Design and Implementation of Optical Character Recognition System to Recognize Gujarati Script using Template Matching

Prof S K Shah, *Fellow*
A Sharma, *Non-member*

Enormous amount of information is available in the world in the form of printed text. Operations such as searching, transporting and processing on required information in a printed form are difficult, time consuming and costly. These operations can be carried out efficiently and at low cost with the advanced technologies in the field of computer .However, it is necessary that information has to be in electronic form. optical character recognition (OCR), technology converts the scanned documents into editable text. Commercial OCR's for english script are already available. Paper describes, design and implementation using template matching prototype system to recognize Gujarati script. It recognizes each word in the input document image and outputs UNICODE text equivalent to it. The overall system was tested on various images from various sources.

Keywords : Segmentation; Pre-processing; Fringe-distance; Post-processing; Template matching; Vyanjans; Maatras; Hraswakshar

INTRODUCTION

An OCR system converts a document image into text format for easy editing, storage, transmission, searching, indexing and integrating into other applications. A typical OCR, Figure 1 contains four phases – preprocessing, segmentation, recognition and post processing. The preprocessing phase includes binarization of the input document image to separate the print and background objects, noise removal, skew correction, extraction of layout information etc. In the segmentation stage the individual lines words and characters are separated in stages or at a time. In the recognition phase each character in the document October 5, 2004 image is recognized. Template-matching, have been used wherein, each character in the input image as seen by OCR is compared against a set of templates and the code of the template that best matches is output. The post-processing phase includes conversion of the output into any standard text-encoding scheme, restoring the layout, detection and correction of errors made in the recognition phase. OCR can be categorized into task specific readers and general-purpose page readers.

Information is not restricted to a language. It is available in various languages. The scripts of different languages have different characteristics, hence new methods have to be designed which make use of unique characteristics of local scripts to recognize them easily. Technologies used for different non roman scripts like Chinese, Japanese and Bangla are described in Krishna[1].

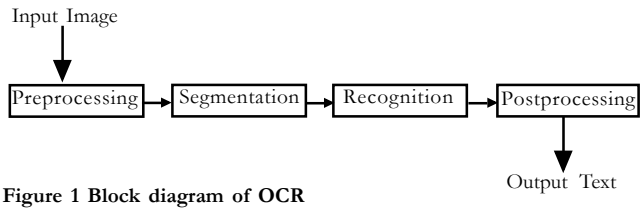

Figure 1 Block diagram of OCR

Prof S K Shah and A Sharma are with Electrical Engineering Department, M S University of Baroda, Kalabhavan, Vododara, Gujarat.

This paper (modified) was received on Octerber 18, 2004. Written discussion on this paper will be received until March 31, 2006.

The research work on various Indian language OCR's is already reported[2-15]. Antani[16] describe the classification of a subset of printed or digitized Gujrati characters, it has low recognition rate of 67 %.

CHARACTERISTICS OF GUJARATI LANGUAGE

Script

Gujarati is a phonetic language in western India. Gujarati script is written from left to right, with each character representing a syllable. Gujarati script has 12 vowels, which are called *Swar* and 34 consonants, which are called *Vyanjan*. These are shown in Figure 2 and Figure 3 respectively. Gujarati consists of a special symbol called

Figure 2 Vowels of Gujarati script

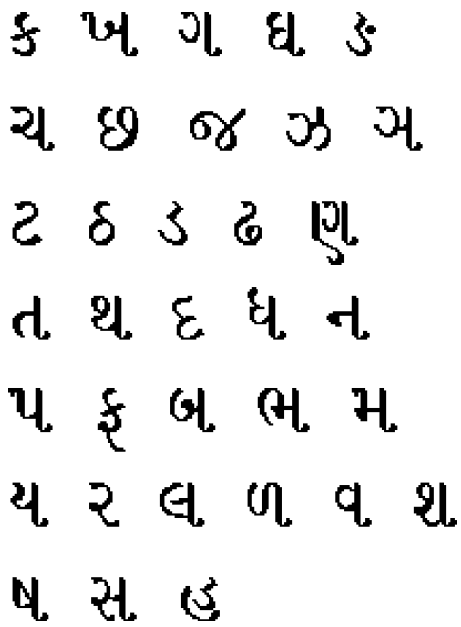

Figure 3 Consonants of Gujarati scripts

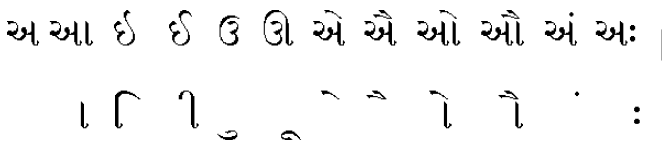

Figure 4 Special symbols of Gujrati scripts

Figure 5 Symbols for consonants without the vowels sounds

Maatra, corresponding to each vowel, which are attached to consonants to modify their sound. Maatras corresponding to each vowel is shown in Figure 4. First, Vowel does not have corresponding maatra but is basic sound for the consonants. Maatras are placed at the top, at bottom right or at bottom part of the consonant. They can be attached at different positions for different consonants. They can occur in different shapes depending on the consonant to which it is attached. In Gujarati each consonant actually is a combination of its pure form, called *hraswakshar* and the vowel sound phonetically. Visually also each consonant is a combination of its corresponding *hraswakshar* and the vowel *maatra* corresponding to vowel, *ie*, (excluding some exceptions). Each *hraswakshar* is obtained by placing ◥ below the consonant. When we want to use consonants without the vowel sound we have to use hraswaksharas Figure 5.

A character is said to be simple if it is a consonant alone or with a *maatra* (Figure 2and3). A charact er is said to be *conjunct* if it is a half consonant along with other consonant shown in Figure 4). It can be seen that shape of some of the consonants is changed while in case of some it is retained.

All the *vyanjans*, *maatras* and *hraswakshar* as together roughly provide basic orthographic units, which are referred as glyphs that are combined together in different ways to represent all the frequently used syllables.

Recognition Technique

Since no special features exist that classify the characters, the method used in Antoni and Agnihotri[16] can only be sufficient on a limited set of characters. Template matching[17, 18] was used in our recognition algorithm. Including the conjuncts along with the individual consonants the number of individual glyphs which can be recognized reaches to about 4500.

A character is split into connected components and each component is then cut so as to remove the lower and upper modifiers from the glyph. They are matched against a database. These connected and cut components are called as OCR glyphs. Their number, which is around 250, is considerably less than all possible characters. A trade off is reached by taking into account the amount of computation undertaken in recognition process of OCR glyphs. To recognize a character in Figure 6(a), we recognize the glyphs in Figure 6 (b) is recognised.

Figure 6 (a) A Character in Gujarati script

Figure 6 (b) OCR glyphs

Template matching is followed for the recognition. To compute distance or dissimilarity between two templates, they should be of same size. So, all the glyph images are normalized to 32 × 32 size. The image of the input glyph is also scaled to 32 × 32 size before comparison. The method used to measure the similarity or distance between is crucial. The challenge in template matching is in making the matching process fast and robust against distortions.

Fringe distance is used as distance measure for the comparison of Gujarati character binary images. It is assumed that the characters are in black on a white background. Fringe distances compare only black pixels and their positions between the templates and the input images. An image distance measure between an image *I* and template *T* is the sum of the distances from each black pixel in *I* to the nearest black pixel in *T* and also from each black pixel in *T*, to the nearest black pixel in image *I*. The total distance between *I* and *T* is the sum of these two sums of nearest distances.

Fringe distances may be even more efficiently computed by pre-computing and storing the distances of the nearest black pixel at each pixel position of the template. This is called the fringe distance map. The distances are computed using city-block distance or L1 metric method. The distance between two pixels (X1,Y1) and (X2,Y2) is the sum of absolute values of X1-X2 and Y1-Y2.

When input is compared to a template, the fringe distance map of the input character is computed and superimposed upon the template. The distance form a black pixel in the template to the closest black pixel in the input is already stored at the pixel underneath it no search for the nearest pixel is needed. The distance between the input and the template is the sum of the values in the template fringe distance map corresponding to the black pixels in the input character. Similarly the distance between the template and the input character is the sum of the values in the input fringe distance map corresponding to the black pixels in the template. Fringe distance is the sum of these two distances.

A character, with the minimum fringe distance, is said to be recognized by the template. A numerical code is assigned to each of the 250 templates and the number corresponding to the recognized template is output.

RECOGNITION ALGORITHM IMPLEMENTATION

Flow chart given in Figure 7 depicts recognition algorithm The image is filtered using low pass filter before binarization operation.

Binarization

Optimal thresholding method[19] is used from the methods reported[19-21]. An optimal threshold is calculated using following algorithm.

1. Assuming no knowledge about the exact location of objects, as a first approximation it is considered that the four corners of the image contain background pixels only and the remainder contains object pixels.

2. At step *t*, compute μ^t_B and μ^t_O as the mean background ground and object gray-level, respectively, where segmentation into background and objects at step *t* is defined by the threshold value T determined in the previous step.

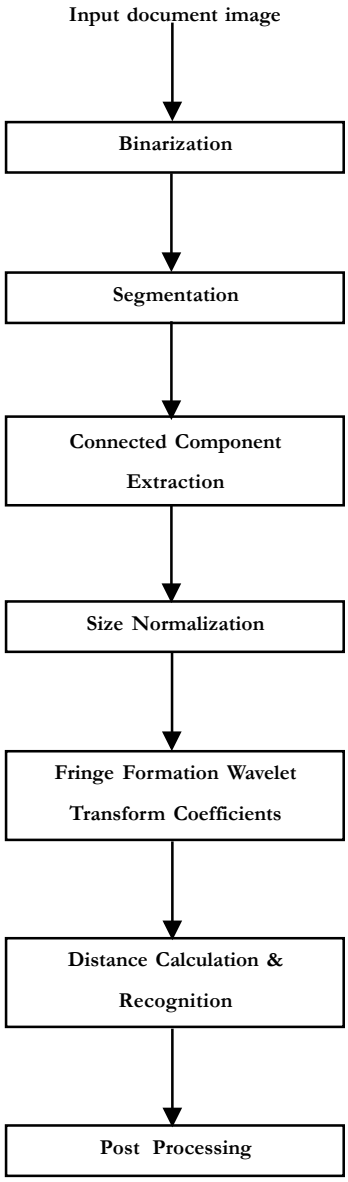

Figure 7 Schematic block diagram of Gujarati OCR

$$\mu^t B = \frac{\sum_{(i,j)\in \text{background}} f(i,j)}{\text{No of background pixels}}$$

$$\mu^t O = \frac{\sum_{(i,j)\in \text{objects}} f(i,j)}{\text{No of object pixels}}$$

3. Set $$T^{(t+1)} = \frac{\mu_B^t + \mu_O^t}{2}$$

$T^{(t+1)}$ now provides an updated background-object distinction

4. If $T^{(t+1)} = T^{(t)}$ halt; otherwise return to step 2.

Figure 8(b) is the result of binarization on the scanned image of Figure 8(a).

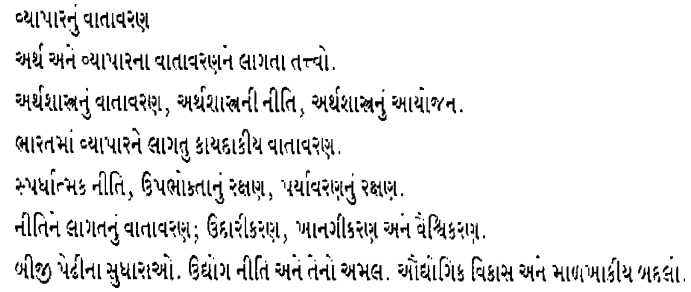

Figure 8 (a) Scanned image

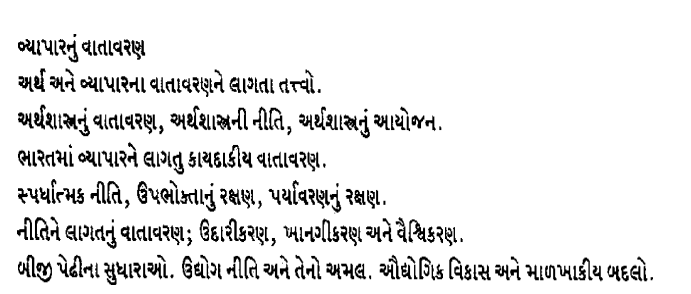

Figure 8 (b) Result of optimal thresholding

Skew Detection

The Skew Detection algorithm[22] can correct skew to within ± 0.05 degrees. Initialization

hzcp is the horizontal crossing count profile for given image.

vhzcp is the variance of the hzcp

minvhzcp is the minimum variance of hzcp

skew is the skew detected in the image

set step = 0.05 degrees

set amount = 0.0 degrees

set max amount = 5 degrees

set minvhzcp = maximum value possible.

set flag as true

Step

until absolute of amount is less than max amount

do

amount = amount + step

rotate the image by amount

get hzcp for the image

calculate variance vhzcp

if minvhzcp > vhzcp

set skew = amount

done

if flag is true

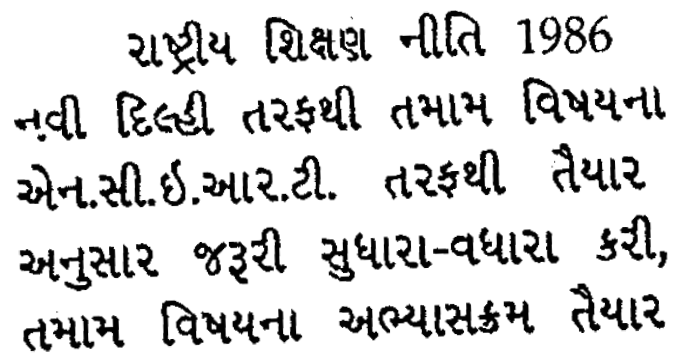

Figure 9 (a) Skewed image

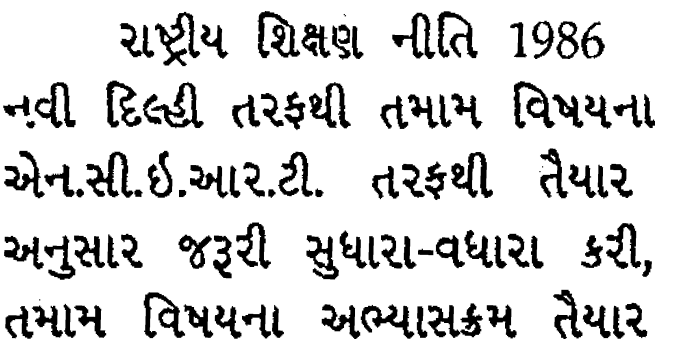

Figure 9 (b) Corrected image

```
        set flag as false
        go to STEP with step = -0.05 and amount = 0.
  end
```

The skewed image of Figure 9(a) was corrected to Figure 9(b) using the algorithm.

Segmentation

It is required to group the lines, words and characters in proper order. This is done using the RLSA algorithm, described here.

Consider horizontal smoothing

```
Assume status=out and some threshold thr.      // in background
  for each row i
    for each column j
        if status = out and Image(i, j) = black pixel
            set status = in
        else if status = in and Image(I, j) = background pixel
            calculate run of black pixels
            if run < thr then
                extend the run of black pixels by threshold
            else set status = out
```

a) Line segmentation is carried out by applying RLSA to a binary image, along columns. *Maatras* and other parts of glyph which lie above and below the base character, are connected to the character but two lines are not connected. A threshold equal to 1/3 of average interline space is used.

b) For word segmentation RLSA is applied horizontally onto each extracted line, with threshold of $1/3^{rd}$ of font size. The word information is extracted using vertical projection profile. The words have non-zero vertical projection, while space has zero as vertical projection.

c) Flood-fill algorithm, described below is used to extract connected component point inside the area to be filled is pushed onto a stack

```
while stack is not empty
pop a point from the stack
label it
if there are any of its neighbouring pixels black
    push them onto the stack
done
```

The extracted component has to be cut into proper zones, *ie*, upper and lower modifiers. The information about where a cut has to be placed is retained when preprocessing of the image is done. The cut-decision procedure is

```
if the component extends well beyond the cutting row on
both  sides
    cut is placed at designated row.
else
    cut is not placed
```

When such component cutting is done the component is relabeled so as to separate the cut components. The cut components are passed on to the recognition phase.

Component Separation

Glyphs have a common property: they come as above or below a character. This can be used to distinguish them from punctuation marks. The properties such as size, aspect ratio and location may be used to identify and recognize the punctuation marks aspect ratio is the ratio of height to width of the glyph. Location is the place where the glyph occurred in the word. Location information is used in recognizing the punctuation mark. The algorithm to separate, punctuation marks, upper modifiers and lower modifiers is given below.

```
To recognize upper/ lower modifiers
height = height of the glyph
width = width of the glyph
by = bottom coordinate of the glyph
ty = top coordinate of the glyph
rx = right co-ordinate of the glyph
lx = left co-ordinate of the glyph
```

Table 1 Results obtained from OCR

	Connected Components		Upper Modifiers		Lower Modifiers		Punctuation Marks		Total	
Correct	825	78.34%	123	50.0%	38	77.55%	8	29.6%	994	72.3%
Incorrect	228	21.66%	123	50.0%	11	22.45%	19	70.3%	381	27.7%
Total	1053		246		49		27		1375	

Thresh = 0.4 * Font size, threshold for the size of the glyph to consider it as a punctuation mark.

if by is above centre of line

return 20 // indicating above centre of line

if ty is below the centre of line

return 30 // indicating lower modifier

returns >10 if the glyph is a punctuation mark or kana or bindu

To recognize a punctuation mark

if height < 0.4*fs AND width < 0.4*fs

if ty > start of line + fs/3 AND by < end of line – fs/3

return 11 // the glyph is '.' in **;** or **:**

if width > 3*height

return 12 //if the glyph is '.' in a character

if height > width AND rx > start of word + width of word -fs/8

AND height < fs/5 AND width < fs/4

return 13 // ' " '

if(width > 2*height)

return 14 // hiphen

else if height > 3 * width

return 15 // kana or exclamation mark

else process as normal character

CONCLUSION

Table 1, shows results obtained from the OCR. These are the raw results with component to component verification (done visually).

A research prototype of Gujarati OCR have been implemented. The accuracy of recognition is low but can be improved upon by modifying distance measures used and tweaking the code. The UNICODE output may be improved by providing additional information such as line boundaries, type of the glyph. For better accuracy following points may be considered

a) Frequently misrecognized glyphs can be identified and analyzed using confusion matrix to improve the accuracy. Confusion matrix is a matrix of size $N \times N$, where N is the number of templates. Special techniques, like modifying the fringe maps can be designed so that they are not misrecognized.

b)When two or more glyphs touch one another, system treats it as a single connected component and could not recognize it. A technology that can separate the touching glyphs can improve the accuracy.

c) Instead of one best match, top 3 or 5 matches can be given in the output so that in the post processing if the 1st match is misrecognized we can search for a correct match in the remaining.

d) Thresholds on distances can be set to label some glyphs as weak recognition or to reject some glyph as misrecognition. This may be used in post processing to locate the errors

ACKNOWLEDGEMENTS

The project was implemented at Resource Centre for Indian Language Technology Solutions Gujarati (RCILTG), MSU, Baroda. Authors would like to express thanks to Prof. Sitanshu Mehta (Principle Investigator, RCILTG) for providing opportunity to work at RCILTG, Shri Jignesh Dholakia , Director IT Solutions, RCILTG, for technical help in successful completion of the project Prof S RamaMohan, Head, Applied Maths Department, FTE, Vadodara for his visionary guidance.

REFERENCES

1. B Krishna. 'Design and Implementation of a Telugu Script Recognition System'. *Technical Report, Department of Computer and Information Sciences*, *University of Hyderabad.*

2. B B Chaudhuri and U Pal. 'A Complete Printed Bangla OCR System'. *Pattern Recognition*, vol 31, no 5, 1997, pp 531-549.

3. B B Chaudhuri, U Pal and P K Kundu. 'OCR Error Correction of an Inflectional Indian Language Using Morphologincal Parsing'. *Journal Of Information Science and Engineering*, vol 16, 2000, pp 903-922.

4. R M K Sinha. 'Visual Text Recognition Through Contextual Processing'. *Pattern Recognition*, vol 21, 1998, pp 463-479.

5. R M K Sinha and H N Mahabala. 'Machine Recognition of Devanagari Script'. *IEEE Transactions on Systems*, Man and Cybernetics, vol SMC-9, 1979, pp 435-441.

6. R M K Sinha. 'Rule Based Contextual Post-processing for Devanagari Text Recognition'. *Pattern Recognition*, vol 20, no 5, 1987, pp 475-485.

7. I K Sethi and B Chatterjee. 'Machine Recognition of Hand-printed Devanagari Numerals'. *Journal of Institution of Electronics and Telecommunication Engineers*, India, vol 22, 1976, pp 532-535.

8. I K Sethi. 'Machine Recognition of Constrained Hand-printed Devanagari'. *Pattern Recognition*, vol 9, 1977, pp 69-75.

9. J C Sant and S K Mullick. 'Handwritten Devanagari Script Recognition using CTNNSE Algorithm'. *International Conference on Application of Information Technology in South Asian Language*, February 1994.

10. I K Sethi. 'Machine Recognition of Constrained Hand-printed Devanagari'. *Pattern Recognition*, vol 9, 1977, pp 69-75.

11. A K Goyal, G S Lehal and J Behal. 'Machine Printed Gurmukhi Script Character Recognition Using Neural Networks'. *Proceedings of 5th International Conference on Cognitive Systems*, Delhi, India, 1999, pp 141- 150.

12. G S Lehal and C Singh. 'A Gurmukhi Script Recognition System'. *Proceedings of 15th International Conference on Pattern Recognition*, Barcelona, Spain, vol 2, 2000, pp 557-560.

13. G S Lehal and C Singh. 'Text Segmentation of Machine Printed Gurmukhi Script'. *In Proceedings of SPIE Conference on Document Recognition and Retrieval VIII*, San Jose, USA, 2001.

14. Dr A Negi, Dr B Chakravarthy and B Krishna. 'An OCR System for Telugu'. *Proceedings of ICDAR*, 2001.

15. A V S R L Prasanna. 'Layout Analysis of Telugu Documents Using RLSA Method'. *Technical Report, Department of Computer Science.*

16. S Antani and L Agnihotri. 'Gujarati Character Recognition'. *Proceedings of the International Conference on Document Analysis and Recognition*, (ICDAR-99), Bangalore, India, 1999, pp 418-421.

17. O D Trier, A K Jain and T Taxt. 'Feature Extraction Methods for Character Recognition – A Survey'. *Pattern Recognition*, vol 29, no 4, 1996, pp 641-662.

18. D M Gavrila, D Benze. 'Multi Feature Hierarchical Template Matching using Distance Transforms'. *Proceedings of ICDAR*, 2001.

19. R C Gonzalez and R E Woods. 'Digital Image Processing'. *Publication Addison-Wesley*, 1993.

20. O D Trier and T Taxt. 'Evaluation of Binarization Methods for Document Images'. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

21. O D Trier and A. K Jain. 'Goal Directed Evaluation of Binarization Methods'. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol 12, no 17, 1995, pp 1191 – 101.

22. B Sankur and M Sezgin. 'Image Thresholding Techniques: A Survey Over Categories'. *Tubitak Marmar Research Centre, Information Technologies Institue*, Gebze, Koeceli, Turkey.

23. O Okun, M Pictikajaen and J Sauvola. 'Robust Skew Detection Based on Line Extraction'. *Machine Vision and Media Processing Group, Infotech Oulu and Department of EE*, University of Oulu.