Name: Dhruv Thakkar

ECE6310-1,#2

In this project the student had to implement a matched filter (normalized cross-correlation) to recognize the letters in an image of a text. The students were provided with an input image and a template image of desired output character “e” and a ground truth file which was used to determine if the desired letter was recognized from the matched filter image.

The second laboratory was divided into five steps:

1.) Read in the input image, template image, and ground truth file
2.) Calculate the matched-spatial filter (MSF) image
3.) Normalize the MSF image into 8-bits
4.) Loop through following steps for a range of different Thresholds (T):
    a.) Threshold at T the normalized MSF image to create a binary image.
    b.) Loop through the ground truth letter locations.
    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.

We have prepared our c- code based on the following steps and have executed and achieved the desired output. We have also recorded the Normalized MSF-Image, ROC cuRve, Truth table along with the ground truth files & also the perfect threshold in which the found/not found letter ‘e’ can be detected perfectly.

Step 1:

Read in “parenthood.ppm” and “parenthood_e_template.ppm” code:

```
    /* open image for reading */
fpt=fopen("parenthood.ppm","rb");
if (fpt == NULL)
  {
  printf("Unable to open %s for reading\n","parenthood.ppm");
  exit(0);
  }

fpt_tempfile=fopen("parenthood_e_template.ppm","rb");
if (fpt_tempfile == 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 ",IMAGE_HEADER,&IMAGE_COLS,&IMAGE_ROWS,&IMAGE_BYTES);
j=fscanf(fpt_tempfile,"%s %d %d %d ",TEMPLATE_HEADER,&TEMPLATE_COLS,&TEMPLATE_ROWS,&TEMPLATE_BYTES);
if (i != 4 && j != 4  ||  strcmp(IMAGE_HEADER,"P5") != 0 && strcmp(TEMPLATE_HEADER, "P5") != 0  ||  IMAGE_BYTES != 255 && TEMPLATE_BYTES != 255)
  {
  printf("not an 8-bit PPM greyscale (P5) image\n");
  fclose(fpt);
  fclose(fpt_tempfile);
  exit(0);
  }
```

Read in “parenthood_gt.txt” code:

```
    /* open GT file for reading */
fpt=fopen("parenthood_gt.txt","r");
```

Step 2:

Calculation for the Zero-mean template

```
    /*calculattion for the zero mean-centered template*/
sums = 0;
Mean = 0;
ZeroMeanVal = (int *)calloc(TEMPLATE_ROWS*TEMPLATE_COLS,sizeof(int));
for(i=0; i<TEMPLATE_ROWS*TEMPLATE_COLS; i++)
{
        sums += Template[i];
}
Mean = sums / (TEMPLATE_ROWS*TEMPLATE_COLS);
for(i=0; i<TEMPLATE_ROWS*TEMPLATE_COLS; i++)
{
        ZeroMeanVal[i] = Template[i] - Mean;
}

    /*calculation for the msf image*/
for(r=7; r<IMAGE_ROWS-7; r++)
{
  for(c=4; c<IMAGE_COLS-4; c++)
  {
    sum = 0;
    for (r2=-7; r2<=7; r2++)
      for (c2=-4; c2<=4; c2++)
        sum+=(image[(r+r2)*IMAGE_COLS+(c+c2)] * ZeroMeanVal[(r2+7)*TEMPLATE_COLS + (4+c2)]);
   MSF[r*IMAGE_COLS + c] = sum;

}}
```

Step 3:

Calculation for MSF-8bit image

```
    /*calculation for the msf image*/
for(r=7; r<IMAGE_ROWS-7; r++)
{
  for(c=4; c<IMAGE_COLS-4; c++)
  {
    sum = 0;
    for (r2=-7; r2<=7; r2++)
      for (c2=-4; c2<=4; c2++)
        sum+=(image[(r+r2)*IMAGE_COLS+(c+c2)] * ZeroMeanVal[(r2+7)*TEMPLATE_COLS + (4+c2)]);
   MSF[r*IMAGE_COLS + c] = sum;

}}
```

Normalization of the MSF-8bit image

```
    /*Minimum and maximum calculation for normalization*/
min = MSF[0];
max = MSF[0];
for(j=0; j<IMAGE_ROWS*IMAGE_COLS; j++)
{
    if(MSF[j] > max){
    max = MSF[j];}
    if(MSF[j] < min){
    min = MSF[j];}
}
    /*Normaliation of MSF image*/
normalized_msf = (unsigned char *)calloc(IMAGE_ROWS*IMAGE_COLS,sizeof(unsigned char));

for(r=0; r<IMAGE_ROWS*IMAGE_COLS; r++)
{
    normalized_msf[r] = ((MSF[r] - min ) * 255/(max - min));
}

    /*Output print*/
Output_Image = fopen("normalized1.ppm", "w");
    fprintf(Output_Image, "P5 %d %d 255\n", IMAGE_COLS, IMAGE_ROWS);
    fwrite(normalized_msf, IMAGE_COLS * IMAGE_ROWS, sizeof(unsigned char), Output_Image);
    fclose(Output_Image);
```

Step 4:

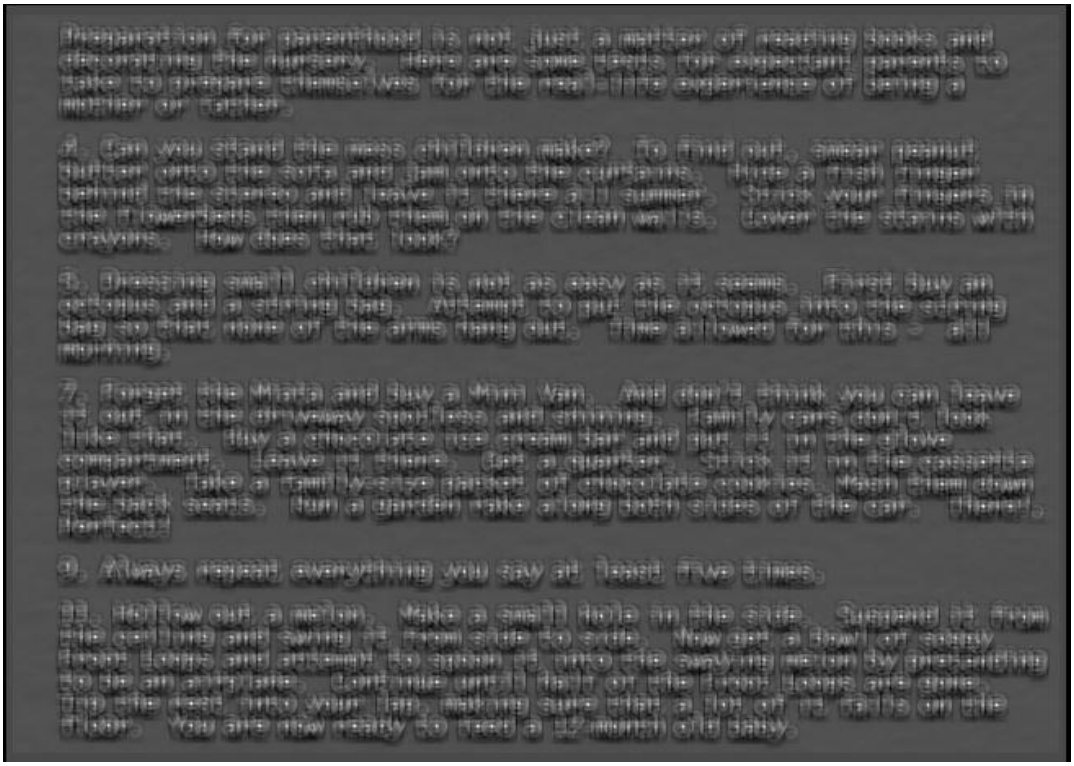
Loop through normalized MSF to calculate Truth Table with different Thresholds code:

```
TP = 0;
FP = 0;
TN = 0;
FN = 0;
while (1)/*checking for points at threshold i*/
{
j=fscanf(fpt,"%s %d %d",gt_letter,&letter_c,&letter_r);
if (j != 3)
  break;
for (r=letter_r-7; r<=letter_r+7; r++)
  {
  for(c = letter_c-4; c <= letter_c+4; c++)
  {
      if(Binary[r*IMAGE_COLS + c] == 255)
      {
          flag = 1;
      }
  }}
```

```
    if(flag == 1 && strcmp(gt_letter, strings) == 0){
    TP++;}
    else if(flag == 1 && strcmp(gt_letter, strings) != 0)
    FP++;
    else if(flag == 0 && strcmp(gt_letter, strings) == 0)
    FN++;
    else if(flag == 0 && strcmp(gt_letter, strings) != 0)
    TN++;
    flag = 0;
    }
  TPR = (double)TP/(double)(TP + FN);
  FPR = (double)FP/(double)(FP + TN);
  if(i%5 == 0){
  fprintf(fpt_tempfile,"%.2lf %.2lf %d\n",TPR, FPR, i);}
  printf("%d\t%d\t%d\n",i, TP, FP);
  rewind(fpt);
}
fclose(fpt_tempfile);
fclose(fpt);
```

Results:

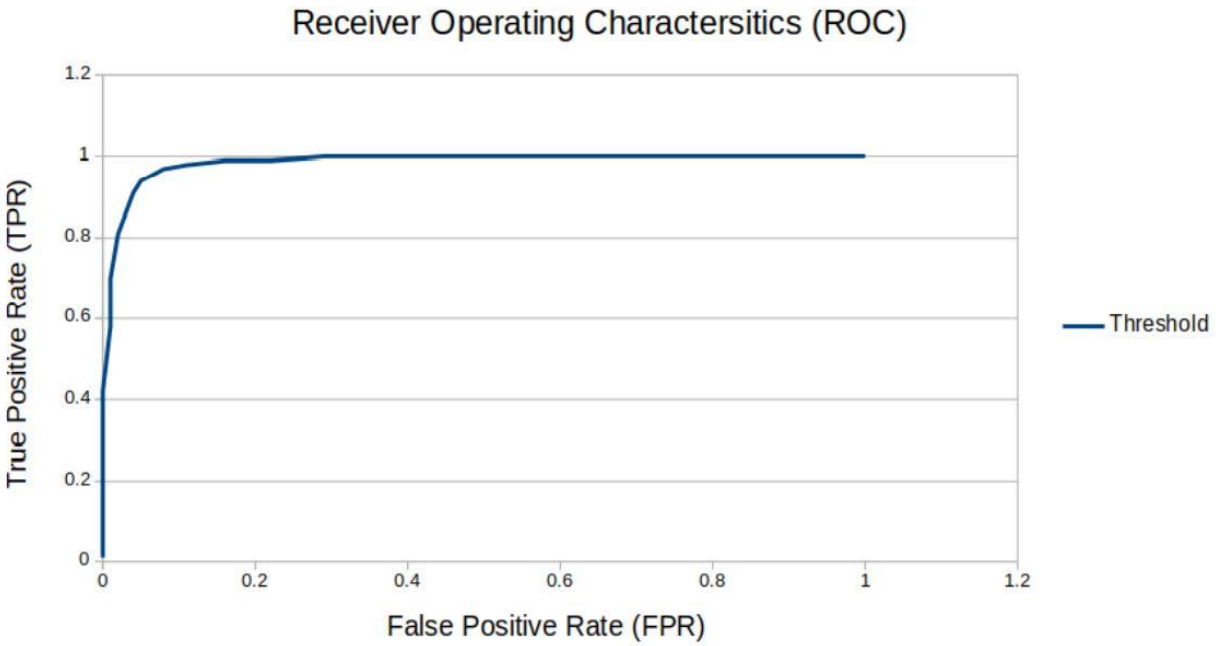
Normalized MSF 8-bit image



TPR and FPR values at different threshold:

Threshold	TPR	FPR	Threshold	TPR	FPR
0	1	1	145	1	0.85
5	1	1	150	1	0.79
10	1	1	155	1	0.69
15	1	1	160	1	0.59
20	1	1	165	1	0.52
25	1	1	170	1	0.47
30	1	1	175	1	0.41
35	1	1	180	1	0.33
40	1	1	185	1	0.26
45	1	1	190	1	0.19
50	1	1	195	1	0.15
55	1	1	200	1	0.1
60	1	1	205	1	0.07
65	1	1	210	1	0.05
70	1	1	215	1	0.04
75	1	1	220	1	0.02
80	1	1	225	1	0.01
85	1	1	230	1	0
90	1	1	235	1	0
95	1	1	240	1	0
100	1	1	245	1	0
105	1	1	250	1	0
110	1	1	255	1	0
115	1	1			
120	1	0.99			
125	1	0.98			
130	1	0.96			
135	1	0.94			
140	1	0.9			

ROC curve:



Based on the ROC graph, the best threshold in which the found/not found letter 'e' ratio is at threshold 210. The following image shows the 'e' found at threshold 210.