Scoring Shape Characters of Monocot Leaves

Yao Zhou
Department of EECS
Oregon State University
Corvallis, OR, 97330
zhouyao@onid.oregonstate.edu

Abstract

Biologists regularly collect images of leaves for their further studies. One such biological study of leaves is scoring the phenomic characters of leaves for the construction of the Tree of Life (ToL), i.e. the evolutionary lineage of taxa in botany. There is an opportunity for computer vision to help biologists automate this character scoring. In this master's report, we describe an automatic system for scoring shape characters of monocot leaves. This system uses computer vision to speed up the current manual scoring process. Each leaf is processed using following computational steps: Segmentation, Orientation Alignment and Scoring. We developed two main frameworks for shape character scoring. One uses global leaf descriptors and then applies a Support Vector Machine (SVM). The other matches the leaf shape contour to exemplar shape contours using Dynamic Time Warping (DTW). Our evaluation shows that both frameworks give high performance on scoring leaves. To the best of our knowledge, this is the first computer vision system that addresses the problem of leaf shape character scoring.

1 Introduction

1.1 Motivation

Biologists collect large amounts of data on living organisms to infer their evolutionary histories. Phenomic data are the fundamentals in the construction of the Tree of Life (ToL) [1] which represents the evolution history of all organisms. Increasing amounts of data and advancing technologies have dramatically expanded the usage of computer vision expertise in biology. One such usages is scoring the phenomic characters of leaves for their further studies. Currently large numbers of images are collected, but only a small number of them have been scored because manual scoring of leaves is inefficient and costly. Therefore we explored the possibility of helping biologists to automatically score leaf shape characters from images. We identified the computer vision problems that are related to leaf scoring and proposed computer vision algorithms to solve them. No prior work has addressed the same problem before. Because there are general research on shapes but not for leaf contours, so we need to adapt these algorithms to suit our problem.

1.2 Problem Statement

Given a monocot leaf image, our objective is to correctly score the shape characters of the leaf. We focus on scoring six shape characters: Leaf apex angle, Leaf apex curvature, Leaf base angle, Leaf base curvature, Length-to-width ratio and Widest position. The shape characters have several different states. For instance, Leaf base angle has four states: Acute, Obtuse, Straight and Reflex as illustrated in Figure1.

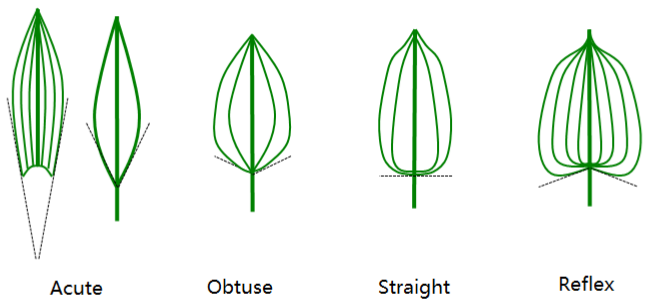


Figure 1: Four States of Leaf Base Angle. Acute: less than 90 degrees; Obtuse: between 90 and 180 degrees; Straight: 180 degrees; Reflex: greater than 180 degrees.

Scoring a character means identifying the correct state of each shape character of a monocot leaf. In computer vision, this task can be represented as a multi-class classification problem for each shape character. The multiple states of the character can be viewed as a set of classes.

1.3 Challenges

The images in the dataset have various image qualities. Some images are high resolution image scans and other images are collected by biologists through cameras. Also the images have various backgrounds, which may include specimen tags, a measuring gauge, immersion liquid as well as inconsistent lighting conditions. The first challenge is how to build a preprocessing module which can work on images with varying qualities. The second challenge is that the leaf specimens are under different preservation conditions. Some specimen leaves are severely damaged: they may have cracked into two parts or lost some tissues around their edge. Other specimen leaves are almost transparent, and therefore hardly to be distinguished from immerse liquid and some have overlay with specimen tags and measuring gauge. This substantial diversity of our real world dataset poses a number of challenges, and we believe there are no off-the-shelf computer vision tools specifically designed for this purpose. The third challenge is that we need to make sure the system we design is a flexible framework so that each system module can be easily replaced by third-party modules. This flexibility is required as shape characters are not the only category of characters that biologists care about.

1.4 Related Work

Currently there is various software available for leaf species recognition [2, 7, 14, 20] and general object detection. Leaf species detection involves identifying the plant species from photographs of their leaves. Usually the recognition process consists of several steps: Removing unqualified leaves, separating the foreground and background and extracting features and comparing. However our task is focused on scoring the leaf shape characters, which may have partial overlap with these existing approaches to leaf species recognition. Some general object detections methods, such as Discriminative deformable model (DPM) [9] or the classical Bag of Words (BoW) model [8], can successfully recognize certain types of objects from images. However each approach in prior work can only address one aspect of the challenges we listed.

To score characters of shapes, the existing contour shape techniques that exploit shape boundary information are very promising. There are generally two different approaches for contour shape representation and matching. One approach generates two sequences of points for two shapes by sampling contours in a specified order, and their distance is calculated using a sequence matching method such as Dynamic Time Warping (DTW). The second approach is a general correspondence-based shape matching method that measures the similarity between shapes using point-to-point mapping. When working with the correspondence methods, several type of relations can be applied, including one-to-one mapping, one-to-many mapping and many-to-many mapping. Generally speaking, all of these matching methods are time-consuming and too difficult to use by people without expert knowledge of computer vision. Therefore we developed a system that has two distinct frameworks to address these challenges while also being easy to use by biologists.

1.5 Overview of Approach

Our approach has two main stages:

- **Preprocessing and Feature Extraction:**
  The first stage begins with foreground background segmentation, which serves the purpose of separating the leaf out of the image. Next the leaf is rotated & aligned to canonical orientation and then features extraction methods are applied to the leaf.
- **Shape Character Scoring**
  The second stage is comprised of two basic frameworks for character scoring: One framework uses Histogram of Oriented Gradient (HOG) features combined with a Support Vector Machine (SVM) classifier. This is a generic method that has potential to be used on other characters, such as vein characters. The second framework matches the leaf shape to exemplar shape contours using DTW and uses K-Nearest Neighbour (K-NN) for scoring. The second framework is more suitable for shape characters.

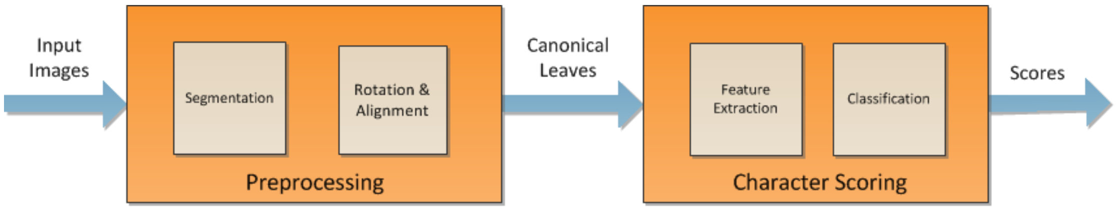


Figure 2: Flow Block Illustration of Our Approach

2 Approaches

2.1 Preprocessing and Feature Extraction

Preprocessing plays an important role before we get to the shape character scoring stage. It serves the purpose of cropping and putting the leaf into canonical orientation, which is leaf apex area on the left side and leaf base area on right side. Preprocessing has two sub-modules: the first sub-module is to segment the input image and separate the leaf from the background. The second module is orientation alignment that puts every leaf into the canonical orientation. Having each leaf in the canonical orientation makes it convenient for shape character scoring. Orientation alignment has two separate steps as shown in Figure 3. The first step is to compute a segmentation mask for the leaf and then crop the leaf out of the original image. Then the leaf is rotated to the horizontal position. The second step is orientation alignment that puts the leaf into the canonical orientation.

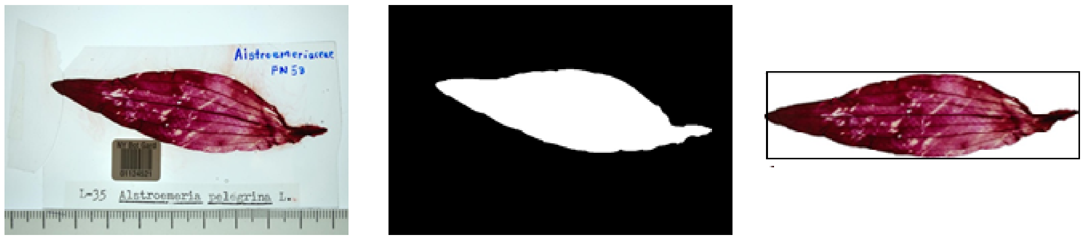


Figure 3: One example of Segmentation & Rotation. Left most is the original input image with cluttered background. Middle is the binary mask generated by CRF. Rightmost is the cropped and rotated leaf area image with the background masked.

2.1.1 Foreground Background Segmentation

There are a wide variety of segmenting methods available in computer vision community. One category of methods is unsupervised segmentation [5, 13, 15, 16] such as Normalized Cut [18] and Graph-based methode [10]. An advantage of these methods is that they do not require an annotated training set. Another category of methods is supervised segmentation which requires human annotations before training the model, such as Conditional Markov Random Field (CRF) [11, 12]. We use a CRF model for multi-class labelling with the CRF output mapped to a binary mask image (1 inside the mask, 0 for the rest). There are several reasons to do the segmentation in this way. Unsupervised methods such as Normalized Cut can easily confuse the leaf area with specimen tags when these two areas have similar color. Interactive segmentation methods can output a very good segmentation image given multiple human interactions. However it requires substantial human labour which is not available in our case since we have a large quantity of raw leaf images. Compared with these methods, the CRF model is very promising. First, it is a supervised pixel-level labelling method, and we can explicitly set up several region definitions in configuration files. In this way, some easily-confused regions can be correctly separated. Second, training and using the model is not costly. We can simply apply the trained model on unseen raw leaf images once the training is finished. Compared with interactive segmentation methods, the CRF requires only a small amount of human labor for the annotation.

2.1.2 Rotation

The first step of orientation alignment is rotation which will put the leaf into the horizontal position. There are two cases: if the aspect ratio of the cropped leaf is larger than certain threshold λ_r, we fit an ellipse to the binary mask image and then rotate the leaf to horizontal position based on its main axis direction. In practice, we take the values of λ_r as 2. The second case is used when the aspect ratio is smaller than λ_r, namely, the leaf area is round shaped. In this case, the ellipse is no longer a good leaf shape approximation. Therefore we apply edge detection to the leaf and then compute Hough transform on these edges to find a set of mini directional line segments. We then compute a directional histogram by counting how many mini line segments have fallen into each bin. The directional bin containing the largest number of mini line segments defines the main axis and we rotate leaf area based on it. The general rationale behind the second approach is to find the main vein direction and rotate the leaf so that this direction is horizontal.

2.1.3 Orientation

After the leaf has been rotated to be horizontal, it is still not clear whether the leaf is in the canonical orientation. We extract feature vectors from both ends of the leaf (Apex and Base) and then train a binary classifier to recognize the leaf orientation. This binary classifier is trained based on the feature vectors extracted from canonical leaves and label predictions are simply ”1” or ”0”. Label ”1” means the leaf is already in canonical position and no more operations need to be done. Label ”0” means the leaf is positioned in the opposite orientation and it has to be rotated 180 degrees.

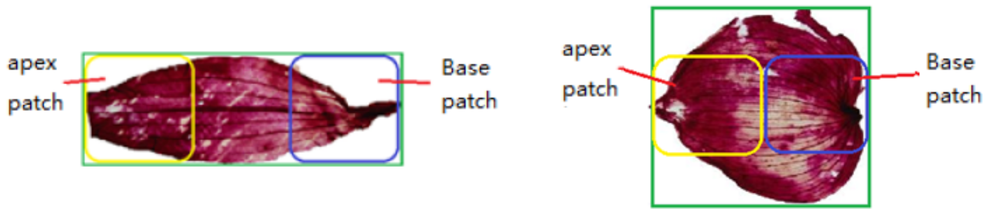


Figure 4: Example of extracting feature patches from leaf area image.

To train a classifier, we need the feature vectors to have all the same length. Therefore all leaf patches cropped from the leaf are resized to squares of same size. In our implementation, all leaf patches are resized to 512-by-512 pixels. Then we extract HOG [6] features on two square patches of each leaf and concatenate them as one feature vector in the end. During the classifier training, one feature vector represents one leaf. If this leaf is already on a canonical view, the feature vector extracted from it is labelled as a positive instance, otherwise the extracted feature vector is labelled as negative instance. For implementation purposes, we have also applied some common techniques to expand the dataset size. This expansion is accomplished by simply flipping every leaf patch vertically. This generates more Apex and Base patches.

After the Orientation alignment stage is finished, we can easily score the first two leaf shape characters: Length-to-width ratio and Widest position. We fit a rectangle on the mask image and the Length-to-width ratio is the outcome of rectangle length divided by rectangle width. For the second shape character, we calculate the accumulated foreground pixel numbers column by column, then the mask image is collapsed to an one dimensional vector v. Finally this vector v is discretized into three bins and the score of Widest position character is the index of the bin that $max(v)$ falls into.

2.1.4 Extracting Features

2.1.4.1 HOG features

The first framework is similar to the method we use in Orientation alignment stage – we use the HOG features extracted from two square patches of the leaf. HOG is the feature descriptor used widely in the computer vision community. It counts the occurrence of gradient orientation of a local portion of image. More specifically, HOG decomposes an image into small square cells, computes a histogram of orientated gradients in each cell, normalizes the histogram and returns the descriptor for each cell. Concatenating all the descriptors of cells in an image as one vector gives an image descriptor. This descriptor has been successfully applied in pedestrian detection problems(and many others). HOG is a capable descriptor that describes local image appearance and shape and thus we hypothesis that it can be a universal descriptor for all our character scoring tasks.

2.1.4.2 Shape feature

The second framework scores the leaf shape characters by using only shape features on the mask image. There are multiple ways to properly describe shape features. We compute two descriptors: Shape context [3] and Beam angle [17].

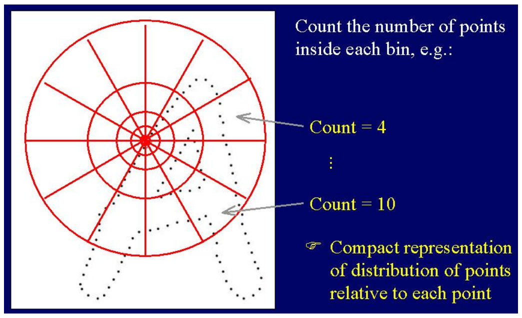

Count the number of points inside each bin, e.g.:
Count = 4
⋮
Count = 10
☞ Compact representation of distribution of points relative to each point

Figure 5: Example of shape context descriptor on one sampling point.

Shape context was originally proposed to describe shapes, and it can be used for measuring shape similarity. The idea of shape context is to sample n points on the contour of shape. For each sampling point p_i, calculate $n-1$ vectors by connecting p_i to the $n-1$ remaining sampling points. Then the set of vectors can be discretized into a log-polar histogram, which still includes enough information to describe the shape. The log-polar histogram is decided by two parameters: number of bins of theta n_θ and number of bins of radius n_r. In our implementation, we take n_θ = 12 and n_r = 5, therefore the total number of bins in one shape context descriptor is 60. The shape context of point p_i is defined as:

$$h_i(k) = \#\{q \neq p_i : (q - p_i) \in bin(k)\} \tag{1}$$

In the character scoring stage, we build the pairwise similarity matrix over the whole dataset and shape context is an ideal descriptor for that purpose.

Beam angle is the second shape descriptor we use. The contour of the leaf can also be characterized as a sequence of beam angle descriptors, which are also histograms at the sampling points along the contour. For each sampling point p_i, it calculates the angle θ of a triplet (p_i, p_{i-j}, p_{i+k}) with angle vertex at p_i. Here, $j + k \leq n$ and n is the total number of sampling points along the contour. Since the histogram on each sampling points includes all possible combination of triplets, beam angle is a very powerful and discriminative descriptor. In the implementation, the total number of bins $n_b a$ = 30. Also there is a special case that needs to be taken care of: when the angular bisector of a triplet is on the opposite direction of mass center of the mask, we need to replace θ with $360 - \theta$.

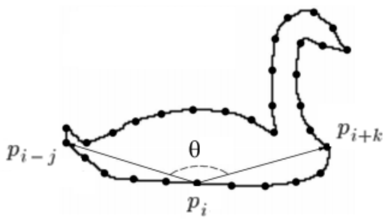


Figure 6: Example of one beam angle triplet (p_i, p_{i-j}, p_{i+k}) on point p_i

2.1.5 Contour Smoothing

Before we generate the contour and extract features from it, we may elect to do the contour smoothing first. The fuzzy boundary of the mask can be filtered by removing high frequency local components. To design a low pass filter for removing the high frequency boundary, we transform the contour into a one dimensional signal first. This transformation uses the center of mass as the origin. Then beginning with the start direction (three o'clock for instance), it rotates the mask along with certain angular step size and record the distance from the contour to the mass center. After the mask is rotated 360 degrees, the one dimensional distance signal that has been recorded can represent this two dimensional mask. Next, we remove the high frequency components using Fourier filtering. We can either heuristically pick a frequency threshold to design a low pass filter for the signal or choose the frequency threshold by considering how much energy we want to retain after filtering. Recovering the two dimensional mask from the one dimensional signal is the reverse of the initial step. The only difference is that we need to compensate for amplitude scaling after a Reverse Fourier Transform.

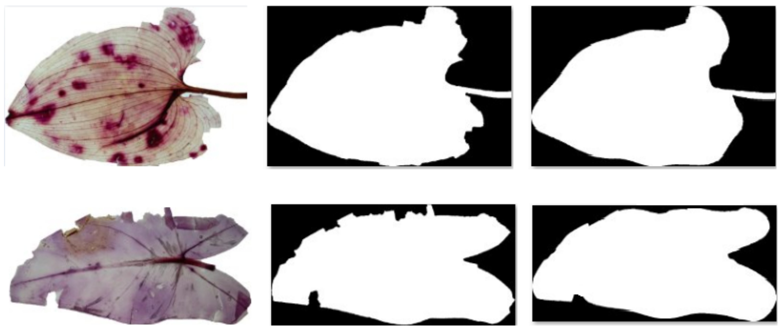


Figure 7: Two examples of smoothing leaf shape contours

3 Shape Character Scoring

For shape character scoring, we have developed two alternative frameworks: One is scoring using SVM on HOG features. The second one is matching the leaf shape contour using DTW to exemplar shape contours to build the pairwise distance matrix first. Then the K-NN classifier is applied for final scoring.

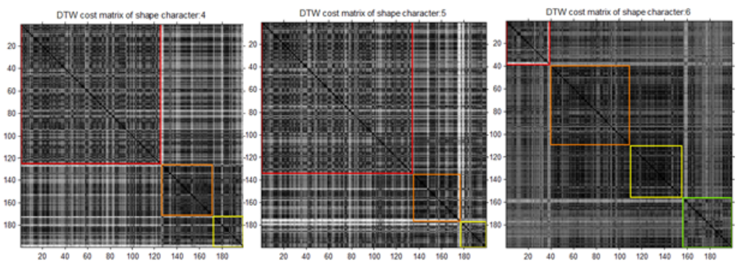
3.1 Framework I: HOG plus SVM

Framework I of shape character scoring is similar to the method we use in the Orientation alignment stage. We extract HOG features from two resized square patches (Apex and Base) of the leaf, then concatenate these two feature vectors as one. The main difference between Framework I and Orientation alignment is that in the scoring stage we train a multi-class SVM for each category of shape characters. However the final feature vector we retrieved has very high dimension. In order to speed up the training, we apply Principle Component Analysis (PCA) to reduce feature dimensions, and this reduced-dimension vector is fed into the multi-class SVM.

3.2 Framework II: Shape plus DTW

Framework II scores the leaf shape characters in a different way such that only the mask image is used for feature extraction and model training. To begin with, an Apex/Base patch is cropped from the contour image and an open contour of Apex/Base is retrieved. Then points are uniformly sampled along this open contour. The distance between two open contours is estimated by using the standard Dynamic Time Warping (DTW) on the corresponding two sequences of shape descriptors. By applying this procedure to every pair of images, a pairwise distance matrix D is built over the whole dataset as shown in Figure 8 . We expect the diagonal elements along the pairwise distance matrix D will be 0, since the cost of between two exactly same contours is zero.

Final shape character scoring uses the K-NN classification algorithm on D to score the test leaves. For instance, the ith row on D represents for the distances between ith leaf and the rest. Then based on the choice of parameter K, K-NN assigns ith leaf the label that occurs most frequently among K leaves nearest to itself. In other words, the output of K-NN is a class membership which is decided by majority vote of K nearest neighbours.



Q

Figure 8: Examples of pairwise distance matrix using beam angle descriptor. The instances in the same class have been re-arranged and shown in color squares (Better view in color).

3.3 DTW Weight Learning

Dynamic Time Warping (DTW) is an algorithm for measuring the edit distance between two sequences even when the sequences are of different lengths. The two open contours of leaves are usually not the same length, therefore DTW matching is ideal for our purpose. DTW works by warping the time axis using dynamic programming until an optimal match between the two sequences is found. Consider two sequences $l = \{x_1, x_2, ...x_m\}$ and $l' = \{y_1, y_2, ...y_n\}$, where x_i is the sampling point of one contour and y_j is the sampling point of the second. For any two points from l and l', we extract the features and define the cost c_{ij} between these two features. Since our two shape descriptors are both histograms, Chi-squared distance is a suitable measure for the difference between x_i and y_j. The final cost of DTW matching of two contours is defined as:

$$S_{DTW(l,l')} = \frac{1}{|\pi_{ll'}|} \sum_{(i,j)\in\pi_{ll'}|i\in l, j\in l'} c_{ij} \tag{2}$$

where π_{ij} defines the optimal alignment of the two sequences and $|\pi_{ij}|$ is the length of the aligned sequences.

In the original DTW matching, all sampling points are assumed to be equally important. However this may not be the case for leaf shape character scoring. For example, the leaf apex curvature can be approximated by several points around the leaf apex peak. Therefore the points in this area are more important than others. If we can estimate the underlying importance of sampling points, the accumulated DTW matching cost will become more accurate. So we proposed a weight learning process to assign a weight to each sampling point.

For each open contour with n sampling points, we aim at learning a weight vector w of length n. Based on the original pairwise distance matrix D, for each leaf we can find another leaf have the same character state that has the largest DTW matching cost. We call it the Farthest Hit (FH). Similarly, the leaf with a different state that has the smallest DTW matching cost is called the Nearest Miss (NM). Our objective function is to maximize the margin between FH and NM w.r.t to the weight vector w:

$$\begin{aligned} \max_{\{w_i\}} \quad & \sum_{i \in l} w_i \left[\sum_{l' \in NM} \underset{j \in l'}{mean}(\{c_{ij}\}) - \sum_{l'' \in FH} \underset{k \in l''}{mean}(\{c_{ik}\}) \right]_{\dagger} \\ s.t. \quad & ||w||_2 = 1 \end{aligned} \tag{3}$$

where $[a]_{\dagger} = max(0, a)$.

Each sampling point on current contour l may have multiple DTW matching points on contour l' of NM and multiple matching points on contour l'' of FH as well. The 2-norm of final weight vector w is constrained to be equal to 1.

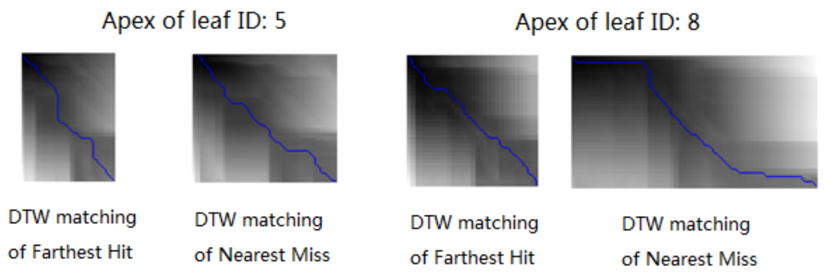


Figure 9: DTW matching of two leaves with their FH& NM respectively (Better view in color).

After the weight vector w of one contour l is learned, the cost of DTW matching between l and new coming contour l' is calculated as below:

$$S_{DTW(l,l')} = \sum_{i \in l} w_i \ \underset{j \in l'}{mean}(\{c_{ij}\}) \tag{4}$$

4 Results

In this section, first we describe the dataset used for our experiments and introduce metrics for our results evaluation. Then we show the qualitative results of weight learning. Finally, we will compare the quantitative results of the approaches.

4.1 Dataset

Our dataset is part of the BisQue database developed by biologists who are working on leaf characters research and it is available at: `http://bovary.iplantcollaborative.org/client_service/` by request. Specifically, it contains over 400 real world monocot leaves which were originally selected by biologists in an attempt to cover the wide diversity of monocot leaves. The monocot leaves diversity includes various leaf sizes, various leaf positions and directions, various lighting condition, leaf colors and background etc.

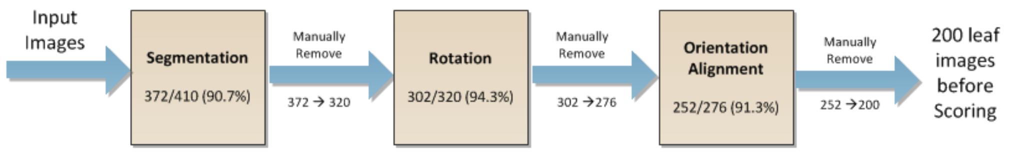


Figure 10: Pipeline of segmentation, rotation, and orientation alignment.

Our system eliminates the unqualified leaves in each preprocessing stage as shown in Figure 10.The final 200 selected leaves are highly representative for scoring shape characters because they are intact monocot leaves that including whole apex and base area, therefore no information is lost; They are all straight flat leaves rather than spiral leaves, so it is much easier to extract features on the both ends of them; They are not damaged by the immerse liquid, therefore the segmentation results are more reliable. For segmentation stage, we manually annotated 40 images for training. These training images are annotated by assigning a false color to each pixel based on the following rules: Leave pixels are assigned green, specimen tag pixels are red, and other background pixels are blue. The RGB value triplets of these regions' definition are (0,255,0), (255,0,0), (0,0,255) respectively. For the orientation alignment stage, we manually labelled all of the cropped leaves after segmentation and rotation as follows: A leaf in the canonical orientation is labelled 1 and a leaf in the opposite orientation is labelled 0. For the shape character scoring stage, we manually labelled six characters according to rules given by our biology collaborators. In this thesis, we have converted the character states into integer values.

4.2 Evaluation

We used several methods to evaluate our output. One is a confusion matrix, which is a table visualization of the performance of an algorithm. In our result, each row of a confusion matrix represents the instances in ground truth class and each column represents the instances in a predicted class. The confusion matrix allows us to visualize how the algorithm is mispredicting. The perfect confusion matrix should have only the diagonal elements filled, which means that no misprediction has been made by the algorithm.

We also measure accuracy. The overall accuracy is defined as the percentage of number of correctly predicted instances divided by total number of instances. Given a confusion matrix K, overall accuracy is calculated as $trace(K)$ divided by $sum(K)$. Our dataset is unbalanced, which means different states have widely varying number of instances. Therefore we also use compute class-wise accuracy as an evaluation metric. Class-wise accuracy is the accuracy on each class separately. The class-wise accuracy weighted by the proportion of images in each class is equivalent to the overall accuracy.

4.3 Qualitative Results

Our system contains modules that serve the purpose of pre-processing the data before the final scoring stage. The results of these intermediate modules are visualized here to help us gain a better understanding the intuition behind the theory. Figure 11 below are the learned DTW weight vectors for two leaves. The first row shows two sets of leaf masks with and without contour smoothing. The second row shows the extracted apex and base patches of the non-smoothed mask; The third row shows the learned weight vector for the apex and base patches. Red and blue lines represent the weight learned with and without contour smoothing. The weight vector of the first leaf puts larger weights in the leaf peak area. This is reasonable because usually the apex angle can be decided by the points in that area. After contour smoothing, the learned weights are more concentrated in the middle of the curve. The second leaf is similar to the first leaf except that it has a more fuzzy boundary. Therefore before contour smoothing, the weight vector learned for the base angle puts larger weights on other points rather than on the base peak area. But after contour smoothing, the learned weight vector has larger weights concentrated in the center of the curve, which is better.

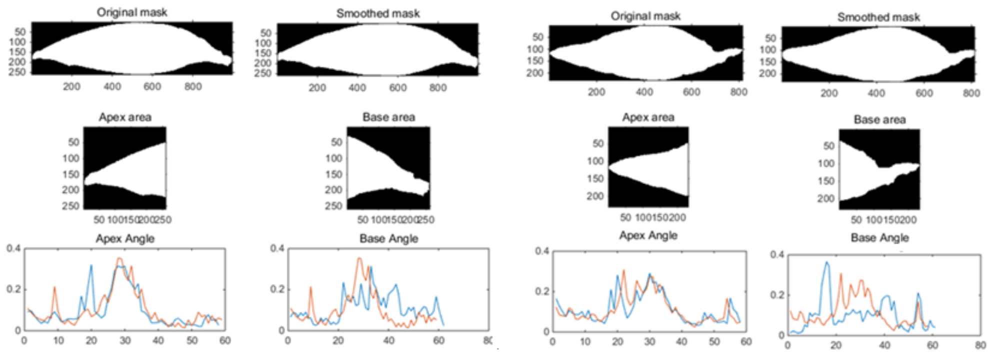


Figure 11: Visualization of learned weight (with and without contour smoothing) on two leaves of shape characters Apex angle and Base angle

4.4 Quantitative Results

4.4.1 Results of Framework I

In our final stage, leaf shape character scoring, the dataset we chose has 200 monocot images that under different background and have six leaf shape characters. We compute HOG features and perform SVM classification. We employed five-fold cross-validation to assess the performance of the method. Within this, we employed 10-fold cross-validation to choose the optimal parameters for the SVM. The summed confusion matrices of the five folds are presented in table 1. We see that all characters have three states except the Leaf base angle, which has four states. Note also that the number of instances in each state varies significantly, so the dataset is unbalanced for every character. In table 1(c), we see that 'State 1' is by far the most common in the dataset, and 'State 3' is easily confused with 'State 1'. In table 1(e), we see that 'State 2' dominates the dataset, and 'State 3' is confused with other states for most of the instances.

4.4.2 Results of framework II

We applied out second framework to the same dataset. The shape descriptor we used in Beam angle, and the results are scored by the K-NN classifier. Again, we employed 5-fold cross-validation to compute classification accuracy. The accumulated results are shown in Figure 12. We compare the results under configuration: **a).** beam angle descriptors extracted directly from mask contours and pairwise distance matrix constructed using DTW algorithm without weight learning; **b).** pairwise DTW with weight learning; **c).** contour smoothing applied to the mask before Beam angle feature extraction; **d).** both contour smoothing and weight learning. The results are given in Table 2

In Figure 12, there are 16 confusion matrices: The rows represent the configurations **a), b), c), d)** and the columns represent the different shape characters. To compare the results visually, we display the 16 confusion matrices as color heatmaps. From the results, we observe that there are only subtle differences between the tables. However we observe the confusion matrices with weight learning and contour smoothing have a majority instances on their diagonals, which means this configuration has decreased the number of states being mis-predicted.

5 System Implementation

Our team is currently building a system called AvatolCV, which will host our segmentation, orientation alignment, and scoring algorithms. It is written in a wizard style to guide the user through a series of steps for selecting either a remote or local dataset, and for choosing which algorithms to run. AvatolCV has subsystems for user interface, question sequencing, data source selection, and algorithm loading and execution, and results viewing. The data sources currently supported are BisQue (`http://bioimage.ucsb.edu/bisque`) , Morphobank (`www.morphobank.org`) and a properly formatted local file set. AvatolCV's algorithm subsystem can detect and use algorithms written in C, C++ and Matlab, with minor integration adjustments. Once the scoring algorithm com-

	State 1	*State 2*	*State 3*
State 1	**190(100%)**	0	0
State 2	2(33%)	**4(67%)**	0
State 3	0	1(25%)	**3(75%)**

(a) Length-to-width ratio (overall accuracy: 98.5 ± 1.7%; baseline accuracy: 95%).

	State 1	*State 2*	*State 3*
State 1	**139(90%)**	11(7%)	4(3%)
State 2	7(17%)	**33(81%)**	1(2%)
State 3	2(40%)	0	**3(60%)**

(b) Widest position (overall accuracy: 87.5 ± 4.6%; baseline accuracy:69.5%).

	State 1	*State 2*	*State 3*
State 1	**126(93%)**	3(2%)	6(5%)
State 2	12(29%)	**28(67%)**	2(4%)
State 3	9(39%)	1(4%)	**13(57%)**

(c) Leaf apex curvature (overall accuracy: 83.5 ± 5.1 %; baseline accuracy: 63%)

	State 1	*State 2*	*State 3*
State 1	**116(92%)**	10(8%)	0
State 2	12(26%)	**31(67%)**	3(7%)
State 3	3(11%)	6(21%)	**19(68%)**

(d) Leaf apex angle (overall accuracy: 84.1 ± 5.1%;baseline accuracy: 58%)

	State 1	*State 2*	*State 3*
State 1	**31(78%)**	9(22%)	0
State 2	2(2%)	**135(96%)**	2(2%)
State 3	8(38%)	**9(43%)**	4(19%)

(e) Leaf base curvature (overall accuracy: 85.1 ± 4.9%; baseline accuracy: 67.5%)

	State 1	*State 2*	*State 3*	*State 4*
State 1	**32(80%)**	8(20%)	0	0
State 2	8(11%)	**47(67%)**	12(17%)	3(5%)
State 3	2(4%)	12(26%)	**27(59%)**	5(11%)
State 4	0	1(2%)	5(12%)	**38(86%)**

(f) Leaf base angle (overall accuracy: 72.0 ± 6.2%; baseline accuracy: 23.5%)

Table 1: Scoring results of six shape characters in confusion matrix. In each confusion matrix, the class-wise accuracy of each state is the percentage numbers in the parenthesis of the diagonal elements. The overall accuracy (with 95% confidence level) is shown in sub-caption. Baseline is the the most frequent class

	Apex curvature	*Apex angle*	*Base curvature*	*Base angle*
Original framework II	80.4%	85.4%	86.4%	76.4%
Weight learning	86.4%	**88.9%**	87.9%	**80.4%**
Contour smoothing	81.4%	81.4%	86.1%	74.3%
Both	**88.9%**	86.0%	**89.4%**	74.4%
Framework I	83.5%	84.1%	85.1%	72.0%

Table 2: Overall accuracy of different configurations on four shape characters

pletes, the results viewing subsystem allows inspection of results. Scores accepted by the users can be added to a local dataset or uploaded into a remote one. The development of AvatolCV is ongoing at the time of writing this thesis.

AvatolCV's framework is implemented in Java. The algorithm subsystem uses DARWIN [11] for segmentation, VLFeat [19] for HOG feature extraction and LIBSVM [4] for SVM classification. The remaining operations are all implemented in MATLAB. We run all our experiments on an machine with an Intel(R) Core(TM) i7-4700MQ CPU @ 2.4GHz and 8GB RAM. All images have been resized so that the longer side of the image has 1000 pixels. The dataset contains 200 images. The running time of segmentation training on 40 images is 45 minutes, and the running time of segmentation testing on each image is around 25 seconds. The running time of orientation alignment is around 15 minutes for the whole dataset. The most time-consuming steps are HOG feature extraction and PCA. The running time of scoring using the HOG plus SVM method is 22 minutes for each character on the dataset. The running time of scoring using the contour plus K-NN method is 50 minutes for each character because beam angle extraction, DTW matching, and pairwise distance matrix construction are all very compute-intensive.

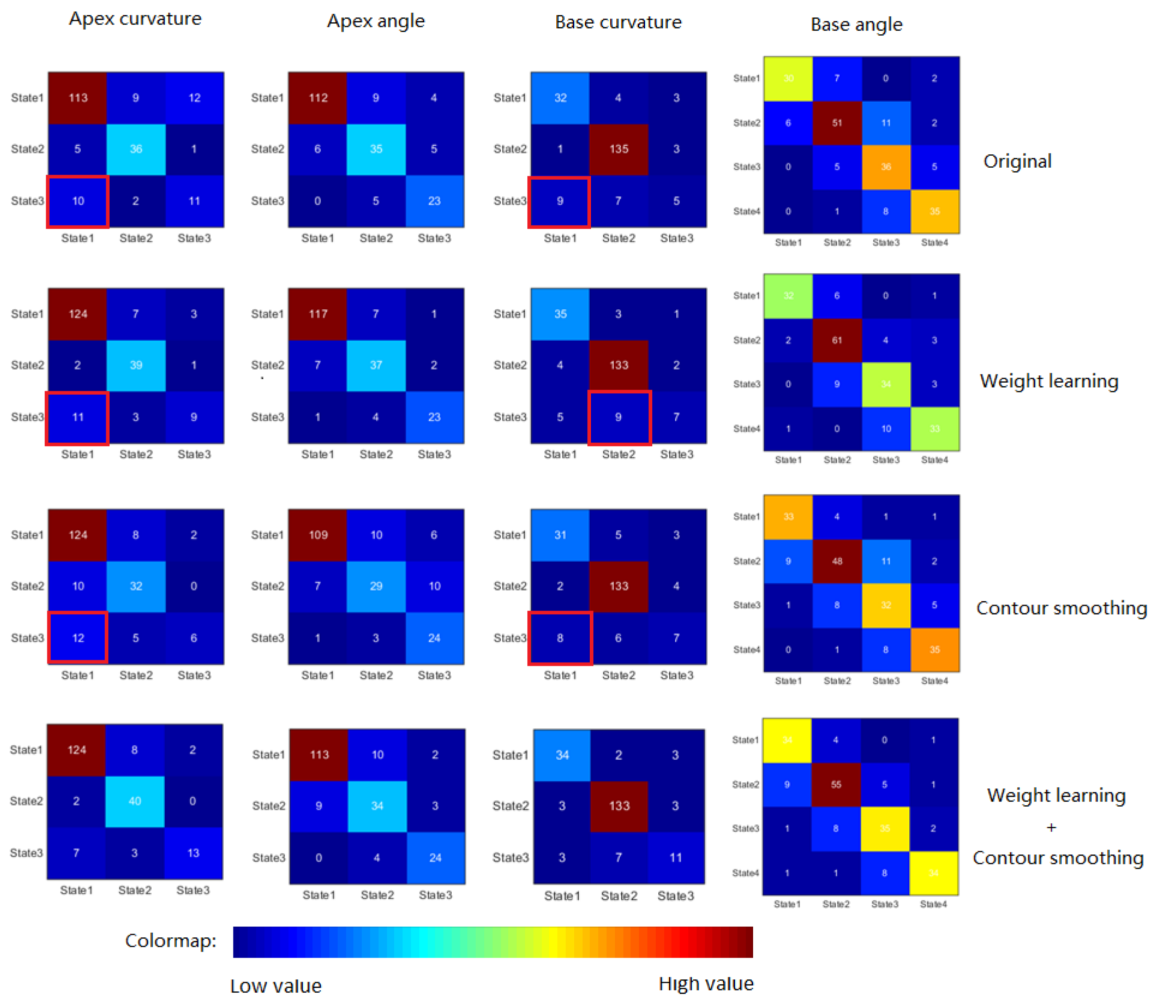


Figure 12: Confusion matrices of four shape characters over four different methods of pairwise distance computation. In heatmaps, Red means high number of instances and Blue means low number of instances. The Red rectangles are marked out predictions showing severe mis-classifications (Better viewed in color).

6 Conclusions and Future Work

We have described the system we designed to score shape characters of monocot leaves. The system includes two stages: the first stage crops the leaf from image and puts it to a canonical orientation. The second stage applies two frameworks for shape character scoring. One is employs SVM classifier on HOG features and the other matches the leaf shape contour to exemplar shape contours using DTW to build the pairwise distance matrix. The K-NN is applied to score the characters. Both frameworks have been evaluated on the real world leaves dataset using confusion matrix and accuracy metrics.

The evaluation shows that the second framework has better performance in overall accuracy, especially for the configurations where the DTW weight learning step is employed. Contour smoothing does not necessarily improve the results. In some cases, it appears to destroy the shape information on Apex/Base peak, which is critical for shape character scoring. However as the first framework is a universal method that can applied on many other categories of characters such as vein characters and secondary vein characters, we may continue use first framework in further study of monocot leaves.

Overall our approach enables the biologists to score the monocot leaves automatically and efficiently with confidence in the accuracy of the system. At this point, the petiole of leaves may or may not be present after segmentation. However the first framework needs HOG features from Apex/Base patches without petiole and second method also only needs the contours of Apex/Base peak area without petiole. For future work, we will try to develop an efficient algorithm to remove the petiole before character scoring.

References

[1] Assembling, visualizing and analyzing the tree of life (avatol), `http://avatol.org/ngp/`.

[2] Identify tree species by leaves: Treetrees, `http://www.tree-app.com/`.

[3] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2002.

[4] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (ACMTIST)*, 2011.

[5] Dorin Comaniciu, Peter Meer, and Senior Member. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2002.

[6] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[7] H. Tang David Corney, Jonathan Clark and Paul Wilkin. Automatic extraction of leaf characters from herbarium specimens taxon. *International Journal of Plant Taxonomy, Phylogeny and Evolution*, 2012.

[8] Li Fei-fei. A bayesian hierarchical model for learning natural scene categories. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[9] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2010.

[10] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision (IJCV)*, 59:2004, 2004.

[11] Stephen Gould. DARWIN: A framework for machine learning and computer vision research and development. *Journal of Machine Learning Research (JMLR)*, Dec 2012.

[12] Xuming He, Richard S. Zemel, and Miguel Á. Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[13] Hossein Mobahi, Shankar Rao, Allen Yang, Shankar Sastry, and Yi Ma. Segmentation of natural images by texture and boundary compression. *International Journal of Computer Vision (IJCV)*, 2011.

[14] Arijit Biswas David W. Jacobs W. John Kress Ida Lopez Neeraj Kumar, Peter N. Belhumeur and Joo V. B. Soares. Leafsnap: A computer vision system for automatic plant species identification. In *The 12th European Conference on Computer Vision (ECCV)*, October 2012.

[15] Caroline Pantofaru and Martial Hebert. A comparison of image segmentation algorithms. Technical report, 2005.

[16] Massimiliano Pavan and Marcello Pelillo. Dominant sets and pairwise clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2007.

[17] Nadia Payet and Sinisa Todorovic. Matching hierarchies of deformable shapes. In *7th International Workshop of GbRPR*, 2009.

[18] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 1997.

[19] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms, 2008.

[20] Pengcheng Wu and Thomas Dietterich. Improving svm accuracy by training on auxiliary data sources. *International Conference on Machine Learning (ICML)*, 2004.