Aparajita Sahay

Windows Phone based Plant Recognition Application Using Leaf Analysis

Aparajita Sahay

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Masters in Computer Science and Software Engineering

University of Washington

2016

Reading Committee:

Min Chen, Chair

Mark Kochanski

Michael Stiber

Program Authorized to Offer Degree:

Masters in Computer Science and Software Engineering

University of Washington

Abstract

**Windows Phone based Plant Recognition Application Using Leaf Analysis**

Aparajita Sahay

Chair of supervisory committee:

Professor Min Chen  
Computer Science and Software Engineering

The capstone project involves developing a windows phone based plant recognition system by analyzing leaf images. Given a single leaf specimen on a white background, the application classifies the unknown leaf against a known data set and returns the top three results to the user.

The leaf recognition process consists of three stages. First, the query image is pre-processed to remove unwanted noise. The second stage includes identifying interesting features and computing Scale Invariant Feature Descriptors (SIFT). Lastly, a weighted K-Nearest Neighbor search compares features of the query image with a database data set and returns the top three plant species as the result.

TABLE OF CONTENTS

Table of Contents

[Chapter 1 INTRODUCTION 8](#_Toc445895079)

[1.1 Motivation and Goal 8](#_Toc445895080)

[1.2 Overview 8](#_Toc445895081)

[1.3 Literature Review 9](#_Toc445895082)

[1.3.1 Preprocessing of leaf image 9](#_Toc445895083)

[1.3.2 Feature extraction: 10](#_Toc445895084)

[1.3.3 Image matching and result retrieval: 11](#_Toc445895085)

[Chapter 2: RESEARCH WORK AND APPROACH 11](#_Toc445895086)

[2.1 Data Set 11](#_Toc445895089)

[2.2 System Architecture Decisions 11](#_Toc445895090)

[2.3 Languages and Tools Selection 12](#_Toc445895091)

[2.4 Feature Selection 13](#_Toc445895092)

[2.5 Algorithm Decisions 13](#_Toc445895093)

[2.5.1 Preprocessing 16](#_Toc445895094)

[2.5.2 Feature Extraction 17](#_Toc445895095)

[2.5.3 Image Search 18](#_Toc445895096)

[Chapter 3: SYSTEM ARCHITECTURE 19](#_Toc445895097)

[3.1 Architecture 19](#_Toc445895099)

[3.2 Leaf Recognition process 22](#_Toc445895100)

[3.2.1 Leaf Image Preprocessing 22](#_Toc445895101)

[3.2.2 Local Feature Extraction 23](#_Toc445895102)

[3.2.3 Image Search 25](#_Toc445895103)

[3.3 Implementation 27](#_Toc445895104)

[3.4 Tools and frameworks 27](#_Toc445895105)

[3.5 Software Development life cycle 28](#_Toc445895106)

[Chapter 4: EVALUATION 28](#_Toc445895107)

[4.1 Experimental Result 28](#_Toc445895109)

[Chapter 5: CONCLUSION 36](#_Toc445895110)

[5.1 Summary 36](#_Toc445895112)

[5.2 Future Work 37](#_Toc445895113)

[Lessons learnt 37](#_Toc445895114)

[5.3 Acknowledgement 38](#_Toc445895115)

[5.4 References 38](#_Toc445895116)

List of Figures

[Figure 1: Layered architecture describing involved in leaf identification. (a) Layer accepting color image from user (b) Layer pre-processing image to remove noise (c) Extraction of distinguishing features. (d) Comparison of features using a matching algorithm. 14](#_Toc445051438)

[Figure 2: Overview of the possible configuration of the system. Dashed line represents implemented and tested element which are however not part of the best and final system configuration. 15](#_Toc445051439)

[Figure 3: SIFT points in gray scale image 16](#_Toc445051440)

[Figure 4: SIFT points in Binary image 16](#_Toc445051441)

[Figure 5: High-level architecture of leaf recognition system 18](#_Toc445051442)

[Figure 6: User interface of the windows phone when a user selects a query image and top 3 result of most relevant result are shown back to the user. 20](#_Toc445051443)

[Figure 7: Leaf Recognition Process 21](#_Toc445051444)

[Figure 8: Preprocessing stage of leaf recognition process. 22](#_Toc445051445)

[Figure 9: Sift key points from leaf images. 23](#_Toc445051446)

[Figure 10: Class Diagram 25](#_Toc445051447)

[Figure 11: Iterative and Incremental model for project development. 26](#_Toc445051448)

[Figure 12: Fifteen species of leaf used for testing leaf recognition algorithm. 27](#_Toc445051449)

[Figure 13: Performance analysis of inter species, intra species and mixed species classification 31](#_Toc445051450)

[Figure 14: Performance analysis of KNN and weighted KNN on 5 data set 32](#_Toc445051451)

[Figure 15: Performance analysis with different Ratio test value on 5 data set. 32](#_Toc445051452)

[Figure 16: Performance analysis with gray scale and binary scale image 33](#_Toc445051453)

List of Tables

[Table 1: Testing the performance of the system on 15 leaf species shown in Figure 12. 26](#_Toc445051325)

[Table 2: Performance analysis of the system on leaf with same family- Inter species classification 27](#_Toc445051326)

[Table 3:Performance analysis of the system on leaf with different family- Intra species classification 29](#_Toc445051327)

[Table 4: Performance of system with weighted and normal KNN algorithm 30](#_Toc445051328)

# Chapter 1 INTRODUCTION

### Motivation and Goal

For inquisitive gardeners who wish to be knowledgeable about their plants in their garden, gathering information about unknown plants manually is time consuming. To gather information, they either have to go to their local nursery, read books or research on the internet.

Plant recognition has always been a time consuming task, as very subtle features may diﬀerentiate members within the same families. Plants have many features that aid in identifying the species, such as dimension, branch shape and flowers. However, one of the most deﬁning features is their leaves. Leaves contain important information about the taxonomic identity of a plant [1].

I propose to automate plant recognition with the following goals.

1. **Mobility**: Develop a mobile application that gardeners can conveniently carry with them to take photographs of leaf and the application will returns the top 3 most closely matched leaf from database in real time.
2. **Robustness**: variation in photograph of a leaf such as rotation and scaling should not affect the outcome of the result.
3. **Accuracy**: To build an accurate application over the wide range of leaf species.
4. **Portability**: Expose key capabilities of this application as a Web Service to facilitate faster future development across platforms and form factors.

### Overview

To improve usability, mobile application will be developed on windows phone platform. The system will focus on features of a leaf to identify its plant species. Real world photographs of leaves can have many variations such as background noise, scale, rotation, color and brightness. To remove noise from background, and to correctly extract leaf boundaries from photograph, leaf has to be placed on a white background. Scale-invariant feature transform (SIFT) algorithm will be used to detect and describe key points in the image, which will ensure scale and rotation invariance. Weighted K-Nearest Neighbor search algorithm is used to compare query image with training dataset to select most relevant images. Result contains names of top 3 plant species, together with standard images of their leaves for conﬁrmation.

To ensure portability of this solution across platforms, all leaf analysis is being done on server side. The server component is an ASP.NET Web API service, which can serve requests to any client capable of communicating over HTTP, including other popular mobile platforms such as iOS and Android.

### Literature Review

Plant identification demands extensive knowledge and understanding of complex terminologies. Even professional botanists require significant time in the field to analyze the plant. As plant leaves holds, useful characteristics for species identification [10], leaf recognition through images can be considered an important research topic for plant recognition.

There are various methods proposed by many researchers to implement a plant recognition system using leaf. In [1], [2], [3], leaf detection procedure is divided into the following sub categories:

* + 1. Preprocessing of leaf image

In the preprocessing step, it is important to represent image in such a way that it is invariant to translation, rotation, scale, and viewing angle [1].

Preprocessing of the image is essential in extracting image features because pictures taken in an uncontrolled environment can affect the accuracy of the result [1]. Therefore, preprocessing methodologies can be used to remove noisy information from the image.

In [3], image background was removed at the outset by applying Mahalanobis distance.

The Mahalanobis distance is a very useful way of determining the “similarity” of a set of values from an “unknown” sample to a set of values measured from a collection of “known” samples. One of the main reasons the Mahalanobis distance method is used is because it is very sensitive to inter-variable changes in the training data. In addition, since the Mahalanobis distance is measured in terms of standard deviations from the mean of the training samples, the reported matching values give a statistical measure of how well the spectrum of the unknown sample matches (or does not match) the original training spectra. This method belongs to the class of supervised classification algorithms [12]. Mahalanobis distance is efficient in segmenting vegetation pixels from natural backgrounds against light changes in a certain range [12].

In [1], preprocessing of an image is needed because the color of the plant is usually green and the color of leaf for the same plant can vary depending upon the water, nutrient and season. Since the color feature is not reliable, a method was proposed in [1] to convert the raw image into gray level image and then further use it for feature extraction.

Another approach [2] is to segment images by estimating foreground and background color distributions and using them to independently classify each pixel. This initial segmentation is solved using Expectation-Maximization method [2]. Expectation-Maximization , computes the parameters and weights of the two Gaussians for the given image and using them for pixel-wise segmentation. However, in practice, some challenges were there in segmenting leaves like pine needles and there were false positives detection related to shadows in the image [2].

* + 1. Feature extraction:

There are different properties of a leaf (such as shape, color, texture, size etc.) that can be used to compute the similarity between query image and database images. The goal of a good representation of each image is to extract the minimal information capable of differentiating one object from another [5]. Some of the features that used to identify leaf of a plant are as follows:

First, in papers [2] and [3], shape feature of a leaf is extracted from the binary image. Edges and curvature are some of the properties of a shape.

In [3], canny edge detection algorithm is used to determine the edges of a leaf. In [2], curvature feature of a leaf is extracted from a binary image, and then histogram of curvature is computed for further image comparison to identify a leaf.

Second, in paper [5] author have mentioned that the factors such as weather condition, diseases in a plant that can affect the picture quality of leaf and which in turn will affect the accuracy of the leaf identification. Author has handled this limitation by merging many features of a leaf into a feature vector. In [5] feature sets are categorized into two categories. First category is the global feature and the second is local descriptor. The first category consists of the length and width of the leaf area or the other properties that define shape such as aspect ratio, convex area ratio,rectangularity and circularity. The second category consist of details that involves texture based feature of a leaf. Local features or texture based properties of a leaf are contrast, correlation, energy, homogeneity, maximum probability and entropy. In [2], author have combined the above mentioned categories i.e. global feature and local descriptor into one category for representing the feature set of an image.

* + 1. Image matching and result retrieval:

In [2], the feature set of the query image is compared with the labeled database of leaf images using nearest neighbor approach with histogram intersection as the distance metric.

In [3],supervised training approach is used to classify the image**.** Active Shape Model (ASM) [3] conducted classification of leaf boundaries using multi-layered perceptron (MLP) prior to detection of the whole leaf. MLP was utilized for differentiating leaf boundaries from veins [3], then a leaf shape model is created using ASM method. Leaf shape model is further used to match query image in a leaf model then verification of matched leaf model is performed and finally the leaf detection result is returned to the user.

In [7], support vector machine (SVM) classifier is used to implement an automated leaf recognition system for plant leaf identification and classification.

# Chapter 2: RESEARCH WORK AND APPROACH

2. 1. Data Set

Data set used for training the system is obtained from Leafsnap. Leafsnap is an electronic field guide developed by researchers from Columbia University and University of Maryland [2].

To promote further research in computer leaf recognition, researchers from Leafsnap have released the Leafsnap dataset. Data set consists of images taken by mobile devices (IPhone mostly) in outdoor environment. These images contain a varying amount of blur, noise, illumination patterns and noise. Dataset consist of plant species from the Northeastern United States.

* 1. System Architecture Decisions

This section will discuss various system architecture and design decisions made.

Initially, a client (Windows phone 8) only system was planned to be implemented. However, the decision was changed from designing a client only model to client server model. The reasons are as following:

1. Training data set is quite large. Mobile clients will have variable memory and disk space across user base and will impact the performance of the application.
2. Similarly, mobile client’s will have varying CPU capabilities and some of them may not be powerful enough to run computationally expensive image processing algorithms and will result in long waiting time for the user.
3. In a client-server architecture, any enhancement to algorithms or data set on sever side will be immediately available to all clients.
   1. Languages and Tools Selection
4. Developing environment: Visual studio 2015 IDE to build the application.

Advantage of using Visual studio 2015 IDE : -

1. Prior experience on working with visual studio IDE and C# language was biggest advantage for choosing visual studio.
2. Throughout building of the leaf recognition application many new computer vision technologies were tried and tested. Therefore, to focus more on new technologies, I choose a familiar developing environment so that more time can be given to computer vision and image analysis part of the project.
3. In addition, openCV library widely used in computer vision also provides .NET wrapper EmguCv to access various algorithms for image analysis.
4. The client side is built on Windows phone 8.1 emulator. Application is designed to offer real time leaf recognition to the users, therefore, phone application was chosen over other platforms such as web and desktop.
5. The server side is build using Asp.Net web API. I also considered building the application using Asp.Net MVC. Advantage of using Asp.Net Web API are as follows : -
6. In Asp.Net web API, service only replies as data and no view is generated. However, in Asp.Net MVC data and view both are generated. In the leaf recognition application, windows phone creates its own view therefore only data is required to the client side to display the output. Hence, Asp.Net web API is used over Asp.Net MVC.
7. In addition, the Web API helps the creation of RESTful services over the .Net Framework but the MVC does not support.
   1. Feature Selection

To identify and compare images, feature selection and extraction is important. The features of each image gives a unique identity to the image and helps to classify image or identify the most closely matched relevant image in the database.

Leaves contain important information about the taxonomic identity of a plant. Moreover, leaves are present on the plants for several months in a year, whereas ﬂowers and fruits may remain only several weeks. This is why most plant identiﬁcation tool based on Content-Based Image Retrieval techniques work on leaf image databases [1], [2], [3].

Color, texture and shape are a few characteristics of a leaf. The color of a leaf may vary with the seasons and climatic conditions. As most plants have similar colors, this feature is not discriminant enough for the species recognition [1].

Most object (leaf) recognition system tend to have either global image features , which describe an image as a whole, or local features, which represent image patches. Global features have the ability to generalize an entire object with a single vector. Global features include contour representations, shape descriptors, and texture features. Global features use in standard classification techniques is straightforward. Local features, on the other hand, are computed at multiple points in the image, are consequently more robust to occlusion, and clutter [19]. Most local features represent texture in an image patch. For example, SIFT features use histograms of gradient orientations [19].

In the leaf recognition system, local feature descriptor of the image is used for object recognition because they are more robust to occlusion and clutter.

* 1. Algorithm Decisions

This section discusses various algorithms and methods investigated for leaf identification. A typical pipeline for leaf recognition involves steps as highlighted in Figure 1 below.



Figure 1: Layered architecture describing involved in leaf identification. (a) Layer accepting color image from user (b) Layer pre-processing image to remove noise (c) Extraction of distinguishing features. (d) Comparison of features using a matching algorithm.

In Figure 2 below, overview of various possible configurations that were investigated are discussed based on layered architecture.

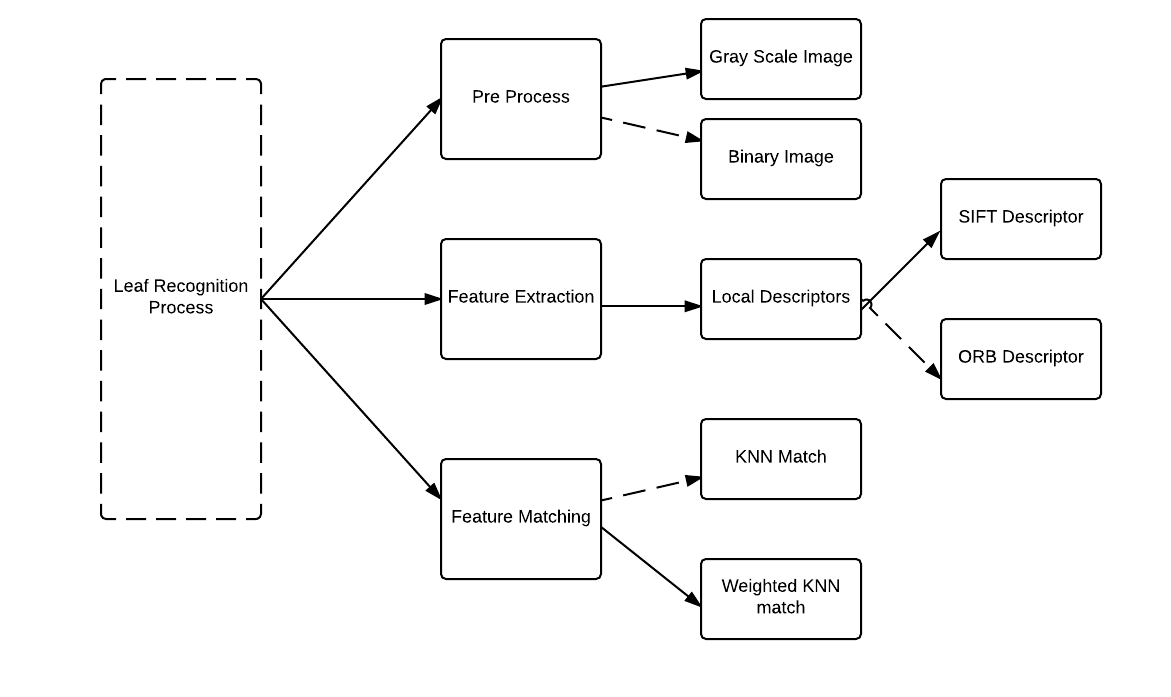


Figure 2: Overview of the possible configuration of the system. Dashed line represents implemented and tested element which are however not part of the best and final system configuration.

* + 1. Preprocessing

The preprocessing layer receives a color image as input.

At the preprocessing stage, two approaches were implemented and tested and they are as follows:

1. Gray scale Image is used for further image processing.
2. Convert the image to gray scale.
3. Applied morphological operation - dilate and erode method on gray scale image to remove noise.
4. Erosion method erodes away the boundary of the foreground object. It is useful for removing small noises from the image.
5. After erosion method, noise is removed however; the boundary of the original image is reduced in the process. After erosion, we can recover the original boundary by applying dilation operation.
6. Binary image is used for further image processing.
7. Convert the image to gray scale.
8. To convert grayscale image into a binary image, a threshold value is selected in which all values less than the threshold become a zero, or black, in the binary image and all values greater than or equal to the threshold value become a one, or white, in the binary image.

* Used Otsu thresholding on each gray scale image to get threshold value. Otsu segmentation (OpenCV) [Ots79], only used with default parameters.

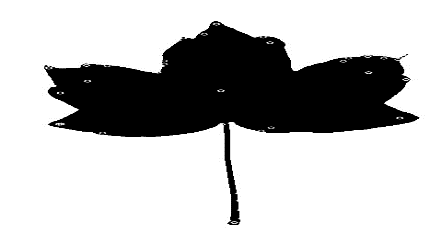


Figure 3: SIFT points in gray scale image



Figure 4: SIFT points in Binary image

Gray scale image is preferred over binary image in our final configuration.

To obtain binary image after applying thresholding will result in loss of information in the image and can reduce the number of interesting points in the image than in turn will affect the performance of the algorithm. Through the experiments it has been analyzed that number of SIFT points detected using binary image is less than grayscale image as shown in Figure 3 and Figure 4. Less number of SIFT point at feature extraction phase reduces the overall performance of the system. In addition, grayscale image is insensitive to illumination unlike binary image.

However, memory requirement for a system working with grayscale image is more than memory requirement with binary image.

Therefore, grayscale image is used over binary image for further processing of data.

* + 1. Feature Extraction

In this system, local features of image are used for feature extraction. Descriptors of an image is computed that describes the local features of an image. System uses two method to find local descriptors of an image and they are as follows,

1. SIFT descriptors are used to identify descriptors from the shape of an image.
   1. The Scale-Invariant Feature Transform (SIFT) [13] algorithm provides a robust method for extracting distinctive features from images that are invariant to rotation, scale and distortion.
   2. OpenCV framework is used to detect and describe SIFT key points in an image.
2. Oriented FAST and Rotated BRIEF (ORB) descriptors are used to identify descriptors from the shape of an image.
   1. ORB algorithm is a fast local feature detector.
   2. It is based on the FAST key point detector and the visual descriptor BRIEF (Binary Robust Independent Elementary Features).
   3. It is a good alternative for SIFT in computation cost and matching performance.
   4. However, BRIEF descriptors used in ORB performs poorly with rotation. It has observed that once an image is oriented, ORB gives different result.

The goal is to identify and describe key points that are invariant to scale and rotation. Therefore, SIFT descriptors are used for feature extraction.

* + 1. Image Search

Supervised instance based learning algorithms are analyzed for image classification.

The classification layer receives a feature set as input and has two functions depending on the type of data used - training or testing. In, instance based learning, all instances of the training data set are used for comparison with test images. After comparison, test image is assigned one or more a class labels.

OpenCV is used for executing KNN match algorithm.

Advantages and limitations of using KNN Match:

1. It is simple to learn and understand.
2. It has lower computation cost during training in comparison to eager learning.
3. KNN has greater storage requirement and higher computational cost during testing.
4. KNN can be made faster by parallel implementation.
5. Selecting the right value of k is very important in KNN match. Selecting a k value can greatly affect the performance of the system as very large value of K increases the computation cost.

Below we discuss the KNN match algorithm in detail

1. KNN match.

For each key point in a query image, 2 nearest sift point from training instance is computed. Out of 2 of the nearest Sift points, the shortest distance is taken for each point. Then average distance is calculated for each instance.

Finally, all the instances are sorted and top three instances having smallest distance are returned.

Equal importance/vote is given to all the neighbors.

1. Distance Weighted KNN search

For each key point in a query image, the 2 nearest SIFT points from training instance is computed.

Distance Average is computed for all the distances in one instance.

Each instance is also assigned a weight, which is inversely proportional to square of its average distance.

In Weighted KNN, closest neighbors are weighted more heavily than the farther ones, using the distance weighted function.

Distance weighted KNN is chosen over simple KNN so that all instances have a vote in deciding the outcome. However, KNN weighted search is sensitive to imbalanced data set. Therefore, in our system, we have used different leaf categories but each category has balanced number of training data set.

# Chapter 3: SYSTEM ARCHITECTURE

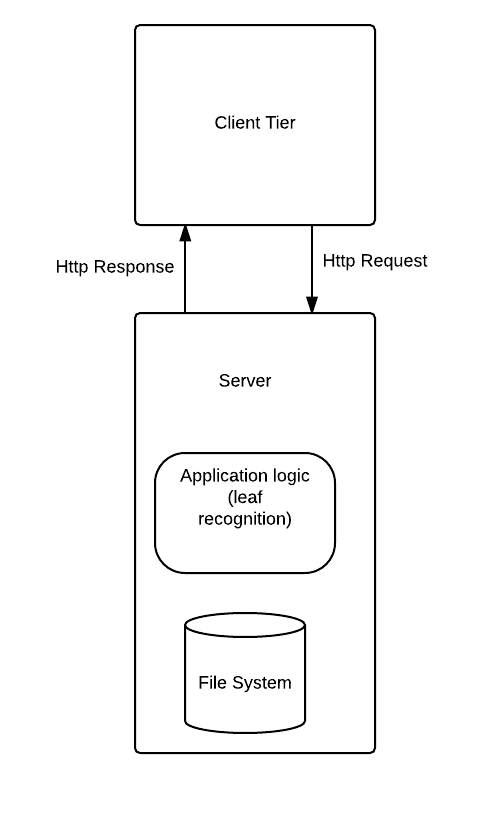


Figure 5: High-level architecture of leaf recognition system

1. 1. Architecture
2. Client

* The client only acts as the periphery, which acquires and sends leaf image to the server and eventually receives the top three matched results from the server.
* The client is currently Windows phone 8 device. Furthermore, a console-based application is implemented to quickly analyze the correctness and performance of the leaf identification algorithm.
* On the client side, XAML is used for user interface design.
* Methods are exposed on client side that selects leaf image from the windows phone storage, converts the image into byte array and sends the HTTP request of query image to the server.
* Once the client receives the response of top 3 leaf species and its image in bytes from server, client deserializes the image from byte to bitmap image and displays leaf name and its image in the UI.

1. Server

* Backend server consists of leaf recognition engine.
* Server accepts input images from various front-end clients. Currently, the client is Windows phone 8 device.
* The connection between the GUI on the mobile device and the identiﬁcation system on the server is done by a RESTful web service which oﬀers an attractive scalable computing architecture.
* The data exchange between the clients i.e. a windows phone device and web service are ensured via HTTP protocol.
* Web service consists of a library called Leaf Core that consist of leaf identification/business logic and file system that stores leaf data set. The leaf data set consists of 450 images from 20 different species.

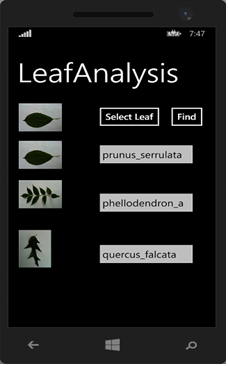
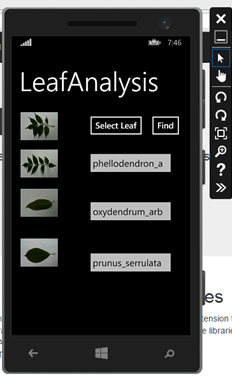




Figure 6: User interface of the windows phone when a user selects a query image and top 3 result of most relevant result are shown back to the user.

* 1. Leaf Recognition process

This section provides details of the image recognition process. The server accesses the business logic layer of the server side system to retrieve the matched result. The leaf recognition library/business logic computes the feature of query image and classifies the image by comparing it with the database image.

The system in this work follows the following procedure:

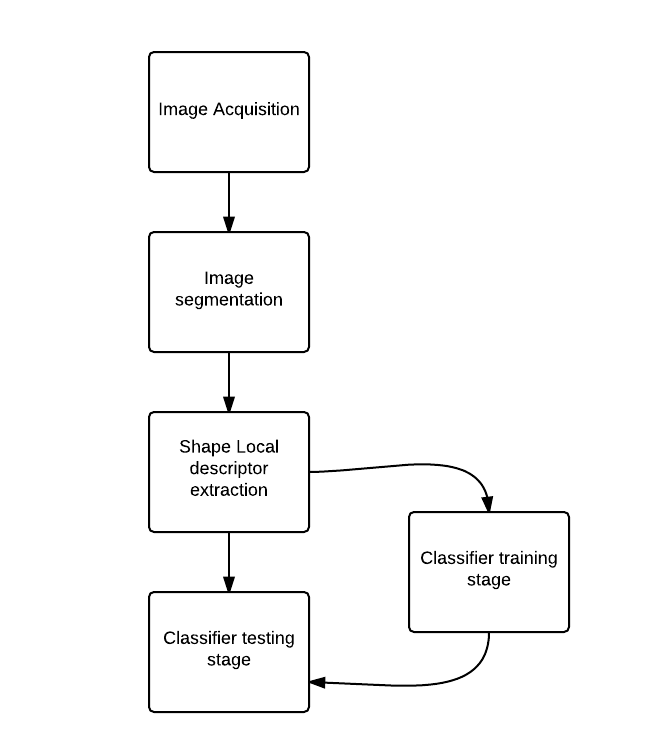


Figure 7: Leaf Recognition Process

* + 1. Leaf Image Preprocessing

Preprocessing in leaf recognition system is necessary to remove unwanted noise from the image and retain as many relevant features from the image.

The Preprocessing stage in leaf recognition system follows the following steps:

1. Image is resized to width and height of 400 pixels. Digital images vary drastically in size; this could lead to unexpected results for different sized images using the same algorithm. In order to unify the approach, an image is resized.
2. Colored image is converted into gray scale. The reason for not using color image for feature extraction is color of all the plants are usually green. Moreover, the shades and the variety of changes of water, nutrient, atmosphere and season can cause change of the color, so the color feature has low reliability. Thus, to recognize various plants the algorithm uses the gray-level image of plant leaf, while ignoring the color information.
3. To remove noise in the image, morphological operation, such as erode-dilation is applied on the gray scale image.
4. Erosion method erodes away the boundary of the foreground object. It is useful for removing small noises from the image. After erosion method, noise is removed however; the boundary of the original image is reduced in the process. Therefore, after eroding the image, we can recover the original boundary by applying dilation.

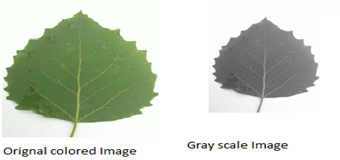


Figure 8: Preprocessing stage of leaf recognition process.

* + 1. Local Feature Extraction

Once image is preprocessed then it moves to the feature extraction stage. In the leaf recognition system the local feature of the image is extracted.

In Leaf recognition system, local features are used for describing the image.

For feature extraction, local features that represents highly localized information from small areas of an image, defined around interest points are used. It is assumed that interest points detected through the same method on similar images will produce similar local features. Hence, to compute features of an image, local feature descriptors are used instead of using general shape features such as contour and edge of an image.

In leaf recognition system, local feature of an image is computed using Scale Invariant Feature Transform (SIFT) as proposed by David G Lowe [17]. SIFT feature is invariant to rotation, scale [2].

The scale invariant feature transform consists of the following major stages of computation:

1. Scale-space extrema detection: The ﬁrst stage of SIFT key point detection is to identify locations and scales that can be repeatedly assigned under differing views of the same object. Detecting locations that are invariant to scale change of the image can be accomplished by searching for stable features across all possible scales, using a continuous function of scale known as scale space (Witkin,1983). It is implemented efficiently by using a difference-of-Gaussian function to identify potential interest points that are invariant to scale and orientation.

2. Key point localization: At each candidate location, a detailed model is fit to determine

location and scale. Key points are selected based on measures of their stability.

3. Orientation assignment: One or more orientations are assigned to each key point location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations.

4. Key point descriptor: The local image gradients are measured at the selected scale in the region around each key point. The key points are detected over a complete range of scales, meaning that small local features are available for matching small and highly occluded objects, while large key points perform well for images subject to noise and blur. Their computation is efﬁcient, so that several thousand key points can be extracted from a typical image [17].

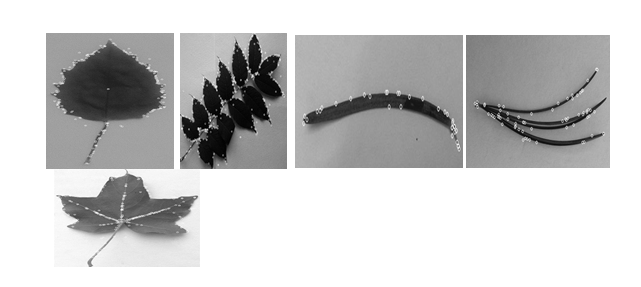


Figure 9: Sift key points from leaf images.

* + 1. Image Search

When a web service starts, all 450 field images from data set of 20 different species undergo preprocessing and feature extraction process. All the feature extracted from each image in database are stored in a data structure. Once query image request is given to the service, leaf recognition logic compares the query image SIFT point with each of training images precomputed SIFT points.

To identify the leaf, an instance based learning called nearest neighbor search is used on feature extracted from query image with all the feature extracted from images in database.

The goal of nearest neighbor search in leaf recognition system is to find the best candidate match for a query image key points among all the key point descriptor from training image. The nearest neighbor is defined as the key point with minimum Manhattan distance for the invariant descriptor vector.

KNN is also based on distance measures in feature space but instead of comparing feature set to a class representative value, it compares it to all samples of the feature training set and selects the first k closest ones.

Advantage of using nearest neighbor approach in the system is that KNN requires no training.

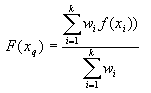
Also, it can be useful for in-the-field plant recognition work as the database of known species can easily be updated on-the-fly so that correctly identified new samples are further used for classification.

To reduce the outliers in KNN match following technique is used:

1. KNN match uses k = 2, to find the two nearest SIFT point in each instance of training image from each SIFT point in query image.
2. Only those SIFT points are considered from a database image instance whose distance from query images SIFT point is less than 300.
3. Ratio test is applied on two match points for each sift points of a query image. As per ratio test, match points are considered as valid match points only if the distance ratio between the first and second match point is greater than 0.75. Ratio test eliminates 90% of false matches and 5% of correct matches [17].
4. Further system performance is improved by using weighted KNN match. In weighted KNN, the closer neighbors are weighted more heavily than farther ones using the distance weighted function [3].

The goal of this algorithm is to give greater weight to closer neighbors.

Following formula is used



Where, the weight is

http://www.data-machine.com/nmtutorial/Images/image35_75.png

1. After applying weight to each instance of a leaf in a database now all the leaf instance that belong to one category are grouped together. Now, for each category a distance value is computed using Sum of each leaf instance average distance \* each leaf instance weight is computed. Each category has a distance. All leaf category is sorted in increasing order and top 3 shortest distance category is returned back to the client/user.
2. Weighted KNN search is sensitive to imbalanced data set. Therefore, each category/leaf species has total 20 leaf images. Images consist of leaf in different orientation, scales and brightness.
   1. Implementation

** Figure 10: Class Diagram

ILeafAnalysis interface serves two purposes

1. Client can provide list of images, which is used for both training and testing. Client will receive back performance metrics of the system. This approach is useful to repeatedly change the algorithms and their parameters and quickly receive back impact on performance of the system.
2. Secondly, clients such as windows phone application can provide a single query image and expect to receive back most relevant results back.
   1. Tools and frameworks

* EmguCV: .NET wrapper for OpenCV
* ASP.NET Web API
* Silverlight for windows phone
* Visual studio 2015 IDE
* Windows phone 8.1 emulator
  1. Software Development life cycle

System development life cycle follows the iterative + incremental model.

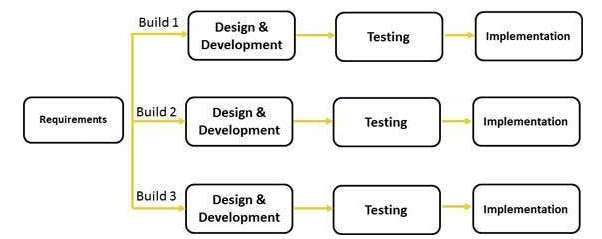


Figure 11: Iterative and Incremental model for project development.

For project development, incremental and iterative model is used.

Each decision of choosing one approach over the other is made after various prototype testing. This process allowed me to adapt and incorporate new changes quickly according to the feedback.

Weekly meeting was held with the committee chair and periodic status report sent to all committee members to show progress in the project and to take feedback.

# Chapter 4: EVALUATION

In this section, we will look at system performance using different types of data set. In addition, we will also analyze how different system configuration and parameters affect the performance of the system.

1. 1. Experimental Result

In order to test the system, dataset from LeafSnap is used for experiment. The data set consist of different combination of species to analyze the performance of algorithm. Each leaf species has balanced number of leaf images in each category. Each leaf species consists of twenty-leaf image in different orientation, scale and brightness.

To test the accuracy of the system 80% of the total leaf image is used for training and 20% of the leaf is used for query image.

Precision and Recall value are used to compute the accuracy of the system, using the following formula:-

Recall =

Precision =

To determine the performance of the system following scenarios are used.

1. Data set consist of 15 species from same and varied leaf family.

Data set used is shown in Figure 12.

|  |  |  |
| --- | --- | --- |
| Leaf species | Precision | Recall |
| abies\_concolor | 100 | 40 |
| abies\_nordmanniana | 100 | 60 |
| acer\_campestre | 50 | 80 |
| acer\_ginnala | 100 | 60 |
| acer\_pensylvanicum | 57.14 | 80 |
| betula\_lenta | 100 | 100 |
| betula\_populifolia | 50 | 100 |
| ginkgo\_biloba | 75 | 60 |
| malus\_hupehensis | 55.55 | 100 |
| pinus\_nigra | 100 | 80 |
| quercus\_bicolor | 0 | 0 |
| salix\_babylonica | 55.55 | 100 |
| taxodium\_distichum | 100 | 20 |
| ulmus\_rubra | 100 | 40 |
| zelkova\_serrata | 50 | 100 |

Each leaf species has 20 images each. Total images used for testing the system is 300.

Mixed species classification gives - Precision is 72.832% and recall is 68%.

Table 1: Testing the performance of the system on 15 leaf species shown in Figure 12.



Figure 12: Fifteen species of leaf used for testing leaf recognition algorithm.

Images containing shadow increases the number of false positive/negative, which in turn effect the precision and recall of the system.

Leaf category called quercus-bicolor has shadow in the image. SIFT detector falsely detects few Sift points on the shadow of the image which in turn effects the overall performance of the system. However, species such as betula-lenta do not have shadows in any of the image and system recognizes the leaf correctly.

1. Data set consist of all species from single leaf family called betula. Leaf family (betula) consist of 5 different species.

Each species has 20 leaf images. Total 100 images used in the system.

Intra leaf classification gives - Precision is 71.088% and recall is 78%.

|  |  |  |  |
| --- | --- | --- | --- |
| Leaf Name | Leaf Image | Precision | Recall |
| betula\_alleghaniensis |  | |  | | --- | | 60 | |  | |  | |  | |  | | |  | | --- | | 60 | |  | |  | |  | |  | |
| betula\_jacqemontii |  | 45.45455 | 100 |
| betula\_lenta |  | 100 | 50 |
| betula\_nigra |  | 83.33333 | 100 |
| betula\_populifolia |  | |  | | --- | | 66.66667 | | 80 |

Table 2: Performance analysis of the system on leaf with same family- Intra species classification

1. Data set consist of leaf species each from 15 different leaf family.

Each species has 20 leaf images. Total 300 images used for testing the system.

Inter Species classification gives - Precision is 80.68% and recall is 75.66%.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Leaf Species | Image | Precision | Recall |
| 1 | abies\_concolor |  | 66.66 | 40 |
| 2 | acer\_ginnala |  | 75 | 60 |
| 3 | betula\_lenta |  | 100 | 100 |
| 4 | carya\_cordiformis |  | 17.24 | 100 |
| 5 | ginkgo\_biloba |  | 75 | 60 |
| 6 | magnolia\_acuminata |  | 50 | 100 |
| 7 | malus\_hupehensis |  | 80 | 80 |
| 8 | pinus\_nigra |  | 100 | 80 |
| 9 | acer\_pensylvanicum |  | 100 | 80 |
| 10 | prunus\_yedoensis |  | 75 | 75 |
| 11 | quercus\_montana |  | 100 | 100 |
| 12 | salix\_babylonica |  | 100 | 100 |
| 13 | taxodium\_distichum |  | 100 | 20 |
| 14 | ulmus\_glabra |  | 100 | 40 |
| 15 | zelkova\_serrata |  | 71.42 | 100 |

Table 3:Performance analysis of the system on leaf with different family- Inetr species classification

When training data set consist of all categories that belongs to different families then system works better in terms of precision in comparison to intraspecies and mixed species classification. However, recall of intra species classification gives better result than inter species and mixed family species classification.

Figure 13: Performance analysis of inter species, intra species and mixed species classification

1. The precision and recall of KNN match and weighted KNN has been ascertained on five different species in the training data set. Each species has 15 leaves. Total 75 leaf images are used for the test.

|  |  |  |
| --- | --- | --- |
|  | Precision | Recall |
| KNN | 66.33 | 50.88 |
| Weighted KNN with Ratio Test | 85.22 | 67.46 |

Table 4: Performance of system with weighted and normal KNN algorithm

Figure 14: Performance analysis of KNN and weighted KNN on 5 data set

Weighted KNN performs better in terms of precision and recall than normal KNN match.

Distance weighted KNN is better over simple KNN because in weighted KNN all instances have a vote in deciding the outcome of the query image.

1. Effect of ratio test on overall precision and recall of the system.

Data set used is on 5 different species. Each species has 20 leaf images. Total image used is 100.

Figure 15: Performance analysis with different Ratio test value on 5 data set.

1. Effect of grayscale image and binary image on the overall accuracy of the system.

Data set consist of 20 different species. Each species has 20 images. Total image used is 400.

Precision and recall of gray scale image is better than binary scale image.

Figure 16: Performance analysis with gray scale and binary scale image

# Chapter 5: CONCLUSION

1. 1. Summary

In this paper, a Windows phone application for leaf species identiﬁcation has been presented. Using the application, user can photograph a single leaf on a white colored background. Given the input image, application will then analyze features of the leaf and identify the plant species at the real time. The leaf identiﬁcation process makes this application useful to amateur stakeholders as well as experts.

The system relies on computer vision for several key aspects, such as preprocessing the image by removing noise from the image, extracting scale and rotation invariant feature set from the image and retrieving the most similar species matches using weighted K nearest neighbor search on a data set of labelled images.

* 1. Future Work

1. In KNN match algorithm, all the instances in a data set are compared with the query image. Comparison of query image with each instance of database image is independent of each other and the time taken for this comparison is linear with the size of data set. With the increase in number of images in training dataset, the computation time will increase accordingly. However, by parallelizing the KNN match comparison process among multiple threads and processes can reduce the computation time of KNN match and can improve the execution time of the system. Using multithreading and map reduce framework KNN match algorithm can improve execution time of the system.
2. To segment leaf image from the background we can use binary image. Binary image can be used to mask leaf from original image.
3. Currently, shadow in the image increases the number of false positive regions and affects the overall performance of the system. It is given that falsely detected background region had a lower saturation value than true leaf region [13]. Using this information, lower saturation region of the image can be deselected using OTSU threshold on HSV color space of the image.
4. Perform usability testing on user interface of the system. Jakob Nielsen’s 10 Usability Heuristics can be used for testing user interface design [18].
5. Currently, top three most closely matched leaves from a query leaf is returned to the user. In future, we can add confidence level to each result by estimating the probability of estimating how close the result is to query image. Number of matched sift points and number of instances in the training data set for each category can help to compute the confidence level for each result.

Lessons learnt

1. Following an incremental development from the beginning helped me to adapt to change quickly. Each decision was made after various prototype testing. This allowed me to adapt and incorporate new changes quickly according to the feedbacks.
2. Learnt and successfully implemented various computer vision technologies for segmenting leaf and feature extraction.
3. Learnt about designing against interfaces, which enabled implementing various algorithms in a plug and play manner.
4. Designed and implemented an HTTP service using ASP.Net web api. Building an HTTP service makes the leaf recognition system available and accessible across cross-platforms.
5. Learnt and implemented client side of the application using windows phone Silverlight 8.1.
6. Learnt about diagnosing memory leaks in software and strategies to avoid unnecessary memory consumption and disposing resources as soon as they were no longer needed.
   1. Acknowledgement

I want to take the opportunity to thank everyone who made this capstone project possible. I gratefully acknowledge the supervision provided by my capstone project chair Professor Min Chen, for having assisted me in numerous ways, including scoping the project goals and coming up with strategies to improve the performance of the system. The discussions I had with her were indispensable to the project. Special thanks to my capstone project committee members Professor Mark Kochanski and Professor. Michael Stiber for their support and invaluable advice. I also extend my heartfelt thanks to my family for their support and guidance, which helped me in achieving my goals.

* 1. References

1. Caballero, C., & Aranda, M. C. (2010, July). Plant species identification using leaf image retrieval. In Proceedings of the ACM International Conference on Image and Video Retrieval (pp. 327-334). ACM.
2. Kumar, N., Belhumeur, P. N., Biswas, A., Jacobs, D. W., Kress, W. J., Lopez, I. C., & Soares, J. V. (2012). Leafsnap: A computer vision system for automatic plant species identification. In Computer Vision–ECCV 2012 (pp. 502-516). Springer Berlin Heidelberg.
3. Xia, C., Lee, J. M., Li, Y., Song, Y. H., Chung, B. K., & Chon, T. S. (2013). Plant leaf detection using modified active shape models. Biosystems Engineering, 116(1), 23-3.
4. Grand-Brochier, M., Vacavant, A., Cerutti, G., Bianchi, K., & Tougne, L. (2013, July). Comparative study of segmentation methods for tree leaves extraction. In Proceedings of the International Workshop on Video and Image Ground Truth in Computer Vision Applications (p. 7). ACM.
5. Shabanzade, M., Zahedi, M., & Aghvami, S. A. (2011). Combination of local descriptors and global features for leaf recognition. Signal & Image Processing: An International Journal, 2(3), 23-31.
6. Kadir, A. (2015). Leaf Identification Using Fourier Descriptors and Other Shape Features.
7. Prasad, S., Kudiri, K. M., & Tripathi, R. C. (2011, February). Relative sub-image based features for leaf recognition using support vector machine. InProceedings of the 2011 International Conference on Communication, Computing & Security (pp. 343-346). ACM.
8. Rui, Y., She, A. C., & Huang, T. S. (1996, August). Modified Fourier descriptors for shape representation-a practical approach. In Proc of First International Workshop on Image Databases and Multi Media Search (pp. 22-23).
9. Quan, L., Tan, P., Zeng, G., Yuan, L., Wang, J., & Kang, S. B. (2006, July). Image-based plant modeling. In ACM Transactions on Graphics (TOG) (Vol. 25, No. 3, pp. 599-604). ACM.
10. Gwo, C. Y., & Wei, C. H. (2013). Plant identification through images: Using feature extraction of key points on leaf contours1. Applications in plant sciences, 1(11).
11. Li, Y., Zhu, Q., Cao, Y., & Wang, C. (2005, October). A leaf vein extraction method based on snakes technique. In Neural Networks and Brain, 2005. ICNN&B'05. International Conference on (Vol. 2, pp. 885-888). IEEE.
12. Manh, A. G., Rabatel, G., Assemat, L., & Aldon, M. J. (2001). AE—automation and emerging technologies: weed leaf image segmentation by deformable templates. Journal of agricultural engineering research, 80(2), 139-146.
13. Arora, Akhil, et al. "A Plant Identification System using Shape and Morphological Features on Segmented Leaflets: Team IITK, CLEF 2012."CLEF (Online Working Notes/Labs/Workshop). 2012.
14. Gou, Jianping, et al. "A new distance-weighted k-nearest neighbor classifier."J. Inf. Comput. Sci 9.6 (2012): 1429-1436.
15. Bama, B. Sathya, et al. "Content based leaf image retrieval (CBLIR) using shape, color and texture features." Indian Journal of Computer Science and Engineering 2.2 (2011): 202-211.
16. Mouine, Sofiene, Itheri Yahiaoui, and Anne Verroust-Blondet. "Advanced shape context for plant species identification using leaf image retrieval."Proceedings of the 2nd ACM international conference on multimedia retrieval. ACM, 2012.
17. Lowe, David G. "Distinctive image features from scale-invariant keypoints."International journal of computer vision 60.2 (2004): 91-110.
18. Nielsen, J., and Molich, R. (1990). Heuristic evaluation of user interfaces, *Proc. ACM CHI'90 Conf.* (Seattle, WA, 1-5 April), 249-256.
19. Lisin, Dimitri A., et al. "Combining local and global image features for object class recognition." Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on. IEEE, 2005.