Leaf Classification Using Fourier Descriptors Project

University of Houston

COSC 4393, Fall 2016

Kevin Kwan

Salim El Awad

Xiaoya Wang

Yichuan Shi

Khoi Pham

Kun Yu

Introduction

Many images of objects in nature can be analyzed and group depending on their shape characteristics; and leaves in particular are very interesting. There are many unique leaf shapes and types, like oak tree leaf, pine leaf and maple tree leaf, etc. Some leaves are easily to recognized, while others only have minor differences between shapes. For example, red maple leaves have acuminate tip and notched lobes; a sugar maple has smooth lobes and obtuse tips. Therefore, different shapes may result in similar shape factors. Fourier descriptors (FD) are usually used for representing the shape of an object by using Fourier transform. That is why for this project, we will analyze the effectiveness of Fourier Descriptor in classifying different leaf shapes.

First, we convert color image to grayscale image, then apply threshold to grayscale image to get the binary image. After that, we smooth the image and extract the boundary, calculate the centroid and use it to find the centroid contour distance curve. Finally, we classify leaves by comparing their CCDC and eccentricity, and use these statistics to group leaves into different "classes."

Abstract

With nearly half a million species of plant in the world, classification of each species has been a complex task. With advances in technology and with the use of image processing algorithms, leaf classification has become a feasible task. This report will describe different procedures used to classify leaves, such as eccentricity, angle code histogram, and Fourier descriptor comparison, and how the approaches differ. This report will also discuss the complications that were discovered and how they were dealt with. Our results show that by combining a time warp distance, eccentricity, and angle code histogram test, we can expect a leaf detection accuracy of 70%-90%.

<u>**Pre-processing**</u>

Before finding the attributes of the leaves, we resized the image. This was done using the built-in function “imresize”. We knew that we would be eroding and dilating the image, so we made the images fairly the same size in terms of pixels. The images were rescaled to be 0.5 Megapixels. Afterwards, we binarized the image. This made it easier to morphologically smooth and find the boundary of the leaf.

After resizing the image, we created structuring elements for the erosion and dilation. This was primarily done to remove the stem, but it also smoothed the image and removed unwanted holes in the leaf. Many attempts were made with different structuring elements, sizes, and morphological smoothing functions, but the combination we found that removed the stem most accurately was erosion with a window of 25x25 pixels followed by dilation with a window of 30x30 pixels.

Once the stem was removed, we found the centroid of the leaf. Although this wasn’t necessarily needed at this time, we used it to find the point on the stem closest to the centroid. To do this, the modified image (the one the stem removed) was compared to the original binarized image, resulting in an image with just the stem left. The distance of all the points in the stem to the centroid were then compared, and the minimal one was saved. This is the starting position for the centroid contour distance curve (CCDC).

The next attribute to be found was the boundary of the leaf. This was done using the built-in function “bwmorph”. After applying these functions, the leaf’s centroid, boundary, and starting point were labelled. These attributes will later be used to calculate the Fourier descriptors of the leaf.

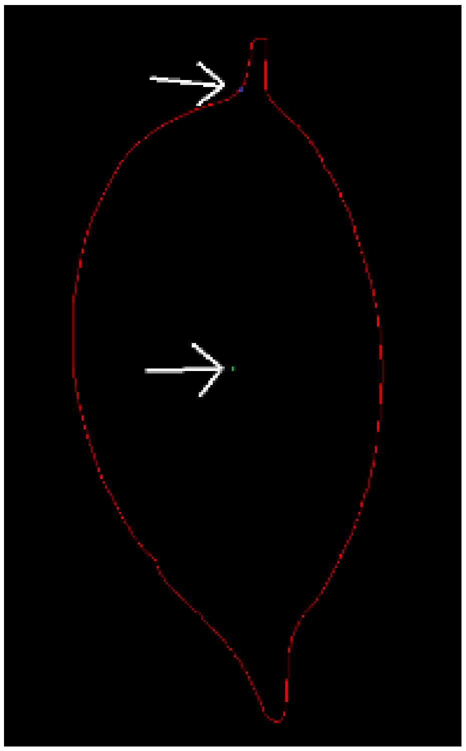
Figure 1 - Leaf with centroid, starting position, and boundary labelled

Process

We made sure to remove any variance that could cause an error in the calculation of the results. This included shift, rotation, and scale variance. In other words, we wanted to make sure that the physical attributes wouldn’t be a large comparison, but rather the key features of the leaf and its proportions.

The use of a common starting point and the CCDC satisfied invariance for all three of these. By calculating the centroid of the leaf from the boundary and using it to calculate the distances, we created shift invariance. No matter where the leaf was in the image, we were able to make sure that the algorithms were able to find it. The algorithms were also calculated relative to the center of the leaf rather than at a fixed point in the image to account for shift invariance.

After calculating the centroid of the leaf, we were able to find the eccentricity of it. This was the first recognition technique we used because it filtered the correct “shape” quickly.

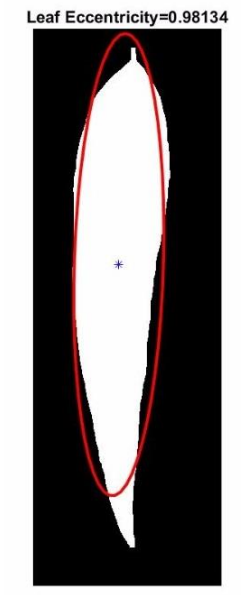

Figure 2 - Leaf with eccentricity labelled

In order to create rotation invariance, we determined the starting point of the algorithm and traced the CCDC clockwise. We decided that the starting point for the CCDC calculation would be the portion of the boundary that was closest to the stem removal. By doing this, we could verify that the plot for the Fourier descriptor would start and end at the stem. When tracing the boundary, we used the function “bwtraceboundary,” which allowed us to specify the rotation method of the tracing. Because the CCDC was calculated relative to the centroid and the starting position, we nullified any rotation variance.

The scale was the last variable accounted for. After calculating the CCDC, we applied the discrete Fourier transformation on it using the built-in function “fft”. We normalized the values between a range of [0,1], and took the first 64 coefficients to make sure we had an equal number of elements to other classes to calculate the IDFT and time warp distance (TWD). This was our second filtering technique. TWD allows us to compare notable features of each time series even if they don’t correspond to the same time.

In order to compute the angle code histogram of the leaf images, points on the contour of the leaf were connected to form line segments. The angles between adjacent line segments were then calculated and accumulated to form an angular histogram of an image. We used 20 line segments for each leaf image in order to keep our angle code histograms scale-invariant. To capture the subtleties of the shape properties of leaves, we found that increasing the number of bins in the angle code histogram lead to more accurate matching.

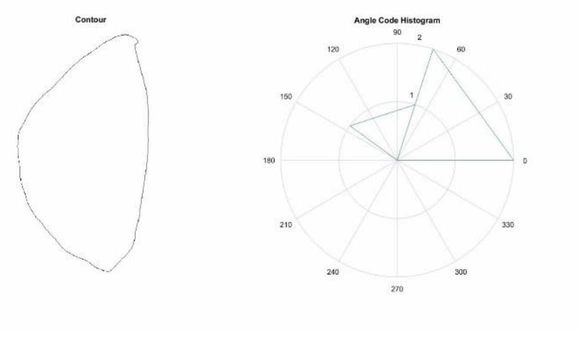

Figure 3 - Contour and angle code histogram

Training

In order to test any one leaf, we needed "prototype" signatures for each class of leaves. The techniques for each test were the same, but the average eccentricity, normalized time series, and angle code histogram for each set of leaves were taken. Because of the small sample size, two types of tests were done; one with the current tested leaf in its training set and one without. Including the leaf in its corresponding set allowed for a larger sample size, but it would skew the data to have higher results.

The results for the average eccentricities are as followed:

Class	a	b	c	d	g	h	i	j	k	l
Average Eccentricity	0.9795	0.7945	0.8705	0.8565	0.5867	0.7433	0.8583	0.782	0.8096	0.9647

Figure 4 - Average eccentricities

The following are some examples of the average normalized time series:

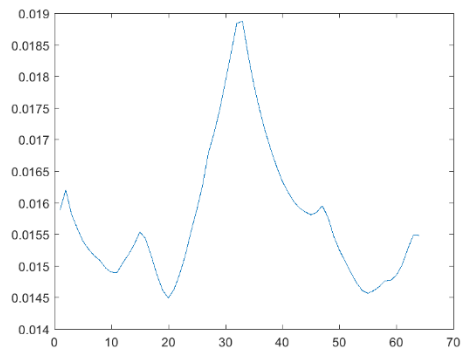

Figure 5 - Average normalized IDFT for set K

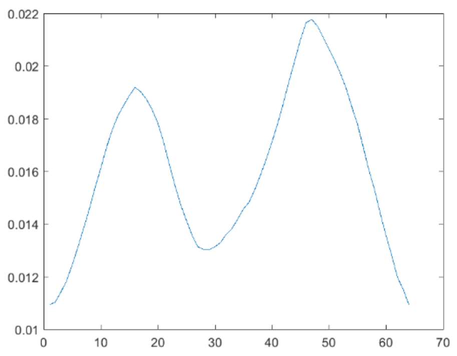

Figure 6 - Average normalized IDFT for set L

Algorithms

1. **readImage** – Reads the image as a matrix.
2. **find_center** – Finds the centroid of the leaf.
3. **getStartingPoint** – Finds the pixel on the stem that is closest to the centroid.
4. **getBoundary** – Finds the boundary of the leaf.
5. **getEcc** – Returns the eccentricity of the leaf.
6. **ccdc** – Returns the centroid contour distance curve the leaf when given the centroid, boundary, and starting point.
7. **fd** – Given the CCDC of a leaf, calculates the DFT of it, extracts the first 64 coefficients, performs IDFT. Returns the time series after all of these calculations.
8. **getDistance** – Returns the time warp distance (TWD) of the test leaf and the training sets.
9. **angleCodeHistogram** – Calculates the angle code histogram when given the boundary of the leaf. Returns a vector of all the angles calculated between line segments and a vector of the frequencies of each angle distributed among the bins.

Results

With the training of all of the sample data, the results were as followed:

Test	With Sample Included	Excluded
Eccentricity	60%	70%
FD + TWD	70%	50%
Angle Code Histogram	70%	40%
Weighted	90%	60%

Our testing consisted of four different tests: eccentricity, time warp distance, angle code histogram, and a weighted test that combined all three. Because of the low number of sample leaves we had, we ran the tests twice, once where we included the test images in the training set and once where we excluded them.

In the eccentricity test, we compared the test image eccentricity to that of all other classes, and selected the class with the closest eccentricity. This gave us a total of 60% correctness in the test where the images were included in the training set and surprisingly a 70% correctness in the test where we excluded the image from the training set.

For the time warp distance tests, we computed the normalized time series by applying inverse fast Fourier transform to the first 64 Fourier descriptors of the CCDC. In order to match our test to one of the leaf classes, we computed the time warp distance between the test's normalized time series and the average normalized time series for the class, and assigned the class with the least distance. This gave us a total of 70% correctness in the test where the images were included in the training set and a 50% correctness in the test where we excluded the image from the training set.

For the angle code histogram test, we computed the angle code histogram of the test and compared it to the one from each of the classes using Euclidean distance. The class with the closet angle code histogram was assigned to the test leaf. Angle code histogram test was the one most affected by excluding the test images from the training set, since we got a 70% correctness when included, but only a 40% correctness when the test images were excluded from the training data.

For the weighted test, we assigned an equal weight (33%) to all the tests. The way we did this was that for each of the test images, we did each of the corresponding calculations, and ranked the classes from least to greatest depending which classes was "closest" to the test image. We then assigned values of 1 to 10 for each class, summed the results for the three tests, and selected the class with the smallest total sum. When the test images were included in the training data, the weighted tests identified leaves with a 90% accuracy, and when the test images where not in the training data, the weighted test identified leaves with a 70% accuracy.

<u>Issues</u>

We ran into many issues while getting the functions to work properly. The first problem we faced was getting the erosion and dilation to work properly. Initially, we had not resized the images, so the pixel sizes of the stems varied greatly. Our first approach was to make the structuring element size proportional to the size of the image, but we thought that the we would run into proportion issues later on too, so we resized the image at the beginning.

Another issue we faced was getting the right structuring element sizes. Many times, the dilation of the image kept removing both the stem and the tip of the leaf, causing the program to sometimes think the tip was the stem. This occurred very often in leaves with large eccentricities because of their elongated shape.

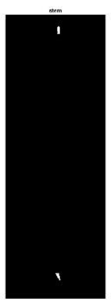
Figure 7 - Dilation with both stem and tip removed

The morphological smoothing technique also took quite a few tries to get right. We initially did erosion once and dilation twice, but we soon realized that erosion and dilation once each was the optimal technique. If done again, we could base the structuring element on the eccentricity of the leaf.

The hardest issue we had by far was getting our boundary tracing function to work properly. Because we needed the CCDC to measure the right angles, we had to ensure that the tracings were all either clockwise or counter-clockwise. Because of the way bwtraceboundary works, we could not get all of the tracings to go the same direction.

For example, if the tracing started at the top of the image for one image and the bottom for another image, and both were using the "clockwise" technique in bwtraceboundary, they would be tracing in different directions. Another issue with this would be if there was a convex bulge in the object. This could cause the function to not find the next pixel or trace in the wrong direction. If we were to do this project again, we could all get a degree in math and find an optimal algorithm which always traces the correct way.

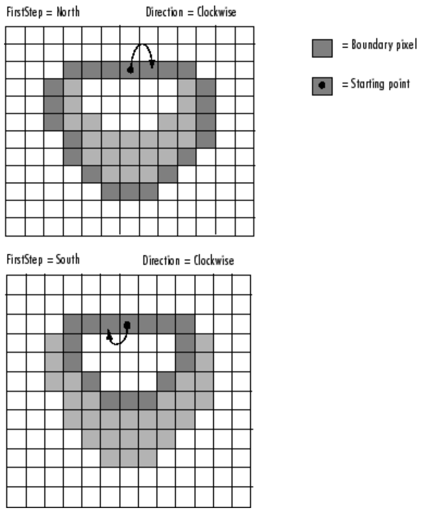

Figure 8 - bwtraceboundary method

Conclusion

Automating plant recognition has many applications such as species preservation, food supply management, and medicinal research. Because of this, the strive to find better techniques and algorithms to accurately classify leaves has become an essential task in technology. While this specific research pertained to leaf classification, a similar approach can be made to other morphologically complex objects.

<u>Group Contribution</u>

1. Pre-processing
    - Salim El Awad
    - Kevin Kwan
2. Eccentricity
    - Kevin Kwan
3. Stem Removal and Starting Point
    - Xiaoya Wang
    - Yichuan Shi
4. CCDC
    - Kevin Kwan
    - Yichuan Shi
5. Fourier Descriptor
    - Salim El Awad
    - Xiaoya Wang
6. Time Warp Distance
    - Khoi Pham
    - Kun Yu
    - Kevin Kwan
7. Result Analysis
    - Khoi Pham
    - Yichuan Shi
8. Report
    - Khoi Pham
    - Kun Yu
9. Main Function
    - Salim El Awad
    - Kun Yu
    - Xiaoya Wang

Name	**Percent Contribution**
Kevin Kwan	17%
Salim El Awad	17%
Xiaoya Wang	17%
Yichuan Shi	17%
Khoi Pham	17%
Kun Yu	17%

Works Cited

1. Karrels, Tyler. "Properties and Utility in Leaf Classification." (n.d.): n. pag. University of Wisconsin. 23 Oct. 2006. Web. 05 Dec. 2016. <https://homepages.cae.wisc.edu/~ece533/project/f06/karrels_rpt.pdf>.
2. Siravenha, Ana. Exploring the Use of Leaf Shape Frequencies for Plant Classification (n.d.): n. pag. Vale Institute of Technology. Vale Institute of Technology, 05 Mar. 2001. Web. 10 Dec. 2016. <http://sibgrapi.sid.inpe.br/col/sid.inpe.br/sibgrapi/2015/06.20.13.19/doc/example_v3.pdf>.
3. Izbicki, Mike. "Converting Images into Time Series for Data Mining." Converting Images into Time Series for Data Mining. N.p., 28 Oct. 2011. Web. 12 Dec. 2016. <https://izbicki.me/blog/converting-images-into-time-series-for-data-mining.html>.