ECE 6310

Introduction to Computer Vision

Fall 2022

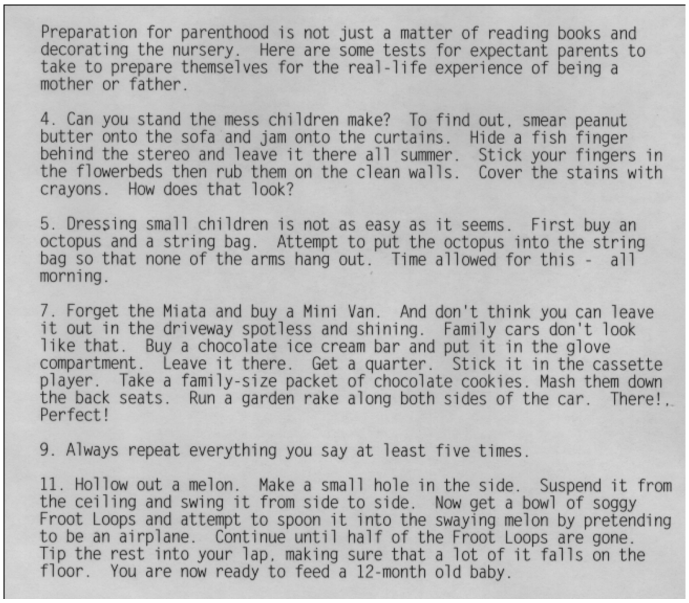
Lab 2

Optical Character Recognition

Harshal B. Varpe

Introduction

In this project/lab, the students were asked to implement a matched filter, otherwise known as normalized cross-correlation or matched spatial filtering, to recognize letters in an image of text. The students were provided an image of a book excerpt, a ground truth text file containing all the letters in the excerpt and their respective pixel coordinates, and a template of a letter to be recognized. The general approach includes looking at each pixel and surrounding area of 9x15 pixels and matching that to the template letter 'e.'



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.

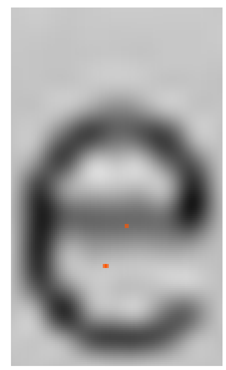


Figure 1: Input image (parenthood.ppm)

Figure 2: Template ‘e’

Implementation

In character recognition problems, it is possible that the template and the input image may not have the same brightness. In such cases, we rely upon matched spatial filtering. In this method, we first create a zero-mean centered template, then we convolve this template with the given image. In the last step, we find a suitable threshold to find actual matches. The equation to create a matched spatial filter is given as –

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

During the process's convolution step, the pixels' range increases greater than 8-bit. Hence the result of such convolution cannot be stored in unsigned characters. Instead, we store the result of convolution in an array of int. However, to get back to the 8-bit format, we normalize. A general equation to normalize the given data is –

$$X_{normalized} = (X-X_{min}) *255 / (X_{max}-X_{min})$$

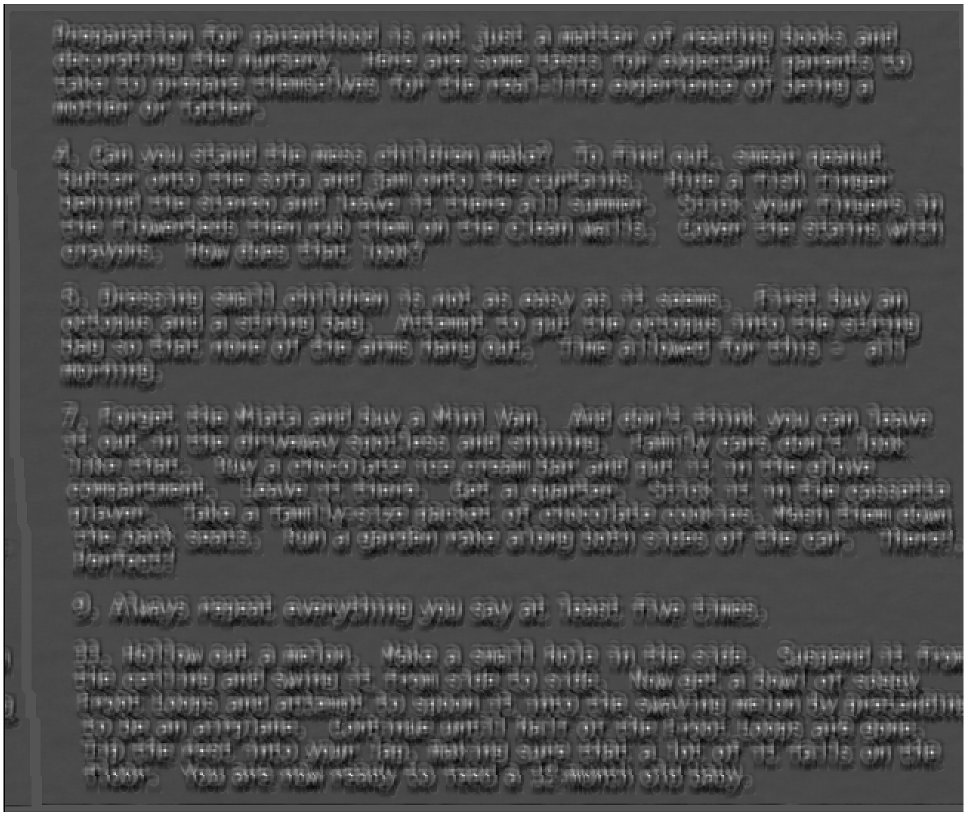


Figure 3: Normalized Image

We can see the tiny bright pixels. These bright pixels are all the possible locations of the letter 'e.' There is a problem here – How to be sure that the bright spot is the letter 'e'? It is possible that some of the bright spots are not the letter 'e.' Here, we rely on thresholding and satisfying a condition where we check if the letter from ground truth and letter 'e' match.

The possible values for threshold T range from 0 to 255. For a single value of threshold T, we create a binary image. If a pixel value in a normalized MSF image is above the said threshold, then for the same pixel in a binary image value is set to 255. Otherwise, the value is set to 0. Then, if the pixel value in binary image is above threshold T for a specific range of pixels, then we set the variable "detected" to 1. Otherwise, we set the variable "non_detected" value to 1. Along with the variables "detected" and "non-detected," If the given letter, in our case the letter 'e,' and the letter read from the ground truth file match, we calculate True Positive, True Negative, False Positive, False Negative, and True Positive Rate (TPR), and False Positive Rate (FPR). We create a ROC curve from TPR and FPR.

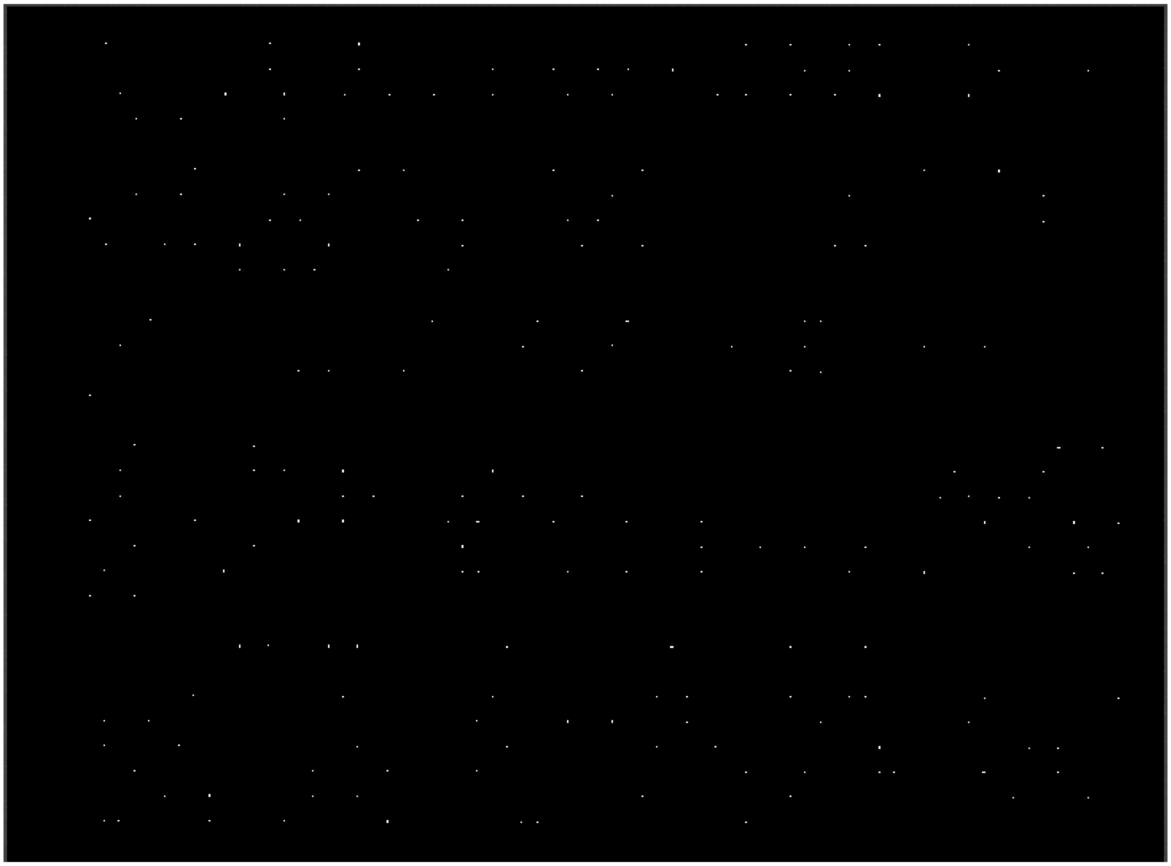


Figure 4: Binary Image at Threshold 206

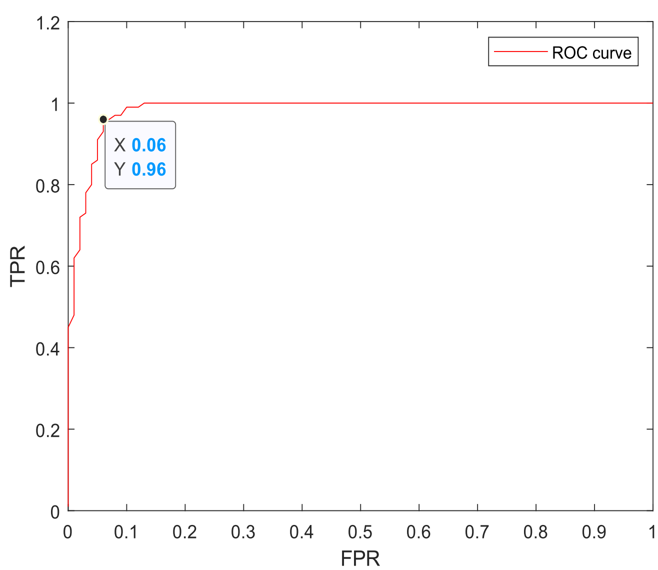


Figure 5: ROC curve

The figure above shows that the behavior of the ROC curve TPR (0.96) and FPR (0.06) changes rapidly. This point is known as the knee of the ROC curve. In an average case like ours, we can select the knee of the curve as the best tradeoff performance. The Threshold value at the knee is 206. At the optimal threshold, there were 142 True positives and 67 False positives.

Code

```
// ECE 6310 Introduction to CV
// Lab 2 Fall 22
// Harshal Varpe
// Clemson University
# include <stdio.h>
# include <stdlib.h>
# include <string.h>

int main (int argc, char *argv[]){
    FILE *fpt, *temp_file, *g_truth, *out;
    int r,c,gt_row,gt_col,detected,not_detected,T,TP,FP,FN,TN,i,j;
    int min,max,sum,r1,c1,ROWS,COLS,BYTES,temp_R, temp_C,temp_B;
    char header[320],gt_letter[320],letter;
    unsigned char *input,*template_img,*img_msf_norm,*final, *binary;
    float TPR,FPR,mean; int *temp_img;
    double *img_msf;

if(argc != 2){printf("Wrong number or arguments! \n Usage : [executable_name]
[letter]");
exit(0);}
letter = argv[1][0];

/* Task 1 - Create a Zero mean centered template from the template image */
fpt = fopen("parenthood.ppm","rb");
if (fpt == NULL)
  {
  printf("Unable to open %s for reading\n","parenthood.ppm");
  exit(0);
  }
fscanf(fpt,"%s %d %d %d",header,&COLS,&ROWS,&BYTES);
input = (unsigned char *)calloc(ROWS*COLS,sizeof(unsigned char));
fread(input,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);
  }
fscanf(fpt,"%s %d %d %d",header,&temp_C,&temp_R,&temp_B);
template_img = (unsigned char *)calloc(temp_R*temp_C,sizeof(unsigned char));
fread(template_img,1,temp_R*temp_C,fpt);
```

```c
fclose(fpt);
// printf("%d this is temp_r \n",temp_R);
// printf("%d \n",template_img[12]);
/* Calculating a ZMC template */
mean = 0;
for(j=0;j<(temp_R*temp_C);j++){
    mean += (float)template_img[j];
}
mean = mean / (float)(temp_R * temp_C);
// printf("%d",(int)mean);
/* We will store the zmc template.*/
temp_img = (int *)calloc(temp_R*temp_C,sizeof(int));
// temp_img = (double *)calloc(temp_R*temp_C,sizeof(double));
// temp_img = (float *)calloc(temp_R*temp_C,sizeof(float));
for(j=0;j<(temp_R*temp_C);j++){
    temp_img[j] = (int)((float)template_img[j] - mean);
}
// for(int l=0;l<ROWS*COLS;l++){printf("%d \n",temp_img[l]);}
// printf("%f is mean \n",mean);
// /* Lets us create an MSF version of original image. */
img_msf = (double *)calloc(ROWS*COLS,sizeof(double));
for(r = (temp_R/2);r<=(ROWS-(temp_R/2));r++){
    for(c = (temp_C/2);c<=(COLS-(temp_C/2));c++){
        sum=0;
        for(r1 = -(temp_R/2);r1<=(temp_R/2);r1++){
            for(c1 = -(temp_C/2);c1<=(temp_C/2);c1++){
                sum += (temp_img[(r1+(temp_R/2))*temp_C+(c1+(temp_C/2))] *
input[(r+r1)*COLS+(c+c1)]);
            }
        }
        img_msf[r*COLS+c] = sum;
    }
}

/* normalization of the msf image */
min =  5000; max = 0;
for(j=0;j<ROWS*COLS;j++){
    if(img_msf[j]>max){max = img_msf[j];}
    else if(img_msf[j]<min){min = img_msf[j];}
}
// printf("%d is min \n",min);
// printf("%d is max \n",max);

img_msf_norm = (unsigned char *)calloc(ROWS*COLS,sizeof(unsigned char));
for(j=0;j<(ROWS*COLS);j++){
```

```
    img_msf_norm[j] = ((img_msf[j]-min)*255)/(max-min);
}

fpt = fopen("normalized_msf.ppm","w");
fprintf(fpt,"%s %d %d 255",header,COLS,ROWS);
fwrite(img_msf_norm,1,ROWS*COLS,fpt);
fclose(fpt);


// /* creating a text file to write the threshold and ROC curve data*/
out  = fopen("confusion_mat.txt","a");
fprintf(out,"T TP FP FN TN TPR FPR \n");
// printf("%c \n",letter);
for(T=0;T<=255;T++){
    // for point 4.a
    binary = (unsigned char *)calloc(ROWS*COLS,sizeof(unsigned char));
    for(j=0;j<(ROWS*COLS);j++){
        if (img_msf_norm[j]>=T){binary[j]=255;}
        else{binary[j]=0;}
    }

    // Although in the
    g_truth = fopen("parenthood_gt.txt","rb");
    TP=TN=FP=FN=0;
    while(1){

        i = fscanf(g_truth,"%s %d %d",gt_letter,&gt_col,&gt_row);
        detected = not_detected = 0;
        if(i!=3){break;}
        else {
            for(r=gt_row-7;r<=gt_row+7;r++){
                for(c=gt_col-4;c<=gt_col+4;c++){
                    if(binary[r*COLS+c]>=T){detected = 1;}
                    else{not_detected = 1;}
                }
            }
            if ((detected == 1) && (*gt_letter == letter)){TP += 1;}
            else if ((detected == 1) && (*gt_letter != letter)){FP += 1;}
            else if ((not_detected == 1) && (*gt_letter == letter)){FN += 1;}
            else if ((not_detected == 1) && (*gt_letter != letter)){ TN +=1 ;}
        }
    }
    // printf("%c is gt \n", gt_letter);
    // printf("%c is input \n",letter);
    TPR = (float)TP / (float)(TP+FN);
```

```
    FPR = (float)FP / (float)(FP+TN);
    fprintf(out,"%d %d %d %d %d %f %f\n",T,TP,FP,FN,TN,TPR,FPR);
    fclose(g_truth);
    // printf("%d",T);
}
fclose(out);
printf("This is the end my friend!");
/* binary image with chosen threshold */
// The variable final is used to store a binary image
T = 206; // Knee point of ROC
final = (unsigned char *)calloc(ROWS*COLS,sizeof(unsigned char));
for(j=0;j<(ROWS*COLS);j++){
        if (img_msf_norm[j]>=T){final[j]=255;}
        else{final[j]=0;}
    }
fpt = fopen("final_output.ppm","w");
fprintf(fpt,"P5 %d %d 255",COLS,ROWS);
fwrite(final,1,ROWS*COLS,fpt);
fclose(fpt);
}
```