SEPTEMBER 14, 2020

CUICAR

CLEMSON UNIVERSITY INTERNATIONAL CENTER FOR AUTOMOTIVE RESEARCH

OPTICAL CHARACTER RECOGNITION

ECE 6310, HW02

HUZEFA KAGALWALA

C48290423

Part1:

MSF Image

Ans:

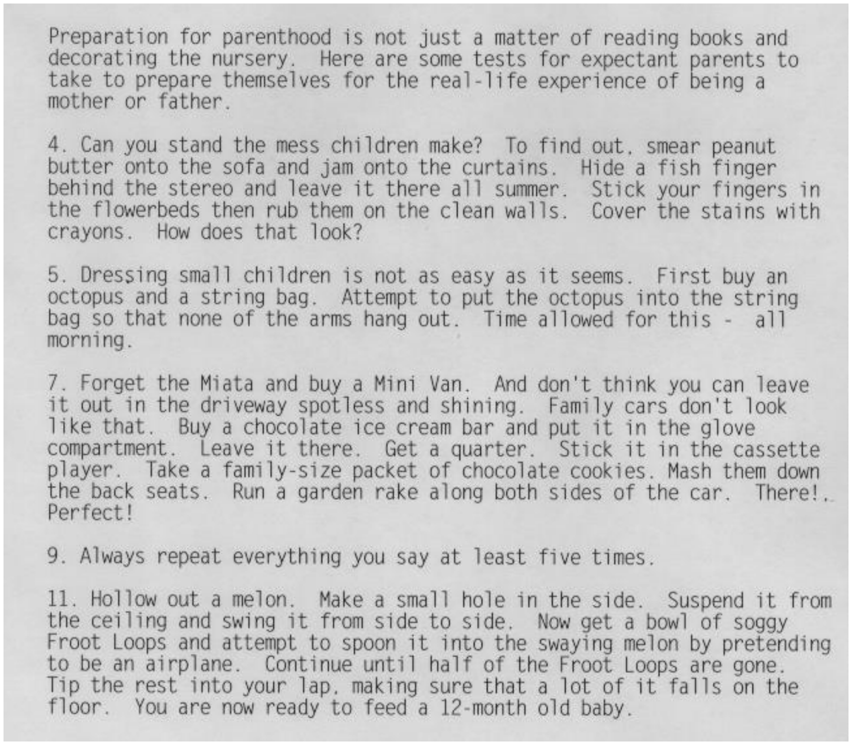
The Mean Spatial Filter image is formed by convolving the image "parenthood.ppm" with the given template "parenthood_e_template.ppm" according to the following formula:

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

Here, I is the image and T is the convolution filter or the template.

The convolution is calculated with the mean zero centered template. A mean zero template is one where the average of all the pixels of the template is subtracted from each pixel of the template. Such a template compensates for variations in brightness (intensity) of the pixels.

The image provided is:



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.

The template is a snippet from this image of the letter "e":

After convolving the image, we get pixel values which are not in the range of 0-255. These need to be normalized to a range of 0-255 to get an 8-bit grayscale image. The following formula is used:

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

where:

$$\begin{aligned}
&newMax = 255\\
&newMin = 0\\
&Min = The\ least\ pixel\ value\ of\ the\ convolved\ image\\
&Max = The\ maximum\ pixel\ value\ of\ the\ convolved\ image\\
&I = Pixel\ value\ of\ non-normalized\ image\\
&I_N = Pixel\ value\ of\ normalized\ image
\end{aligned}$$

The final MSF image obtained is:

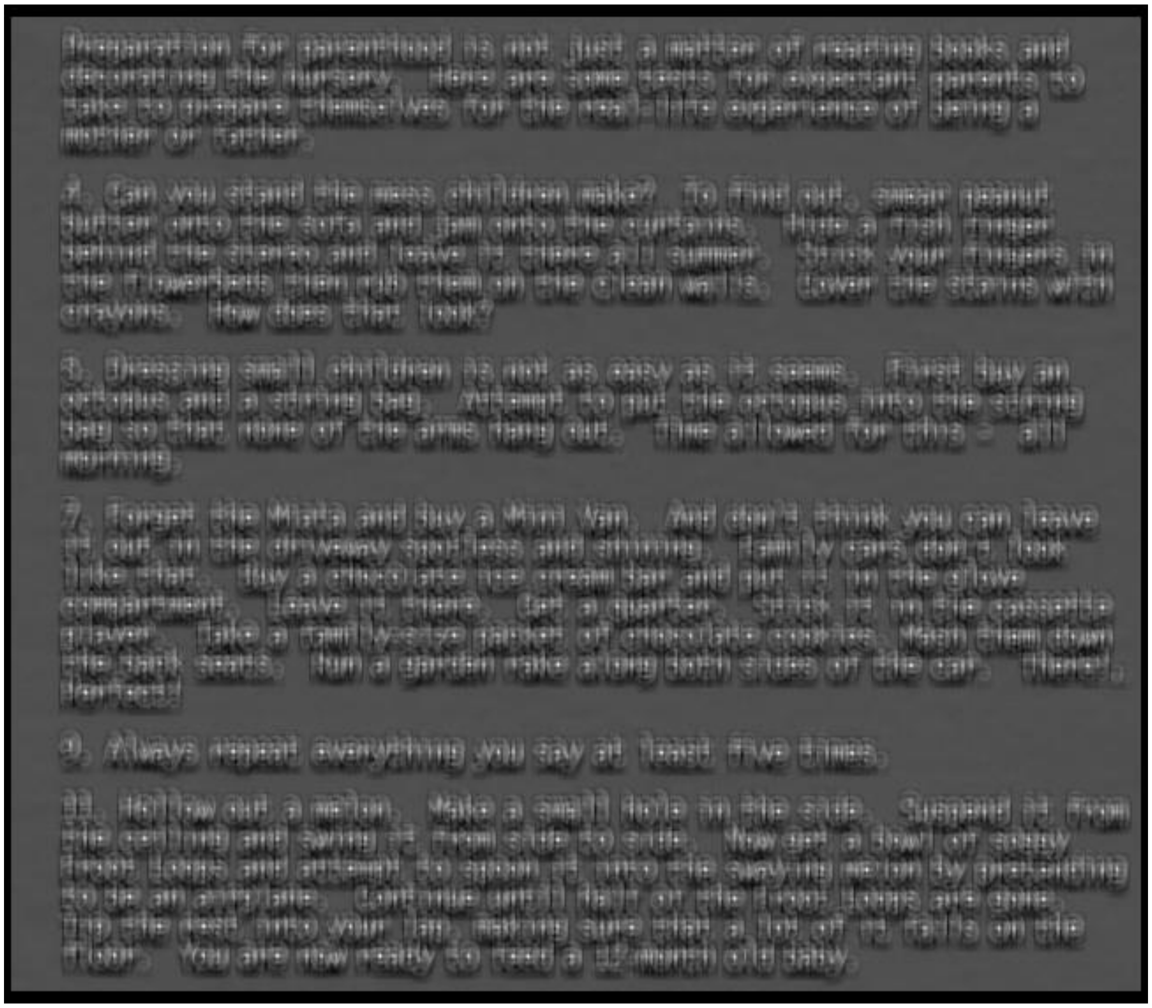


Figure 1. Mean Spatial Filtered Image

The image has black borders around it, which are indicative of the pixels set to zero for edge handling.

Part2:

ROC Curves

Ans:

Receiver Operating Characteristic (ROC) curves are those which characterize the performance of the algorithm in detecting the contents of the provided template. It is plotted with False Positive Rate on the X-axis and True Positive Rate on the Y-axis.

For this part of the lab, a 9x15 window around the locations of the ground truth data was checked and if there existed a pixel above a certain threshold value, it was marked as detected. This creates the data for the system response. For each iteration of the ground truth file, the number of true positives, false positives, true negatives, and false negatives were calculated. From this data the ROC characteristics of True Positive Rate and False Positive Rate were calculated from the following formulae:

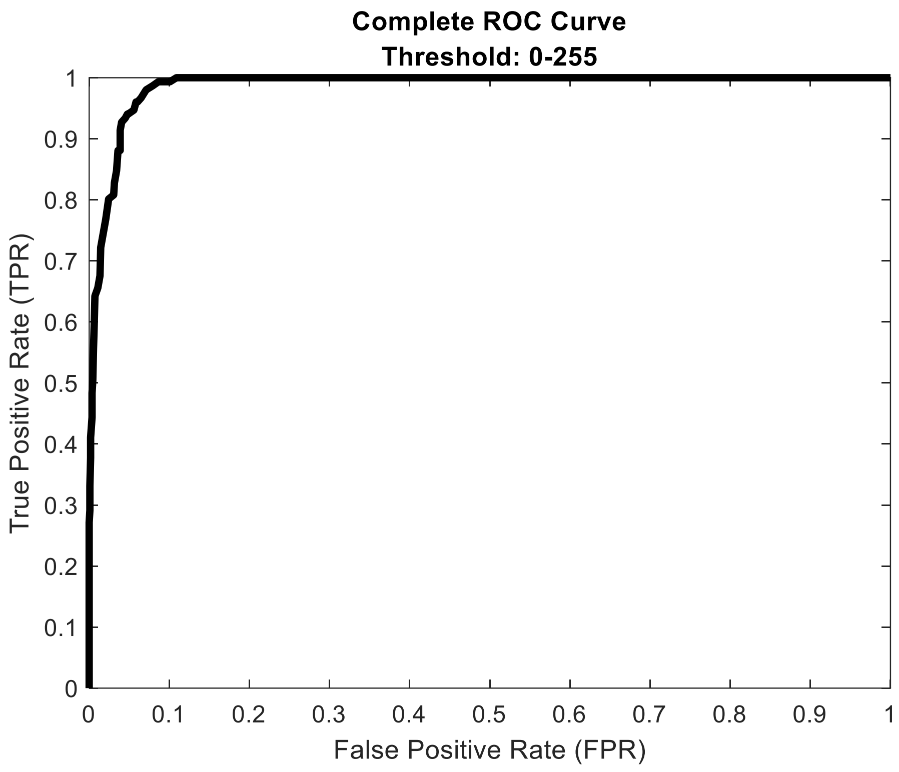
$$TPR = \frac{TP}{TP + FN}$$

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

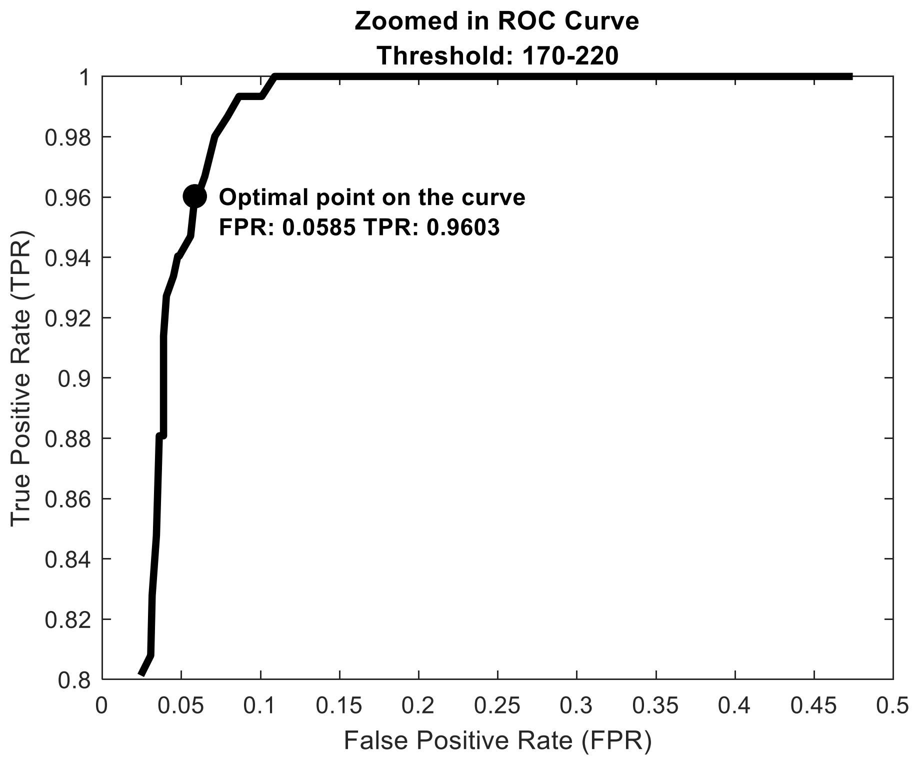
The above methodology was applied from a threshold value from 0 to 255. A value of 0 indicates that all letters are detected and a value of 255 means that nothing will get detected. Binary images were also calculated for these threshold values using the same logic.

$$Binary[r,c] = \begin{matrix} 255 \\ 0 \end{matrix}\left.\vphantom{\begin{matrix} 255 \\ 0 \end{matrix}}\right\} \begin{matrix} MSF[r,c] > T \\ MSF[r,c] \leq T \end{matrix}$$

The full ROC curve is:



The above curve has been zoomed in for threshold values of 170 to 220 for better understanding.



The ideal point for us would be TPR=1 and FPR=0. This indicates that there are only true positive detections for our system. To find the knee of the curve, i.e., the best point for us on the curve, we calculate the distance of every point to the ideal point which is (0,1). The point on the curve with the least distance to that point will be the one which gives us the most optimum threshold.

The values for the optimum threshold and their corresponding TP and FP are:

Threshold	**TP**	**FP**	**TPR**	**FPR**	**TN**	**FN**
208	145	65	0.9603	0.0585	1046	6

As can be seen in the table, we get a TPR of 96% and a FPR of only 5.85%. Thus, the algorithm is robust.

A comparison of the binary images at various thresholds including the ideal threshold of 208 are given for better visual understanding:

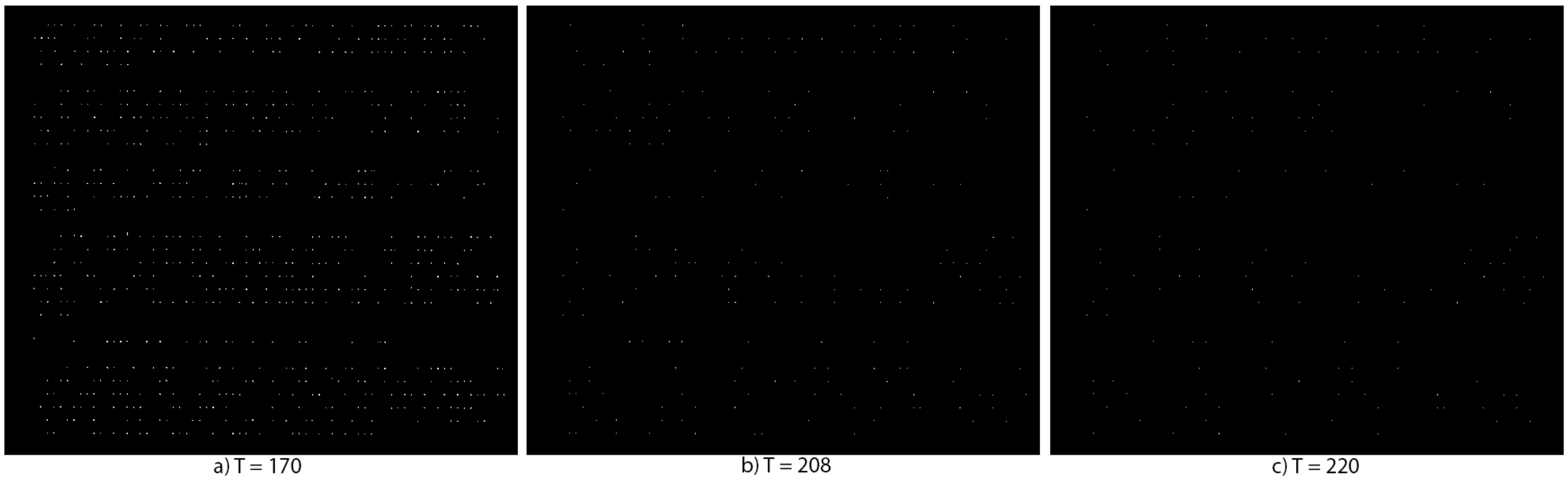

a) T = 170 b) T = 208 c) T = 220

Figure 2. Comparison of binary images at various threshold values

As we can see, for a threshold value of 170, there are a lot of detections in the binary image. These lead to a high number of false positives. Conversely for a value of 220, there are very few detections and there would be a high number of false negatives. As explained above, upon calculating for the optimal point, we get an optimal threshold of 208.

APPENDIX:

C Code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

int main(int argc, char *argv[])
{
  FILE *fpt_temp, *fpt_image, *fpt_gt, *fpt;
  unsigned char *image, *image_temp, *msf, *binary;
  char header_image[80], header_temp[80], template, temp_letter;
  int ROWS_image, COLS_image, BYTES_image, ROWS_temp, COLS_temp, BYTES_temp;
  int row, col, i, m, r, c, r2, c2, sum = 0, tp, tn, fp, fn, det, not_det, tp_array[256], fp_array[256], fn_array[256], tn_array[256], t_array[256];
  double avg = 0.0, *mz_temp, *prelim_msf, max, min, dummy, tpr, fpr, fpr_array[256], tpr_array[256];

  if (argc != 4)
  {
    printf("Usage: HW2 [template] [image] [groundtruth]\n");
    exit(0);
  }

  // Opening the image and reading it into a variable
  fpt_image = fopen(argv[2],"rb");
  if (fpt_image == NULL)
  {
    printf("Unable to open parenthood.ppm for opening\n");
    exit(0);
  }
  i=fscanf(fpt_image,"%s %d %d %d ",header_image,&COLS_image,&ROWS_image,&BYTES_image);
  if (i != 4  ||  strcmp(header_image,"P5") != 0  ||  BYTES_image != 255)
  {
    printf("%s is not an 8-bit PPM greyscale (P5) image\n",argv[2]);
    fclose(fpt_image);
    exit(0);
  }
  image = (unsigned char *)calloc(ROWS_image*COLS_image, sizeof(unsigned char));
  if (image == NULL)
  {
  printf("Unable to allocate %d x %d memory\n",COLS_image,ROWS_image);
  exit(0);
  }
  fread(image, sizeof(unsigned char), ROWS_image*COLS_image, fpt_image);
  fclose(fpt_image);

  // Opening the template and reading it into a file
  fpt_temp = fopen(argv[1],"rb");
  if (fpt_temp == NULL)
  {
    printf("Unable to open parenthood_e_template.ppm for opening\n");
    exit(0);
  }
  i=fscanf(fpt_temp,"%s %d %d %d ",header_temp,&COLS_temp,&ROWS_temp,&BYTES_temp)
;
  if (i != 4  ||  strcmp(header_temp,"P5") != 0  ||  BYTES_temp != 255)
  {
    printf("%s is not an 8-bit PPM greyscale (P5) image\n",argv[1]);
    fclose(fpt_image);
    exit(0);
  }
  image_temp = (unsigned char *)calloc(ROWS_temp*COLS_temp, sizeof(unsigned char)
);
  if (image_temp == NULL)
  {
  printf("Unable to allocate %d x %d memory\n",COLS_temp,ROWS_temp);
  exit(0);
  }
  fread(image_temp, sizeof(unsigned char), ROWS_temp*COLS_temp, fpt_temp);
  fclose(fpt_temp);

  // Calculating the normalized MSF Image:
  // STEP 1: Calculating the zero mean centered template

  for (r = 0; r < ROWS_temp; r++)
  {
    for (c = 0; c < COLS_temp; c++)
    {
      sum = sum + image_temp[(r*COLS_temp)+c];
    }
  }
  double total = ROWS_temp*COLS_temp;
  avg = sum/total;
  mz_temp = (double *)calloc(ROWS_temp*COLS_temp, sizeof(double));
  for (r = 0; r < ROWS_temp; r++)
  {
    for (c = 0; c < COLS_temp; c++)
    {
      mz_temp[(r*COLS_temp)+c] = image_temp[(r*COLS_temp)+c] - avg;
    }
  }

  //STEP 2: Convolving the image with the zero mean centred template
  prelim_msf = (double *)calloc(ROWS_image*COLS_image, sizeof(double));
  int row_lim = ROWS_temp/2; int col_lim = COLS_temp/2;
  for (r = 0; r < ROWS_image; r++)
  {
    for ( c = 0; c < COLS_image; c++)
    {
      dummy = 0.0;
      // Handling edge cases by setting those pixels to zero
      if ( r<row_lim || c<col_lim || r>=(ROWS_image-row_lim) || c>=(COLS_image-
col_lim) ){
        prelim_msf[r*COLS_image+c] = 0.0;
      }
      else{
        for (r2 = -row_lim; r2 <= row_lim; r2++)
        {
          for (c2 = -col_lim; c2 <= col_lim; c2++)
          {
            // Performing the convolution
            dummy = dummy + (image[(r+r2)*COLS_image+(c+c2)]*mz_temp[(r2+row_lim)
*COLS_temp+(c2+col_lim)]);
          }
        }
        prelim_msf[r*COLS_image+c] = dummy;
      }
    }
  }

  // STEP 3: Normalizing the image
  max = 0; min = 99999999;
  for (int j = 0; j < ROWS_image*COLS_image; j++)
  {
    if (prelim_msf[j] > max)
    {
      max = prelim_msf[j];
    }
    if (prelim_msf[j] < min)
    {
      min = prelim_msf[j];
    }
  }
  float factor = max - min; float feed = 0;
  msf = (unsigned char *)calloc(ROWS_image*COLS_image, sizeof(unsigned char));
  for (r = 0; r < ROWS_image; r++)
  {
    for (c = 0; c < COLS_image; c++)
    {
      //This condition just ensures that the pixels omitted in the edge case hand
ling aren't normalized.
      if (r<row_lim || c<col_lim || r>=(ROWS_image-row_lim) || c>=(COLS_image-
col_lim)){
        msf[r*COLS_image+c] = 0;
      }
      else{
        // Normalizing the image
        feed = (255*(prelim_msf[r*COLS_image+c]-min))/(factor);
        int f = round(feed); // Rounding at the last ensures precision isn't lost
 in the arithmetic operations
        msf[r*COLS_image+c] = f;
      }
    }
  }
  // Writing the normalized MSF image into an 8-bit grayscale image
  fpt=fopen("normalized_msf.ppm","wb");
  fprintf(fpt,"P5 %d %d 255\n",COLS_image,ROWS_image);
  fwrite(msf,COLS_image*ROWS_image,sizeof(unsigned char),fpt);
  fclose(fpt);

  // STEP 4: Calculating the ROC Curves
  template = (char)('e'); m = 0;
  for (int T = 0; T <= 255; T++)
  {
    // Creating binary image
    binary = (unsigned char *)calloc(ROWS_image*COLS_image, sizeof(unsigned char)
);


   for (r = 0; r < ROWS_image; r++)
    {
      for (c = 0; c < COLS_image; c++)
      {
        if (msf[r*COLS_image+c] > T) {binary[r*COLS_image+c] = 255;}
        else if (msf[r*COLS_image+c] <= T) {binary[r*COLS_image+c] = 0;}
      }
    }
    tp = tn = fp = fn = 0;
    // Opening the groundtruth file for comparison
    fpt_gt = fopen(argv[3], "rb");
    if ((fpt_gt=fopen(argv[3],"rb")) == NULL)
        {
        printf("Unable to open parenthood_gt.txt for reading\n");
        exit(0);
        }
    while(1)
    {
      det = not_det = 0;
      i = fscanf(fpt_gt, "%s %d %d", &temp_letter, &col, &row);
      if ( i != 3) {break;}
      else
      {
        for (r = row-row_lim; r <= row+row_lim; r++)
      {
        for (c = col-col_lim; c <= col+col_lim; c++)
        {
          if (msf[r*COLS_image+c] > T)
          {
            det = 1;
          }
          else if (msf[r*COLS_image+c] <= T) {not_det = 1;}
        }
      }
      if (det == 1 && temp_letter==template) {tp++;}
      else if (det == 1 && temp_letter!=template) {fp++;}
      else if (not_det == 1 && temp_letter==template) {fn++;}
      else if (not_det == 1 && temp_letter!=template) {tn++;}
      }
    }
    fclose(fpt_gt);
    // Storing the values in an array to export
    t_array[m] = T;
    tp_array[m] = tp;
    fp_array[m] = fp;
    tn_array[m] = tn;
    fn_array[m] = fn;
    tpr_array[m] = tp/(double)(tp + fn);
    fpr_array[m] = fp/(double)(fp + tn);
    m++;
  }

  // Exporting the arrays for plotting
  fpt = fopen("Data.txt","wb");
  fprintf(fpt, "Threshold\tTP\tFP\tTN\tFN\tTPR\tFPR\n");
  for (int j = 0; j < 256; j++)
  {
    fprintf(fpt, "%d\t%d\t%d\t%d\t%d\t%0.7f\t%0.7f\n", t_array[j], tp_array[j], f
p_array[j], tn_array[j], fn_array[j], tpr_array[j], fpr_array[j]);
  }
  fclose(fpt);
}
```

MATLAB Code:

INITIALIZE WORKSPACE

```
clear
close all
clc
```

PLOTTING

```
data = load('Data.txt');
fpr = data(:,7); tpr = data(:,6);
figure(1)
plot(fpr, tpr, 'k', 'LineWidth', 3)
xlabel('False Positive Rate (FPR)'), ylabel('True Positive Rate (TPR)');
title({'Complete ROC Curve','Threshold: 0-255'})
```

FINDING THE KNEE OF THE CURVE

```
thresh_array = data(:,1);
tn_array = data(:,4);
tp_array = data(:,2);
fn_array = data(:,5);
fp_array = data(:,3);
distance = zeros(256,1);
for i = 1:length(tpr)
    distance(i) = sqrt((fpr(i)-0)^2+(tpr(i)-1)^2);
end
mind = min(distance);
best_thresh_idx = find(distance == mind);
```

```
best_fpr = fpr(best_thresh_idx); best_tpr = tpr(best_thresh_idx);
best_thresh = thresh_array(best_thresh_idx);
best_tn = tn_array(best_thresh_idx);
best_tp = tp_array(best_thresh_idx);
best_fn = fn_array(best_thresh_idx);
best_fp = fp_array(best_thresh_idx);
```

PLOTTING

```
figure(2)
plot(fpr(171:221), tpr(171:221), 'k', 'LineWidth', 3)
xlabel('False Positive Rate (FPR)'), ylabel('True Positive Rate (TPR)');
txt1 = 'Optimal point on the curve';
txt2 = 'FPR: %0.4f TPR: %0.4f';
text(best_fpr+0.015, best_tpr, sprintf(txt1), 'color', 'k', 'FontSize', 10,
'FontWeight', 'bold');
text(best_fpr+0.015, best_tpr-0.01, sprintf(txt2, best_fpr, best_tpr), 'color',
'k', 'FontSize', 10, 'FontWeight', 'bold');
hold on
plot(best_fpr, best_tpr, 'ko', 'LineWidth', 5, 'MarkerSize', 5)
hold off
title({'Zoomed in ROC Curve','Threshold: 170-220'})
%xlim([0 1]), ylim([0 1]);
```