# Handwritten Character Recognition using Template Matching

Abhinav Pandey
Department of Electrical and Computer Engineering
University of Florida
Gainesville, Fl 32611
+1 352 214 4586
abhinavp@ufl.edu

Shashank Sawant
Department of Electrical and Computer Engineering
University of Florida
Gainesville, Fl 32611
+1 352 214 1688
sgsawant@gmail.com

Dr. Eric M. Schwartz
MAEB 321
University of Florida
Gainesville, Fl 32611
+1 352 392 6605
ems@mil.ufl.edu

## ABSTRACT

*This paper is aimed at the recognition of handwritten characters of any arbitrary size and shape. A novel template based matching scheme is employed which resulted in high matching efficiency while maintaining a low sample size for training. As the number of samples used for training is very low, the templates can be easily modified and thus this system can be simply modified to an adaptive system exploiting the uniqueness of one's handwriting. This property turns out to be extremely useful in making better handwritten character recognition system for cell phones, PDA and tablet computers where the input is given primarily by few users. The template can be modified by including the user given input to further increase the efficiency.*

## Keywords

Character Recognition, Template Matching.

## 1. INTRODUCTION

Handwritten character recognition is used in many devices including cell phones, PDAs, and tablet computers. Most of these systems employ character recognition using a fixed template. As the template cannot be altered, it limits the user to adapt to the system rather than system adapting to the user. In this paper a new template matching based approach is discussed which produces highly accurate recognition system while maintaining a very small sample data size. Small sample data size allows template to be changed dynamically to adapt to user's writing style. This paper is divided into 4 sections. Section 2 discusses the algorithms used for pre processing of image. Section 3 discusses the template matching algorithm and section 4 shows the results obtained..

## 2. PRE PROCESSING STEPS

Pre processing is an essential step in normalizing the input image [1]. Images are first converted into gray and then into binary images with black being 0 and white being 1. After that following operations are performed

### 2.1 Crop and Resize

This is the first step in the pre processing. The input binary image is shown in Fig.1a. The image is then first cropped by drawing a rectangle circumscribing the image. This step ensures that the relative position of image will not affect the matching. The cropped image is shown in Fig. 1b. The cropped image is then resized to a prefixed resolution, which in this case is 36x24. To maintain the aspect ratio of image white space are added symmetrically to left, right, up and down of the image if aspect ratio is not 3:2.

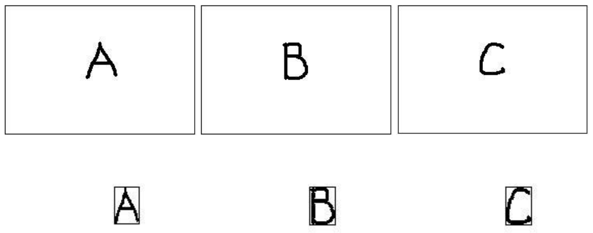

**Figure 1. A) Input Image, B) Cropped and Resized Image**

### 2.2 Slant Removal

This slant correction method uses the vertical projection histogram. The basic idea is that the histogram of a character that is written straight up will have larger and more distinct peaks[2], [3]. Therefore, the histogram of the character at different shear angles is taken. In this paper slant removal algorithm angles between −15 and 15 degrees is taken as shear angle. For each angle, the vertical histogram is computed and a measure function is applied that measures the height of the peaks. The angle with highest measure will be used as shear angle. The measure function looks for the peaks in the histogram and selects the highest 10 peaks, and returns the average height of those. If less than 10 peaks occur, it returns the average of those available.

### 2.3 Image Thinning

The third and last important step is thinning of the image. Thinning provides a single pixel thick continuous skeleton of the image. Thinning discards the redundant pixels of the image and at the same time it maintains the basic structure of the image. A brief fast parallel algorithm [4] is given as follows

In thinning a point P(i,j), its 8 neighbor are chosen as shown in Table 1. Any nth iteration is subdivided into two sub iterations and the values of neighboring point are used from end of (n-1)th iteration.

**Table 1. Neighboring Point Order**

| | | |
|---|---|---|
| P9 (i-1,j-1) | P2 (i-1,j) | P3 (i-1,j+1) |
| P8 (i,j-1) | **P (i,j)** | P4 (i,j+1) |
| P7 (i+1,j-1) | P6 (i+1,j) | P5 (i+1,j+1) |

Let A(P) be number of $0 \rightarrow 1$ transitions in the ordered pair $P2 \rightarrow P3 \rightarrow P4 \ldots \rightarrow P2$ and let B(P) be number of non zero neighbors of P(i,j). Now the point P(i,j) is deleted if the following conditions are satisfied

$1 < B(P) < 7$ $\quad$ $A(P) = 1$

$P_2P_4P_6 = 0$ $\quad$ $P_4P_6P_8 = 0$

In the second sub iteration P(i,j) is deleted if the following conditions are met

$1 < B(P) < 7$ $\quad$ $A(P) = 1$

$P_2P_4P_8 = 0$ $\quad$ $P_2P_6P_8 = 0$

The following process is repeated until no point is left for deletion. The thinned image is shown in Fig. 2

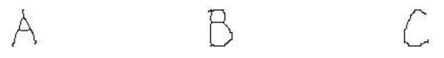

**Figure 2. Skeleton of Input Image**

## 3. TEMPLATE MATCHING

A template for each character is generated by using the skeleton images generated in pre processing stage. Template is made by taking all the black pixels of thinned image while excluding all the duplicate black pixels. This operation is performed by first inverting the image, i.e. flipping 0's with 1's and vice versa. To make a template of a character all the samples are taken and converted. Now an OR operation is performed on all the samples to generate the template of the image. A template image is shown in Fig. 3

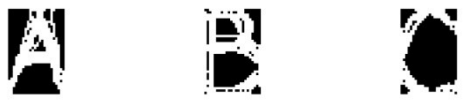

**Figure 3. Template Images**

Similar pre processing steps are also performed on the input image to be matched. The thinned input image is also inverted. The next step is to match the input stage to all the template images. The following two functions are calculated.

$$sim(char) = \sum_{i=1}^{36}\sum_{j=1}^{24} AND\{input(i,j), template[char](i,j)\}$$

$$dis(char) = \sum_{i=1}^{36}\sum_{j=1}^{24} XOR\{input(i,j), template[char](i,j)\}$$

sim(char) defines the similarity between input image with a given template image. It is computed by using a bit wise and operator over the entire image.

dis(char) defines the dissimilarity between input image with a given template image. It is computed by using a bit wise exclusive or operator over the entire image.

Both sim(char) and dis(char) are used to define the overall similarity of a character to a template character. The more the sim(char) is the more the similarity and the lesser the dis(char) is more is the similarity. Using these two properties the overall matching function of a character to a template is defined as follows

$$mem(char) = \frac{(sim(char))^{\frac{3}{2}}}{\sqrt{dis(char)}}$$

The index of similarity alone cannot generate reliable data. If alone sim(char) is used the results will be skewed towards dense characters as the and operation will generate high value. To overcome this index of dissimilarity is put in the denominator. This ensures that if the two samples match closely their similarity index is quite high. An array of matching value mem(char) is thus generated. Template with highest amount of matching is taken as a successful match.

## 4. RESULTS AND DISCUSSION

The main power of the algorithm lies in the very small number of training data. This allows it to update the template based upon the user input. Every incorrect recognition of the input character can be used to update the template. This results in very high accuracy in very small iterations

To demonstrate this property 6 sets say H1 to H6 or 26x6 = 156 samples of handwritten characters are taken as test data. The characters are generated using a touch screen phone and imported into the computer. The template is made using 10 sets say F1-F10 (26*10 = 260) of typed characters in various fonts. In the first iteration all 156 samples are matched using the template made by computer fonts and the numbers of incorrect recognitions are plotted.

In the next iteration one set of typed sample data is replaced by one set of hand written data i.e. the sample data for template generation is taken as F1-F9, H1. The test data which is being used is H2-H7. Again the numbers of incorrect matches are plotted.

A plot of this is shown in Fig. 4. It can be observed from the plot that in the beginning the error rate was roughly 27% which in the 6th iteration was reduced to 8%. In all the iterations sample data size is taken as 10 sets and test data size is taken as 6 sets of data.

So by this it is demonstrated that a better more efficient handwritten character recognition system can be implemented. With the ever growing market of touch screen phones and tablet notebooks this application can prove to be very useful. It can very easily, effectively and readily adapt itself to a particular hand writing style and can reduce the use of on screen keyboards.

## 5. ACKNOWLEDGMENTS

I would really like to express my thanks to my friend Rahul Agarwal and Abhishek Mishra for some of their valuable inputs and discussions.

## 6. REFERENCES

[1] M. Hanmandlu, K.R. Murali Mohan, Vivek Gupta, "Fuzzy Logic Based Handwritten Character Recognition," Image Processing, International Conference on, p. 714, 1997 International Conference on Image Processing (ICIP'97) - Volume 3, 1997.

[2] Bertolami, S. Uchida, M. Zimmermann, H. Bunke, "Non-Uniform Slant Correction for Handwritten Text Line Recognition," Document Analysis and Recognition, International Conference on, pp. 18-22, Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 1, 2007.

[3] Frank de Zeeuw, "Slant Correction using Histograms".

[4] Zhang, T. Y. and Suen, C. Y. 1984. A fast parallel algorithm for thinning digital patterns. Commun. ACM 27, 3 (Mar. 1984).

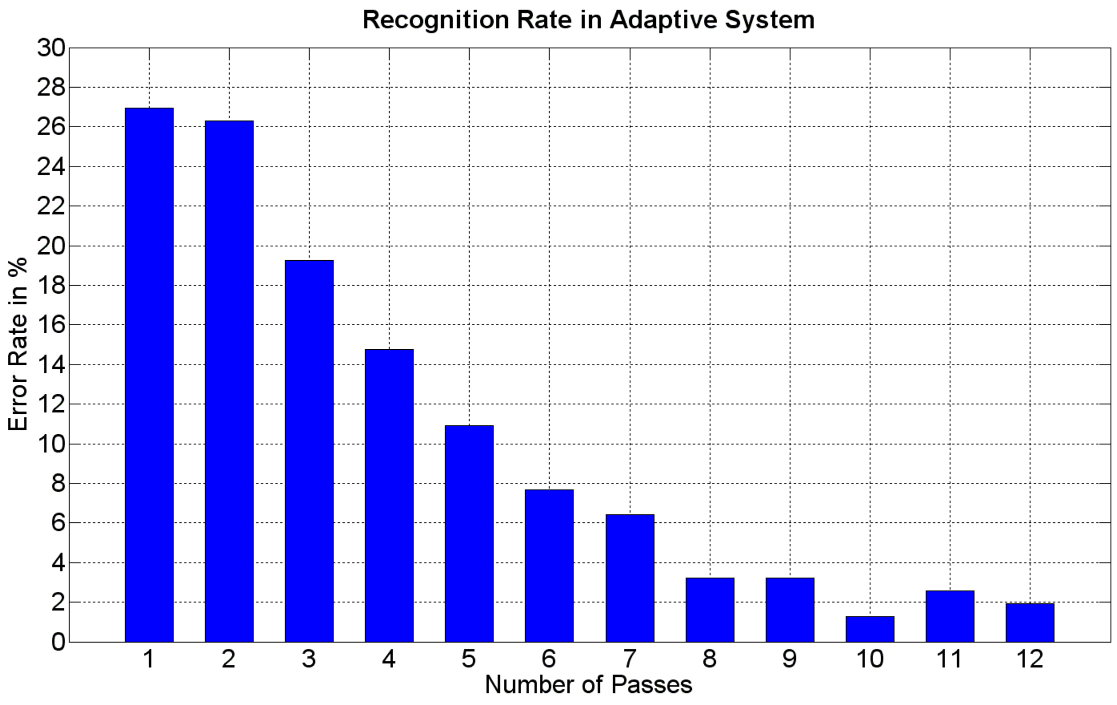

**Figure 4. Recognition Error Rate in Adaptive Mode**