International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970)
Volume-4 Number-2 Issue-15 June-2014

Printed Text Character Analysis Version-I: Optical Character Recognition with the new User Training Mechanism

Satyaki Roy[1], Ayan Chatterjee[2], Rituparna Pandit[3], Kaushik Goswami[4]

Abstract

The present system aspires to analyse snapshots of written text and create fully customizable text files using Optical Character Recognition (OCR) technology. It is well known that the discrepancies in typed optical language have led to the advent of new technology for assessing the written text. Many font sizes and styles are introduced everyday calling for frequent updates in recognition technology and associated systems. There are existent systems for character recognition but they are limited in their scope because of their inability to recognize the latest writing styles, however the present system gives the user complete liberty to effortlessly train the system to handle new fonts using the character dictionary and user training mechanism.

Keywords

User Training Mechanism, Independent Scaling of Characters, Resizing Algorithm, Character Dictionary, Character Recognition

1. Introduction

The greatest problem with any software system is its limited lifecycle. The problem is manifold in case of systems that process the text characters. Every day new writing styles are effortlessly introduced. However the software systems are only trained to handle a limited range of writing styles. Therefore such systems become outmoded very soon. In this paper, we propose a system that would analyse printed text and recognize characters from text images like any other Optical Character Recognition (OCR) system. The system is also equipped to cope with newly emerging writing formats and font styles. The system incorporates a User Training Mechanism that would allow the user to train the system to incorporate new character font styles. The User Training Mechanism would therefore ensure that in the near future all occurrences of the trained font would be recognized instantly by the system.

The ability of this system to adapt to new writing styles would ensure that it would last over long periods of time unlike the existing character recognition devices. This system will have immense potential in day to day life because currently people are more comfortable clicking pictures of valuable documents. This system would help them to convert any text image into a customizable text file instantly.

It also includes the user requirement of introducing new font styles thus broadening the range of its utility. This paper covers the first stage of printed text analysis and there is a good deal of improvements that we shall incorporate in the subsequent versions to make the process of character recognition more accurate and effective.

2. Overview of System Components

This system has two distinctive modules:

- The User Training Module
As mentioned before, the User Training Module is allows the user to train the system to recognize new characters and font styles. The character information gets stored in the Character Dictionary which may be used for all future references of training.

- The Character Recognition Module
The Character Recognition module actually performs the recognition of characters. It uses the character dictionary (discussed later) as the database to recognize the characters from a text image. The User Training and Character Recognition modules use a set of components which have been depicted in the block diagram in figure 1.

Manuscript received May 19, 2014.
Satyaki Roy, St. Xavier's College, Kolkata, India.
Ayan Chatterjee, St. Xavier's College, Kolkata, India.
Rituparna Pandit, St. Xavier's College, Kolkata, India.
Kaushik Goswami, St. Xavier's College, Kolkata, India.

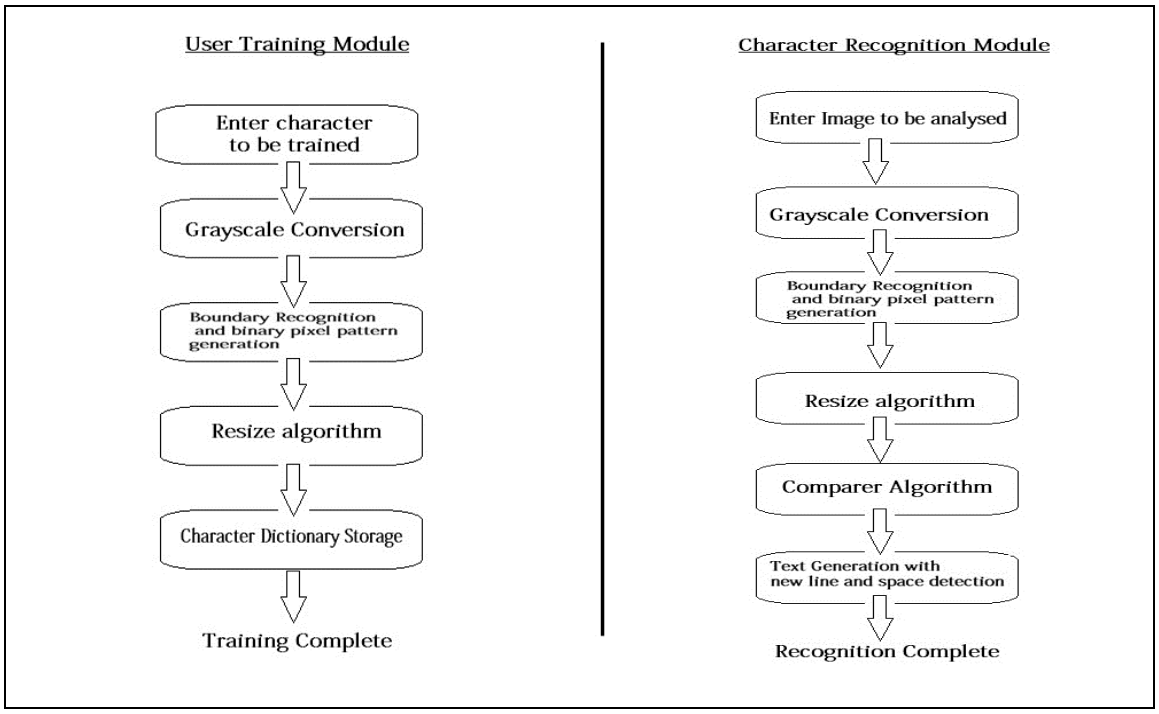

Figure 1: Overview of the Components of User Training and Character Recognition Modules

3. Details of System Implementation

This section provides a detailed insight into the various components of this system.

A.Grayscale Conversion

The prime objective of this module is to reduce the number of shades in the image. We know that grayscale images only provide shades ranging from black (generally denoted by 0) to white (generally denoted by 255). This module would therefore assist the system in detecting the difference between character pixels and background pixels. Typically character pixels have shades closer to black and background pixels have shades closer to white. The grayscale conversion algorithm works in the usual way by extracting every pixel value 'px' and finds its red, green and blue components and replaced the R, G and B values of 'px' by the average of the red, green and blue components denoted by 'avg'. In the algorithm shown, px1 is the converted pixel (Let us assume that the image is in Alpha-Red-Green-Blue format. **Here '>>' represents right shift operation and '&' is bitwise AND operation**.)

for every pixel 'px' in the text image 'IMG',
alpha= (px>>24)& 0xff;
red = (px>>16)& 0xff
green = (px>>8)& 0xff
blue = px & 0xff
alpha = 255
avg = (red+green+blue)/3
px1 = (alpha<<24) + (avg<<16)+(avg<<8)+avg

B. Character Boundary Recognition

This is the most significant component in the system. This section deals with the identification of the character boundaries. This process is very crucial as more effective the boundary recognition, better is the quality of character extraction. The underlying principle behind the recognition of character boundary is the positioning and alignment of the darker pixel values. This algorithm extracts four character boundaries for every line of written text:
-the Vertical Top Line
-the Vertical Bottom Line
-Horizontal Left Character Line
-Horizontal Right Character Line

Each of these boundaries is depicted in the illustration figure 2.

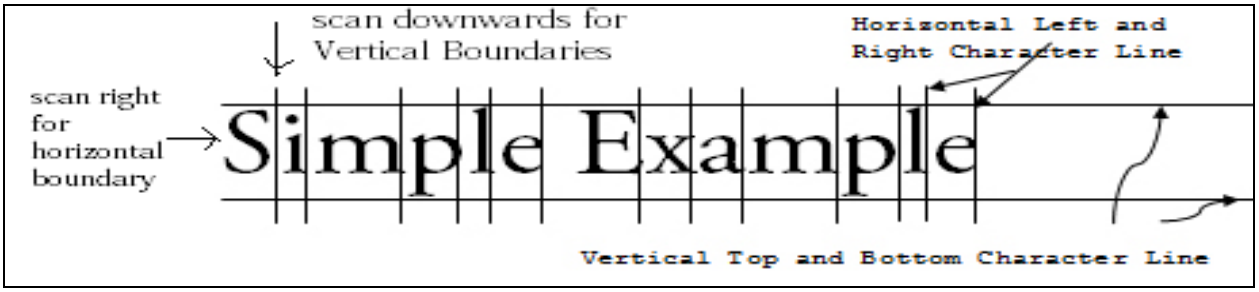

Figure 2: Boundary Extraction Process

As mentioned before the character boundary extraction is a key component in both the modules i.e. user training and character recognition.

With the help of figure 2, let us see how this component works. The first target here is to determine the vertical top and bottom character boundaries. These boundaries together demarcate a line of printed text.

Vertical Top Boundary Line Extraction
Let x = 0 and y = starting row 'st' (initially)
for every y less than the height of image IMG do

for every x less than the width of the image IMG do

if IMG[x][y] < = thresh then
return the value of y;

end loop
end loop

If we have an image matrix 'IMG [x] [y]' where 'y' is the row of the image matrix and 'x' is the column of the matrix, we scan downwards from the start row number 'st' (where st is initially set to 0 and otherwise represents the previous vertical bottom character line). As soon as it detects a pixel value which is less than the threshold value 'thresh', it considers the row to be the Vertical Top Character line and returns the row 'y'. For the Vertical Bottom Boundary extraction, the same process is repeated, only this time, the downward scanning starts from top boundary line. Therefore st = Vertical Top Boundary Line.

Horizontal Left Boundary Line Extraction

C. Binary Pixel Pattern Generation

Once the character boundary has been recognized, the system generates the binary pixel pattern which is a binary string that stores the pixel values lying within the window of the character boundaries.

As we have mentioned before, in the pixel pattern 'pix', character pixels are represented by 0 and background pixels are represented by the value 1.
for every 'i' ranging from Vertical Top Boundary to Bottom Boundary do

for every 'j' ranging from Horizontal Left to Right Boundary do

if IMG [j][i] < = thresh then
Append "0" to pix // to signify character pixel

else
Append "1" to pix // to signify background pixel
end inner loop
end outer loop

This binary pixel pattern is resized and stored in the database called the character dictionary for character matching process.

D. Resizing Algorithm

Ravina Mithe, Supriya Indalkar, Nilam Diveka has specified in [6] that normalization techniques have to be incorporated to make sure that uniform sizes are maintained to enhance the process of character recognition. In our system, after extraction and generation of the binary pixel pattern 'pix', the extracted character is resized to a default value of 10 x 10. This ensures a few things:
-It helps the pixel-wise character matching process, because every character is reduced to the same size.
-It reduces memory overhead because 100 character values are stored for each character
-It speeds up the matching process because the time taken to perform 100 comparisons is not very high.

The resizing algorithm works on the principle of mapping. It simply scales down a matrix of certain

pixel size to a 10 x 10 matrix. We would like to point out here that the size of 10 X 10 is not fixed.
We have experimentally determined that we are obtaining an optimum performance and quality with the 10 X 10 binary strings. If the window size is increased then the time to perform character recognition will also be quite high.

In the resizing algorithm below:
The values 'h1', 'w1' are the height and width of the original binary pixel matrix, whereas both 'h2' and 'w2' are set to 10 (which is the size of the reduced or resized pixel matrix).
The resized character matrix is stored in 1-D array 'temp'.

w1=original character width
h1=original character height
h2=10
w2=10
'a' is the one dimensional array that stores the original binary pixel string
*'temp' is the one dimensional array of size (h2*w2) that stores the resized binary pixel string*
x_ratio = w1/w2
y_ratio = h1/h2
Let px and py be the x and y components of the resized pixel value.
for every 'i' ranging fro 0 to h2 do,
for every 'j' ranging from 0 to w2 do,
*px = j*x_ratio*
*py = i*y_ratio*
*temp[(i*w2)+j] = a[(int)((py*w1)+px)]*
end inner loop
end outer loop

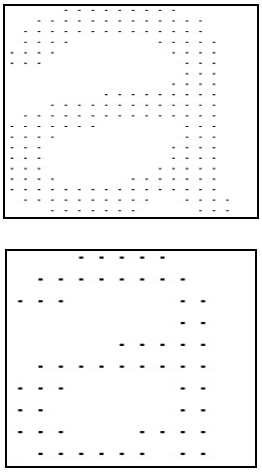

Figure 3: Original Character after extraction (top) and resized character (bottom)

E. Independent Character Scaling
There is another small task which remains to be done after the extraction of characters. When there are two characters adjacent to each other that have varying heights, then the character which is shorter tends to have a white space on top as evidenced by figure 4 shown below.

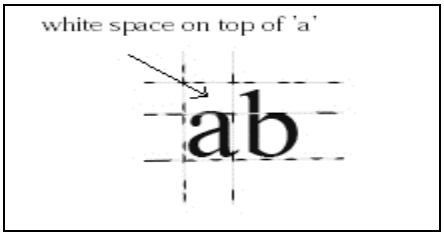

Figure 4: The white space on top of character 'a' that exists because 'b' has a greater height

The independent character scaling algorithm simply removes the white space on any character after extraction. This is very important for accurate character recognition.

F. Storage into Character Dictionary
The system must be able to record the patterns for new character as discussed by Sukhpreet Singh in [7]. It makes sure that all future references of a known character are successfully recognized. The database called character dictionary does that by being the mainstay of the User Training Mechanism. Once the binary pixel pattern has been generated for a certain character, the pixel pattern is transferred to the character dictionary. In our algorithm we have used a MS Access database but any other data storage can be utilized. The dictionary has the following fields:

- Character name
- Binary Pixel Pattern String
- Aspect Ratio (to be used for optimization purposes in the next version of our algorithm)

We must understand that the character dictionary does not function as specified in [8] which detect misspellings. This dictionary is for character recognition alone.

Table 1: Character Dictionary Sample for character 'a'

Character Name	Binary Pixel String 100 pixel values	Aspect Ratio
A	11111000111…	1.1818

It must be remembered that the system already has a character dictionary in place. The advantage here is that the user may add new characters to the dictionary with the user training module and introduce new writing styles at his will.

G. Comparison and Matching Algorithm

This technique of character recognition/ matching works with the pixel matching method where the corresponding 10 X 10 values of the extracted pixel pattern is resized and matched against the pixel pattern of the character dictionary. If the number of matched pixels exceeds a threshold value (generally set to 85 out of 100 pixels) we can call it successful character recognition. The simplistic matching algorithm is described here,

For every character entry in the character dictionary, do
Define counter = 0
For i = 0 to (10 X10 or) 100 do,
'pix_a' is the ith pixel value of extracted and resized character
'pix_b' is the ith pixel value of the given entry of the dictionary
If pix_a is equal to pix_b
Increment counter
End loop
If 'counter' exceeds a threshold value (generally 85 pixels out of 100), then the match is deemed successful.
End loop

H. Newline and Space Detection

We must understand that merely recognizing characters is not enough. The system must be able to detect end of character lines and spaces in between characters as well. Therefore we have a separate mechanism to recognise newlines and spaces.

- Newline Detection: Previously we have spoken about the recognition of the Horizontal Right Character Boundary and the Vertical Bottom Character Boundary.

The algorithm works as follows:
If the right boundary returns -1, it indicates end of line. If bottom boundary returns -1, it indicates end of the page.

-Space Detection: If the algorithm detects a gap of atleast 10 pixel columns between adjacent characters, it considers the gap to be a space. The value of 10 pixel columns has been determined experimentally.

4. Overall System Algorithm

We have just discussed the independent components of the system. Now let us discuss the overall algorithm we have implemented in the system.

System algorithm for Insertion Module –
Step I: Read image- IMG of graphical symbol or character to be trained to the system
Step II: Convert IMG to its grayscale equivalent with shades ranging from 0 (black) to 255 (white).
Step III: Extract the boundary of the character to be trained.
Step IV: Calculate the aspect ratio = width/height for the character.
Step V: Create the binary pixel pattern generation.
Step VI: Resize the extracted character and store the information in the character dictionary.
Step VIII: End

System algorithm for Character Recognition Module –
Step I: Read image- IMG of graphical symbol or character to be recognised.
Step II: Convert IMG to its grayscale equivalent with shades ranging from 0 (black) to 255 (white).
Step III: Extract boundary for every character that is encountered.
Step IV: Calculate aspect ratio = width/height for the character.
Step V: Create the binary pixel pattern generation and resize the character
Step VI: Perform top and bottom independent scaling of character
Step VII: Apply the comparison and matching algorithm to recognise the character
Step VIII: If match is found print character and space (if necessary).
Step IX: GOTO step III for the extraction and recognition of the next character.
Step X: End

5. Test Results

The system we are proposing is already in working condition and we would like to illustrate the ways in which the two main modules, namely the User Training Module and the Character Recognition Module function. The working of both the modules have been briefly illustrated in the following sections A and B, by applying them on a character images-

A. Working for the User training module

We have shown the steps involved in the training module using the character 'b'.

Step 1: The user may use the User Interface to upload an image of the character he wishes to introduce into the character dictionary. The character image is converted into the corresponding grayscale.

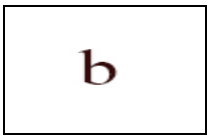

Figure 5: Character to be trained

Step 2: The boundary is recognized and character is extracted (Figure 6).

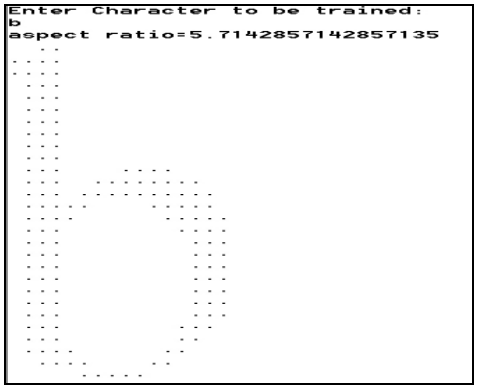

Figure 6: Character after boundary recognition and extraction.

Step 3: The character is resized and independently scaled (Figure 7).

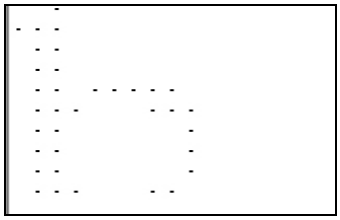

Figure 7: The character is resized.

Step 4: The user training process is complete.

B. Working for the Character Recognition Module

Step 1: The image is uploaded for character recognition (Figure 8).

Figure 8: The image for recognition

Step 2: The individual characters are identified. In this case the characters 'a' and 'b' are recognized and resized.

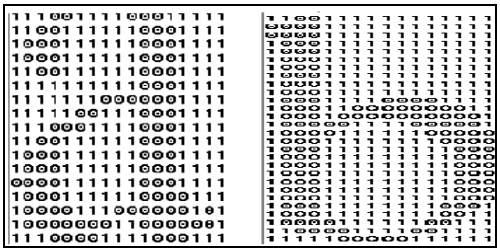

Figure 9: The binary matrix for the extracted character where 1 represents background and 0 represents character pixels.

Step 3: As depicted in Figure 10, for every character whose boundary is recognized, we perform a match with a database entry. (In figure 10, after extraction we find a 95 percent match with a database entry of 'a' even though the percentage match may be as low as 80 percent.

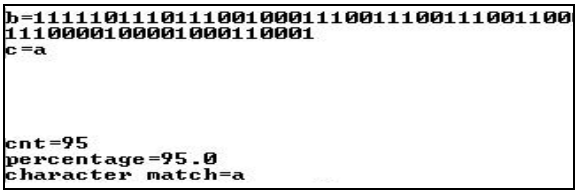

Figure 10: The final character recognition stage is depicted in which the extracted character is yielding a 95 percent match with a database entry of 'a'. This is an example of a successful match.

In the diagram Figure 10, shown above, the value of variable b is the resized binary string and 'cnt' represents the counter representing the number of matched pixel values. The percentage match is seldom 100 percent because the character shape undergoes some change after resizing.

Step 4: The process of character recognition continues until all the characters are recognized. In

figure 11, we have a printed text with only two characters.

Figure 11: Final Printed text 'ab'

6. Comparison with existing systems

Like we mentioned in the introduction, the existing systems of optical character recognition become less effective because they are hardcoded. They are unable to incorporate unconventional characters or graphical symbols as required by specific users.
Consider the image in figure 12 –

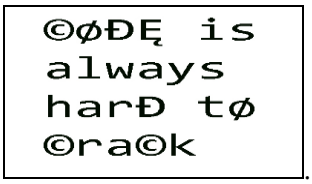

Figure 12: The image with unconventional graphic symbols

In figure 12, you shall see that the characters 'Ð', 'ø', '©', 'Ę' are unconventional graphic symbols. They are seldom used in text documents and hence the existing OCRs struggle to recognise them because they are rarely trained for such characters.

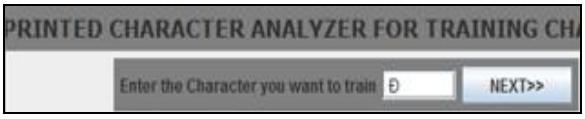

Figure 13: The character training/insertion module being used to train character 'Ð' which is going to be recognized on all future occurrences.

Printed text characters provide the user the liberty of training the system to recognise any unusual graphic symbol as shown above. Once such characters are trained the process of recognition of unconventional graphic symbols becomes lot simpler. The user can thus customize the system as per his needs.

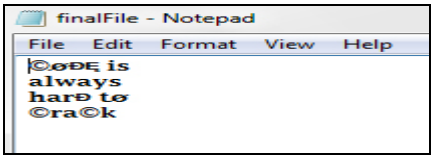

Figure 14: The characters in the image in figure are easily recognized by the system because unconventional graphic symbols have been trained.

As we consider **related work**, let us understand that our present system has implemented the concept of Pixel Grabber class in java to read the text images, as specified in [1] and [3]. The entire concept of brighter pixels for the image background and darker pixels for the characters has been conceived from [2]. It must be understood that the system we are proposing in this paper is fundamentally different from the existing literature on optical character recognition. Mori and Suen, in literature [4], have discussed the concept of neural networks and template analysis whereas our system is based on the concept of pixel matching algorithm (described before). Many surveys, as described in [5], show that a system should have the pre-processing and recognition techniques in place that would cope with online and offline character recognition techniques. Our test results clearly show how our system can be utilized to recognize conventional and unconventional characters without any difficulty. However we have seen examples of cursive font handling where the system requires incorporating shape recognition and slant-angles to handle cursive fonts. The system will look to incorporate cursive fonts in its future versions as discussed in the conclusion section.

7. Conclusion and Future Scope

While working on this system, we came up with a few interesting insights into the future improvements on this system. We have been working extensively on the given system and tried out multiple font sizes, writing styles and formats. The results have been rather promising. In the conclusive section of the paper, we shall discuss the few aspects which are imperative to address in our subsequent versions.
-Firstly, the printed text analyzer must be capable of processing long text images in little time. Therefore the character recognition process must be enhanced. We are working on a series of optimization mechanisms which will be based on the intrinsic characteristics of the written text. For example, we

have come up with ideas of working on the aspect ratio, and pixel rows and columns to make the process of character recognition quicker and more effective.

-Secondly we are working on a module of character recognition technology which will be capable of recognizing the background and foregrounds of the text image automatically. Therefore the system will be able to detect character pixels instantly and this would ensure greater accuracy of the printed text analyser. This module will be a part of the pre-processing stage of the current algorithm.

-Thirdly and most importantly we have theorized a module will would work for cursive and irregular fonts as well. The module will be able to define character boundaries even when the writing is joined. This module offers multiple challenges of inconsistency during the training and recognition process. We have performed a few preliminary tests on existing cursive fonts and we have seen some early success.

-Finally, we are designing a separate noise removal module which will be able to detect and remove existing noise from poor quality images. This is very important because we do not have the luxury of a high quality image to work with, every time. We are currently trying to work with various algorithms of median filter and looking for the best possible algorithm for our system. We are optimistic that with the following additions in the subsequent versions, we are going to design a system which would serve the purpose of recognizing most text images. As mentioned before, it has a marked advantage over other existing systems because it allows the user to interact with the system and introduce fonts and writing styles as and when he wants to do so.

Acknowledgment

We are grateful to the Department of Computer Science, St. Xavier's College, Kolkata, India for giving us the unique opportunity of working on this project. We wholeheartedly thank the 2012-14 batch of M.Sc. Computer Science for their encouragement and continued support.

References

[1] Nick Efford, "Digital Image Processing a Practical Introduction using Java"- Pearson Education.
[2] Herbert Schildt, "Java- The Complete Reference, 8th Edition", McGraw-Hill Companies.
[3] Gonzalez, Woods and Eddins, "Digital Image Processing Using Matlab", Gatesmark Publishing
[4] Mori S, Suen C Y and Yamamoto K,"Historical review of OCR research and development", Proceedings of IEEE 80,1029–1058,1992.
[5] J. Mantas,"An overview of character recognition methodologies",Pattern Recognition,Volume 19, Issue 6, Pages 425–430,1986.
[6] Ravina Mithe, Supriya Indalkar, Nilam Divekar, "Optical Character Recognition" International Journal of Recent Technology and Engineering (IJRTE) Volume 2 Issue 1.
[7] Sukhpreet Singh, "Optical Character Recognition Techniques: A Survey" Journal of Emerging Trends in Computing and Information Sciences, Vol. 4, No. 6 June 2013.
[8] Youssef Bassil, Mohammad Alwani, "OCR Post-Processing Error Correction Algorithm Using Google's Online Spelling Suggestion", Journal of Emerging Trends in Computing and Information Sciences, Vol. 3 No.1.

Satyaki Roy is a student of M.Sc. final year at St. Xavier's College Kolkata, India. He has a number of publications in bit and byte-level symmetric key cryptographic algorithms including one single authorship publication. His work Ultra Encryption Standard (UES) Version-I has been cited on several occasions. His research interests include networking, image processing, machine learning and genetic algorithms.

Ayan Chatterjee is currently his Master's degree from St. Xavier's College, Kolkata, India. During his bachelor's studies, he has worked in fields of Graphics Design and AI. His research interests include Image Processing, Network Security and Networking.

Rituparna Pandit is a post-graduation student at St. Xavier's College, Kolkata. In the past she has been involved in projects of web design, image processing etc. Her research interest includes Network Security and Microprocessors.

Kaushik Goswami is a Professor in Computer Science at St. Xavier's College, Kolkata, India. He has extensive knowledge in the fields of SQL and scripting languages. His research interests include Image Processing and Network Security. He has a number of publications in data encryption, randomization algorithms and green software engineering.