# Optical Character Segmentation and Recognition from a Rochester Flag

Thomas Kollar and Jonathan Schmid
05/07/02

## Abstract

This project investigated segmentation and recognition of characters from a Rochester Flag as visual behaviors for landmark recognition and nametag reading. We successfully segmented and correctly classified the “ROCHESTER” characters from arbitrary images of the flag.

## Introduction

Optical Character Recognition (OCR) has as its objective to match a sequence of input images to some sequence of characters in a given alphabet. Some of the harder problems in OCR include recognizing damaged or fuzzy text, text in the presence of heavy noise, multi-font text, and unconstrained hand printed characters (Mori 4). This project deals with taking a 320x240 color image, segmenting the text from the image, and classifying each letter in an OCR fashion. We have built three applications that demonstrate this process. There is a program that takes as input an image of the Rochester flag, and outputs the letters of the word "Rochester" in separate images. Another program takes the segmented images as input and trains the classifier. It logs this training data to a text file. The third program verifies the output of the trained classifier on a new set of letter images.

Our hope is that the classifier and segmenter can eventually be intertwined to read nametags on a live image stream from a robotic agent. This project, however, deals only with a subset of the nametag reading problem. The goal is to take a picture of a University of Rochester banner and segment the letters of the word Rochester from it (see Figure 1).


Figure 1

## Segmentation Techniques

The purpose of our character segmentation utility is to produce rotation and scale invariant images of the characters in a specific word from the Rochester flag. The images are binary, and 20x20 pixels in size.

The character segmentation techniques used to read "R O C H E S T E R" from the U. R. flag rely on a number of structural invariants. A mobile robot will be able to make these assumptions about the structure of the flag during the AAAI contest because the rules allow our team to place it at the hors d'oeuvres refill station. The indoor environment will allow us to prevent motion of the flag and ensure that it remains flat. A-priori knowledge of the flag's two-dimensional structure allows us to design simple reactive behaviors that can segment the desired characters.

The flag contains many text regions at varying orientations and scales. We have chosen to read the large collinear letters near the bottom. These characters are written in solid yellow material against the dark blue background, and each occupies nearly 9 square inches. Our text segmentation utility relies on a signal from the camera that can

image these characters to an area of at least 20 x 20 pixels. This resolution constraint was required because the character classifier utility expects images of size 20 x 20.

## Scale Invariance

We investigated stretching segmented characters that were smaller than 20 pixels in either dimension, but the interpolated signal was intolerably noisy. We found this to be the case for any of the interpolation methods in the Intel Image Processing Library (nearest neighbor, linear, bi-linear, or cubic). If characters are imaged at a size smaller than 20 x 20, our system simply copies them into the upper left corner of a black 20 x 20 image. Large numbers of training examples could possibly allow for recognition even in violation of this resolution constraint.

Our long-term solution to this resolution constraint in nametag reading will be to use a higher resolution digitizer. Work in this project so far has used a *Hauppage* Win-TV USB digitizer. An example image of the flag digitized with the *Hauppage* is shown in figure 1. Future character recognition work will be done using a *Dazzle* Hollywood DV Bridge. This digitizer uses a FireWire connection to achieve 640x480 pixel images at NTSC frame rates.

## Character Segmentation

Segmentation of individual letters is achieved through a formalism that we call the Fuzzy Spike. We define a Spike to be a continuous one-dimensional region of non-zero histogram values from a one-dimensional histogram. A Fuzzy Spike is also a group of histogram values, but intermittent regions of zeros are allowed. When labeling the clusters of non-zero values, we set a limit on the number of continuous zeros allowed between 1s of a single spike. When segmenting letters, we sum down the columns of the image into a 1-D histogram. We also sum across the rows of the image into another 1-D histogram. We apply the fuzzy spike finder to these histograms to segment each letter. Figures 2 through 9 show the results of applying the fuzzy spike technique to segment letters.

Figures 2 through 9

In order to find the “R O C H E S T E R” region of text at the bottom of the flag, we assume that the large, round *Meliora* symbol at center of the flag is the largest vertical Fuzzy Spike. The vertical location of the text that we wish to read can be found by investigation of the next vertical Fuzzy Spike below this large round symbol. Using the vertical location, we can then obtain a horizontal Fuzzy Spike of the entire word. The red spike in Figure 10 corresponds to the *Meliora* symbol. The green spike below it is the desired Rochester label.

Figure 10

Rotation Invariance

The angle between the line of underscore for this word and the x-axis is then calculated. We take the arctangent of the slope (deltaY / deltaX). This angle is used to rotate the entire image about the center of the word (shown in Figure 11). After rotation, each character in the word is upright (Figure 12). This allows the character classification to achieve invariance to rotation. For this technique to be valid, the text being recognized must share a common line of underscore.

Figure 11

Figure 12

Perspective Invariance

We investigated doing a perspective warp of the image in order to make the classification invariant to camera location in the world. Figures 13 and 14 show the quadrangle and region of interest used for the warp. Figures 15 and 16 show a text image and the resulting warped text image. Achieving the correct quadrangle for the text proved to be quite difficult. Figure 16 demonstrates the problem of an incorrect quadrangle, and the resulting loss of data. We will rely on robust training data to achieve perspective invariance. This is not a problem if the camera images the poster from a fairly constrained set of locations.

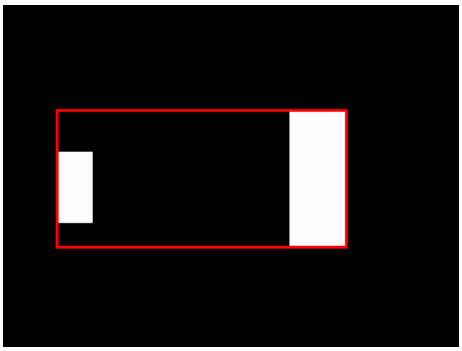

Figure 13

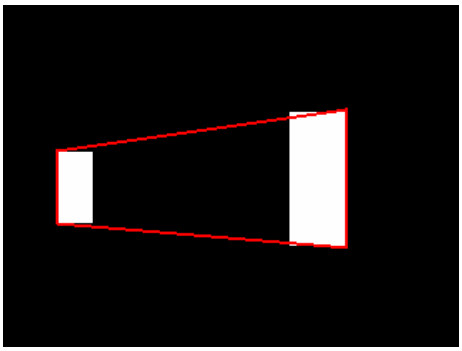

Figure 14

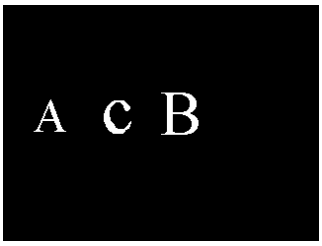


Figure 15

Figure 16

**Classifier:**

The purpose of the classifier is to take a 20x20 pixel binary image and classify it as a letter in the English alphabet. This particular classifier needs to be trained on each letter that will be recognized. This process requires 20x20 pixel binary images. It uses vector-clustering techniques and ideas taken from vector calculus and Mori (1-39). Mori mentions the use of line crossings as attributes in classification. This classifier substitutes histogram techniques for line crossings.

The classifier works by taking an image and extracting some number of attributes from that image. An array of these attributes is interpreted as a vector, in the mathematical sense. This vector has a dimensionality that varies with the number of attributes garnered from the image. We implemented classification using 40 spatial attributes, and 6 spatial moments.

To analyze the closeness of one vector to another, we calculate the distance between the two vectors. Distance is measured by $\|v1 - v2\|$, the norm of the difference of v1 and v2. This distance is useful for classifying a particular letter either as one of the training set, or as of an unknown type. In this case, there is a mean vector that describes each letter in the alphabet. For any new images that need to be classified, we find the

distance from the vector that describes this new image to each mean vector in the alphabet. The classifier chooses the smallest distance, and the associated letter, as its result. A threshold can be used with the vector distance to classify an example as outside of the training set. Thus, there are two parts to classification: training and retrieving.

Training:

To train this classifier, an arbitrary number of examples for each letter are taken to be the training set for the letter. Each example this letter's training set is given to a function named trainWhichCharacterCluster(). This function takes in an image and a letter. The function is run on all of the examples in the training set, and logs the vector obtained from the image into a cluster of vectors. This cluster of vectors describes all of the examples in the training set for some letter, and is written into a configuration text-file.

Each cluster of vectors is analyzed to find a mean vector for the particular letter. For example if we have vectors v1,v2… vN, then the mean vector would be: v1 + v2 + … vN / n.

Retrieving:

To retrieve the identity of a new image we take the distance from each mean vector to the input image vector (as described above). A distance is computed to each mean-vector of the various letters. The letter that yields the smallest distance is used to classify the image. If the smallest distance exceeds a threshold, then the image is declared to belong outside the training set.

Attributes:

Chart 1 shows the sum along each pixel-row of the 20x20 image. Chart 2 shows the sum along each pixel-column of the image. These form the first 40 attributes used in classification. Chart 3 shows the first through third spatial moments for the training set. These form the last 6 attributes used in classification.

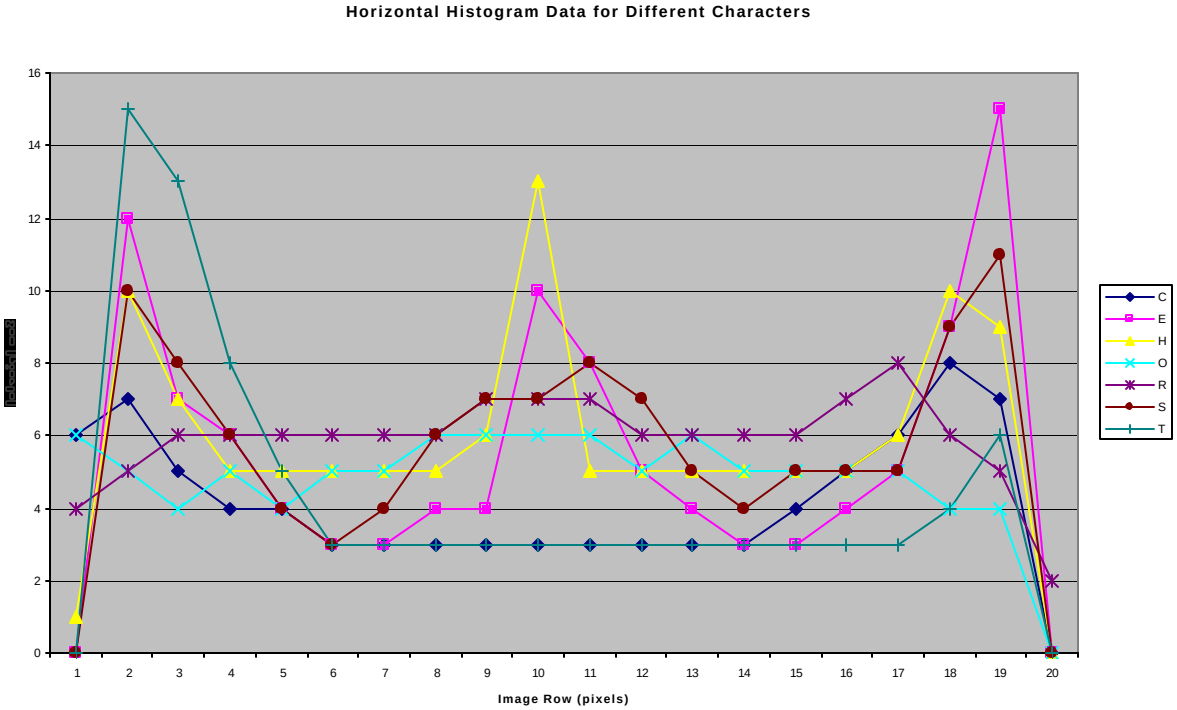


Chart 1

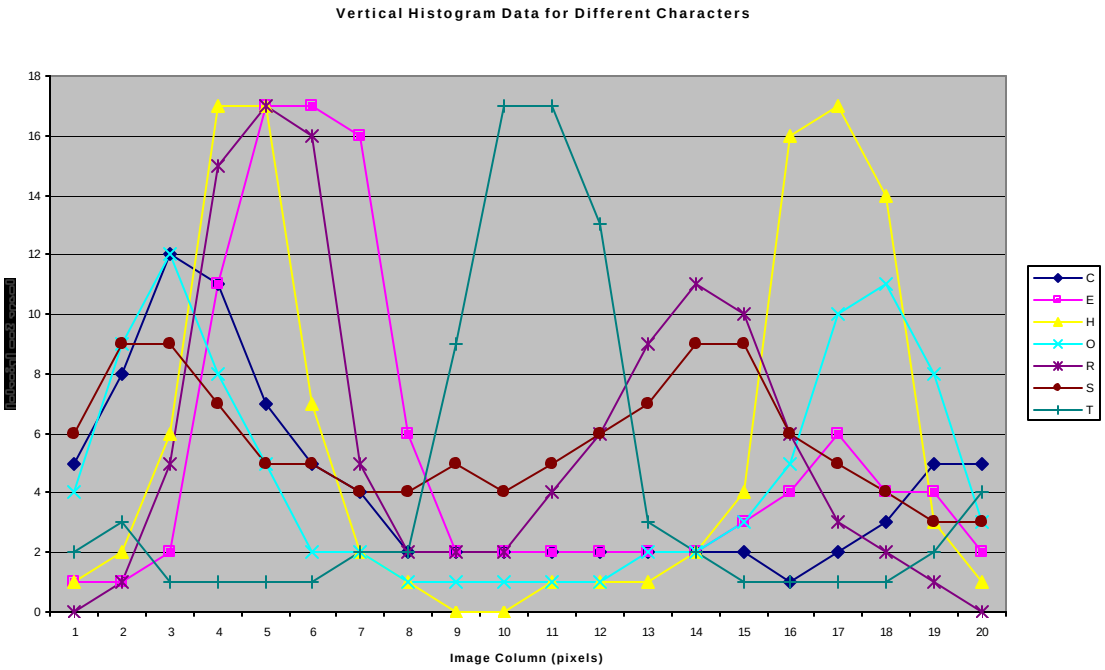


Chart 2

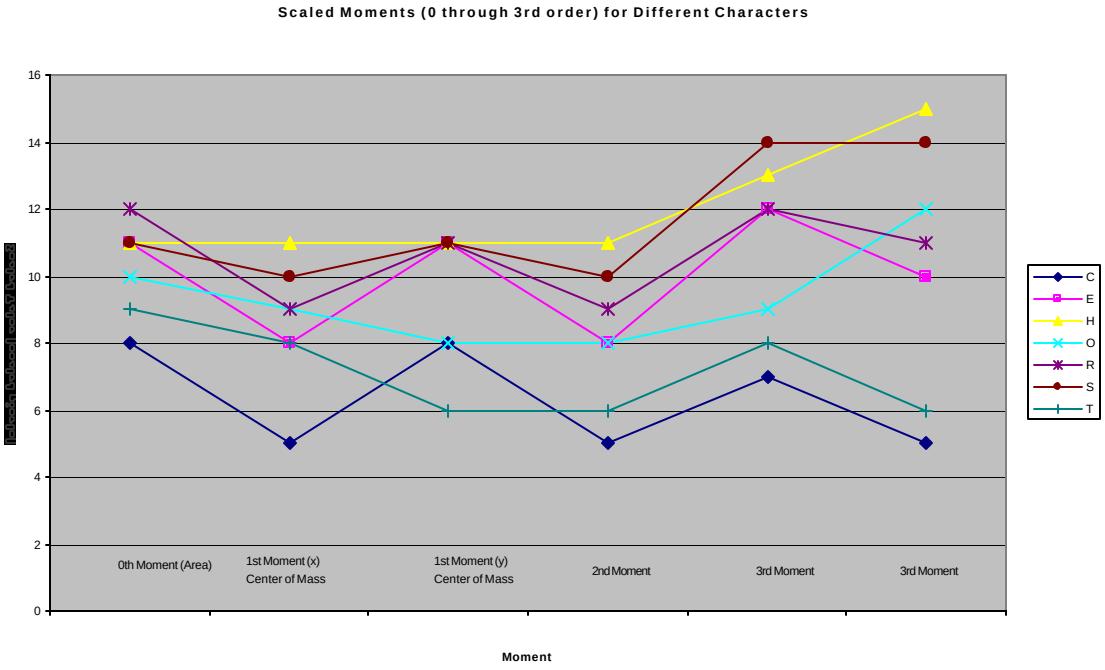


Chart 3

## Results

### Segmentation

The segmentation of the characters from the Rochester flag was successful in every example image we ran it on. Some examples of the segmentation can be seen in figures 2 through 9. Our results will improve even more when we can mount the flag in a glass frame. Since the flag was taped to the wall, some of the images appear incorrectly rotated. This did not appear to negatively impact classification.

### Classifier

The classifier worked on every example case that we tried. It correctly classified the characters in the set {R, O, C, H, E, S, T}. The ability to recognize the banner letters is more will lead to more robust landmark detection than color alone.

## Future Work

### Landmark Detection

Our intent is for the University of Rochester "Mabel" robot to recognize the poster as a landmark during the Robot Host competition. The utilities currently work from a bitmap stored on disk, and log the digitized images into a database of binary 20x20 images. All the samples of each letter are stored in their own directory. The recognizer utility currently fetches these images from the database when producing a classification configuration file for a given set of characters.

Reactive Behaviors

We will need to rework our training utilities into a set of visual behaviors so that they can be integrated into the MabelVision system. When we create these vision behaviors, we will most likely make use of online training. A human operator can supervise the character classification training using our off-board java parameter adjustment program. Supervised training through these behaviors will allow us to automate the classifier training process. Thus we will be able to quickly re-train the system in new environments if needed.

We will most likely make use of wheel encoding information to navigate the robot into the vicinity of the poster. Then, we will pan the camera until a large blue region with some yellow is detected. We will pan, tilt, and zoom the camera to an orientation similar to Figure 1. Finally, we will run the segmentation and classification utilities created in this project on a digitized frame from this viewpoint.

## References:

Mori, Shunji, Hirobumi, Nishida and Hiromitsu Yamada. Optical Character Recognition. New York: John Wiley & Sons, 1999.