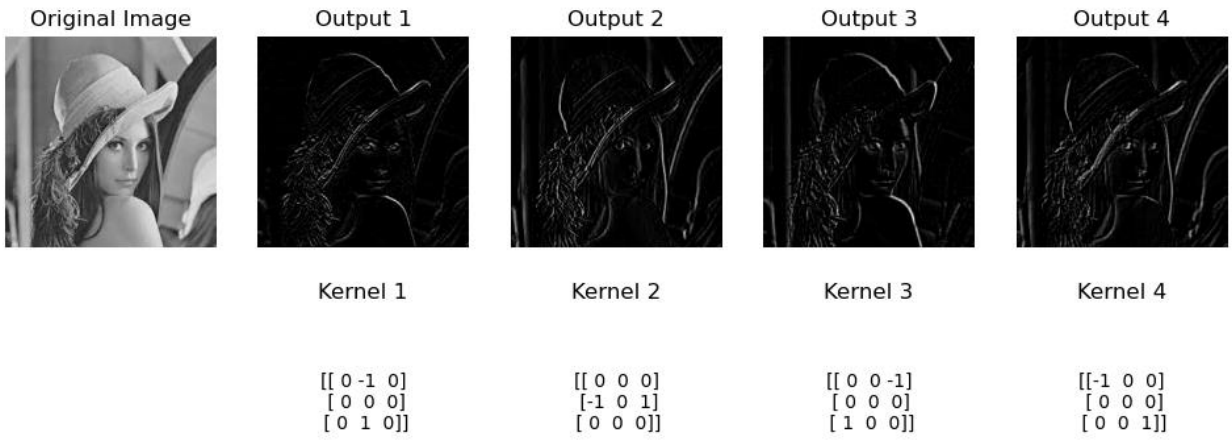
Mehrdad Baradaran - 99222020

Assignment 02

professor: Dr. Mahmoudi Aznaveh

Q1 Answer:

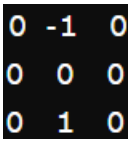
At first, regardless of considering the impact of each kernel and the effects it can exert, we apply the code of each of the kernels on the original image and observe the result, which is as follows:



Now that the correct mapping of each kernel and its output as an image has been presented, we will continue discussing the reasons behind this mapping and the effects of each kernel.

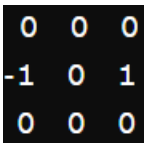
Next, we will write each kernel separately, first describing its application and then highlighting its differences from the other kernels mentioned. Subsequently, we will elucidate its effects.

Kernel1:



- **Usage**: This kernel is also used for edge detection, but specifically for detecting horizontal edges.
- **Differences**: it emphasizes the difference in intensity between adjacent pixels, but this time horizontally. It subtracts the intensity value of the pixel on the left from the pixel on the right, hence highlighting horizontal edges.
- **Effect**: When convolved with an image, it enhances horizontal edges by producing a high response where there is a significant intensity difference between the pixels on the left and right.

Kernel2:



- **Usage**: This kernel is commonly used for edge detection, specifically for detecting vertical edges.
- **Differences**: It emphasizes the difference in intensity between adjacent pixels vertically. It subtracts the intensity value of the pixel below from the pixel above, hence highlighting vertical edges.
- **Effect**: When convolved with an image, it enhances vertical edges by producing a high response where there is a significant intensity difference between the pixels above and below.

Kernel3:

```
 0  0 -1
 0  0  0
 1  0  0
```

- **Usage**: this kernel is used for detecting diagonal edges, but specifically those running from top-left to bottom-right.
- **Differences**: It emphasizes the difference in intensity between pixels diagonally, but in the opposite direction to kernel 3. It subtracts the intensity value of the pixel in the top-right from the pixel in the bottom-left.
- **Effect**: When convolved with an image, it enhances diagonal edges running from top-left to bottom-right by producing a high response where there is a significant intensity difference between the pixels in those directions.

Kernel4:

```
-1  0  0
 0  0  0
 0  0  1
```

- **Usage**: This kernel is often used for detecting diagonal edges, specifically those running from top-right to bottom-left.
- **Differences**: It emphasizes the difference in intensity between pixels diagonally. It subtracts the intensity value of the pixel in the top-left from the pixel in the bottom-right.
- **Effect**: When convolved with an image, it enhances diagonal edges running from top-right to bottom-left by producing a high response where there is a significant intensity difference between the pixels in those directions.

Q2 Answer:

In the beginning, we examine what each of the filters is and how it operates.

A **Laplacian filter**, commonly known as the Laplacian of Gaussian (LoG), is a spatial filter that is used in image processing to detect edges and for other purposes. It is a second-order derivative operator that determines the rate at which image intensity changes. The Laplacian operator is particularly useful for detecting edges since it identifies regions with fast intensity variations.
This is how it works.
Convolution with Gaussian Filter: Before using the Laplacian operator, the picture is convolved with a Gaussian filter. This stage smoothes the image and reduces noise, hence improving the recognition of relevant edges while suppressing noise.
Application of the Laplacian Operator: After smoothing with the Gaussian filter, the Laplacian operator is used. It calculates the second derivative of image intensity.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Edge detection: Areas of the picture where the intensity fluctuates quickly are highlighted by the Laplacian operator. These areas usually line up with the image's edges. In the Laplacian-filtered image, edges are identified as zero-crossings, where the intensity's sign shifts. An edge from light to dark is indicated by a positive to negative zero-crossing, while an edge from dark to light is indicated by a negative to positive zero-crossing.

The Laplacian filter is a potent tool for edge recognition in image processing jobs because it locates edges in an image by identifying these zero-crossings.

For edge detection in particular, the **Sobel filter** is another popular spatial filter in image processing. The method computes the gradient of the image intensity using convolution with small, separable kernels. Detecting regions of high spatial intensity change allows the Sobel filter to highlight edges.

Sobel filter operation is as follows:

Gradient Calculation: The two independent kernels that make up the Sobel filter are one for identifying changes in the horizontal direction and one for the vertical direction. The gradient in the horizontal and vertical directions is independently calculated using these kernels convolved with the image. Both the vertical and horizontal kernels draw attention to different aspects of the edges.

The following operations are used to convolve the kernels with the image:

Horizontal Sobel Kernel:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Vertical Sobel Kernel:

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Gradient Magnitude: After convolving the image with the horizontal and vertical kernels, the gradient magnitude is computed by combining the absolute values of the gradients obtained from both directions.

$$G = \sqrt{{G_x}^2 + {G_y}^2}$$

Edge Detection: The gradient magnitude image obtained highlights areas of significant intensity change, indicating the presence of edges in the image.

The Sobel filter is computationally efficient and widely used due to its simplicity and effectiveness in detecting edges. It's a fundamental tool in many image processing tasks, including object detection, image segmentation, and feature extraction.

A well-liked multi-stage technique for edge detection in digital photos is the **Canny edge detector**. It was created in 1986 by John F. Canny and is renowned for its excellent accuracy and capacity to identify a variety of edges with the least amount of false positives.

The following are the steps in the Canny edge detection algorithm:

Smoothing: To start, the image is subjected to a Gaussian filter to remove noise. Noise can lead to the detection of spurious edges, which can be avoided by smoothing the image.

Calculating the Gradient: The algorithm then determines the size and direction of the gradient for each pixel in the smoothed picture. Usually, Sobel kernels are used to compute the gradient both vertically and horizontally.

Non-maximum Suppression: This stage entails reducing the edges such that only the local maxima in the gradient direction remain. In other words, the algorithm evaluates each pixel to see if the gradient magnitude is at its greatest along the gradient direction. If it is, the pixel is saved; otherwise, it is suppressed.

Double Thresholding: The technique uses two thresholds to identify edge pixels as strong, weak, or non-edge pixels. Pixels with gradient magnitudes more than the high threshold are termed strong edge pixels, whereas those less than the low threshold are regarded non-edge pixels. Pixels with gradient magnitudes between the two thresholds are considered weak edge pixels.

Edge Tracking using Hysteresis: Lastly, the validity of the weak edge pixels is assessed. Strong edge pixels are connected by chains of weak edge pixels to achieve this. A weak edge pixel is regarded as a component of the edge if it is close to a strong edge pixel. Otherwise, it gets thrown away.

A binary image with white pixels representing edge pixels and black pixels representing non-edge pixels

is the output of the Canny edge detection technique.

The stability and accuracy of the Canny edge detector under different settings make it a popular choice for tasks like object detection, image segmentation, and feature extraction in computer vision and image processing applications.

Now that we've understood the definitions and functionalities of each, let's examine three reasons why the Laplacian filter performs poorly.

Sensitivity to Noise: The Laplacian filter is more susceptible to noise than the Sobel and Canny techniques. Because it computes the second derivative immediately, picture noise might produce false edges or edge detection problems. Sobel and Canny methods usually contain a smoothing step (commonly with a Gaussian filter) to minimize noise before edge detection, making them more resistant in noisy situations.

Edge Localization: When compared to the Sobel and Canny approaches, the Laplacian filter may provide poor edge localization. It detects edges by looking for zero-crossings in the second derivative of intensity, which may not accurately pinpoint the edges. In contrast, the Sobel and Canny approaches compute gradients and apply thresholding techniques to precisely detect edges, resulting in superior edge localization in the image.

Edge Connectivity: The Laplacian filter may cause the image's edges to be broken or discontinuous. Because it just detects zero-crossings without taking gradient direction into account, it may fail to connect neighboring edge pixels. Sobel and Canny approaches, which compute gradients and execute non-maximum suppression before edge tracking, are more effective at preserving edge connectivity, resulting in more continuous and meaningful edge contours in the output image.

Q3 Answer:

Part A:

The Harris corner identification technique is a way to identify corners in images. It determines the "cornerness" of each pixel in the image. The above equation is part of the Harris corner response function, which is used to calculate the amount of corner-like structure at each pixel.

Let's break down the equation.

- E(u,v) is the corner response function for pixel coordinates (u,v).

- The image's pixel coordinates are denoted as (u,v).
- The structure tensor (M) is a 2x2 matrix that represents the local image structure surrounding a pixel.
- w(x,y) is a window function (usually Gaussian) that weights the contributions of surrounding pixels.
- Ix and Iy represent image gradients in the horizontal (x) and vertical (y) directions, respectively.

The structural tensor M's elements are computed by adding the gradients Ix and Iy within the window.

- M11 = ∑(x,y)w(x,y) ·Ix2.
- M12 = M21 = ∑(x,y)w(x,y)·Ix·Iy

- M22 = ∑(x,y)w(x,y)·Iy2

The structure tensor matrix M is represented by the entries M11, M12, and M22. They are the second-moment matrix of image gradients within the current window.
The Harris response function assesses the eigenvalues of this matrix M to detect the presence of corners. The determinant and trace of M are used to compute a corner response E(u,v) at each pixel.

E(u,v)=det(M)−k·trace(M)2

- *The determinant of M (det(M)) indicates the image's gradient spread.*
- *Trace(M) is the sum of M's diagonal elements and reflects the local intensity variations.*

The parameter k controls the algorithm's sensitivity and is often set to a minor value (e.g., 0.04 - 0.06). Higher values of E(u,v) imply a greater corner response at the pixel (u,v), indicating the presence of a corner. Corners are usually discovered when both eigenvalues of M are large, which corresponds to locations with high intensity changes in various directions.

Part B:

In the Harris corner identification procedure, the link between the eigenvalues of the structural tensor matrix M and the image gradients Ix and Iy is critical. The local image structure is shown by the eigenvalues of M, which also aid in identifying if a pixel refers to a corner, an edge, or a flat area.
The definition of the structure tensor M is as follows:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Upon calculating M's eigenvalues, we derive two values, commonly represented as $\lambda 1$ and $\lambda 2$. The local visual structure at each pixel is described by these eigenvalues.

Magnitudes of Eigenvalues: The spread or magnitude of gradients in the corresponding directions is shown by the eigenvalue magnitudes. A corner is indicated if there are notable fluctuations in intensity in various directions, as suggested by both big eigenvalues.
Eigenvalue Ratio: To distinguish between corners, edges, and flat sections, one can frequently use the ratio of the eigenvalues, $\lambda 1/\lambda 2$:
Given that both eigenvalues are roughly equal and important, a corner is implied if $\lambda 1/\lambda 2$ is big.
One eigenvalue predominates over the other, indicating a large intensity change in one direction, and if $\lambda 1/\lambda 2$ is near to 1, it denotes an edge.

Because both eigenvalues are tiny and suggest little to no intensity variation, a small value for $\lambda 1/\lambda 2$ denotes a flat region.

Part C:

To establish whether each pixel represents an edge, a corner, or a flat zone, we must compute the corner response measure for each pixel using the Harris corner detection algorithm. The corner response measure $E(u,v)$ for a pixel at coordinates (u,v) is given by: $\mathrm{E}(u,v)$=det(M)−k·trace(M)^2.
Where M denotes the structural tensor :

$$M = \begin{bmatrix} \sum_{x,y} w(x,y).I_x^2 & \sum_{x,y} w(x,y).I_x I_y \\ \sum_{x,y} w(x,y).I_x I_y & \sum_{x,y} w(x,y).I_y^2 \end{bmatrix}$$

Here, $w(x,y)$ is the window function, commonly a Gaussian, and Ix and $\mathrm{I}y$ are the picture gradients in the horizontal and vertical directions respectively.
To calculate the corner response measure for each pixel, use the accompanying image and gradients (d/dx and d/dy).

Input Image:

0	0	1	4	9
1	0	5	7	11
1	4	9	12	16
3	8	11	14	16
8	10	15	16	20

And the image gradients:

```
d/dx: -1 0 1
d/dy: -1 0 1
```

I will start by obtaining the derivative in terms of X using this differentiation kernel. For the first pixel, I'll place the center of the kernel on the pixel itself. The zero will be on the zero, the negative 1 will be on the one, and the one will be on the five. Then, I will calculate 1 multiplied by -1, which equals -1, plus 0 multiplied by 0, which equals 0, plus 5 multiplied by 1, which equals 5. This gives us -1 + 5, which equals 4.

For the next pixel, I'll again position the center of the kernel on the pixel itself. The zero will be on the five, the -1 will be on the 0, and the 1 will be on the 7. Multiplying 0 by -1 gives 0, plus 5 multiplied by 0 gives 0, plus 7 multiplied by 1 gives 7. This yields the answer 7.

For the following pixel, I'll once more place the center on the pixel. The zero will be on the 7, the -1 will be on the 5, and the 1 will be on the 11. Multiplying -1 by 5 gives -5, plus 7 multiplied by 0 gives 0, plus 11 multiplied by 1 gives 11. This results in -5 + 11, which equals 6. This process continues for the other pixels.

Ix

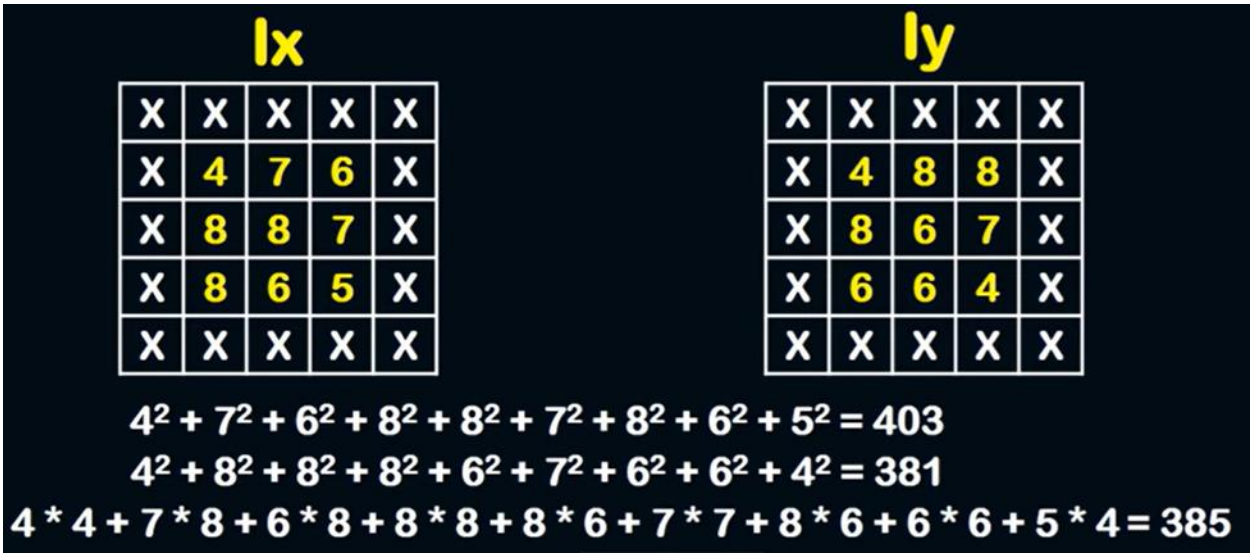
X	X	X	X	X
X	4	7	6	X
X	8	8	7	X
X	8	6	5	X
X	X	X	X	X

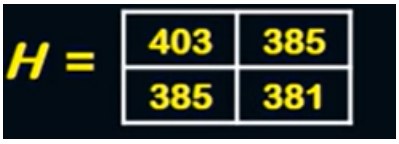
Now, using this differentiation kernel, we will begin by obtaining the derivative in terms of Y. For this first pixel, we will once more place the kernel's center on the pixel, causing this 0 to be on the zero, this negative one to be on the zero, and this one to be on the four. In this way, 0 by negative 1 is equal to 0 plus 0 by 0 is equal to zero plus four by one, which is equal to 4, as this is the case once more for this pixel.

The center, which is zero, will be placed on this five, causing a negative one to be on this line. One by negative one is equal to negative one plus five multiplied by zero is equal to zero plus nine multiply by one, which is equal to nine, so we have negative one plus nine, and the answer is eight in this manner. This process continues for the other pixels in this instance after obtaining the X and Y derivatives.

We can now get the Harris Matrix to do this by saying 4 power 2 like this plus 7 power 2 plus 6 power 2 plus 8 power 2 plus a power 2 again plus seven power two plus eight power two plus six power two plus five power two and this will be the result now we will put this result in this upper left corner like this okay and we will do the same for the line so we will say four power two plus a power 2 plus a power 2 Plus 8 power 2 plus 6 power 2 plus 7 power 2 plus 6 power 2 plus 6 power 2 again plus 4 power 2 and so on

will be the result and we will put this result in the right bottom corner like this okay now we will do the following we will say 4 multiply by 4 plus 7 multiplied by eight plus six multiplied by eight plus eight multiplied by eight plus eight multiplied by six plus seven multiplied by seven plus eight multiplied by six plus six multiplied by six plus five multiplied by four and this will be the result and we will put this result in the upper right corner an





now after getting the Harris Matrix like this now we can determine if the region is a corner age or a flat to do this we will get the Harris Corner score this would be equal to the determinant of H which is the Matrix and to get the determinant of a two by two Matrix like this simply we will multiply these two numbers together minus these two numbers multiplied by each other and this will be the result my NSK which is a constant the value of this K in this case will be equal to 0.04 like this multiplied by the trace of H power 2 and to get the trace of a matrix simply add the values on the diagonal so in this case this is the diagonal so we will sum this Value Plus this value and this will be the result and this will be the final result now to determine if the region is a corner H or a flat we will check these conditions now if E is a large value this will be a corner now in this case this is a negative so it will not be a corner now if E is a negative this will be an H in this case E is a negative so in this case this will be an edge and if the magnitude of E is small this will be a flat so in this case since it is negative it will be an age

E(u,v)=det(H)−k·trace(H)^2

E = 5318 – 0.04*(784)^2 = -19268.24

If E is positive: corner

If E is negative: edge

If |E| is small: flat region

Q4 Answer:

Filtering with Replacement Filter: We start by applying the replacement filter to the given image. The replacement filter is defined as:

$$\begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix}$$

Then we convolve this filter with the given image to get the filtered image.

```
0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
0.0 0.4375    1.0   1.75 2.8125    3.5 2.8125   1.75    1.0 0.4375    0.0
0.0    1.5  3.125    4.5 6.5625  8.125 6.5625    4.5  3.125    1.5    0.0
0.0    2.5 4.8125 6.4375  9.125  11.25  9.125 6.4375 4.8125    2.5    0.0
0.0 3.6875 6.1875   7.75   10.5 12.625   10.5   7.75 6.1875 3.6875    0.0
0.0    4.5  7.125  8.625  11.25  13.25  11.25  8.625  7.125    4.5    0.0
0.0 3.6875 6.1875   7.75   10.5 12.625   10.5   7.75 6.1875 3.6875    0.0
0.0    2.5 4.8125 6.4375  9.125  11.25  9.125 6.4375 4.8125    2.5    0.0
0.0    1.5  3.125    4.5 6.5625  8.125 6.5625    4.5  3.125    1.5    0.0
0.0 0.4375    1.0   1.75 2.8125    3.5 2.8125   1.75    1.0 0.4375    0.0
0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
```

Calculating Gradient with Sobel Operator: After obtaining the filtered image, we apply the Sobel operator to calculate the gradient magnitude. The Sobel operator is defined in both horizontal and vertical directions as follows:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

We convolve the image with both Gx and Gy to get the gradient images in the horizontal and vertical directions, respectively. Then we calculate the gradient magnitude using the formula:

Gradient Magnitude = $\sqrt{G_x^2 + G_y^2}$

This gives us the gradient magnitude of the image.

Gradient X:

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	5.125	5.625	7.0625	7.125	0.0	-7.125	-7.0625	-5.625	-5.125	0.0
0.0	12.0625	11.25	13.0	13.8125	0.0	-13.8125	-13.0	-11.25	-12.0625	0.0
0.0	18.9375	14.9375	16.375	18.125	0.0	-18.125	-16.375	-14.9375	-18.9375	0.0
0.0	24.3125	16.1875	17.0625	19.1875	0.0	-19.1875	-17.0625	-16.1875	-24.3125	0.0
0.0	26.625	16.375	16.875	19.0	0.0	-19.0	-16.875	-16.375	-26.625	0.0
0.0	24.3125	16.1875	17.0625	19.1875	0.0	-19.1875	-17.0625	-16.1875	-24.3125	0.0
0.0	18.9375	14.9375	16.375	18.125	0.0	-18.125	-16.375	-14.9375	-18.9375	0.0
0.0	12.0625	11.25	13.0	13.8125	0.0	-13.8125	-13.0	-11.25	-12.0625	0.0
0.0	5.125	5.625	7.0625	7.125	0.0	-7.125	-7.0625	-5.625	-5.125	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Gradient Y:

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	6.125	12.25	18.6875	25.75	29.375	25.75	18.6875	12.25	6.125	0.0
0.0	7.9375	14.375	19.5	25.0625	28.125	25.0625	19.5	14.375	7.9375	0.0
0.0	7.4375	11.5625	13.5	15.625	16.875	15.625	13.5	11.5625	7.4375	0.0
0.0	6.3125	8.8125	8.8125	8.4375	8.25	8.4375	8.8125	8.8125	6.3125	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	-6.3125	-8.8125	-8.8125	-8.4375	-8.25	-8.4375	-8.8125	-8.8125	-6.3125	0.0
0.0	-7.4375	-11.5625	-13.5	-15.625	-16.875	-15.625	-13.5	-11.5625	-7.4375	0.0
0.0	-7.9375	-14.375	-19.5	-25.0625	-28.125	-25.0625	-19.5	-14.375	-7.9375	0.0
0.0	-6.125	-12.25	-18.6875	-25.75	-29.375	-25.75	-18.6875	-12.25	-6.125	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Gradient Magnitude:

Column	0	1	2	3	4	5	6	7	8	9	10
Row											
0	0.0	0.000000	0.000000	0.000000	0.000000	0.000	0.000000	0.000000	0.000000	0.000000	0.0
1	0.0	7.986316	13.479730	19.977526	26.717562	29.375	26.717562	19.977526	13.479730	7.986316	0.0
2	0.0	14.439800	18.253852	23.436083	28.616675	28.125	28.616675	23.436083	18.253852	14.439800	0.0
3	0.0	20.345646	18.889688	21.222409	23.930237	16.875	23.930237	21.222409	18.889688	20.345646	0.0
4	0.0	25.118625	18.430825	19.203881	20.960715	8.250	20.960715	19.203881	18.430825	25.118625	0.0
5	0.0	26.625000	16.375000	16.875000	19.000000	0.000	19.000000	16.875000	16.375000	26.625000	0.0
6	0.0	25.118625	18.430825	19.203881	20.960715	8.250	20.960715	19.203881	18.430825	25.118625	0.0
7	0.0	20.345646	18.889688	21.222409	23.930237	16.875	23.930237	21.222409	18.889688	20.345646	0.0
8	0.0	14.439800	18.253852	23.436083	28.616675	28.125	28.616675	23.436083	18.253852	14.439800	0.0
9	0.0	7.986316	13.479730	19.977526	26.717562	29.375	26.717562	19.977526	13.479730	7.986316	0.0
10	0.0	0.000000	0.000000	0.000000	0.000000	0.000	0.000000	0.000000	0.000000	0.000000	0.0

Applying Thresholding: Finally, we apply a threshold to the gradient magnitude image to detect edges. Any pixel with a gradient magnitude above a certain threshold value (let's say 17 in this case) is considered an edge pixel, and all others are suppressed. This gives us the final edge-detected image.

Edge Image:

Column	0	1	2	3	4	5	6	7	8	9	10
Row											
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	255	255	255	255	255	0	0	0
2	0	0	255	255	255	255	255	255	255	0	0
3	0	255	255	255	255	0	255	255	255	255	0
4	0	255	255	255	255	0	255	255	255	255	0
5	0	255	0	0	255	0	255	0	0	255	0
6	0	255	255	255	255	0	255	255	255	255	0
7	0	255	255	255	255	0	255	255	255	255	0
8	0	0	255	255	255	255	255	255	255	0	0
9	0	0	0	255	255	255	255	255	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0

Due to the high volume of calculations, we obtained the answers and results of each stage using the implemented code.

Lets calculate these steps for two pixels :

For pixel (2, 3) and (2, 4) we consider they neighbors

pixel (2, 3) =

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 4 & 5 \end{bmatrix}$$

pixel (2, 4) =

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 4 & 5 & 7 \end{bmatrix}$$

Now we calculate gradients using sobel operation:

For pixel (2, 3) we have:

$G_x = 6 = (2 * 1) + (-1 * 1) + (1 * 5)$

$G_y = (-1 * 1) + (5 * 1) = 4$

For pixel (2, 4) we do the same:

$G_x = 6$

$G_y = -20$

After this we should compute the gradient magnitude for both:

G (2, 3) = 7.21

G (2, 4) = 20.88

Now we will apply our thresholding which is 17

So we can assume that pixel (2, 3) is not a edge but pixel (2, 4) is a strong edge we can see this from the final results in the previous page and set the pixels which are edges to 255 and other pixel value which is under threshold is set to 0 (black)