Shashi Shivaraju
XID:C88650674

REPORT
ECE 6310 Lab #3 – Letters(UPDATED 6th October,2017)

Objective: To implement thinning, branch-point and endpoint detection algorithms to recognize letters in an image of text.

Implementation: Usage of thinning, branch-point and endpoint detection algorithms to recognize a template in the input image follows the below steps for a particular pixel threshold **(T):**

a)Threshold the input image at provided pixel value 128

[Code snippet]:

```
/*Threshold the original image Image at 128 */
    for(i=0;i<row*col;i++)
    {
            /*letter is in black colour range and background is in white colour range*/
            if(128 < image[i])
            {
                  binary_image[i] = 0;
            }
            else
            {
                  binary_image[i] = 255;
            }
    }
```

Here,
unsigned char* image is a pointer to memory to store input image.
unsigned char* binary_image is a pointer to memory to store thresholded binary image.
int row,int col = height and width of the input image of text in pixels.

Shashi Shivaraju
XID:C88650674

b)Thin the thresholded binary image down to single-pixel wide components

A thinning algorithm which is currently implemented follows the below steps in loop until no further thinning is possible:

- Consider each pixel (X) of the the image at a time.
- Count the number of edge->non-edge transitions in CW order around the pixel X.
- Count the number of edge neighbor pixels.
- Check that at least one of the North, East, or (West and South) are not edge pixels.
- The edge pixel is marked for erasure if it has exactly 1 edge -> non-edge transition,
  3 <= edge neighbors <= 7 and one of the North, East, or (West and South) are not edge pixels
- Once all pixels have been checked for the above conditions, erase the marked pixels and repeat the above steps until no pixels are marked for erasure.

```
[Code snippets]:

/*thin the thresholded Image to single pixel wide component image */
thin_2_single_pixel(binary_image,row,col);

Here,
int row          = height of the input image of text in pixels.
int col          = width of the input image of text in pixels.
unsigned char* binary_image is a pointer to memory storing detected binary image.
thin_2_single_pixel is a function to thin the image to single pixel width image.

[Function]:
void thin_2_single_pixel(unsigned char *binary_image,int row,int col)
{
      while(1)
      {
            erasal_count = 0;
            for(i = 0;i< row*col;i++)
            {

                  if(0 == binary_image[i])
                        continue;

                  extract_8_neighbour_pixels(binary_image,row,col,threeXthree_pixels,i);

                  detect_8_neighbour_pixels(threeXthree_pixels,
                  &edge2non_egde_transistion,&neighbour_edge_count,
                  &north_nonedge,&east_nonedge ,&south_nonedge,&west_nonedge);

                  /*Check for erasure as per thinning algorithm*/
                  if(
                  edge2non_egde_transistion == 1 &&
                  3 <= neighbour_edge_count && neighbour_edge_count <= 7 &&
                  (north_nonedge || east_nonedge || (south_nonedge && west_nonedge))
                  && 255 == threeXthree_pixels[0]
```

```
                )
                {
                      erasal_index[erasal_count] = i;
                      erasal_count ++;
                }

                /*reset the variables for next pixel*/
                edge2non_egde_transistion = 0;neighbour_edge_count = 0;
                north_nonedge = 0;east_nonedge = 0;south_nonedge = 0;west_nonedge = 0;
           }

           for(i=0;i<erasal_count;i++)
           {
                      index = erasal_index[i];
                      binary_image[index] = 0;
                      erasal_index[i] = 0;
           }

           if(!erasal_count)/*exit loop since further thinning is not possible*/
                break;
      }
}
```

```
Here,
int row          = height of the binary image in pixels.
int col          = width of the binary image in pixels.
unsigned char* binary_image is a pointer to memory storing detected binary image.
unsigned char* threeXthree_pixels is a pointer to memory storing 9 pixels.
extract_8_neighbour_pixels()is a function to extract the neighborhood pixels of a pixel.
detect_8_neighbour_pixels() is a function  to detect counts of edge to non-edge
transition,edge pixels and non-edge pixels(non-edge pixels in NSEW directions only).
```

```
[Function]:
void extract_8_neighbour_pixels(unsigned char *binary_image,
      int row,int col,unsigned char *threeXthree_pixels,int index)
{
      int i = index;

      /*check 8 neighbour pixels to determine erasal*/

      if(0 < i-1 && (i%col))
           threeXthree_pixels[0] = binary_image[i-1]; /*west*/
      else
           threeXthree_pixels[0] = 1; /*represents center is border pixel*/

      if(0 < i-col-1 && (i%col))
           threeXthree_pixels[1] = binary_image[i-col-1]; /*north_west*/
      else
           threeXthree_pixels[1] = 1; /*represents center is border pixel*/
```

Shashi Shivaraju
XID:C88650674

```
      if(0 < i-col)
            threeXthree_pixels[2] = binary_image[i-col]; /*north*/
      else
            threeXthree_pixels[2] = 1; /*represents center is border pixel*/

      if(0 < i-col+1 && ((i+1)%col))
            threeXthree_pixels[3] = binary_image[i-col+1]; /*north_east*/
      else
            threeXthree_pixels[3] = 1; /*represents center is border pixel*/

      if(i+1 < row*col && ((i+1)%col))
            threeXthree_pixels[4] = binary_image[i+1]; /*east*/
      else
            threeXthree_pixels[4] = 1; /*represents center is border pixel*/

      if(i+col+1 < row*col && ((i+1)%col))
            threeXthree_pixels[5] = binary_image[i+col+1]; /*south_east*/
      else
            threeXthree_pixels[5] = 1; /*represents center is border pixel*/

      if(i+col < row*col)
            threeXthree_pixels[6] = binary_image[i+col]; /*south*/
      else
            threeXthree_pixels[6] = 1; /*represents center is border pixel*/

      if(i+col-1 < row*col && (i%col))
            threeXthree_pixels[7] = binary_image[i+col-1]; /*south_west*/
      else
            threeXthree_pixels[7] = 1; /*represents center is border pixel*/
}
```

```
Here,
unsigned char* binary_image is a pointer to memory storing detected binary image.
unsigned char* threeXthree_pixels is a pointer to memory storing 9 pixels.
```

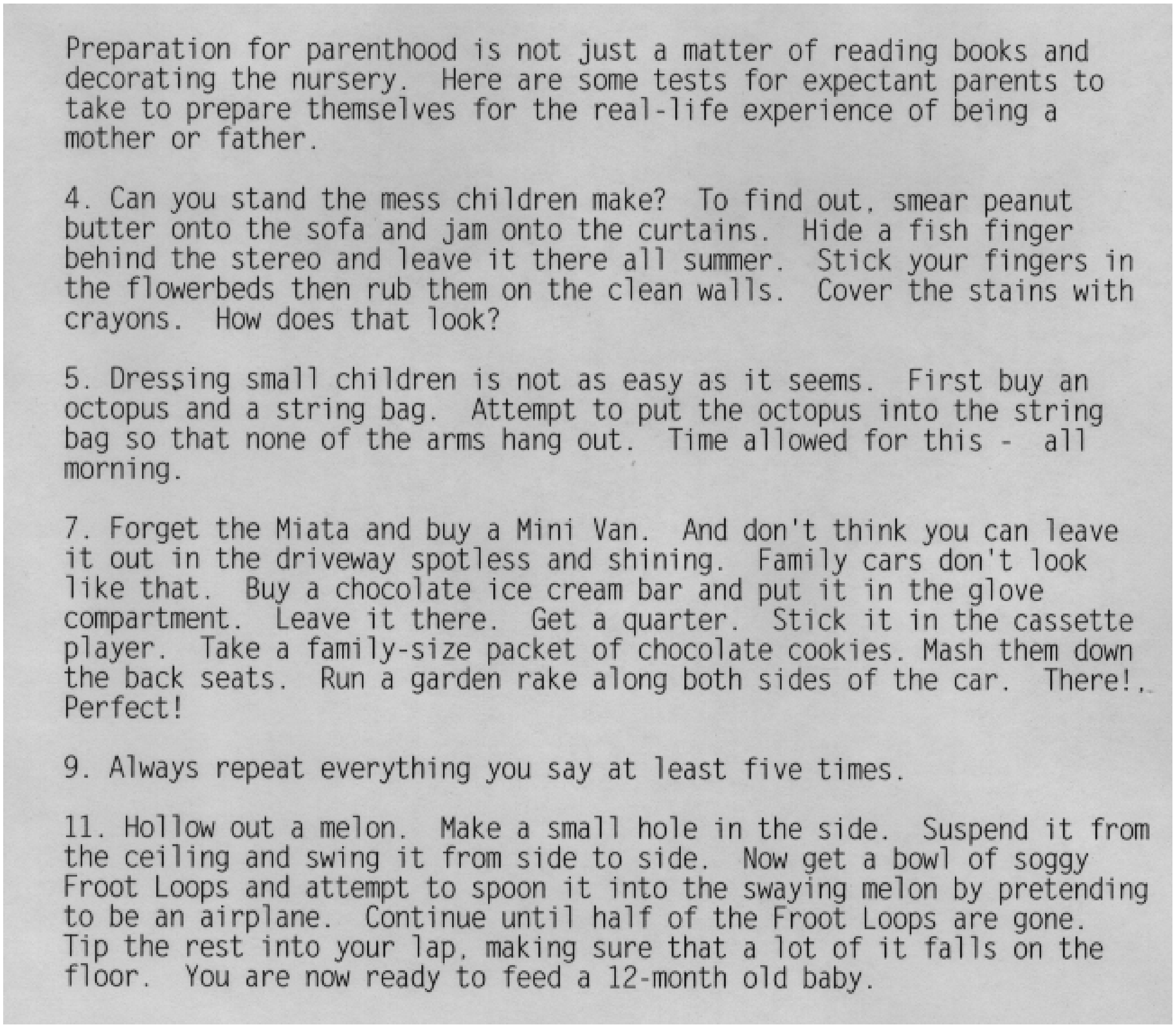
```
[Function]:
int detect_8_neighbour_pixels( unsigned char *threeXthree_pixels,
      int *a_edge2non_egde_transistion,int *a_neighbour_edge_count,
      int *a_north_nonedge,int *a_east_nonedge ,int *a_south_nonedge,
      int *a_west_nonedge)
{
      prev = threeXthree_pixels[0];
      for(j=0;j<8;j++)
      {
            cur = threeXthree_pixels[j];
            if(255 == cur)
            {
                  neighbour_edge_count++;
```

Shashi Shivaraju
XID:C88650674

```
                ///*check for edge to non edge transition from south_west to west */
                if(7 == j && 0 == threeXthree_pixels[0])
                     edge2non_egde_transistion ++;

                prev = cur;
           }
           else if(0 == cur)
           {
                /*check for edge to non edge transition*/
                if(255 == prev)
                     edge2non_egde_transistion++;

                if(2 == j)
                     north_nonedge = 1;
                else if(4 == j)
                     east_nonedge = 1;
                else if(6 == j)
                     south_nonedge = 1;
                else if(0 ==j)
                     west_nonedge = 1;

                prev = cur;
           }
           else if(1 == cur)/*neighbour pixel is not inside the image*/
           {
                /*check for edge to non edge transition at
                border clockwise (from south_west to west) */
                if(7 == j && 255 == prev)
                     edge2non_egde_transistion ++;

                if(2 == j)
                     north_nonedge = 1;
                else if(4 == j)
                     east_nonedge = 1;
                else if(6 == j)
                     south_nonedge = 1;
                else if(0 ==j)
                     west_nonedge = 1;
           }
      }
}
```

Shashi Shivaraju
XID:C88650674

[Input image containing text]:
649 x 567 8bit Gray scale image of PPM format named parenthood.ppm



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.

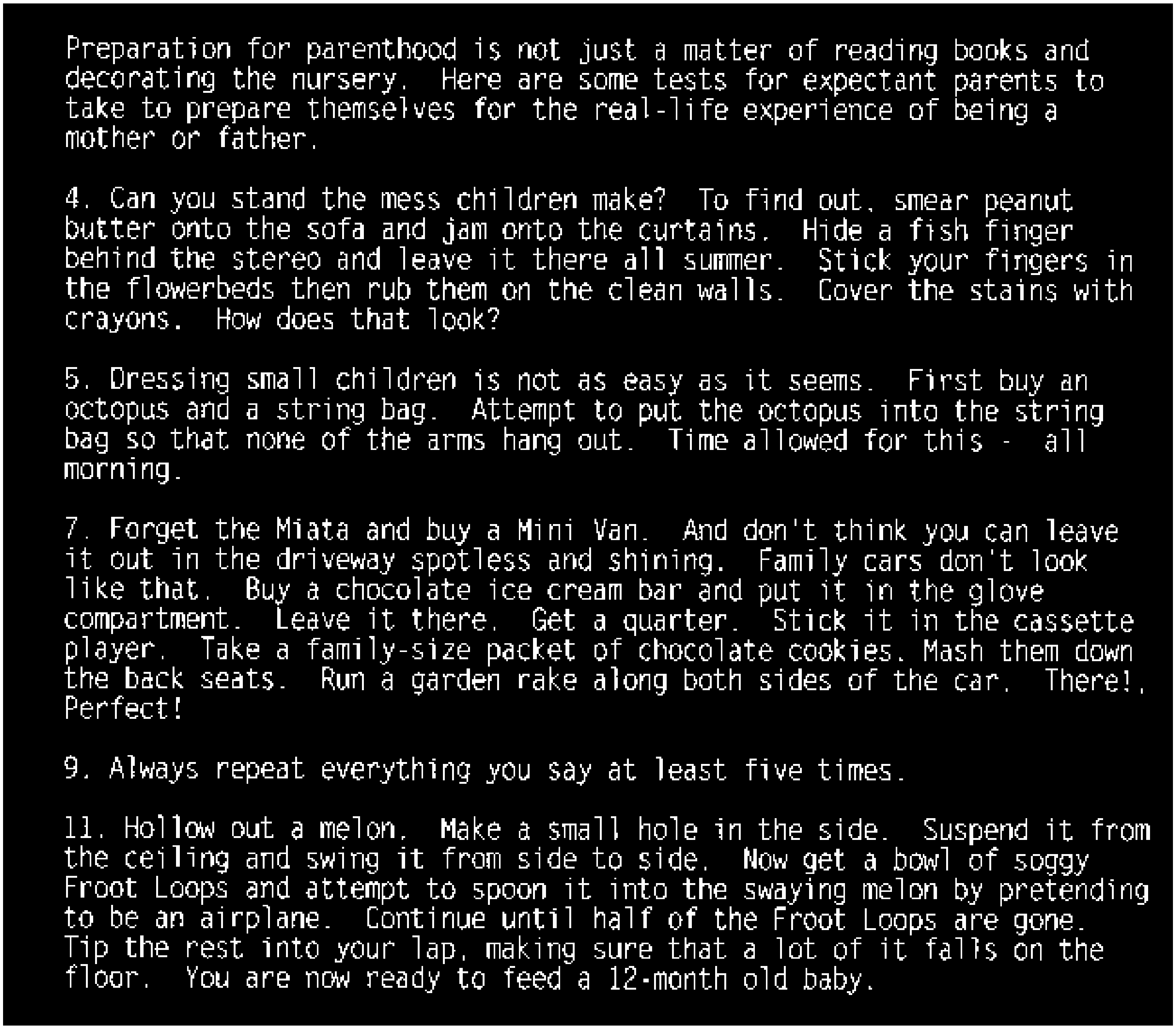
7. Forget the Miata and buy a Mini Van. And don't think you can leave it out in the driveway spotless and shining. Family cars don't look like that. Buy a chocolate ice cream bar and put it in the glove compartment. Leave it there. Get a quarter. Stick it in the cassette player. Take a family-size packet 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.

Shashi Shivaraju
XID:C88650674

[Thresholded image at provided pixel value 128]:
649 x 567 8bit Gray scale image of PPM format named Threshold_output.ppm



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.

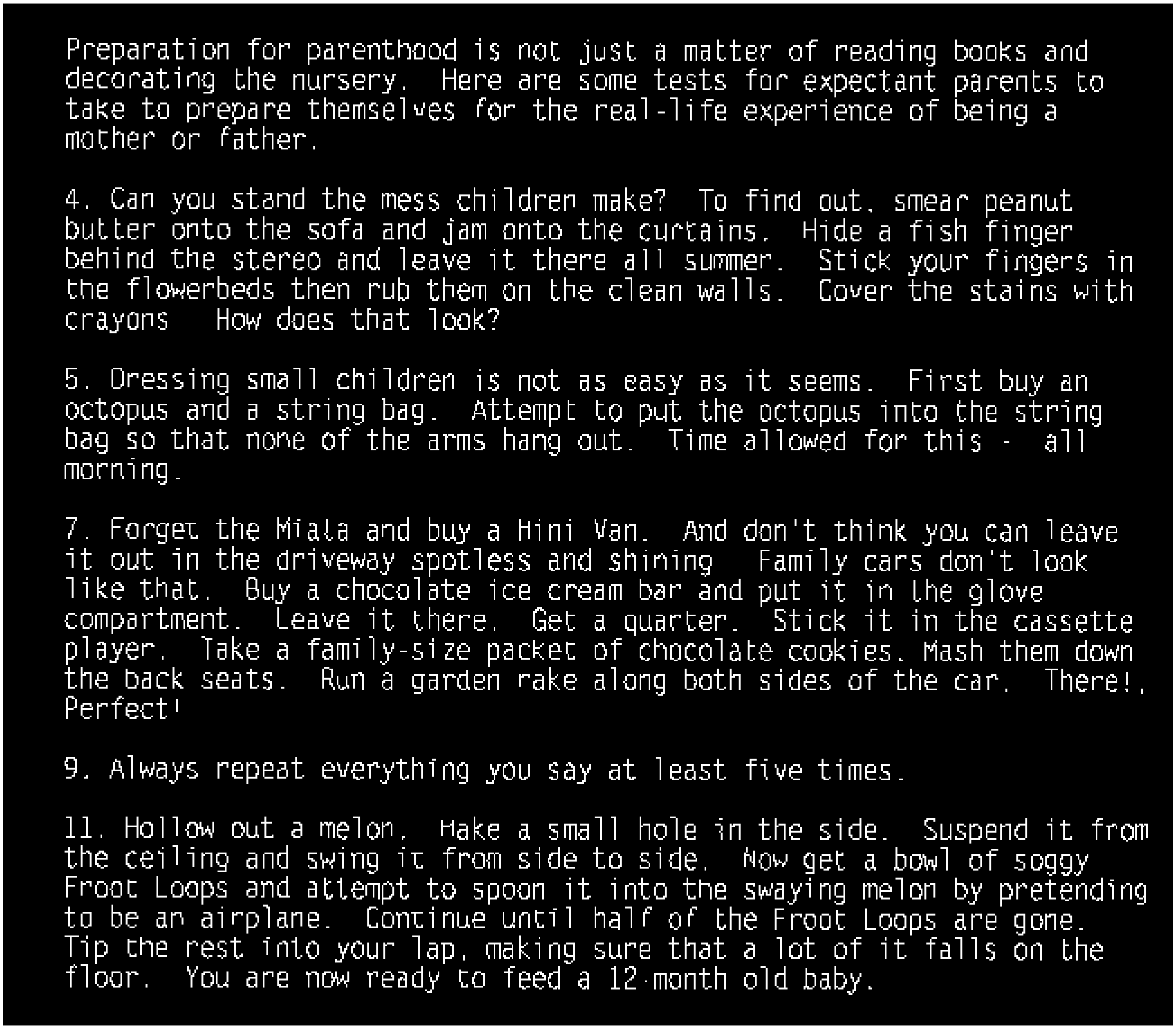
7. Forget the Miata and buy a Mini Van. And don't think you can leave it out in the driveway spotless and shining. Family cars don't look like that. Buy a chocolate ice cream bar and put it in the glove compartment. Leave it there. Get a quarter. Stick it in the cassette player. Take a family-size packet 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.

Shashi Shivaraju
XID:C88650674

[Single pixel thin thresholded Image]:
649 x 567 8bit Gray scale image of PPM format named Skeleton.ppm**]**



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.

Shashi Shivaraju
XID:C88650674

c)At each at the ground truth location,check a 9 x 15 pixel area centered at ground truth location in the input normalized MSF image.

Any pixel that is greater than the threshold T represents initial detections of template. If none of the pixels in this 9 x 15 area are greater than the threshold, the letter is considered not detected.

[Code snippet]:

```
while(0<fscanf(fp,"%c %d %d ",&gt_char,&gt_col,&gt_row))
{
      detection_flag = 0;
      /*Check a 9 x 15 pixel area centered at the ground truth
        location of the MSF Image file to detect the character*/
      for(R=gt_row-(row_template/2);R<=gt_row+(row_template/2);R++)
      {
            for(C=gt_col-(col_template/2);C<=gt_col+(col_template/2);C++)
            {
                  if(threshold < MSF_image[R*col+C])
                   {
                        detection_flag = 1;
                        break;
                   }
            }
            if(detection_flag)
                  break;
      }
      ...
      ...
}
```

Here,
unsigned char* MSF_Image is a pointer to memory storing Normalized MSF image.
int row_template,int col_template = height and width of the template image in pixels.
int row,int col = height and width of the input image of text in pixels.
unsigned char gt_char represents the ground truth value.
int gt_row,int gt_col represents the ground truth pixel location.

Shashi Shivaraju
XID:C88650674

d)If initial detection is true from step (c),then 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 single pixel thin thresholded image.

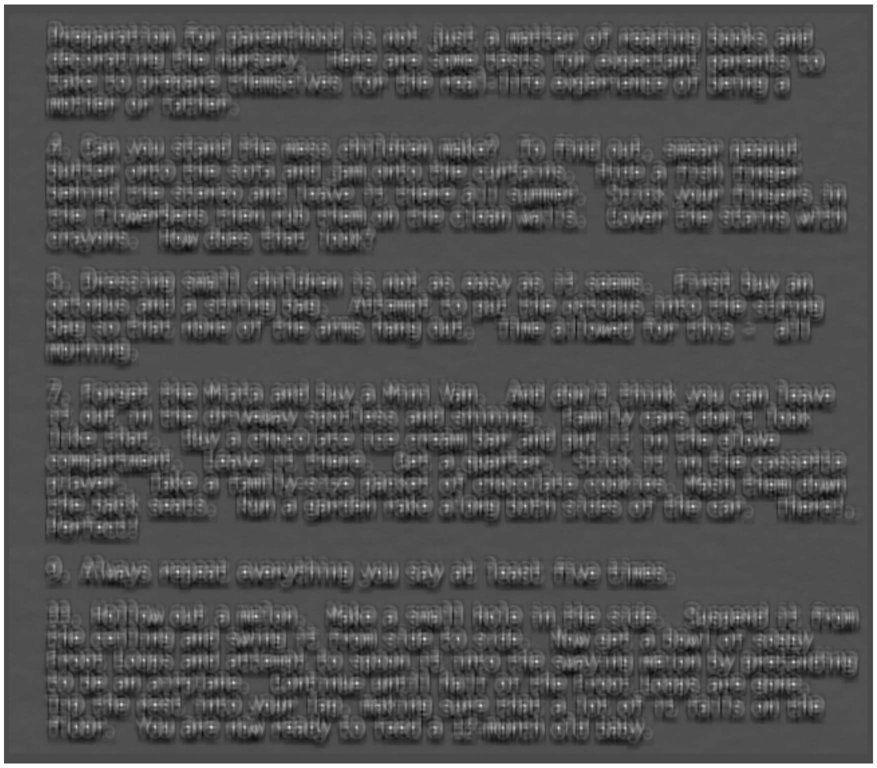
[Code snippet]:

```
           ...
           ...


           if(detection_flag)
           {
                /*
                 * Create a 9 x 15 pixel image centered at
                 * the ground truth location from the original
                 *  image of detected letter
                 */
                 i=0;
                 for(R=gt_row-(row_template/2);R<=gt_row+(row_template/2);R++)
                 {
                      for(C=gt_col-(col_template/2);C<=gt_col+(col_template/2);C++)
                      {
                          detected_image[i] = binary_image[R*col+C];
                          i++;
                     }
                 }
           ...
           ...
           }

           ...
           ...
```

Here,
int row_template = height of the template image in pixels=15.
int col_template = width of the template image in pixels=9.
unsigned char* binary_image is a pointer to memory storing thinned binary image.
unsigned char* detected_image is a pointer to memory storing detected image.
int gt_row,int gt_col represents the ground truth pixel location.

Shashi Shivaraju
XID:C88650674

[Input normalized MSF image]:
649 x 567 8bit Gray scale image of PPM format named MSF_Normalized_output.ppm



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.

1. 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?

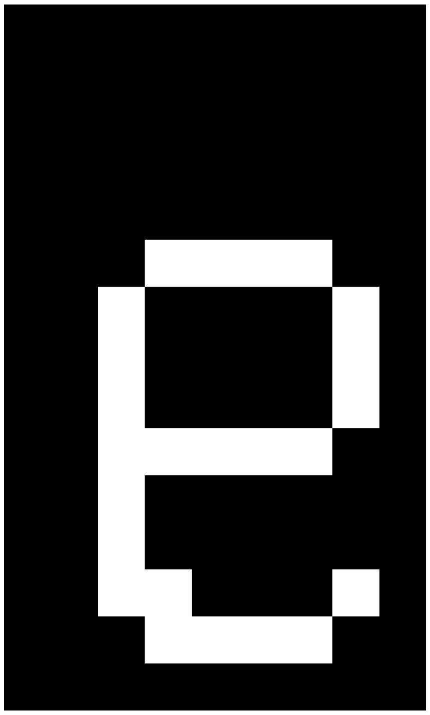
3. Dressing small children is not as easy as it seems. First buy an
octopus and a string bag. Attempt to put the octopus into the string
bag so that none of the arms hang out. Time allowed for this - all
morning.

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.

[Detected Image]:
9 x 15 8bit Gray scale image of PPM format named detected_character.ppm



Shashi Shivaraju
XID:C88650674

e)Endpoints and Branch-points detection.

Check all remaining edge pixels of the thinned detected image to determine if they are endpoint or branch-point.

A edge pixel is marked as endpoint,if its neighboring pixels have exactly one edge -> non-edge transition in clockwise direction.
A edge pixel is marked as branch-point if its neighboring pixels have more than two edge -> non-edge transitions in clockwise direction.

Based the number of endpoint pixels and branch-point pixels we can determine the character/letter in the thinned binary image.

[Code snippets]:

```
/*detect the number of endpoints and branchpoints*/
detect_endpoints_branchpoints(detected_binary_image,row_template,
                                  col_template,&endpoint_count,
                                  &branchpoint_count);
```

[Function]:

```
void detect_endpoints_branchpoints(unsigned char *binary_image,
      int row,int col,int *a_branchpoint_count,int *a_endpoint_count)
{
      for(i = 0;i< row*col;i++)
      {
            if(binary_image[i])/*Check for only edge pixels*/
            {
                  /*function call to extract the neighbourhood pixels*/
                  extract_8_neighbour_pixels(binary_image,row,col,threeXthree_pixels,i);

                  /*function call to detect count of edge to non edge transition,
                   * edge pixel and nonedge pixels(NSEW directions only )*/
                  detect_8_neighbour_pixels(threeXthree_pixels,
                  &edge2non_egde_transistion,
                                          NULL,NULL,NULL,NULL,NULL);

                  if(1 == edge2non_egde_transistion)
                  {
                        endpoint_count++;
                  }
                  else if(2 < edge2non_egde_transistion)
                  {
                        branchpoint_count++;
                  }
            }
      }
```

Shashi Shivaraju
XID:C88650674

f)Letter Detection and calculating the True Positive Rate (TPR) and False Positive Rate (FPR).

For the letter ‘e’,the thinned binary image should contain exactly one endpoint and one branch-point.

If the thinned binary image **contains** exactly one endpoint
and one branch-point,we consider that the template(letter ‘e’)is detected.
Further if the ground truth value is also ‘e’ then it is considered True Positive(TP) else it is considered False Positive(FP).

If the thinned binary image **does not have** exactly one endpoint
and one branch-point,we consider that the template(letter ‘e’)is not detected.
Further if the ground truth value is equal to template (letter ‘e’) then it is considered False Negative(FN) else it is considered True Negative(TN).

Additionally for each location obtained from the ground truth file (parenthood_gt.txt),we check an area of the template dimension centered at the ground truth location in the input normalized MSF image.
If none of the pixels in this area is above threshold T,then we consider the template(letter ‘e’) is not detected.
Further if the ground truth value is equal to template (letter ‘e’) then it is considered False Negative(FN) else it is considered True Negative(TN).

The True Positive Rate(TPR) and False Positive Rate(FPR) of the system is calculated as below:

- **TPR = TP/(TP+FN)**
- **FPR = FP/(FP+TN)**

[Code snippet]:

```
        if(detection_flag)
        {
         ...
         ...
         /*TPR and FPR calculated based on branchpoint and endpoint detection for e*/
             if(1 ==  endpoint_count &&
                1 == branchpoint_count)
             {
                  if('e' == gt_char)
                       TP_Count++; /*detected and the letter is 'e'*/
                  else
                       FP_Count++; /*detected and the letter is not'e'*/
             }
```

```
                else
                {
                      if('e' == gt_char)
                            FN_Count++; /*not detected and the letter is 'e'*/
                      else
                            TN_Count++; /*not detected and the letter is not 'e'*/
                }

          }
          else
          {
                if('e' == gt_char)
                      FN_Count++; /*not detected and the letter is 'e'*/
                else
                      TN_Count++; /*not detected and the letter is not 'e'*/
          }
          detection_flag = 0;

TP_Rate  = ((float)TP_Count/(float)(TP_Count+FN_Count));
FP_Rate  = ((float)FP_Count/(float)(FP_Count+TN_Count));
```

```
Here,
int endpoint_count = number of endpoint pixels in the thinned binary image.
int branchpoint_count = number of branch-point pixels in the thinned binary image.
unsigned char gt_char represents the ground truth value.
int FP_Count = instances of detected but the letter is not template.
int TP_Count = instances of detected and the letter is template.
int FN_Count = instances of not detected but the letter is template.
int TN_Count = instances of not detected and the letter is not template.
float TP_Rate = True Positive Rate(TPR)
float FP_Rate = False Positive Rate(FPR)
```

Shashi Shivaraju
XID:C88650674

System Analysis for Range of Threshold:

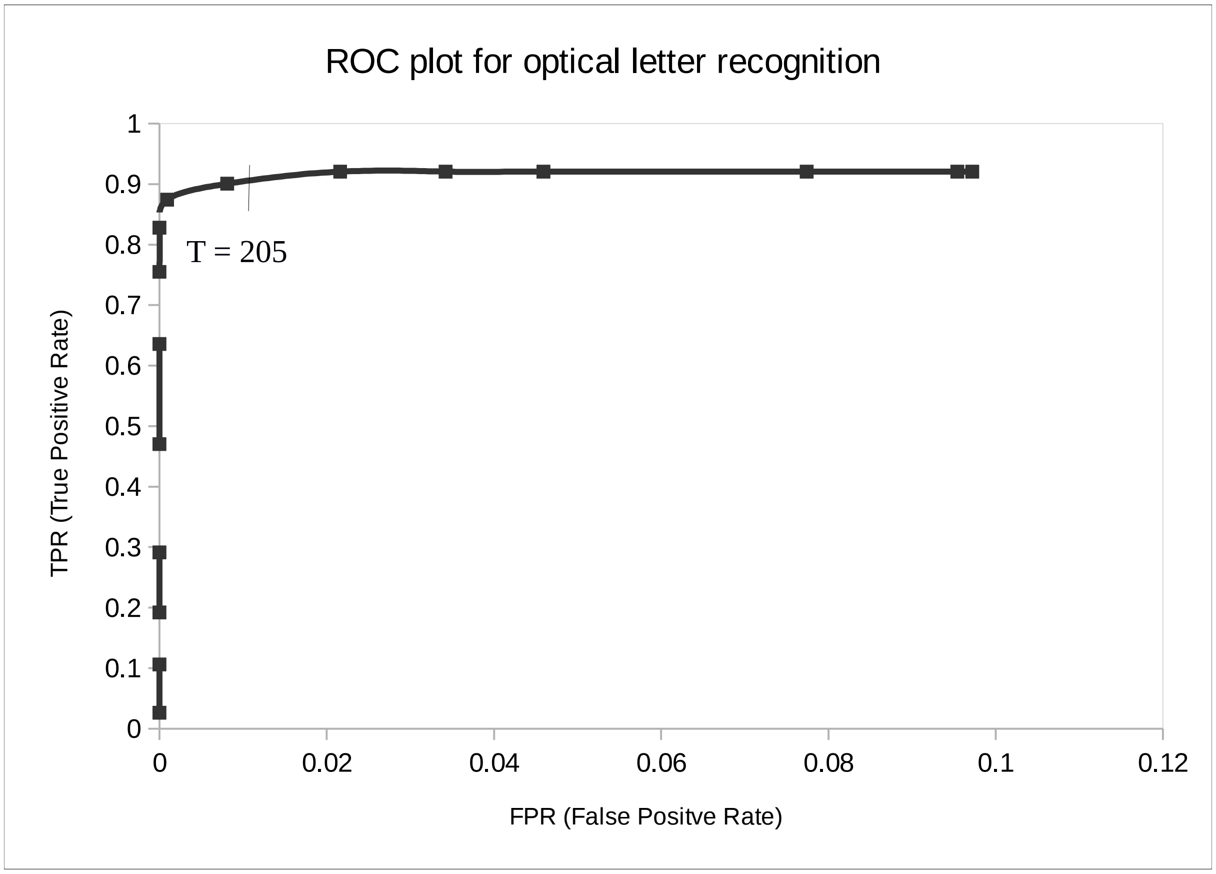
T (Threshold Pixel Value for Detection)	TPR (True Positive Rate)	FPR (False Positive Rate)	TP (True Positive Count)	FN (False Negative Count)	FP (False Positive Count)	TN (True Negative Count)
254	0.006623	0.000000	1	150	0	1111
250	0.026490	0.000000	4	147	0	1111
245	0.105960	0.000000	16	135	0	1111
240	0.192053	0.000000	29	122	0	1111
235	0.291391	0.000000	44	107	0	1111
230	0.470199	0.000000	71	80	0	1111
225	0.635762	0.000000	96	55	0	1111
220	0.754967	0.000000	114	37	0	1111
215	0.827815	0.000000	125	26	0	1111
210	0.874172	0.000900	132	19	1	1110
205	**0.900662**	**0.008101**	**136**	**15**	**9**	**1102**
200	0.920530	0.021602	139	12	24	1087
195	0.920530	0.034203	139	12	38	1073
190	0.920530	0.045905	139	12	51	1060
180	0.920530	0.077408	139	12	86	1025
170	0.920530	0.095410	139	12	106	1005
160	0.920530	0.097210	139	12	108	1003
140	0.920530	0.097210	139	12	108	1003
120	0.920530	0.097210	139	12	108	1003
100	0.920530	0.097210	139	12	108	1003
75	0.920530	0.097210	139	12	108	1003
50	0.920530	0.097210	139	12	108	1003
25	0.920530	0.097210	139	12	108	1003
5	0.920530	0.097210	139	12	108	1003

Shashi Shivaraju
XID:C88650674

Receiver Operating Characteristic (ROC):

ROC is a plot of True Positive Rate(TPR) vs False Positive Rate(FPR) as a function system variable and in this scenario the system variable is the pixel value Threshold(T) used to determine the initial template detections.

Below is the plot of ROC for the implemented Optical letter Recognition of letter ‘e’ in image “parenthood.ppm”



Selection of Optimal T(Threshold Pixel Value):

As per the ground truth file “parenthood_gt.txt” provided there are 151 instances of the given template in the input image which an ideal system will detect.

Thus an optimal T would be one in which we get high TP count(detected as template and the letter is template) and low FP count (detected as template but the letter is not template).From the ROC curve plot above, it can be observed the this condition is best represented at the arc of the curve by the value of T = 205.
Hence the Optimal pixel threshold value that should be selected for this system is T = 205 for which TPR = 90.0662% and FPR = 0.8101%.