Plant Classification with Relative Angles

Christopher B. Smith

Abstract – A description and implement an object classification system based on using relative angles on scan and scan-like images a set of plant leaves.

Keywords – ImageCLEF, Plant, Leaves, Images, Identification, Classification, Object Recognition

1 Introduction

Classification of objects is a difficult problem, so in an attempt to move towards a solution to this problem, work on a subset of classification of plant leaves. This will be done working soley on the shape of the plant, and not by color, texture, or location. The approach taken is to measure the angles along the edge of the leaf in a manner relative to a single edge point. This method was chosen because it allows independence from the image rotation and scale.

2 Literature / Related Work

[1] ImageCLEF2012 Plant Data Set gave insight on the best results in the field and insight into each implementation, such as human assisted or not.

3 Methodology

3.1 Context

The method was chosen out of interest on what could be possible, as well as a possible classification method that avoids the direct use of color, rotation, location, and scale. A method that only relies on the shape of the object to be tested. Not without difficulty of course.

3.2 Dataset

ImageCLEF Plant Identification Package

This image set was chosen for the topic and the yearly results comparing methodologies. The set of images contains three distinct types of images.

- Scan images where the leaf is pressed flat on a white background. These are simple to work with.
- Scan-like or Psuedoscan images is a leaf that is placed on an inconsistant background of single color. These add difficulty with the new ranges of color that need to be managed, as well as shadows caused by the leaf.
- Photograph images are photographs taken in the wild without any setup. All of the photograph type images have been removed due to the difficulty in segmentation being beyond my purposes.

Scan:

Scan-like:

3.3 Methodology (train.m)

1. Primary Segmentation
   - simpleSegment.m
   - This method was choosen after many failed attempts. It still needs work, but this is a good starting point.
   - Image segmentation is proving to be one of the most important steps in the entire process.
   - A method to remove shadow or blur in the image is still required for better segmentation results
   - Issues with discoloration due to shadows or varied background gradiants has been handled by implementation of Regions or subimages of an individual leaf.
2. Clean up the segmentation
   - Remove edge touching segments
   - A series of morphological operations including: Close, Fill
   - Erode image to separate individual leaves from any stem structure
3. Generate a number of Regions
   - breakImage.m
   - Identifies each possible leaf
   - Generates an image per identified leaf area from the source image using a mask made from dilating the eroded leaf area found
4. Cycle through Steps 5 - 10 for each identified Leaf
5. Canny edge detection
   - Cycles through each Region (individual leaf) identified
   - Thin to make the outline one pixel thick
   - Remove any Spurs for good measure
6. Crop image
   - autoCrop.m
   - Scales down image to the minimum size to contain the detected edge
   - Adds one pixel buffer along the outside edge for next part
7. Traverse Edge
   - traceNeighbor.m
   - This method is the core of the concept in my approach
   - Mapping which 8-neighbor is traveled to in a clock-wise manner
8. Filter Results
   - processChain.m
   - This method adapts the 8-neighbor mapping from the previous step into a series of changes in angle
   - For Example: Changing from an East neighbor to a South East neighbor would give a value of 1
9. Clean Results
   - reduceChain.m
   - This method removes any unchanging angles from the results of the filtering
10. Produce Training Data
    - bestMatch.m
    - This method finds any previous classification of the current type of leaf and attempts to line up the angles to have the smallest difference
      - If no previous classifications of the current class are found, it simply assigns the number set
    - When the smallest difference is locate, the results are averaged to form a representative result set
11. Run Test Data Set
    - test.m
    - The how-to is similar to the training system, with some notable differences
    - Once the data set is generated per

region, the set is aligned with each known classification to find the best match
    - Once all regions have been classified, the most common classification among the leaves is selected as the best guess and returned
    - All unclassifable images are assigned a token class of 0.
12. Generate a Confusion Matrix
    - OutputResults.m
    - Displays a figure displaying the results
    - Returns a matrix representation of the results

4 Results

In general the results gotten are not ideal in many aspects. There are many aspects that can be addressed.

4.1 Results with Scan & Scan-like images

*Table of results given in file: Results 1 – 88.mat (matchTable)

Accuracy over tested images: 4.167%
Tested: 88 Images
Correct: 3 Images
Total Classifications: 122

Certain images failed to generate proper leaf segmentation, and other images were chosen as the classification too often. While the result does show a better than simply guessing average, the results show that certain classifications seem to be easier to align with than others.

Time constraints limited completion. The results are given for the first 88 images of the test set out of the 2667 images after the photographs are removed. The images were being processed at a rate of nearly 4 images per hour.

5 Conclusions

There have been many misgivings about this project. The concept I think is quite possible and interesting, but the execution is dire. This system should not be used independently of other faster classifications which could narrow down possibilities before testing. This method is very much the last type of classification to use. While the results may not have been impressive, I can not regret the effort put forward to implement and learn from this project.

5.1 Future Improvements

Many aspects of this project have failed in their respective implementation. Some ideas are good, while others are negligent. Most of these improvements would take about a week to implement completely. The creation of super sets to narrow down the possible classifications reducing the computational time significantly would take at least two weeks in itself due to the need for heavy testing.

1. Creation of Super Sets
    - Create multiple Super Sets of common reduced images
    - Reduce each image to the sizes of 16x16, 32x32, and 64x64.
    - At these sizes, most of the differences will be removed and allow for finding common generalized shapes
    - Reduces the possible images a tested image could be, by only testing possible classes based on comparison to the multiple Super Sets.
    - +Significantly reduced cost of computation
2. Culling of extraneous regions
    - Remove impossibly small and large regions, such as artifacts remaining after erosion or leaf overlaps.
    - +Reduces incorrect data inclusion
    - +Reduces computational cost

3. Improvements in determining alignment cost of skipping a value
   - Adjust cost based on variable string length sizes
   - +Improve score on variable sized string
4. Rotating string after each alignment to find the most optimal alignment (best score)
   - +Improve accuracy significantly
   - +Allows ignoring leaf rotation
   - -Reduce computational speed
5. Improved Sequence Alignment Algorithm
   - There is a more memory efficient version of the algorithm used, which would also reduce the needed calculation cost for the regions not tested.
6. Multi-core support
   - Each image can be tested independently, but not trained as such if they share a classification.

References

[1] ImageCLEF Plant Data Set, http://imageclef.org/2012/plant

[2] tayyab's Image Segmentation Method, http://www.docstoc.com/docs/43883774/MATLAB-SOURCE-CODE-FOR-IMAGE-SEGMENTATION