ASSIGNMENT 4

General Info:

Student Name: Poornima Manjunath

Student Id: 01752385

Assignment number: 4

Assignment due date: 5/10/2018

ASSIGNMENT 4

<u>**Objective:**</u>

The objective of the assignment is to learn the different image processing techniques used in recognizing the objects from their binary images.

The first part of the assignment deals with morphological operation i.e. closing (dilation followed by erosion).

The second part deals with the design of a binary vision system that recognizes the objects based on the features like, Euler number.

Background:

Morphological operators are the useful tools for extracting the image components that help in the representation and description of region shape.

Dilation adds a layer of pixels around a periphery of region. It fills small holes within regions.

Erosion removes pixels from the periphery of a region.

In the first part of the assignment, we are performing a closing operation. Closing is a combination of a dilation followed by an erosion fusing narrow brakes, eliminating small holes, filling gaps. We use morphological operations to clean the binary (thresholded) image marker. Basically, we apply dilation successively a certain number of times, X, on the marker binary image until the marker hole is closed. Then we apply the same number X of successive erosions to the binary image marker.

In the second part, we design a vision system that recognizes the objects based on their binary images. Simple geometric features like compactness, centroid and topological descriptors like Euler number are used.

Algorithm:

1. Read the image file and put into a 2D array. In the case of marker, the dimension of the image is 160X200. Width = 160 so the (number of columns = 160) and height = 200(number of rows = 200).

2. Perform thresholding with a fine-tuned threshold value to create a correct binary image.
Note here, that the object has dark intensities as well. And the difference in the intensities between that of the object and the background is less. In fact, the background has lighter pixels and the marker has the dark pixels.
So, I used the logic, if the intensity of the image pixel is less than threshold than make it TRUE, else if the intensity of the image pixel is greater than the threshold than make it FALSE.

3. Perform dilation to fill the hole on the marker. Perform padding of one layer of 0's around the image.

4. Then to perform dilation, use 3X3 sub-array of the padded image with pixel of the original image as the center of the 3X3 sub-image. If any of the neighboring elements of the center pixel is 'BRIGHT' or 'TRUE' then the center pixel value becomes 'TRUE'. If all the neighboring elements of the center pixel are dark, then only the center pixel becomes dark.

5. Perform dilation successively for 3 times, until the hole is closed.

6. Then perform erosions 3 times on the output image.
To perform erosion, use 3X3 sub-array of the padded image with pixel of the original image as the center of the 3X3 sub-image. If any of the neighboring elements of the center pixel is 'DARK' or 'FALSE' then the center pixel value becomes 'DARK' or “FALSE'. If all the neighboring elements of the center pixel are 'TRUE' then only the center pixel becomes 'TRUE'. Thus, erosion eliminates the irrelevant image pixels and smooths the contour.

8. Put the resultant 2 D arrays of the dilated images and eroded images to image file to view them.

Part b. c) a) To calculate area of marker

1. For marker image, obtain the output of the closing operation as before.

2. Count the pixels with the 'TRUE' pixel value. This will give the area of the object.

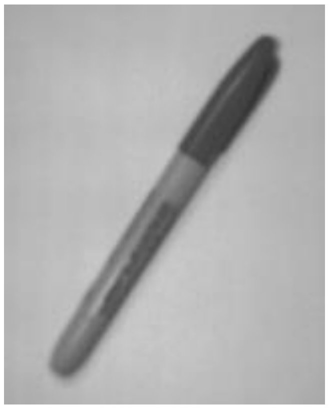
Part b. c) a) To calculate the area of coins

1. Perform thresholding. The objects in the image, i.e coins will have dark intensities.
So, while thresholding, make the pixel as 'TRUE' if the intensity is less than threshold. Else make them dark. (as background)

2. Count the pixels with the 'TRUE' pixel value. This will give the area of the object.

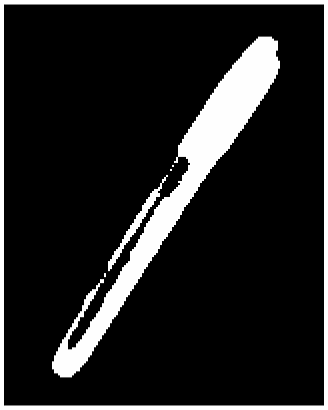
<u>**Results:**</u>

1. Original Marker image

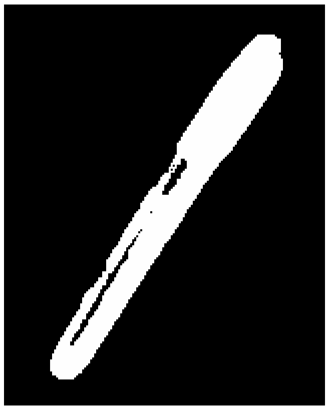


ASSIGNMENT 4

2. Binary marker image

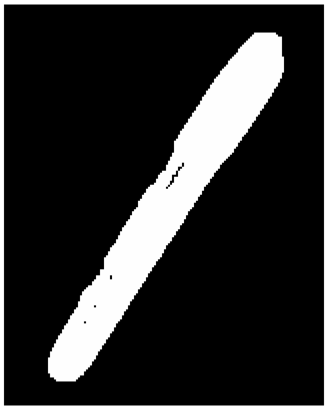


3. Dilated image 1

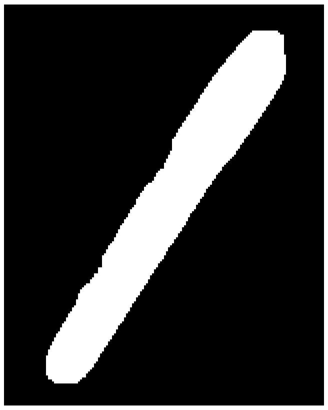


4. Dilated image 2

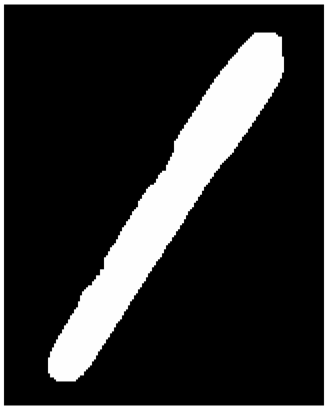
ASSIGNMENT 4



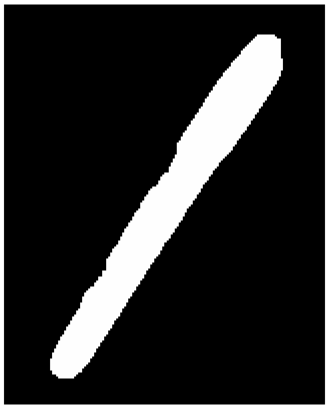
5. Dilated image 3



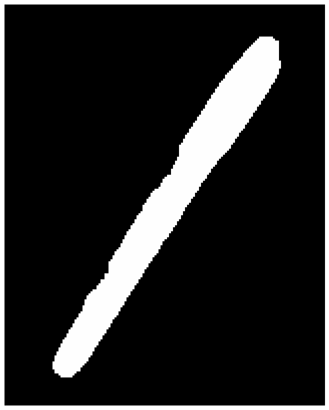
4. Eroded image 1



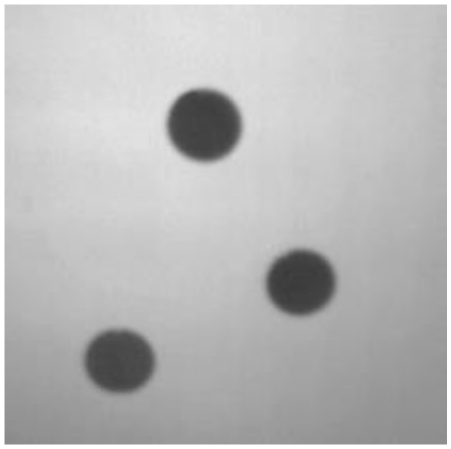
5. Eroded image 2



6. Eroded image 3

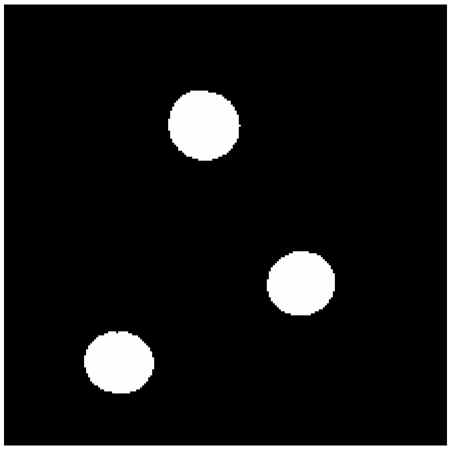


7. Original coins image



ASSIGNMENT 4

8. Binary coins image



Part 2:

The given 5 objects are pen, coins, book, cup and a shell.

a) To recognize simple objects based on their binary images,

   I use simple geometric descriptors i.e centroid, compactness. MAT skeleton to recognize the images.

b) To know the location of the object on the image, centroid is a good parameter.

   Centroid is the point such that the sum of the squares of the distance from it to all other points within the object is minimum. It can be expressed in terms of moments.

   X' = M10/M00 = $\sum_i \sum_j iB[i,j]$ / A

   Y' = M01/M00 = $\sum_i \sum_j jB[i,j]$/ A

ASSIGNMENT 4

d) **Schematic representation of a binary vision system**.

1. Start → Receive an input binary image. Perform morphological operations and get a better shape of the object.

2. Calculate area (A) and perimeter (P) of the object. Hence the compactness. Compactness = P^2/ A.

3. Calculate the centroid as mentioned in (b).

4. To differentiate between pen, book, cup and coin, shell, check if any of the two boundary edges are parallel or not. If they are parallel, then the object is one among pen, book, and cup.

5. To differentiate between pen, book, cup, calculate Euler number = C-H
   where C= number of connected components, H = number of Holes.

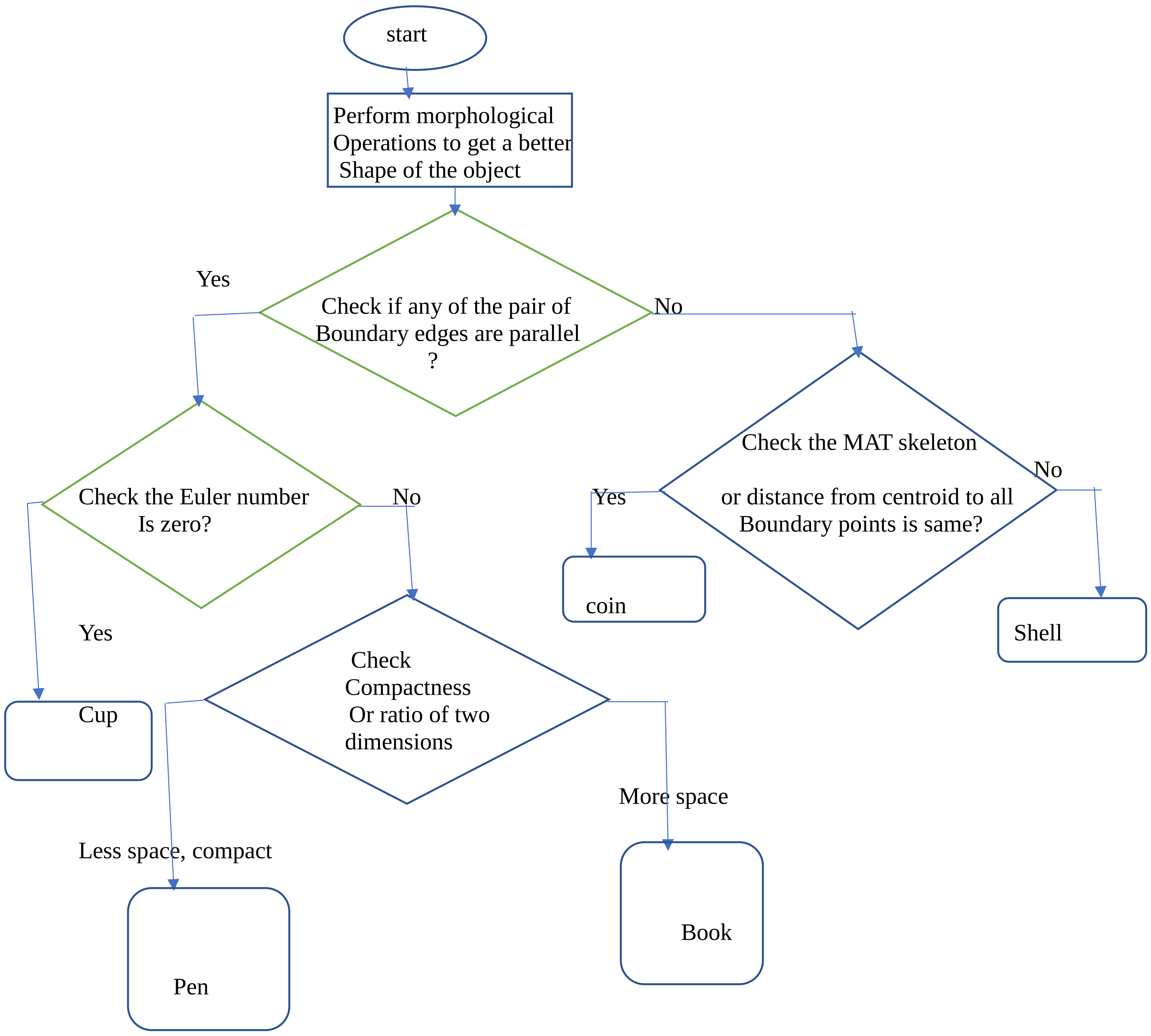
6. For cup, Euler number = 0. For the book and pen Euler number = 1

7. To differentiate between book and pen, compactness will help, pen will occupy less space (more compact).

   Also, ratio of the two dimensions (width and height) can be considered as a factor. For pen, Width will be very less compared to height. For book, width will be less compared to height. But the ratio will be greater than that of pen.

8. To differentiate between coins and shell, MAT (medial axis transformation) skeleton will help. For coins the MAT will be a pixel in the center. For shells, it is not the case. In other words, the distance from the centroid to all points on the boundary will be same for the coins. Whereas it will vary for the shell.

ASSIGNMENT 4


start
Perform morphological Operations to get a better Shape of the object
Check if any of the pair of Boundary edges are parallel ?
Yes
No
Check the Euler number Is zero?
Yes
No
Cup
Check Compactness Or ratio of two dimensions
Less space, compact
More space
Pen
Book
Check the MAT skeleton or distance from centroid to all Boundary points is same?
Yes
No
coin
Shell

ASSIGNMENT 4

Observations:

We can see that the threshold classified some parts of the receiver as background.
Closing operation fills the hole. The output image better represents the shape of the object.

The area of the marker is calculated after the closing operation. The area is nothing but the count of the object pixels. ('TRUE' value pixels). It is estimated to be 3574.

The area of the coins is calculated after the thresholding operation. The area is nothing but the count of the object pixels. ('TRUE' value pixels). It is estimated to be 2685.

Conclusion:

Morphological operators like closing operation output better represents the shape of the object. One of the uses of dilation is to fill in small background color holes in images, e.g. `pepper noise'. One of the problems with doing this, however, is that the dilation will also distort all regions of pixels indiscriminately. By performing an erosion on the image after the dilation, i.e. a closing, we reduce some of this effect. The effect of closing can be quite easily visualized.

Use of simple geometric descriptors like area, perimeter, compactness and centroid (for location of the object) and topological descriptors like Euler number can be used for designing a binary image recognition system.

Source code

1. A) Marker – closing and area calculation

```
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
/*function to dilate the binary image*/
void dilate_test(unsigned char new_image[201][161], unsigned char dilated_image[200][160])
{
	//if any of the neighbourung pixel is true then the pixel becomes true
	      int i,j;
	int i_new,j_new;
	 i_new  = 0;
	for(i= 1;i<200;i++)
	{
			j_new = 0;
			for(j=1;j<160;j++)
			{
			   //	printf("The neigbouring pixels are %d,%d,%d,%d,%d,%d,%d,%d\n",new_image[i-1][j-1],new_image[i-1][j],new_image[i-1][j+1],new_image[i][j-1],new_image[i][j+1],new_image[i+1][j-1],new_image[i+1][j],new_image[i+1][j+1]);
		//	printf("The current pixel value was %d\n",dilated_image[i][j]);
				if((new_image[i-1][j-1] == 255) || (new_image[i-1][j] == 255) || (new_image[i-1][j+1] == 255) ||
(new_image[i][j-1]==255)||(new_image[i][j+1]          ==          255)          ||(new_image[i+1][j-1]          ==
255)||(new_image[i+1][j]==255)||(new_image[i+1][j+1] == 255))
                                        dilated_image[i_new][j_new] = 255;
                      else
                      dilated_image[i_new][j_new] = 0;
                                        j_new++;
                //      printf("The new value of the current pixel is %d\n",dilated_image[i][j]);
                           }
                           i_new++;
              }
}
/*fuction to erode the binary image*/
void erode(unsigned char new_image[201][161], unsigned char eroded_image[200][160])
{
        //if any of the neighbourung pixel is false(dark) then the pixel becomes false(dark)
              int i,j;
        int i_new,j_new;
        i_new = 0;
         for(i=1;i<200;i++)
        {
                           j_new = 0;
              for(j=1;j<160;j++)
                           {
                                //    printf("The    neigbouring    pixels    are    %d,%d,%d,%d,%d,%d,%d,%d\n",new_image[i-1][j-
1],new_image[i-1][j],new_image[i-1][j+1],new_image[i][j-1],new_image[i][j+1],new_image[i+1][j-
1],new_image[i+1][j],new_image[i+1][j+1]);
                    //    printf("The current pixel value was %d\n",dilated_image[i][j]);
                                        if((new_image[i-1][j-1]  ==  0)  ||  (new_image[i-1][j]  ==  0)  ||  (new_image[i-1][j+1]  ==  0)  ||
(new_image[i][j-1]==0)||(new_image[i][j+1]          ==          0)          ||(new_image[i+1][j-1]          ==
0)||(new_image[i+1][j]==0)||(new_image[i+1][j+1] == 0))
                                        eroded_image[i_new][j_new] = 0;
                      else
                      eroded_image[i_new][j_new] = 255;
                                        j_new++;
                //      printf("The new value of the current pixel is %d\n",dilated_image[i][j]);
                           }
                           i_new++;
              }
}
/*main function*/
int main(int argc, char *argv[]){
  unsigned sizeX=200; //image width
  unsigned sizeY=160; //image height
  unsigned char image[sizeX][sizeY]; //image array
  unsigned char  new_image[201][161]; //padded image array
  unsigned levels;
  unsigned char dilated_image1[200][160]= {0,};
  unsigned char dilated_image2[200][160]= {0,};
  unsigned char dilated_image3[200][160]= {0,};
 unsigned char eroded_image1[200][160]= {0,};
 unsigned char eroded_image2[200][160]= {0,};
 unsigned char eroded_image3[200][160]= {0,};
  //read image file
  FILE *fp;
  fp = fopen("marker.pgm","r");
  int i,j,k;
  //reading the image file and putting it to a 2 D array
  if(fp==0) {
   printf("Error in reading marker image\n");
   return 1;
   }
  if(3!=fscanf(fp, "P5 %d %d %d ", &sizeX, &sizeY, &levels)) return 1;
  fread(image,sizeof(unsigned char),sizeX*sizeY,fp);
  fclose(fp);
  int threshold = 128;
 /*here the obhect marker has darker intensity than the background, so I made if the pixel is less than threshold than make it
TRUE else, make it false(as Background)*/
  for(i=0;i<200;i++)
  {
      for(j=0;j<160;j++)
            {
             if(image[i][j] < threshold)
                        image[i][j] = 255;
                        else
             image[i][j] = 0;
            }
  }
        //write  o/p array  to  image file
        FILE *pFile = fopen("binary_image.pgm","w");
        if(pFile==0) return 1; //error handling
        fprintf(pFile, "P5 %d %d %d ",sizeX,sizeY, levels);//write header
        fwrite(image,sizeof(unsigned char),sizeX*sizeY,pFile);//write binary image
        fclose(pFile);
  /*create an 201X161 array full of  0's*/
  for(i=0;i<201;i++)
  {
            for(j=0;j<161;j++)
            {
                        new_image[i][j] = 0;
       }
  }
            //add elements of the binary image to the current array
             //keeping one layer of zeros around the elements of the binary image array
             int i_old,j_old;
             i_old =0;
             for(i=1;i<201;i++)
            {
            j_old =0;
                        for(j=1;j<161;j++)
                        {
                                    new_image[i][j] = image[i_old][j_old];
                                    j_old++;
                        }
                        i_old++;
            }
        dilate_test(new_image,dilated_image1);
             //write  o/p array  to  image file
        FILE *oFile = fopen("dilated_image1.pgm","w");
        if(oFile==0) return 1; //error handling
        fprintf(oFile, "P5 %d %d %d ",sizeX,sizeY, levels);//write header
        fwrite(dilated_image1,sizeof(unsigned char),sizeX*sizeY,oFile);//write dilated image
        fclose(oFile);
             /*create an 201X161 array full of  0's*/
            for(i=0;i<201;i++)
            {
                        for(j=0;j<161;j++)
```

```
                {
                            new_image[i][j] = 0;
        }
        }
         //add elements of the dilated image to the current array
         //keeping one layer of zeros around the elements of the dilated image array
         i_old =0;
         for(i=1;i<201;i++)
        {
        j_old =0;
                for(j=1;j<161;j++)
                {
                            new_image[i][j] = dilated_image1[i_old][j_old];
                            j_old++;
                }
                i_old++;
        }
    dilate_test(new_image,dilated_image2);
          //write  o/p array  to  image file
     oFile = fopen("dilated_image2.pgm","w");
    if(oFile==0) return 1; //error handling
    fprintf(oFile, "P5 %d %d %d ",sizeX,sizeY, levels);//write header
    fwrite(dilated_image2,sizeof(unsigned char),sizeX*sizeY,oFile);//write dilated image
    fclose(oFile);
          /*create an 201X161 array full of  0's*/
        for(i=0;i<201;i++)
        {
                for(j=0;j<161;j++)
                {
                            new_image[i][j] = 0;
        }
        }
         //add elements of the dilated image to the current array
         //keeping one layer of zeros around the elements of the dilated image array
         i_old =0;
         for(i=1;i<201;i++)
        {
        j_old =0;
                for(j=1;j<161;j++)
                {
                            new_image[i][j] = dilated_image2[i_old][j_old];
                            j_old++;
                }
                i_old++;
        }
   dilate_test(new_image,dilated_image3);
          //write  o/p array  to  image file
     oFile = fopen("dilated_image3.pgm","w");
    if(oFile==0) return 1; //error handling
    fprintf(oFile, "P5 %d %d %d ",sizeX,sizeY, levels);//write header
    fwrite(dilated_image3,sizeof(unsigned char),sizeX*sizeY,oFile);//write dilated image
    fclose(oFile);
          /*create an 201X161 array full of  0's*/
        for(i=0;i<201;i++)
        {
                for(j=0;j<161;j++)
                {
                            new_image[i][j] = 0;
        }
        }
         //add elements of the dilated image to the current array
         //keeping one layer of zeros around the elements of the dilated image array
         i_old =0;
         for(i=1;i<201;i++)
        {
        j_old =0;
                for(j=1;j<161;j++)
                {
                            new_image[i][j] = dilated_image3[i_old][j_old];
                            j_old++;
                }
                i_old++;
        }
        erode(new_image,eroded_image1);
          //write  o/p array  to  image file
     oFile = fopen("eroded_image1.pgm","w");
    if(oFile==0) return 1; //error handling
    fprintf(oFile, "P5 %d %d %d ",sizeX,sizeY, levels);//write header
    fwrite(eroded_image1,sizeof(unsigned char),sizeX*sizeY,oFile);//write eroded image
    fclose(oFile);
          /*create an 201X161 array full of  0's*/
        for(i=0;i<201;i++)
        {
                for(j=0;j<161;j++)
                {
                            new_image[i][j] = 0;
        }
        }
         //add elements of the eroded image to the current array
         //keeping one layer of zeros around the elements of the eroded image array
         i_old =0;
         for(i=1;i<201;i++)
        {
        j_old =0;
                for(j=1;j<161;j++)
                {
                            new_image[i][j] = eroded_image1[i_old][j_old];
                            j_old++;
                }
                i_old++;
        }
        erode(new_image,eroded_image2);
          //write  o/p array  to  image file
     oFile = fopen("eroded_image2.pgm","w");
    if(oFile==0) return 1; //error handling
    fprintf(oFile, "P5 %d %d %d ",sizeX,sizeY, levels);//write header
    fwrite(eroded_image2,sizeof(unsigned char),sizeX*sizeY,oFile);//write eroded image
    fclose(oFile);
         /*create an 201X161 array full of  0's*/
        for(i=0;i<201;i++)
        {
                for(j=0;j<161;j++)
                {
                            new_image[i][j] = 0;
        }
        }
```

```
              //add elements of the eroded image to the current array
              //keeping one layer of zeros around the elements of the eroded image array
              i_old =0;
              for(i=1;i<201;i++)
             {
             j_old =0;
                     for(j=1;j<161;j++)
                     {
                             new_image[i][j] = eroded_image2[i_old][j_old];
                             j_old++;
                     }
                     i_old++;
             }
             erode(new_image,eroded_image3);
               //write  o/p array  to  image file
            oFile = fopen("eroded_image3.pgm","w");
           if(oFile==0) return 1; //error handling
           fprintf(oFile, "P5 %d %d %d ",sizeX,sizeY, levels);//write header
           fwrite(eroded_image3,sizeof(unsigned char),sizeX*sizeY,oFile);//write eroded image
           fclose(oFile);
          int area = 0;
          for(i=0;i<200;i++)
              {
                     for(j=0;j<160;j++)
                     {
                             if(eroded_image3[i][j] == 255) //here the object has bright pixels
                             {
                                     area++;
                             }
                     }
             }
             printf("Area of the marker is %d\n", area);
             return 0;
}
```

2. Coins image area calculation

```
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
/*main function*/
int main(int argc, char *argv[]){
  unsigned sizeX=220; //image height
  unsigned sizeY=221; //image width
  unsigned char image[sizeX][sizeY]; //image array
  unsigned levels;
  //read image file
  FILE *fp;
  fp = fopen("coins.pgm","r");
  int i,j;
  int area = 0;
  //reading the image file and putting it to a 2 D array
  if(fp==0) {
   printf("Error in reading coin image\n");
   return 1;
```

ASSIGNMENT 4

```
   }
  if(3!=fscanf(fp, "P5 %d %d %d ", &sizeX, &sizeY, &levels)) return 1;
  fread(image,sizeof(unsigned char),sizeX*sizeY,fp);
  fclose(fp);
  int threshold = 120; //to create a binary image
              for(i=0;i<220;i++)
              {
                         for(j=0;j<221;j++)
                         {
                                     if(image[i][j] < threshold)  //here the coins which are object have dark intensities, so I made dark
ppixels as the object and the light pixels as background
                                     {
                                                 image[i][j] = 255;
                                     }
                                     else
                                     {
                                                 image[i][j] = 0;
                                     }
                         }
             }
              //write  o/p array  to  image file
        FILE *oFile = fopen("binary_coin.pgm","w");
        if(oFile==0) return 1; //error handling
        fprintf(oFile, "P5 %d %d %d ",sizeX,sizeY, levels);//write header
        fwrite(image,sizeof(unsigned char),sizeX*sizeY,oFile);//write eroded image
        fclose(oFile);
             for(i=0;i<220;i++)
              {
                         for(j=0;j<221;j++)
                         {
                                     if(image[i][j] == 255) //if it is a coin pixel , count it towards the area
                                     {
                                                 area++;
                                     }
                         }
             }
            printf("Area of the coins is %d\n",area);
            return 0;
}
```