ASSIGNMENT

Course Title: Image Processing & Pattern Recognition.
Course ID: CSC 420
Submitted By: Abu Bakar Siddik Nayem.
ID: 1510190.
Section: 01.
Date of Submission: 13th October, 2018.

Submitted To: Md. Ashraful Amin, PhD.

As part of OCR system, we intend to separate alphabets, more precisely characters of a language.
The steps of my approach are listed below:

1.**Skew** detection is one the first operations to be applied to scanned documents when converting data to a digital format. Its aim is to align an **image** before **processing** because text **segmentation** and recognition methods require properly aligned next lines. So to get rid of any skewness we apply **deskew. Deskew** is the **process** of detecting and fixing this issue on scanned files (ie, bitmap) so deskewed **images** will have the text/**images** correctly and horizontally aligned.

2.Then we convert the image to **grayscale** if it's not already a grayscale image.

3.Our next target is to extract the lines individually so that we can keep the sequence of the alphabets belonging to specific lines.

4. Then we find out our area/region of interest in the image by getting their index values.

5. After that we segment the lines and store them with the naming convention 'L1','L2', …. 'Ln'.

6. We use another Matlab script extract the alphabets from each line we generated in the above step.

7. To extract the alphabets from each segmented line we first remove any empty and unwanted objects that are less than 30 pixels.

8. Next we Label all the connected components in each line.

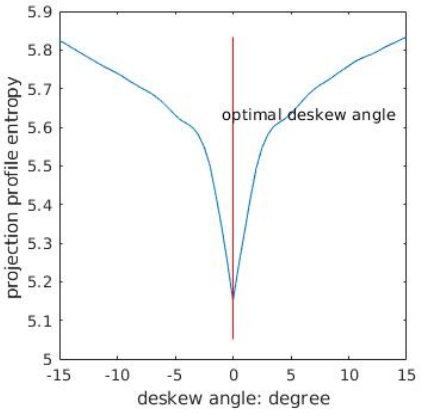
9. Then we begin extracting alphabets from each line with the following foemula:

```
n1 = Inputimage(min(r):max(r),min(c):max(c));
```

Here **n1** is the extracted alphabet, **InputImage** is the input line, **r** is row, **c** is column.

10. After extracting each alphabet/component from each lines we save them in a new folder named **Alphabets** with the naming convention **'L1_1','L1_2',……..,'Ln_m'**

Projection profile entropy:



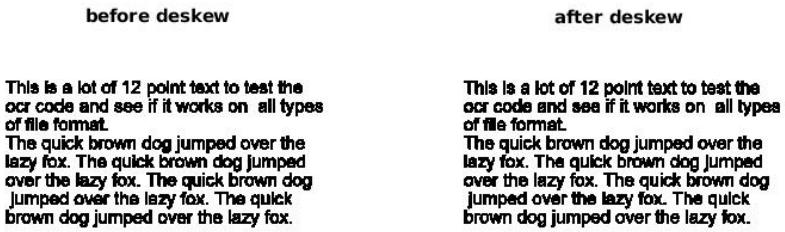


Fig-1: Deskew angle in degrees; Before & after deskew images.

Code:

main.m

```
% Read image
i=imgetfile();
input=imread(i);
figure, imshow(input), title('Original Image');

% Deskew a document
[ im, theta ] = imdeskew( input );
figure, imshow(im), title('Deskewed image');

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


%% Remove all object containing fewer than 60 pixels
im = bwareaopen(im,60);
a=(im);
ijk=~a;
L=sum(ijk,2);
L=0;
```

```
measurements = regionprops(L, 'PixelIdxList','Area','BoundingBox');
% Get indexes of those regions that are >= 20 in length.
fiveLongRegions = find([measurements.Area] >= 5);
theIndexes = vertcat(measurements(fiveLongRegions).PixelIdxList);
c=a;
c(theIndexes,:)=0;
figure, imshow(c);
title('AREA OF INTEREST FOR LINE SEGMENT');

%Segmentation of line:
[x,y]=size(c);
mat1=sum(c,2);%sum the elements of bwrowise and save into column matrix mat1
mat2=y-mat1;%subtract each element of the sum matrix(mat1) from the width length(no. of columns)
mat3=mat2~=0;
mat4=diff(mat3);
index1=find(mat4);
[q,w]=size(index1);%size of index2 matrix is q*w
kap=1;
lam=1;
%iii=1;
%ii=1;
while kap<((q/2)+1);%number of loops=number of lines
   k=1;
   mat5=([]);
     for j=(index1(lam)+1):1:index1(lam+1)
      mat5(k,:)=a(j,:); %store the line segmented matrix
        k=k+1;
     end



      number =  num2str(kap);
      imwrite(mat5,strcat('L', number, '.bmp'));
      figure, imshow(mat5);
      lam=lam+2;
      kap=kap+1;

%       imsave();
end
%deskew

function [ im, theta ] = imdeskew( src, max_angle, resolution, plotOn )

% 0. parameter settings
if nargin <= 1
   max_angle   = 15;
   resolution  = .5;
   plotOn      = 1;
elseif nargin <= 2
   resolution  = .5;
   plotOn      = 1;
elseif nargin <= 3
   plotOn      = 1;
else
   error('unsupported input format')
end
% input settings
if size( src, 3 ) > 1
   display('warning: automatic converting color image to binary')
```

```
   gray = rgb2gray( src );
   src  = gray > graythresh( gray ) * 255;
else
   if ~islogical( src )
      display('warning: automatic converting grayscale image to binary')
      src = src > graythresh( src ) * 255;
   end
end
% 1. extract black text pixels for analysis
[ h, w ]    = size( src );
[ text_x, text_y ] = ind2sub(  [ h,w ], find( src(:) == 0 ) );
% 2. compute the information entroy of a projection profile
angles = -max_angle : resolution : max_angle;
cx  = h/2;
cy  = w/2;
len = size( text_x, 1 );
hist_to_prob = @( h ) ( h( h ~= 0 ) / len );
score = [];
for a = angles
   sin_a = sin( a / 180 * pi );
   cos_a = cos( a / 180 * pi );
   sx    = round( ( text_x - cx ) * cos_a + ( text_y -cy ) * sin_a + cx );
   freq  = hist( sx, unique(sx) );
   prob  = hist_to_prob( freq );
   entropy = -prob .* log( prob );
   score(end+1) = sum( entropy(:) );
end
% 3. generate output
[ val, min_idx ] = min( score );
theta = -angles( min_idx );
im = not( imrotate( not( src ), theta, 'loose' ) );
if plotOn
     figure,subplot(131),plot( -angles, score ), xlabel( 'deskew angle: degree' ), ylabel( 'projection profile
entropy ' );
   hold on, line( [ -angles( min_idx ), -angles( min_idx )]' , [ min( score )-.1, max( score ) ]', 'color',[1,0,0] )
   hold on, text(  -angles( min_idx )-1, max( score )-.2 , 'optimal deskew angle' ),axis square
   subplot(132),imshow( imerode(src, ones(3) ) ); title( 'before deskew ');
   subplot(133),imshow( imerode(im, ones(3) ) ); title( 'after deskew' );
end

end
```

myOcr.m

```
%% Read Image

main;
clc;
mkdir Alphabets;
for fileNo = 1:kap-1

   fName = strcat('L',num2str(fileNo),'.bmp');
   Inputimage=imread(fName);
   %% Show image
   figure(1)
   imshow(Inputimage);
   title('INPUT IMAGE WITH NOISE')

   %% Convert to binary image
   threshold = graythresh(Inputimage);
   Inputimage =~im2bw(Inputimage,threshold)

   %% Remove all object containing fewer than 30 pixels
   Inputimage = bwareaopen(Inputimage,30);
   pause(1);

   %% Label connected components
    [L, Ne]=bwlabel(Inputimage);

   %% Objects extraction
   % figure
   cd Alphabets;
   for n=1:Ne
     [r,c] = find(L==n);
     n1=Inputimage(min(r):max(r),min(c):max(c));
   %   imshow(~n1);
     imwrite(n1,strcat('L',num2str(fileNo),'_',num2str(n),'.tif'));
   %   pause(0.5)
   end
   cd .. ;

end
```

The input images is:

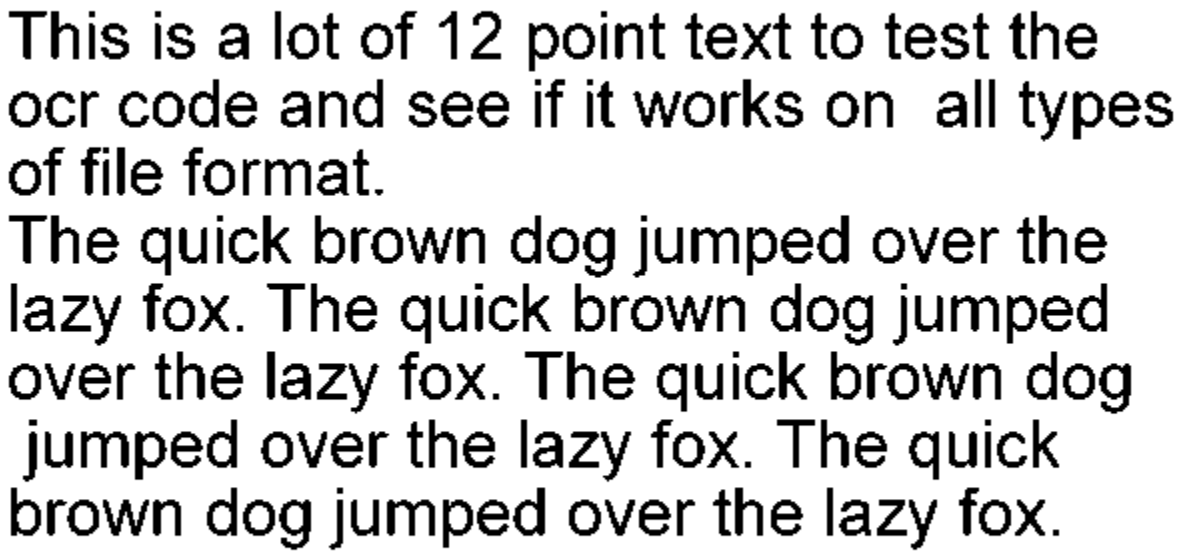


Fig-2: Input image.

Output image samples:

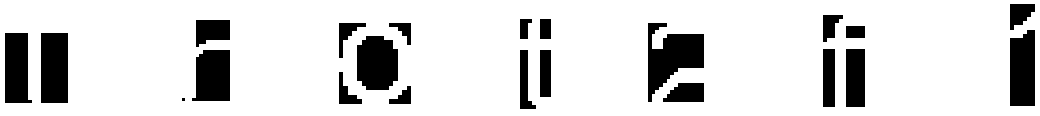


Fig-3: Output sample images.

Bangla OCR:

code:

```
%% Read Image
Inputimage=imread('Bangla_testocr.bmp');

%% Convert to binary image
threshold = graythresh(Inputimage);
Inputimage =~im2bw(Inputimage,threshold)

%% Remove all object containing fewer than 30 pixels
Inputimage = bwareaopen(Inputimage,30);

%% Objects extraction
% figure
for n=1:Ne
  [r,c] = find(L==n);
  n1=Inputimage(min(r):max(r),min(c):max(c));
%   imshow(~n1);
  imwrite(n1,strcat(num2str(n),'.bmp'));
%   pause(0.5)
end
```

Input Image:

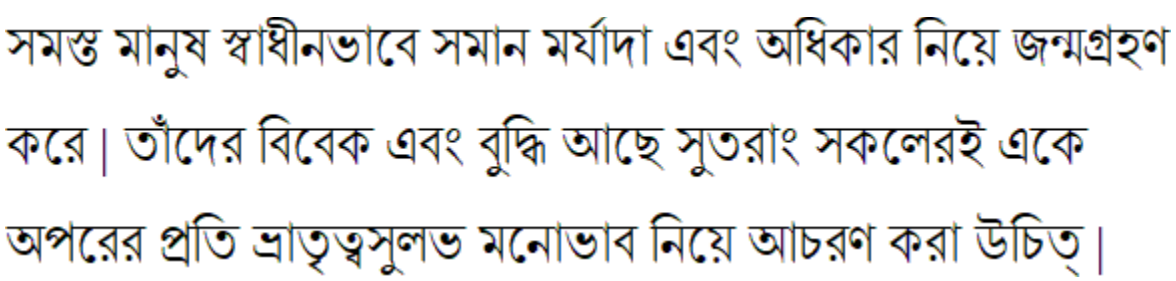


Fig-4: Input Image.

Output Sample:

Fig-5: Sample output Image.

Analysis:

We will compare our OCR system to the matlab's built in 'ocr' function.

We use the ocr function to get the text from the input image. Then we can get any alphabet from the text we got from our function to the matlab's default one.

Code:

```
OCR_input_image = imread('English_testocr.png');
OCR_input_image = rgb2gray(OCR_input_image);
results = ocr(OCR_input_image);

alphabet = results.Text(1);

imwrite(insertText(zeros(24,19),[0    0],alphabet,'BoxOpacity',0,'FontSize',20,'TextColor','w'),
'T1.tif');

img1 = imread('L1_1.tif');
img2 = imread('T1.tif');

img1 = im2uint8(img1);
img2 = rgb2gray(img2);
```

Observation:

The return type of our OCR system is of type image and the default 'ocr' returns 'ocrText' type object which is actually of type text.

When we explore varibale 'img1' and 'img2' from workspace we can see that the thickness of the alphabet differs considerably even though they are of same shape. In this case the recognized alphabet is a 'T'. In our OCR system the alphabet has 3 pixels thickness horizontally and 4 pixels vertically in contrast to 2 pixels in both ways in the default 'ocr' function.

[Note: Since the position of the alphabet in both the systems are inconsistent due to casting of the alphabet from 'ocrText' type to image type, a numerical difference(i.e. imsubtract) could not be calculated.]

BanglaOCR:

Since there are continious horizontal straight lines(মাত্রা) on the upper side of most Bangla words, our OCR system fails to extract Bangla alphabets. Our system can mostly extract words from an image of bangla text, but where there is a gap in the prominent continious horizontal straight lines(মাত্রা) , sometimes our system can extract alphabets successfully.