International Journal of Computer and Communication Technology

Volume 1 | Issue 2 Article 1

April 2010

Development of Optical Character Recognition Software Package for Mobile Phones

Sukhwant Kaur
Fr.CRIT, Vashi Navi Mumbai, Maharashtra, sukhwantsagar@gmail.com

H. K. Kaura
Fr.CRIT, Vashi Navi Mumbai, Maharashtra, hkkaura@gmail.com

Mritunjay Ojha
Fr.CRIT, Vashi Navi Mumbai, Maharashtra, mritunjayojha@rediffmail.com

Development of Optical Character Recognition Software Package for Mobile Phones

Sukhwant Kaur
Asst. Professor
Fr.CRIT, Vashi
Navi Mumbai, Maharashtra
sukhwantsagar@gmail.com

H.K.Kaura
Professor
Fr.CRIT, Vashi
Navi Mumbai, Maharashtra
hkkaura@gmail.com

Mritunjay Ojha
Lecturer
Fr.CRIT, Vashi
Navi Mumbai, Maharashtra
mritunjayojha@rediffmail.com

***Abstract*— Optical Character Recognition (OCR) is a technique through which any textual information contained in images are extracted and converted into editable text format. The various OCR software packages which are available in desktop computer with scanner suffer from one primary constraint- MOBILITY.**

We have developed an OCR application for mobile phones. All the procedures needed for extracting the text would be performed within the mobile phone, eliminating the need for bulky devices like scanners, desktops and also laptops. Hence it would provide the user the much needed 'anywhere, anytime' feature for OCR. The computational power of mobiles is increasing day by day making it easier to run image processing operations for OCR application. Also the resolution of camera in mobile is increasing to match the resolution of scanners. After the document is processed, it can be communicated to another user by email facility of mobile phones as text files.

The aim of this paper is to investigate the various issues involved in developing this OCR application in mobile phones. Further design and future scope for this application is elaborated giving insight to the development process. The motivation here was to provide a general purpose framework for OCR application in mobile phones. The framework is developed in a modular fashion.

Keywords: Image Processing, Normalization, Feature extraction, Thinning, OCR, Thresholding.

I. INTRODUCTION

Optical Character Recognition (OCR) is the process by which the printed characters in a document or an image are converted automatically into textual data which can be easily processed by computer.

We have developed an OCR application for mobile phones wherein all the procedures needed for extracting the text would be performed within the mobile phone, eliminating the need for bulky devices like scanners, desktops and also laptops.

The user should be able to capture the image of a document. Images captured at different distance from document should result in the same output. The document may contain text [alpha-numeric] of varying size but of a fixed font, say Times New Roman. It extracts each individual character from document image and also its characteristic features. These are compared with the database of extracted features of the same font of an appropriate size, say size 12. The recognized text is displayed on the screen and can be saved to the phones memory if desired so by the user. It should be editable by any standard text editor.

Developing such a system on a mobile phone needs to overcome several issues.

a) The image reproduction in mobile phone is different from flatbed scanners.

b) The processing power and memory available on the mobile phones is limited. So the software developed should be compact and very efficient.

c) Also 1:1 mapping of image and the document is not always possible due to variation in distance and inclination of the camera from the document.

d) Insufficient contrast and brightness may render the image ineffective for further processing steps.

e) Images captured with the integrated camera usually suffer from motion blur and perspective distortion problem as well.

f) Noise in the input image is either due to the quality of paper on which printing is done or due to printing both sides of paper "Back paper Noise". All these noises contribute to the decrease in accuracy of OCR system. As a result of this having a noise correction routine in place becomes inevitable.

Pursuing this application, we have dealt with all these issues and developed a simplistic OCR application, which at the present, can recognize a fixed font i.e. Times New Roman of size 12. However in future the algorithms can be extended for recognizing several fonts of various sizes.

II. SYSTEM DESIGN SPECIFICATIONS

Mobile phones have limited processing and memory capabilities as compared to a desktop system. However, Windows Mobile 6 phones has a processor of at-least 400 MHz and 128 MB RAM. OCR is a processor intensive process as it involves lots of image processing. Also, due to the size of images in question [3.2 Megapixels = 3.2 * 10^6 * 3 = 9.6 MB as there are 3 bits per RGB Pixel] a lot of memory is also required. Of the 128 MB of RAM, the operating system and other critical task consume almost 70-80 MB of RAM. The remaining memory is shared by the users own applications.

The more pixels the image contains, the better will be the representation of each character. A minimum of 3.2 megapixels is required to get the required image quality. .Net Compact Framework 3.5 is the latest version of the framework. It offers many performance improvements over the previous versions. The places where the performance isn't up-to the mark, we code them in unmanaged code, i.e. native C++ code. Using the combination of .Net CF and unmanaged code, we get both the ease of development as well as performance where needed. Writing the entire application in native C++ will take much more time to accomplish and the performance improvements might not be worth the effort.

Considering these constraints the minimum hardware and software requirements of the application is as follows:

a) 400 MHz processor

b) 128 MB RAM

c) 3.2 Megapixel camera

d) Windows Mobile 6.1 Professional

e) .Net Compact Framework 3.5

III. PROPOSED SYSTEM

We are proposing the technical feasibility of an OCR application developed for a Windows Mobile 6 mobile phone using the C# programming language on the Microsoft .Net Compact Framework 3.5.

First the image of the document to be digitized is captured using the inbuilt camera in the mobile phone. Images can also be selected from the gallery in the mobile phone's memory. This system can handle formats like JPEG, PNG, GIF, and BMP. However mobile devices prefer storing images in JPEG format due to the fact that it requires lesser amount of memory space for storage.

Initially the application would ask the user to choose an area of interest in the document image, say for example a single paragraph from the image of an entire A4 size paper. When the user has selected the desired area, the image is converted into grayscale. This is followed by generation of histogram for the image. This histogram is used to calculate the threshold value needed to binarize the image. This image is then processed to remove salt and pepper noise. This is followed by horizontal segmentation of the entire image followed by vertical segmentation to separate individual characters. The image is then normalized for each character and is made to undergo thinning. Further the geometric features of all the characters extracted are used to finally identify the printed letters.

A. Pre-Processing Steps

The user will click the picture of an entire document. Before the image can be processed for recognition, several pre-processing steps need to be performed. At the end of these steps we get images of each individual character in the document. These are passed on to the feature extraction and recognition module.

The steps involved are as follows:

a) Convert image to grayscale

The camera captures a colour image. To reduce the burden of processing the image we need to convert the image to black and white. But before that, the colour image must first be transformed to grayscale.

The conversion is carried out using the following formula on each pixel values.

Luminance = (.299 * red + .587 * green + .114 * blue)

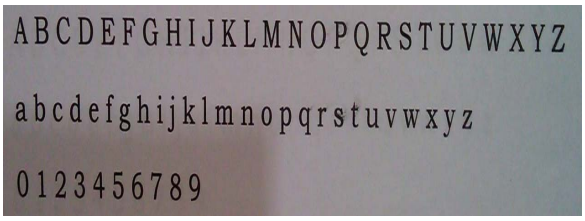


Figure 1. Original Colored Image

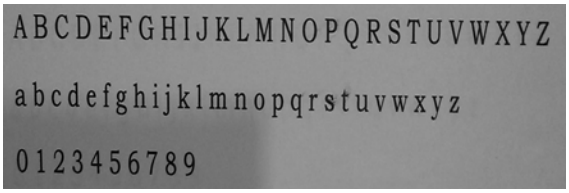


Figure 2. After converting to grayscale using the luminance formula

b) Noise removal

The most common form of noise in black and white images is salt and pepper noise. Salt noise is random occurrence of white pixels in regions of black. And pepper noise is the occurrence of black pixels in regions of white.

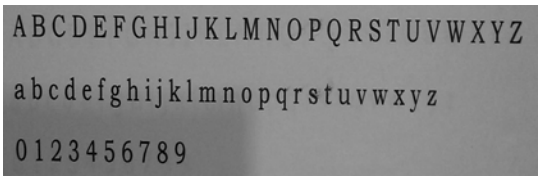


Figure 3. After noise removal

c) Threshold the image

We select a particular grayscale intensity value which will act as our threshold for binarizing the image. All intensities below this value will be treated as the color black and all values above it as white. The resulting image is black and white.

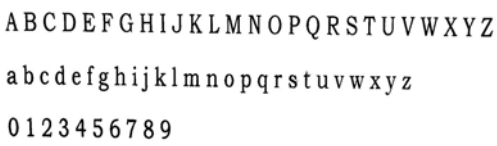


Figure 4. After Thresholding

d) Segmentation

The image, though black and white, still contains rows of words and not individual characters. Segmenting the image will help us extract individual characters from the document. First the document will be segmented horizontally to get individual rows of text.

Each row will be further segmented vertically to get each character.

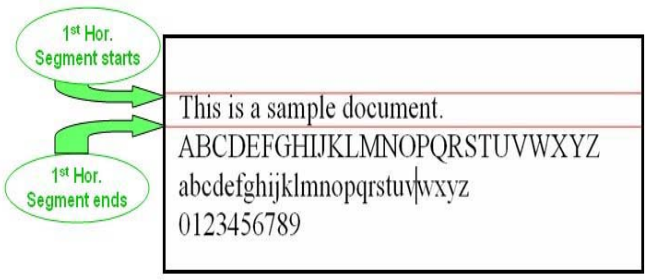


Figure 5. Horizontal Segmentation

Each row will be further segmented vertically to get each character.

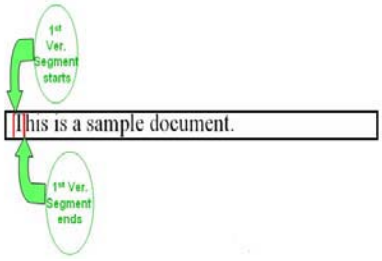
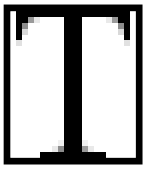


Figure 6. Vertical Segmentation

As a result of whole segmentation process, characters will be obtained as individual segments as shown below:

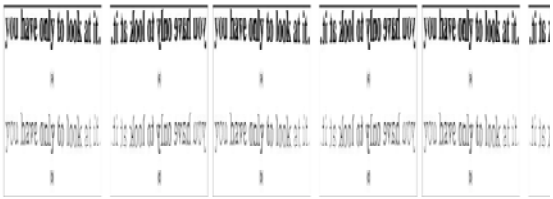


e) Normalization

The individual characters extracted in the previous step may be of varying dimensions. The images need to be brought to a standard size. This helps in reducing the size of the database of extracted features as the features may or may not be invariant to size, translation, rotation.

f) Thinning

Thinning an image results in a single line representation of the image that still has the characteristic features of the original image. Thinning is carried out to make recognition of the character easier. Thinning the image also helps reduce the memory and processing needs for the image.



B. Feature Extraction Steps

The final output of the steps above contains images of individual characters. From these character images we have to extract certain characteristics features. These features must be selected such that they are invariant to size and translation. The features can be boundary descriptors, area descriptors, region descriptors etc.

Depending on how good the extracted features are able to differentiate between characters of the same size same font, character of different size same font and characters of different fonts, we will suggest the most optimal extraction method.

In our application we have tried three different feature extraction methods -

i. **Hu's Invariant Moments:** Moment-based invariants explore information across an entire image rather than providing information just at single boundary points. They can capture some of the global properties missing from the pure boundary-based representations like the overall image orientation. Central moments are computed using the centroid of image, which is equivalent to regular moments of an image whose center has been shifted to coincide with its centroid therefore central moments are invariant to translations.

ii. **Sectional Features:** It divides the unknown character image into multiple regions for its feature extraction. In this method of feature extraction 14 features are extracted from a character. They are:

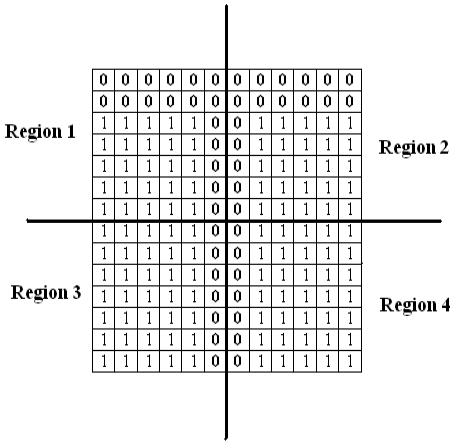
   i. Height / Width ratio

   ii. Number of black pixels / Number of white pixels ratio

   iii. Number of horizontal transitions

   iv. Number of vertical transitions

The horizontal and vertical transition is a technique used to detect the curvature of each character. In addition the image is divided in to four regions as shown and the following are calculated:



   v. Black pixels in region 2/ White pixels in region 2 ratio

   vi. Black pixels in region 3/ White pixels in region 3 ratio

   vii. Black pixels in region 4/ White pixels in region 4 ratio

   viii. Black pixels in region 1/ Black pixels in region 2 ratio

   ix. Black pixels in region 3/ Black pixels in region 4 ratio

   x. Black pixels in region 1/ Black pixels in region 3 ratio

   xi. Black pixels in region 2/ Black pixels in region 4 ratio

   xii. Black pixels in region 1/ Black pixels in region 4 ratio

   xiii. Black pixels in region 2/ Black pixels in region 3 ratio

   xiv. Black pixels in region 2/ Black pixels in region 3 ratio

Distance Vectors: It performs character recognition by quantification of the character into a mathematical vector entity using the geometrical properties of the character image.

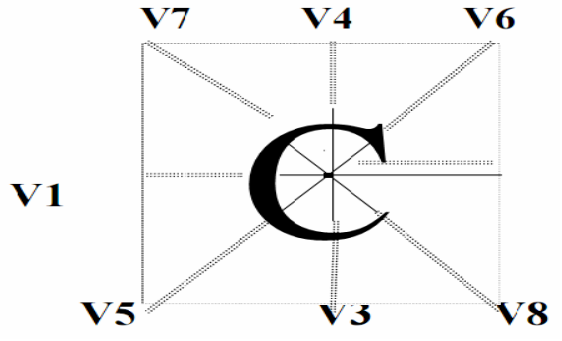
Algorithm for training database:

   i. We find the centre of character.

   ii. 8 concurrent lines are drawn taking centre of character as point of concurrency. Inclinations of lines are 0, 45, 90, 135, 180, 225, 270, 315 degrees.

   iii. For each line, if it intersects character, point of intersection line of with character is found and its distance from centre of character is determined. If there is no intersection, distance = 0.

These 8 values can now be used as characteristic features of respective character.



Feature extraction of 'C' by calculating 8 distances

V1	V2	V3	V4	V5	V6	V7	V8
19	19	10	37	43	19	19	43

The vectors are stored in the database

Algorithm for character recognition:

i. In order to recognize any unknown character, we find the 8 vectors as stated previously.

ii. Each row of existing database is divided by this vector and array ratios containing 8 values are obtained after each division.

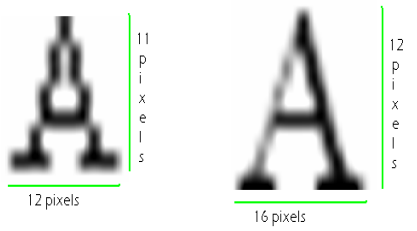
iii. Now array of ratios with minimum variance is found out and the corresponding row in database is selected.

iv. Then we say, given unknown character is equal to character associated with selected row in database.

IV. Recognition Results for all the three Feature Extraction algorithms:

The recognition results for all the three algorithms worked as a tool for us to find the most optimum algorithm for our application. We found that Distance Vector algorithm gives us maximum percentage of recognition of characters.

a) Hu's Invariant Moments:



Consider the two A's above. Their font sizes and the dimensions are displayed. From the figure the following can be understood:

i. Thickness of letter doesn't change with font size. Because of this number of white pixels increase, but the number of black pixels doesn't increases proportionally. So when moments are calculated for both A's (i.e.A with size 12 and A with size 16) with the first Hu's formula, values obtained are:

   a. Times New Roman A Size 12 - > **0.13403**

   b. Times New Roman A Size 16 -> **0.17197**

ii. One can see here that even if the alphabets under consideration were almost identical, there lies large difference in their moments. Moreover, alphabets with no connection in terms of looks also possess moment values which are mathematically close. For example, for Times New Roman O Size 14 -> **0.129277.** These close values make the recognition highly difficult!

iii. Aspect ratio for different font sizes is different i.e. the ratio (height / width) for a specific alphabet is not same for samples of different sizes. But in the discussion done above, we have seen that moments are invariant to scaling only when scaling along both X axis and Y axis, are same. But different sized samples of certain alphabet are not perfect scales of each other (i.e. scaling along X axis ≠ scaling along Y axis).

Hence we cannot use Normalized Central Moments for our feature extraction.

b) Sectional Features:

On testing the algorithm against the sample images, the following observations were made. Consider character A of the same font but sizes 14, 16, 18. The sectional feature values are as follows:

Size 14 A	1.0278 0.4718 124 93 0.2857 0.3966 0.52 0.7449 0.7826 0.8014 0.6154 0.6301 0.4932 0.7863
A Size 16	
1 0.4824 140 88 0.2783 0.3824 0.5312 0.8529 0.7869 0.7537 0.6275 0.601 0.4729 0.7974	
Size 18 A	
1 0.4713 174 120 0.2414 0.4083 0.5319 0.817 0.6707 0.7722 0.56 0.6448 0.4324 0.835	

Even though it's the exact same character, but a different size, there is huge variation in the same feature values. For eg, the horizontal transitions for each vary from 124 – 174. Because of these variations, recognition becomes difficult.

If we try to normalize to the images to a standard size, 30x30, even then there are too many variations between images of the same character of different font size.

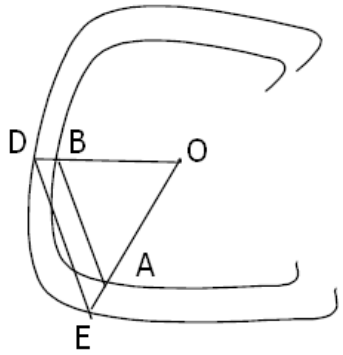
Size 14 A
1 0.3761 96 78 0.2363 0.3314 0.3889 0.5957 0.7679 0.75 0.6825 0.6667 0.5119 0.8889
Size 16 A
1 0.3025 84 74 0.1658 0.2363 0.3393 0.5203 0.7442 0.7403 0.5614 0.5584 0.4156 0.7544
Size 18 A
1 0.4173 104 77 0.2097 0.3473 0.4803 0.7308 0.6724 0.7684 0.5342 0.6105 0.4105 0.7945

Thinning the images worsen the results. The thinned image is a 1 pixel width representation. As a result, many times when forming sections, the only pixel available might go over to the one of the two possible regions, i.e. there is no guarantee that character A's thinned image will always have a particular ratio of pixels in the regions.

This played a part in the algorithm having poor accuracy in our system.

c) Distance Vectors

The reason Distance Vectors works so very well is that its invariant to the size of the image. In our case that also translates in to the fact that it is invariant to the height at which the camera is kept above the document.



Consider the following. Here "O" is the centroid of both the figures. In the figure, we have ΔOAB ≡ ΔODE, Therefore, OA/ OE = OB/ OD.

Therefore, in terms of the elements of the characteristic vectors of a characters of two different sizes, we have, Ratio(r) = X1/X1′ = X2/X2′ = X3/X3′…… Xn/Xn′ where X1 and X1′ are a particular characteristic value of the smaller and the larger character images of the same character.

Finding the ratios actually normalizes the ideal database to be in terms of the unknown image. From this normalized state, it's just a matter of finding the least variance. In the other techniques the normalization was the main reason for the features values being so varied.

The results also highlights that certain letters were almost always misinterpreted by the same set of letters like

a. "I" is mainly recognized as "1".
b. "G" is mainly recognized as "O".
c. "S" is mainly recognized as "8".
d. "O" is mainly recognized as "o".

For that, unknown character "G" vectors should be compared with the vectors of "C", "G" and "O".

V. Conclusion

We have implemented an OCR system using only a mobile phone for all tasks. The system can convert to text images of documents with font Bookman Old Style of any size. The system has an accuracy of around 75%. The recognition happens quickly within 2-3 minutes. The system is invariant to font size and the height at which the camera is placed above the document. The accuracy and recognition is not sufficient for practical use. It may require significant improvement.

References

[1] Haidar Almohri, Gray John S., Hisham Alnajjar. – A REAL-TIME DSP-BASED OPTICAL CHARACTER RECOGNITION SYSTEM FOR ISOLATED ARABIC CHARACTERS USING THE TITMS320C6416T. *www.ijme.us/cd_08/PDF/228%20ENT%20201.pdf.*

[2] Nadeem Danish, Rizvi Saleha. CHARACTER RECOGNITION USING TEMPLATE MATCHING. www.cs.uic.edu/~srizvi/BIT_Thesis.pdf.

[3] Mohammed Cheriet, Nawwaf Kharma, Cheng-lin Liu – CHARACTER RECOGNITION SYSTEM, Wiley Publications.

[4] Gonzalez and Woods, Digital Image Processing, 3rd Edition, Prentice Hall.