ECE 6310 Introduction to Computer Vision

Lab 2 – Optical Character Recognition

Prajval Vaskar

C20664702

September 15, 2020

CLEMSON

UNIVERSITY

The program should perform the following steps: 1. Read the input image, template image, and ground truth files.

2. Calculate the matched-spatial filter (MSF) image (remember, this is not 8-bits).

3. Normalize the MSF image to 8-bits.

4. Loop through the following steps for a range of T: a. Threshold at T the normalized MSF image to create a binary image. b. Loop through the ground truth letter locations. i. Check a 9 x 15 pixel area centered at the ground truth location. If any pixel in the msf image is greater than the threshold, consider the letter “detected”. If none of the pixels in the 9 x 15 area are greater than the threshold, consider the letter “not detected”. c. Categorize and count the detected letters as FP (“detected” but the letter is not ‘e’) and TP (“detected” and the letter is ‘e’). d.

Output the total FP and TP for each T. Using any desired program, you must create an ROC curve from the program output.

Solution:

Template image letter = ‘e’

To calculate the matched spatial filter, zero centered mean image of the template is required.

For that find, the mean of the template image pixel and subtract the mean from each pixel from the template image.

Mean is calculated as 165.39.

zero_cenred_image[i] = image[i] – mean.

After that, convolution of image and zero centered mean image is done by using following formula.

$$MSF[r,c] = \sum_{dr=-Wr/2}^{+Wr/2} \sum_{dc=-Wc/2}^{+Wc/2} \left[I[r+dr, c+dc] * T[dr+Wr/2, dc+Wc/2] \right]$$

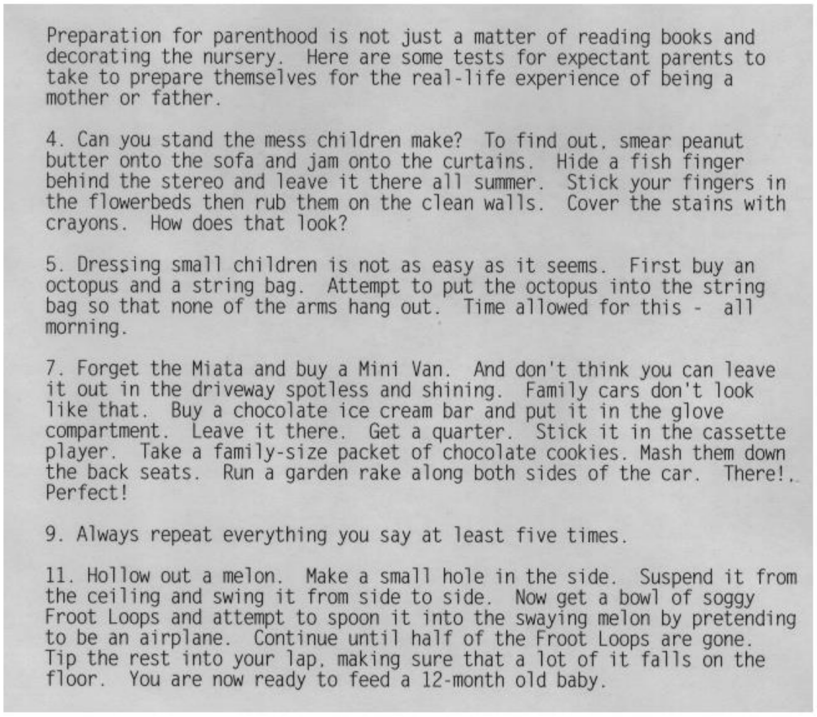
where,

Wr, Wc are rows and columns of template image.

After convolution as the pixel value gets changed and no longer in between 0 to 255 so, we have to normalize the image between 0 to 255.

For normalization following formula is used,

$$I_N = (I - \text{Min}) \frac{\text{newMax} - \text{newMin}}{\text{Max} - \text{Min}} + \text{newMin}$$



Preparation for parenthood is not just a matter of reading books and decorating the nursery. Here are some tests for expectant parents to take to prepare themselves for the real-life experience of being a mother or father.

4. Can you stand the mess children make? To find out, smear peanut butter onto the sofa and jam onto the curtains. Hide a fish finger behind the stereo and leave it there all summer. Stick your fingers in the flowerbeds then rub them on the clean walls. Cover the stains with crayons. How does that look?

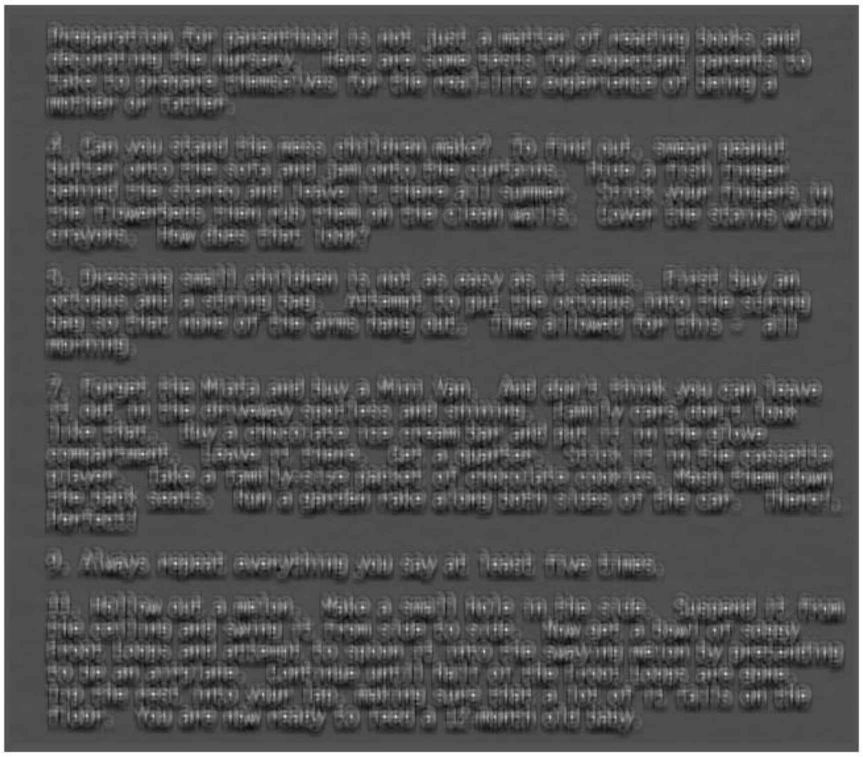
5. Dressing small children is not as easy as it seems. First buy an octopus and a string bag. Attempt to put the octopus into the string bag so that none of the arms hang out. Time allowed for this - all morning.

7. Forget the Miata and buy a Mini Van. And don't think you can leave it out in the driveway spotless and shining. Family cars don't look like that. Buy a chocolate ice cream bar and put it in the glove compartment. Leave it there. Get a quarter. Stick it in the cassette player. Take a family-size packet of chocolate cookies. Mash them down the back seats. Run a garden rake along both sides of the car. There!, Perfect!

9. Always repeat everything you say at least five times.

11. Hollow out a melon. Make a small hole in the side. Suspend it from the ceiling and swing it from side to side. Now get a bowl of soggy Froot Loops and attempt to spoon it into the swaying melon by pretending to be an airplane. Continue until half of the Froot Loops are gone. Tip the rest into your lap, making sure that a lot of it falls on the floor. You are now ready to feed a 12-month old baby.

Figure 1 Original parent image



Preparation for parenthood is not just a matter of reading books and decorating the nursery. Here are some tests for expectant parents to take to prepare themselves for the real-life experience of being a mother or father.

4. Can you stand the mess children make? To find out, smear peanut butter onto the sofa and jam onto the curtains. Hide a fish finger behind the stereo and leave it there all summer. Stick your fingers in the flowerbeds then rub them on the clean walls. Cover the stains with crayons. How does that look?

5. Dressing small children is not as easy as it seems. First buy an octopus and a string bag. Attempt to put the octopus into the string bag so that none of the arms hang out. Time allowed for this - all morning.

7. Forget the Miata and buy a Mini Van. And don't think you can leave it out in the driveway spotless and shining. Family cars don't look like that. Buy a chocolate ice cream bar and put it in the glove compartment. Leave it there. Get a quarter. Stick it in the cassette player. Take a family-size packet of chocolate cookies. Mash them down the back seats. Run a garden rake along both sides of the car. There!, Perfect!

9. Always repeat everything you say at least five times.

11. Hollow out a melon. Make a small hole in the side. Suspend it from the ceiling and swing it from side to side. Now get a bowl of soggy Froot Loops and attempt to spoon it into the swaying melon by pretending to be an airplane. Continue until half of the Froot Loops are gone. Tip the rest into your lap, making sure that a lot of it falls on the floor. You are now ready to feed a 12-month old baby.

Figure 2 Normalized image after applying MSF filter

After applying threshold binary image is created to find actual matches of letter e.

$$O[r,c] = \left. \begin{matrix} 1 \\ 0 \end{matrix} \right\} \begin{matrix} MSF[r,c] \geq T \\ MSF[r,c] < T \end{matrix}$$

Threshold is applied in a range of 0 to 255 threshold.

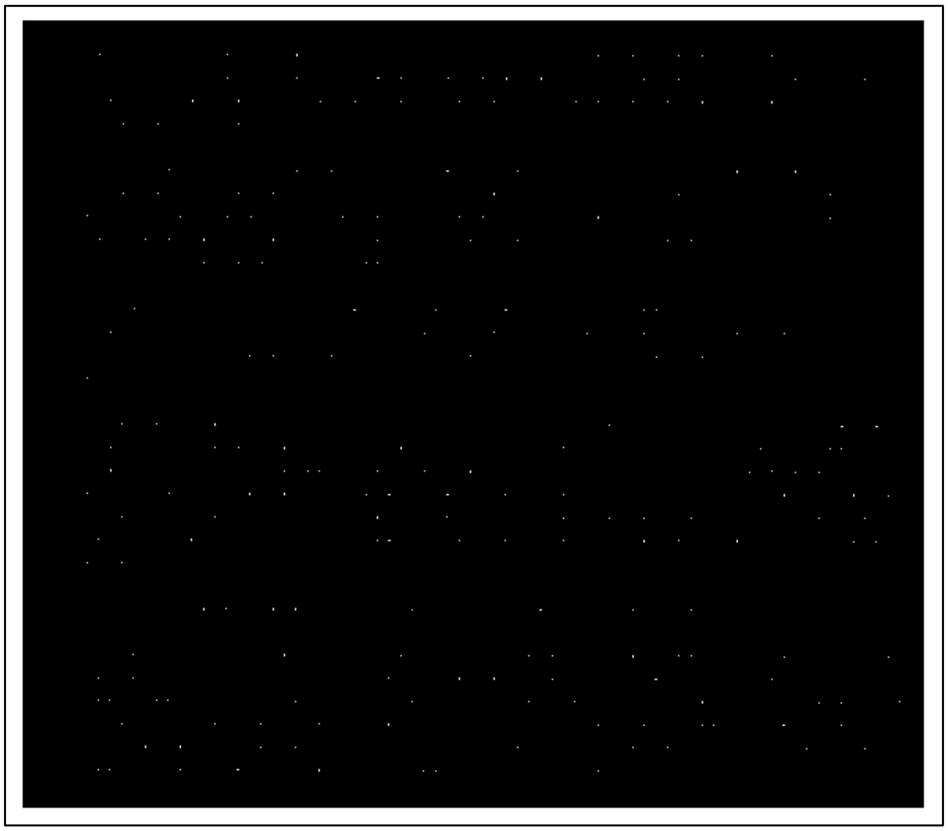


Figure 3 Binary image for optimum threshold

For each threshold, FPR and TPR are calculated by following formulae,

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

Where,

TP – True Positive

FP – False Positive

TN – True Negative

FN – False Negative

ROC curve

Values of TP, FP, FN, TN are calculated from C code and TPR and FPR for each threshold is calculated to plot ROC curve.

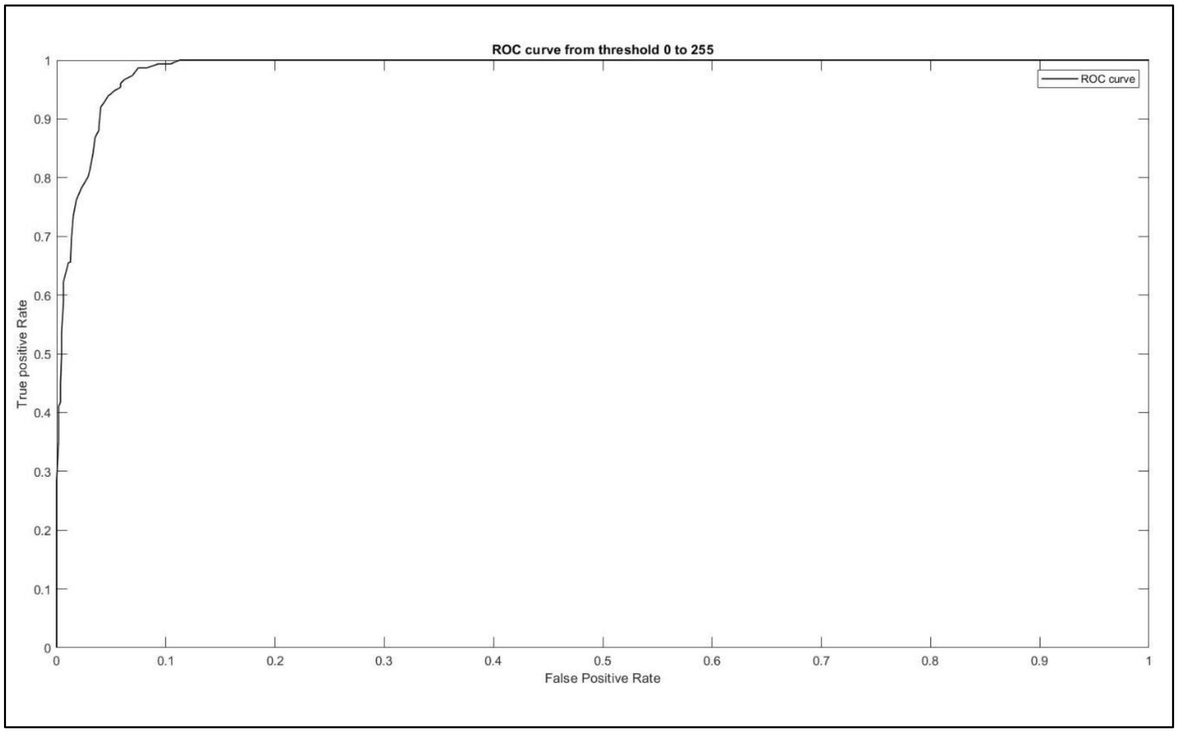


Figure 4

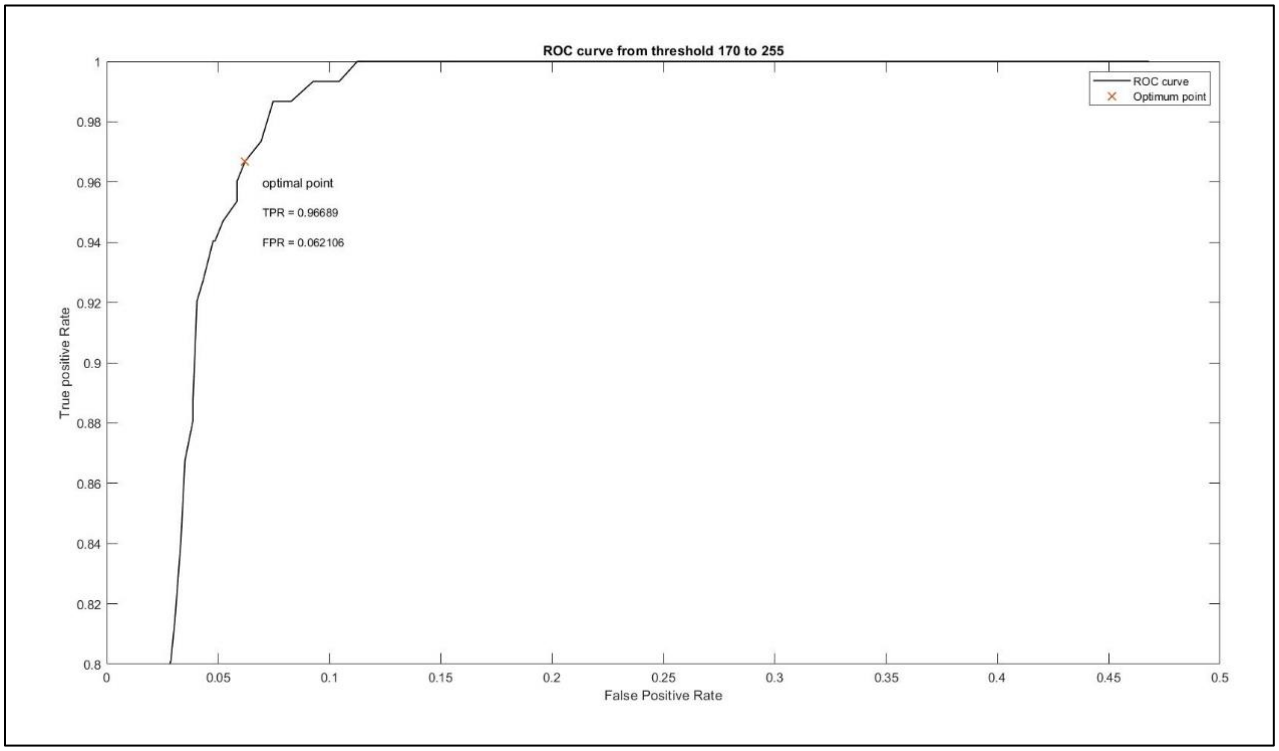


Figure 4

For the optimum value of threshold from the ROC curve, Euclidian distance is calculated from (0,1) i.e. FPR = 0 and TPR = 1 to point of each point on ROC curve. The point with minimum Euclidian distance is taken as optimum point and following result is found.

The minimum distance comes out to be 0.00704 and respective threshold value is 206.

Optimum Point

T = 206.

TP = 146

FP = 69

TPR = 0.966.

FPR = 0.062.

Appendix

C code

```
  /*
  ** Demonstrates how to load a PPM image from a file,
  ** do some simple image processing, and then write out
  ** the result to another PPM file.
  */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main ()

{
FILE    *fpt;
unsigned char *image,*template,*normalized,*threshold;
char    header[80];
char header_temp[80],gt_letter[1260];
int   ROWS,COLS,BYTES,rows,cols,detect,detect_N,TP,FP,TN,FN,count;
int   i,j,r,c,r2,c2,r3,c3,sum,new_min,new_max,or,oc;
int final, initial,num_values,step;
float mean,min_old,max_old,*zero_mean,*convoluted,difference;
int *thresholdvalues;


fpt=fopen("parenthood.ppm","rb");
if (fpt == NULL)
  {
  printf("Unable to open %s for reading\n","parenthood.ppm");
  exit(0);
  }
  /* read image header (simple 8-bit greyscale PPM only) */
i=fscanf(fpt,"%s %d %d %d",header,&cols,&rows,&BYTES);
if (i != 4  ||  strcmp(header,"P5") != 0  ||  BYTES != 255)
  {
  printf("Image is not an 8-bit PPM greyscale (P5) image\n");
  fclose(fpt);
  exit(0);
  }
  /* allocate dynamic memory for image */
image=(unsigned char *)calloc(rows*cols,sizeof(unsigned char));
if (image == NULL)
```

```
  {
  printf("Unable to allocate %d x %d memory\n",cols,rows);
  exit(0);
  }
  /* read image data from file */
header[0] = fgetc(fpt);
fread(image,1,rows*cols,fpt);
fclose(fpt);


fpt=fopen("parenthood_e_template.ppm","rb");
if (fpt == NULL)
  {
  printf("Unable to open %s for reading\n","parenthood_e_template.ppm");
  exit(0);
  }
  /* read image header (simple 8-bit greyscale PPM only) */
i=fscanf(fpt,"%s %d %d %d",header,&COLS,&ROWS,&BYTES);
if (i != 4  ||  strcmp(header,"P5") != 0  ||  BYTES != 255)
  {
  printf("Image is not an 8-bit PPM greyscale (P5) image\n");
  fclose(fpt);
  exit(0);
  }
  /* allocate dynamic memory for image */
template=(unsigned char *)calloc(ROWS*COLS,sizeof(unsigned char));
if (template == NULL)
  {
  printf("Unable to allocate %d x %d memory\n",COLS,ROWS);
  exit(0);
  }
  /* read image data from file */
header_temp[0] = fgetc(fpt);
fread(template,1,ROWS*COLS,fpt);
fclose(fpt);


  /* simple CV operation to add b to every pixel */
sum = 0;
for (r=0; r<ROWS; r++)
for (c=0;c<COLS;c++)
  {
  sum = sum + template[r*COLS+c];
  }
printf("sum is %d\n",sum);
```

```
mean = (float)sum/(ROWS * COLS);
printf("mean is %f\n",mean);

zero_mean=(float *)calloc(ROWS*COLS,sizeof(float));

/*for (i=0;i<ROWS*COLS;i++){
  zero_mean[i] = template[i];
}*/

for (r=0; r<ROWS; r++){
for (c=0;c<COLS;c++){
 j = (float)template[r*COLS+c]-mean ;
 zero_mean[r*COLS+c] = (float)j;
}
}

convoluted = (float *)calloc(rows*cols,sizeof(float));

for (r=7; r<(rows-7); r++){
for (c=4;c<(cols-4);c++){
  sum = 0;
  for (r2=-7;r2<(ROWS-7);r2++){
  for (c2=-4;c2<(COLS-4);c2++){
      sum += (image[(r+r2)*cols+c+c2] * zero_mean[(r2+(ROWS/2))*COLS+(c2+(COLS/2)
)]);
    }
  }
  convoluted[r*cols+c] = sum;
}
}

max_old = 0;
for(i=0;i<rows*cols;i++){
  if (convoluted[i] > max_old){
    max_old = convoluted[i];
  }
}

min_old = 0;
for(i=0;i<rows*cols;i++){
  if (convoluted[i] < min_old){
    min_old = convoluted[i];
  }
}
```

```
printf("max %f min %f\n",max_old,min_old);

new_min = 0;
new_max =255;
normalized = (unsigned char *)calloc(rows*cols,sizeof(unsigned char));
for (r=0; r<rows; r++){
for (c=0;c<cols;c++){
  normalized[r*cols+c] = ((convoluted[r*cols+c]-min_old)*((new_max-
new_min)/(max_old-min_old)))+new_min;
}
}
fpt=fopen("normalized.ppm","wb");
fprintf(fpt,"P5 %d %d 255\n",cols,rows);
fwrite(normalized,cols*rows,1,fpt);
fclose(fpt);


threshold = (unsigned char *)calloc(rows*cols,sizeof(unsigned char));

for (i=0;i<rows*cols;i++){
  threshold[i] = normalized[i];
}

  initial = 0;
  final = 255;
  num_values = 255;
  thresholdvalues = (int *)calloc(num_values,sizeof(int));
  step = (final - initial) / num_values;
  int k = 0;
  int ct = 0;
  thresholdvalues[j] = initial;
  for(k = 1; k<=num_values; k++){
      thresholdvalues[k] = thresholdvalues[k-1] + step;
      ct+=1;
  }

float TPR=0.0;
float FPR=0.0;
float *TPR_Array = (float *)calloc(num_values+1,sizeof(float));
float *FPR_Array = (float *)calloc(num_values+1,sizeof(float));
int *TP_Array = (int *)calloc(num_values+1,sizeof(int));
int *FP_Array = (int *)calloc(num_values+1,sizeof(int));
int *T = (int *)calloc(num_values+1,sizeof(int));

/* For Loop for T*/
```

```
int a;
for (a =initial; a<=final;a++){
  threshold = (unsigned char *)calloc(rows*cols,sizeof(unsigned char));
  for (r=0; r<rows; r++){
  for (c=0;c<cols;c++){
    if (normalized[r*cols+c] >= thresholdvalues[a]) {
      threshold[r*cols+c] =255;
    }
    else {
      threshold[r*cols+c] =0;
    }
  }
  }
  /*fpt=fopen("threshold.ppm","wb");
  fprintf(fpt,"P5 %d %d 255\n",cols,rows);
  fwrite(threshold,cols*rows,1,fpt);
  fclose(fpt);*/


  fpt = fopen("parenthood_gt.txt","rb");
  detect = 0;
  detect_N = 0;
  TP = 0;
  FP = 0;
  FN = 0;
  TN = 0;

  while(1)
  {
    detect = 0;
    detect_N = 0;

    i = fscanf(fpt,"%s %d %d",&gt_letter, &oc,&or);

    if (i != 3)
    break;

    for (r=(-ROWS/2);r<=(ROWS/2);r++){
      if (detect == 0){
        for(c = (-COLS/2);c<=(COLS/2);c++){
          if (normalized[(or+r)*cols+(oc+c)] > thresholdvalues[a]){
            detect = 1;
          }
        }
      }
    }

    if (detect ==0){
      detect_N = 1;
    }
    if (detect ==1 && gt_letter[0] == 'e'){
      TP++;
      }
    else if(detect==1 && gt_letter[0] != 'e'){
      FP++;
      }
    else if(detect_N==1 && gt_letter[0] != 'e'){
      TN++;
      }
    else if(detect_N==1 && gt_letter[0] == 'e'){
      FN++;
      }
    }
  printf("TP FP TN FN for %d are %d %d %d %d \n", a,TP,FP,TN,FN);

  fclose(fpt);
  TPR = TP/(float)(TP+FN);
  FPR = FP/(float)(FP+TN);
  printf("TPR and FPR for %d are %f %f", a,TPR,FPR);
  TP_Array[a] = TP;
  FP_Array[a] = FP;
  T[a] = a;
  TPR_Array[a] = TPR;
  FPR_Array[a] = FPR;

}

fpt=fopen("TP_FP.txt","wb");
    if (fpt == NULL)
    {
    printf("Unable to open TPR_FPR.txt for writing\n");
    exit(0);
    }
    for(int b=initial; b<=final;b++){
        fprintf(fpt,"%d %d %d %f %f\n",T[b],TP_Array[b], FP_Array[b],TPR_Array[b]
, FPR_Array[b]);

    }
    fclose(fpt);
```

```
}
```

MATLAB Code for ROC graph generation from the data created by algorithm.

TP_FP.txt file is needed.

```
%% Prajval Vaskar
% Computer Vision
% Lab 02 - Optical Character Recognition
% ROC curve plot

%TP_FP.txt file is created using C code of letter
recognition algorithm.
filename = 'TP_FP.txt';
delimiterIn = ' ';
headerlinesIn = 1;
A = importdata(filename,delimiterIn,headerlinesIn);

Data = A.data;
T = Data(:,1);
TPR = Data(:,4);
FPR = Data(:,5);
x = FPR(170:255);
y = TPR(170:255);
t = T(170:255);

figure()
plot(FPR,TPR,'k','LineWidth',1);
title("ROC curve from threshold 0 to 255");
xlabel("False Positive Rate");
ylabel("True positive Rate");
legend('ROC curve');


% Optimal point
for i= 1:length(x)
    dist(i) = sqrt((x(i)-0)^2 + (y(i)-1)^2);
end
min_dist = min(dist);
min_index = find(dist == min(dist(:)));
%Optimal point FRP
```

```
optimal_x = x(min_index);
%optimal point TRP
optimal_y = y(min_index);
%Optimal Threshold
optimal_T = t(min_index);

figure()
plot(x,y,'k','LineWidth',1);
hold on
plot(optimal_x,optimal_y,'x','MarkerSize',9,'LineWidth',1);
str = {'optimal point'};
text(0.07,0.96,str,'FontSize',10);
t = text(0.07,0.94,['FPR = ',num2str(optimal_x)]);
t.FontSize = 9;
t = text(0.07,0.95,['TPR = ',num2str(optimal_y)]);
t.FontSize = 9;
title("ROC curve from threshold 170 to 255");
xlabel("False Positive Rate");
ylabel("True positive Rate");
legend('ROC curve','Optimum point');
xlim([0,0.5]);
ylim([0.8,1]);
```