using a Canon PowerShot SD780 IS camera at the Sabana Park.

against white, uniform backgrounds (normally a sheet of paper). Each image has a 3000x4000 pixel resolution, in JPG format. Figure 3.3 presents a noisy leaf image sample. The used camera specifications are the following:

- Name: Canon PowerShot SD780 IS
- 12.1 megapixels
- 1/2.3-inch CCD Sensor Type
- Highest resolution size: 4,000x3,000 pixels
- Year of release: 2009
- Accessible price to general public.



Figure 3.4: INBio Herbarium samples with noise, objects, transposed leaves, among other noisy elements

INBio Herbarium DataSet We attempted to make use of the herbarium at National Biodiversity Institute (INBio) to get leaf images. The plan was to scan several samples of several Costa Rican species from the herbarium, clean them with Photoshop, and use them for model training. This, given the high noise of the scans, was not possible. Current samples of the herbarium are appropriate for analysis performed by human beings, but when it comes to computer algorithms, they have too much noise: shadows, holes, plastic objects, interposed leaves, incomplete leaves, among others. We decided not make use of the herbarium samples because of these reasons. Figure 3.4 shows two samples provided by INBio where high noise is present.

3.3.2 Image Leaf Segmentation: Color Based

The first step to process the leaf image is to segment which pixels belong to a leaf and which do not. We used the same approach as LeafSnap, by applying color-based segmentation.

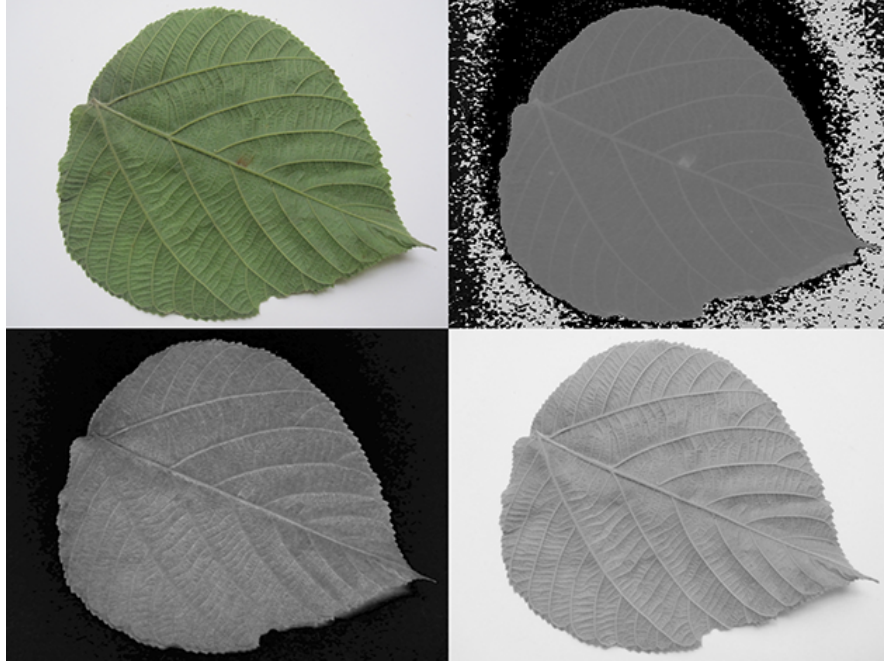


Figure 3.5: HSV decomposition of a leaf image

The top-left image shows the original sample. The top-right image shows the Hue channel of the image containing noise. The bottom-left image shows the Saturation component, and the bottom-right image shows the Value component.

HSV Color Domain

When segmenting with color it is imperative to use the right color domains that exclude undesired noise. Kumar et al. [24] states how, in the HSV domain during their tests, Hue had a tendency to contain greenish shadows from the original pictures. Saturation and Value however, had a tendency to be clean. So we also used those two color components for leaf segmentation. Figure 3.5 shows the noise present in the Hue channel, but also shows how Saturation and Value are cleaner. This was useful for posterior segmentation using Expectation-Maximization (EM). We used OpenCV to convert the original images into the HSV domain. Then, using NumPy, we extracted the Saturation and Value components, which were fed to the EM algorithm.



Figure 3.6: Segmented Samples
After applying EM to different Costa Rican species

Expectation-Maximization (EM)

Once images were converted to HSV and the desired channels were extracted, we applied EM to the color domain in order to cluster the pixels into one of 2 possible groups: leaf and non-leaf groups [24]. Figure 3.6 shows several samples of the final segmentation after applying EM. As shown, EM segments the image into the leaf and non-leaf pixel groups by assigning a 1 to the leaf pixels and a 0 to the non-leaf pixels. This method also works well on both simple and compound leaves. It is important to highlight that we didn't assign weights to each cluster manually as the work done by Kumar et al. [24], because we wanted to leave the process as automatic as possible because of the time constraints. In their work, they improve the segmentation of certain types of leaves, especially skinny ones, by manually assigning different weights to each cluster.

Training Algorithm 1 describes the process to train the EM algorithm. We used OpenCV’s implementation of EM. First we stacked all the pixels of the image matrix into a single vector. Then we trained the model using a diagonal matrix as a co-variance matrix, and we assigned two clusters to it, which internally were translated into two Gaussian Distributions, one for the leaf cluster and one for the non-leaf cluster. Once trained, we returned the EM object.

Algorithm 1 EM Training

```

stackedPixels  $\leftarrow \emptyset$ 
for all pixelRow in image do
  for all pixel in pixelRow do
    stackedPixels  $\leftarrow$  stackedPixels  $\cup$  pixel
  end for
end for
EM  $\leftarrow$  OpenCV.EM(nClusters = 2, covMatType = OpenCV.DIAGONAL)
EM.train(stackedPixels)
return EM

```

Pixel Prediction Algorithm 2 explains how the owning cluster of a single pixel of the image was predicted. Once the EM object was trained, the OpenCV’s implementation allowed to compute the probabilities of the pixel belonging to each cluster. However, for more efficiency, we created a dictionary containing each unique (*Saturation*, *Value*) pair as key, and the cluster as value. This way, we predicted only the unique keys once, which decreased computing time significantly. If the key was not found in the dictionary, we then proceeded to predict the probabilities for each cluster, added the key and cluster to the dictionary, and returned the associated cluster with the biggest probability.

3.3.3 Image Enhancements/Post-Processing

After segmentation of the leaf using EM, some extra work was needed to clean up several false positives areas. We followed the process of LeafSnap [24]. First of all, a

Algorithm 2 EM Pixel Prediction

```

key  $\leftarrow$  hash(pixel[S], pixel[V])
if hash in pixelDictionary then
    return pixelDictionary[key]
end if
probabilities  $\leftarrow$  EM.predict(pixel[S], pixel[V])
pixelDict[key] = probabilities[0] > probabilities[1]
return pixelDict[key]

```

heuristic was applied to delete undesired objects. Then, the stem was deleted since it added noise to the model of curvature (not that much to the texture model). Finally, each image was clipped to the internal leaf size provided by the segmentation and the image then was resized to a common leaf area.

Deleting Undesired Objects

Even when uniform background images were used, initial segmentation turned out not to be enough when the image contained undesired objects. These objects could be dust, shadows, corners of furniture, among other things. Kumar et al. [24] attempted to delete these noisy objects by using the same heuristic we implemented as shown in Algorithm 3. By using scikit-learn we calculated the connected components of the segmented image. We deleted the "small" components by area (in pixels). Small components were normally dust, small bugs or pieces of leaves, among other things. Once all small components were deleted, if the remaining was only one then we took that to be the leaf. If more than one component remained, then we calculated for each remaining component how many pixels had intersections with the image margin. We then deleted the component with the biggest number of intersections. The thinking behind this is to get rid of components that were not centered on the image, which tend to be non-leaf objects. Finally, the component with the biggest area from the remaining components was taken as the leaf.

Algorithm 3 Deleting Undesired Objects Heuristic

```

n, components  $\leftarrow$  connectedComponents(segmentedImage)
components  $\leftarrow$  deleteSmallComponents(components, kMinimumArea)
if size(components) == 1 then
    return components[0]
end if
inters  $\leftarrow$  empty
areas  $\leftarrow$  empty
for all component in components do
    inters  $\leftarrow$  inters  $\cup$  getImageMarginIntersections(component)
    areas  $\leftarrow$  areas  $\cup$  getComponentArea(component)
end for
noisyObject  $\leftarrow$  max(inters)
return max(areas - noisyObject)

```

Deleting stem

We followed the description of the stem deletion described in [24]. If the stem was left intact, it would add noise to the model of curvature, given all the possible sizes it may take. Algorithm 5 shows the procedure. First, all connected components were calculated from the segmented image, and also their quantity, using the scikit-learn library. Then, using OpenCV, a Top Hat transformation was applied to the segmented image in order to leave only possible stem regions, as shown in Figure 3.7. Once the Top Hat image was calculated, we looped over all the components, deleting every single one from the original segmentation and recalculating the new number of connected components. If the original number of connected components was the same as the new one after deletion, that meant the current component was a good stem candidate (heuristically, a stem does not affect how many original connected components there are). Once all stem candidates were calculated, the one with the biggest area and largest aspect ratio was chosen to be the stem, as described in Algorithm 4.

Algorithm 4 Calculate Aspect Ratio Combined with Area

```

width, height  $\leftarrow$  calculateRectangleAround(component)
area  $\leftarrow$  calculateArea(component)
return width/height * area

```

Algorithm 5 Deleting the Stem

```

candidates  $\leftarrow$  empty
candidatesRatios  $\leftarrow$  empty
n, components  $\leftarrow$  connectedComponents(segmentedImage)
possibleStemsImage  $\leftarrow$  topHatTransformation(segmentedImage)
for all component in components do
  tempSegmentation  $\leftarrow$  delete(component, segmentedImage)
  currentN  $\leftarrow$  connectedComponents(tempSegmentation)
  if currentN = n then
    candidates  $\leftarrow$  candidates  $\cup$  component
    candidatesRatios  $\leftarrow$  candidatesRatios  $\cup$  calculateAspectRatio(component)
  end if
end for
bestCandidate  $\leftarrow$  candidates[max(candidatesRatios).index]
segmentedImage  $\leftarrow$  delete(bestCandidate, segmentedImage)
  
```

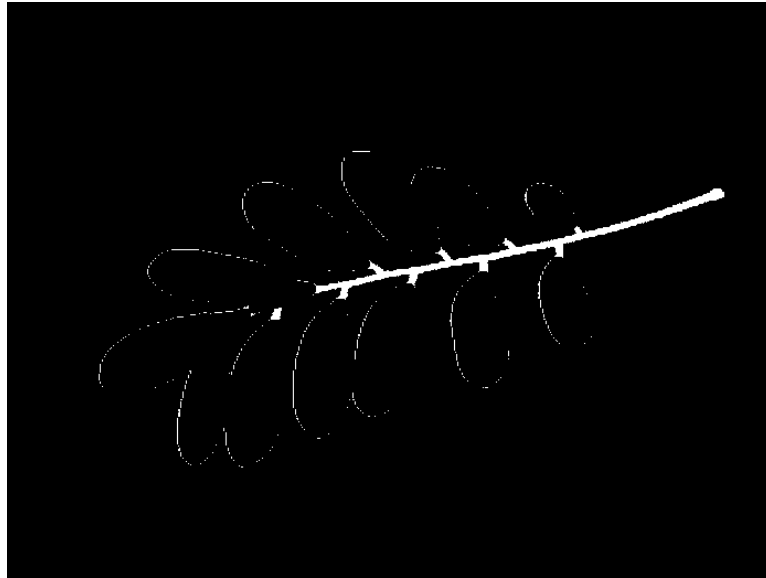


Figure 3.7: Top Hat Transformation applied to a leaf segmentation image.



Figure 3.8: Clipping of a *Coccoloba floribunda* sample
The left image is the original leaf image, and the right one is clipped to the leaf size.

Clipping

Before extracting features, we deprecated non-leaf regions by cutting the image around the leaf area. This was done in order to prepare the image for the subsequent resizing needed for the model of curvature. The clipping algorithm was trivial to implement once the contours were calculated using OpenCV. As shown in Algorithm 6, the minimum and maximum coordinates were calculated for all contour x and y components, followed by a cut of the leaf image matrix to those resulting minimum and maximum coordinates. The results of the Clipping phase can be seen in Figure 3.8.

Algorithm 6 Clipping Leaf Portion of the Image

```

 $xmin \leftarrow \min(contours.xs) - \epsilon$ 
 $ymin \leftarrow \min(contours.ys) - \epsilon$ 
 $xmax \leftarrow \max(contours.xs) + \epsilon$ 
 $ymax \leftarrow \max(contours.ys) + \epsilon$ 
 $clipped \leftarrow image[xmin : xmax, ymin : ymax]$ 

```

Resize Leaf Area

Once the leaf area had been clipped, a resize was applied in order to standardize the leaf areas inside all images. If not, the model of curvature was affected negatively since the amount of contour pixels varied significantly [24]. Our implementation of the resize was applied to the whole clipped image, however the calculations of height and width of the image depended on the leaf area. This means resulting images may had different

sizes, but the internal leaf areas were the same or almost the same. Algorithm 7 shows how a new width and height were obtained for resizing, by calculating the growth ratio between the current leaf area, the desired new leaf area, and the current height and width of the image. This relation turned to be a quadratic equation. Finally, OpenCV was used to resize the clipped image to a leaf size. We used 100,000 as that constant size for the normalized leaf area.

Algorithm 7 Common Leaf Area Resize

```

newLeafArea  $\leftarrow$  100000
imgArea  $\leftarrow$  height  $\times$  weight
newImgArea  $\leftarrow$  (imgArea  $\times$  newLeafArea)/leafArea
wGrowth  $\leftarrow$  weight/height + weight
hGrowth  $\leftarrow$  height/height + weight
a  $\leftarrow$  wGrowth  $\times$  hGrowth
x  $\leftarrow$   $\text{abs}(\sqrt{4 \times a \times \text{newImgArea}} / (2 \times a))$ 
newWidth = wGrowth  $\times$  x
newHeight = hGrowth  $\times$  x
return OpenCV.resize(image, newWidth, newHeight)

```

3.3.4 Leaf Feature Extraction

The feature extraction was designed and implemented considering three main concepts:

- Efficiency: we wanted to build a model capable of supporting future mobile apps, thus, it had to be fast and efficient.
- Rotation invariance: the leaf within the image may be rotated to any angle, thus, it was imperative to build a rotation invariant approach for texture.
- Multiscale: the datasets contain different sizes of leaves so we had to capture multiple scales.

Two different feature sets were calculated. The first one was an implementation of the Histogram of Curvature over Scale (HCoS) [24]. We attempted to download

this code implementation from the original URL provided in the paper but was not available, thus we coded the solution in Python based on the description of the paper. Then, an implementation of Local Binary Pattern Variance (LBPV) was used after the segmentation and leaf normalization process. Both models generated histograms which are suitable for distance metric calculations.

Model of Curvature Using HCoS

The model of curvature used by LeafSnap comprises several steps when it comes to feature extraction. Previously explained segmentation and post-processing resulted in a mask of leaf and non-leaf pixels. The non-leaf pixels have values of 0, and the leaf pixels have values of 1. First, the different contour pixels were found, then 25 different masks with discrete disk shapes were applied on top of each contour point, providing both an area of the intersection and an arc length. Then all calculations at each scale were assigned together into a histogram, generating 25 different histograms per image, one per scale. Finally, all the 25 resulting histograms were concatenated, conforming the HCoS.

Contours In a binary image (resulted of the previous segmentation), the OpenCV implementation of contour finding worked very well, based on the original algorithm of Satoshi Suzuki [42] for contour finding. The algorithm resulted in a vector of pairs (x, y) that represented the coordinates where a contour pixel was found. A contour pixel can be defined as a pixel which is surrounded by at least another pixel with the opposite color of it. Figure 3.9 shows the contour pixels detected in the original image, calculated from the segmented mask. Notice how shadows affect the contour algorithm, since they were not segmented perfectly.

Scales The original algorithm of Kumar et al. [24] makes use of 25 different scales, creating one disk per scale. Each disk is a discrete matrix representing a binary image of



Figure 3.9: Croton niveus contours extracted using OpenCV.

a disk, as round as possible in the discrete domain. We implemented a discrete version of the disks making use of matrices. The implementation is based on Manay et al. [31] work, available in Matlab ¹⁰.

The disks used were actually matrices of 1's and 0's. They were applied as masks over specific parts of the segmented leaf image (mostly contour points). The idea was to count how many pixels intersected the segmented image and each disk mask. We created two different versions of the disks: one completely filled up with 1's, as shown in Figure 3.10, used to count the area of intersection, and one where 1's are present only in the circumference of the disk, used to find out the arc's length of the intersection of the disk with the leaf, at a given contour point, as shown on Figure 3.11.

Once all disks were created for both area and arc length versions, we applied them to each pixel of the contour vector, as shown by Algorithm 8.

Figure 3.12 shows how one specific area disk was applied to the segmentation image, for an specific scale (radius=18 in this case), at a given contour pixel. The gray area shows the intersection of pixels with the leaf segmentation. This procedure was then

¹⁰https://www.ceremade.dauphine.fr/~peyrennumericaltourtoursshapes_4_shape_matching

[0.	0.	0.	0.	1.	0.	0.	0.	0.]
[0.	0.	1.	1.	1.	1.	1.	0.	0.]
[0.	1.	1.	1.	1.	1.	1.	1.	0.]
[0.	1.	1.	1.	1.	1.	1.	1.	0.]
[1.	1.	1.	1.	1.	1.	1.	1.	1.]
[0.	1.	1.	1.	1.	1.	1.	1.	0.]
[0.	1.	1.	1.	1.	1.	1.	1.	0.]
[0.	0.	1.	1.	1.	1.	1.	0.	0.]
[0.	0.	0.	0.	1.	0.	0.	0.	0.]

Figure 3.10: A discrete filled disk of radius=4 pixels
used to calculate area of the intersection with the leaf segmentation.

[0.	0.	0.	0.	0.	1.	0.	0.	0.	0.	0.]
[0.	0.	1.	1.	1.	0.	1.	1.	1.	0.	0.]
[0.	1.	0.	0.	0.	0.	0.	0.	0.	1.	0.]
[0.	1.	0.	0.	0.	0.	0.	0.	0.	1.	0.]
[0.	1.	0.	0.	0.	0.	0.	0.	0.	1.	0.]
[1.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.]
[0.	1.	0.	0.	0.	0.	0.	0.	0.	1.	0.]
[0.	1.	0.	0.	0.	0.	0.	0.	0.	1.	0.]
[0.	1.	0.	0.	0.	0.	0.	0.	0.	1.	0.]
[0.	0.	1.	1.	1.	0.	1.	1.	1.	0.	0.]
[0.	0.	0.	0.	0.	1.	0.	0.	0.	0.	0.]

Figure 3.11: A discrete unfilled disk of radius=5 pixels
used to calculate arc length of the intersection with the leaf segmentation.

Algorithm 8 Area and Arc Length Vector Calculation

```

arcs ← empty
areas ← empty
for all pixel of the contour vector do
  for all areaMask, arcMask = 1 to 25 do
    center areaMask, arcMask at current contour pixel
    area ← count(areaMask ∩ segmentation)
    areas ← areas ∪ area
    arc ← count(arcMask ∩ segmentation)
    arcs ← arcs ∪ arc
  end for
end for

```

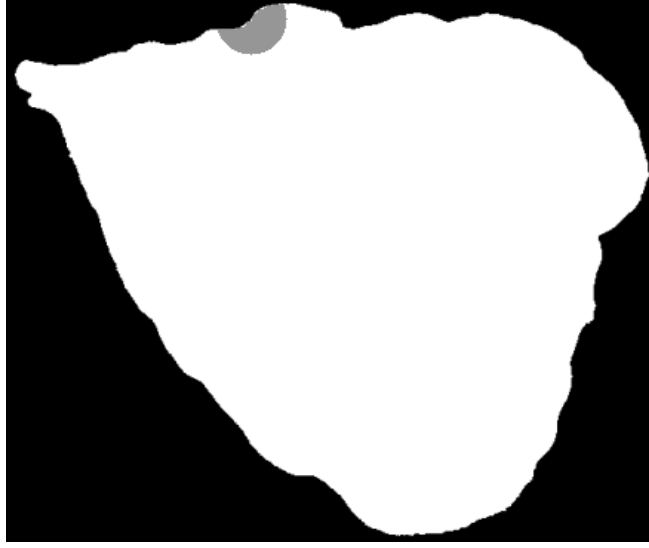


Figure 3.12: Area disk applied to a *Croton niveus* sample at an specific pixel of the contour, with radius=18

repeated over all the pixels from the contour vector in the same way.

Histograms Using NumPy at each scale, a histogram was created from all the values generated from all contour pixels, as described by Algorithm 8. We used histograms of 21 bins, as Kumar et al. [24] did. This means a total of 25 different histograms were created, each with 21 bins, per image. At each scale, each histogram was normalized to 1, by using Equation 3.1, where $N(x)$ is the normalization of histogram x and x_i is each bin of the histogram. Then, all histograms were concatenated together (both the 25 for area and 25 for arc length), generating what Kumar et al. [24] describes as the Histogram of Curvature over Scale (HCoS).

$$N(x) = \forall x_i \in x, \frac{x_i}{\sum_{i=1}^n x_i} \quad (3.1)$$

Our implementation of HCoS has a computing time that approximately ranges between 1 to 6 seconds, depending on the size of the leaf within the image. Future improvements may be implemented for further speed optimizations.

Variant	Radius	Pixels
R1P8	1	8
R2P16	2	16
R3P16	3	16
R1P8, R2P16 concatenated	1 and 2	8 and 16
R1P8, R3P16 concatenated	1 and 3	8 and 16
R3P24	3	24

Table 3.3: Variants of LBPV

Texture Using LBPV

We aimed to improve the model of curvature by adding texture analysis. We used a Local Binary Pattern Variance (LBPV) implementation that is invariant to rotation, multiscale and efficient [11]. The mahotas' implementation of LBPV is based on the algorithm of Ojala et al. [35], and makes use of NumPy libraries to represent the image and the resulting histograms. It works on gray images, so we used OpenCV to convert the RGB images to gray scale images. The LBPV approach aims to detect micro structures such as lines, spots, flat areas, and edges [35]. This is useful to detect patterns of the veins, areas between them, reflections, and even roughness. Figure 3.13 shows how 2 different LBPV implementations look. The upper image shows a *radius* = 2, 16 pixels implementation, and the one below shows a *radius* = 1, 8 pixel implementation. The different variants of the LBPV used are shown in Table 3.3. It is important to note that we did not use the variant which samples 24 pixels, since it generated histograms that were too big and memory demanding. We did, however, run some tests where we noticed the 24 pixels variation didn't add more accuracy, so we decided to ignore this method.

Histograms Just like the HCoS, LBPV generated histograms that can be used for similarity search. Several histograms were generated at different radius sizes and different circumference pixel sampling, in order to validate which combinations provided the best results. The mahotas' implementation returned a histogram of the feature counts,

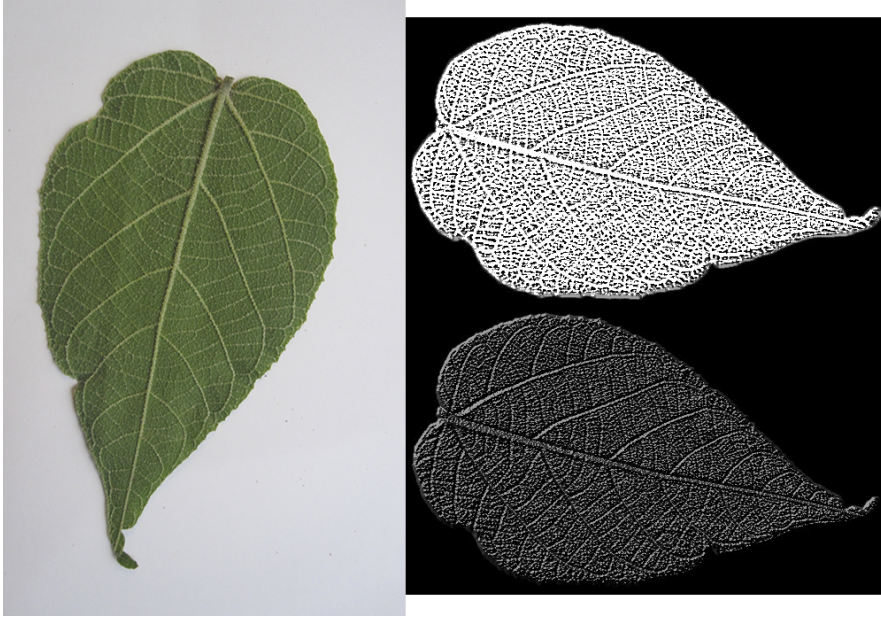


Figure 3.13: LBPV patterns of Croton draco sample
The upper image shows a radius=2, 16 pixels and the lower one is a radius=1, 8 pixels pattern.

where position i corresponds the count of pixels in the leaf texture that had code i . Also, given that the implementation is a LBPV, non-uniform codes are not used, thus, the bin number i is the i - *th* feature, not just the binary code i [11]. Figure 3.14 describes in very high level how the process of extracting the local patterns histograms works. First, the image is converted to a gray scale image. Then, for each pixel inside the segmented leaf area, we calculated the local pattern with different radius and circumference using the mahotas implementation. Finally, each pattern was assigned to a bucket inside the resulting histogram.

The implementation of LBPV applied to the internal pixels of the leaf is efficient. It takes approximately 1 up to 3 seconds to compute, depending of the variant of LBPV calculated, and also the size of the area of the lead within the image.

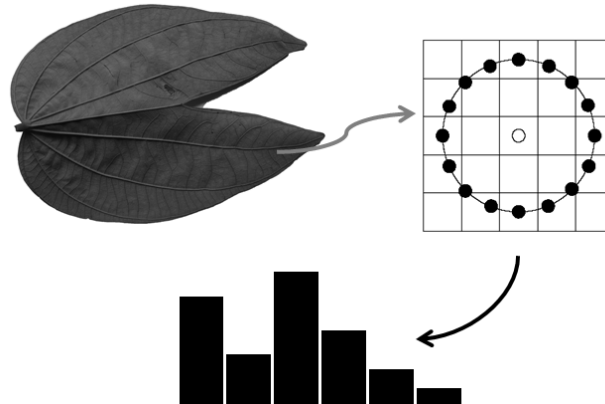


Figure 3.14: Process of extracting LBPV

Each pixel has a number assigned to it corresponding to a pattern, and the histogram was created using all those numbers from the segmented leaf pixels.

3.3.5 Species Classification based on Leaf Images

Once all histograms were ready and normalized, a machine learning algorithm had to be used to classify unseen images into species. We implemented the same classification scheme used by LeafSnap. The following subsection describes how k Nearest Neighbors (kNN) was implemented.

k Nearest Neighbors (kNN)

Scikit-learn’s k Nearest Neighbors (kNN) implementation was used for leaf species classification. This process was fed with previously generated histograms from both the model of curvature using HCoS and the texture model using LBPV. Additional code was created to take into consideration only the first matching k species, not the first k images, as shown by Algorithm 9. The difference resides in taking into account only the best matching image per species, until completing the first k species [24].

We used k between 1 and 10, differently than LeafSnap which uses only $k = 5$. The reason behind using several k values was to see how the different algorithms and versions of them behaved as the k is increased.

Algorithm 9 k Species Ranking

```

neighborImages, distances  $\leftarrow$  knnSearch(histogram,  $k + \epsilon$ )
resultSpecies  $\leftarrow$  empty
while each neighborImage and  $k > 0$  do
  if not neighborImage.species in resultSpecies then
    resultSpecies  $\leftarrow$  resultSpecies  $\cup$  neighborImage.species
     $k \leftarrow k - 1$ 
  end if
end while

```

Distance Metric: Histogram Intersection We tested the basic Euclidean distance to measure similarity between histograms, however the results were not encouraging. As stated in [24], we implemented the histogram intersection shown on Equation 3.2, where $I(x, y)$ is the histogram intersection between a histogram x and y of same size, n is the number of bins, and x_i and y_i are each bin of each histogram. This distance metric is also normalized to 1, hence the first sum.

$$I(x, y) = \sum_{i=1}^n x_i - \sum_{i=1}^n \min(x_i, y_i) \quad (3.2)$$

Accuracy Accuracy was the key metric we used to measure the results of this research. As in Equation 3.3, we defined accuracy a as the ratio between hits (right classifications) and all classification attempts.

$$a = \frac{\text{hits}}{\text{attempts}} \quad (3.3)$$

3.4 Experiment Design

We describe the design of the experiments executed during this research. Several algorithm variations were used in the experiments.

- Our implementation of LeafSnap’s model of curvature HCoS.
- Several scales of the texture model based on LBPV.

Model Name	Description	Type
HCoS	25 scales, 21 bins per scale	Curvature
R1P8	$radius = 1, 8pixels$	Texture
R2P16	$radius = 2, 16pixels$	Texture
R3P16	$radius = 3, 16pixels$	Texture
R1P8, R2P16	$radius = 1, 8pixels, radius = 2, 16pixels$	Texture
R1P8, R3P16	$radius = 1, 8pixels, radius = 3, 16pixels$	Texture
HCoS combined with R1P8, R3P16 concatenated	Assigned a factor to curvature and texture. Factors summed 1, increasing by 0.10	Curvature, Texture

Table 3.4: Models used in the experiments including curvature, variants of texture model, and combination of both.

- The combination of the best LBPV variant and the model of curvature. These combinations were done by assigning a factor to texture and to curvature.

Table 3.4 shows the different algorithms and variations used. A single curvature algorithm was used, however several texture algorithm variations, with different scales, were tested too. Combinations of the curvature algorithm and texture algorithms were included also, with different factors of importance assigned to each one.

Combining Curvature and Texture When combining two different models, we faced the issue of having different scales in the resulting ranking of each model. We had then to search for a way to normalize the ranking to a common interval of values. We normalized each ranking of each model to values between $[0, 1]$. Equation 3.4 shows how ranking distances were normalized:

$$N(x) = \forall x_i \in x, x_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \quad (3.4)$$

After normalizing the rankings (one per combined algorithm), we assigned a factor to each combined model in order to combine the predicted species into a single ranking. This factor sums 1 in total, however we varied the factor associated with each model to

3.4. EXPERIMENT DESIGN

see the behavior across different combinations. We used factors of (0.10, 0.90), (0.20, 0.80), (0.30, 0.70), (0.40, 0.60), (0.50, 0.50), (0.60, 0.40), (0.70, 0.30), (0.80, 0.20), (0.90, 0.10). For example, (0.50, 0.50) means we gave the same level of importance to each model on that combination. Algorithm 10 describes how the merge between 2 methods was achieved. A dictionary was created to store all the results associated with each factor combination. Then, for each factor, we first saved the species' distance resulted from each method that was not available in the other method's results. For the species results in common, we multiplied the distance value of the first algorithm results by the current factor, and the distance value of the second algorithm by its complement. Finally, the results were stored in a the dictionary with the key as the factor used.

Algorithm 10 Combining 2 Rankings

```
mergedResults  $\leftarrow \emptyset$ 
PERC_COMBINATIONS  $\leftarrow \{0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90\}$ 
for all factor in PERC_COMBINATIONS do
  results  $\leftarrow$  empty
  results  $\leftarrow$  results  $\cup$  (resultsAlgorithm1.species –
  resultsAlgorithm2.keys).distance
  results  $\leftarrow$  results  $\cup$  (resultsAlgorithm2.species –
  resultsAlgorithm1.keys).distance
  for all species, distance in resultsAlgorithm1 do
    if species in resultsAlgorithm2 then
      v1  $\leftarrow$  value * factor
      v2  $\leftarrow$  resultsAlgorithm2[species] * (1 – factor)
      results[species]  $\leftarrow$  v1 + v2
    end if
  end for
  mergedResults[factor]  $\leftarrow$  results
end for
```

Testing/Training Approaches Depending on the data available for each dataset, two testing and training approaches were used:

- One Versus All: We used this approach when there were no separated testing and training subsets. We took each one of the dataset images, separated them from

the others, used the rest for training, and used the separated image for accuracy testing (we knew beforehand the correct species of that image). We repeated this process for all the images in the dataset. Algorithm 11 shows how this approach works for an specific model. We, however, applied this approach to more than one model accordingly.

- **Separate Training and Testing Subsets:** when available, we used a training subset to train all the models, and then the separate testing subset to measure accuracy. Algorithm 12 describes how the algorithm was implemented.

Algorithm 11 One Versus All

```

hits  $\leftarrow$  0
attempts  $\leftarrow$  0
for all testingImage of the dataset do
  tempDataset  $\leftarrow$  dataset – testingImage
  for all trainingImage of the tempDataset do
    knn.addTrainingHistogram(trainingImage.histogram)
  end for
  knn.train()
  species  $\leftarrow$  knn.predict(testingImage.histogram)
  attempts  $\leftarrow$  attempt + 1
  if species == testingImage.expectedSpecies then
    hits  $\leftarrow$  hits + 1
  end if
end for

```

In regards of classification, we used $k \in [1, 10]$ in all the experiments, measuring how the accuracy behaved as we increased the error acceptance. The chosen classifier was kNN.

Three experiments were conducted. The Flavia Experiment aimed to compare the results of LeafSnap model of curvature and our texture approach versus other studies that used the Flavia dataset. The second experiment called The Accuracy of HCoS on Costa Rica And LeafSnap Incomplete Dataset Experiment, used the available portion of the LeafSnap dataset and the Costa Rican dataset to compare the accuracy of the

Algorithm 12 Training and Testing Subset Approach

```

for all trainingImage of the trainingSubset do
  knn.addTrainingHistogram(trainingImage.histogram)
end for
knn.train()
hits  $\leftarrow$  0
attempts  $\leftarrow$  0
for all testingImage of the testingSubset do
  species  $\leftarrow$  knn.predict(testingImage.histogram)
  attempts  $\leftarrow$  attempt + 1
  if species == testingImage.expectedSpecies then
    hits  $\leftarrow$  hits + 1
  end if
end for

```

implementation of curvature. Finally, the Experiment of Texture and Curvature Models in Costa Rica aimed to understand how the accuracy of the model of curvature increased as it was combined with the LBPV model.

The Flavia Experiment

The Flavia dataset is a very common benchmark dataset for leaf recognition systems [48]. We used the same dataset in order to understand where our implementation of the LeafSnap model of curvature can be placed among other studies, and also where our combination of texture and curvature is. This dataset was a good benchmark for clean images given its clean nature. The complete dataset was processed, image by image, and histogram files were saved in a folder inside each one of the 32 species folders of the dataset. This was done for each each model. Classification was applied using $k \in [1, 10]$. Since we didn't have a training and testing subset, we used the approach of One Vs All to calculate the accuracy of all the models over this dataset.

The Accuracy of HCoS on Costa Rica And LeafSnap Incomplete Dataset Experiment

This experiment was used to understand how our implementation of the model of curvature behaved in the data from USA and the data from Costa Rica. It was used to stressed out our first hypothesis, by comparing the accuracy of our HCoS implementation on data from USA and Costa Rica. As a highlight, the available LeafSnap dataset is not the complete dataset from the original LeafSnap research.

- LeafSnap Lab Subset vs Costa Rican Clean Subset: the idea was to compare the accuracy obtained in the cleanest subsets of both USA and Costa Rica. This portion of the experiment used One vs All approach for both Costa Rica and USA data subsets.
- LeafSnap Field Subset vs Costa Rican Noisy Subset: measured accuracy on noisy images containing shadows and other artifacts, for both Costa Rica and USA. This portion of the experiment used One vs All approach for both Costa Rica and USA data subsets.

The main goal of this experiment was to compare accuracy between USA and Costa Rica, using the same baseline implementations of the curvature and texture algorithms. We used a Proportion Test to determine if the results of running HCoS over the Costa Rican data was statistically smaller than the results from the LeafSnap Incomplete Dataset. The null hypothesis is that the accuracy obtained in both Costa Rica and USA is the same, and the alternative hypothesis is that the Costa Rican results are less-accurate than the obtained with the USA data.

Experiment of Texture and Curvature Models in Costa Rica

In order to test our second hypothesis, we ran all the implementations of model of curvature, texture, and combinations of both, to measure all their accuracies. Our

objective with this experiment was to measure if the accuracy increased significantly by adding texture to the model of curvature. We used $k \in [1, 10]$ to see the behavior across several k . We ran this experiment in 3 versions:

- A One vs All version with clean Costa Rica images.
- A One vs All version with noisy Costa Rican images, which include shadows, small objects, greenish regions, lighting conditions, blur, and others.
- A Separate Training and Testing Subsets version, using the clean images for training, and the noisy images for testing. This allowed to measure the accuracy of predicting the species of a noisy image, given clean images.

To measure the statistical significance of the accuracy increase we used a Proportion Hypothesis Test for 2 Samples between the HCoS results and the combination of it with texture. We used different factor combinations. Several tests were calculated based on the following approach:

- A series of tests were calculated for 3 subsets: The Costa Rican Clean Subset, The Costa Rican Noisy Subset, and the Complete Costa Rican Dataset.
- The tests were applied at each level of k , where $k \in [1, 10]$.
- The accuracy results of our implementation of HCoS were used as the first proportion p_1 , and then tested against 3 different combinations of HCoS and LBPV as the second proportions p_2 : 0.1 of HCoS combined with 0.9 of LBPV, 0.5 of HCoS combined with 0.5 of LBPV, and finally 0.9 of HCoS combined with 0.1 of LBPV.
- For all cases, Equation 3.5 shows the null hypothesis used, and Equation 3.6 describes the alternative hypothesis. We basically aimed to test if the accuracy

obtained by HCoS alone, was significantly smaller than the one obtained by the combination with texture.

$$H0 : p_1 = p_2 \tag{3.5}$$

$$H1 : p_1 < p_2 \tag{3.6}$$

CHAPTER 4

Results and Discussion

We should preserve every scrap of biodiversity as priceless while we learn
to use it and come to understand what it means to humanity.
E. O. Wilson, *The Diversity of Life*

4.1 Introduction

This chapter covers the results obtained during this research. Several experiments were executed, measuring accuracy of our implementations of the HCoS model of curvature and the LBPV model of texture. We report the accuracy results obtained from mainly 3 experiments: The Flavia Experiment, the Accuracy of HCoS on Costa Rica and LeafSnap Incomplete Dataset Experiment, and the Experiment of Texture and Curvature Models in Costa Rica.

This chapter is organized as follows: Section 4.2 describes the results of the first experiment with Flavia clean images. Section 4.3 presents a comparison of running our implementation of HCoS against Costa Rican data and the incomplete LeafSnap data,

k	HCoS	R1P8	R2P16	R3P16	R1P8,R2P16	R1P8,R3P16	R2P16,R3P16
1	0.371	0.796	0.879	0.866	0.882	0.892	0.878
2	0.508	0.899	0.940	0.930	0.943	0.948	0.939
3	0.590	0.932	0.962	0.951	0.962	0.964	0.961
4	0.649	0.952	0.970	0.962	0.970	0.975	0.972
5	0.697	0.964	0.974	0.974	0.976	0.980	0.977
6	0.732	0.971	0.977	0.977	0.978	0.982	0.982
7	0.764	0.976	0.980	0.982	0.980	0.982	0.983
8	0.783	0.977	0.981	0.983	0.981	0.983	0.984
9	0.797	0.978	0.982	0.983	0.982	0.984	0.985
10	0.813	0.978	0.983	0.983	0.983	0.984	0.985

Table 4.1: Individual models against Flavia

to test our first hypothesis. Finally, Section 4.4 focuses on the results obtained on the Costa Rican dataset, with both clean and noisy images, to measure the accuracy of the HCoS against the combination of it with texture, to stress the second hypothesis.

4.2 The Flavia Experiment

We ran the models against the well known Flavia dataset. A One Vs All approach was used for all the images. Table 4.1 shows the accuracy reported for each k value, per each individual model. In general, the LBPV variation R1P8,R3P16 with $radius = 1$, $8pixels$ concatenated with $radius = 3$, $16pixels$ had the maximum accuracy for $k = 1$ and $k = 10$, with 0.892 and 0.985 respectively. HCoS has the lowest accuracy results individually, within a range from 0.371 to 0.813 depending on the value of k . Chart 4.1 shows the best LBPV variant results and the HCoS results too across the different k values.

Table 4.2 shows the results obtained by combining HCoS and LBPV. The combination of 0.5 HCoS and 0.5 LBPV achieves an outstanding 0.99 of accuracy with $k = 10$. Chart 4.2 shows also the results from the HCoS combined with the LBPV method with different combination factors. For the combinations, we used R1P8,R3P16 since it reported the best results individually.

HCoS=a, R1P8.R3P16=b									
	a=0.1	a=0.2	a=0.3	a=0.4	a=0.5	a=0.6	a=0.7	a=0.8	a=0.9
k	b=0.9	b=0.8	b=0.7	b=0.6	b=0.5	b=0.4	b=0.3	b=0.2	b=0.1
1	0.831	0.819	0.799	0.782	0.745	0.697	0.641	0.575	0.512
2	0.919	0.917	0.907	0.897	0.884	0.849	0.808	0.748	0.670
3	0.950	0.950	0.949	0.941	0.933	0.914	0.883	0.832	0.759
4	0.964	0.966	0.967	0.962	0.958	0.945	0.927	0.889	0.830
5	0.975	0.973	0.974	0.973	0.970	0.965	0.949	0.922	0.877
6	0.979	0.978	0.978	0.977	0.977	0.974	0.968	0.947	0.906
7	0.981	0.982	0.982	0.982	0.981	0.980	0.976	0.961	0.930
8	0.984	0.985	0.986	0.986	0.985	0.982	0.980	0.972	0.947
9	0.988	0.988	0.989	0.988	0.989	0.988	0.984	0.979	0.964
10	0.988	0.988	0.990	0.991	0.991	0.991	0.987	0.983	0.970

Table 4.2: Combined models against Flavia

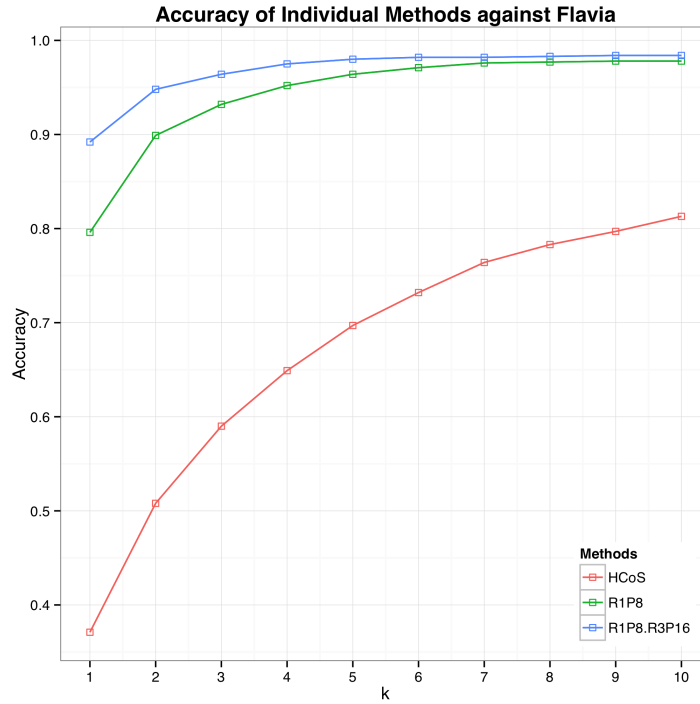


Figure 4.1: Individual models against Flavia

Study	Features	Classifier	Precision	Accuracy
Nguyen et al. [34]	SURF	SVM	0.959	N/A
Lee et al. [27]	FFT	Centroid	N/A	0.9719
Wu et al. [48]	Morphological Features	PNN	0.859	N/A
Kadir et al. [23]	Morphological, Color Features, FFT	PNN	N/A	0.9375
R.D and S [41]	Morphological Features	Euclidean Distance	N/A	0.919
Mouine et al. [33]	Triangle Side Lengths and Angle (TSLA)	kNN, k=1	0.69	N/A
Our Texture Model	LBPV R1P8, R3P16	kNN, k=1	N/A	0.892
Our Texture Model	LBPV R1P8, R3P16	kNN, k=5	N/A	0.98
Our Texture Model	LBPV R1P8, R3P16	kNN, k=10	N/A	0.985
Our HCoS Implementa- tion	HCoS	kNN, k=1	N/A	0.371
Our HCoS Implementa- tion	HCoS	kNN, k=5	N/A	0.697
Our HCoS Implementa- tion	HCoS	kNN, k=10	N/A	0.813
0.5 HCoS + 0.5 R1P8.R3P16	HCoS + LBPV R1P8, R3P16	kNN, k=10	N/A	0.991

Table 4.3: Comparison of obtained accuracies on Flavia

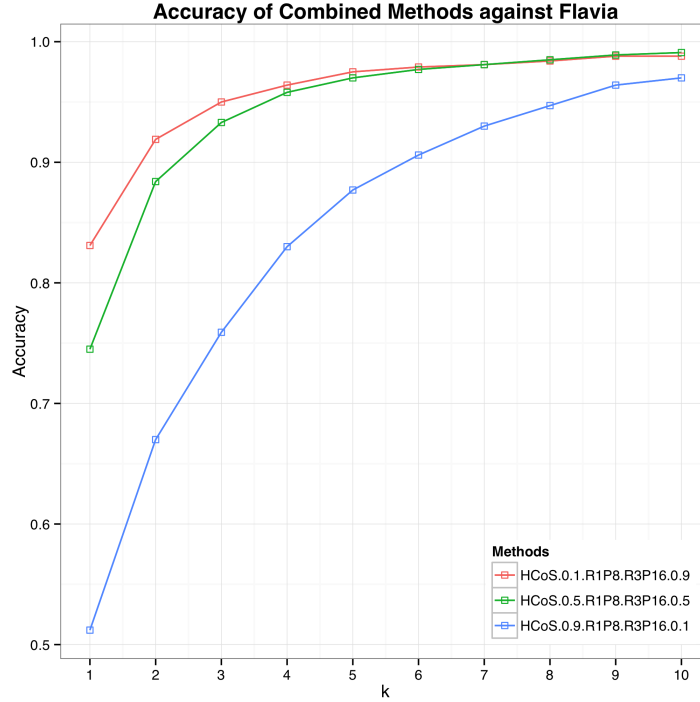


Figure 4.2: Combined models against Flavia

Discussion

The idea behind this experiment was to be able to compare our results with other studies that used the same Flavia dataset and to understand where our model is located within the realm of leaf recognition. There are other studies that have used LBPV however ours is different when preprocessing the images, since we use the LeafSnap preprocessing that segments, cleans and normalizes the images. Also, we wanted to know how our implementation of the HCoS behaved over the Flavia dataset and compare it with other studies too. Table 4.3 shows other studies with their corresponding precision and accuracy when applicable. According to these results, it can be observed how both our HCoS implementation and the LBPV implementation are viable for recognition on clean datasets like Flavia, achieving high levels of accuracy, especially if the allowed error is increased up to a 10 species ranking. The combination of HCoS and R1P8,R3P16 shows the best accuracy with $k = 10$, reaching 0.99 accuracy. As expected, the accuracy grows as the k grows for all models, since more error is allowed.

4.3 The Accuracy of HCoS on Costa Rica And Leaf-Snap Incomplete Dataset Experiment

Our first hypothesis claims that the model of curvature may not obtain such high accuracy in Costa Rica as the one obtained with the LeafSnap dataset in USA, given the high biodiversity from Costa Rica. We applied our HCoS implementation to the available portion of the LeafSnap dataset and to our Costa Rican dataset in order to compare the accuracy results. Our accuracy results are lower than the reported accuracy by the LeafSnap paper, which may be due to implementation differences. This experiment, however, uses our implementation as baseline for the comparison between the Costa Rican field data and the LeafSnap incomplete dataset.

LeafSnap Lab Subset Vs Costa Rican Clean Subset We ran our HCoS implementation against the incomplete LeafSnap lab subset and the Costa Rican clean subset, both representing clean datasets. As shown in Table 4.4, the best obtained accuracy was reported by the Costa Rican runs, with better accuracy across all k values. For $k = 5$, the results from Costa Rica were far better with 0.631, compared to the LeafSnap run with 0.136. The best accuracy was achieved by HCoS over the Costa Rican clean subset at $k = 10$ with 0.790. There is a clear improvement of the accuracy when classifying images from Costa Rica, with an increase ranging between 0.0268 to 0.567 depending on the value of k . This improvement is significant when using the Costa Rican clean subset, however we cannot reject the null hypothesis which states the accuracy obtained for Costa Rican clean images and USA clean images is the same.

LeafSnap Field Subset Vs Costa Rican Noisy Subset We did the same experiment with noisy images. For Costa Rica we calculated the accuracy of our HCoS implementation against the Costa Rican noisy subset, and then compared the results with the ones obtained from the LeafSnap field subset which also contains noise. Table

4.3. THE ACCURACY OF HCoS ON COSTA RICA AND LEAFSNAP INCOMPLETE DATASET EXPERIMENT

HCoS								
Confidence Level=0.95								
Costa Rica Clean Subset=1468								
LeafSnap USA Lab Subset=23147								
H0:Costa Rica=USA								
H1:Costa Rica<USA								
k	Costa Rica Clean	LeafSnap Lab	Costa Rica Clean Hits	LeafSnap Lab Hits	zValue	p-Value	Reject H0?	Accuracy Improvement
1	0.311	0.043	457	1006	42.0964	1.000	NO	0.268
2	0.446	0.071	655	1649	47.8291	1.000	NO	0.375
3	0.535	0.096	785	2229	49.6896	1.000	NO	0.438
4	0.587	0.116	861	2695	49.6807	1.000	NO	0.470
5	0.631	0.136	927	3155	49.4616	1.000	NO	0.495
6	0.674	0.157	989	3628	49.2045	1.000	NO	0.517
7	0.710	0.174	1043	4038	49.2089	1.000	NO	0.536
8	0.740	0.192	1086	4444	48.7673	1.000	NO	0.548
9	0.768	0.208	1127	4819	48.5655	1.000	NO	0.560
10	0.790	0.224	1160	5174	48.1620	1.000	NO	0.567

Table 4.4: Accuracy of our HCoS implementation over the LeafSnap Lab subset and Costa Rican Clean subset

4.5 shows the obtained results across all k values. For $k = 5$ the best results were obtained by the Costa Rican noisy subset with an accuracy of 0.364. Overall, the best reported accuracy was obtained at $k = 10$ with the Costa Rican noisy subset at 0.521 of accuracy. There was an improvement of the accuracy when the Costa Rican noisy subset was used. The improvement obtained in the accuracy ranged between 0.051 to 0.188, with the Costa Rican noisy subset as the better one.

Discussion

Our first hypothesis claims that the model of curvature would not achieve in Costa Rica an accuracy as good as in USA. This, however, does not seem to be the case according to our experiments. Regardless of the noise in the images, the Costa Rican counterpart got always better accuracy across all values of k . As shown on both Tables 4.5 and 4.4 the p-Values were 1 in all cases. This means the Null Hypothesis cannot be rejected, meaning, the accuracy gotten from the Costa Rican dataset is not smaller than the one obtained with the USA dataset, but bigger or at least equal statistically. The reasons behind this could be several: the complexity of the leaves from USA may be higher than the leaves used in the Costa Rican dataset creation, causing our

4.4. EXPERIMENT OF TEXTURE AND CURVATURE MODELS IN COSTA RICA

HCoS								
Confidence Level=0.95								
Costa Rica Noisy Subset=2345								
LeafSnap USA Field Subset=7719								
H0:Costa Rica=USA								
H1:Costa Rica<USA								
k	Costa Rica Noisy	LeafSnap Field	Costa Rica Noisy Hits	LeafSnap Noisy Hits	zValue	p-Value	Reject H0?	Accuracy Improvement
1	0.151	0.101	222	2327	4.9576	1.000	NO	0.051
2	0.225	0.147	331	3396	6.8880	1.000	NO	0.078
3	0.277	0.185	406	4291	7.7669	1.000	NO	0.091
4	0.325	0.214	476	4949	9.4197	1.000	NO	0.111
5	0.364	0.238	534	5516	10.8637	1.000	NO	0.125
6	0.399	0.261	586	6036	12.3710	1.000	NO	0.138
7	0.435	0.281	638	6507	14.3376	1.000	NO	0.153
8	0.470	0.301	690	6964	16.8460	1.000	NO	0.170
9	0.496	0.317	729	7333	19.0763	1.000	NO	0.180
10	0.521	0.333	764	7703	21.8109	1.000	NO	0.188

Table 4.5: Accuracy of our HCoS implementation over the LeafSnap Field subset and Costa Rican Noisy subset

implementation of HCoS to be less precise for them. There is also the variable of the data size. Given our time and resource constraints, our Costa Rican data set has less species and less images per species compared to the LeafSnap incomplete dataset, which may end up affecting the results.

4.4 Experiment of Texture and Curvature Models in Costa Rica

In order to stress our second hypothesis, we measured the accuracy of our HCoS implementation and the combination of it with texture. We used the texture variation of LBPV R1P8,R3P16, which reported the best individual texture accuracy in the Flavia Experiment. We measured the accuracy against the clean subset, the noisy subset, and the complete dataset, with several factor combinations of HCoS and LBPV R1P8,R3P16, calculated for several k values where $k \in [0, 1]$.

Clean Subset As shown in Table 4.6, the best results were obtained with $k = 10$ with the combination of 0.5 HCoS and 0.5 R1P8,R3P16 resulting in a 0.909 of accuracy,

4.4. EXPERIMENT OF TEXTURE AND CURVATURE MODELS IN COSTA RICA

k	Clean				Noisy				All			
	HCoS	HCoS=a, R1P8.R3P16=b			HCoS	HCoS=a, R1P8.R3P16=b			HCoS	HCoS=a, R1P8.R3P16=b		
		a=0.1 b=0.9	a=0.5 b=0.5	a=0.9 b=0.1		a=0.1 b=0.9	a=0.5 b=0.5	a=0.9 b=0.1		a=0.1 b=0.9	a=0.5 b=0.5	a=0.9 b=0.1
1	0.311	0.480	0.390	0.278	0.151	0.393	0.313	0.241	0.070	0.071	0.063	0.066
2	0.446	0.609	0.551	0.429	0.225	0.512	0.438	0.337	0.119	0.163	0.143	0.144
3	0.535	0.676	0.656	0.516	0.277	0.586	0.536	0.411	0.148	0.218	0.194	0.184
4	0.587	0.743	0.727	0.598	0.325	0.643	0.603	0.475	0.176	0.259	0.244	0.227
5	0.631	0.789	0.781	0.668	0.364	0.690	0.657	0.525	0.204	0.292	0.282	0.259
6	0.674	0.822	0.830	0.710	0.399	0.719	0.700	0.571	0.228	0.320	0.315	0.289
7	0.710	0.843	0.855	0.756	0.435	0.744	0.736	0.612	0.253	0.343	0.338	0.313
8	0.740	0.862	0.874	0.795	0.470	0.770	0.770	0.649	0.273	0.362	0.365	0.342
9	0.768	0.880	0.893	0.825	0.496	0.793	0.788	0.686	0.295	0.384	0.386	0.366
10	0.790	0.893	0.909	0.854	0.521	0.810	0.809	0.716	0.318	0.407	0.406	0.383

Table 4.6: Increase of accuracy when combining curvature and texture over the clean subset, the noisy subset, and the complete Costa Rican dataset

which contrasts with the individual HCoS which obtained 0.79 of accuracy. Chart 4.3 shows the accuracy over the clean subset of Costa Rica with the best 3 combinations of HCoS and R1P8,R3P16, and the individual HCoS.

Noisy Subset Table 4.6 shows the results obtained when running HCoS against the combination of it with R1P8,R3P16 for the noisy Costa Rican subset. In this case, the best combination was 0.1 HCoS and 0.9 R1P8,R3P16 achieving a 0.81 of accuracy with $k = 10$. In case of the individual HCoS, the best achieved accuracy for the noisy subset was 0.521. Chart 4.4 shows the reported accuracy over the noisy subset from Costa Rica and shows the different trends of 3 combinations and the HCoS implementation.

Complete Dataset Table 4.6 shows the reported accuracy when the Clean subset was used for training and the Noisy subset was used for testing. The best reported accuracy comes from the combination of 0.1 HCoS and 0.9 R1P8,R3P16 with 0.407, with $k = 10$, however most combination factors show similar accuracy. The HCoS once again shows the lowest accuracy with a maximum of 0.318 with $k = 10$. Chart 4.5 shows the obtained accuracy for the whole Costa Rican dataset with 3 different combinations and the implementation of HCoS.

4.4. EXPERIMENT OF TEXTURE AND CURVATURE MODELS IN COSTA RICA

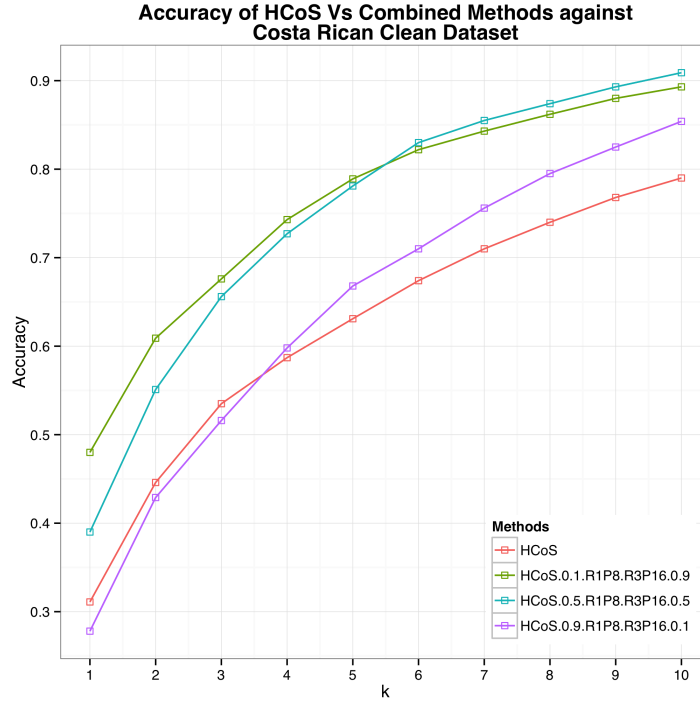


Figure 4.3: HCoS Vs Combined Methods against Costa Rican clean dataset

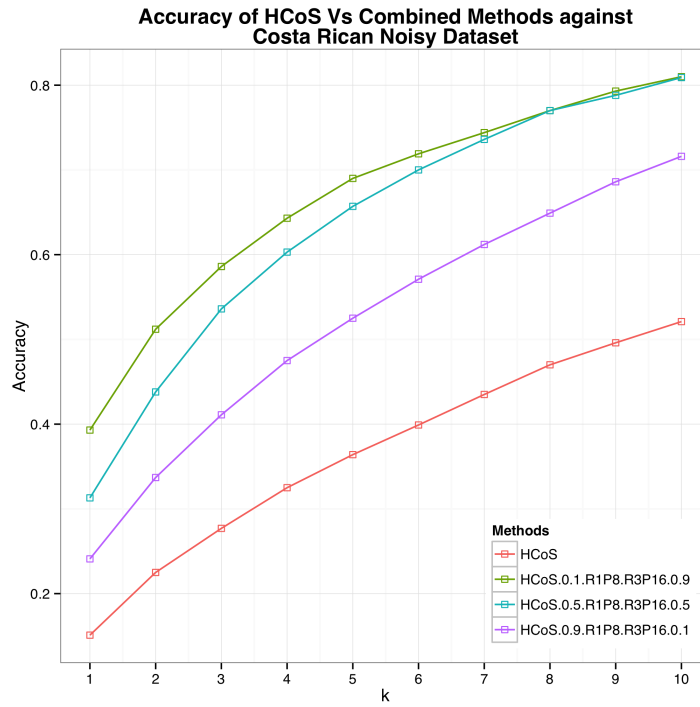


Figure 4.4: HCoS Vs Combined Methods against Costa Rican noisy dataset

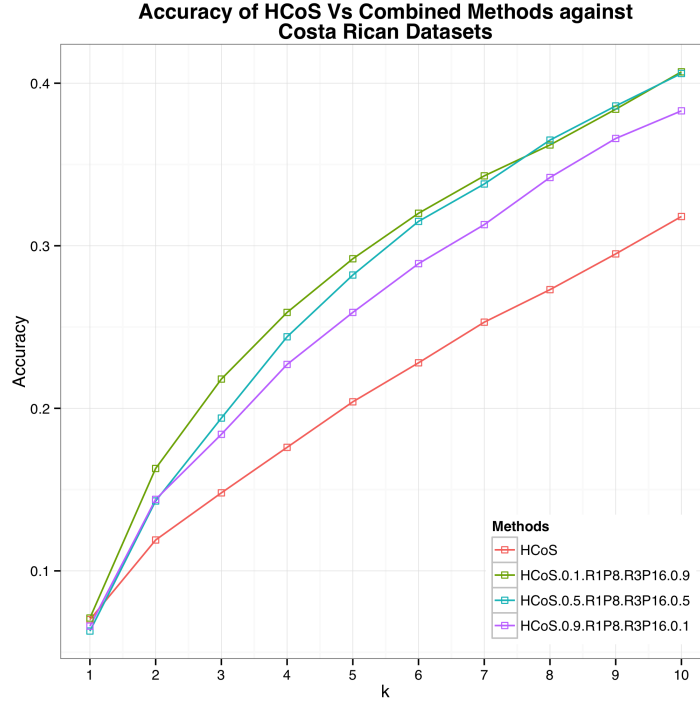


Figure 4.5: HCoS Vs Combined Methods against the complete Costa Rican dataset
Used the clean subset for training and the noisy subset for testing

Discussion

This experiment shows how the combination of HCoS and LBPV increases the accuracy of the model as the implementation of the model of curvature alone seems to obtain less accuracy. In general, regardless of using clean or noisy images, the results show how texture is relevant when it comes to recognition. The accuracy declines as the combination factor assigned to curvature reaches 1. Overall, the best combination seems to be 0.5 HCoS and 0.5 LBPV which is basically assigning the same factor of importance to each model implementation. It is also important to notice how the accuracy is sensitive to the quality of the dataset. The clean subset has a tendency to improve the recognition accuracy, in contrast with the noisy subset. This reflects the importance of good pre-processing and good segmentation. Shadows, dust, and other artifacts affect the final accuracy results.

4.4.1 Measuring Significance of the Accuracy Increase

As shown previously there is an increase in the accuracy when texture is added to our implementation of the model of curvature. This, however, may not be statistically significant. We proceeded then to apply a Statistical Proportion Test for 2 Samples, in order to check if the increase is actually significant and not just random. Our null hypothesis is that the accuracy of the implementation of HCoS equals the ones obtained by combining curvature and texture. In contrast, our alternative hypothesis is that the accuracy of the implementation of HCoS is less than the combinations.

Proportion Tests on the Clean Subset Table 4.7 shows the results obtained for all the Proportion Tests for the clean subset. Most combinations of HCoS and R1P8.R3P16 across all $k \in [1, 10]$ resulted in very low p-Values, rejecting the null hypothesis. However a few from the combination of 0.9 HCoS combined with 0.1 of R1P8.R3P16 did fail the test. This means as the factor increases for HCoS, it starts getting non-significant accuracy increases, which makes sense since it is almost equal to HCoS alone. For combinations where a bigger factor was assigned to texture, the improvement ranges between 0.078 and 0.168, which is significant as the p-Values show.

Proportion Tests on the Noisy Subset Table 4.8 shows the results obtained for all the Proportion Tests for the noisy subset. Similar to the clean subset, most combinations of HCoS and R1P8.R3P16 across all $k \in [1, 10]$ resulted in very low p-Values, rejecting the null hypothesis. A few from the combination of 0.9 HCoS combined with 0.1 of R1P8.R3P16 did fail the test, meaning assigning a factor of 0.1 to the HCoS does not make much difference as using it alone, at least statistically. The level of improvement is shown as high with a reported accuracy between 0.328 and 0.425 for a combination of 0.1 HCoS combined with 0.9 of R1P8.R3P16. The combination of 0.5 HCoS combined with 0.5 of R1P8.R3P16 shows the best improvement over HCoS alone with 0.431 when $k = 6$.

4.4. EXPERIMENT OF TEXTURE AND CURVATURE MODELS IN COSTA RICA

Costa Rica Clean Subset Confidence Level=0.95 Sample Size=1468 H0:HCoS=HCoS+R1P8.R3P16 H1:HCoS<HCoS+R1P8.R3P16							
k	HCoS	HCoS=0.1, R1P8.R3P16=0.9	Hits HCoS	Hits HCoS=0.1, R1P8.R3P16=0.9	p-Value	Reject H0?	Accuracy Improvement
1	0.311	0.480	457	704	5.65023E-21	YES	0.168
2	0.446	0.609	655	894	4.99997E-19	YES	0.163
3	0.535	0.676	785	993	2.00608E-15	YES	0.142
4	0.587	0.743	861	1091	1.21109E-19	YES	0.157
5	0.631	0.789	927	1158	2.81689E-21	YES	0.157
6	0.674	0.822	989	1207	9.64321E-21	YES	0.149
7	0.710	0.843	1043	1238	2.70704E-18	YES	0.133
8	0.740	0.862	1086	1266	4.32615E-17	YES	0.123
9	0.768	0.880	1127	1292	6.49779E-16	YES	0.112
10	0.790	0.893	1160	1311	1.14726E-14	YES	0.103
k	HCoS	HCoS=0.5, R1P8.R3P16=0.5	Hits HCoS	Hits HCoS=0.5, R1P8.R3P16=0.5	p-Value	Reject H0?	Accuracy Improvement
1	0.311	0.390	457	572	4.32788E-06	YES	0.078
2	0.446	0.551	655	809	6.56883E-09	YES	0.105
3	0.535	0.656	785	963	1.09341E-11	YES	0.121
4	0.587	0.727	861	1067	5.88439E-16	YES	0.140
5	0.631	0.781	927	1147	2.42945E-19	YES	0.150
6	0.674	0.830	989	1219	4.19306E-23	YES	0.157
7	0.710	0.855	1043	1255	1.18899E-21	YES	0.144
8	0.740	0.874	1086	1283	1.62723E-20	YES	0.134
9	0.768	0.893	1127	1311	7.26426E-20	YES	0.125
10	0.790	0.909	1160	1335	7.84393E-20	YES	0.119
k	HCoS	HCoS=0.9, R1P8.R3P16=0.1	Hits HCoS	Hits HCoS=0.1, R1P8.R3P16=0.9	p-Value	Reject H0?	Accuracy Improvement
1	0.311	0.278	457	408	0.976355356	NO	-0.033
2	0.446	0.429	655	630	0.823819993	NO	-0.017
3	0.535	0.516	785	758	0.840833982	NO	-0.018
4	0.587	0.598	861	878	0.26158887	NO	0.012
5	0.631	0.668	927	980	0.0201783	YES	0.036
6	0.674	0.710	989	1042	0.017077481	YES	0.036
7	0.710	0.756	1043	1110	0.002586312	YES	0.046
8	0.740	0.795	1086	1167	0.000201496	YES	0.055
9	0.768	0.825	1127	1211	5.92221E-05	YES	0.057
10	0.790	0.854	1160	1253	3.63353E-06	YES	0.063

Table 4.7: Proportion Test results over the Costa Rican Clean Subset
By adding texture with a bigger factor, the model of curvature improves significantly the accuracy. As the factor assigned to texture declines, the improvement becomes statistically insignificant.

4.4. EXPERIMENT OF TEXTURE AND CURVATURE MODELS IN COSTA RICA

Costa Rica Noisy Subset Confidence Level=0.95 Sample Size=2345 H0:HCoS=HCoS+R1P8.R3P16 H1:HCoS<HCoS+R1P8.R3P16							
k	HCoS	HCoS=0.1, R1P8.R3P16=0.9	Hits HCoS	Hits HCoS=0.1, R1P8.R3P16=0.9	p-Value	Reject H0?	Accuracy Improvement
1	0.151	0.480	730	1125	1.7283E-129	YES	0.3282
2	0.225	0.609	1046	1428	7.7632E-157	YES	0.3838
3	0.277	0.676	1254	1586	1.3354E-165	YES	0.3997
4	0.325	0.743	1375	1743	6.4313E-182	YES	0.4187
5	0.364	0.789	1481	1850	5.3369E-191	YES	0.4251
6	0.399	0.822	1580	1928	2.8814E-194	YES	0.4231
7	0.435	0.843	1666	1978	5.1936E-187	YES	0.4088
8	0.470	0.862	1735	2022	1.3596E-178	YES	0.3920
9	0.496	0.880	1800	2064	2.6133E-177	YES	0.3837
10	0.521	0.893	1853	2094	5.9603E-173	YES	0.3724
k	HCoS	HCoS=0.5, R1P8.R3P16=0.5	Hits HCoS	Hits HCoS=0.5, R1P8.R3P16=0.5	p-Value	Reject H0?	Accuracy Improvement
1	0.151	0.390	730	914	1.2405E-75	YES	0.238
2	0.225	0.551	1046	1292	2.2453E-116	YES	0.326
3	0.277	0.656	1254	1538	1.1237E-149	YES	0.379
4	0.325	0.727	1375	1704	7.6123E-168	YES	0.402
5	0.364	0.781	1481	1832	5.4143E-184	YES	0.418
6	0.399	0.830	1580	1947	1.5749E-202	YES	0.431
7	0.435	0.855	1666	2005	5.0885E-199	YES	0.420
8	0.470	0.874	1735	2049	8.0747E-191	YES	0.404
9	0.496	0.893	1800	2094	1.7097E-191	YES	0.397
10	0.521	0.909	1853	2133	2.0950E-191	YES	0.389
k	HCoS	HCoS=0.9, R1P8.R3P16=0.1	Hits HCoS	Hits HCoS=0.1, R1P8.R3P16=0.9	p-Value	Reject H0?	Accuracy Improvement
1	0.151	0.278	730	652	2.4494E-26	YES	0.127
2	0.225	0.429	1046	1006	1.9667E-50	YES	0.204
3	0.277	0.516	1254	1211	1.9949E-63	YES	0.240
4	0.325	0.598	1375	1403	4.4262E-79	YES	0.274
5	0.364	0.668	1481	1565	1.5164E-96	YES	0.304
6	0.399	0.710	1580	1665	6.3080E-102	YES	0.311
7	0.435	0.756	1666	1773	8.9291E-112	YES	0.322
8	0.470	0.795	1735	1864	6.4232E-118	YES	0.325
9	0.496	0.825	1800	1934	4.2650E-125	YES	0.329
10	0.521	0.854	1853	2002	9.9417E-134	YES	0.333

Table 4.8: Proportion Test results over the Costa Rican Noisy Subset
By adding texture with a bigger factor, the model of curvature improves significantly the accuracy. As the factor assigned to texture declines, the improvement becomes statistically insignificant.

Proportion Tests on the Complete Dataset Table 4.9 shows the results obtained for all the Proportion Tests over the complete dataset of Costa Rica. Almost every single test rejected the null hypothesis. The only ones that cannot reject that the proportions are the equal are the different combinations for $k = 1$. The accuracy improvement in these cases was not as big as in previous clean and noisy images, with values ranging between 0.001 to 0.093.

Discussion

We have proven statistically that the accuracy increases significantly when texture is added to the curvature model and that it is not related to chance. For most combinations where texture had a big assigned factor the null hypothesis was rejected, meaning the accuracy obtained by HCoS alone is not the same statistically as the one from the different combinations where a bigger factor is assigned to texture. The only ones that failed are the ones that have a very small factor assigned to the texture component. The results also reveal how the quality of the images does not affect the relation between the accuracy of curvature alone and how texture improves it, since the same p-Value behavior happened on both noisy and clean images. The best accuracy improvement was achieved with the noisy images with 0.431 and $k = 6$, for 0.5 HCoS combined with 0.5 of R1P8.R3P16. This means the best accuracy improvement is achieved by assigning half importance to each model.

4.4. EXPERIMENT OF TEXTURE AND CURVATURE MODELS IN COSTA RICA

Costa Rica All Dataset Confidence Level=0.95 Sample Size=2345 H0:HCoS=HCoS+R1P8.R3P16 H1:HCoS<HCoS+R1P8.R3P16							
k	HCoS	HCoS=0.1, R1P8.R3P16=0.9	Hits HCoS	Hits HCoS=0.1, R1P8.R3P16=0.9	p-Value	Reject H0?	Accuracy Improvement
1	0.070	0.071	164	167	4.3210E-01	NO	0.001
2	0.119	0.163	278	382	6.2947E-06	YES	0.044
3	0.148	0.218	346	512	1.8102E-10	YES	0.071
4	0.176	0.259	413	607	3.2827E-12	YES	0.083
5	0.204	0.292	479	685	1.6580E-12	YES	0.088
6	0.228	0.320	534	751	6.0361E-13	YES	0.093
7	0.253	0.343	593	804	8.0774E-12	YES	0.090
8	0.273	0.362	641	848	4.1983E-11	YES	0.088
9	0.295	0.384	692	901	5.8190E-11	YES	0.089
10	0.318	0.407	746	955	1.0927E-10	YES	0.089
k	HCoS	HCoS=0.5, R1P8.R3P16=0.5	Hits HCoS	Hits HCoS=0.5, R1P8.R3P16=0.5	p-Value	Reject H0?	Accuracy Improvement
1	0.070	0.063	164	148	8.2576E-01	NO	-0.007
2	0.119	0.143	278	336	6.0228E-03	YES	0.025
3	0.148	0.194	346	454	1.3785E-05	YES	0.046
4	0.176	0.244	413	573	4.9141E-09	YES	0.068
5	0.204	0.282	479	661	2.9011E-10	YES	0.078
6	0.228	0.315	534	738	1.0408E-11	YES	0.087
7	0.253	0.338	593	792	9.4610E-11	YES	0.085
8	0.273	0.365	641	856	8.2167E-12	YES	0.092
9	0.295	0.386	692	905	2.6311E-11	YES	0.091
10	0.318	0.406	746	953	1.6020E-10	YES	0.088
k	HCoS	HCoS=0.9, R1P8.R3P16=0.1	Hits HCoS	Hits HCoS=0.1, R1P8.R3P16=0.9	p-Value	Reject H0?	Accuracy Improvement
1	0.070	0.066	164	155	6.9915E-01	NO	-0.004
2	0.119	0.144	278	338	4.7461E-03	YES	0.026
3	0.148	0.184	346	432	3.6781E-04	YES	0.037
4	0.176	0.227	413	532	7.3870E-06	YES	0.051
5	0.204	0.259	479	608	4.0212E-06	YES	0.055
6	0.228	0.289	534	678	7.8066E-07	YES	0.061
7	0.253	0.313	593	733	2.8185E-06	YES	0.060
8	0.273	0.342	641	801	2.0626E-07	YES	0.068
9	0.295	0.366	692	859	1.0903E-07	YES	0.071
10	0.318	0.383	746	898	1.6458E-06	YES	0.065

Table 4.9: Proportion Test results over the Costa Rican Complete Dataset
By adding texture with a bigger factor, the model of curvature improves significantly the accuracy. As the factor assigned to texture declines, the improvement becomes statistically insignificant.

Conclusions and Future Work

When I hear of the destruction of a species, I feel just as if all the works of some great writer have perished.

Theodore Roosevelt

5.1 Conclusions

- Our implementation of HCoS obtained better results in the Costa Rican dataset compared to the LeafSnap Incomplete dataset. This is contrary to our first hypothesis, which stated that the current multiscale curvature model used by LeafSnap would not have the same high level of accuracy when recognizing Costa Rican species. The Costa Rican clean subset obtained better accuracy ranging from 0.268 to 0.567, compared to the LeafSnap lab subset. For the Costa Rican noisy subset, the improvement ranges from 0.051 to 0.188, compared to the LeafSnap field subset. This indicates that HCoS worked better in Costa Rica regardless of the subset image quality. Several variables may have caused this behavior, such

as the complexity of the shape from the USA leaves against the selected leaves from Costa Rica, or the data size, since the LeafSnap incomplete dataset has more species and more images per species than the Costa Rican dataset. Further research needs to be done to understand how these variables affect the results at the end.

- The addition of texture increases significantly the accuracy of our implementation of the model of curvature. When comparing HCoS versus the combination of 0.5 HCoS and 0.5 R1P8,R3P16, for the Costa Rican clean subset, the improvement ranges from 0.103 to 0.168, depending on the value of k . Similarly with the noisy subset, the improvement ranges from 0.238 to 0.431. These improvements were proved to be statistically significant in our experiments. We believe this improvement is due that current camera technologies allow to capture images with better quality for texture analysis. Mobile phones too have improved enough to capture high quality images that contain detailed texture patterns that can be exploited with algorithms such as LBPV.
- Leaf segmentation is a key component of a leaf recognition algorithm. The segmentation approach implemented in this research, same as LeafSnap, should be improved to be more precise, since it has direct implications on both the model of curvature and the texture model presented in this research. The model of curvature seems to be even more affected by false segmentations, making a texture based model more resilient to noise caused by wrong segmentation. It must be noted that, as noisy data is used, the accuracy improvement becomes wider when texture analysis is added to curvature, proving how texture by itself is more resilient to noise. For the combination of 0.5 HCoS and 0.5 R1P8,R3P16, the improvement over HCoS alone ranges between 0.238 and 0.431 for noisy images, while the same combination gets an accuracy improvement between 0.078 and 0.157 for cleaner images.

- Related to segmentation, shadows add too much noise and should be deleted during or post segmentation. When using the Costa Rican shadow-less subset, the accuracy obtained was higher than with the noisy subset. When comparing the results of HCoS between the Costa Rican clean and noisy subsets, the clean subset shows an improvement between 0.16 to 0.276 depending on the value of k . Additionally, the combination of 0.5 HCoS and 0.5 R1P8,R3P16 shows an improvement that ranges from 0.077 to 0.13 when clean images were used, also depending on the value of k . Other artifacts should also be taken care of too. However, shadows are the most prominent ones.
- Even the best herbaria do not have suitable clean physical samples useful for an image reference dataset creation. Physical samples from INBio's herbarium, which meet the highest standards worldwide, turned out too noisy, incomplete, had holes in the margin and inside, and some even had objects on top of them. The resulting images were suitable for leaf recognition executed by humans, however not for the presented algorithms. This caused the image capturing from those physical samples to become useless for this research.

5.2 Future Work

- Nowadays the professionals in charge of Herbaria capture images of physical samples that are not suitable for current Computer Vision research. We believe they should start generating pictures of not only the entire sample, but also of leaves, fruits, textures, and any other useful part of the plant for automatic recognition in separate images. This could impact positively the Biodiversity Informatics community. The images should also be captured before and after the pressing/drying process.
- A better segmentation algorithm must be created for both uniform-background

leaf images and also complex leaf images. The GrabCut Algorithm is worth exploring, paired with an iterative algorithm such as Genetic Algorithms or similar.

- In order to remove shadows, we explored illumination invariant images, however we did not include the results because of time constraints. This type of images have been used for generic shadow removal in other image processing sub-fields, with very good results. However, to our knowledge, no one has yet tried to add them for shadow removal during the pre-processing phases of the plant species recognition process.
- A next step in this research line is to create a mobile app in order to capture geographical data, that can be later used as an additional element to classify species. Particularly, INBio has Atta, an extensive georeferenced database that can be used as a gold set to eliminate false positives, since species tend to be present in certain elevations or areas of the country in particular.
- Additional effort is needed to capture more species and increase the dataset size of Costa Rican species images. By involving other professionals, researchers, and students from other areas such as forest engineering, we can increase significantly the amount of data available for leaf recognition projects. It is important to explore how the dataset sizes affect the obtained accuracy when more data is available.
- Research on crowd sourcing approaches for certain leaf recognition activities is worth exploring. Tasks such as shadow removal may be executed partially by a human, for example from a mobile device with their fingers, to build up better datasets. Also, by crowd sourcing the image capturing, bigger datasets may be created.
- Algorithms used to separate the leaf texture into its atomic layers should be researched. By layers we mean venation, edge layers, reflections, porosity, and

others, which would allow to calculate separately which ones add more value to the recognition process.

- Other morphological features of the leaf may be worth exploring, such as the aspect ratio, circularity, among others.

Bibliography

- [1] N. Aggarwal and R. K. Agrawal. First and second order statistics features for classification of magnetic resonance brain images. *Journal of Signal and Information Processing*, 3(2):146–153, 2012. doi: 10.4236/jsip.2012.32019.
- [2] Akhil Arora, Ankit Gupta, Nitesh Bagmar, Shashwat Mishra, and Arnab Bhattacharya. A plant identification system using shape and morphological features on segmented leaflets: Team iitk, clef 2012. In Pamela Forner, Jussi Karlgren, and Christa Womser-Hacker, editors, *CLEF (Online Working Notes/Labs/Workshop)*, 2012. ISBN 978-88-904810-3-1. URL <http://dblp.uni-trier.de/db/conf/clef/clef2012w.html#AroraGBMB12>.
- [3] C. H. Arun, W. R. Sam Emmanuel, and D. Christopher Durairaj. Texture feature extraction for identification of medicinal plants and comparison of different classifiers. *International Journal of Computer Applications*, 62(12), January 2013.
- [4] Thibaut Beghin, JamesS Cope, Paolo Remagnino, and Sarah Barman. Shape and texture based plant leaf classification. In Jacques Blanc-Talon, Don Bone, Wilfried Philips, Dan Popescu, and Paul Scheunders, editors, *Advanced Concepts for Intelligent Vision Systems*, volume 6475 of *Lecture Notes in Computer Science*, pages 345–353. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-17690-6. doi: 10.1007/978-3-642-17691-3_32. URL http://dx.doi.org/10.1007/978-3-642-17691-3_32.
- [5] Anant Bhardwaj, Manpreet Kaur, and Anupam Kumar. Recognition of plants by leaf image using moment invariant and texture analysis. *International Journal of Innovation and Applied Studies*, 3(1):237–248, 2013. ISSN 2028-9324. URL <http://www.ijias.issr-journals.org/abstract.php?article=IJIAS-13-087-01>.
- [6] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York,

- NY, USA, 1992. ACM. ISBN 0-89791-497-X. doi: 10.1145/130385.130401. URL <http://doi.acm.org/10.1145/130385.130401>.
- [7] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.
- [8] Samuel E. Buttrey and Ciril Karo. Using k-nearest-neighbor classification in the leaves of a tree. *Computational Statistics & Data Analysis*, 40(1):27–37, July 2002. URL <http://www.sciencedirect.com/science/article/B6V8V-44R2P3K-2/1/e332a5233807357bd3010778be0aeeb4>.
- [9] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.
- [10] James Clarke, Sarah Barman, Paolo Remagnino, Ken Bailey, Don Kirkup, Simon Mayo, and Paul Wilkin. Venation pattern analysis of leaf images. In *Proceedings of the Second International Conference on Advances in Visual Computing - Volume Part II*, ISVC'06, pages 427–436, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-48626-7, 978-3-540-48626-8. doi: 10.1007/11919629_44. URL http://dx.doi.org/10.1007/11919629_44.
- [11] Luis Pedro Coelho. Mahotas: Open source software for scriptable computer vision. *Journal of Open Research Software*, 1, 2013. doi: 10.5334/jors.ac. URL <http://dx.doi.org/10.5334/jors.ac>.
- [12] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.
- [13] Robert Fisher, Simon Perkins, Ashley Walker, and Erik Wolfart, 1997.
- [14] Hervé Goëau, Alexis Joly, Pierre Bonnet, Vera Bakic, Daniel Barthélémy, Nozha Boujemaa, Jean-François Molino, Philippe Birnbaum, Elise Mouysset, and Marie Picard. The imageclef plant identification task 2011. 2011.
- [15] Hervé Goëau, Alexis Joly, Pierre Bonnet, Vera Bakic, Daniel Barthélémy, Nozha Boujemaa, Jean-François Molino, and Itheri Yahiaoui. The imageclef plant identification task 2012. 2012.
- [16] Hervé Goëau, Alexis Joly, Pierre Bonnet, Vera Bakic, Daniel Barthélémy, Nozha Boujemaa, and Jean-François Molino. The imageclef plant identification task 2013. In *Proceedings of the 2Nd ACM International Workshop on Multimedia Analysis for Ecological Data*, MAED '13, pages 23–28, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2401-4. doi: 10.1145/2509896.2509902. URL <http://doi.acm.org/10.1145/2509896.2509902>.
- [17] Frank González. Protocolo para la digitalización de especímenes del herbario inb para el proyecto lapi.

- [18] Rafael C. Gonzalez, Richard E. Woods, and Steven L. Eddins. *Digital Image Processing Using MATLAB*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2003. ISBN 0130085197.
- [19] Y. Herdiyeni and I. Kusmana. Fusion of local binary patterns features for tropical medicinal plants identification. In *Advanced Computer Science and Information Systems (ICACSIS), 2013 International Conference on*, pages 353–357, Sept 2013. doi: 10.1109/ICACSIS.2013.6761601.
- [20] Y. Herdiyeni and M.M. Santoni. Combination of morphological, local binary pattern variance and color moments features for indonesian medicinal plants identification. In *Advanced Computer Science and Information Systems (ICACSIS), 2012 International Conference*, pages 255–259, Dec 2012.
- [21] Chaki . J. and Parekh. R. Designing an automated system for plant leaf recognition. *International Journal of Advances in Engineering and Technology (IJAET)*, pages 149–158, 2012.
- [22] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL <http://www.scipy.org/>. [Online; accessed 2014-10-06].
- [23] Abdul Kadir, Lukito Edi Nugroho, Adhi Susanto, and Paulus Insap Santosa. Leaf classification using shape, color, and texture features. *International Journal of Computer Trends and Technology*, 2011.
- [24] Neeraj Kumar, PeterN. Belhumeur, Arijit Biswas, DavidW. Jacobs, W.John Kress, IdaC. Lopez, and JoãoV.B. Soares. Leafsnap: A computer vision system for automatic plant species identification. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, Lecture Notes in Computer Science, pages 502–516. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-33708-6. doi: 10.1007/978-3-642-33709-3_36. URL http://dx.doi.org/10.1007/978-3-642-33709-3_36.
- [25] Mónica G. Larese, Rafael Namías, Roque M. Craviotto, Miriam R. Arango, Carina Gallo, and Pablo M. Granitto. Automatic classification of legumes using leaf vein image features. *Pattern Recogn.*, 47(1):158–168, January 2014. ISSN 0031-3203. doi: 10.1016/j.patcog.2013.06.012. URL <http://dx.doi.org/10.1016/j.patcog.2013.06.012>.
- [26] Kue-Bum Lee and Kwang-Seok Hong. An implementation of leaf recognition system using leaf vein and shape. *International Journal of Bio-Science and Bio-Technology*, pages 57–66, Apr 2013.
- [27] Kue-Bum Lee, Kwang-Woo Chung, and Kwang-Seok Hong. An implementation of leaf recognition system based on leaf contour and centroid for plant classification. In Youn-Hee Han, Doo-Soon Park, Weijia Jia, and Sang-Soo Yeo, editors, *Ubiquitous Information Technologies and Applications*, volume 214 of

- Lecture Notes in Electrical Engineering*, pages 109–116. Springer Netherlands, 2013. ISBN 978-94-007-5856-8. doi: 10.1007/978-94-007-5857-5_12. URL http://dx.doi.org/10.1007/978-94-007-5857-5_12.
- [28] Yan Li, Zheru Chi, and D.D. Feng. Leaf vein extraction using independent component analysis. In *Systems, Man and Cybernetics, 2006. SMC '06. IEEE International Conference on*, volume 5, pages 3890–3894, Oct 2006. doi: 10.1109/ICSMC.2006.384738.
- [29] Yan Li, Zheru Chi, and D.D. Feng. Leaf vein extraction using independent component analysis. In *Systems, Man and Cybernetics, 2006. SMC '06. IEEE International Conference on*, volume 5, pages 3890–3894, Oct 2006. doi: 10.1109/ICSMC.2006.384738.
- [30] Manal S .Khawasik M. Z. Rashad, B.S.el-Desouky. Plants images classification based on textural features using combined classifier. *International Journal of Computer Science and Information Technology*, 3(4), 2011.
- [31] S. Manay, D. Cremers, Byung-Woo Hong, A.J. Yezzi, and S. Soatto. Integral invariants for shape matching. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(10):1602–1618, Oct 2006. ISSN 0162-8828. doi: 10.1109/TPAMI.2006.208.
- [32] S. Mouine, I. Yahiaoui, and A. Verroust-Blondet. Plant species recognition using spatial correlation between the leaf margin and the leaf salient points. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*, pages 1466–1470, Sept 2013. doi: 10.1109/ICIP.2013.6738301.
- [33] Sofiène Mouine, Itheri Yahiaoui, and Anne Verroust-Blondet. A shape-based approach for leaf classification using multiscaletriangular representation. In Ramesh Jain, Balakrishnan Prabhakaran, Marcel Worring, John R. Smith, and Tat-Seng Chua, editors, *ICMR*, pages 127–134. ACM, 2013. ISBN 978-1-4503-2033-7. URL <http://dblp.uni-trier.de/db/conf/mir/icmr2013.html#MouineYV13>.
- [34] Q. Nguyen, T. Le, and N. Pham. Leaf based plant identification system for android using surf features in combination with bag of words model and supervised learning. In *International Conference on Advanced Technologies for Communications (ATC)*, October 2013.
- [35] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):971–987, Jul 2002. ISSN 0162-8828. doi: 10.1109/TPAMI.2002.1017623.
- [36] Travis E. Oliphant. *Guide to NumPy*. Provo, UT, March 2006. URL <http://www.tramy.us/>.

- [37] Raphael Ortiz. Freak: Fast retina keypoint. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 510–517, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-1-4673-1226-4. URL <http://dl.acm.org/citation.cfm?id=2354409.2354903>.
- [38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [39] Matti Pietikäinen, Abdenour Hadid, Guoying Zhao, and Timo Ahonen. Local binary patterns for still images. In *Computer Vision Using Local Binary Patterns*, volume 40 of *Computational Imaging and Vision*, pages 13–47. Springer London, 2011. ISBN 978-0-85729-747-1. doi: 10.1007/978-0-85729-748-8_2. URL http://dx.doi.org/10.1007/978-0-85729-748-8_2.
- [40] C. A Price, O Symonova, Y Mileyko, T. Hilley, and J. S Weitz. Leaf extraction and analysis framework graphical user interface: Segmenting and analyzing the structure of leaf veins and areoles. *Plant Physiology*, 2011.
- [41] Lagerwall R.D and Viriri S. Plant classification using leaf recognition. In *Proceedings of the 22nd Annual Symposium of the Pattern Recognition Association of South Africa*, page 91–95, November 2011.
- [42] KeiichiA Be Satoshi Suzuki. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32 – 46, 1985. ISSN 0734-189X.
- [43] JoãoV.B. Soares and DavidW. Jacobs. Efficient segmentation of leaves in semi-controlled conditions. *Machine Vision and Applications*, 24(8):1623–1643, 2013. ISSN 0932-8092. doi: 10.1007/s00138-013-0530-0. URL <http://dx.doi.org/10.1007/s00138-013-0530-0>.
- [44] Oskar J O Söderkvist. Computer vision classification of leaves from swedish trees. Master’s thesis, Linköping University, SE-581 83 Linköping, Sweden, September 2001. LiTH-ISY-EX-3132.
- [45] Donald F. Specht. Probabilistic neural networks. *Neural Netw.*, 3(1):109–118, January 1990. ISSN 0893-6080. doi: 10.1016/0893-6080(90)90049-Q. URL [http://dx.doi.org/10.1016/0893-6080\(90\)90049-Q](http://dx.doi.org/10.1016/0893-6080(90)90049-Q).
- [46] Jayshree Ghorpade Vishakha Metre. An overview of the research on texture based plant leaf classification. *CoRR*, abs/1306.4345, 2013.
- [47] D Wijesingha and FMMT Marikar. Automatic detection system for the identification of plants using herbarium specimen images. *Tropical Agricultural Research*, 23 (1), 2012. URL <http://www.sljol.info/index.php/TAR/article/view/4630>.

- [48] S.G. Wu, F.S. Bao, E.Y. Xu, Yu-Xuan Wang, Yi-Fan Chang, and Qiao-Liang Xiang. A leaf recognition algorithm for plant classification using probabilistic neural network. In *Signal Processing and Information Technology, 2007 IEEE International Symposium on*, pages 11–16, Dec 2007. doi: 10.1109/ISSPIT.2007.4458016.
- [49] Nelson Zamora, May 2014. Private Communication, National Biodiversity Institute, Costa Rica.

Acronyms

BMP Bitmap File

EM Expectation-Maximization

FREAK Fast Retina Keypoint Descriptor

FFT Fast Fourier Transform

GIST A low dimensional representation of the scene, which does not require any form of segmentation

GTSDM Gray Tone Spatial Dependency Matrix

HMT Hit Miss Transformation

HCoS Histogram of Curvature over Scale

HSV Hue Saturation Value

ICA Independent Component Analysis

INBio National Biodiversity Institute

kNN k Nearest Neighbors

LBP Local Binary Pattern

LBPV Local Binary Pattern Variance

LSH Locality Sensitive Hashing

LVQ Learning Vector Quantization

NN Neural Network

ORB Oriented FAST and Rotated BRIEF

PCA Principal Component Analysis

PFT Polar Fourier Transform
PNN Probabilistic Neural Network
RBF Radial Base Function
RGB Red Green Blue
SDK Software Development Kit
SIFT Scale-Invariant Feature Transform
SURF Speeded Up Robust Features
SVM Support Vector Machines
TAR Triangle Area Representation
TOA Triangle Oriented Angles
TSL Triangle Side Lengths
TSLA Triangle Side Lengths and Angle
UHMT Unconstrained Hit-or-Miss Transform
XML Extensible Markup Language