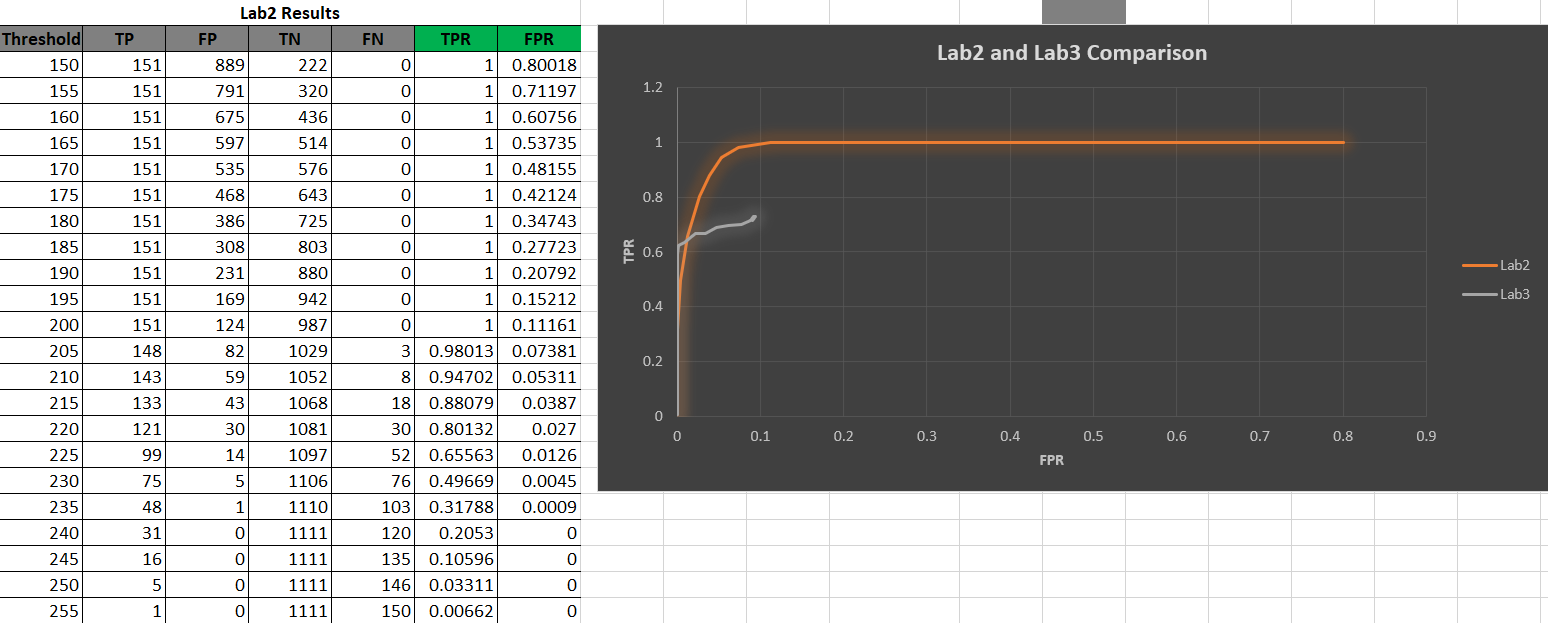
**ECE 4310/6310 Introduction to Computer Vision**

Lab #3 – letters

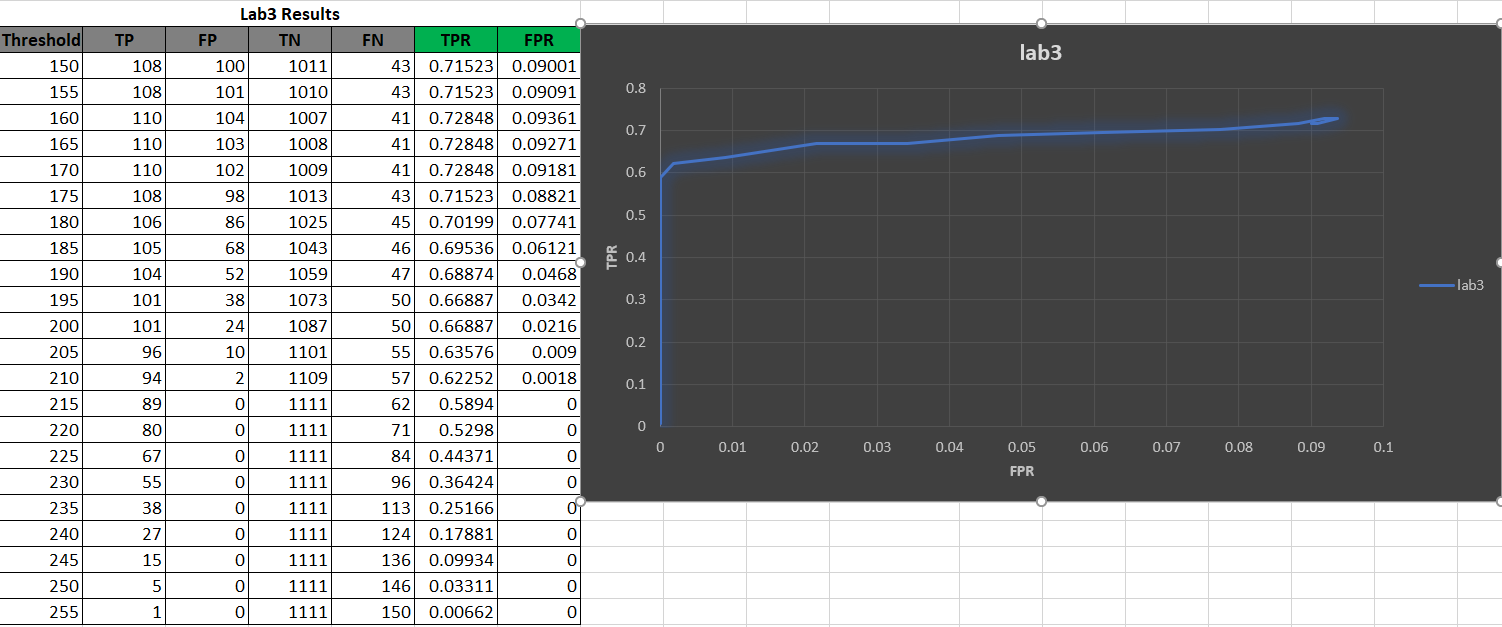
Name-Pinak Kelkar.

In the code below

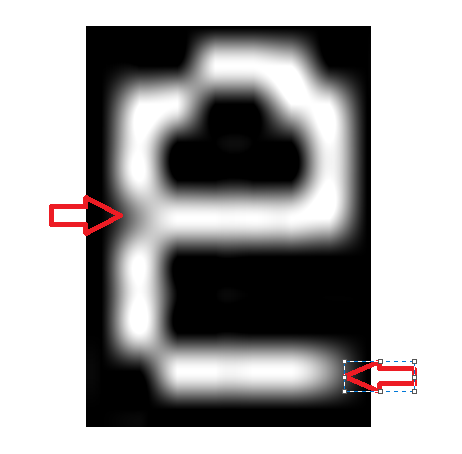
So, the below is the Comparison of the Lab2 and lab3 ROC curve in which we clearly see that the False positive rate has dropped drastically.



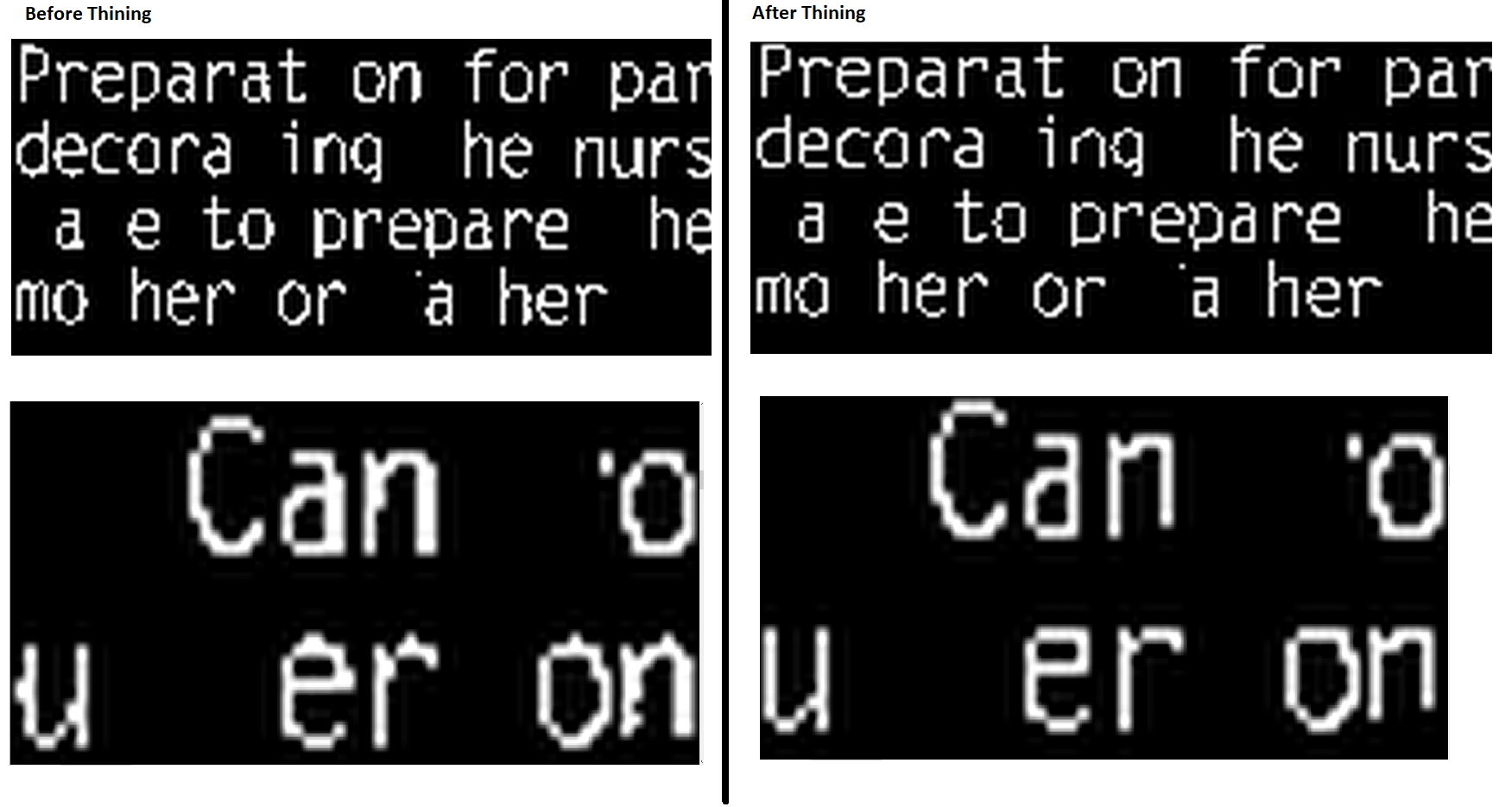
The optimal threshold is 200 to 210

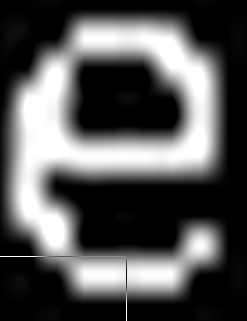


Example of Branch point and End point calculation.

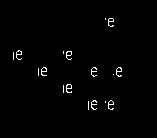
 

The Thinning is done till there no pixel was not deleted in the last run. This helps to thin all the edges.

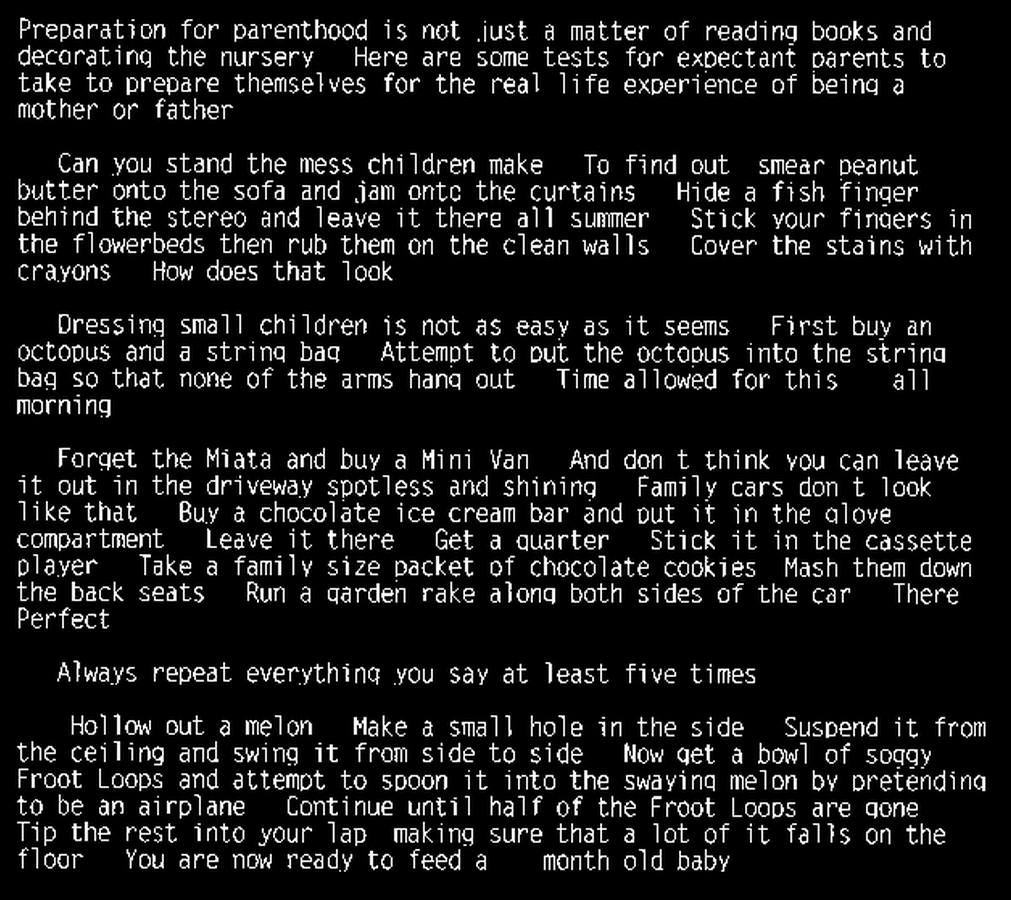
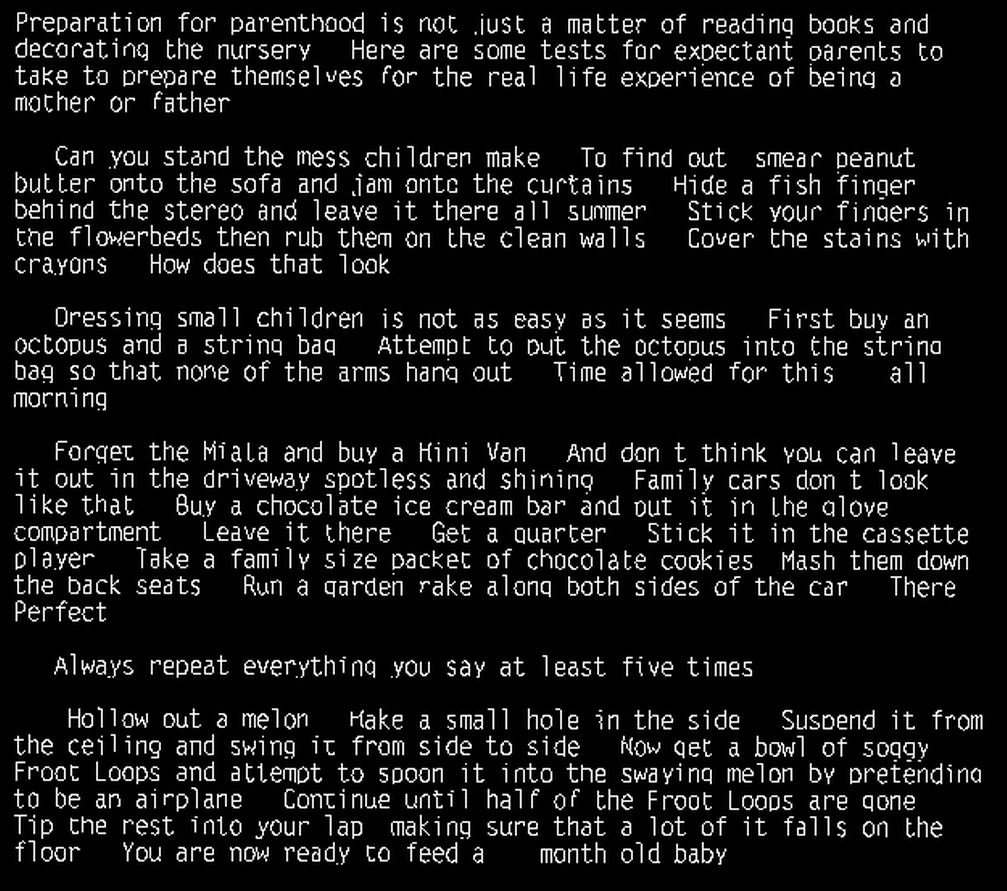


Detecting the Branch point and the end point the False positive rate did drop. Taking an example for threshold 150 in lab2 there were 889 false positive and in this one it drops to 100 which is drastic drop. But this method also dropped the rate of True positive as some of the e’s which were getting thinned had some pixel from nearby value as seen below in the exam. This made it difficult to detect all the e’s, but we can do this by segmentation. Using segmentation, we can find the bigger segment and then count branch point and the end point on the bigger segment. The problem is that in the window size of 9\*15 the pixel from the other side is getting picked up which changes the Branch point and end point calculation. Examples are below.



**128 threshold Image Binary image Thinned 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 \*fpt,\*fpt1,\*fpt2;

unsigned char \*MSF;

unsigned char \*MSF\_Binary;

unsigned char \*Image;

unsigned char \*Thin\_Image;

unsigned char \*dummy,\*dummy2,\*testing;

int i=0,r=0,c=0;

char T\_header[80];

int T\_ROWS,T\_COLS,T\_BYTES,Threshold=atoi(argv[1]);

char \* line = NULL;

size\_t len = 0;

ssize\_t read;

int bp=0,ep=0;

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

fpt1=fopen("parenthood.ppm","r");

fpt2=fopen("parenthood\_gt.txt","r");

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

i=fscanf(fpt1,"%s %d %d %d",T\_header,&T\_COLS,&T\_ROWS,&T\_BYTES);

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

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

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

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

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

dummy2=(unsigned char \*)calloc(T\_ROWS\*T\_COLS,sizeof(unsigned char));

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

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

fread(MSF,1,T\_ROWS\*T\_COLS,fpt);

fclose(fpt);

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

fread(Image,1,T\_ROWS\*T\_COLS,fpt1);

fclose(fpt1);

int TP=0,FP=0;

int TN=0,FN=0;

while ((read = getline(&line, &len, fpt2)) != -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;

**///\*\*\*\*\*\*\*\*\*\*\*\*Threshold\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/////////////////**

for(r=-7;r<=7;r++){

for(c=-4;c<=4;c++){

if(MSF[(pixelr+r)\*T\_COLS+(pixelc+c)]>=Threshold){

Flag=1;

break;

}

}

if(Flag==1){break;}

}

**///\*\*\*\*\*\*\*\*\*\*\*\*Creating Binary Image\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/////////////////**

if(Flag==1){

for(r=-7;r<=7;r++){

for(c=-4;c<=4;c++){

if(Image[(pixelr+r)\*T\_COLS+(pixelc+c)]<128){ MSF\_Binary[(pixelr+r)\*T\_COLS+(pixelc+c)]=255; }

else{MSF\_Binary[(pixelr+r)\*T\_COLS+(pixelc+c)]=0;}

}

}

for (i=0; i<T\_ROWS\*T\_COLS; i++) {

dummy[i]=MSF\_Binary[i];

}

**///\*\*\*\*\*\*\*\*\*\*\*\*Thining\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/////////////////**

int Thining=1;

while(Thining==1){

Thining=0;

for(r=-7;r<=7;r++){

for(c=-4;c<=4;c++){

int C=pixelc+c;

int R=pixelr+r;

int CA=0;

int CB=0;

int A1=0,B1=0,C1=0,D1=0;

if(dummy[R\*T\_COLS+C]==255){

if(dummy[(R-1)\*T\_COLS+(C-1)]==255){

if(dummy[(R-1)\*T\_COLS+(C)]==0){

CA+=1;

}

CB+=1;

}

if(dummy[(R-1)\*T\_COLS+(C)]==255){

if(dummy[(R-1)\*T\_COLS+(C+1)]==0){

CA+=1;

}

CB+=1;A1=1;

}

if(dummy[(R-1)\*T\_COLS+(C+1)]==255){

if(dummy[(R)\*T\_COLS+(C+1)]==0){

CA+=1;

}

CB+=1;

}

if(dummy[(R)\*T\_COLS+(C+1)]==255){

if(dummy[(R+1)\*T\_COLS+(C+1)]==0){

CA+=1;

}

CB+=1;B1=1;

}

if(dummy[(R+1)\*T\_COLS+(C+1)]==255){

if(dummy[(R+1)\*T\_COLS+(C)]==0){

CA+=1;

}

CB+=1;

}

if(dummy[(R+1)\*T\_COLS+(C)]==255){

if(dummy[(R+1)\*T\_COLS+(C-1)]==0){

CA+=1;

}

CB+=1;D1=1;

}

if(dummy[(R+1)\*T\_COLS+(C-1)]==255){

if(dummy[(R)\*T\_COLS+(C-1)]==0){

CA+=1;

}

CB+=1;

}

if(dummy[(R)\*T\_COLS+(C-1)]==255){

if(dummy[(R-1)\*T\_COLS+(C-1)]==0){

CA+=1;

}

CB+=1;C1=1;

}

if(CA==1 && (CB>=3 && CB<=7) && (A1==0 || B1==0 || (C1==0 && D1==0))){

dummy2[R\*T\_COLS+C]=0;Thining=1;

}else{

dummy2[R\*T\_COLS+C]=255;

}

}

else{

dummy2[R\*T\_COLS+C]=0;

}

}

}///FOR LOOP

for (i=0; i<T\_ROWS\*T\_COLS; i++) {

dummy[i]=dummy2[i];

}

}////while loop

for (i=0; i<T\_ROWS\*T\_COLS; i++) {

Thin\_Image[i]=dummy[i];

dummy[i]=0;

}

bp=0;

ep=0;

**///\*\*\*\*\*\*\*\*\*\*\*\*Counting Branch Point and EndPoint\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/////////////////**

for(r=-7;r<=7;r++){

for(c=-4;c<=4;c++){

int C=pixelc+c;

int R=pixelr+r;

int count=0;

if(Thin\_Image[R\*T\_COLS+C]==255){

if(Thin\_Image[(R-1)\*T\_COLS+(C-1)]==255){

if(Thin\_Image[(R-1)\*T\_COLS+(C)]==0){

count++;}

}

if(Thin\_Image[(R-1)\*T\_COLS+(C)]==255){

if(Thin\_Image[(R-1)\*T\_COLS+(C+1)]==0){

count++;}

}

if(Thin\_Image[(R-1)\*T\_COLS+(C+1)]==255){

if(Thin\_Image[(R)\*T\_COLS+(C+1)]==0){

count++;}

}

if(Thin\_Image[(R)\*T\_COLS+(C+1)]==255){

if(Thin\_Image[(R+1)\*T\_COLS+(C+1)]==0){

count++;}

}

if(Thin\_Image[(R+1)\*T\_COLS+(C+1)]==255){

if(Thin\_Image[(R+1)\*T\_COLS+(C)]==0){

count++;}

}

if(Thin\_Image[(R+1)\*T\_COLS+(C)]==255){

if(Thin\_Image[(R+1)\*T\_COLS+(C-1)]==0){

count++;}

}

if(Thin\_Image[(R+1)\*T\_COLS+(C-1)]==255){

if(Thin\_Image[(R)\*T\_COLS+(C-1)]==0){

count++;}

}

if(Thin\_Image[(R)\*T\_COLS+(C-1)]==255){

if(Thin\_Image[(R-1)\*T\_COLS+(C-1)]==0){

count++;}

}

if(count==1){

ep++;

}

if(count>=3){

bp++;

}

}////IF

}////FOR

}///FOR

if(bp==1 && ep==1){

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

TP++;

}

else{

FP++;

}

}

else{

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

FN++;

}

else{

TN++;

}

}

}//Flag==1

else{

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

FN++;

}

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

TN++;

}

}

}//While Loop

printf("Tp=%d FP=%d FN=%d TN=%d Total=%d TotalE=%d\n",TP,FP,FN,TN,TP+FP+FN+TN,TP+FN);

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

fprintf(fpt,"P5 %d %d 255\n",T\_COLS,T\_ROWS);

fwrite(MSF\_Binary,1,T\_ROWS\*T\_COLS,fpt);

fclose(fpt);

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

fprintf(fpt,"P5 %d %d 255\n",T\_COLS,T\_ROWS);

fwrite(Thin\_Image,1,T\_ROWS\*T\_COLS,fpt);

fclose(fpt);

}