**ECE 4310/6310 Introduction to Computer Vision**

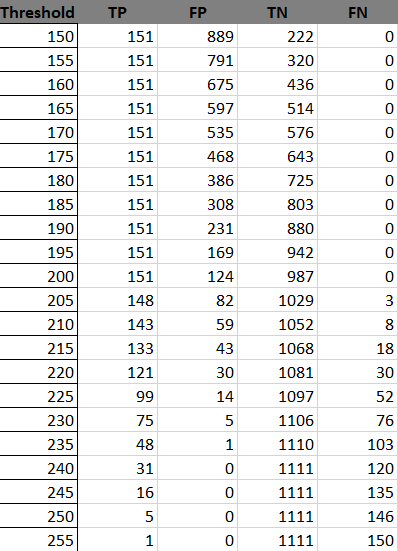
**Lab #2 – Optical Character Recognition**

**Name- Pinak Kelkar**

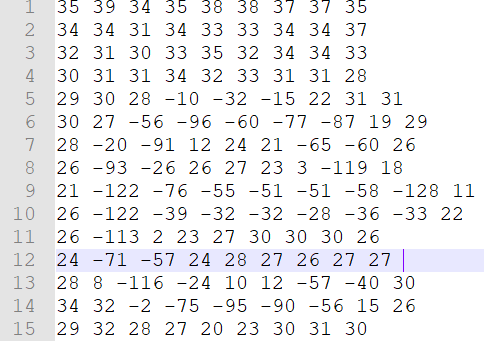
First the MSF filter was calculated and with which you can see below.

After that the we start putting the filter on the real image and then I have used the maximum value to normalize the image in 8-bit format by using this formula (Value-min/ Max-min) \*255. I have used a in Array to hold all the large numbers and then normalize hem in 8-bit format

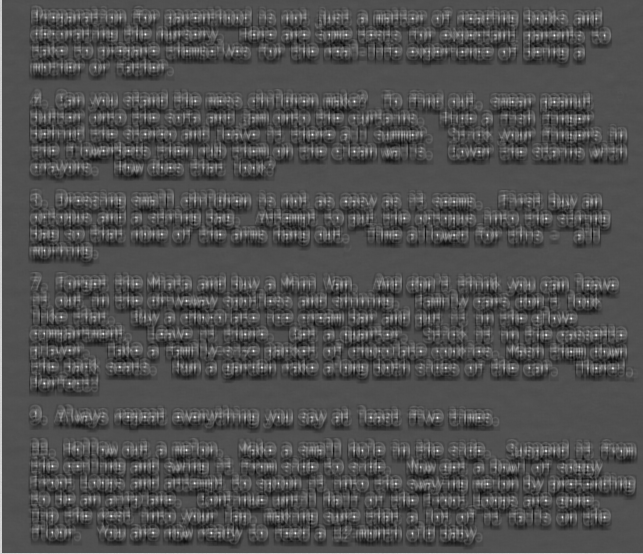
This is the truth table True Positive, False Positive True Negative and False Negative Calculation for all the thresholds from 150 to 255. So according to the ROC curve we can see that the optimal threshold for this image for detecting ‘e’ would be between 200 to 220. According to the truth table there are 151 ‘e’ in the image, we can detect all of them at a threshold of 175 but with a false negative of 139 detections.

****

MSF Filter Template



MSF Image



#include <stdio.h>

#include <stdlib.h>

#include <stdint.h>

#include <string.h>

int main (int argc, char \*argv[])

{

if (argc!=2)

{

printf("Enter Threshold");

return 0;

}

FILE \*fpt2,\*fpt,\*ftr;

unsigned char \*template;

unsigned char \*image;

int \*MSF;

int \*MSF\_Image;

char header[80];

int ROWS,COLS,BYTES;

char T\_header[80];

int T\_ROWS,T\_COLS,T\_BYTES;

int i,j,Threshold=atoi(argv[1]);

int r=0,c=0;

int sum=0,total=0;

int max=0,min=0;

int counter=0;

int r1=0,c1=0;

char \* line = NULL;

size\_t len = 0;

ssize\_t read;

fpt2=fopen("parenthood\_e\_template.ppm","r");

fpt=fopen("parenthood.ppm","r");

ftr=fopen("parenthood\_gt.txt","r");

if (fpt2 == NULL || fpt == NULL) {

printf("Unable to open for reading\n");

exit(0);

}

i=fscanf(fpt2,"%s %d %d %d",T\_header,&T\_COLS,&T\_ROWS,&T\_BYTES);

j=fscanf(fpt,"%s %d %d %d",header,&COLS,&ROWS,&BYTES);

/\* allocate dynamic memory for image \*/

template=(unsigned char \*)calloc(T\_ROWS\*T\_COLS,sizeof(unsigned char));

image=(unsigned char \*)calloc(ROWS\*COLS,sizeof(unsigned char));

MSF=(int \*)calloc(T\_ROWS\*T\_COLS,sizeof(int));

MSF\_Image=(int \*)calloc(ROWS\*COLS,sizeof(int));

if (template == NULL || image==NULL || MSF==NULL) {

printf("Calloc Failed\n");

exit(0);

}

T\_header[0]=fgetc(fpt2);

fread(template,1,T\_ROWS\*T\_COLS,fpt2);

header[0]=fgetc(fpt);

fread(image,1,ROWS\*COLS,fpt);

fclose(fpt2);

fclose(fpt);

///creating MSF Image

for(r=0;r<T\_ROWS;r++){

for(c=0;c<T\_COLS;c++){

sum+=template[r\*T\_COLS+c];

}

}

total=(T\_COLS\*T\_ROWS);

sum=sum/total;

for(r=0;r<T\_ROWS;r++){

for(c=0;c<T\_COLS;c++){

MSF[r\*T\_COLS+c]=template[r\*T\_COLS+c]-sum;

}

}

i=(T\_ROWS/2);

j=(T\_COLS/2);

sum=0;

for(r=i;r<ROWS-i;r++){

for(c=j;c<COLS-j;c++){

int i1=0,j1=0;

for(r1=-i;r1<=i;r1++){

j1=0;

for(c1=-j;c1<=j;c1++){

sum+=(MSF[i1\*T\_COLS+j1]\*image[(r+r1)\*COLS+(c+c1)]);

j1++;

}

i1++;

}

if(max<sum){

max=sum;

}

if(min>sum){

min=sum;

}

MSF\_Image[r\*COLS+c]=sum;

sum=0;

}

}

for(r=i;r<ROWS-i;r++){

for(c=j;c<COLS-j;c++){

image[r\*COLS+c]=((((double)MSF\_Image[r\*COLS+c]-min)/(double)(max-min))\*255);

}

}

int TP=0,FP=0;

int TN=0,FN=0;

while ((read = getline(&line, &len, ftr)) != -1) {

int pixelc=0;

int pixelr=0;

char \*token = strtok(line, " ");

token = strtok(NULL, " ");

pixelc=atoi(token);

token = strtok(NULL, " ");

pixelr=atoi(token);

int Flag=0;

for(r=-i;r<=i;r++){

for(c=-j;c<=j;c++){

if(image[(pixelr+r)\*COLS+(pixelc+c)]>=Threshold){

if(line[0]=='e'){

TP++;

Flag=1;

}

else{

FP++;

Flag=1;

}

break;

}

}

if(Flag==1){break;} ///Leave if Flag is set and a value above threshold is found

}

///Enter if no no value is above Threshold

if(Flag==0){

if(line[0]=='e'){

FN++;

}

else{

TN++;

}

}

Flag=0;

}///End While

printf("TP-%d FP-%d TN-%d FN-%d Total-%d Threshold-%d\n",TP,FP,TN,FN,TP+FP+TN+FN,Threshold);

/\* write out brighter.ppm, the output result \*/

fpt=fopen("brighter.ppm","w");

fprintf(fpt,"P5 %d %d 255\n",COLS,ROWS);

fwrite(image,1,ROWS\*COLS,fpt);

fclose(fpt);

}

Shell used to run the program from 150 to 255 threshold.

**gcc -o lab2 lab2.c**

**for i in {150..255..5}**

**do**

**a=./lab2**

**a+=' '$i**

**$a**

**done**