Homework 2

“Computer vision”

professor : Dr. Mahmoudi Aznaveh

Spring 2024

By Katayoun Kobraei (99222084)

Theory questions:

1. Each kernel is a convolutional filter used in edge detection, each emphasizing different directions for detecting edges in an image. Now we check each kernel separately:

   a)

   $$\begin{matrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{matrix}$$

   This kernel detects edges that are oriented vertically. It is similar to the Sobel or Prewitt operator but simplified to consider vertical changes. So it identifies vertical edges with a positive gradient from bottom to top.

   b)

   $$\begin{matrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{matrix}$$

   This kernel detects edges that are oriented horizontally. It emphasizes changes in pixel values along the horizontal direction. So it identifies horizontal edges with a positive gradient (from left to right)

   c)

   $$\begin{matrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{matrix}$$

   This kernel detects edges oriented diagonally. Specifically the edges that change diagonally from the bottom-left to the top-right.

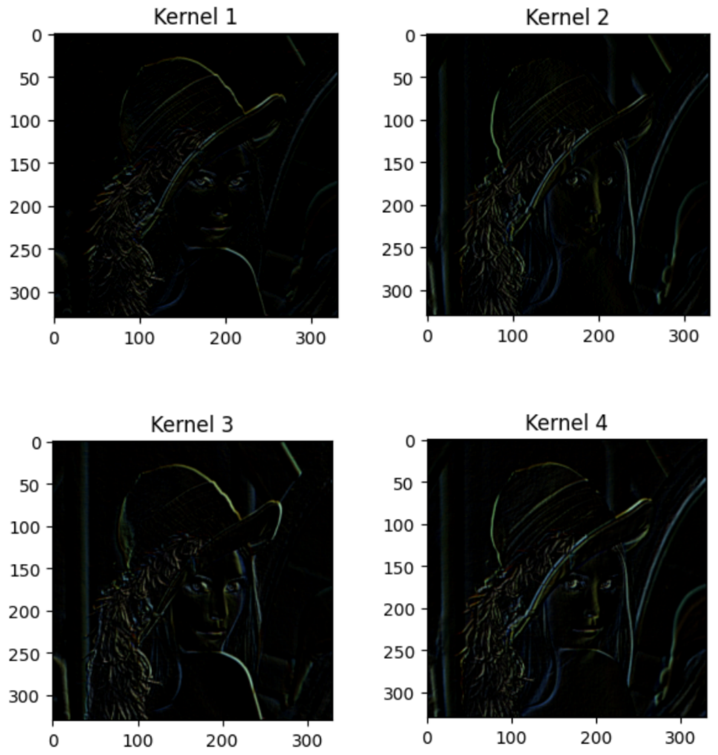
d)

-1	0	0
0	0	0
0	0	1

This kernel detects edges oriented diagonally in the opposite direction of Kernel 3. It focuses on changes that occur diagonally from the top-left to the bottom-right.

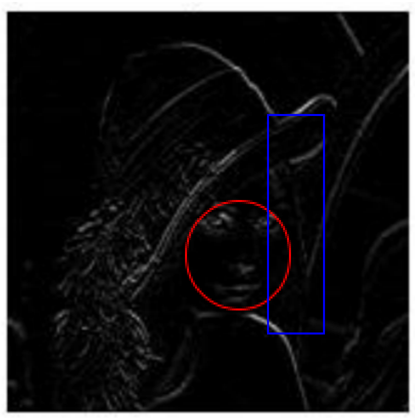
Each kernel is sensitive to edges where pixel intensity increases in a specific direction. For instance, Kernel 1 detects positive vertical gradients but Kernel 2 detects positive horizontal gradients.

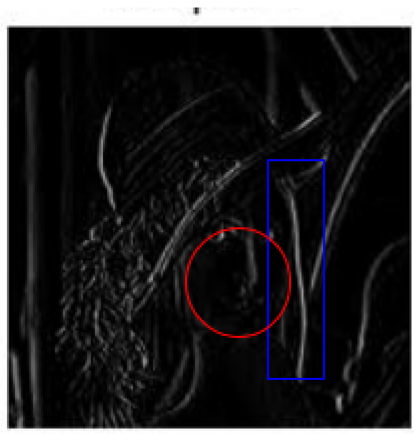
This is the image we run myself for kernels:



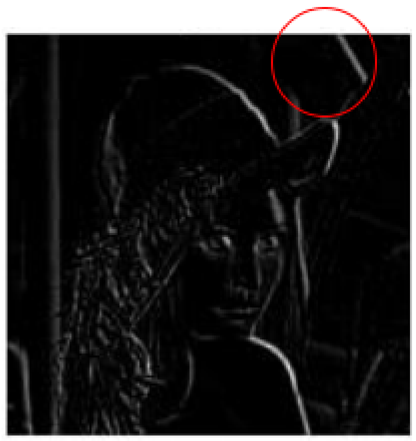
And this is the image from the question:

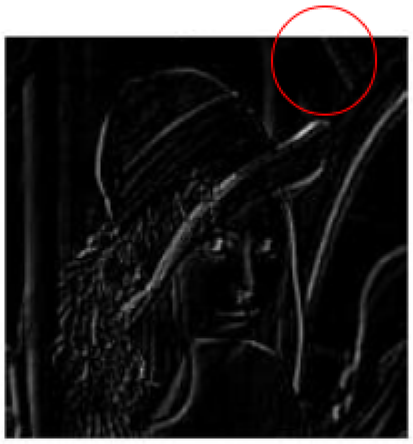
This is for kernel 1 cause we see just vertical edges and this is for kernel 2 for horizontal edges. We can see the difference in hair line and nose line

a. 

b. 

And this is for kernel 3 and 4:

C. 

D. 

2. The Laplacian filter has disadvantages compared to Sobel and Canny filters. These are three reasons why Sobel and Canny filters are better:

    1. We know that Laplacian filter is a second-order derivative operator. So this makes this filter very sensitive to noises. Any small fluctuations in pixel intensity produce significant gradients and this causes noisy edge maps. This high sensitivity makes it difficult to distinguish actual edges from noise so we will have much more false positives. On the other side Sobel filter combines smoothing with differentiation that helps noise reduction. It uses a convolution mask that effectively smooths the image before detecting edges. The Canny filter includes steps for noise reduction with Gaussian smoothing too. It uses both gradient magnitude and direction to detect edges. It also applies non-maximum suppression to remove spurious responses to edge detection, further reducing the impact of noise.

    2. As we know Laplacian filter directly computes second-order derivatives. This leads to potential issues with edge localization. The edges would not be as well-localized or as thin as those detected by more sophisticated methods. This can result in broader or less accurate edge representations. But the Sobel filter is a first-order derivative operator. So it includes a

smoothing component that helps in better edge localization. The Canny filter has a multi-step process that includes Gaussian smoothing, gradient calculation, non-maximum suppression, and edge tracking by hysteresis. These make edges well-localized and represented more accurate with minimal noise and better connectivity of edges. A multi-step approach for noise reduction and improving accuracy of edges make them more effective and widely used for edge detection in practical applications.

3. Laplacian filter often detects more edges than necessary. Because it responds to changes in intensity in all directions uniformly. This can lead to over-detection which even minor variations in intensity are marked as edges. But both Sobel and Canny filters are more discriminative. Sobel's gradient-based approach and Canny's thresholding (both low and high) help in distinguishing significant edges from minor intensity variations. So they will not have the same issue.

3. **A.** The Harris Corner Detector algorithm is used for identifying points of interest within an image which are corners. Corners are points that the intensity changes significantly in multiple directions. Here we explain the process step by step:

   a. First we compute the gradients of the image in the x and y directions. These gradients represent the change in intensity in each direction and are typically denoted as I_x and I_y.
   b. Then we calculate the structure tensor (also known as the second-moment matrix) M for each pixel in the image. M encapsulates the gradient information in a local neighborhood around the pixel. The structure tensor is a 2×2 matrix defined as:

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

where $w(x,y)$ is a window function that weights the pixels in the local neighborhood.

c. In the next step we define an energy function $E(u,v)$ that measures the change in intensity for a small shift (u,v) within the window:

$$E(u, v) = [u \quad v]\, M \begin{bmatrix} u \\ v \end{bmatrix}$$

$$\Rightarrow \quad M = [u \quad v] \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

d. Then we determine whether a pixel is a corner by analyzing the eigenvalues λ_1 and λ_2 of the matrix M. These eigenvalues represent the intensity changes in orthogonal directions within the window.

- If both eigenvalues are large then the pixel is a corner.
- If one eigenvalue is large and the other is small so the pixel is an edge.
- If both eigenvalues are small, it means the pixel is part of a flat region.

We searched and saw instead of directly computing the eigenvalues the original authors proposed using a corner response function R defined as:

$$R = det(M) - k\,.\, trace((m))^2$$

where det(M)=$\lambda_1 \lambda_2$ and trace(M)=$\lambda_1 + \lambda_2$. The parameter k is a constant typically set between 0.04 and 0.06. det(M) measures the product of the eigenvalues. trace(M) measures the sum of the eigenvalues.

B. Harris corner detection calculates the matrix M. From this matrix M, we derive its eigenvalues and eigenvectors.
Similar to PCA, where we calculate the covariance of data in space and understand the direction of maximum data variation from the covariance matrix, we then perform SVD to get the sigma values. The matrix M in Harris corner detection gives us the absolute values of our eigenvalues.

The eigenvalues indicate the direction of maximum variation. If one eigenvalue is large and the other is small, it indicates an edge. If both eigenvalues are small, it indicates noise. If both eigenvalues are large, it indicates a corner.

The eigenvalues of the matrix M (the structure tensor) have a direct relationship with the image gradients I_x and I_y.

This is the structure tensor M:

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

The eigenvalues λ_1 and λ_2 of the matrix M provide crucial information about the local image structure based on the gradients I_x and I_y :
We found eigenvalues by solving this equation:

$$det(M - \lambda I) = 0$$

$$\lambda_1, \lambda_2 = \frac{trace(M) \pm \sqrt{trace(m)^2 - 4.det(M)}}{2}$$

Where $trace(M) = I_x^2 + I_y^2$ and $det(M) = I_x^2 . I_y^2 - (I_x I_Y)^2$

If both λ_1 and λ_2 are large so this indicates significant intensity changes in both directions, suggesting a corner.

If one eigenvalue is large the other is small indicates an edge, as there is significant change in intensity in one direction but not the other.

If both eigenvalues are small it means a flat region with little to no change in intensity in any direction.

The eigenvalues reflect the magnitude of intensity changes. Since I_x and I_y are the gradients, their squares (along with the product) contribute directly to the entries of M and thus determine the eigenvalues. The eigenvalues can also hint at the direction of the intensity change. While I_x and I_y provide the directional gradients, the eigenvalues provide a measure of how strong these gradients are in orthogonal directions.

C. This is the 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**

First we compute the Harris corner response measure $E(u,v)$ for the given image to determine whether a pixel represents an edge, a corner, or a flat region.

We will start by getting the derivative in terms of X using this differentiation kernel: [-1 0 1]

now for the first pixel we will have:

[-1 0 1] * [1 0 5] $\rightarrow$ -1 + 5 = 4
[-1 0 1] * [0 5 7] $\rightarrow$ 7 = 7
[-1 0 1] * [5 7 11] $\rightarrow$ -5 + 11 = 6

And similar to that we calculate all values:

$$I_x : \begin{array}{ccccc} x & x & x & x & x \\ x & 4 & 7 & 6 & x \\ x & 8 & 8 & 7 & x \\ x & 8 & 6 & 5 & x \\ x & x & x & x & x \end{array}$$

Now we will start by getting the derivative in terms of Y using this differentiation kernel:

$$\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix} \rightarrow 4 = 4$$

And similar to that we will compute other pixels.

$$I_y : \begin{matrix} x & x & x & x & x \\ x & 4 & 8 & 8 & x \\ x & 8 & 6 & 7 & x \\ x & 6 & 6 & 4 & x \\ x & x & x & x & x \end{matrix}$$

After getting the X derivative and the Y derivative now we can get the Harris Matrix:

$$4^2 + 7^2 + 6^2 + 8^2 + 8^2 + 7^2 + 8^2 + 6^2 + 5^2 = 403$$

$$4^2 + 8^2 + 8^2 + 8^2 + 6^2 + 7^2 + 6^2 + 6^2 + 4^2 = 381$$

$$4 * 4 + 8 * 7 + 8 * 6 + 8 * 8 + 6 * 8 + 7 * 7 + 6 * 8 + 6 * 6 + 5 * 4 = 385$$

$$H = \Sigma_{(x,y)} w(x, y) . \begin{bmatrix} 403 & 385 \\ 385 & 381 \end{bmatrix}$$

$$R = det(H) - k * trace(h)^2 = 5318 - 0.04 * (784)^2 =- 19268.24$$

If R is large → corner
If R is negative → edge
If R is small → flat

4. To apply the Canny edge detector on the given image, follow these steps:

a. First we smooth the image using the given filter.
b. Secondly we calculate gradients using the Sobel operator to Find the edges.
c. Then we compute gradient magnitude and direction.
d. Next step we apply non-maximum suppression.
e. Finally we apply hysteresis thresholding.

Let's apply the filter to each pixel in the image, ignoring borders for simplicity:

Pixel (2,3):

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

Pixel (2,4):

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

Then we should apply the filtering process…
In the next step we calculate gradients using Sobel operator.

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

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

For each pixel we have:

$G_x = I * G_x$
$G_y = I * G_y$

For pixel (2,3) :

$G_x = 2 * 1 + (-1) * 1 + 1 * 5 = 6$
$G_y = (-1) * 1 + 5 * 1 = 4$

For pixel (2,4) :

$G_x = 4$
$G_y =- 20$

So we compute Gradient Magnitude and Direction and Hysteresis Thresholding:

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

$\theta = arctan(\frac{G_x}{G_y})$

For pixel (2,3) :

$G = 7.21$

For pixel (2,4) :

$G = 20.88$

Then we should do non maximum suppression:

For pixel (2, 3):
We compare pixel (2, 3) with interpolated pixels in the gradient direction and its perpendicular.

For pixel (2, 4):
Similarly, we compare with interpolated neighbors.

Next we choose a high and a low threshold for edge tracking. We assume:

High threshold = 15
Low threshold = 5

Then we consider thresholds:

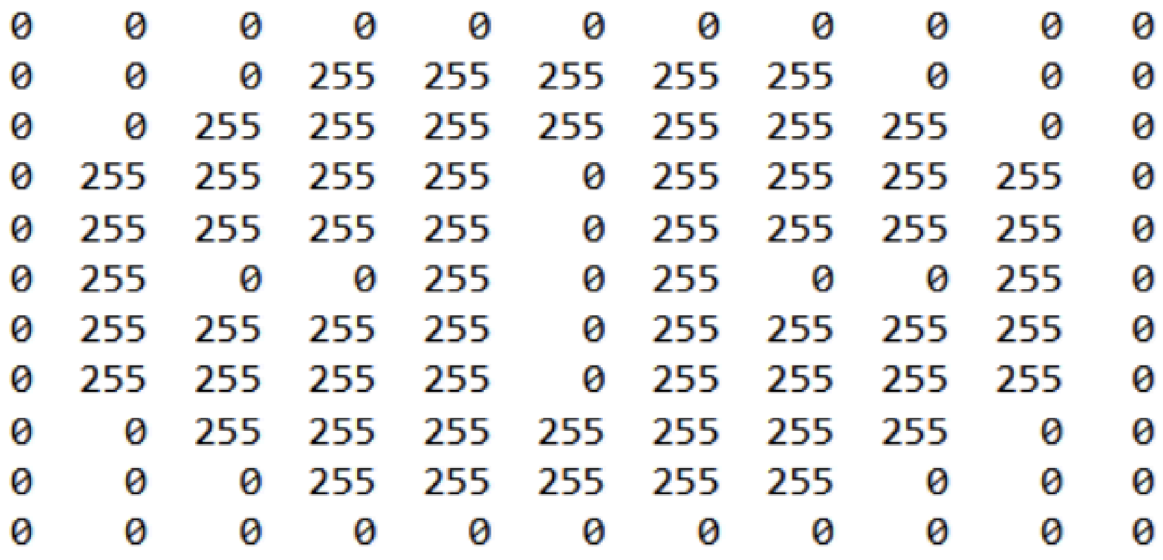
- Keep strong edges $G > T_{high}$
- Link weak edges $T_{low} < G < T_{high}$ if connected to strong edges.

So for (2, 3) and (2, 4), we found:

(2,3) has $G \simeq 7\ weak\ edge\ (between\ T_{high}\ and\ T_{low})$
(2,4) has $G \simeq 20.4\ strong\ edge\ (above\ T_{high})$

Finally we do Edge tracking by hysteresis to find best decisions.

For a complete image, we repeat this for every pixel, adjusting for boundaries and ensuring to link weak edges properly. So this would be the completed image:



```
0    0    0    0    0    0    0    0    0    0    0
0    0    0  255  255  255  255  255    0    0    0
0    0  255  255  255  255  255  255  255    0    0
0  255  255  255  255    0  255  255  255  255    0
0  255  255  255  255    0  255  255  255  255    0
0  255    0    0  255    0  255    0    0  255    0
0  255  255  255  255    0  255  255  255  255    0
0  255  255  255  255    0  255  255  255  255    0
0    0  255  255  255  255  255  255  255    0    0
0    0    0  255  255  255  255  255    0    0    0
0    0    0    0    0    0    0    0    0    0    0
```