CP467 - Group 10: Image Recognition Project

Brayan Boukhman, Erwin Guczuly, Duc Nguyen

Abstract - A collection of useful image functions is proposed in this paper. The primary functions are thinning and digit recognition. The thinning system thins symbols to their central line. This preserves the shape as that is the key to recognizing what the symbol is. Secondary utility systems have been created to help the primary functions achieve the goal of recognizing symbols. Such utility includes cropping and scaling.

1 Introduction

This collection of image functions all support the same goal of recognizing digits (0-9). The entire collection includes: apply 3x3 filters (etc. blur, sharpen, edge detection), convert grayscale images into black and white, finding connected regions, print the number of black pixels found in each region, scale each connected region into the same predetermined size, which all aid in the thinning of symbols and subsequently recognizing the symbol (exclusive to digits 0-9).

2 Primary Functions

2.1 Thinning

The program will run through every pixel and if the pixel matrix matches one of the 20 rules (see Fig. 1.), then it will trigger a flag that signals that some change has occurred. The change will be applied then the program will run through every pixel once more and repeats the process. The program will stop when there are no further changes.

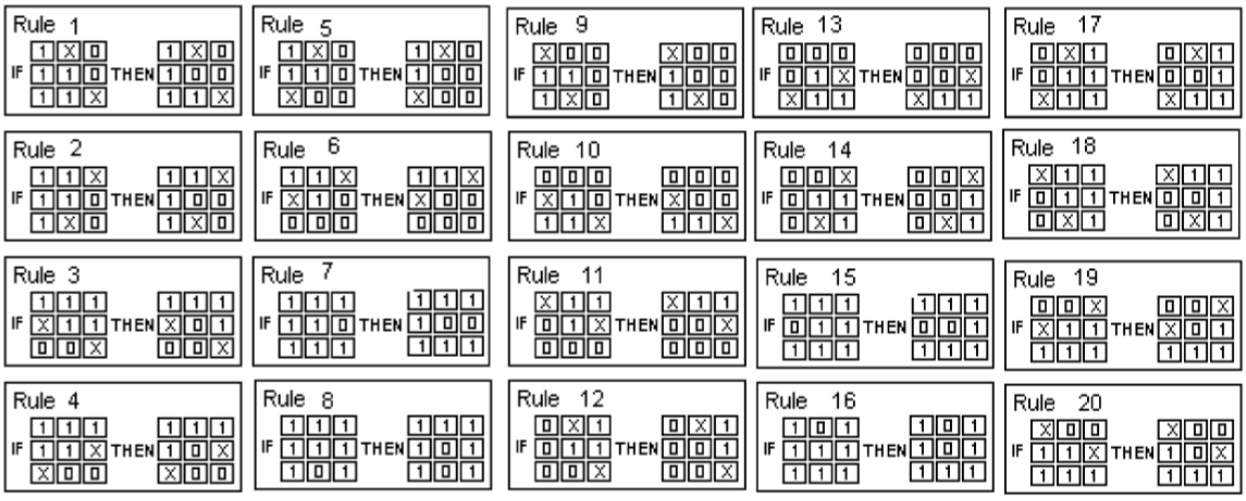

Fig. 1. The thinning rules [1]

2.2 Digit Recognition

Before we proceed with obtaining the feature vector, we will use the thinning function (2.1) to ensure that images with extremely thick black pixel regions are comparable to thinner black pixel regions. We will then apply the cropping function (3.1) to remove the excess white pixels so our resulting image will represent the respective digit. Afterwards, we use the scaling function (3.2) to resize the image to 100x100 (pixels) so the resulting feature vector is consistent with one another regardless of the original size. Lastly we get the feature vector for this image using the function (3.5)

Our feature vector based system attempts to recognize images using k-means that is stored within our text file database and is used to formulate the euclidean distance. Since we are using k-means we can determine the number written in the image by calculating the euclidean distance 10 times (0-9) and then by comparing the resulting values we can determine the number with a fair degree of accuracy.

3 Utility Functions

3.1 Crop

This utility function is mainly used in calculating the feature vector as images may have white spaces between the end of the image and the actual symbol itself. As this whitespace is unnecessary and has the potential to influence the calculation of the feature vector, it will simply be removed. In order to do this we find the coordinates for maxX, maxY, minX, minY and use these values to crop out the symbols in the image. See Fig. 2.

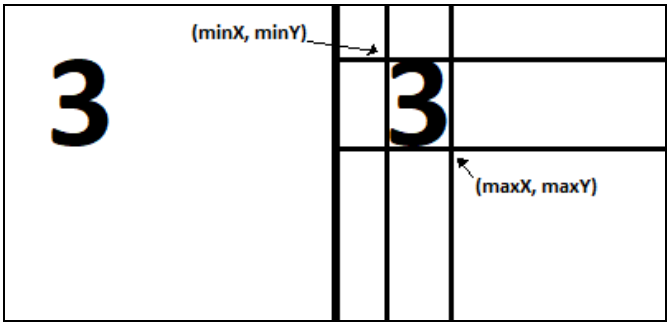

Fig. 2. Cropping

3.2 Scale

The scaling utility function takes an image and the desired height/width of the new image (pixels). First the new image is created then we populate that image by extracting values from the original image using getpixel (x/factorX, y/factorY) by multiplying them by a factor we can find the appropriate pixel, in the case that the resulting x, y values are decimals it would then find the 4 closest pixels and calculate the appropriate pixel color and the place that pixel into the new image, see Fig. 3..

Original image (200, 200)

Half the size (100,100)

Flattened (200,50)

Fig. 3. Scaling

3.3 Connected Regions

*Convert to black and white image prior to call

Connected regions are found by going through each pixel in the original image. Once a black pixel is found a recursive algorithm called; this creates a new image which adds every adjacent pixel to the new image while also keeping track of every pixel that has been interacted with in order to avoid unnecessary recursion calls of already found black pixels. Finally, return all the new images containing one connected region each, see Fig. 4.

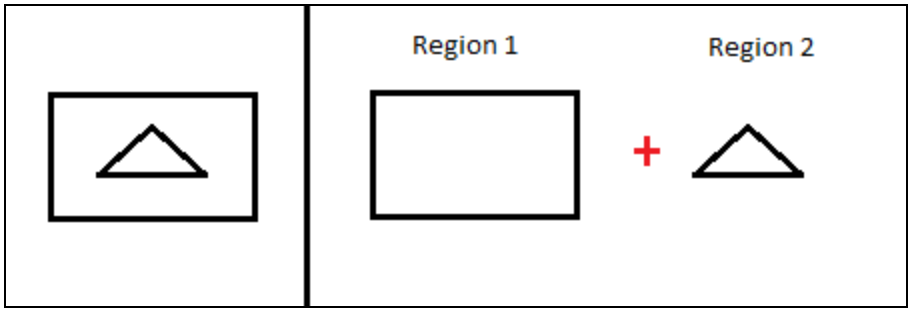

Fig. 4. Connected Regions

3.4 Connect regions

Essentially, the reverse of Connected Regions (3.4). It takes an array of images as input which then finds the x, y value of their first black pixel and then calls Crop (3.1). Then it stitches the resulting image into the new image. See Fig. 3.4; read right to left

3.5 Feature Vector

In order to find the feature vector for an image we need to divide the image into 9 equal sized regions. Afterwards we find the percentage of pixels in each region and by doing so we end up with the feature vector, see Fig. 5.

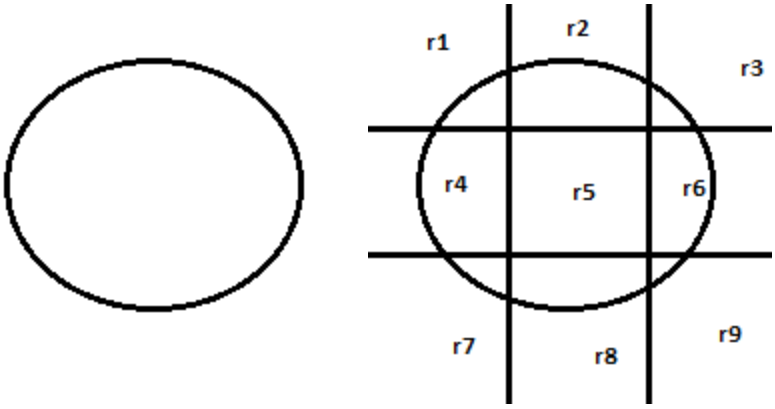

Fig. 5. Feature Vector

Resulting feature vector = [r1, r2, r3, r4, r5, r6, r7, r8, r9]

4 Conclusion

As shown, to be able to recognize symbols with a fair degree of accuracy there must be supplementary functions to achieve this goal. The degree of accuracy is greater due to the access cut from symbols such as whitespace, size, thickness of black regions, etc. Reducing the influence of the “access” will ensure that the calculations involved in recognition are true to black pixels and their shape. These functions allow us to construct everything necessary to recognize digits 0 through 9.

References

[1] Ahmed, Maher, and Rabab Ward. “A Rotation Invariant Rule-Based Thinning Algorithm for Character Recognition.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 12, Dec. 2002.