CLEMSON UNIVERSITY

ECE 6310: INTRODUCTION TO COMPUTER VISION

Lab 2

C14109603

Rahil Modi
rmodi@g.clemson.edu

1. Introduction

In this Lab we were asked to detect a letter from the entire parenthood.ppm with the help of a template image of the letter e which we will use to scan over every pixel and compare the surroundings pixels with the template and if the pixels match it can be detected as the letter we are looking for.

2. Implementation

To match a pixel with a template first we need to use a Zero Mean filter on the template in which the mean of the pixel values of the template is calculated and subtracted from every pixel for the template.

Now the parenthood.ppm image and filtered template is convoluted to create a Matched Spatial Filter (MSF), due to convolution the pixels values go above the normal 8- bit images so we need to normalize the image after the convolution. There are many ways to normalize the images but the equation which I found online is:

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

After the normalization we can see the image with white dots where the letter e located and thresholding the MSF image value of the pixels the letter e pixels are highlighted with white dots.

To find the threshold I read through the ground truth text file provided and looped from 0 – 255 and set every value as the threshold on the MSF image and at each coordinate a 9x15 area was looked to see if the pixels matched with the template and check that with the help of if the value was equal to 255 if yes than it was detected as e. That would be crosschecked with the ground truth and if it was e it would be True Positive (TP), if it was detected as e but did not match with the ground truth if would be False Positive (FP), if the letter was not detected as e and the ground truth said the same thing it will be a False Positive (FP) and if it detected as not e and it was a e according to the ground truth then it is a False Negative (FN)

Total of the TP, TN, FP and FN were exported to a csv file for storing and using these values the ROC curve was calculated and the best threshold value was calculated with the help of the distance to the corner where it was 1 for TP and 0 for FP rate.

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

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

$$Sensitivity = TPR$$

$$Specifity = 1 - FPR$$

3. Results:

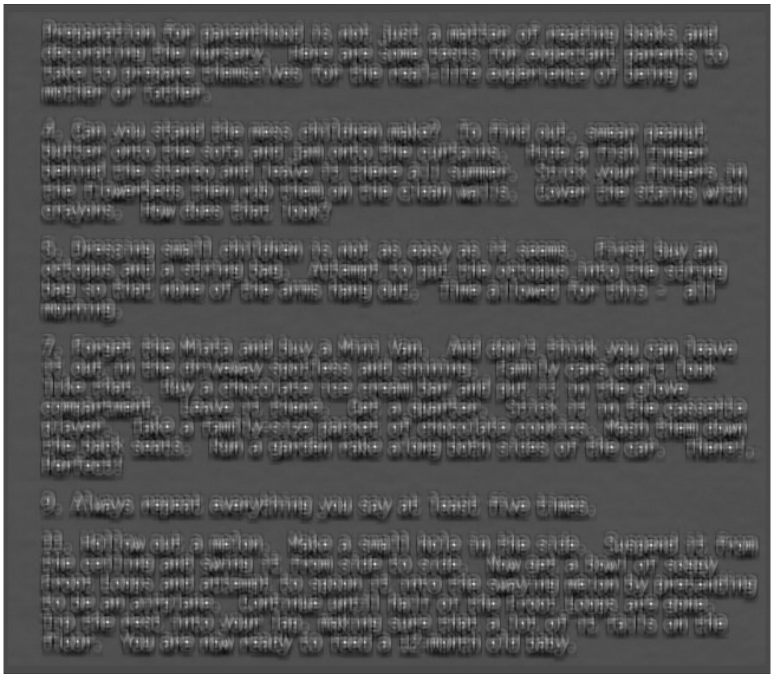


Figure 1 Matched Spatial Filtered Image

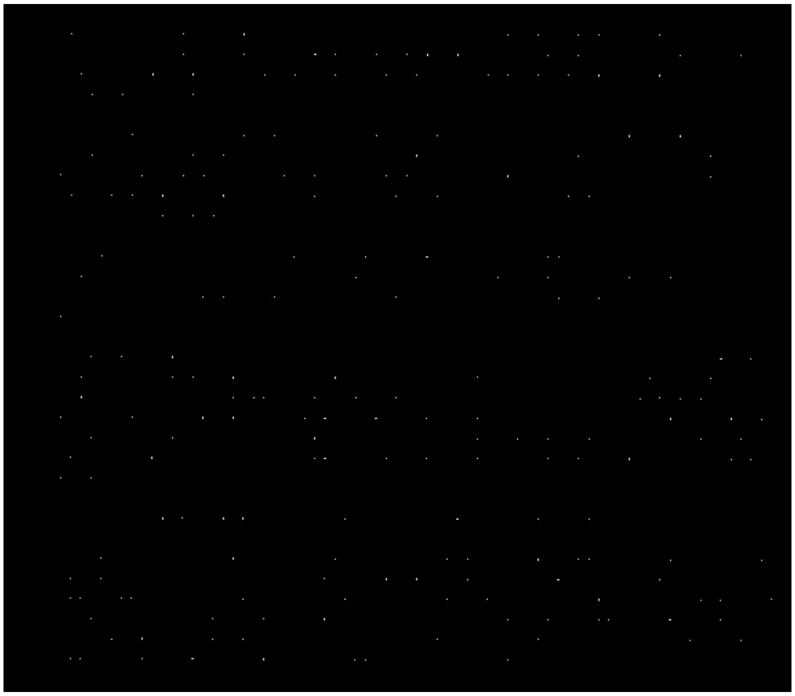


Figure 2 Binary Image with white dots as letter e at threshold 206.

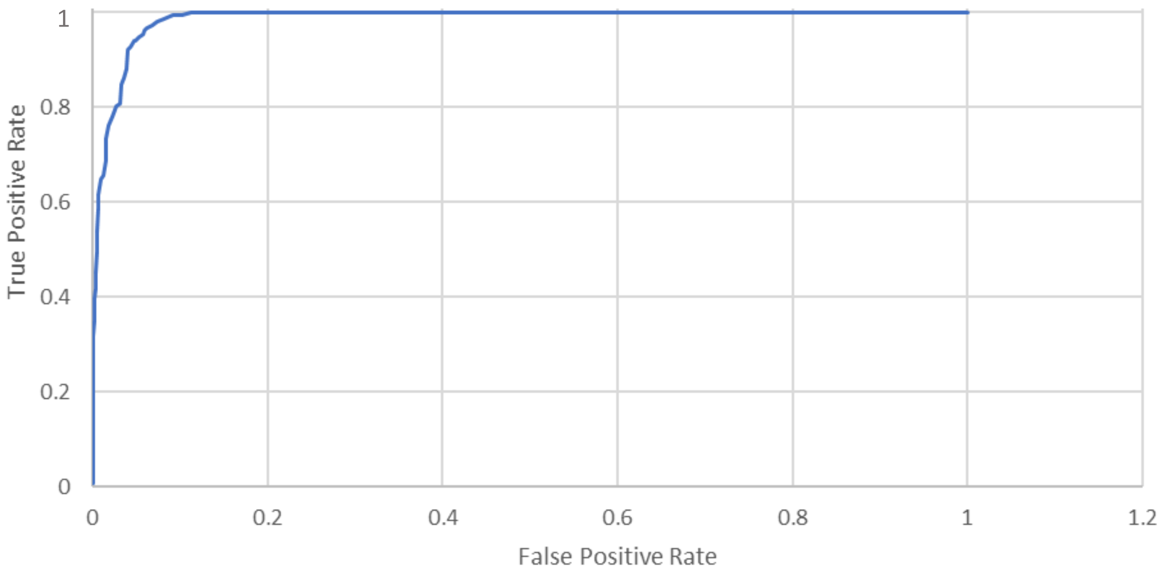
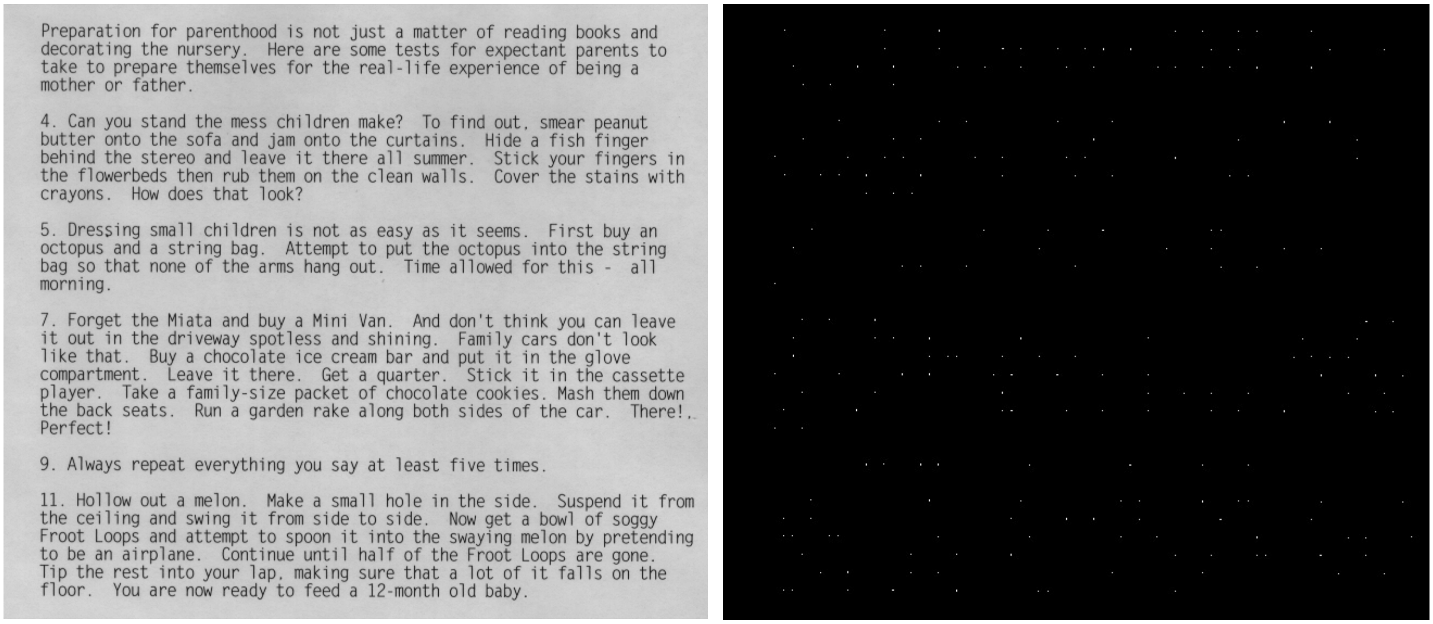


Figure 3 ROC curve

The best determined threshold value was 206 with **TP =146 and FP = 69**. The distance at the corner at (0,1) on the curve for threshold = 206 is 0.0703.



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 4 Comparison to show input image next to binary image

4. Appendix:

```c
/* Rahil Modi
 * Lab2 Optical Character Recognition
 * In this lab we are asked to develop a code
 * that will recognition a particular letter from a image.
 * This program will take a image full of text,
 * a template of specific letter and ground truth of all the letter
 * and detect the position of the specific letter.
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
struct coordinates_s
{
  int  column;
  int  row;
  char letter;
};

void Zero_Mean(int *dummy_image, int ROWS, int COLS)
{
  int rc;
  int sum = 0;
  int mean;
  for (rc = 0; rc < ROWS*COLS; rc++)
  {
    sum += dummy_image[rc];
  }
  printf("The sum of the image is = %d\n", sum);
  mean = sum / (ROWS*COLS);
  printf("The mean of the image is = %d\n", mean);
  for (rc = 0; rc < ROWS*COLS; rc++)
  {
    dummy_image[rc] -= mean;
  }
  return;
}

void normalize(int *dummy_image, int ROWS, int COLS)
{
  int i, max, min;

  min = dummy_image[0];
  max = dummy_image[0];

  for (i = 1; i < ROWS * COLS; i++)
  {
    if (dummy_image[i] > max)
    {
      max = dummy_image[i];
    }
    if (dummy_image[i] < min)
    {
      min = dummy_image[i];
    }
  }
  printf("The min is = %d\n", min);
  printf("The max is = %d\n", max);
  /* normalization equation found online wikipedia */
  for (i = 0; i < ROWS*COLS; i++)
  {
    dummy_image[i] = (dummy_image[i] - min)*255/(max - min);
  }

  return;
}

void threshold(unsigned char *image, unsigned char *dummy_image, int ROWS, int COLS, int threshold_value)
{
  int i;
  for (i = 0; i < ROWS*COLS; i++)
  {
    if (image[i] > threshold_value)
    {
    dummy_image[i] = 255;
    }
    else
    {
    dummy_image[i] = 0;
    }
  }
  return;
}

void M_S_F(unsigned char *image, unsigned char *template, int ROWS, int COLS, int t_ROWS, int t_COLS, unsigned char *MSF)
{
  int           i, r, c, r2, c2, sum;
  int           *dummy_template;
  int           *dummy_image;
  unsigned char *dummy;
  FILE          *fpt;

  /* allocating memory to the dummy image and the dummy template */
  dummy_template = (int *)calloc(t_ROWS * t_COLS, sizeof(int));
  dummy_image    = (int *)calloc(ROWS * COLS, sizeof(int));
  dummy          = (unsigned char*)calloc(ROWS*COLS,sizeof(unsigned char));

  /* Transfering template to a dummy template to avoid errors */
  for (i = 0; i < t_ROWS*t_COLS; i++)
  {
    dummy_template[i] = (int)template[i];
  }

  /* Function to calculate the Zero Mean of the image */
  Zero_Mean(dummy_template, t_ROWS, t_COLS);

  /* Convolution of the image */
  for (r = 7; r < (ROWS - 7); r++)
  {
    for (c = 4; c < (COLS - 4); c++)
    {
      sum = 0;
      for (r2 = -7; r2 < t_ROWS-7; r2++)
      {
        for (c2 = -4; c2 < t_COLS - 4; c2++)
        {
          sum += image[(r+r2)* COLS + (c+c2)] * dummy_template[(r2+(t_ROWS/2)) * t_COLS + (c2+(t_COLS/2))];
        }
      }
      dummy_image[r * COLS + c] = sum;
    }
  }

  /* normalize the image */
  normalize(dummy_image, ROWS, COLS);

  for (i = 0; i < ROWS*COLS; i++)
  {
    MSF[i] = (unsigned char)dummy_image[i];
  }
```

```c

  /* writing the image with best threshold which will show the letter e as white dots */
  threshold(MSF, dummy, ROWS, COLS, 206);
  fpt=fopen("D:/Computer_Vision/Lab2/binary.ppm","w");
  fprintf(fpt,"P5 %d %d 255\n",COLS,ROWS);
  fwrite(dummy,COLS*ROWS,1,fpt);
  fclose(fpt);

  free(dummy_template);
  free(dummy_image);
  return;
}

int main()
{
  FILE          *fpt, *fpt1;
  unsigned char *template;
  unsigned char *image, *dummy;
  char           header[320];
  int            ROWS, COLS, BYTES;
  char           t_header[320];                 // Header for the template
  int            t_ROWS, t_COLS, t_BYTES;       // Rows, Columns and Bytes for the template
  unsigned char  *MSF;
  int            threshold_value;
  bool           result;
  int            tp, fp, fn, tn,r,c, iter, i, tpr, fpr, sensitivity, specifity;
  struct coordinates_s *coordinates;

  /* Read Image */
  if ((fpt=fopen("D:/Computer_Vision/Lab2/parenthood.ppm","rb")) == NULL)
  {
    printf("Unable to open image for reading.\n");
    exit(0);
  }

  fscanf(fpt, "%s %d %d %d", header, &COLS, &ROWS, &BYTES);
  if (strcmp(header, "P5") != 0 || BYTES != 255)
  {
    printf("Not a greyscale 8-bit PPM image\n");
    exit(0);
  }

```

```
    image = (unsigned char *)calloc(ROWS*COLS, sizeof(unsigned char));
    header[0] = fgetc(fpt); /* read white-space character that separates header */
    fread(image, 1, COLS*ROWS, fpt);
    fclose(fpt);

    /* Read the template*/
    if ((fpt=fopen("D:/Computer_Vision/Lab2/parenthood_e_template.ppm","rb")) == NULL)
    {
      printf("Unable to open image for reading.\n");
      exit(0);
    }

    fscanf(fpt, "%s %d %d %d", t_header, &t_COLS, &t_ROWS, &t_BYTES);
    if (strcmp(t_header, "P5") != 0 || t_BYTES != 255)
    {
      printf("Not a greyscale 8-bit PPM image\n");
      exit(0);
    }

    template = (unsigned char *)calloc(ROWS*COLS, sizeof(unsigned char));
    header[0] = fgetc(fpt); /* read white-space character that separates header */
    fread(template, 1, t_COLS*t_ROWS, fpt);
    fclose(fpt);

    /* allocate memory for the Matched Spatial Filtering */
    MSF = (unsigned char *)calloc(ROWS*COLS, sizeof(unsigned char));
    M_S_F(image, template, ROWS, COLS, t_ROWS, t_COLS, MSF);

    /* write out smoothed image to see result */
    fpt = fopen("D:/Computer_Vision/Lab2/MSF.ppm", "wb");
    fprintf(fpt, "P5 %d %d 255\n", COLS, ROWS);
    fwrite(MSF, COLS*ROWS, 1, fpt);
    fclose(fpt);

    /* opening the ground truth */
    fpt = fopen("D:/Computer_Vision/Lab2/parenthood_gt.txt", "rb");
    if (fpt == NULL)
    {
      printf("Error opening ground truth\n");
      exit(0);
    }

```

```
  dummy = (unsigned char *)calloc(ROWS*COLS, sizeof(unsigned char));
  iter = 0;
  coordinates = (struct coordinates_s*)calloc(1300,sizeof(struct coordinates_s));
  while (!feof(fpt))
  {
    fscanf(fpt, "%c %d %d\n", &coordinates[iter].letter, &coordinates[iter].column, &coordinates[iter].row);
    iter++;
  }
  for (threshold_value = 0; threshold_value < 256; threshold_value++)
  {
    threshold(MSF, dummy, ROWS, COLS, threshold_value);

    fpt1 = fopen("D:/Computer_Vision/Lab2/ROC.csv", "w");
    fprintf(fpt1, "Threshold,TP,FP,TN,FN,TPR,FPR,Sensitivity,Specificity\n");
    fpt1 = fopen("D:/Computer_Vision/Lab2/ROC.csv", "a");

    tp = 0;
    fp = 0;
    tn = 0;
    fn = 0;

  // Loop through the ground truth data
    for (i = 0; i < iter; i++)
    {
      result = false;
    // Loop through 9x15 area around the coordinates from the ground truth data
      for (r = coordinates[i].row - 7; r <= coordinates[i].row + 7; r++)
      {
        for (c = coordinates[i].column - 4; c <= coordinates[i].column + 4; c++)
        {
          if (dummy[r*COLS+c] == 255)
          {
            result = true;
          }
        }
      }
      if (result == true && coordinates[i].letter == 'e')
      {
        tp++;
      }
      else if (result == true && coordinates[i].letter != 'e')
      {
        fp++;
      }
      else if (result != true && coordinates[i].letter == 'e')
      {
        fn++;
      }
      else if (result != true && coordinates[i].letter != 'e')
      {
        tn++;
      }
    }
    tpr = tp/(tp+fn);
    fpr = fp/(fp+tn);
    sensitivity = tpr;
    specifity = 1 - fpr;
    fprintf(fpt1, "%d,%d,%d,%d,%d,%d,%d,%d,%d\n", threshold_value, tp, fp, tn, fn, tpr, fpr, sensitivity, specifity);
  }
  fclose(fpt1);

 return(1);
}
```