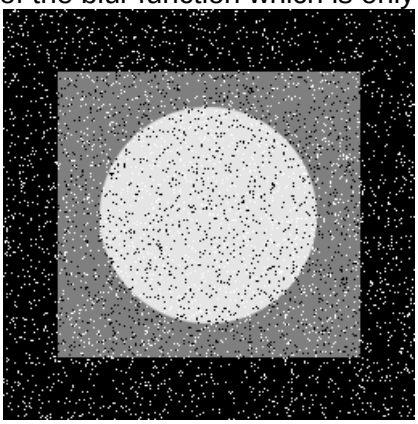
Lab4: Edge Detection and Hough Transform

Ex1. For the image “salt.tif”.
You’re required to apply canny edge detection on the image as in sample #1 “Canny Edge Detection.cpp”, and using appropriate canny parameters produce an edge image that will be robust to noise, and produce only the actual edges for the square and the circle.

Use the edge image to draw a red rectangle around the rectangle in the image.
Hint: To be able to do this you’ll need to tune canny parameters (Low threshold, High threshold, Sigma for Gaussian).
Hint: The Gaussian is applied explicitly before canny edge detection, so you’ll need to call Gaussian Blur explicitly instead of the blur function which is only a normal average filter.



To Access RGB Pixels use the following code

```
Mat x = imread("asd.jpg");  // --> asd.jpg is the file name
for(int i = 0;i<x.rows;i++)
	for (int j = 0; j < x.cols; j++)
	{
		if(i = x.row/2)
		{
			x.at<Vec3b>(i,j).val[0] = 255; // ->val[0] -> blue channel
			// val[1] -> green channel
			// val[2] -> red channel
		}
	}
```

This code will draw a horizontal blue line at center.

Ex2. Using the createTrackbar function in sample #1 Canny Edge Detector.cpp

```
createTrackbar( "Min Threshold:", window_name, &lowTh, maxlowTh, CannyTh);
```

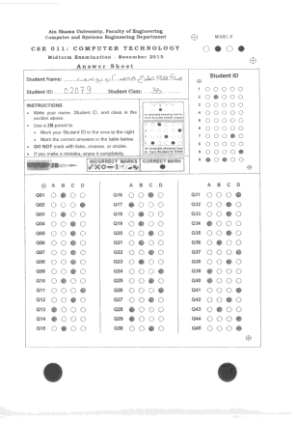
Make a program to draw the image after thresholding and make a trackbar to control the threshold, refer to sample#1 for guidance.

For the 4 images (coins.png, cameraman.tif, answersheet.jpg,ad2.jpg), what are the appropriate threshold values that can help in further processing (differentiate between foreground and background)?

Instead of explicitly defining the threshold value use the following function from opencv

threshold(im_gray, img_bw, 0, 255, CV_THRESH_BINARY | CV_THRESH_OTSU);

- `im_gray` is a source 8-bit image, if it's a colored image convert it to gray scale first
- `img_bw` is a result,
- 0 means threshold level which actually is omitted because we used CV_THRESH_OTSU flag,
- 255 is a value that is going to be assigned to respectively pixels in the result (namely, to all pixels which value in the source is greater then computed threshold level)
- `CV_THRESH_BINARY | CV_THRESH_OTSU` is a required flag to perform Otsu thresholding. Because in fact we would like to perform binary thresholding, so we use `CV_THRESH_BINARY` (you can use any of 5 flags opencv provides) combined with `CV_THRESH_OTSU`



Ex3.Try the function adaptive threshold on “ada2.jpg” and “intensity.tif” with different parameters.
Make them available for you in trackBars.
What do you observe? Is it better to apply an adaptive threshold or a global threshold? When to use each?

void **adaptiveThreshold**(InputArray **src**, OutputArray **dst**, double **maxValue**, int **adaptiveMethod**, int **thresholdType**, int **blockSize**, double **C**)

Parameters:

src – Source 8-bit single-channel image.

dst – Destination image of the same size and the same type as src .

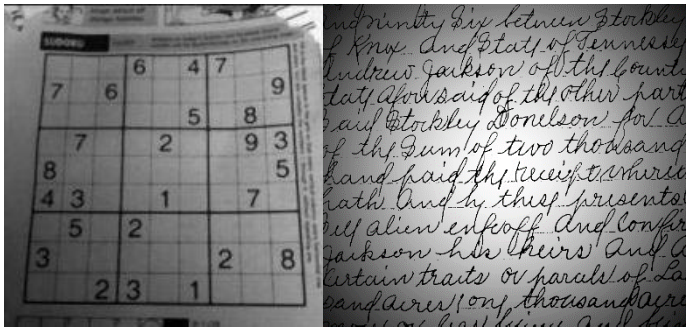
maxValue – Non-zero value assigned to the pixels for which the condition is satisfied. See the details below.

adaptiveMethod – Adaptive thresholding algorithm to use,ADAPTIVE_THRESH_MEAN_C or ADAPTIVE_THRESH_GAUSSIAN_C . See the details below.

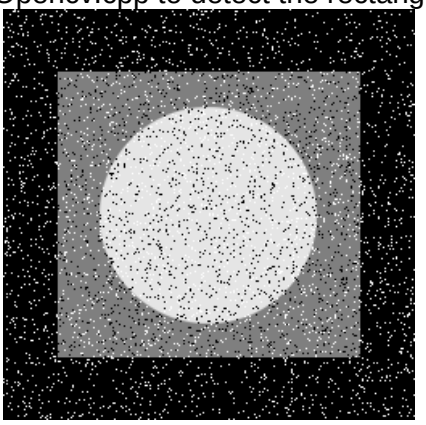
thresholdType – Thresholding type that must be either THRESH_BINARY orTHRESH_BINARY_INV .

blockSize – Size of a pixel neighborhood that is used to calculate a threshold value for the pixel: 3, 5, 7, and so on.

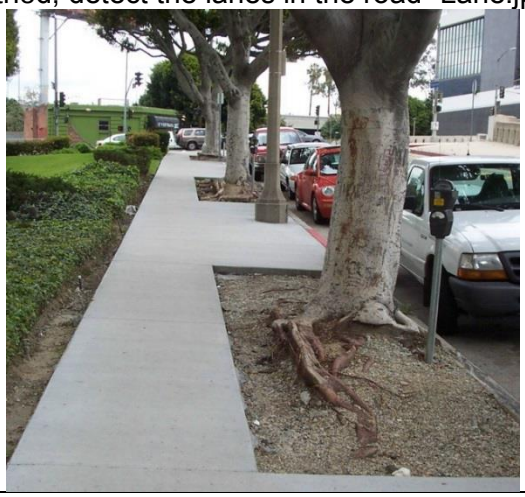
C – Constant subtracted from the mean or weighted mean (see the details below). Normally, it is positive but may be zero or negative as well.



Ex4. From Exercise 1, using the same edge image generated and the Hough lines methods used in Sample #5 HoughLinesOpencv.cpp to detect the rectangle four lines.



Ex5. Using HoughLines method, detect the lanes in the road “Lane.jpg”



a. **The Standard Hough Transform**

- It consists in pretty much what we just explained in the previous section. It gives you as result a vector of couples (θ, r_θ)
- In OpenCV it is implemented with the function HoughLines

b. **The Probabilistic Hough Line Transform**

- A more efficient implementation of the Hough Line Transform. It gives as output the extremes of the detected lines (x_0, y_0, x_1, y_1)
- In OpenCV it is implemented with the function HoughLinesP

void **HoughLinesP**(InputArray **image**, OutputArray **lines**, double **rho**, double **theta**, int **threshold**, double **minLineLength**=0, double **maxLineGap**=0)

image – 8-bit, single-channel binary source image. The image may be modified by the function.
lines – Output vector of lines. Each line is represented by a 4-element vector (x_1, y_1, x_2, y_2), where (x_1, y_1) and (x_2, y_2) are the ending points of each detected line segment.
rho – Distance resolution of the accumulator in pixels.
theta – Angle resolution of the accumulator in radians.
threshold – Accumulator threshold parameter. Only those lines are returned that get enough votes ($>$ `threshold`).
minLineLength – Minimum line length. Line segments shorter than that are rejected.
maxLineGap – Maximum allowed gap between points on the same line to link them.

Q6. Using the hough circles sample #6, try to count how much money in the image “coins.png”, and the bigger coin value is 100 cents, and the small one is 50 cents.

void **HoughCircles**(InputArray **image**, OutputArray **circles**, int **method**, double **dp**, double**minDist**, double **param1**=100, double **param2**=100, int **minRadius**=0, int **maxRadius**=0)

image – 8-bit, single-channel, grayscale input image.

circles – Output vector of found circles. Each vector is encoded as a 3-element floating-point vector $(x, y, radius)$.

circle_storage – In C function this is a memory storage that will contain the output sequence of found circles.

method – Detection method to use. Currently, the only implemented method is`CV_HOUGH_GRADIENT` , which is basically *21HT* , described in [Yuen90].

dp – Inverse ratio of the accumulator resolution to the image resolution. For example, if `dp=1` , the accumulator has the same resolution as the input image. If `dp=2` , the accumulator has half as big width and height.

minDist – Minimum distance between the centers of the detected circles. If the parameter is too small, multiple neighbor circles may be falsely detected in addition to a true one. If it is too large, some circles may be missed.

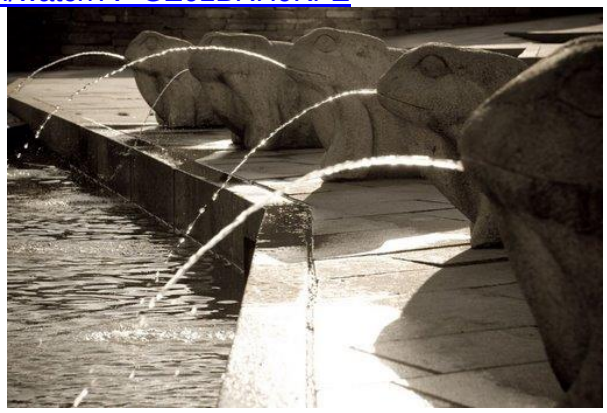
param1 – First method-specific parameter. In case of `CV_HOUGH_GRADIENT` , it is the higher threshold of the two passed to the **`Canny()`** edge detector (the lower one is twice smaller).

param2 – Second method-specific parameter. In case of `CV_HOUGH_GRADIENT` , it is the accumulator threshold for the circle centers at the detection stage. The smaller it is, the more false circles may be detected. Circles, corresponding to the larger accumulator values, will be returned first.

minRadius – Minimum circle radius.

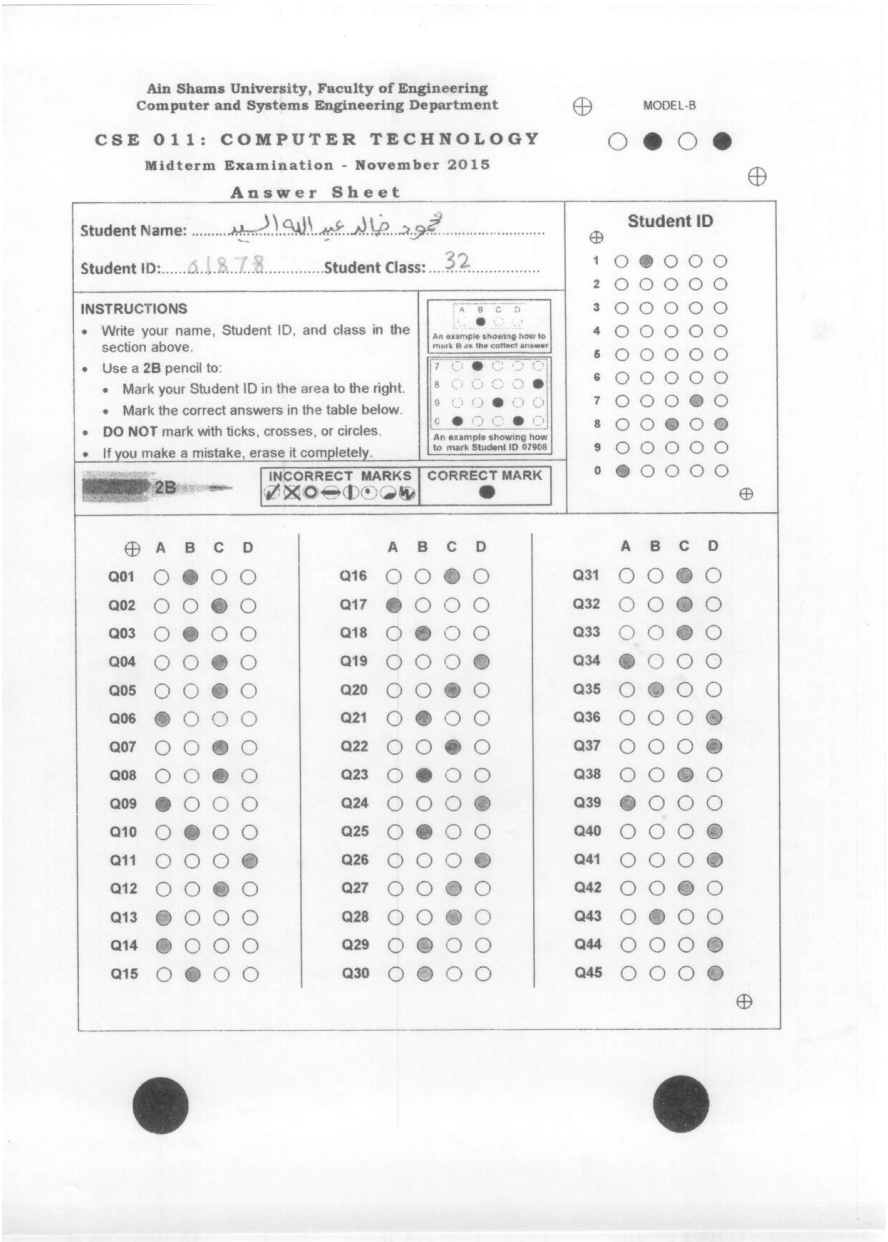
maxRadius – Maximum circle radius.

Q7.Using the parabola model, $y- y_o = a(x-x_o)$ ^2, write your own code for Hough transform implementation to detect water streams for image “Grenouilles.jpg”.
This video will help you
https://www.youtube.com/watch?v=GZ61BRH9KFE



Q8.Given the folder “Marks Problem” that contains answer sheets for students, you are required to make an auto correction algorithms to take as input a list of images and produce an excel sheet exactly the same as “degrees.xlsx” programmatically showing each seat number and the corresponding degree using what you’ve learned so far.

A sample of the answer sheet is



Ain Shams University, Faculty of Engineering
Computer and Systems Engineering Department

MODEL-B

CSE 011: COMPUTER TECHNOLOGY

Midterm Examination - November 2015

Answer Sheet

Student Name: محمود خالد عبد الوالي

Student ID: 21878 Student Class: 32

INSTRUCTIONS

- Write your name, Student ID, and class in the section above.
- Use a 2B pencil to:
  - Mark your Student ID in the area to the right.
  - Mark the correct answers in the table below.
- DO NOT mark with ticks, crosses, or circles.
- If you make a mistake, erase it completely.

An example showing how to mark B as the correct answer

An example showing how to mark Student ID 07908

2B

INCORRECT MARKS

CORRECT MARK

Student ID

	1	2	3	4	5	6	7	8	9	0

	A	B	C	D
Q01		●		
Q02			●	
Q03		●		
Q04			●	
Q05			●	
Q06	●			
Q07			●	
Q08			●	
Q09	●			
Q10		●		
Q11				●
Q12			●	
Q13	●			
Q14	●			
Q15		●		
Q16			●	
Q17	●			
Q18		●		
Q19				●
Q20			●	
Q21		●		
Q22			●	
Q23		●		
Q24				●
Q25		●		
Q26				●
Q27			●	
Q28			●	
Q29		●		
Q30		●		
Q31			●	
Q32			●	
Q33			●	
Q34	●			
Q35		●		
Q36				●
Q37				●
Q38			●	
Q39	●			
Q40				●
Q41				●
Q42			●	
Q43		●		
Q44				●
Q45				●