Automatic Plant Leaf Classification for a Mobile Field Guide

An Android Application

David Knight
knightd@stanford.edu

James Painter
jpainter@stanford.edu

Matthew Potter
mpotter@stanford.edu

Stanford University
Department of Electrical Engineering
Stanford, California

***Abstract*—In this paper we describe the development of an Android application that gives users the ability to identify plant species based on photographs of the plant's leaves taken with a mobile phone. At the heart of this application is an algorithm that acquires morphological features of the leaves, computes well-documented metrics such as the angle code histogram (ACH), then classifies the species based on a novel combination of the computed metrics. The algorithm is first trained against several samples of known plant species and then used to classify unknown query species. Aided by features designed into the application such as touchscreen image rotation and contour preview, the algorithm is very successful in properly classifying species contained in the training library.**

I. INTRODUCTION

Historically, identifying an unknown plant species required the consultation of a heavy field guide, where the user had to make sometimes obscure observations of the plant's features and navigate a complex decision tree. The process was nontrivial even for seasoned botanists, and carrying the guides into the field was often impractical – particularly for hikers and other casual users. The advent of highly-portable, computationally-powerful smart phones with large storage capacity presents an opportunity to not only replace and improve the database and decision tree functions of the field guide, but also to create automatic leaf classification applications based on well-known image processing methods currently becoming standardized on mobile platforms. A user-friendly application on a popular mobile platform such as Android that is capable of identifying a good number of plant species could achieve widespread adoption, increasing the public's knowledge of and appreciation for their environment.

II. PRIOR AND RELATED WORK

The application of digital image processing techniques to the problem of automatic leaf classification began two decades ago and it has since been proceeding in earnest. In 1989, Petry and Kühbauch were the first to extract digital morphological features for use with identification models [1]. The technology found some of its earliest applications in industrial agriculture, where the desire was to separate crop species from weed species, allowing for decreased use of pesticides [2]. The problem is complicated in this application by complex backgrounds that make image segmentation difficult, but simplified by foreknowledge of one or two desirable crops amongst unwanted weed species [3]. Color and texture information is often sufficient to make this distinction, as opposed to more general applications, where numerous shape features must be acquired [4].

More recently, several groups have approached the problem of automatic leaf classification. Though the groups often use similar digital morphological features, e.g. rectangularity, sphericity, eccentricity, etc., there is great variation in how these measures are combined and used in classification. *Wang et al.*, for instance, used a two-tiered system, eliminating grossly different samples on the basis of eccentricity alone before making a finer judgment based on the combination of the centroid-contour distance (CCD) curve, ACH, and eccentricity [5]. *Du et al.* used roughly a dozen morphological features and moments and defined a classification method called the move median centers (MMC) hypersphere classifier, achieving a correct classification rate of over 90% [6]. *Wu et al.*, on the other hand, achieved similar results employing a probabilistic neural network [7]. Another approach looks at a polygonal approximation of the leaf's shape. [8]

III. DESCRIPTION OF ALGORITHM

At the highest level, our algorithm for classifying plant species proceeds as follows. An image of the leaf of interest is acquired and preprocessed to obtain a binary image of the leaf's contour. For several samples of leaves from a given species, morphological features are extracted from the contour image. This data is used to train the algorithm by determining the median value of each feature for a given species. As more samples are added to the training set, the algorithm becomes for effective at identifying a given species.

A. *Image Acquisition*

Currently, every mobile phone running the Android operating system is equipped with a high-quality digital camera with support for auto-focus and flash photography. The input to our algorithm is a photograph of a leaf of unknown species taken with the mobile phone's camera. Because the user has an interest in taking a picture which will provide useful information, it is reasonable to assume a certain degree of uniformity in the acquired images, i.e. the picture will be taken at a reasonable distance, in decent lighting, roughly normal to the surface, and against a background which provides sufficient contrast.

B. *Preprocessing*

Before extraction of morphological features can begin, the outline contour of the leaf must be found. The first step in this process is to convert the acquired color image to a grayscale image. Following this, image segmentation is performed to identify leaf pixels and background pixels. After holes have been closed and small regions removed, the segmented image is converted to binary and the interior of the leaf is subtracted, leaving an image of the leaf's outline contour [Fig. 1].

C. *Morphological Feature Extraction*

After an extensive review of literature and experimentation with various combinations of digital morphological features, we decided to include the following features in our final algorithm: aspect ratio, rectangularity, convex area ratio, convex perimeter ratio, sphericity, circularity, eccentricity, form factor, regional moments of inertia, and angle code histogram. The latter two features are based on invariable moments while the rest are geometrical. Each of the features, along with a helper feature, the centroid-contour distance curve is described below.

1) Centroid-contour Distance Curve: The CCD describes the distance from the leaf centroid to the leaf contour as the contour is traced in either a clockwise or counter-clockwise direction. In [5] *Wang, et al*. use CCD as a comparison metric between leaves; but, for a number of reasons, this is unsuited for our leaf comparison implementation. Although the CCD is translation-invariant and can be made scale-invariant by normalization, it requires accurate rotational alignment. *Wang et al*. further describe a leaf rotation algorithm, but its high complexity, along with high dimensionality in feature space, make real-time Android implementation impractical. Non-rotationally-aligned, unnormalized CCDs, however, are still useful for calculating parameters useful in calculating sphericity and circularity, as below.

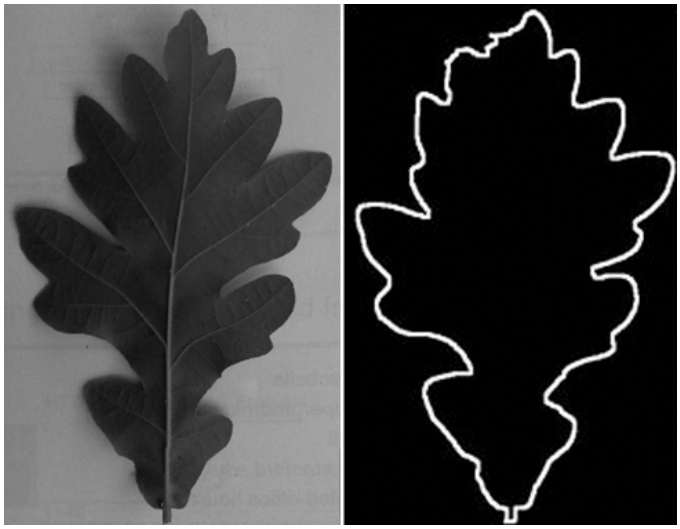

Figure 1. Sample leaf before and after preprocessing

2) Aspect Ratio (AR): The aspect ratio is the ratio between the maximum length D_{max} and the minimum length D_{min} of the minimum bounding rectangle (MBR) [Fig. 2a]

$$AR = D_{max} / D_{min}. \quad (1)$$

3) Rectangularity (R): Rectangularity is defined as the ratio between the region-of-interest (ROI) area and the MBR area

$$R = A_{ROI} / (D_{max} / D_{min}) \quad (2)$$

4) Convex Area Ratio (CAR): The convex area ratio is the ratio of the ROI area and the convex hull area (A_C) [Fig. 2b]

$$CAR = A_{ROI} / A_C \quad (3)$$

5) Convex Perimeter Ratio (CPR): The convex perimeter ratio is the ratio of the ROI perimeter (P_{ROI}) and the convex hull perimeter (P_C)

$$CPR = P_{ROI} / P_C. \quad (4)$$

6) Sphericity (S): Sphericity is the ratio of the radius of the incircle of the ROI (r_i) and the radius of the excircle of the ROI (r_c) [Fig. 2c]

$$S = r_i / r_c. \quad (5)$$

7) Circularity (C): Circularity is based on the bounding points of the ROI and is the ratio of the mean distance between the center of the ROI and all of the bounding points (μ_R)and the quadratic mean deviation of the mean distance (σ_R)

$$C = \mu_R / \sigma_R. \quad (6)$$

8) Eccentricity (E): Eccentricity is the ratio of the length of the main inertia axis of the ROI (E_A) and the length of the minor inertia axis of the ROI (E_B)

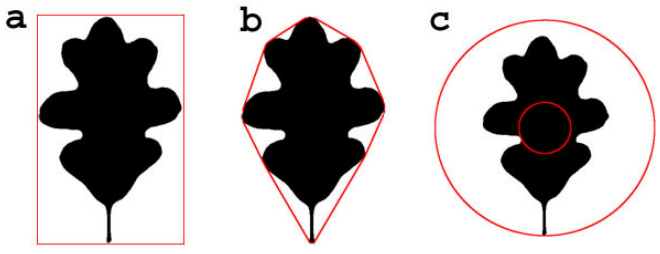

Figure 2. a) MBR, b) Convex Hull, c) Incircle and Excircle

$$E = E_A / E_B. \quad (7)$$

9) Form Factor (FF): Form factor is a well-known shape description characteristic given by

$$FF = 4\pi A_{ROI} / P^2_{ROI}. \quad (8)$$

10) Regional Moments of Inertia (RMI): Regional moments of inertia (RMI) capture spatial information about the weight distribution of the leaf at different positions along its vertical axis. Because our leaves maintain constant orientation and are generally symmetric about their vertical axis, I_{xx} quantities for four different regions are used as a descriptor. In order to make these descriptions scale-invariant, leaf area images are first cropped with a bounding box and uniformly resized to a height of 240 pixels before calculations are performed. After cropping and resizing, the four regions are defined as successive 60-pixel tall bands (Fig. 3). I_{xx} quantities are calculated independently for the area mass in each region

$$I_{xx} = \frac{1}{N} \sum_{x_i \in R} (x_i - x_{centroid})^2 \quad (9)$$

N: the number of points in the region R.

11) Angle Code Histogram (ACH): In [5] *Wang et al.* propose using the ACH to classify tree leaves. Points along the leaf contour are joined to make line segments and angles between adjacent line segments are measured. A histogram consisting of five uniformly-sized angle bins, each designated by a 1-5 angle "code," is populated with all angles measured along the contour. The histogram is normalized and used as a five-dimensional leaf classifier. Our test results show that contour point resolution (contour distance between two points on a line segment) plays a large role in ACH similarity measurements, and in order to make the ACH scale-invariant, we suggest the contour point resolution be a constant fraction of the leaf perimeter, or contour length.

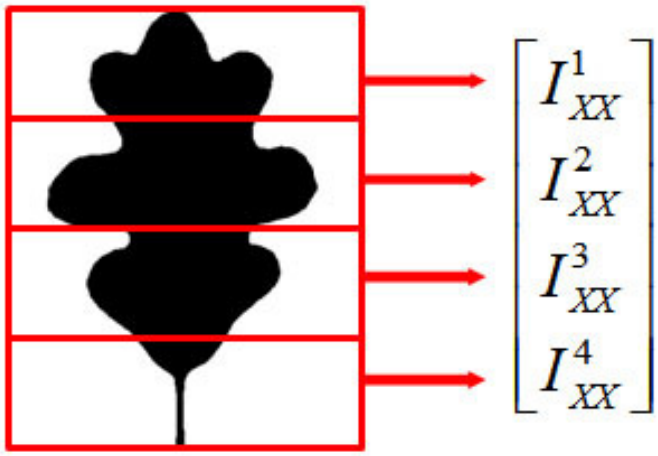

Figure 3. Regional moments of inertia

D. Training, Classification, and Matching

Fifteen samples of each leaf class were acquired for class training. They were segmented from the background and their contours were extracted in the same manner that query leaves were pre-processed before matching. The fifteen scores for each of the seventeen features within a class were averaged to determine the feature vector for the class. The resulting point in seventeen-dimensional space represents the class's identity.

Based on preliminary experimental results that showed non-uniform effectiveness of each feature, feature weighting was manually assigned. First, the four regional moments of inertia were scaled to 1/4 each, and the five ACH bins were scaled to 1/5 each, since the regional moments of inertia and ACH essentially describe one feature with multiple parts. However, because each of these features, in particular the center bin (zero-degree-range) of the ACH, proved to very accurately predict the correct class match, they were further weighted 2X, except for the ACH center bin, which was weighted 2.5X. The eccentricity feature at times was the most accurate match predictor, but other times was by far the worst, so its weight was assigned 1/2X. Total weights for the ten features are shown in Table 1.

To determine a query leaf's correct match, the query leaf's feature vector is computed and its difference from each leaf class represents its feature similarity vector.. The feature similarity vectors are then normalized and weighted, and their Euclidean distance is calculated in the high-dimensional space. The order of the resulting vector of class distances represents the order of similarity.

E. Android Implementation

The classification algorithm described above was initially verified in MATLAB. The MATLAB algorithms were then used as a guide to develop the Android implementation. Since many of MATLAB's image processing toolbox functions are not available in the standard Java or Android libraries, higher level operations like convex hull, fill area, moments of inertia, and others were written manually. Note also that because a network connection might not be available while hiking, we opted to store pre-generated training data and perform classification calculations locally on the handset.

Aspect ratio	1
Rectangularity	1
Convex hull area ratio	1
Convex hull perimeter ratio	1
Sphericity	1
Circularity	1
Eccentricity	0.5
Form Factor	1
Regional moments of inertia	[0.5, 0.5, 0.5, 0.5]
Angle code histogram	[0.4, 0.4, 0.5, 0.4, 0.4]

Table 1. Feature Weights

The first screen presented to the user is a live viewfinder with an overlay of the current leaf segmentation [Fig. 3a]. This feature allows the user to review the results of the segmentation algorithm and take the picture when the region is acceptably highlighted. The second screen shows the region extracted by the segmentation algorithm and presents a touch-based rotation interface that lets the user orient the leaf shape in an upright position [Fig. 3b]. Transforming all leaf images into a uniform orientation can be a computationally intensive process [5], so avoiding this process in a usable manner is advantageous given the restricted processing resources available on mobile phones. Additionally, the user's experience is improved through the interactivity of these features.

Once the user corrects the leaf shape orientation, a busy animation is displayed while the application generates a feature vector and performs distance calculations against class centers. Finally, the distance between the generated feature vector and each of the six class centers is displayed in a list. The class with the lowest reported distance is the class the algorithm has identified as the likely species of the query leaf.

IV. EXPERIMENTAL RESULTS

A. *MATLAB Implementation Results*

In order to test our classification system, fifteen query leaves for each of the six leaf classes were acquired and queried against the trained classes. Average similarity scores to each class are shown in Table 2. Every query class was correctly matched on average, and few individual leaves were incorrectly matched. Despite the strong visual similarity between classes 1 and 2, they were never cross-matched. Class 6 performed similarly well when queried against the visually similar class 3, but reversing the situation and querying class 3 against class 6 triggered the most false positives in the experiment. In spite of this, the 80% success rate shows the robustness of the chosen classifiers.

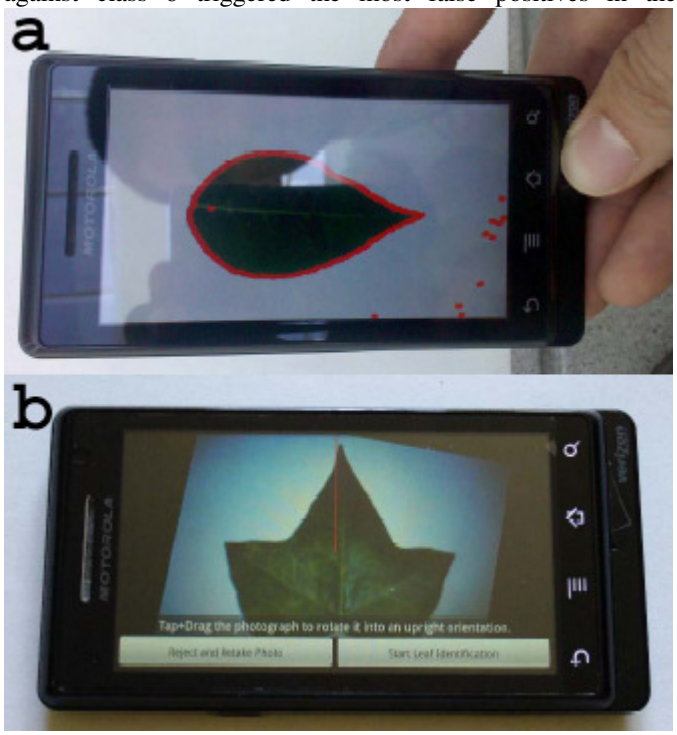

Figure 4. a) Contour preview b) touchscreen rotation UI

B. *Android Field Test Results*

As a field test of our Android application, we attempted to classify five leaves from each of our six classes in nature. To maintain statistical independence, a new set of leaves was gathered for the field test, while the leaves used to generate the offline MATLAB results were used to train the application.

Looking at Table 3, leaf classes 1, 2, and 6 had high rates of successful classification, leaf classes 3, 4, and 5 faired significantly worse than results obtained offline in MATLAB. Leaf classes 3 and 4 were systematically classified as classes 6 and 5, respectively. This consistent behavior may be due to poor representative class centers, in which case larger training sets may offer improvement. Additionally, we discovered that the Java implementation of the angle-code histogram was unreliable, sometimes giving erratic results such as completely empty bins. This malfunction likely affects classification results since the angle-code histogram occupies five dimensions of the feature space.

V. CONCLUSIONS AND FUTURE WORK

To improve usability, a welcome improvement would be to optimize the speed of the live viewfinder segmentation algorithm. The current implementation uses a large amount of iteration that might be avoidable, and the low framerate and slight delay of the current viewfinder overlay can make it difficult to point precisely at skinny leaf regions.

At the morphological feature level, the latest implementation of the ACH feature is not scale invariant, due to the constant separation of points along the contour. Contour point separation should change proportionally with the total contour length (leaf perimeter size). Although this did not create a large problem between our trained set (contour images of 500-pixel length in max dimension) and query set (contour images of 400-pixel length in max direction), it could be problematic in other scenarios, especially when weighted as heavily as it is.

Scale-wise, as the number of classes begins to increase, the feature space becomes more crowded, limiting the precision with which the classification system can distinguish between similar leaf classes. Additionally, because each query leaf will need to be queried against each class, computation time will increase with the number of classes. However, future refinement of the weighting used with the classifying features may show that certain feature vector components can be removed with little impact, thereby reducing the dimensionality of the feature space.

Lastly, while it is possible for this classification method to

	Class1	Class2	Class3	Class4	Class5	Class6	Margin
Class1	**0.31**	0.89	2.44	2.18	2.11	1.71	187%
Class2	1.43	**0.17**	2.23	2.02	1.91	1.49	741%
Class3	2.53	2.12	**0.53**	1.93	1.82	0.88	66.0%
Class4	2.49	2.18	2.05	**0.39**	0.85	1.61	118%
Class5	2.41	2.07	2.10	1.01	**0.32**	1.55	216%
Class6	2.36	1.91	1.15	1.90	1.84	**0.24**	379%

Table 2. Leaf class match scores (MATLAB)

distinguish between leaf classes that are significantly different, performance may suffer when attempting to separate samples from plant species that are either related or have similar shape characteristics. However, if a hierarchy of leaf classes were constructed, with shape at the highest level, our method may function well as a first-stage algorithm, applying general labels to leaves (e.g. long and skinny, squat and wavy, fan-like, etc.) before passing the sample off to a classifier that is more adept at distinguishing between leaves sharing the identified characteristics.

We have described an Android mobile application, the core of which is an algorithm that uses digital morphological features to classify plant species based on the nearest neighbor distance of the query leaf's features from the median features of each species in the training set. Our method proved quite robust under reasonable conditions. With further additions to the training set, the application can easily be tuned to recognize more plant species. This capability coupled with a polished user interface and hyperlinked information on the identified plant species could result in a very compelling mobile application, opening the surrounding environment to large numbers of users.

	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Margin
Class 1	**1.133**	1.383	2.203	1.982	1.967	1.518	22%
Class 2	1.469	**0.860**	2.147	1.957	1.846	1.321	54%
Class 3	2.000	1.595	1.650	2.053	1.926	**1.385**	-16%
Class 4	2.025	1.582	2.127	1.551	**1.308**	1.565	-16%
Class 5	2.129	1.687	2.101	1.514	**1.379**	1.615	10%
Class 6	2.089	1.675	2.001	1.841	2.022	**1.350**	24%

Table 3. Leaf class match scores (Android)

REFERENCES

[1] W. Petry and W. Kuhbauch, "Automated discrimination of weed species with quantitative image analysis," Journal of Agronomy & Crop Science, vol. 163, pp. 345-351, 1989.

[2] R.D. Tillet, "Image analysis for agricultural processes: a review of potential opportunities," Journal of Agricultural Engineering Research, vol. 12, pp. 247-258, 1991.

[3] J. Hemming and T. Rath, "Computer-vision based weed identification under field conditions using controlled lighting," Journal of Agricultural Engineering Research, vol. 78-3, pp. 233-243, March 2001.

[4] B. Åstrand and A. Baerveldt, "An agricultural mobile robot with vision-based perception for mechanical weed control," Autonomous Robots, vol. 13-1, pp. 21-35, July 2002.

[5] Z. Wang, Z. Chi, and D. Feng, "Shape based leaf image retrieval," IEEE Proceedings: Vision, Image, and Signal Processing, vol. 150-1, pp. 34-43, February 2003.

[6] J. Du, X. Wang, and G. Zhang, "Leaf shape based plant species recognition," Applied Mathematics and Computation, vol. 185-2, pp. 883-893, February 2007.

[7] S. Wu, F. Bao, E. Xu, Y. Wang, Y. Chang, and Q. Xiang, "A leaf recognition algorithm for plant classification using probabilistic neural network," in *Proceedings of 2007 IEEE International Symposium on Signal Processing and Information Technology*, Giza, December 2007.

[8] C. Im, H. Nishida, and T.L. Kunil, "Recognizing plant species by leaf shapes—a case study of the *acer* family," in *Proceedings of 1998 IEEE International Conference on Pattern Recognition*, Brisbane, August 1998.

APPENDIX A: BREAKDOWN OF WORK

A. David Knight

Review of literature, algorithm research, preparation of final report, Android feature vectors, Android UI, Android classifier, leaf samples, moment of inertia feature.

B. James Painter

Review of literature, algorithm research, preparation of final report, MATLAB implementation, leaf training, Android preprocessing, poster layout, leaf samples.

C. Matthew Potter

Review of literature, algorithm research, preparation of final report.

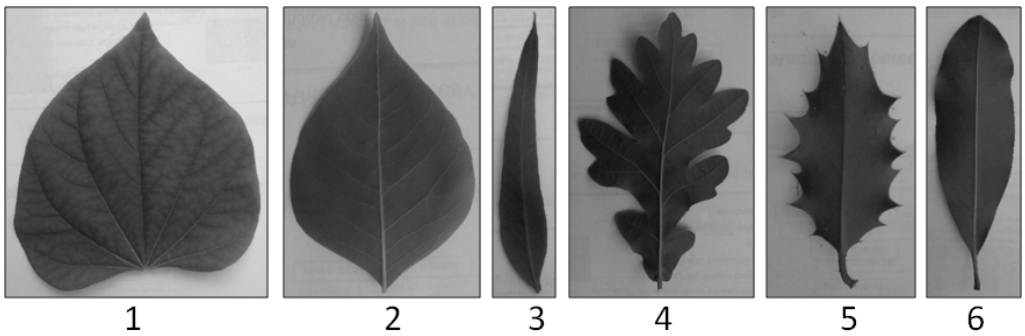

Figure 5. Leaf classes