ECE 6310 Introduction to Computer Vision
Lab #3 – Letters

Christin Wilson
CUID: C21515896

October 3, 2019

1 Assignment

In this project each student must implement thinning, branchpoint and endpoint detection to recognize letters in an image of text.

Your program should perform the following steps:
1. Read the input image, your msf image, and ground truth file.
2. Loop through the following steps for a range of T:

a. Loop through the ground truth letter locations.

i. Check a 9 x 15 pixel area centered at the ground truth location. If any pixel in the msf image is greater than the threshold, consider the letter "detected". If none of the pixels in the 9 x 15 area are greater than the threshold, consider the letter "not detected".

ii. If the letter is "not detected" continue to the next letter.

iii. Create a 9 x 15 pixel image that is a copy of the area centered at the ground truth location (center of letter) from the original image.

iv. Threshold this image at 128 to create a binary image.

v. Thin the thresholded image down to single-pixel wide components.

vi. Check all remaining pixels to determine if they are branchpoints or endpoints.

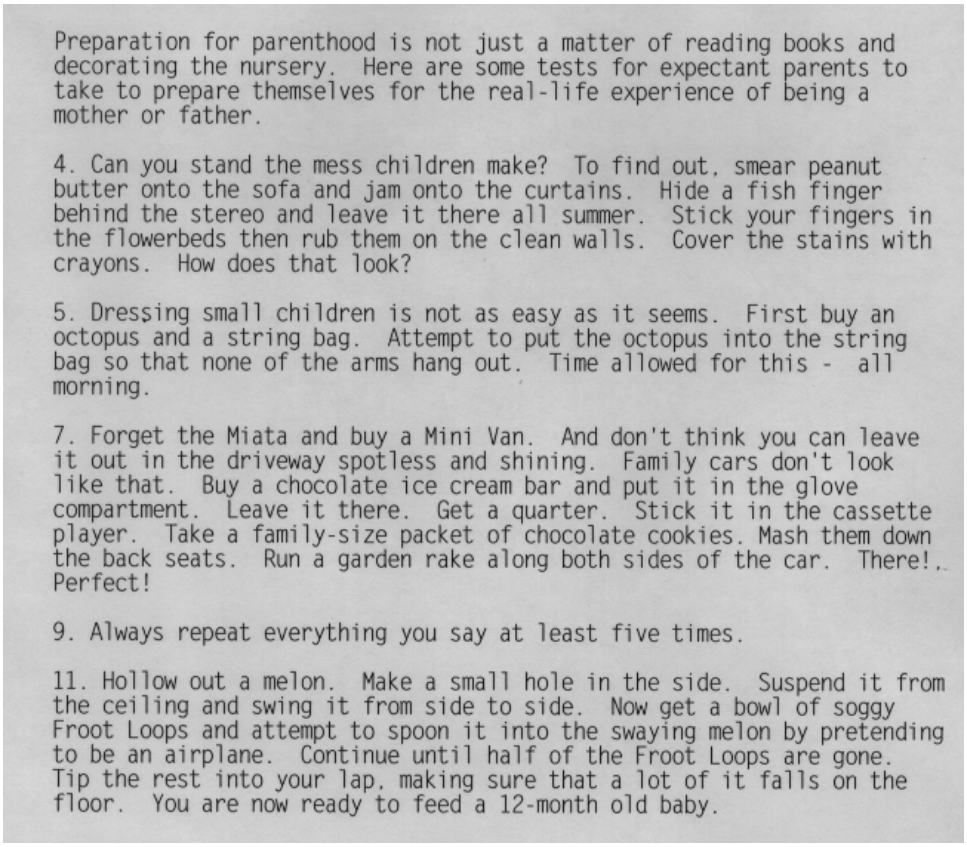
vii. If there are not exactly 1 branchpoint and 1 endpoint, do not further consider this letter (it becomes "not detected").

b. Count up the number of FP (letters detected that are not 'e') and TP (number of letters detected that are 'e').

c. Output the total TP and FP for each T. Using any desired program, you must create an ROC curve from the program output.

You must write a brief report that includes the code and the ROC curve. Show an example of your image copy after thresholding (step iv above), the thinned image (step v), and the detection of branchpoints and endpoints (step vi). In order to be clearly visible you can use color or overlay symbols to highlight the detected branchpoints and endpoints. Identify the optimal T and its corresponding FP and TP values.

2 Images



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: Original image

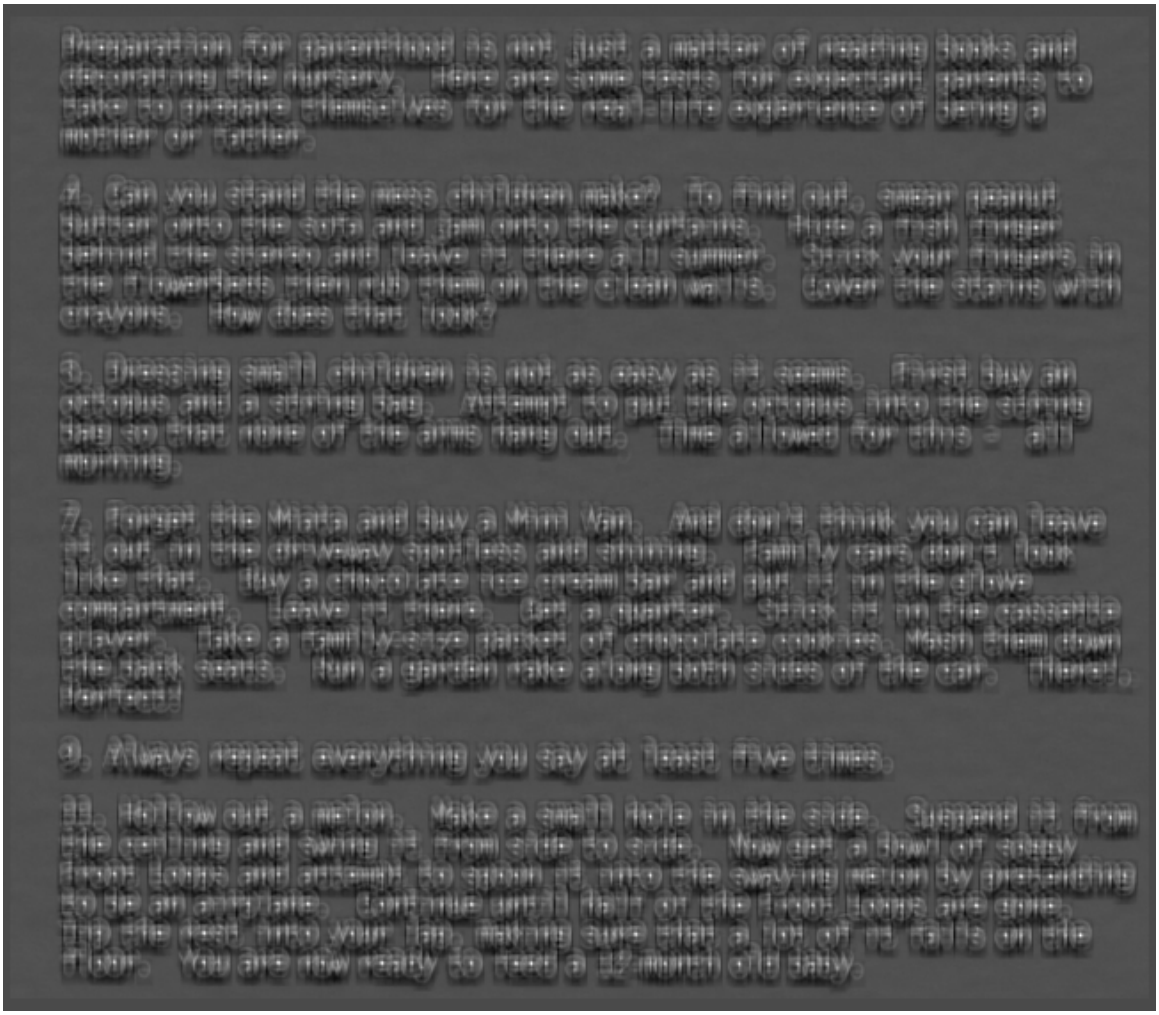


Figure 2: MSF image after normalization

Figure 3: image copied

Figure 4: image converted to binary

Figure 5: binary image after thinning

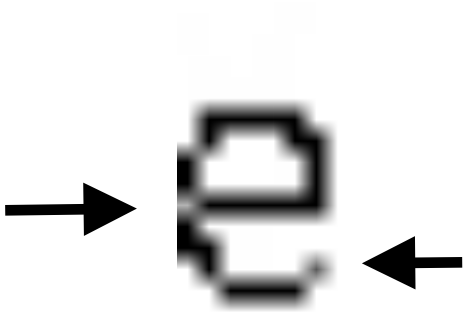


Figure 6: image with branchpoint and endpoint represented with intensity 180 and 90 respectively

3 Output Values

Threshold	TP	FP	TN	FN	TPR	FPR
0	141	152	959	10	0.933775	0.136814
5	141	152	959	10	0.933775	0.136814
10	141	152	959	10	0.933775	0.136814
15	141	152	959	10	0.933775	0.136814
20	141	152	959	10	0.933775	0.136814
25	141	152	959	10	0.933775	0.136814
30	141	152	959	10	0.933775	0.136814
35	141	152	959	10	0.933775	0.136814
40	141	152	959	10	0.933775	0.136814
45	141	152	959	10	0.933775	0.136814
50	141	152	959	10	0.933775	0.136814
55	141	152	959	10	0.933775	0.136814
60	141	152	959	10	0.933775	0.136814
65	141	152	959	10	0.933775	0.136814
70	141	152	959	10	0.933775	0.136814
75	141	152	959	10	0.933775	0.136814
80	141	152	959	10	0.933775	0.136814
85	141	152	959	10	0.933775	0.136814
90	141	152	959	10	0.933775	0.136814
95	141	152	959	10	0.933775	0.136814
100	141	152	959	10	0.933775	0.136814
105	141	152	959	10	0.933775	0.136814
110	141	152	959	10	0.933775	0.136814
115	141	152	959	10	0.933775	0.136814
120	141	152	959	10	0.933775	0.136814
125	141	152	959	10	0.933775	0.136814
130	141	152	959	10	0.933775	0.136814
135	141	152	959	10	0.933775	0.136814
140	141	152	959	10	0.933775	0.136814
145	141	152	959	10	0.933775	0.136814
150	141	152	959	10	0.933775	0.136814

Threshold	TP	FP	TN	FN	TPR	FPR
155	141	152	959	10	0.933775	0.136814
160	141	152	959	10	0.933775	0.136814
165	141	150	961	10	0.933775	0.135014
170	141	147	964	10	0.933775	0.132313
175	141	134	977	10	0.933775	0.120612
180	141	112	999	10	0.933775	0.10081
185	141	82	1029	10	0.933775	0.073807
190	141	50	1061	10	0.933775	0.045005
195	141	30	1081	10	0.933775	0.027003
200	140	16	1095	11	0.927152	0.014401
205	138	7	1104	13	0.913907	0.006301
210	134	3	1108	17	0.887417	0.0027
215	127	1	1110	24	0.84106	0.0009
220	113	0	1111	38	0.748344	0
225	93	0	1111	58	0.615894	0
230	66	0	1111	85	0.437086	0
235	42	0	1111	109	0.278146	0
240	28	0	1111	123	0.18543	0
245	13	0	1111	138	0.086093	0
250	1	0	1111	150	0.006623	0
255	0	0	1111	151	0	0

4 ROC Curve

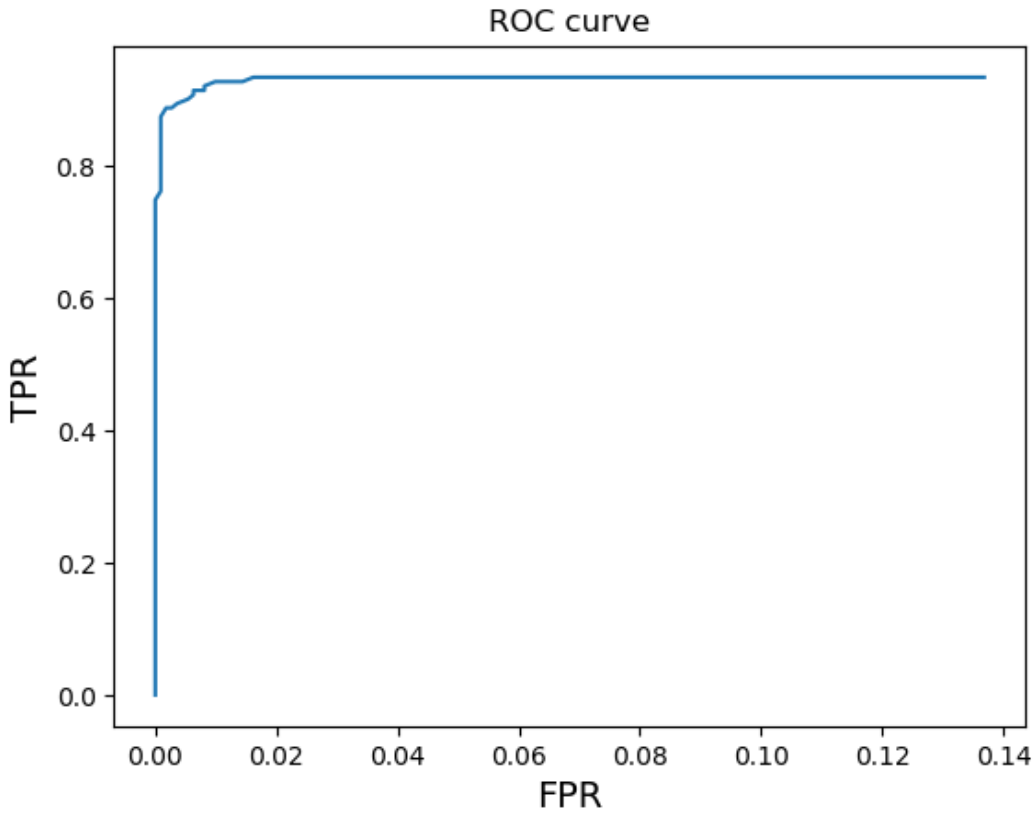


Figure 7: ROC Curve generated

Based on the ROC Curve, the optimal threshold can be calculated as about Threshold=200. The values of TP and FP is 140 and 16 respectively. I presume that the the lower max value of TP in this lab is due to me not considering the pixels in the ends of the images specially. Instead of making the neighbouring pixel value as 0 while considering for thinning, my code is just considering the value before or after in the array representation of the image.

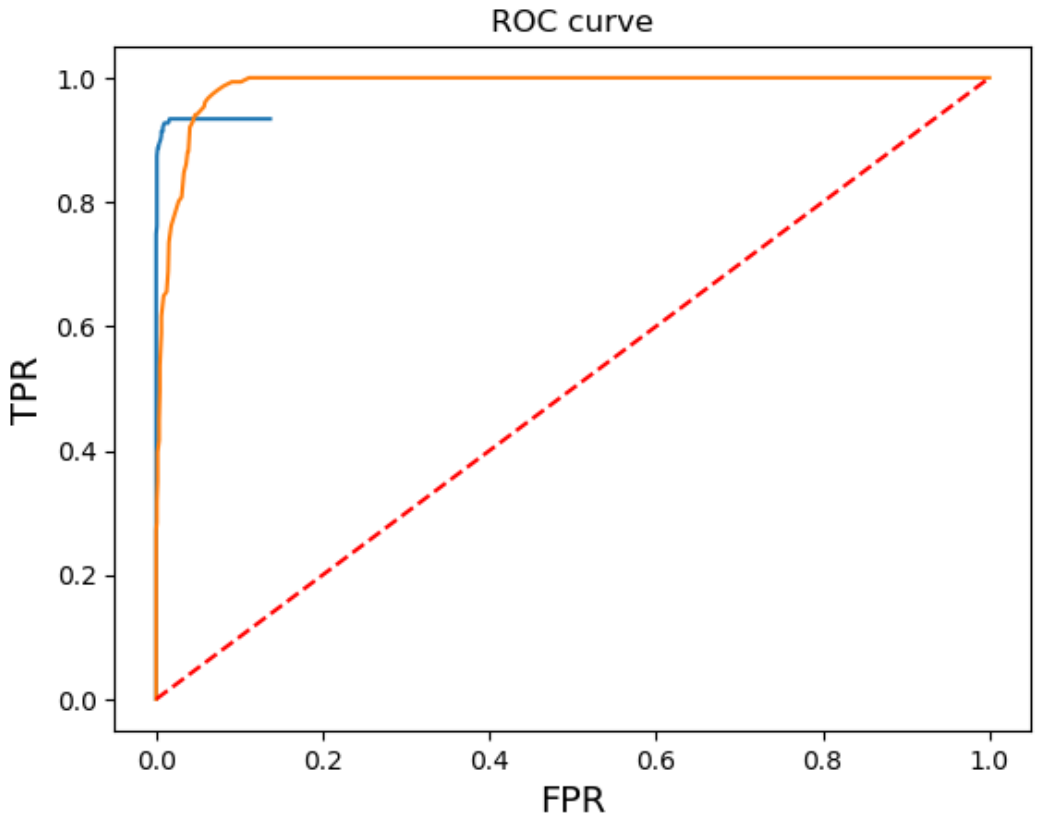


Figure 8: ROC Curve in comparison with previous lab

5 Source code

```
 /*
        ** This program implements Lab 3: Letters
        **.
        */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
int main()

 {
 FILE            *fpt1,*fpt2;
 unsigned char          *image,*MSF,*temp,*check;
 int threshold=0;
 unsigned char          *final;
 char            header[320];
 char    header2[320];
 int             ROWS,COLS,BYTES;
 int   ROWS2,COLS2,BYTES2;
 int             r,c,r2,c2,sum,count;
 int   r3,c3;
 char alphabet;
 int row,col;
 FILE *fpt3;

        /* read image */
 if ((fpt1=fopen("parenthood.ppm","rb")) == NULL)
 {
  printf("Unable to open parenthood.ppm for reading\n");
  exit(0);
 }
 fscanf(fpt1,"%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);
 }
 if ((fpt2=fopen("temp.ppm","rb")) == NULL)
 {
  printf("Unable to open parenthood.ppm for reading\n");
  exit(0);
 }
 fscanf(fpt2,"%s %d %d %d",header2,&COLS2,&ROWS2,&BYTES2);
 if (strcmp(header2,"P5") != 0  ||  BYTES2 != 255)
 {
  printf("Not a greyscale 8-bit PPM image\n");
  exit(0);
 }

 image=(unsigned char *)calloc(ROWS*COLS,sizeof(unsigned char));
 header[0]=fgetc(fpt1);         /* read white-space character that separates header */
 fread(image,1,COLS*ROWS,fpt1);
 fclose(fpt1);
```

```
        /* allocate memory for final version of image */
MSF=(unsigned char *)calloc(ROWS*COLS,sizeof(unsigned char));
header2[0]=fgetc(fpt2);  /* read white-space character that separates header */
fread(MSF,1,COLS*ROWS,fpt2);
fclose(fpt2);
final=(unsigned char *)calloc(ROWS*COLS,sizeof(unsigned char));
temp=(unsigned char *)calloc(15*9,sizeof(unsigned char));
check=(unsigned char *)calloc(15*9,sizeof(unsigned char));

r3=15/2;
c3=9/2;
ROWS2=15;
COLS2=9;

for(threshold=236;threshold<=236;threshold+=5)
{
  count=0;
  for (r=0; r<ROWS; r++)
  {
    for (c=0; c<COLS; c++)
    {
      if(MSF[r*COLS+c]>threshold)
        final[r*COLS+c]= 255;
      else
        final[r*COLS+c]= 0;
    }
  }
  int gt=0,ob=0;
  int tp=0,fp=0,fn=0,tn=0;
  int m=0;
  int k=0,x,y,ov=0;
  int transitions, neighbours, edgecheck, MARKED=1,branch,end;
  fpt3 = fopen("parenthood_gt.txt" , "r");
  r=fscanf(fpt3,"%c %d %d\n",&alphabet,&col,&row);
  while(m != EOF)
  {
  int MARKED=1;
    if(alphabet=='e')
    {

      gt=1;
    }
    else
    {
    gt=0;
    }

    ob=0;ov=0;

    for (r=row-r3; r<=row+r3; r++)
    {
      for (c=col-c3; c<=col+c3; c++)
      {
        if(final[r*COLS+c]==255)
```

```
    ob=1;
  }
}
if (ob==1)
{
  for (r=row-r3,x=0; r<=row+r3; r++,x++)
  {
    for (c=col-c3,y=0; c<=col+c3; c++,y++)
    {
      temp[x*COLS2+y]=image[r*COLS+c];
    }
  }
  fpt1=fopen("temp1.ppm","w");
  fprintf(fpt1,"P5 %d %d 255\n",COLS2,ROWS2);
  fwrite(temp,COLS2*ROWS2,1,fpt1);
  fclose(fpt1);
  for(x=0;x<ROWS2;x++)
  {
    for(y=0;y<COLS2;y++)
    {
      if(temp[x*COLS2+y]<128)
      {
        temp[x*COLS2+y]=0;
      }
      else
      {
        temp[x*COLS2+y]=255;
      }
    }
  }
  fpt1=fopen("temporbin.ppm","w");
  fprintf(fpt1,"P5 %d %d 255\n",COLS2,ROWS2);
  fwrite(temp,COLS2*ROWS2,1,fpt1);
  fclose(fpt1);
  while(MARKED!=0)
  {
    //printf("hi");
    MARKED=0;
    for(x=0;x<ROWS2;x++)
    {
      for(y=0;y<COLS2;y++)
      {
        check[x*COLS2+y]=0;
        if(temp[x*COLS2+y]==0)
        {
          transitions=0;

          if((temp[(x-1)*COLS2+(y-1)])==0 && (temp[(x-1)*COLS2+y])==255)
          {  transitions++;     }

          if((temp[(x-1)*COLS2+y])==0 && (temp[(x-1)*COLS2+(y+1)])==255)
          {  transitions++;     }

          if((temp[(x-1)*COLS2+(y+1)])==0 && (temp[(x)*COLS2+(y+1)])==255)
```

```
{  transitions++;    }

if((temp[(x)*COLS2+(y+1)])==0 && (temp[(x+1)*COLS2+(y+1)])==255)
{  transitions++;    }

if((temp[(x+1)*COLS2+(y+1)])==0 && (temp[(x+1)*COLS2+(y)])==255)
{  transitions++;    }

if((temp[(x+1)*COLS2+(y)])==0 && (temp[(x+1)*COLS2+(y-1)])==255)
{  transitions++;    }

if((temp[(x+1)*COLS2+(y-1)])==0 && (temp[(x)*COLS2+(y-1)])==255)
{  transitions++;    }

if((temp[(x)*COLS2+(y-1)])==0 && (temp[(x-1)*COLS2+(y-1)])==255)
{  transitions++;    }

neighbours=0;

if((temp[(x-1)*COLS2+(y-1)])==0)
{  neighbours++;    }

if((temp[(x-1)*COLS2+y])==0)
{  neighbours++;    }

if((temp[(x-1)*COLS2+(y+1)])==0)
{  neighbours++;    }

if((temp[(x)*COLS2+(y+1)])==0)
{  neighbours++;    }

if((temp[(x+1)*COLS2+(y+1)])==0)
{  neighbours++;    }

if((temp[(x+1)*COLS2+(y)])==0)
{  neighbours++;    }

if((temp[(x+1)*COLS2+(y-1)])==0)
{  neighbours++;    }

if((temp[(x)*COLS2+(y-1)])==0)
{  neighbours++;    }

edgecheck=0;

if((temp[(x-1)*COLS2+(y)])==255)
{  edgecheck=1;    }

else if((temp[(x)*COLS2+(y+1)])==255)
{  edgecheck=1;    }

else if((temp[(x+1)*COLS2+(y)])==255 && (temp[(x)*COLS2+(y-1)])==255)
{  edgecheck=1;    }
```

```
        if(transitions==1 && neighbours<=6 && neighbours>=2 && edgecheck==1)
        {
          check[x*COLS2+y]=1;
          MARKED=1;
        }
      }
    }
  }
  for(x=0;x<ROWS2;x++)
  {
    for(y=0;y<COLS2;y++)
    {
      if(check[x*COLS2+y]==1)
      {
      temp[x*COLS2+y]=255;
      }
    }
  }

}


fpt1=fopen("temporthinned.ppm","w");
fprintf(fpt1,"P5 %d %d 255\n",COLS2,ROWS2);
fwrite(temp,COLS2*ROWS2,1,fpt1);
fclose(fpt1);
branch=0;
end=0;
for(x=0;x<ROWS2;x++)
{
  for(y=0;y<COLS2;y++)
  {

    if(temp[x*COLS2+y]==0)
    {
      check[x*COLS2+y]=0;
      transitions=0;

        if((temp[(x-1)*COLS2+(y-1)])==0 && (temp[(x-1)*COLS2+y])==255)
        {  transitions++;    }

        if((temp[(x-1)*COLS2+y])==0 && (temp[(x-1)*COLS2+(y+1)])==255)
        {  transitions++;    }

        if((temp[(x-1)*COLS2+(y+1)])==0 && (temp[(x)*COLS2+(y+1)])==255)
        {  transitions++;    }

        if((temp[(x)*COLS2+(y+1)])==0 && (temp[(x+1)*COLS2+(y+1)])==255)
        {  transitions++;    }

        if((temp[(x+1)*COLS2+(y+1)])==0 && (temp[(x+1)*COLS2+(y)])==255)
        {  transitions++;    }

        if((temp[(x+1)*COLS2+(y)])==0 && (temp[(x+1)*COLS2+(y-1)])==255)
```

```
        {  transitions++;     }

        if((temp[(x+1)*COLS2+(y-1)])==0 && (temp[(x)*COLS2+(y-1)])==255)
        {  transitions++;     }

        if((temp[(x)*COLS2+(y-1)])==0 && (temp[(x-1)*COLS2+(y-1)])==255)
        {  transitions++;     }

       if (transitions>2)
       {
         check[x*COLS2+y]=1;
         branch++;
       }
       if(transitions==1)
       {
         check[x*COLS2+y]=2;
         end++;
       }
     }
   }
  }
  for(x=0;x<ROWS2;x++)
    {
      for(y=0;y<COLS2;y++)
      {
        if(check[x*COLS2+y]==1)
        {
        temp[x*COLS2+y]=180;
        }
        else if(check[x*COLS2+y]==2)
        {
        temp[x*COLS2+y]=90;
        }
      }
    }
  fpt1=fopen("temporbranched.ppm","w");
  fprintf(fpt1,"P5 %d %d 255\n",COLS2,ROWS2);
  fwrite(temp,COLS2*ROWS2,1,fpt1);
  fclose(fpt1);
  if(branch==1&&end==1)
  {
    ov=1;
  }
  else
  {
    ov=0;
  }
}


if(gt==1&&ov==1)
{tp++;}
else if(gt==0&&ov==1)
{ fp++;}
```

```
      else if(gt==0&&ov==0)
      { tn++;}
      else if(gt==1&&ov==0)
      {fn++;}
      m=fscanf(fpt3,"%c %d %d\n",&alphabet,&col,&row);
    }
    fclose(fpt3);
    float tpr,fpr;
    tpr=tp*1.0/((tp+fn)*1.0);
    fpr=fp*1.0/(fp+tn)*1.0;
    printf("Threshold: %d tp: %d fp: %d tn: %d fn: %d tpr: %f fpr:
%f\n",threshold,tp,fp,tn,fn,tpr,fpr);

  }

          /* write out final image to see result */
fpt1=fopen("final.ppm","w");
fprintf(fpt1,"P5 %d %d 255\n",COLS,ROWS);
fwrite(final,COLS*ROWS,1,fpt1);
fclose(fpt1);
}
```