CHAPTER 1

1. INTRODUCTION

1.1 PROJECT OVERVIEW

Optical Character Recognition (OCR) has been in quite a few research works in the last decade and it is getting much more attention day by day. Optical character recognition (OCR) is the conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, or from subtitle text superimposed on an image. The goal of (OCR) is to classify optical patterns (often contained in a digital image) corresponding to alphanumeric or other characters. The process of OCR involves several steps including segmentation, feature extraction, grey scaling, edge detection, binarization, template loading and matching and classification. OCR is a part of image processing.

Image processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image. Usually image processing system includes treating images as two dimensional images while applying already known images processing methods to them. It is among rapidly growing technologies today, with its application in various aspects of business. Image processing forms core research areas within engineering and computer science disciplines too

In this project, we apply OCR technique for translation of handwritten Tulu characters into Kannada editable character. By using image preprocessing technique, creating a database of Kannada character templates and mapping with the handwritten Tulu character, and features are extracted from the above image.

These features are matched with the templates.

1.2 PURPOSE OF PROJECT

The sole purpose of project is to develop software that can efficiently recognize handwritten Tulu character and generate output in Kannada character. Our main emphasis is on feature extraction part where, we do grey scaling, binarization and extracting the feature and compares the extracted feature values with stored database characters, and which gives the highest correlation with database character is taken as a recognized character and gives the output.

1.3 SCOPE OF PROJECT

The primary scope of project is to provide optimal character recognition with less computational time and more accuracy. By less computational time we mean using small size database which would reduce the time required for the matching and recognition.

1.4 PROBLEM STATEMENT

Tulu Kannada characters involve lot of conjunction points which makes difficult to identify when they are handwritten. Moreover translation of Tulu Kannada characters to classical Kannada is one of the least explored areas.

1.5 EXISTING SYSTEM

Automatic recognition of handwritten characters from scanned images helps to convert characters in an image into convenient editable and readable form. Tulu is a south Indian Dravidian language with rich set of handwritten patterns. The recognition of handwritten Tulu characters is based on the AdaBoost algorithm using Haar-like [1] features. Finally, recognized characters are mapped into an equivalent editable document of Kannada characters.

This system is used recognize the Tulu characters and reintroduce it to next generation by mapping it to equivalent editable Kannada characters. Currently Tulu script is not used to write the Tulu language, as it uses the Kannada script for

documentation, is the reason of mapping to Kannada. It is the first proposal focused on Tulu handwriting recognition, essentially helpful for giving new view into historical Tulu documents.

The Haar-like features based AdaBoost algorithm for recognition is proposed by taking into consideration of the different features of Tulu script.

1.5.1 DISADVANTAGES OF EXISTING SYSTEM

- As mentioned in the conclusion of the existing system proposal it's one of the least explored areas and has been tested on fewer samples which make this topic open for more improvement.
- The approach used here is machine learning which requires large sample size.
- There is no quantitative evidence given in paper about the accuracy of the proposed system.

1.6 PROPOSED SYSTEM

The first step in our proposed system involves creation of a database for Tulu Kannada characters. Each character is unique in many ways and we can extract unique features of the individual character and store. In this method input handwritten Tulu characters are processed and translated into Kannada characters by matching with the input Tulu Kannada database samples. We are using Image processing technique of MATLAB for character recognition. Correlation function is used to match handwritten Tulu character with input Kannada character.

In the proposed system the user captures the live image by the external device or uses the stored image. After that, the document processing system with OCR usually starts the process with the scanning. This is the step by which the images are digitalized and stored in a file.

Image processing steps involved are grey scaling, edge detection, binarization.

Grey scaling converts the true color image RGB to the greyscale intensity image.

rgb2gray is the inbuilt function used for grey scaling, which involves conversion of 24bit image (rgb) to 8bit image (grey scaled).

Edge detection involves finding out the boundaries of objects within images it works by detecting discontinuities in brightness. Edge detection is used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision.

Binarization converts the greyscale image to binary image (black and white), by replacing all pixels in the input image with luminance greater than threshold with the value 1 (white) and replacing all other pixels with the value 0 (black). By using the optical character recognition algorithm we extract the information from the template data and store in data base storage.

Template matching is used to the scanned handwritten character sample, the extracted information is compared with the template from the database storage by cross correlation algorithm we obtain the correlation coefficient with which the template is matched. After template matching it recognizes the character and gives the output.

1.6.1 OBJECTIVES

- The proposed system should recognize and translate Tulu Kannada characters which would be handwritten samples.
- The translated Kannada characters should be in digitized format (computer editable).
- Test the accuracy of the character recognition with the cross correlation algorithm.

1.6.2 ASSUMPTIONS

- The input handwritten character should be on a white paper with any color pen.
- The character need not be aligned along horizontal axis (X-axis) and may be tilted but aligned along (Y-axis).
- The character shouldn't be half written and it must be written completely.
- The ambiance is considered to be ideal for the proposed system to produce correct output.

1.6.3 CONSTRAINTS

- The proposed system may process and recognize only a single character.
- Due to the nature of Tulu Kannada characters there are some minute anomalies between characters.

Ex:

In these cases we may expect to show the character to have matched most.

- The character input is expected to be continuous and not separate, in case which it would be treated as a separate character.

1.6.4 SUMMARY

The proposed system works as expected, which is to recognize and translate the Tulu Kannada characters to classical Kannada characters using cross correlation.

Since there is only one technique used so far in this area which lacks the quantitative measurement of accuracy, proposed system fulfills that by providing substantially better accuracy.

Below is the sample input and output screenshot of the system.

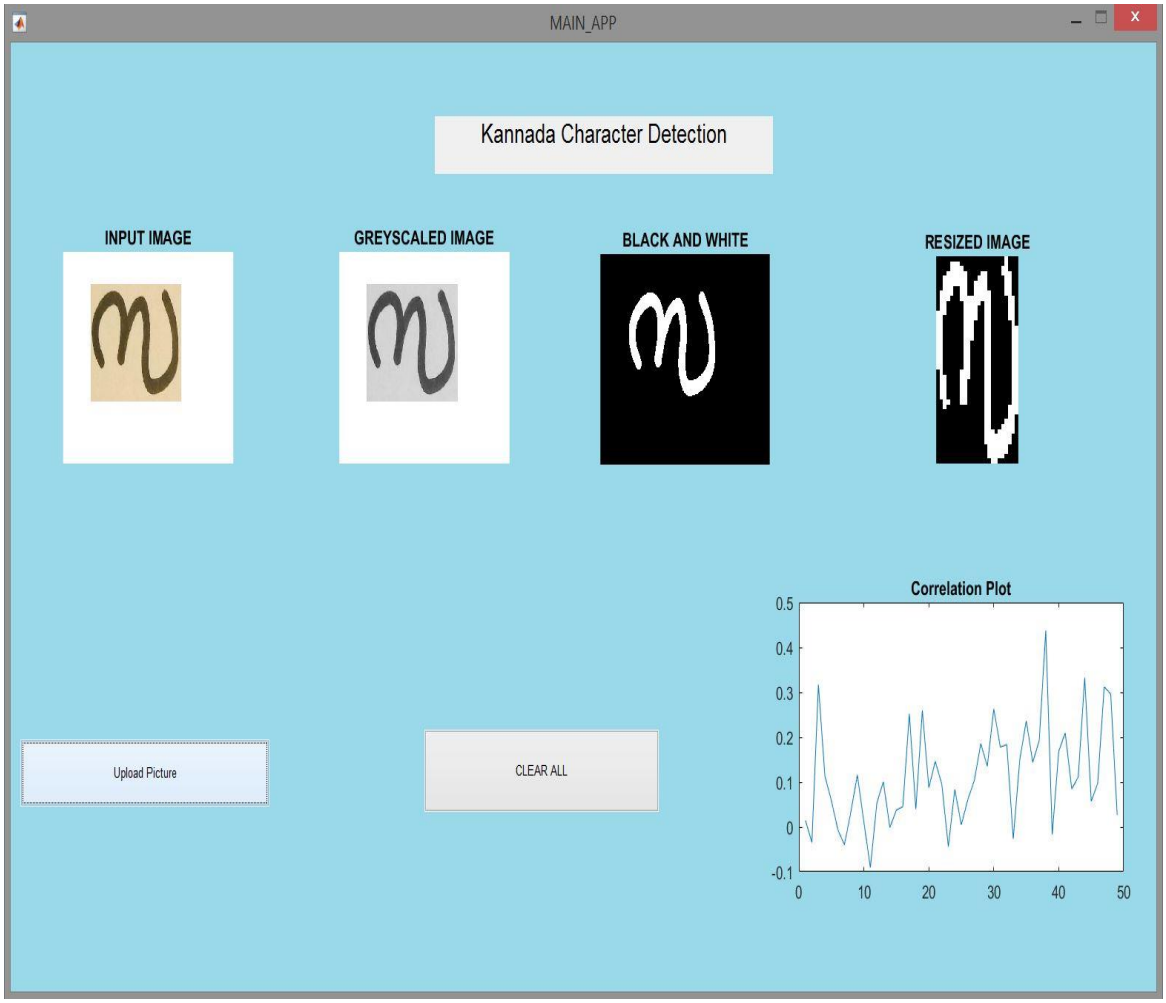


Figure 1 Input

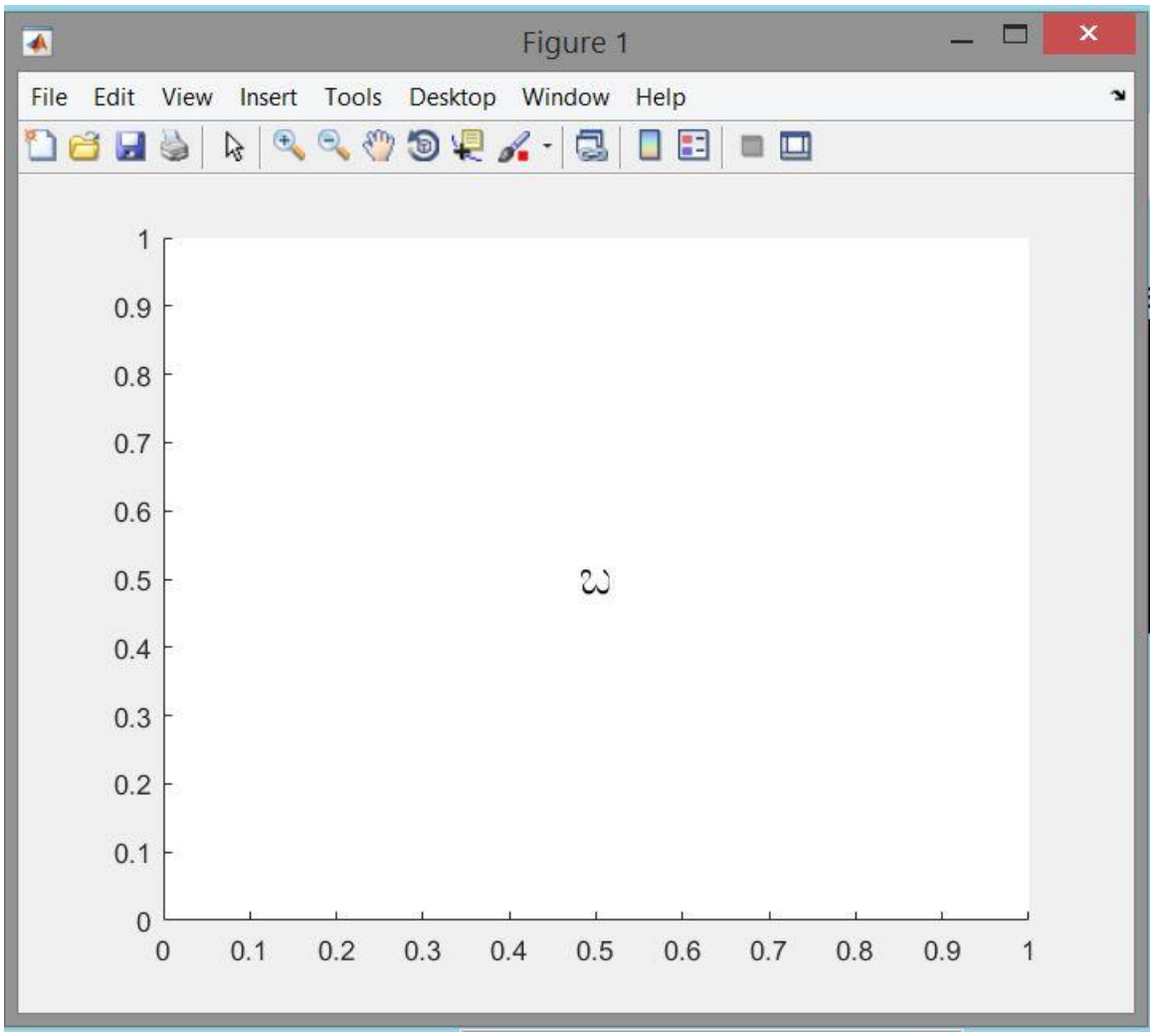


Figure 2 Output

CHAPTER 2

2. LITERATURE SURVEY

2.1 OVERVIEW

Literature survey is carried out to get all the related information of current project, which is used to get an idea for the enhancement as well as changes that can be made to improve existing approaches. A literature survey is done on various OCR techniques such as Freeman chain code, Persian handwritten objects and numerals database (PHOND), Scale-invariant feature transform descriptor (SITF), Histogram of gradients (HOG) and Local binary patterns (LBP) descriptor.

2.2 RELATED PAPERS

This section surveys relevant work related to this project.

Following are the works carried out in this field.

Fewer researches have been done on character recognition of South Indian scripts like Tulu script. One and the only such work is the Haar features based Adaboost algorithm for recognition was proposed by taking into consideration of the different features of Tulu script. Efficient binarization by using an adaptive contrast map and a combination of Otsu thresholding with canny edge detection is presented in this paper. Otsu thresholding is not sufficient as degradation is more [1]. Hassan Althobaiti and Chao Lu discussed optical character recognition (OCR) of Arabic characters, which is very difficult due to its cursive writing style. In conclusion the proposed method successfully recognizes printed Arabic characters but also mentions that the proposed Arabic OCR systems cannot fully detect Arabic characters. The freeman chain code is a very useful descriptor since it provides some crucial information that can be used to measure the curviness of a character [2].

Saleem Pasha discusses in his papers of the known works for Kannada handwritten character recognition which are very few, so this paper proposes one of the efficient methods to recognize handwritten Kannada characters. The proposed method is experimented on 4800 images of handwritten Kannada characters and obtained an average accuracy of 91.00% [3].

The paper of the field of Persian handwritten OCR is less explored and the writing style of for Persian sentences, dates, words its right to left and the other way for numerals. In conclusion the modified framing feature was successful for Persian handwritten character recognition (accuracy up to 99%) and PHOND database was a new standard in Persian language [4].

N Prameela, P Anjushal's paper introduces a practical handwritten OCR system for Telugu language which one of script extracted from Dravidian language. The proposed system was shape and font dependent but was successful in classifying the Telugu characters. To achieve a good recognition system for handwritten format it is quite still challenging because of variation in writing styles, shapes and orientation. There are various methods have been proposed for feature extraction and feature matching so far for different Indian languages. Compression of different recognition free techniques has been discussed here. We found that SIFT descriptor, Histogram of gradients (HOG) and Local binary patterns (LBP) descriptor, and shape descriptor performs well for various Indian scripts like Devanagari, Bengali, Telugu, etc. Since there is less work reported for other Indian scripts like Gujarati, Dogri, Gurumukhi, Kannada, etc [5].

CHAPTER 3

3. SYSTEM ANALYSIS AND DESIGN

3.1 SYSTEM DEVELOPMENT CYCLE

3.1.1 WATERFALL MODEL

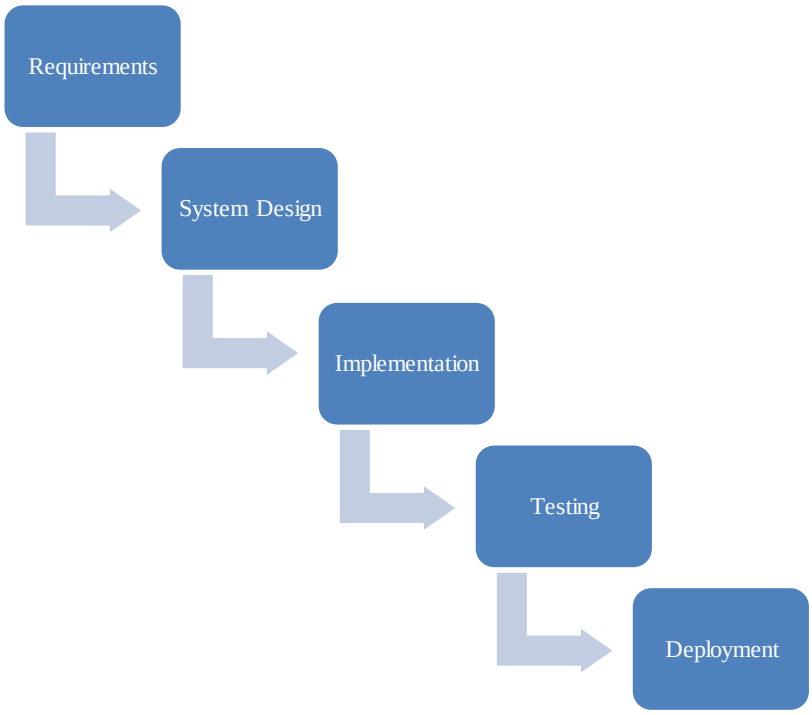


Figure 3 Waterfall Model

This section discusses the development process cycle.

- Waterfall model was used because the requirements of the project will remain constant.
- Since project is to be developed in short span of time this model is suitable for short time span of application development.
- This model requires one phase to be accomplished to proceed to next phase.

- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Process and results are well documented.

1. REQUIREMENTS

- Recognize and translate Tulu Kannada characters to classical Kannada characters.
- To serve as a character translator (Tulu Kannada to classical Kannada).
- Tulu Kannada character as input.
- Classical Kannada character as output in digitized format.

2. SYSTEM DESIGN

- Image processing modules like grey scaling, edge detection, binarization.
- Graphic user interface.
- Original character of Tulu Kannada sample templates.

3. IMPLEMENTATION

- Image pre-processing steps [grey scaling, edge detection, binarization] of MATLAB.
- Resizing the handwritten character in MATLAB.
- Feature extraction from handwritten character.
- Load template in the database.
- Text matching with the template.

4. Testing

- Testing of different handwritten Tulu samples.

5. Deployment

- Demonstrating the translation of Tulu Kannada characters to classical Kannada characters.

3.2 SYSTEM FLOWCHART

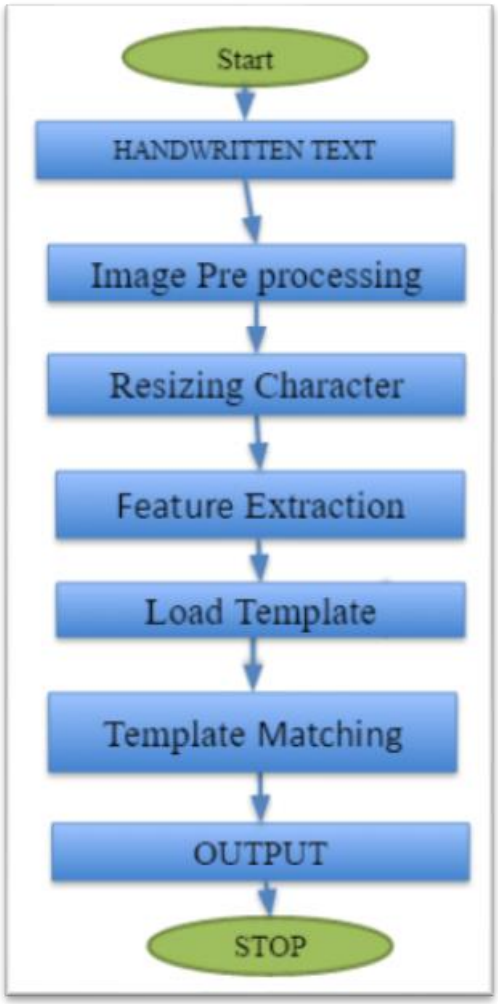


Figure 4 System flow chart.

- In the system flow chart, the process is been divided into blocks.
- Each block contains a task that is needed to be processed.
- The Tulu handwritten text is scanned and pre-processed.
- After the image pre-processing, the character is been resized to the required format.
- In the next step the feature is extracted from the image and is loaded as the template in the storage.
- In the next step, by the cross-correlation method, the template matching is done and produces the output to the user and the processes get stopped.

3.2.1 HANDWRITTEN TEXT

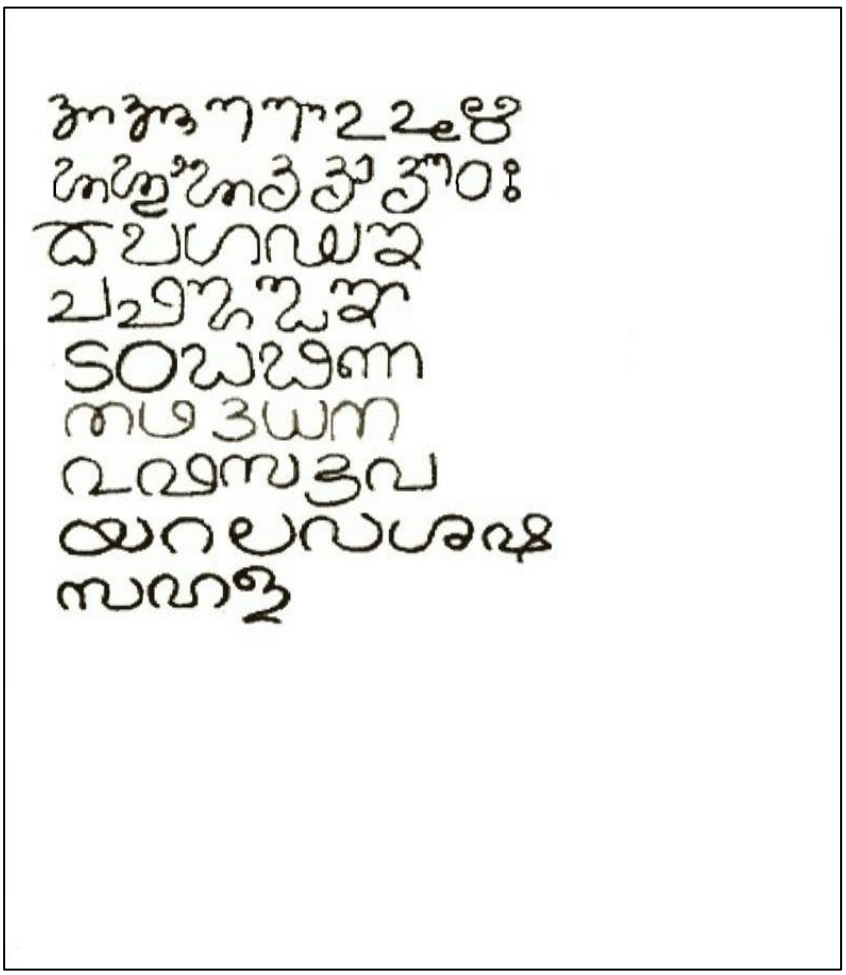


Figure 5 Handwritten text.

- The handwritten text is written on the white background paper.
- It is taken by a scanning the image by external device or use the stored image.

3.2.2 IMAGE PREPROCESSING

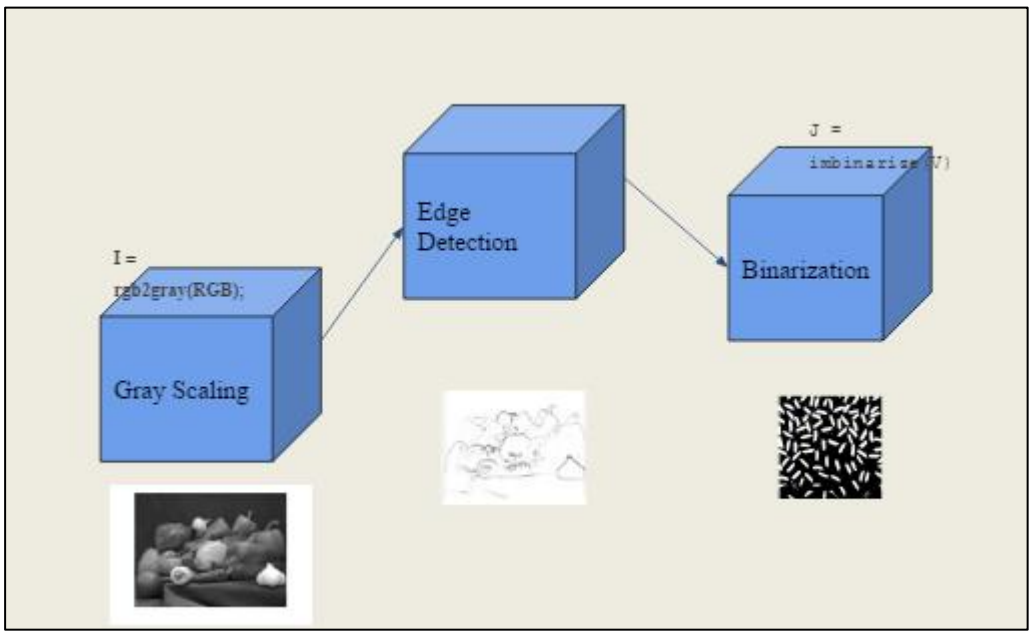


Figure 6 Image preprocessing.

- In image pre-processing we do grey scaling, by removing the RGB colors from the image.

  I = rgb2gray (RGB)

- The next step is edge detection finds out the boundaries of objects within images it works by detecting discontinuities in brightness. Edge detection is used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision.

  BW = edge (I,'Canny');

- Binarization converts the greyscale image to binary image (Black & White), by replacing all pixels in the input image with luminance greater than level with the value 1 (white) and replacing all other pixels with the value 0 (black).

  BW = im2bw (I, level)

- This range is relative to the signal levels possible for the image's class. Therefore, a level value of 0.5 corresponds to an intensity value halfway between the minimum and maximum value of the class.

3.2.3 RESIZING CHARACTER

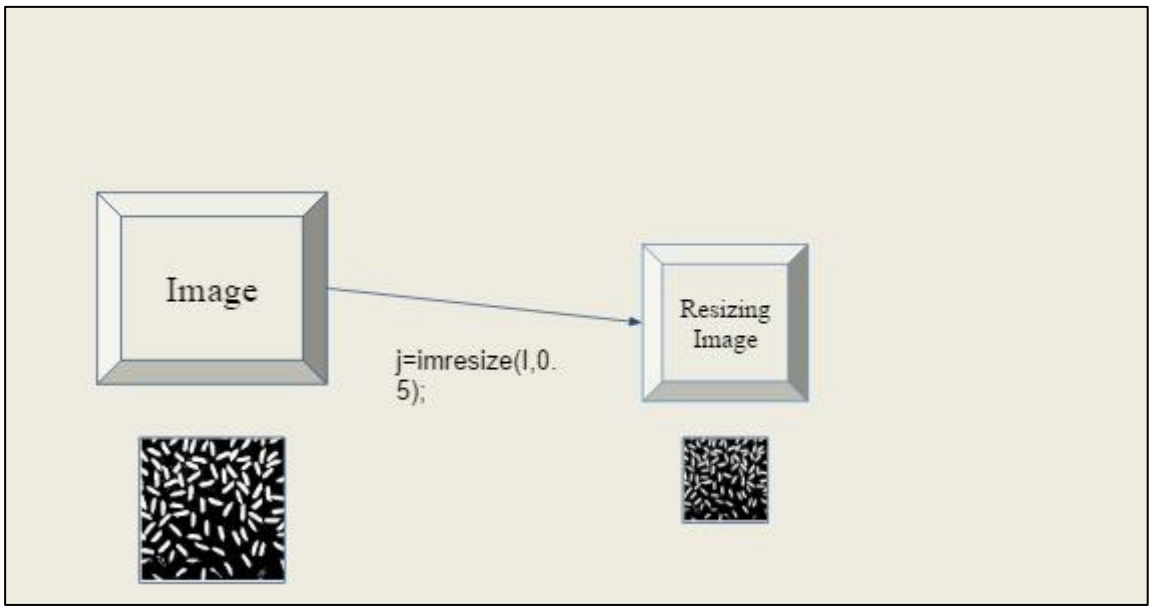


Figure 7 Resizing Character.

- In resizing the image, MATLAB uses B = imresize (A, scale) which returns image B that is scale times the size of A. The input image A can be a greyscale, RGB, or binary image. If A has more than two dimensions, imresize only resizes the first two dimensions. If scale is in the range [0, 1], B is smaller than A. If scale is greater than 1, B is larger than A. By default, imresize uses bicubic interpolation.
  B= Imresize (A, scale);

3.2.4 LOADING TEMPLATE

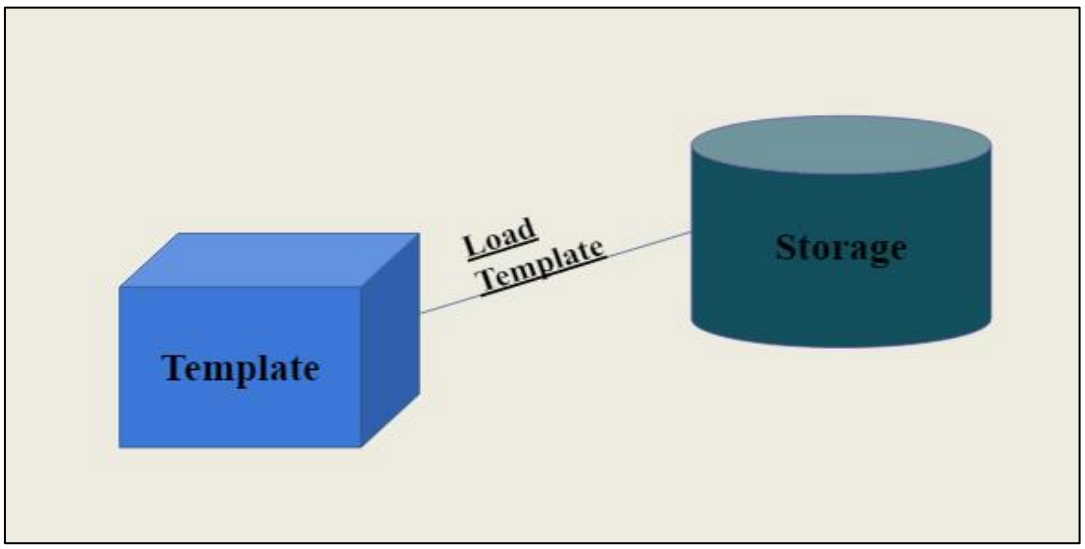


Figure 8 Loading template.

- The template of each and every Tulu and Kannada character in created and loaded the template in the data base.
- This storage will serve as a database.
- This database is used for matching with the input samples for matching.

3.2.5 TEMPLATE MATCHING

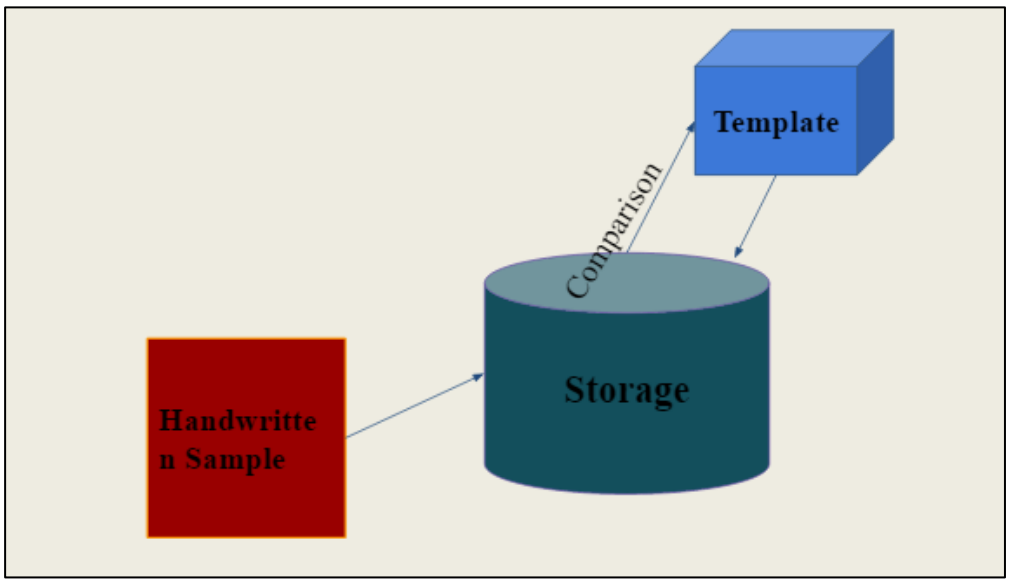


Figure 9 Template loading.

- In template matching, the scanned handwritten sample is been compared with each and every Tulu character template in data base storage.

- The technique used is cross-correlation, which is done by the comparison of the graph of each and every template with the given sample and mapped with the Kannada editable character (template).

- Based upon the comparison of correlation coefficients between the template and the sample it allots the Kannada character and gives the output.

3.3 DATA FLOW DIAGRAM

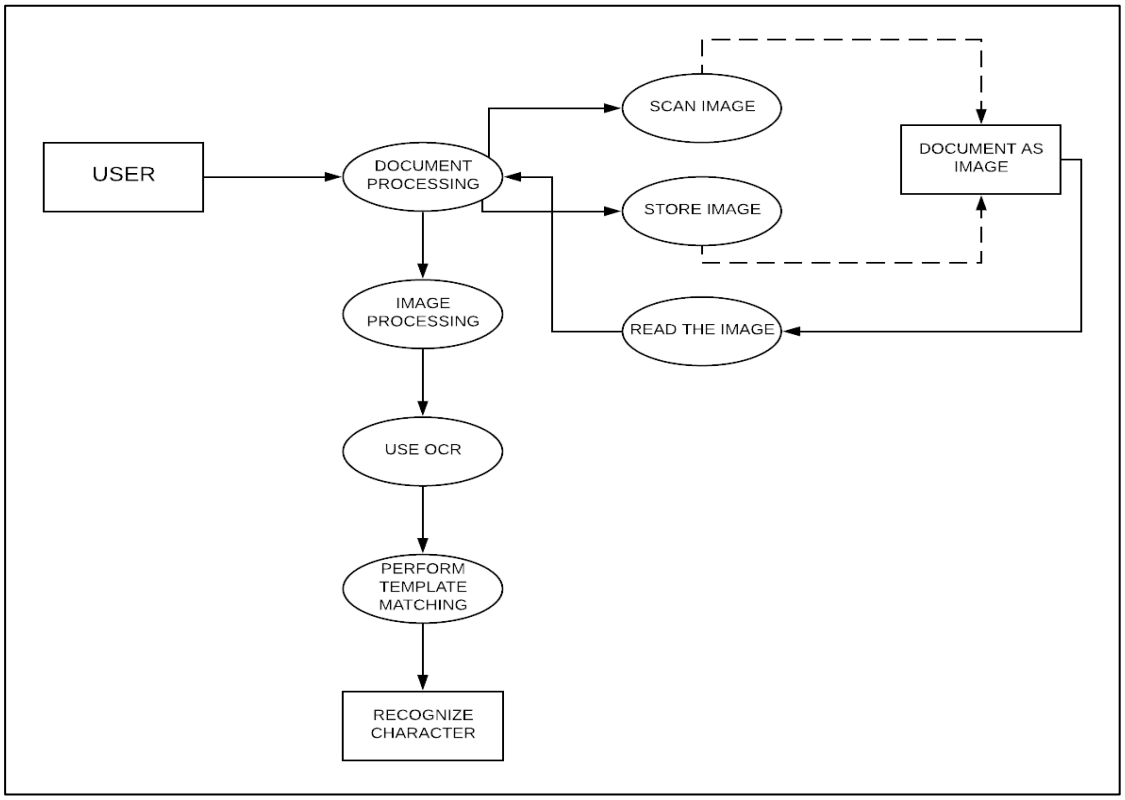


Figure 10 Data flow diagram.

USER: - The user captures a live image by the external device or uses the stored image.

DOCUMENT PROCESSING: -The document processing system with OCR usually starts with scanning. This is the step by which the images are digitalized and stored in a file.

IMAGE PROCESSING:-In image processing the steps involved are grey scaling, edge detection, binarization.

GREYSCALING: - Grey scaling converts the true color image RGB to the greyscale intensity image. rgb2gray is the inbuilt function used for grey scaling, which involves conversion of 24bit image (rgb) to 8bit image (grey scaled).

EDGE DETECTION: - Edge detection, finds out the boundaries of objects within images, by detecting discontinuities in brightness. Edge detection is used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision.

BINARIZATION:-Binarization converts the greyscale image to binary image (Black & White), by replacing all pixels in the input image with luminance greater than certain threshold with the value 1 (white) and replacing all other pixels with the value 0 (black).

OCR: -By using the optical character recognition algorithm, we extract the information from the template data and store in database.

CHARACTER RECOGNITION: - In template matching the scanned handwritten character sample is extracted and this information is compared with the template from the database by cross correlation techniques and which gives the nearest correlation coefficient of the sample. After template matching it recognizes the character and gives the output.

CHAPTER 4

4. SYSTEM IMPLEMENTATION

4.1 BRIEF ALGORITHM FLOW CHART

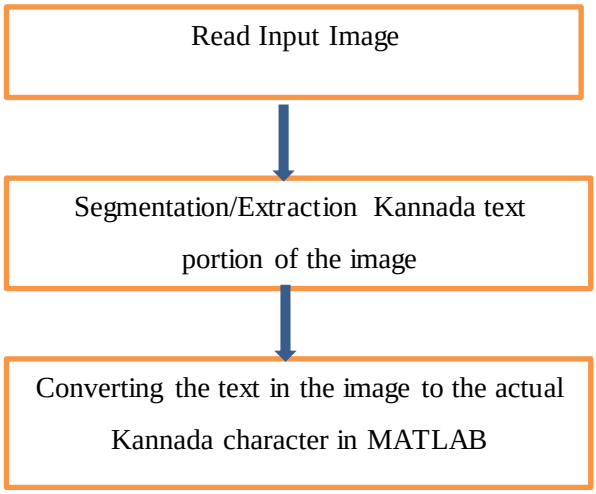


Figure 11 Algorithm flow chart.

- Step1

  Reading an input image, if the image is a RGB image (i.e. 3 dimensional image) convert it to black and white (i.e. 2 dimensional image).

  Following algorithms are involved in step1:-

  1. RGB to grey scale conversion.
  2. Image preprocessing.

- Step2

  Extraction/segmentation of image portion for the text using threshold detection/filtering algorithm.

  1. Threshold detection
  2. Binary conversion
  3. OCR

- Step3

  Converting the text-image label to actual text

  Steps used are:-

  1. Data base formation
  2. Cross correlation

4.1.1 DETAILED CODE FLOW WITH ALGORITHM EXPLAINATION

1. DATABASE FORMATION

➢ RGB to Grey Scale Conversion

A greyscale digital image is an image in which the value of each pixel is a single sample, that is, it carries only intensity information. Images of this sort, also known as black-and-white, are composed exclusively of shades of grey, varying from black at the weakest intensity to white at the strongest.

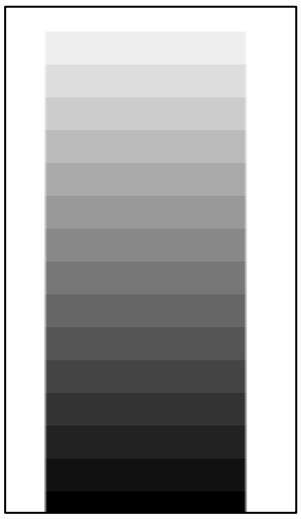


Figure 12 Grayscale Intensity (0-1).

Greyscale images are often the result of measuring the intensity of light at each pixel in a single band of the electromagnetic spectrum (e.g. infrared, visible light,

ultraviolet, etc.), and in such cases they are monochromatic proper when only a given frequency is captured.

So, in this project we are using the built-in function rgb2gray () which does the job of converting RGB image to grayscale image.

Algorithm /loop for converting RGB-Grey

```
for k1 = 1 : row
    for k2 =1 : col
     % Assume you have an RGB image of class double, or create a random one
grey(k1,k2) = 0.29 * rgb(k1,k2,1) + 0.59 * rgb(k1,k2,2) + 0.11 * rgb(k1,k2,3);
    end
end
```

- **Thresholding**

Thresholding is the simplest method of image segmentation. From a greyscale image, thresholding can be used to create binary images. It computes a global threshold (LEVEL) that can be used to convert an intensity of image to a binary image with IM2BW. LEVEL is a normalized intensity value that lies in the range [0, 1].It indicates the effectiveness of thresholding of the input image and it is in the range [0, 1].

The input image can be uint8, uint16, int16, single, or double, and it must be non-sparse.

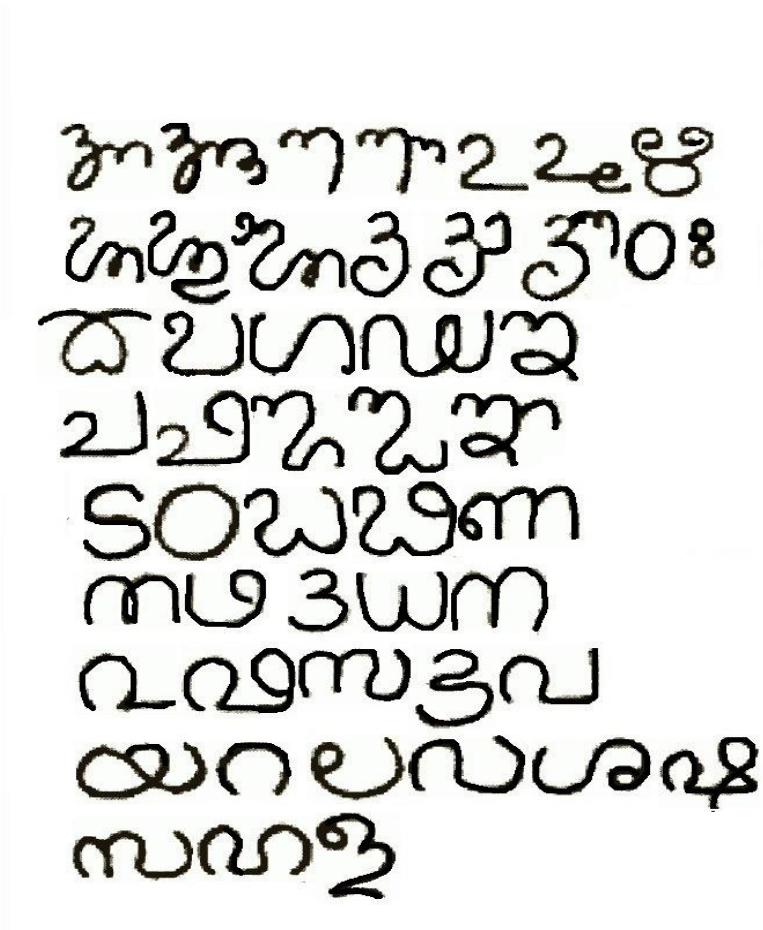


Figure 13 Greyscale Image.

➢ Binary Conversion

- The image is converted to binary depending on the threshold level.
- Binary image will have only local values i.e. True or False i.e. '1' or '0' for a pixel.
- Each pixel value is compared with the threshold and a decision is made to convert.

- **Removal of small connected components**

Removes from a binary image all connected components (objects) that have fewer pixels, producing another binary image.

- **Clipping character from the line**

a) Find the connected component in the binary (logical) Image.

b) Find its location in the image

c) Extract that portion

It extracts the entire text and then separates the connected components.

Each connected component is extracted in the similar manner for the complete image.

This character is saved in a matrix.

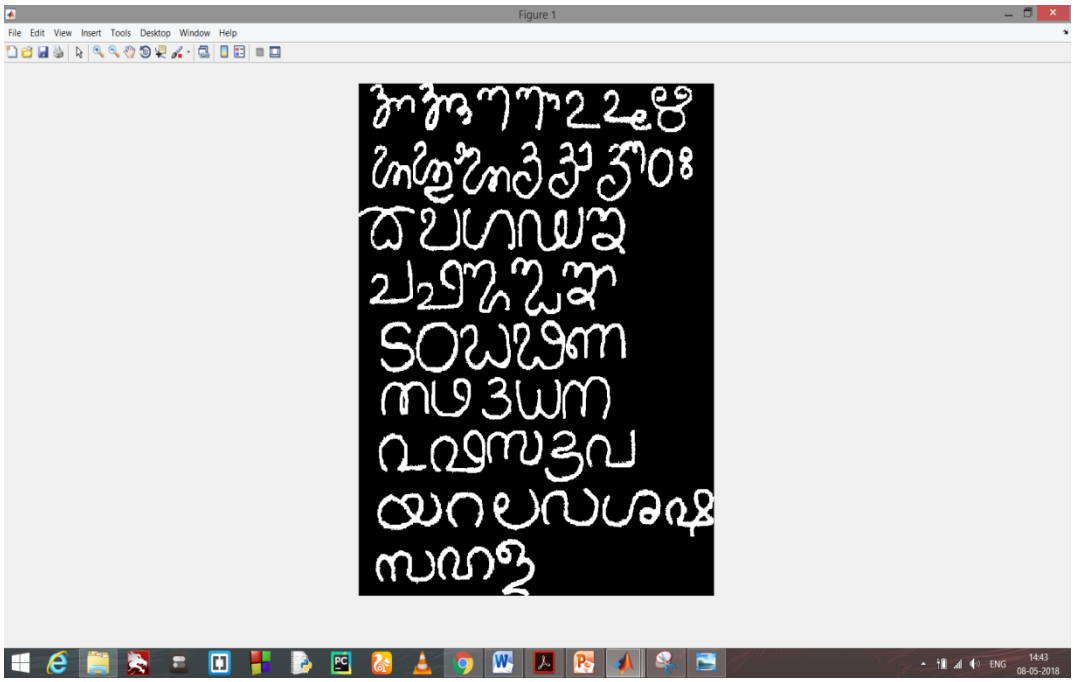


Figure 14 Binarized image from the greyscale image.

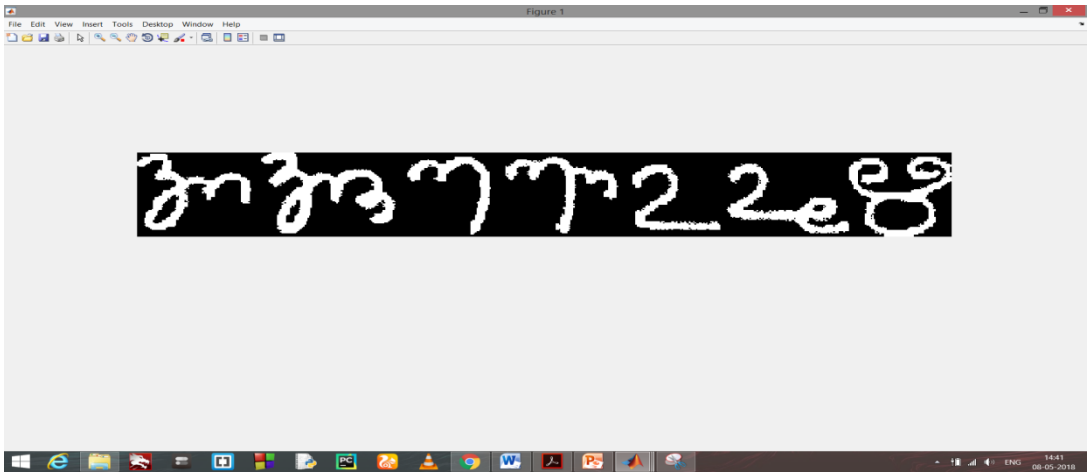


Figure 15 Line clipped from the binarized image.

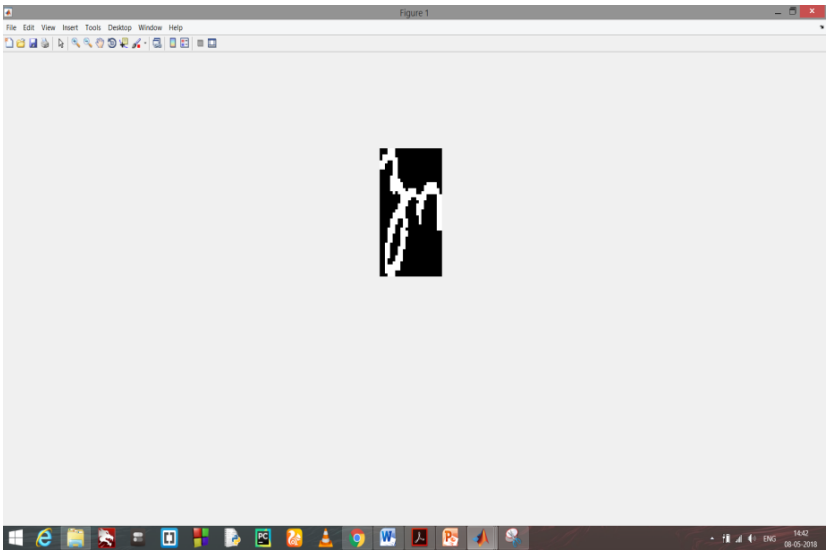


Figure 16 Extracted character from the clipped binarized image.

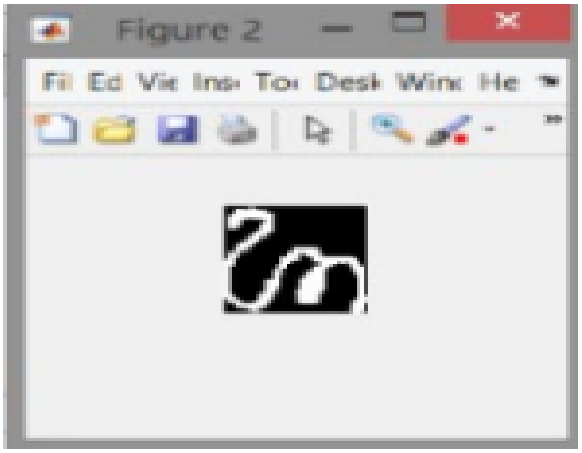


Figure 17 Image size will be resized to the standard data base size i.e. 42x24.

➢ Storing of characters to database

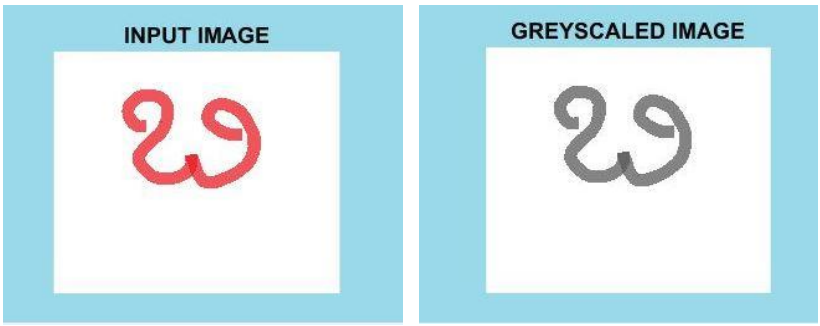
- Once the image is extracted it is compared with the standard database.
- The database stores template in the standard size i.e. (42x24).

2. CROSS CORRELATION

Steps involved in this process are:

1. RGB to Greyscale conversion

This step is similar to the step used for creation of database which involves the conversion of input rgb image (3d) to greyscale image (2d).



2. Binarization:

This is the second step in this process where the grey scaled image is converted to binary image i.e. is 2-bit image.

3. Resizing:

This is the third step in this process where the binarized image is resized into 42*24cm format.

4. Template matching using correlation algorithm (corr2):

This is the fourth step, in this process the handwritten Tulu character is matched with the Tulu Kannada input character sample and converted to Kannada characters in digital format.

Cross-correlation is a measure of similarity of two series as a function of the language of one relative to the other. This is also known as a sliding dot product or sliding inner-product. It is commonly used for searching a long signal for a shorter, known feature. It has applications in pattern recognition, single particle analysis, electron tomography, averaging, cryptanalysis, and neurophysiology.

The correlation coefficient is a number representing the similarity between 2 images in relation with their respective pixel intensity.

Here A and B are the images you are comparing, whereas the subscript indices m and n refer to the pixel location in the image. Basically what MATLAB does is to compute, for every pixel location in both images, the difference between the intensity value at that pixel and the mean intensity of the whole image, denoted as a letter with a straight line over it.

corr2 computes the correlation coefficient using

$$r = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{\left(\sum_m \sum_n \left(A_{mn} - \bar{A}\right)^2\right)\left(\sum_m \sum_n \left(B_{mn} - \bar{B}\right)^2\right)}}$$

where $\bar{A}$ = mean2(A), and $\bar{B}$ = mean2(B).

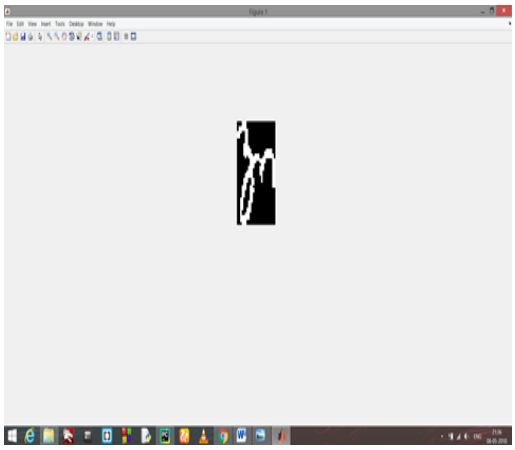


Fig. A) Standard image

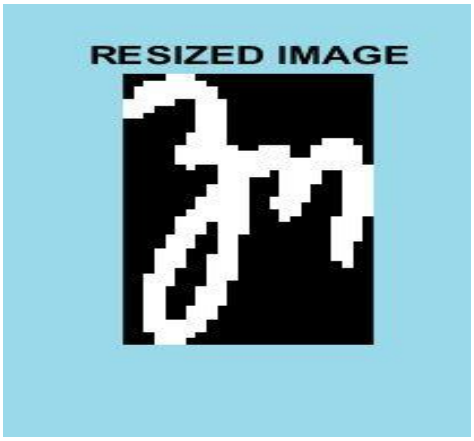


Fig. B) Resized input image

CHAPTER 5

5. CODE SNIPPETS

1. RGB to Greyscale conversion

```
% Convert to gray scale
if size(imagen,3)==3 %RGB image
    imagen=rgb2gray(imagen);
end
```

2. Binarization

```
% Convert to BW
threshold = graythresh(imagen);

imagen =~im2bw(imagen,threshold);
```

3. Resizing

```
% Resize letter (same size of template)
img_r=imresize(n1,[42 24]);
```

4. Database creation

Editor - C:\Users\RAHUL\Documents\MATLAB\Ocr_TULU_KANNADA_Final\create_data_base.m

MAIN_APP.m | create_data_base.m | lines.m

```
1
2 -     warning off %Off all warnings
3       % Clear all
4 -     clc;
5 -     close all;
6 -     clear all;
7
8       %Creating data base for small letters
9
10 -    originalFolder = pwd;
11 -    [baseFileName, folder] = uigetfile('*.*', 'Specify an image file');
12 -    fullImageFileName = fullfile(folder, baseFileName);
13
14      % Read in image into an array.
15 -    [imagen storedColorMap] = imread(fullImageFileName);
16
17
18      % Show image
19 -    imshow(imagen);
20 -    title('INPUT IMAGE')
21
22      % Convert to gray scale
23 -    if size(imagen,3)==3 %RGB image
24 -        imagen=rgb2gray(imagen);
25 -    end
26      % Convert to BW
27 -    threshold = graythresh(imagen);
28
29 -    imagen =~im2bw(imagen,threshold);
```

Editor - C:\Users\RAHUL\Documents\MATLAB\Ocr_TULU_KANNADA_Final\create_data_base.m

MAIN_APP.m | create_data_base.m | lines.m

```
35 -    word=[ ];
36 -    re=imagen;
37
38 -     count=1;
39 -     index=1;
40      %  template_kann=ones(42,24,10);
41 -     template_kann = false(42, 24,1);
42
43 -    while 1
44          %Fcn 'lines' separate lines in text
45          [fl ,re]=lines(re);
46 -        imgn=fl;
47
48 -         [L ,Ne] = bwlabel(imgn);
49
50 -         for n=1:Ne
51 -             [r,c] = find(L==n);
52              % Extract letter
53 -             n1=imgn(min(r):max(r),min(c):max(c));
54              % Resize letter (same size of template)
55 -             img_r=imresize(n1,[42 24]);
56 -             template_kann(:,:,count)=img_r;
57 -    count=count+1;
58 -         end
59
60 -         if isempty(re)  %See variable 're' in Fcn 'lines'
61 -             break
62 -         end
63 -    end
```

5. OCR logic

Editor - C:\Users\RAHUL\Documents\MATLAB\Ocr_TULU_KANNADA_Final\main_func.m

MAIN_APP.m | main_func.m

```
function handles= main_func(handles)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here

originalFolder = pwd;
        [baseFileName, folder] = uigetfile('*.*', 'Specify an image file');
        fullImageFileName = fullfile(folder, baseFileName);

            % Read in image into an array.
    [imagen storedColorMap] = imread(fullImageFileName);


% Show image
axes(handles.axes1)
imshow(imagen);
title('INPUT IMAGE')

% Convert to gray scale
if size(imagen,3)==3 %RGB image
    imagen=rgb2gray(imagen);
end
axes(handles.axes7)
imshow(imagen);
title('GREYSCALED IMAGE')
% Convert to BW

threshold = graythresh(imagen);

imagen =~im2bw(imagen,threshold);
```

Editor - C:\Users\RAHUL\Documents\MATLAB\Ocr_TULU_KANNADA_Final\main_func.m

MAIN_APP.m | main_func.m

```
% Remove all object containing fewer than 30 pixels
imagen = bwareaopen(imagen,30);
axes(handles.axes8)
imshow(imagen);
title('BLACK AND WHITE')

%Storage matrix word from image
word=[ ];
re=imagen;

% Load templates

load template_kann
global template_kann

num_letras=size(template_kann,3);
num_plate=[];
while 1
    %Fcn 'lines' separate lines in text
    [fl re]=lines(re);
    imgn=fl;

    [L Ne] = bwlabel(imgn);
    for n=1:Ne
        [r,c] = find(L==n);
        % Extract letter
        n1=imgn(min(r):max(r),min(c):max(c));
        % Resize letter (same size of template)
        img_r=imresize(n1,[42 24]);
```

6. Character recognition and displaying output

Editor - C:\Users\RAHUL\Documents\MATLAB\Ocr_TULU_KANNADA_Final\main_func.m

MAIN_APP.m | main_func.m

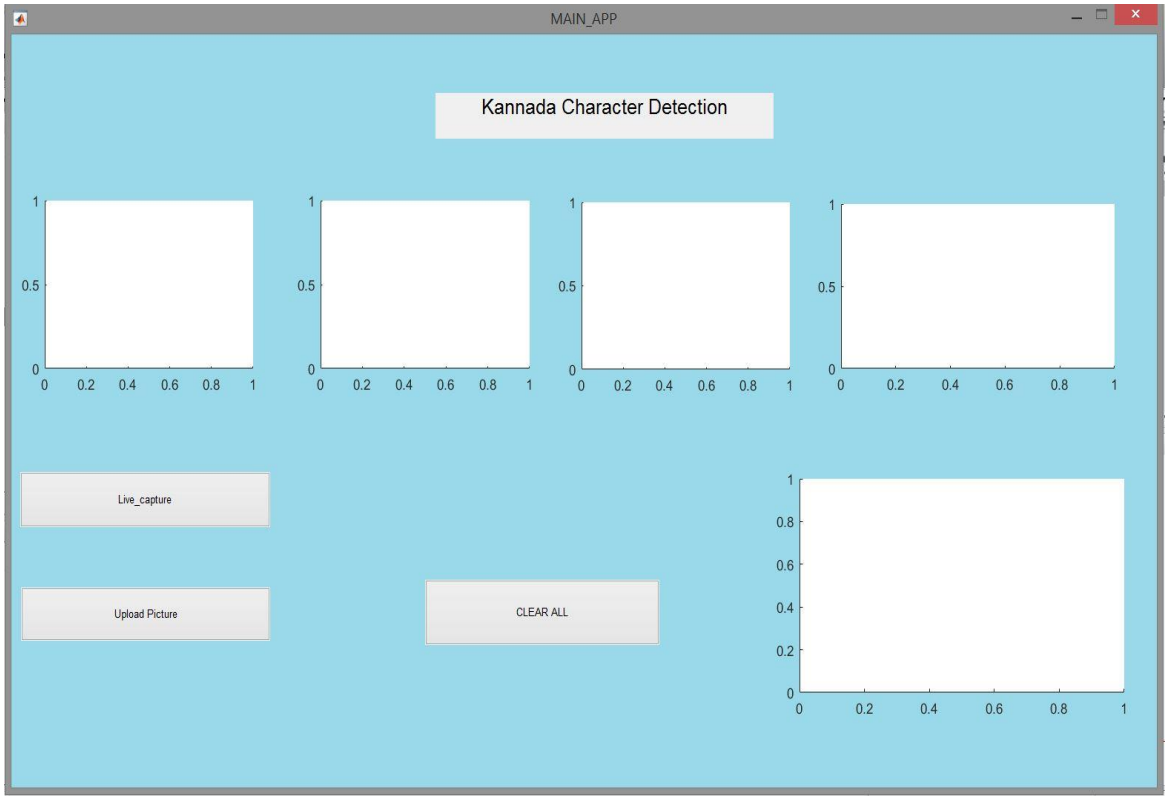
```
% Remove all object containing fewer than 30 pixels
imagen = bwareaopen(imagen,30);
axes(handles.axes8)
imshow(imagen);
title('BLACK AND WHITE')

%Storage matrix word from image
word=[ ];
re=imagen;

% Load templates

load template_kann
global template_kann

num_letras=size(template_kann,3);
num_plate=[];
while 1
    %Fcn 'lines' separate lines in text
    [fl re]=lines(re);
    imgn=fl;

    [L Ne] = bwlabel(imgn);
    for n=1:Ne
        [r,c] = find(L==n);
        % Extract letter
        n1=imgn(min(r):max(r),min(c):max(c));
        % Resize letter (same size of template)
        img_r=imresize(n1,[42 24]);
```

Editor - C:\Users\RAHUL\Documents\MATLAB\Ocr_TULU_KANNADA_Final\read_letter_db.m

MAIN_APP.m | read_letter_db.m

```
if vd==1
    figure,
    c = char(3205);
    text(.5,.5,c,'HorizontalAlignment','center','FontName','Arial Unicode MS','FontSize',18)
elseif vd==2
    figure,
    c = char(3206);
    text(.5,.5,c,'HorizontalAlignment','center','FontName','Arial Unicode MS','FontSize',18)
elseif vd==3
    figure,
    c = char(3207);
    text(.5,.5,c,'HorizontalAlignment','center','FontName','Arial Unicode MS','FontSize',18)
elseif vd==4
    figure,
    c = char(3208);
    text(.5,.5,c,'HorizontalAlignment','center','FontName','Arial Unicode MS','FontSize',18)
elseif vd==5
    figure,
    c = char(3209);
    text(.5,.5,c,'HorizontalAlignment','center','FontName','Arial Unicode MS','FontSize',18)
elseif vd==6
    figure,
    c = char(3210);
    text(.5,.5,c,'HorizontalAlignment','center','FontName','Arial Unicode MS','FontSize',18)
elseif vd==7
    figure,
    c = char(3211);
    text(.5,.5,c,'HorizontalAlignment','center','FontName','Arial Unicode MS','FontSize',18)
```

7. Graphical user interface



8. Standard Tulu characters

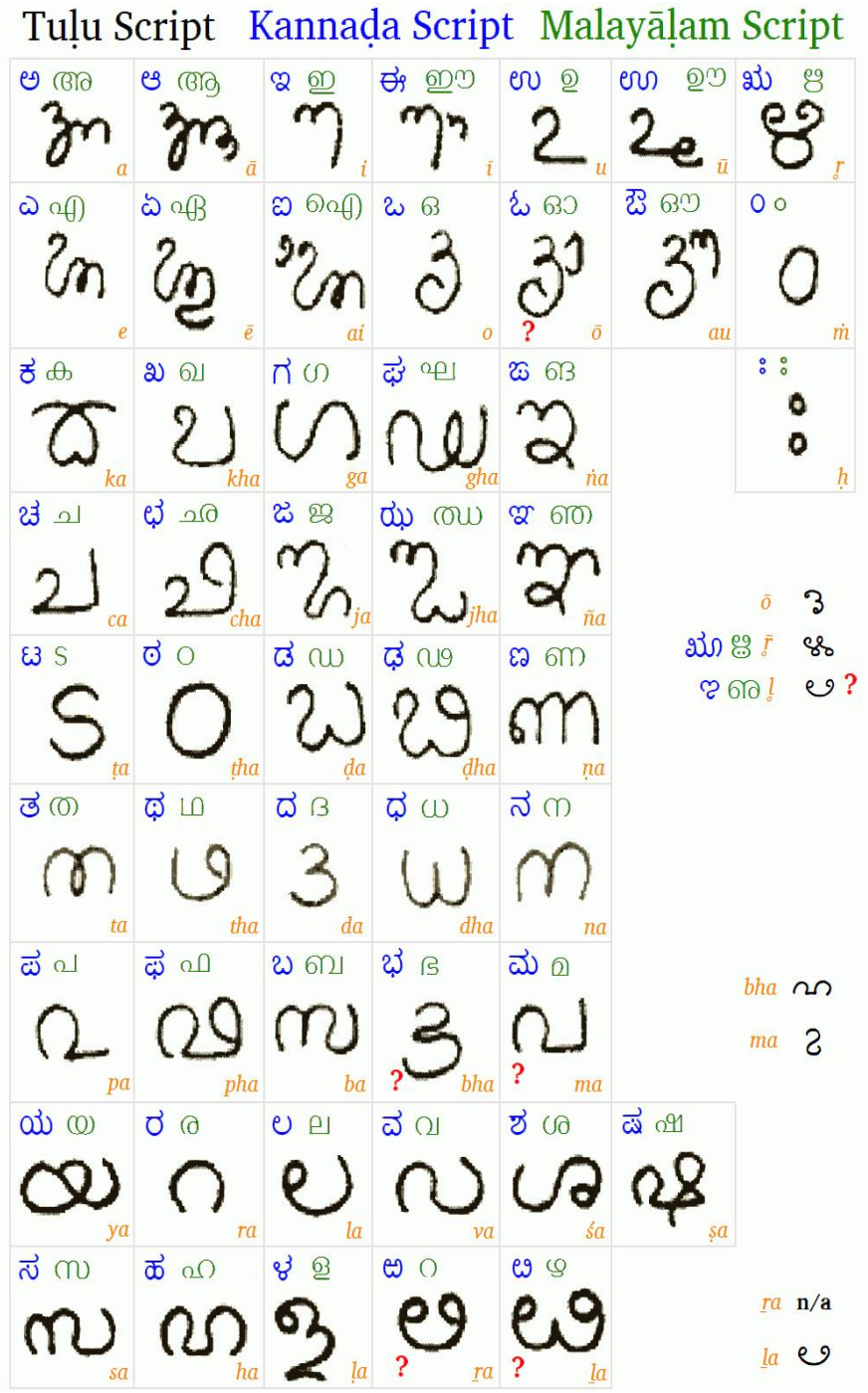


Figure 18 Standard Tulu Characters.

9. Standard Kannada characters

ಅ ಆ ಇ ಈ ಉ ಊ ಋ ಎ ಏ ಐ ಒ ಓ ಔ

ಅಂ ಅಃ

ಕ ಖ ಗ ಘ ಙ

ಚ ಛ ಜ ಝ ಞ

ಟ ಠ ಡ ಢ ಣ

ತ ಥ ದ ಧ ನ

ಪ ಫ ಬ ಭ ಮ

ಯ ರ ಲ ವ ಶ ಷ ಸ ಹ ಳ

CHAPTER 6

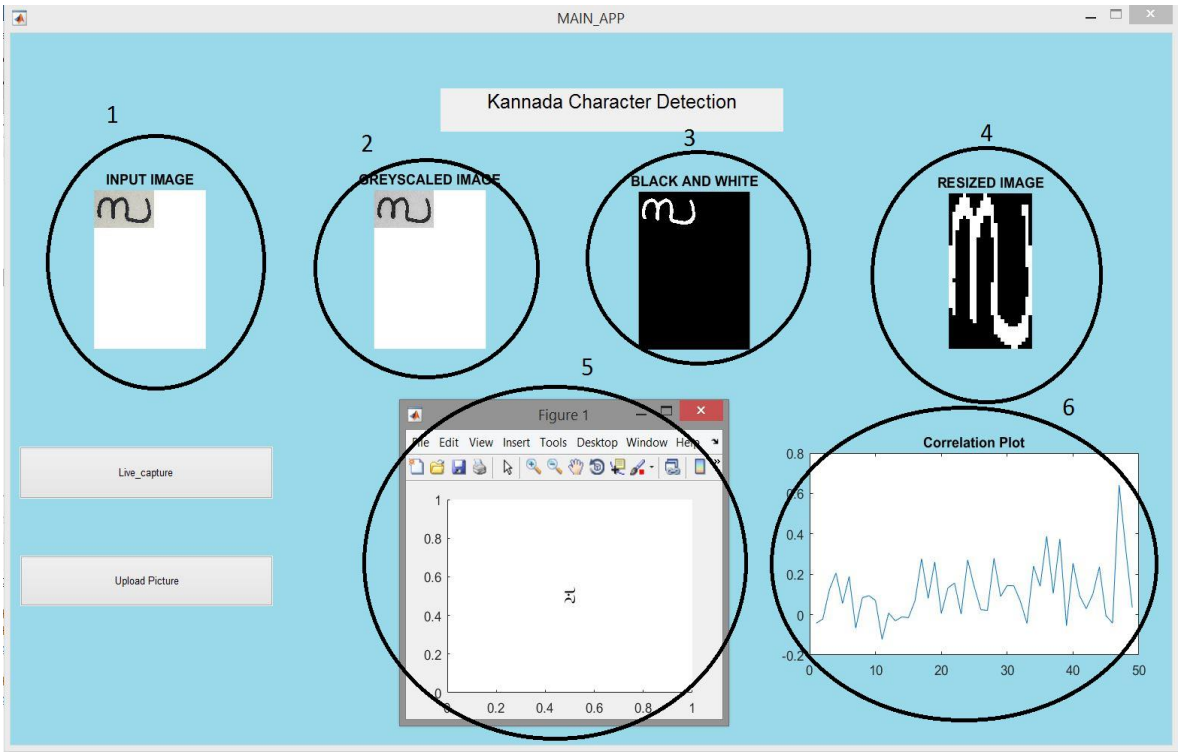
6. TESTING

6.1 TESTING

In general, testing is finding out how well something works. In terms of human beings, testing tells what level of knowledge or skill has been acquired. In computer hardware and software development, testing is used at key checkpoints in the overall process to determine whether objectives are being met.

Test cases for the applications are:-

1. Detect different handwritten Kannada character of Tulu.
2. Generate greyscale image.
3. Binarization of greyscale image
4. Resize the input image to standard size.
5. Recognize the corresponding classical Kannada characters.
6. Plot the correlation plot.

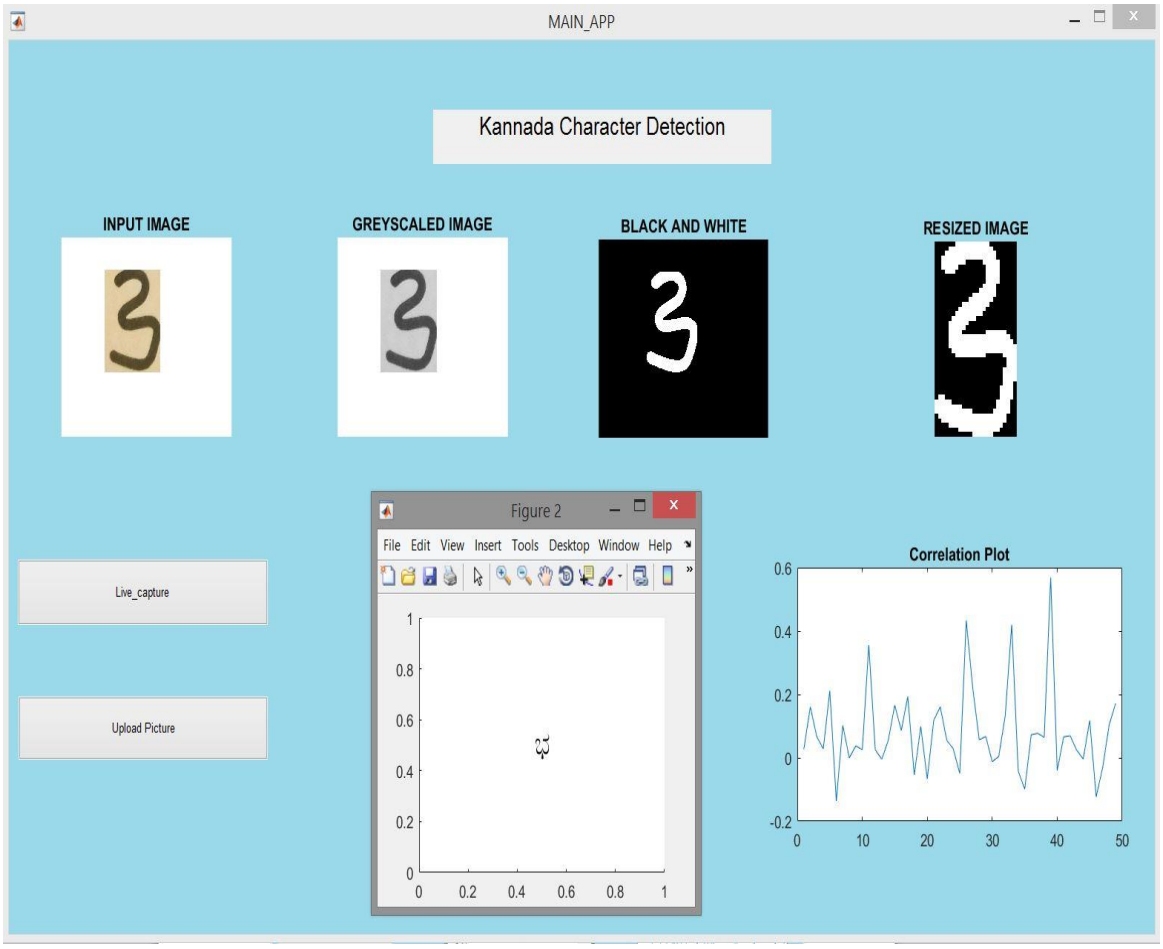

MAIN_APP
Kannada Character Detection
1
INPUT IMAGE
2
GREYSCALED IMAGE
3
BLACK AND WHITE
4
RESIZED IMAGE
5
Figure 1
File Edit View Insert Tools Desktop Window Help
6
Correlation Plot
Live_capture
Upload Picture

CHAPTER 7

7 RESULTS UNDER VARIOUS CONDITIONS

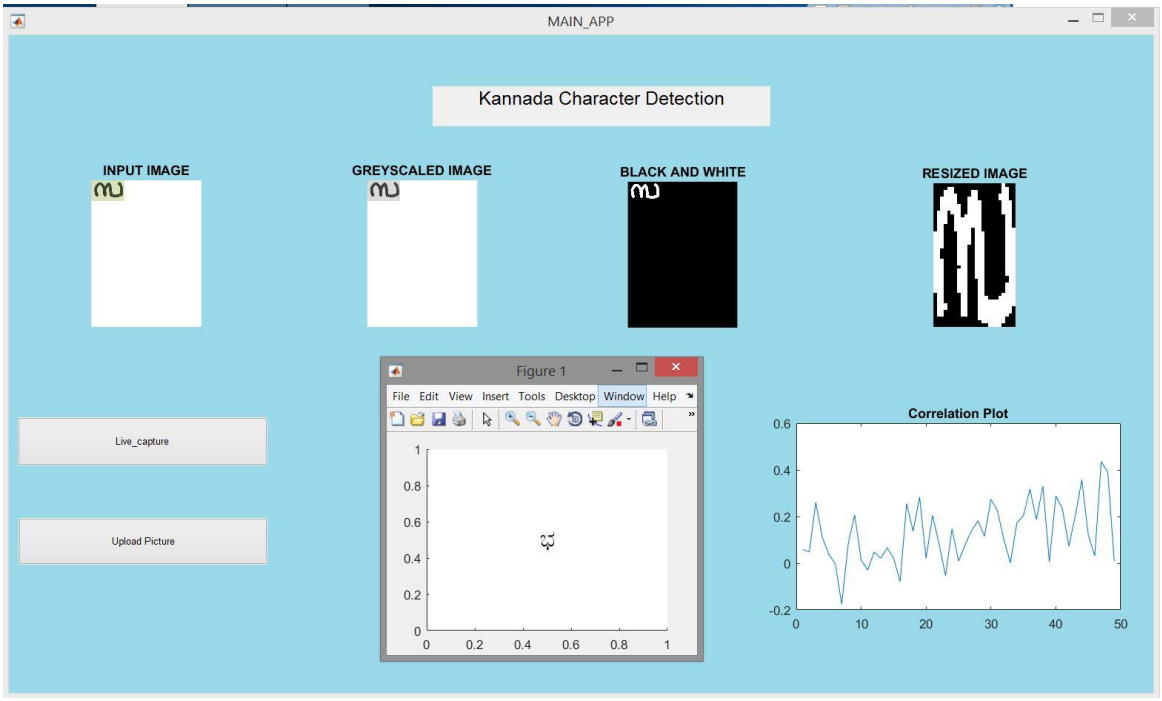
This section explains the output generated when tested against different samples under different condition. Some of which are explained below.

1) Detection and translation under yellow background

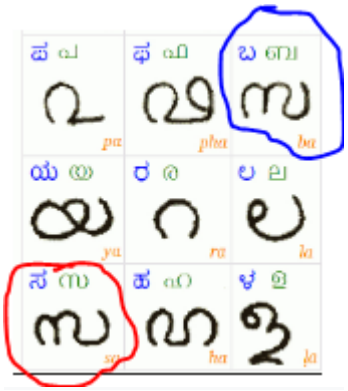


Successful output for input image was written on yellowish background

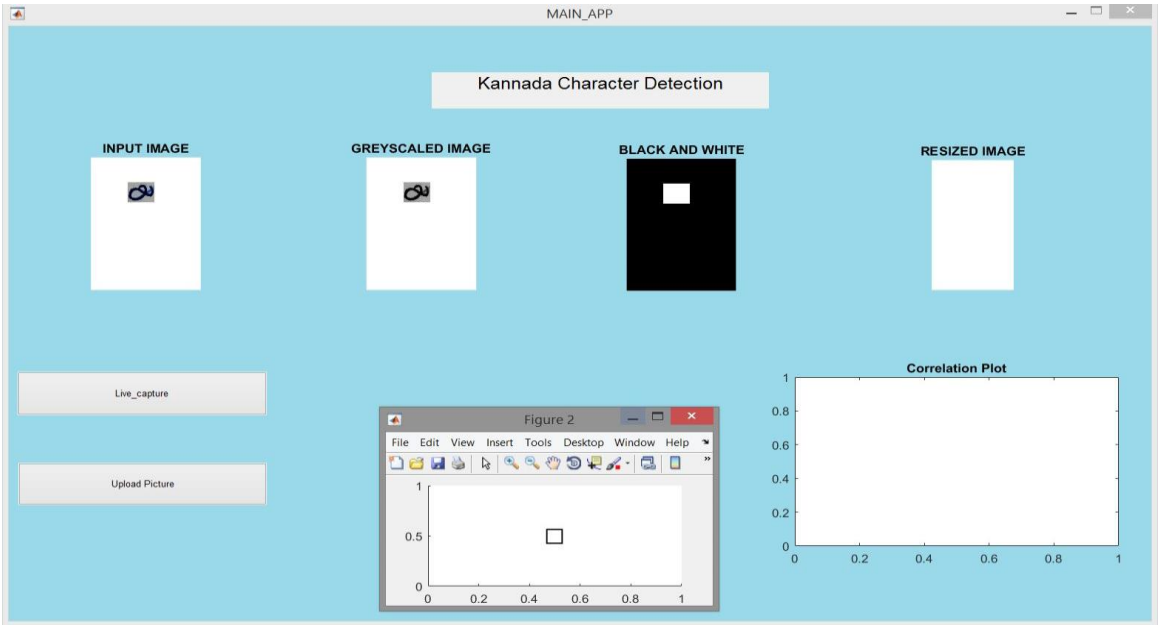
2) Detection and translation of similar appearing characters



The character under testing is ಸ (sa) which does not produces the expected output as there exist a similar character which produces output to be ಭ (bha)

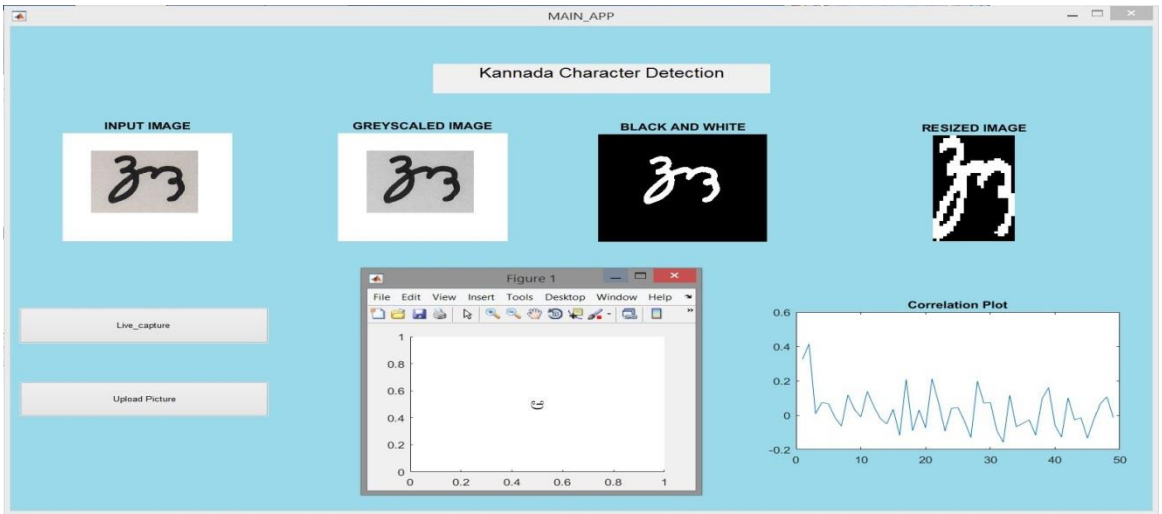


3) Detection and translation which involves overwriting

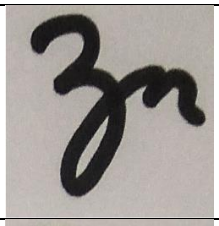
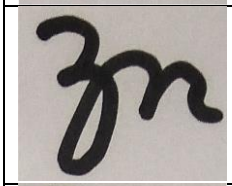
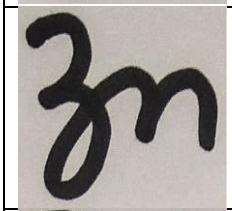
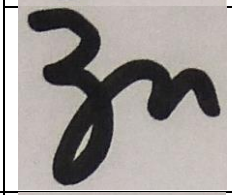
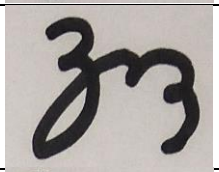
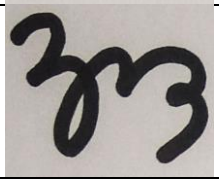


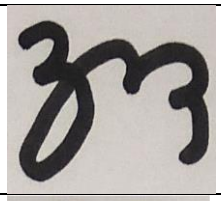
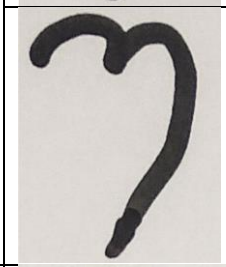
Since there is over writing in the input character the image is not processed and the output is a null character.

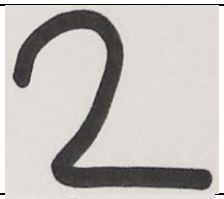
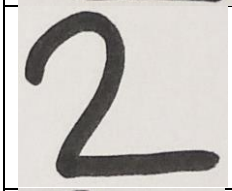
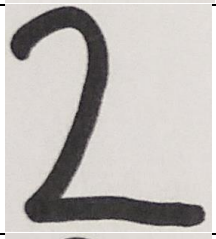
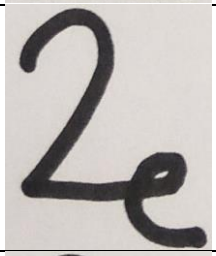
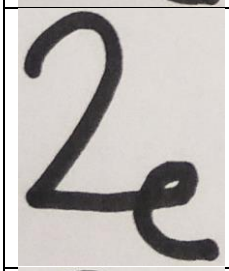
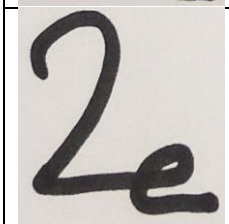
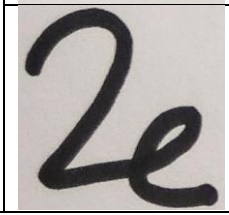
4) Detection and translation on white paper

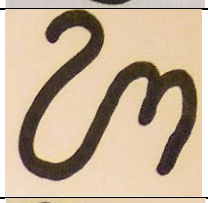
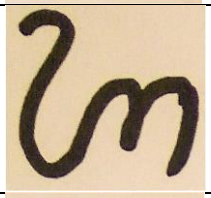
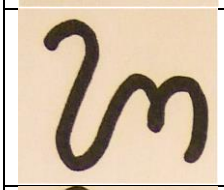
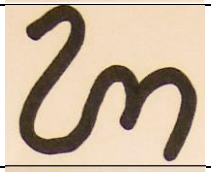
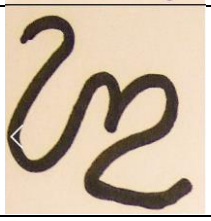


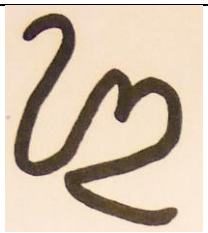
7.1 SAMPLES TABLE

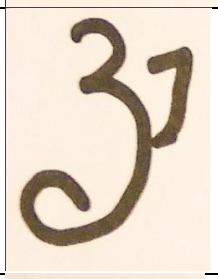
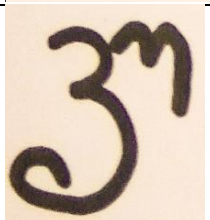
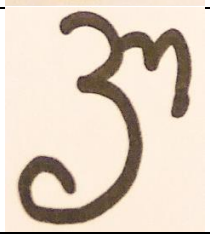
SL.NO	CHARACTER	SAMPLES	SAMPLE STATUS	CHARACTR MATCHING IN KANNADA
1.			**PASS**	
			**PASS**	
			**PASS**	
			**PASS**	
2.			**PASS**	
			**PASS**	
			**FAIL**	NO CHARACTES MATCHED

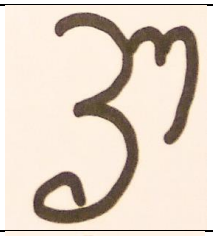
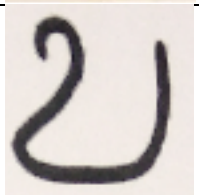
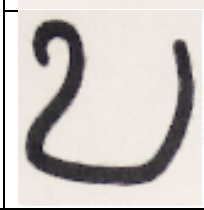
			**PASS**	
3.			**PASS**	
			**PASS**	
			**PASS**	
			**PASS**	
4.			**FAIL**	

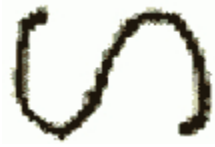
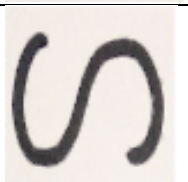
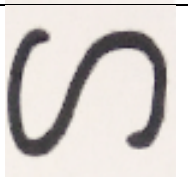
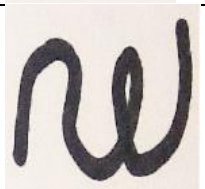
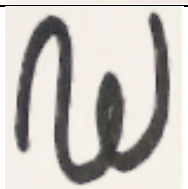
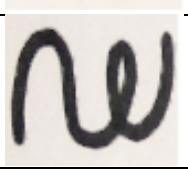
			**PASS**	
			**PASS**	
			**PASS**	
5.			**PASS**	
			**PASS**	
			**FAIL**	NO CHARACTERS MATCHED
			**FAIL**	NO CHARACTERS MATCHED

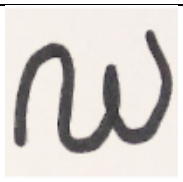
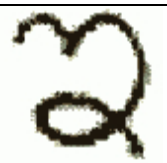
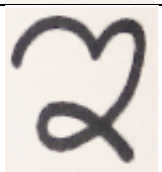
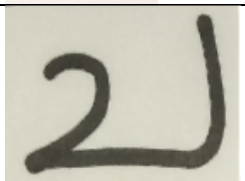
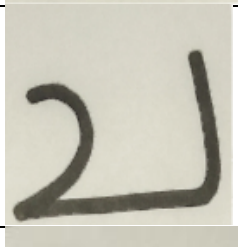
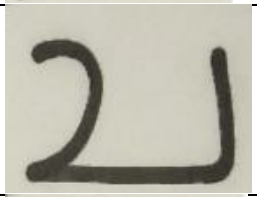
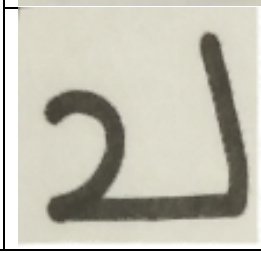
6.			**PASS**	ಋ
			**PASS**	ಋ
			**PASS**	ಋ
7.			**FAIL**	NO CHARACTERS MATCHED
			**PASS**	ಎ
			**PASS**	ಎ
			**PASS**	ಎ
8.			**PASS**	ಏ

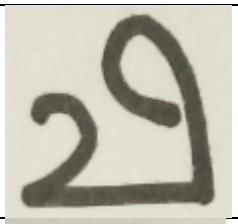
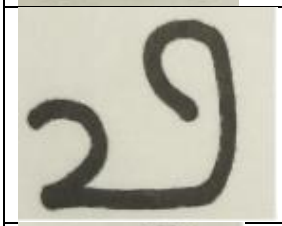
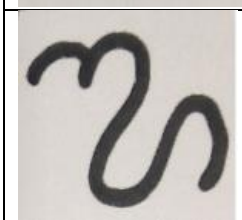
			**PASS**	
9.			**FAIL**	NO CHARACTERS MATCHED
			**PASS**	
10.			**PASS**	
			**PASS**	
			**PASS**	

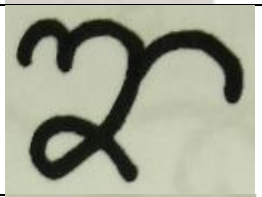
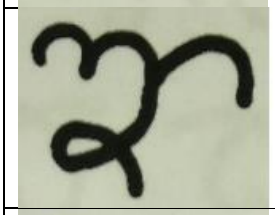
			**PASS**	
11.			**FAIL**	NO CHARACTERS MATCHED
			**PASS**	
			**PASS**	
12.			**PASS**	
			**FAIL**	NO CHARACTERS MATCHED

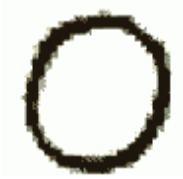
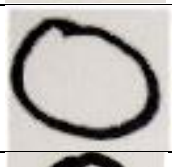
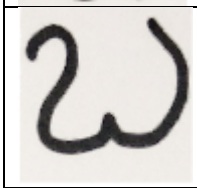
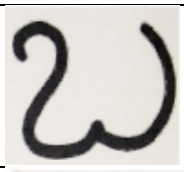
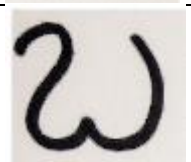
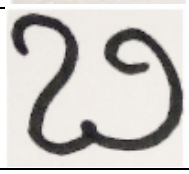
			**PASS**	ಪ
			**PASS**	ಪ
13.			**PASS**	ಂ
			**PASS**	ಂ
			**PASS**	ಂ
			**PASS**	ಂ
14.			**PASS**	ಐ
			**PASS**	ಐ

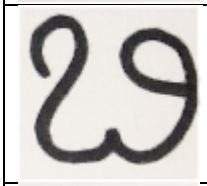
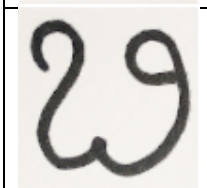
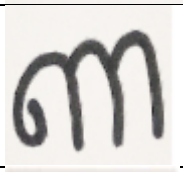
			**FAIL**	NO CHARACTERS MATCHED
			**PASS**	
15.			**PASS**	
			**PASS**	
			**PASS**	
			**PASS**	
16.			**PASS**	
			**PASS**	
			**PASS**	

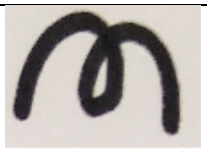
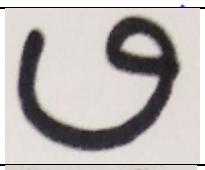
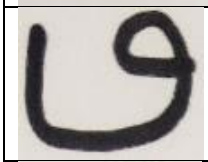
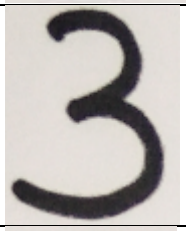
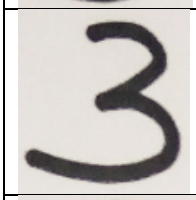
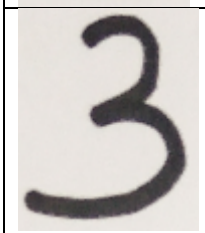
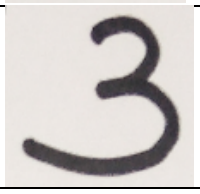
			**PASS**	ಫ
17.			**PASS**	ಐ
			**PASS**	ಐ
			**PASS**	ಐ
			**PASS**	ಐ
18.			**PASS**	ಚ
			**PASS**	ಚ
			**PASS**	ಚ
			**PASS**	ಚ

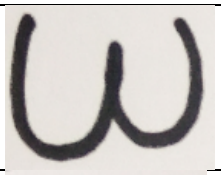
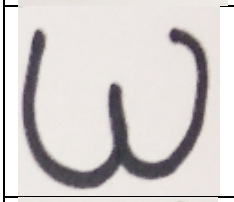
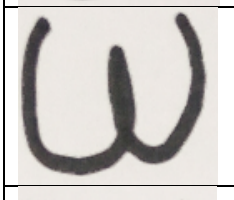
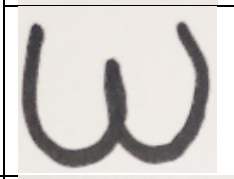
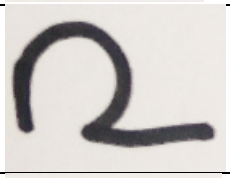
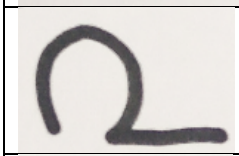
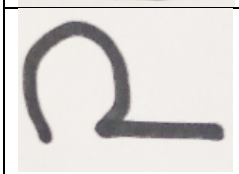
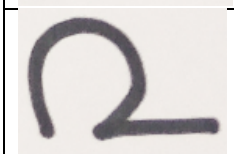
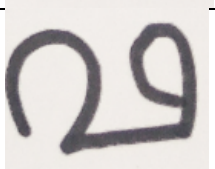
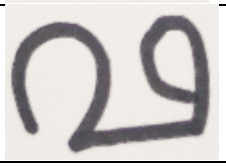
19.			**PASS**	ಛ
			**PASS**	ಛ
			**PASS**	ಛ
			**PASS**	ಛ
20.			**PASS**	ಜ
			**PASS**	ಜ
			**FAIL**	NO CHARACTERS MATCHED

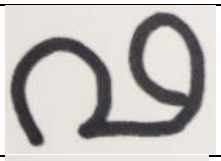
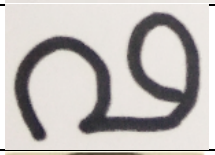
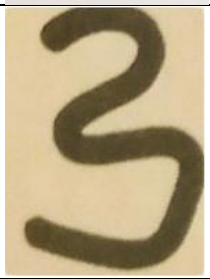
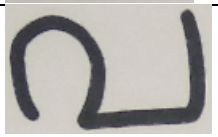
			**FAIL**	NO CHARACTERS MATCHED
21.			**FAIL**	NO CHARACTERS MATCHED
			**PASS**	
			**PASS**	
			**PASS**	
22.			**FAIL**	NO CHARACTERS MATCHED
			**PASS**	
			**PASS**	

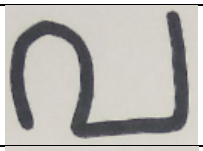
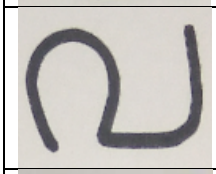
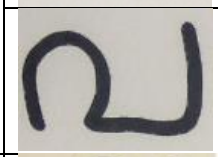
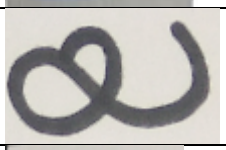
			**PASS**	ಟ
23.			**FAIL**	NO CHARACTERS MATCHED
			**PASS**	ಠ
			**PASS**	ಠ
			**PASS**	ಠ
24.			**PASS**	ಡ
			**PASS**	ಡ
			**PASS**	ಡ
			**FAIL**	NO CHARACTERS MATCHED
25.			**FAIL**	NO CHARACTERS MATCHED

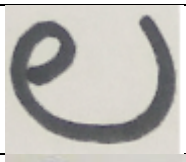
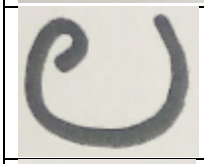
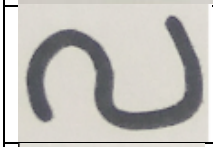
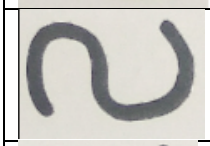
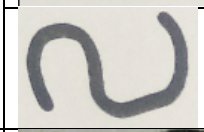
			**PASS**	
			**PASS**	
			**PASS**	
26.			**PASS**	
			**PASS**	
			**FAIL**	NO CHARACTERS MATCHED
			**PASS**	
27.			**PASS**	
			**PASS**	
			**PASS**	

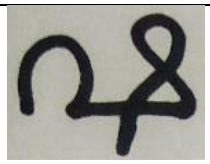
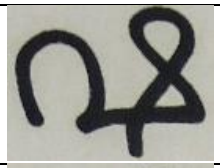
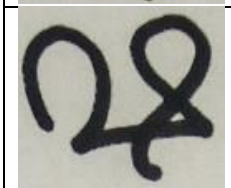
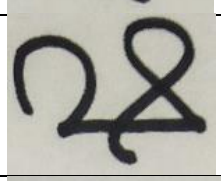
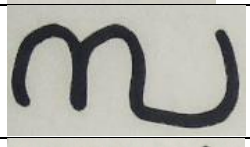
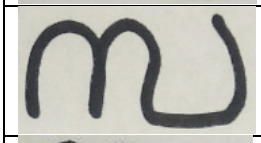
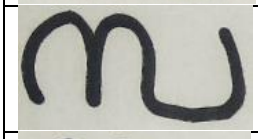
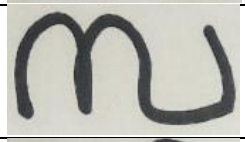
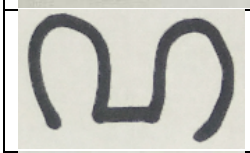
			**FAIL**	NO CHARACTERS MATCHED
28.			**FAIL**	NO CHARACTERS MATCHED
			**PASS**	ಛ
			**PASS**	ಛ
			**PASS**	ಛ
29.			**PASS**	ಡ
			**PASS**	ಡ
			**PASS**	ಡ
			**PASS**	ಡ

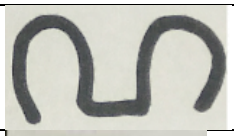
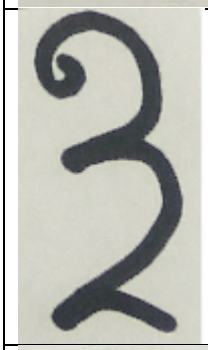
30.			FAIL	NO CHARACTERS MATCHED
			PASS	
			PASS	
			PASS	
31.			PASS	
			PASS	
			PASS	
			PASS	
32.			FAIL	NO CHARACTERS MATCHED
			FAIL	NO CHARACTERS MATCHED

			**PASS**	ಛ
			**PASS**	ಛ
33.			**PASS**	ಭ
			**PASS**	ಭ
			**PASS**	ಭ
			**FAIL**	NO CHARACTERS MATCHED
34.			**PASS**	ಮ

			**PASS**	ಮ
			**FAIL**	NO CHARACTERS MATCHED
			**PASS**	ಮ
35.			**PASS**	ಯ
			**PASS**	ಯ
			**FAIL**	NO CHARACTERS MATCHED
			**FAIL**	NO CHARACTERS MATCHED
36.			**PASS**	ರ
			**PASS**	ರ
			**PASS**	ರ
			**PASS**	ರ

37.			**PASS**	ಲ
			**PASS**	ಲ
			**PASS**	ಲ
			**PASS**	ಲ
38.			**PASS**	ವ
			**PASS**	ವ
			**PASS**	ವ
			**PASS**	ವ
39.			**FAIL**	NO CHARACTERS MATCHED
			**PASS**	ಶ
			**PASS**	ಶ
			**PASS**	ಶ

40.			**PASS**	
			**PASS**	
			**PASS**	
			**PASS**	
41.			**PASS**	
			**PASS**	
			**PASS**	
			**PASS**	
42.			**PASS**	
			**FAIL**	NO CHARACTERS MATCHED
			**PASS**	

			**PASS**	ಹ
43.			**PASS**	ಜ
			**PASS**	ಜ
			**PASS**	ಜ
			**PASS**	ಜ

The above tabular section presents a part of samples of 245 total tested samples of which 42 were not recognized due to some conditions to mention like lack of ambience, involving multiple strokes and due to anomalies between the characters.

Thus to conclude on the tested sample size total of 245 samples, 203 samples were successfully recognized and a total of 42 samples remained unrecognized, which gives an accuracy of 82.85%.

Accuracy = (Number of recognized sample/Total sample size)

= 203/245

= 82.85%

CHAPTER 8

8 CONCLUSION AND FUTURE WORK

8.1 CONCLUSION

This system describes about recognizing handwritten Tulu characters and translates in classical Kannada characters. Correlation function for recognition was implemented by taking different features of Tulu characters.

This system is tested with 245 character samples and has given an accuracy of 82.85% where we have considered average of 5 samples for each character and further expect more work in this field with a large set of character samples.

8.2 FUTURE WORK

The proposed system is designed only for the detection and translation of single character and can extended for detection of words and sentences.

This system can also be extended for the Dravidian languages like Malayalam, Telugu, and Tamil.

This system can also be extended and improved upon the shortcoming to make it more accurate by using more sophisticated algorithm.

CHAPTER 9

9 REFERENCES

1. Antony P. J, Savitha C.K, Ujwal U J. "Haar Features based Handwritten Character Recognition System for Tulu Script" IEEE International Conference on Recent Trends in Electronics Information Communication Technology, May 20-21, 2016, India 978-1-5090-0774-5/16/$31.00 © 2016 IEEE 65.
2. Hassan Althobaiti and Chao Lu Department of Computer and Information Sciences Towson University Towson, USA Email: {althobaiti, clu}@towson.edu "A Survey on Arabic Optical Character Recognition and an Isolated Handwritten Arabic Character Recognition Algorithm using Encoded Freeman Chain Code" 978-1-5090-4780-2/17/$31.00 ©2017 IEEE.
3. Saleem Pasha Department of Information Science and Engineering P.E.S. College of Engineering Mandya, Karnataka, India saleempashapes@gmail.com M.C.Padma Department of Computer Science and Engineering P.E.S. College of Engineering Mandya, Karnataka, India padmapes@gmail.com "Handwritten Kannada character recognition using wavelet transform and structural features" International Conference on Emerging Research in Electronics, Computer Science and Technology – 2015.
4. Hedieh Sajed "Handwriting recognition of digits, signs and numeral strings in Persian" 0045-7906/© 2015 Elsevier Ltd. All rights reserved Science Direct.
5. N Prameela, P Anjusha1 1Department of Computer Science and Engineering, MLR Institute of Technology Hyderabad, India prameelakotipalli@gmail.com R Karthik2* 2Department of Electronics and Communication Engineering, MLR Institute of Technology Hyderabad, India rayam16@gmail.com"Offline Telugu Handwritten Character Recognition using Optical Character Recognition" International Conference on Electronics, Communication and Aerospace Technology ICECA 2017.