**INTEGRATED PROJECT REPORT**

**On**

**AUTOMATED ATTENDANCE SYSTEM**

Submitted in partial fulfillment of the requirement for the

Course Integrated Project III (CSP2208) of

**COMPUTER SCIENCE AND ENGINEERING**

**B.E. Batch-2015**

**In**

**May-2018**



**SUBMITTED TO -**

**ANKUL SINGH**

* **1510991086**

**ANGAD BACHHAL**

* **1510991073**

**GURSIMAR SINGH**

**- 1510991224**

**DEPARTMENT OF COMPUTER SCIENCES**

**CHITKARA UNIVERSITY**

**PUNJAB**

**CERTIFICATE**

This is to be certified that the project entitled “**AUTOMATIC ATTENDANCE SYSTEM** ” has been submitted for the Bachelor of Computer Science Engineering at Chitkara University, Punjab during the academic semester January 2018- May-2018 is a bonafide piece of project work carried out by “**ANKUL SINGH (1510991086), ANGAD BACHHAL (1510991073), GURSIMAR SINGH (1510991224)**” towards the partial fulfillment for the award of the course Integrated Project (Subject CODE) under the guidance of “**NITIN GOEL**” and supervision.

**Sign. of Guide** :

Name of Guide:- NITIN GOEL

(Designation & Department)

**CANDIDATE’S DECLARATION**

We, **ANKUL SINGH (1510991086), ANGAD BACHHAL (1510991073), GURSIMAR SINGH (1510991224) ,** B.E.-2056 of the Chitkara University, Punjab hereby declare that the Integrated Project Report entitled **“AUTOMATED ATTENDANCE SYSTEM”** is an original work and data provided in the study is authentic to the best of our knowledge. This report has not been submitted to any other Institute for the award of any other course.

|  |  |  |
| --- | --- | --- |
| **Sign. of Student 1** | **Sign. of Student 2** | **Sign. of Student 3** |
| ANKUL SINGH | ANGAD BACHHAL | GURSIMAR SINGH |
| 1510991086 | 1510991073 | 1510991224 |

**Place: Chitkara University**

**Date: 18 May 2018**

**ABSTRACT**

Every Educational Institutes need some kind of formatted attendance record. Here in our project we make work simpler for this institutes.

Transforming printed or handwritten documents directly to csv file.

In this application the user just need to scan the copy of attendance and rest of the thing is done by code. The scanned file is stored as Image file so this Image file undergoes processing and the useful data is extracted. This data is stored and CSV file is created to store the data , thus reducing the manual burden . Our project works for any kind of format, so this makes our project Dynamic.

Thus our project has to undergo various Image Processing task. This makes things simpler for the Educational Institutes.

**ACKNOWLEDGEMENT**

It is our pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and who influenced my thinking, behavior and acts during the course of study.

We express our sincere gratitude to all for providing me an opportunity to undergo Integrated Project as the part of the curriculum.

We are thankful to “NITIN GOEL” for his support, cooperation, and motivation provided to us during the training for constant inspiration, presence and blessings.

We also extend our sincere appreciation to ***“***who provided his valuable suggestions and precious time in accomplishing our Integrated project report.

Lastly, We would like to thank the almighty and our parents for their moral support and friends with whom we shared our day-to day experience and received lots of suggestions that improve our quality of work.

|  |  |  |
| --- | --- | --- |
| **Ankul Singh** | **Angad Bachhal** | **Gursimar Singh** |
| **1510991086** | **1510991073** | **1510991224** |
|  |  |  |

**INTRODUCTION**

**STATEMENT**

This Application Requirements Specification provides a complete description of all the functions and specifications of this automatic attendance system.This documentation presents a study of requirements where we can click a picture of documents and convert its data in the form of CSV file after processing thus reducing the manual burden of entering the data .

**BACKGROUND**

The system transformations printed and hand written text. The text could be on plain paper . The data design including normalization is manually done. The scanned document, which will be in the image format, will be the input. This scanned image undergoes pre-processing and data will stored in form of CSV File.

**Overview**

It provides an introduction to this document and also provides a full description of the project along with a detailed list of requirements for the user. It lists all the functions performed by the system.

**Application**

Application areas of this system are very large as in many organizations, for eg:

* E-Governance
* Government organizations
* Administrative Offices
* Off-line competitive examination management system

**Goal**

Goal of this project is to develop a system that focuses on following major criteria:

* Record keeping and archiving
* Efficient and automated record storage and indexing
* Increased accuracy
* Reduced Time
* Reduced Manual Burden
* To recognize Hand written character
* Make data available for long time without any loss.

**METHODS AND DESIGNED APPROACH**

**Project Description**

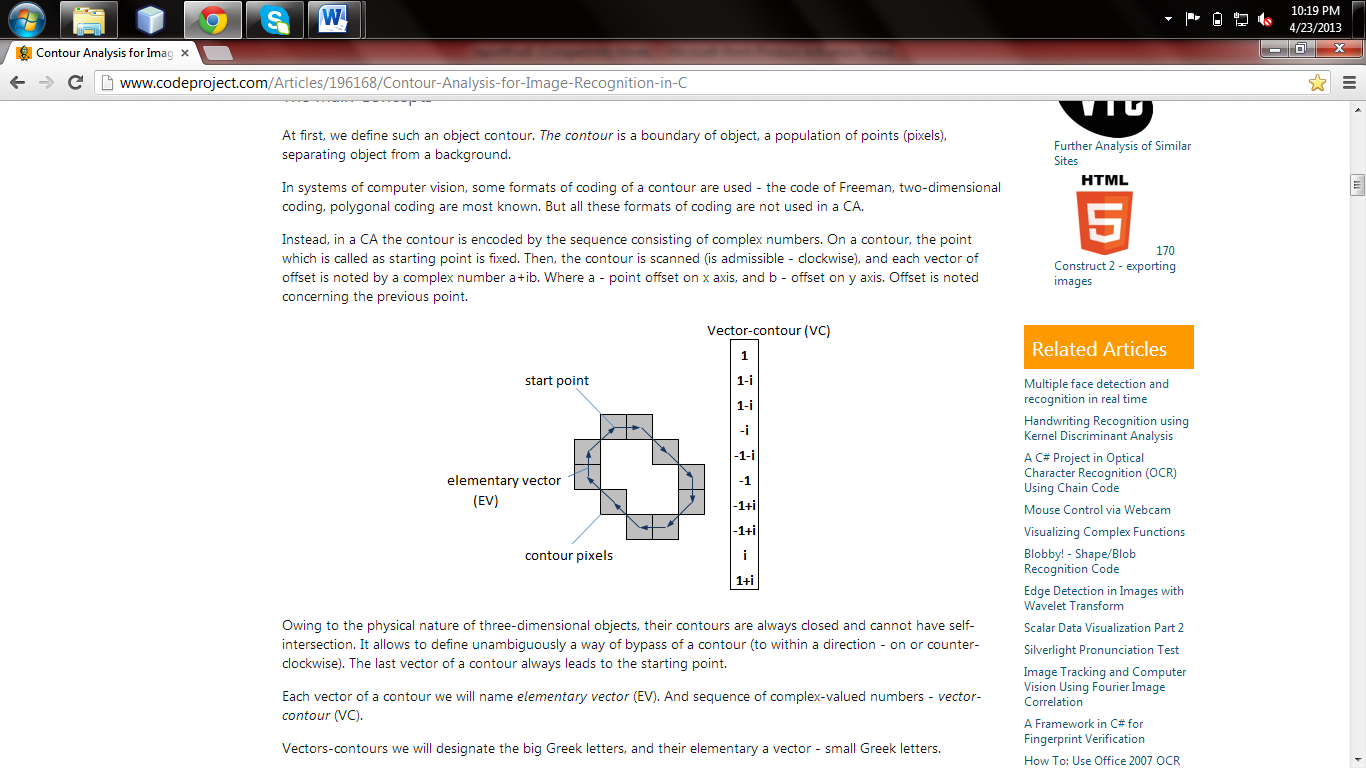
Tess4j OCR is a complex technology that converts images with text into editable formats. Tess4j OCR allows you to process scanned books, screenshots and photos with text and get editable documents like TXT, DOC or PDF files. This technology is widely used in many areas and the most advanced Tess4j OCR systems can handle almost all types of images, even such complex as scanned magazine pages with images and columns or photos from a mobile phone.

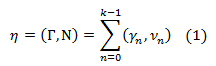
**2. Character Recognition**

Algorithms used :

1. Contour Analysis
2. Correlation Formula
3. **Contour Analysis**

* The Contour Analysis allows to describe, store, compare and find the objects presented in the form of the exterior outlines - contours.
* At first, we define such an object contour. *The contour* is a boundary of object, a population of points (pixels), separating object from a background.
* In systems of computer vision, some formats of coding of a contour are used - the code of Freeman, two-dimensional coding, polygonal coding are most known. But all these formats of coding are not used in a CA.
* Instead, in a CA the contour is encoded by the sequence consisting of complex numbers. On a contour, the point which is called as starting point is fixed. Then, the contour is scanned (is admissible - clockwise), and each vector of offset is noted by a complex number a+ib. Where a - point offset on x axis, and b - offset on y axis. Offset is noted concerning the previous point.



* + Owing to the physical nature of three-dimensional objects, their contours are always closed and cannot have self-intersection. The last vector of a contour always leads to the starting point.
  + Each vector of a contour we will name *elementary vector* (EV). And sequence of complex-valued numbers - *vector-contour* (VC).
  + Vectors-contours we will designate the big Greek letters, and their elementary a vector - small Greek letters.
  + Thus, vector-contour Γ of length k can be designated as:
  + http://www.codeproject.com/KB/graphics/ContourAnalysis/f0.png
  + As scalar product of contours, Γ and N are called such complex number:
  + 
  + Where k - dimensionality of a VC, γn - n the elementary vector of contour Γ, νn - n EV of contour N. (γn, νn) - the scalar product of complex numbers calculated as:

http://www.codeproject.com/KB/graphics/ContourAnalysis/f2.png

* + The scalar product of usual vectors and scalar product of complex numbers –
  + If we multiplied an EV as simple a vector, their scalar product would look so:

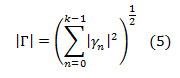
http://www.codeproject.com/KB/graphics/ContourAnalysis/f3.png

Compare this formula to the formula (2) and you note that:

* Outcome of scalar product of vectors is the real number. And outcome of product of complex numbers - a complex number.
* The real part of scalar product of complex numbers coincides with scalar product of appropriate vectors. That is complex product includes vectorial scalar product.
* Let's introduce one more concept - the normalized scalar product (NSP):

http://www.codeproject.com/KB/graphics/ContourAnalysis/f4.png

* Where |Γ| and |N| - the norms (length) of contours calculated as:



* The NSP in space of complex numbers, also is a complex number.
* Thus, unity is greatest possible value of norm of NSP (it follows from a Cauchy–Bunyakovsky–Schwarz inequality: |ab| <= |a||b|), and it is reached only if...

http://www.codeproject.com/KB/graphics/ContourAnalysis/f6.png

* ...where μ - the arbitrary complex number.
* Let's introduce the concept of *intercorrelation function* (ICF) of two contours:
* http://www.codeproject.com/KB/graphics/ContourAnalysis/f7.png
* Where N(m) - a contour received from N by cycle shift by its EV on m of elements.
* For an example, if N = (n1, n2, n3, n4), N(1) = (n2, n3, n4, n1), N(2) = (n3, n4, n1, n2) and so on.
* Values of this function show contours Γ and N are how much similar if to shift starting point N on m positions.
* Let's discover the magnitude having the maximum norm among values an ICF:

http://www.codeproject.com/KB/graphics/ContourAnalysis/f8.png

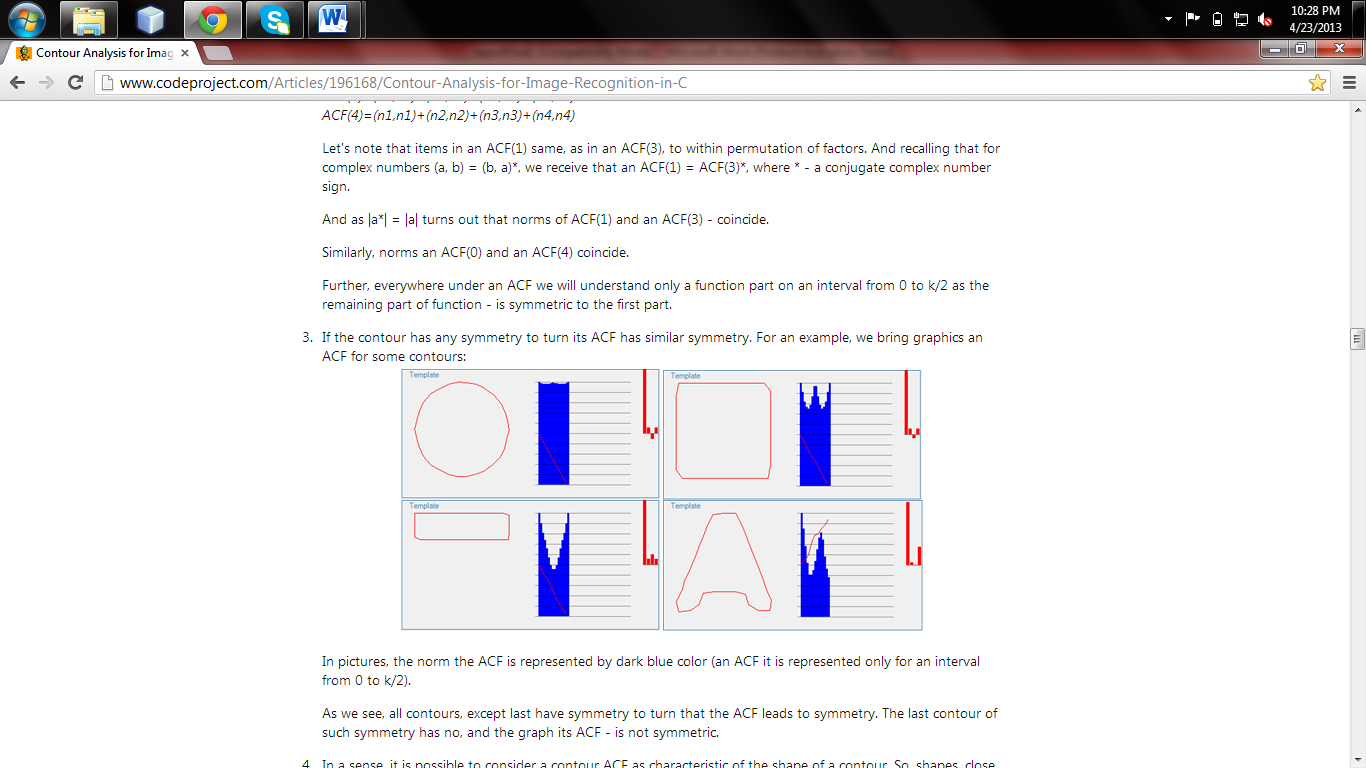
* From determinations a NSP and an ICF, it is clear that τmax is a measure of similarity of two contours, invariant to transposition, scaling, rotation and starting point shift.
* Thus, the norm |τmax| shows a level of similarity of contours, and reaches unity for identical contours, and the argument arg(τmax) gives an angle of rotation of one contour, concerning another.
* Let's introduce one more concept - *an autocorrelation function* (ACF). The Autocorrelation function is an ICF for which N=Γ. As a matter of fact is a scalar product of a contour most on itself at various shifts of starting point:

http://www.codeproject.com/KB/graphics/ContourAnalysis/f9.png

* For this purpose, we take the image a size n\*n pixels. Then breed its uniform grid with a step s. The total length of all grid lines is:

http://www.codeproject.com/KB/graphics/ContourAnalysis/f10.png

* It turns out that passage from the plane two-dimensional image to contours does not reduce dimensionality of the task. We as before work in complexity O(n2).



**3. Symbol image detection**

The process of image analysis to detect character symbols by examining pixels is the core part of input set preparation in both the training and testing phase. Symbolic extents are recognized out of an input image file based on the color value of individual pixels, which for the limits of this project is assumed to be either black **RGB (255,0,0,0)** or white **RGB (255,255,255,255)**. The input images are assumed to be in bitmap form of any resolution which can be mapped to an internal bitmap object in the Microsoft Visual Studio environment. The procedure also assumes the input image is composed of only characters and any other type of bounding object like a boarder line is not taken into consideration.

The procedure for analyzing images to detect characters is listed in the following algorithms:

**i. Determining character lines**

Enumeration of character lines in a character image (‘page’) is essential in delimiting the bounds within which the detection can proceed. Thus detecting the next character in an image does not necessarily involve scanning the whole image all over again.

**Algorithm**:

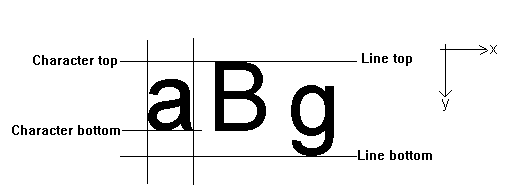
1. start at the first x and first y pixel of the image pixel(0,0), Set number of lines to 0
2. scan up to the width of the image on the same y-component of the image
   1. if a black pixel is detected register y as top of the first line
   2. if not continue to the next pixel.
   3. if no black pixel found up to the width increment y and reset x to scan the next horizontal line
3. start at the top of the line found and first x-component pixel(0,line\_top)
4. scan up to the width of the image on the same y-component of the image
   1. if no black pixel is detected register y-1 as bottom of the first line. Increment number of lines
   2. if a black pixel is detected increment y and reset x to scan the next horizontal line
5. start below the bottom of the last line found and repeat steps 1-4 to detect subsequent lines
6. If bottom of image (image height) is reached stop.

**ii. Detecting Individual symbols**

Detection of individual symbols involves scanning character lines for orthogonally separable images composed of black pixels.

**Algorithm**:

1. start at the first character line top and first x-component
2. scan up to image width on the same y-component
   1. if black pixel is detected register y as top of the first line
   2. if not continue to the next pixel
3. start at the top of the character found and first x-component, pixel(0,character\_top)
4. scan up to the line bottom on the same x-component
   1. if black pixel found register x as the left of the symbol
   2. if not continue to the next pixel
   3. if no black pixels are found increment x and reset y to scan the next vertical line
5. start at the left of the symbol found and top of the current line, pixel(character\_left, line\_top)
6. scan up to the width of the image on the same x-component
   1. if no black characters are found register x-1 as right of the symbol
   2. if a black pixel is found increment x and reset y to scan the next vertical line
7. start at the bottom of the current line and left of the symbol, pixel(character\_left,line\_bottom)
8. scan up to the right of the character on the same y-component
   1. if a black pixel is found register y as the bottom of the character
   2. if no black pixels are found decrement y and reset x to scan the next vertical line



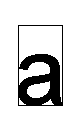
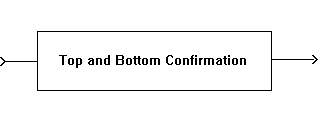
**Fig 3. Line and Character boundary detection**

From the procedure followed and the above figure it is obvious that the detected character bound might not be the actual bound for the character in question. This is an issue that arises with the height and bottom alignment irregularity that exists with printed alphabetic symbols. Thus a line top does not necessarily mean top of all characters and a line bottom might not mean bottom of all characters as well.

Hence a confirmation of top and bottom for the character is needed.

An optional confirmation algorithm implemented in the project is:

1. start at the top of the current line and left of the character
2. scan up to the right of the character
   1. if a black pixels is detected register y as the confirmed top
   2. if not continue to the next pixel
   3. if no black pixels are found increment y and reset x to scan the next horizontal line



**Fig 4. Confirmation of Character boundaries**

**3. Symbol Image Matrix Mapping**

The next step is to map the symbol image into a corresponding two dimensional binary matrix. An important issue to consider here will be deciding the size of the matrix. If all the pixels of the symbol are mapped into the matrix, one would definitely be able to acquire all the distinguishing pixel features of the symbol and minimize overlap with other symbols. However this strategy would imply maintaining and processing a very large matrix (up to 1500 elements for a 100x150 pixel image). Hence a reasonable tradeoff is needed in order to minimize processing time which will not significantly affect the separability of the patterns. The project employed a sampling strategy which would map the symbol image into a 10x15 binary matrix with only 150 elements. Since the height and width of individual images vary, an adaptive sampling algorithm was implemented. The algorithm is listed below:

**Algorithm:**

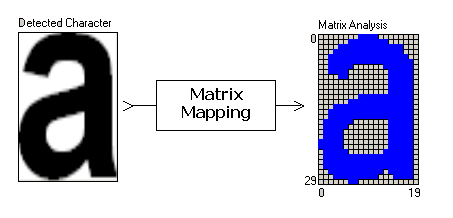
a. For the width (initially 20 elements wide)

1. Map the first (0,y) and last (width,y) pixel components directly to the first (0,y) and last (20,y) elements of the matrix
2. Map the middle pixel component (width/2,y) to the 10th matrix element
3. subdivide further divisions and map accordingly to the matrix

b. For the height (initially 30 elements high)

1. Map the first x,(0) and last (x,height) pixel components directly to the first (x,0) and last (x,30) elements of the matrix
2. Map the middle pixel component (x,height/2) to the 15th matrix element
3. subdivide further divisions and map accordingly to the matrix

c. Further reduce the matrix to 10x15 by sampling by a factor of 2 on both the width and the height



**Fig. 5 Mapping symbol images onto a binary matrix**

In order to be able to feed the matrix data to the network (which is of a single dimension) the matrix must first be linearized to a single dimension. This is accomplished with a simple routine with the following algorithm:

1. start with the first matrix element (0,0)
2. increment x keeping y constant up to the matrix width
   1. map each element to an element of a linear array (increment array index)
   2. if matrix width is reached reset x, increment y
3. repeat up to the matrix height (x,y)=(width, height)

Hence the linear array is our input vector for the MLP Network. In a training phase all such symbols from the trainer set image file are mapped into their own linear array and as a whole constitute an input space. The trainer set would also contain a file of character strings that directly correspond to the input symbol images to

serve as the desired output of the training. A sample mini trainer set is shown below:

**Algorithm:**

* load image file
* analysis of image by the code for character lines
* for each character line detect consecutive character symbols
  + analyze and process symbol image to map into an input vector
  + feed input vector to network and compute output
  + convert the Unicode binary output to the corresponding character and render to a text box

**Flowchart:**

**RESULTS**

**1. Specifications**

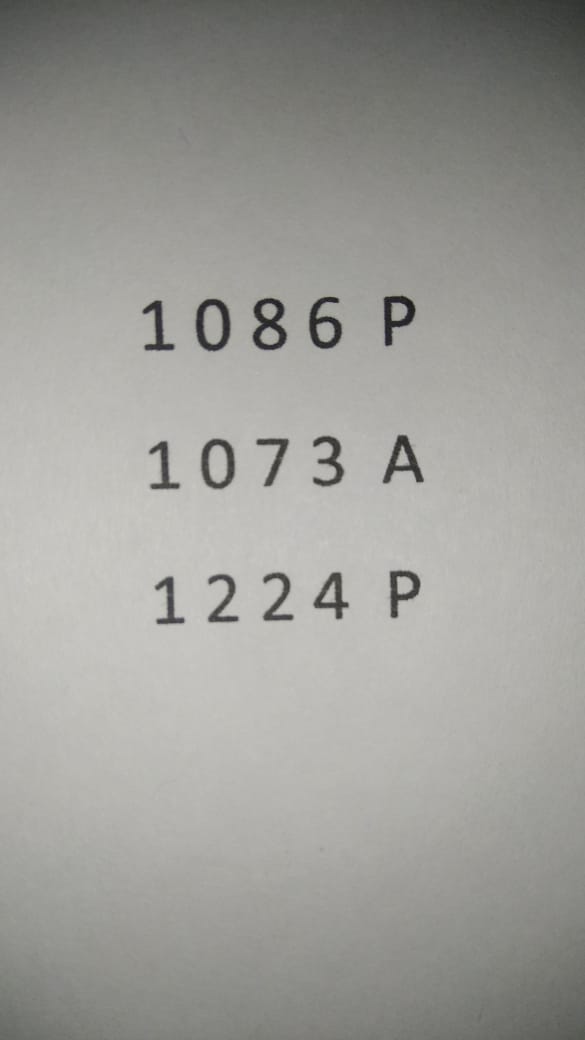
**Input Specification**

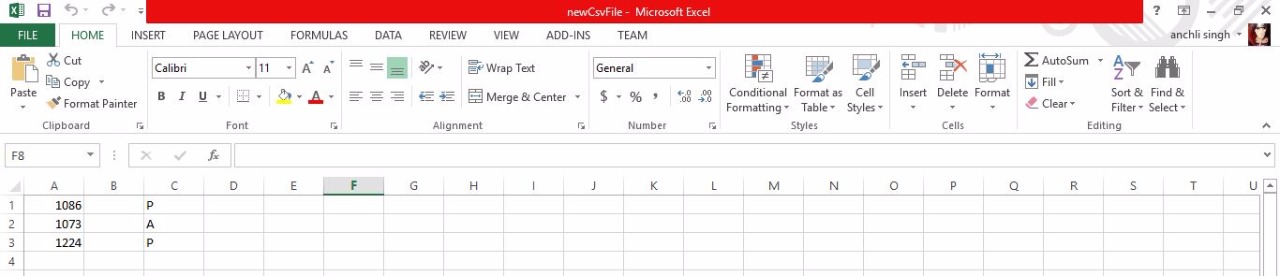
**1. Scanned document image file format supported :** jpg and jpeg.

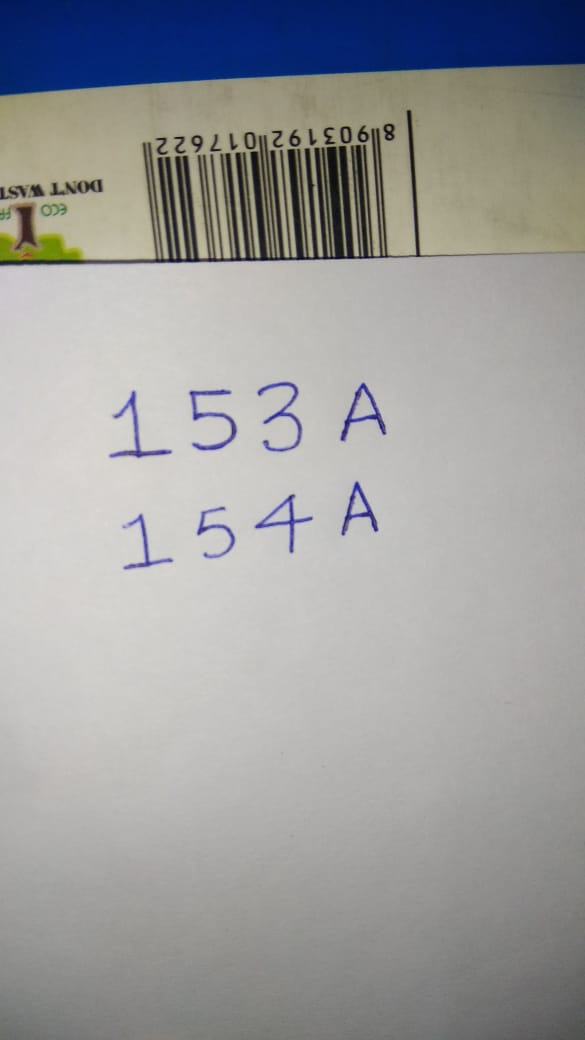
**2. Input image scanned in portrait mode.**

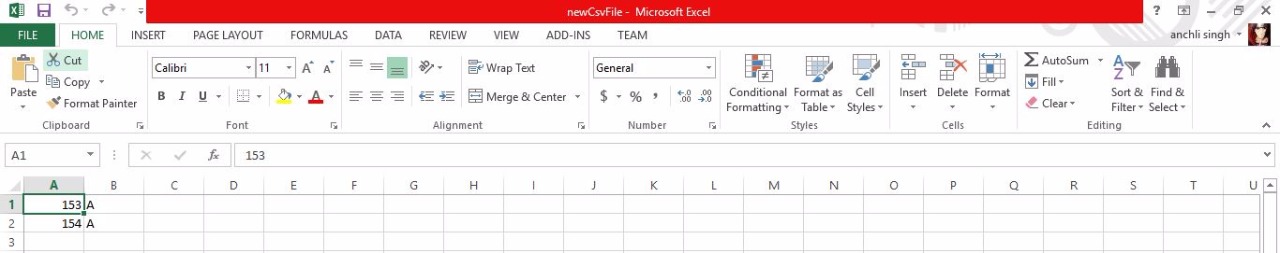
**Output Specification**

1. Tess4j OCR processed output
2. CSV file generated.

****

****

****

****

**4.1.3 Functional Specifications**

The proposed TESS4J OCR system specifications, as per the common conclusive decision by the members of the project group are listed below :

**Operations**

According to our project, we have performed various steps for storing printed documents into CSV File.

**CONCLUSION**

Our Project is on Transforming printed documents to CSV File. So printed document is given and that image of documents is converted to text form according to user requirement. Thus our project looked upon the problems faced by Educational Institutes, for storing data which is in Printed format. It helped in reducing manual burden and is also less time consuming. This application can also be use for any kind of format of printed documents as our application provides dynamic definition of templates. And according to the templates added by the admin user the data is processed by the application and gives the appropriate answer. Thus not only Educational Institutes are benefited by this but also the government offices or any other organizations using such printed documents are benefited.

We looked upon various algorithms and techniques for pre-processing and character recognition from a image and implemented most optimal ones amongst them, thus resulting in more speed and accuracy.

This makes our project dynamic and is feasible for any kind of organization.

We have successfully completed our project.

**Scope Of Future Enhancement**

This project can be further extended for recognizing handwritten documents. This application can be further upgraded in which functionality can be added to train handwriting of a particular individual and then can be used to recognize documents written by that individual. Also application can be trained to recognize handwriting of multiple individuals and also different fonts. There is also scope of increasing accuracy of the recognizer so that no manual watch should be needed on the application other than inputting the data. Thus application can be automized to a higher level.

**REFRENCES**

**WEBSITE:-**

**Sourceforge.net**

**Stackoverflow.com**

**Geeksforgeeks.com**