

МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА
ПОЛІТЕХНІКА»



Лабораторна робота №1
З дисципліни «Обробка зображень методами штучного інтелекту»

Виконав:
студент групи КН-408
Черещук Любомир

Викладач:
Пелешко Д. Д.

Львів – 2022

Тема роботи: Попередня обробка зображень.

Мета роботи: Вивчити просторову фільтрацію зображень, методи мінімізації шуму, морфології, виділення країв і границь та елементи бібліотеки OpenCV для розв'язання цих завдань.

Теоретичні відомості.

У світі комп'ютерного зору фільтрація зображень використовується для модифікації зображень на етапі попереднього опрацювання. Ці зміни, по суті, дозволяють прояснити зображення, щоб отримати потрібну інформацію. Фільтрація може включати в себе все, що завгодно - видобуток країв з зображення, його розмиття, видалення небажаних об'єктів тощо.

Існує багато причин для використання фільтрації зображень. Наприклад, зйомка при сонячному свіtlі або в темряві вплине на чіткість зображення, тому можливо необхідно використовувати фільтри зображень, щоб змінити зображення згідно з власних потреб. Аналогічно, зображення може бути розмитим або зашумленим, яке потребує уточнення і фокусування.

Є такі методи фільтрації зображення – лінійна та нелінійна фільтрація.

До лінійної фільтрації зображення належать:

- 1D лінійна фільтрація зображення.
- 2D лінійна фільтрація зображення.
- Box фільтрація.

До нелінійної фільтрації зображення належать:

- Фільтр Гауса.
- Метод вирівнювання гістограми.
- Медіанна фільтрація зображення.

Хід роботи

Варіант 2 (номер в списку групи – 32). Вибрати з інтернету два зображення з різною деталізацією об'єктів та два зображення з різним контрастом. Без використання жодних бібліотек для обробки зображень (наприклад Open CV), виконати відповідне завдання (номер завдання вказано у рейтинговій таблиці).

Завдання: Виконати 2D лінійну фільтрацію зображення з різними значеннями ядра. Провести порівняльний аналіз.

2D лінійну фільтрація можна зобразити наступним чином:

0	0	0	0	0	0	0	0
0	60	113	56	139	85	0	0
0	73	121	54	84	128	0	0
0	131	99	70	129	127	0	0
0	80	57	115	69	134	0	0
0	104	126	123	95	130	0	0
0	0	0	0	0	0	0	0

Kernel		
0	-1	0
-1	5	-1
0	-1	0

114					

Код програми:

```
# Виконати 2D лінійну фільтрацію зображення з різними значеннями ядра. Провести
# порівняльний аналіз

import matplotlib.pyplot as plt
import matplotlib.image as Image
import numpy as np

def plot(img1, img2, image_type, kernel_type):
    _, ax = plt.subplots(1, 2, figsize=(12, 6))
    ax[0].imshow(img1, cmap='gray')
    ax[0].set_title('Original ' + image_type)
    ax[0].axis('off')

    ax[1].imshow(img2, cmap='gray')
    ax[1].set_title('With ' + kernel_type)
    ax[1].axis('off')
```

```

plt.show()

def img_to_gray(rgb):
    return np.dot(rgb[...,:3], [0.2989, 0.5870, 0.1140])

def add_padding(img, padding_size):
    img_with_padding = np.zeros(shape=(
        img.shape[0] + padding_size * 2,
        img.shape[1] + padding_size * 2
    ))
    img_with_padding[padding_size:-padding_size,
                     padding_size:-padding_size] = img
    return img_with_padding

def get_output_img_size(img_size, kernel_size):
    output_img_size = 0

    for i in range(img_size):
        count = i + kernel_size
        if count <= img_size:
            output_img_size += 1

    return output_img_size

def apply_filter2D(image, kernel):
    kernel_size = kernel.shape[0]
    padding_size = kernel_size // 2
    img_with_padding = add_padding(image, padding_size)

    output_size = get_output_img_size(
        img_size=img_with_padding.shape[0],
        kernel_size=kernel_size
    )

    k = kernel_size
    convolved_img = np.zeros(shape=(output_size, output_size))

    for i in range(output_size):
        for j in range(output_size):
            mat = img_with_padding[i:i+k, j:j+k]
            convolved_img[i, j] = np.sum(np.multiply(mat, kernel))

    return convolved_img

```

```

# define kernels

# blur filter
blur_kernel = np.ones((10, 10), np.float32)/25

# edge detection
edge_kernel = np.array([
    [-1, -1, -1],
    [-1, 8, -1],
    [-1, -1, -1]
])

# Sobel filter
sobel_kernel = np.array([
    [-1, 0, 1],
    [-2, 0, 2],
    [-1, 0, 1]
])

kernels = [[blur_kernel, 'blur_kernel'], [
    edge_kernel, 'edge_kernel'], [sobel_kernel, 'sobel_kernel']]

images = ['high_contrast.jpeg', 'low_contrast.jpeg',
          'high_detailed.jpeg', 'low_detailed.jpeg']

for image_name in images:
    for kernel in kernels:
        # image
        image = np.array(img_to_gray(Image.imread(image_name)))
        # filtered img
        filtered_img = apply_filter2D(image=image, kernel=kernel[0])
        # plot result
        plot(img1=image, img2=filtered_img,
              image_type=image_name, kernel_type=kernel[1])

```

Результат роботи програми:



Рис. 1 Високо-контрастне зображення з ядром blur.

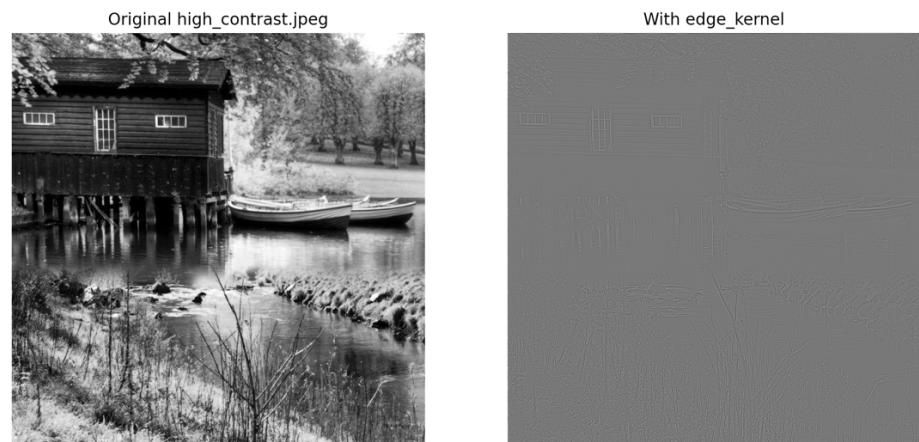


Рис. 2 Високо-контрастне зображення з ядром edge.

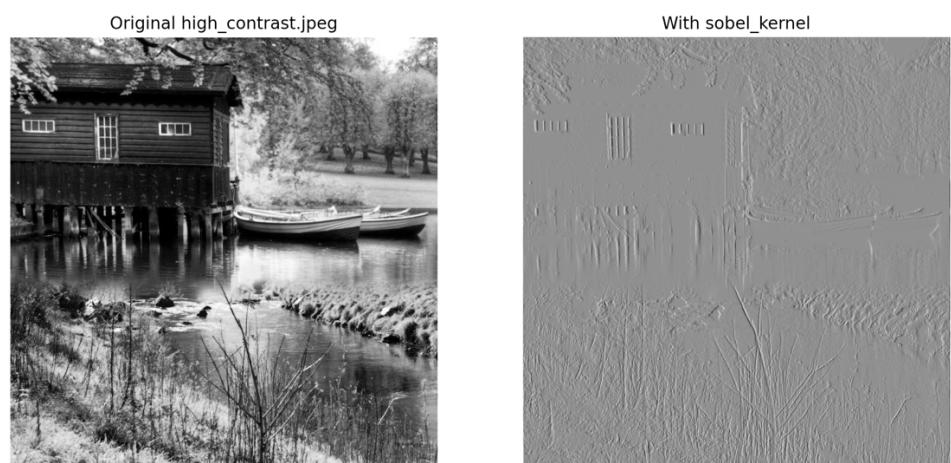


Рис. 3 Високо-контрастне зображення з ядром sobel.

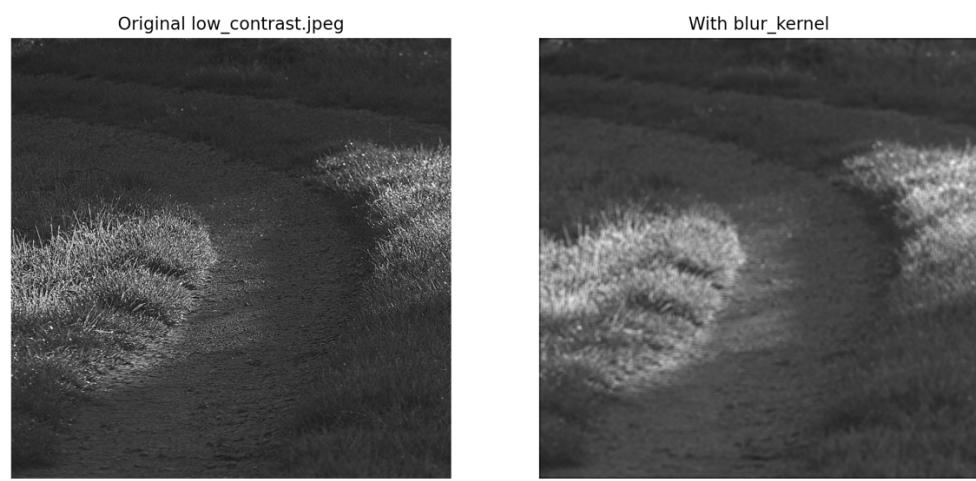


Рис. 4 Низько-контрастне зображення з ядром blur.

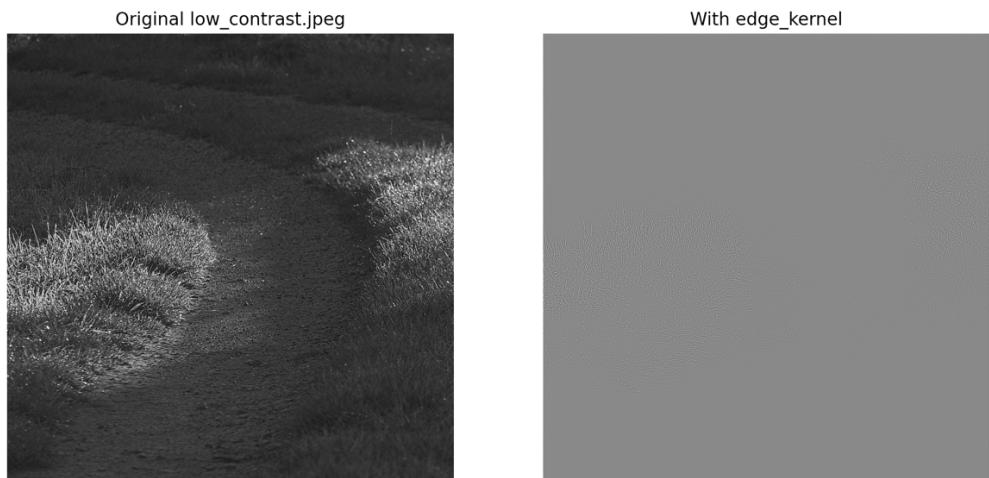


Рис. 5 Низько-контрастне зображення з ядром edge.

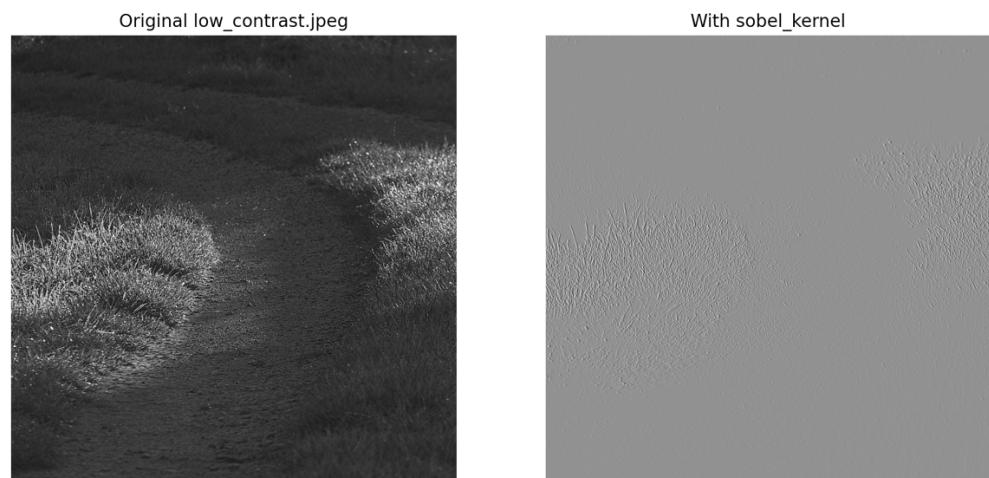


Рис. 6 Низько-контрастне зображення з ядром sobel.

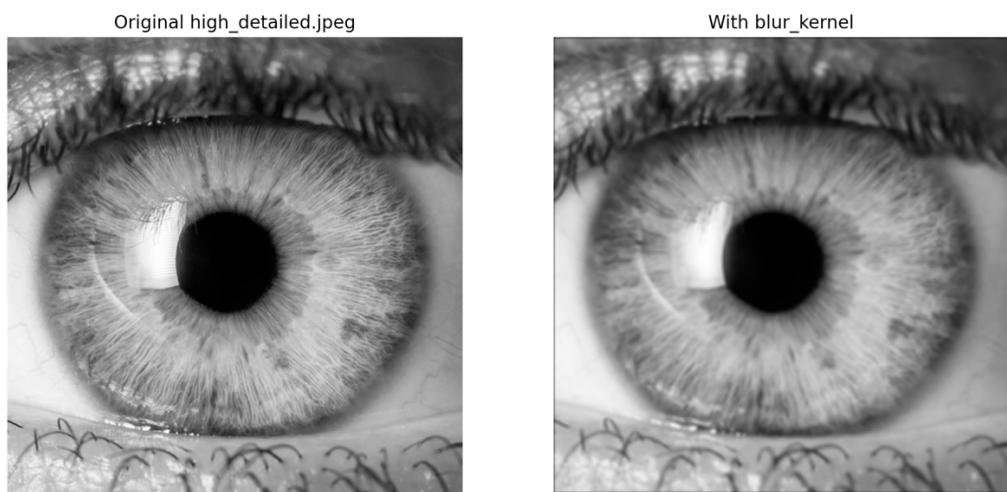


Рис. 7 Високо-детальне зображення з ядром blur.

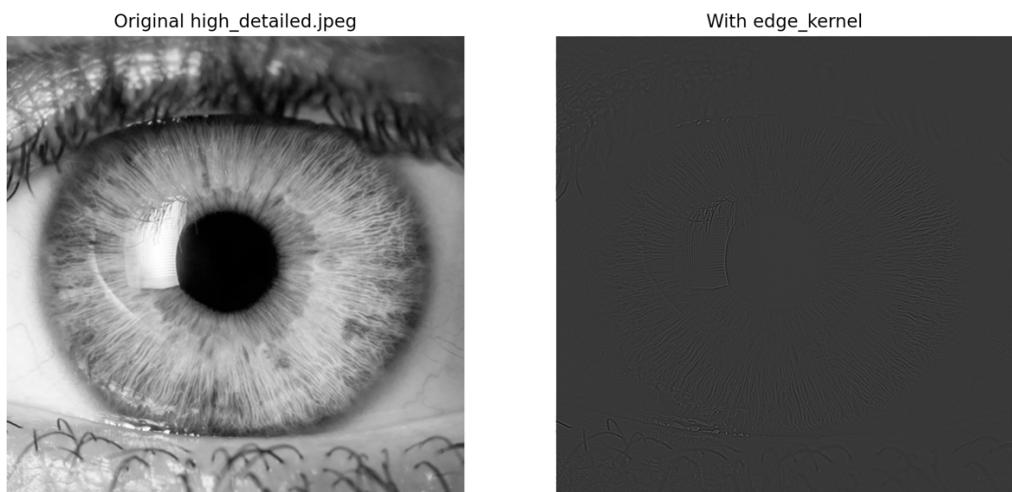


Рис. 8 Високо-детальне зображення з ядром edge.

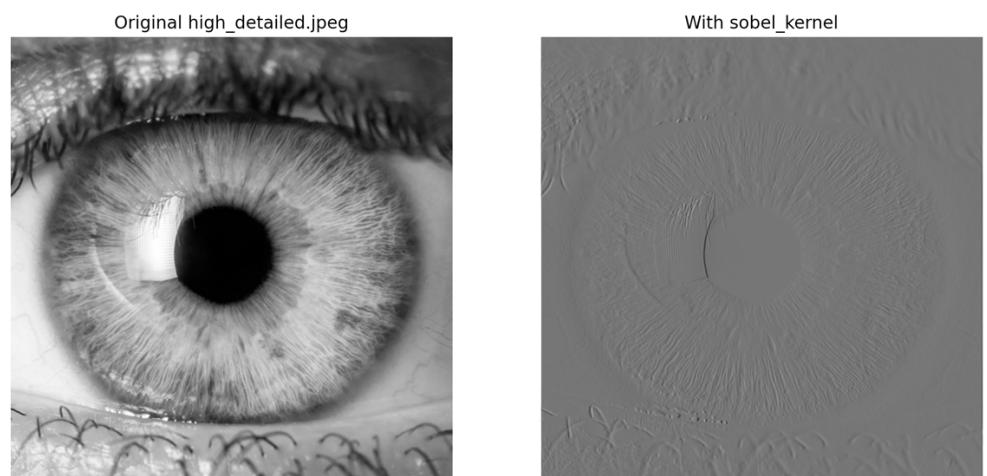


Рис. 9 Високо-детальне зображення з ядром sobel.

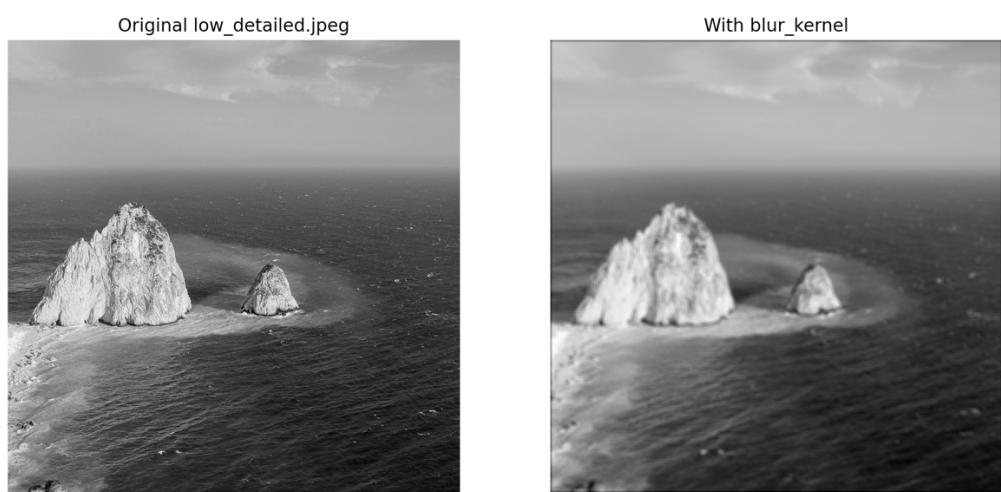


Рис. 10 Мало-детальне зображення з ядром blur.

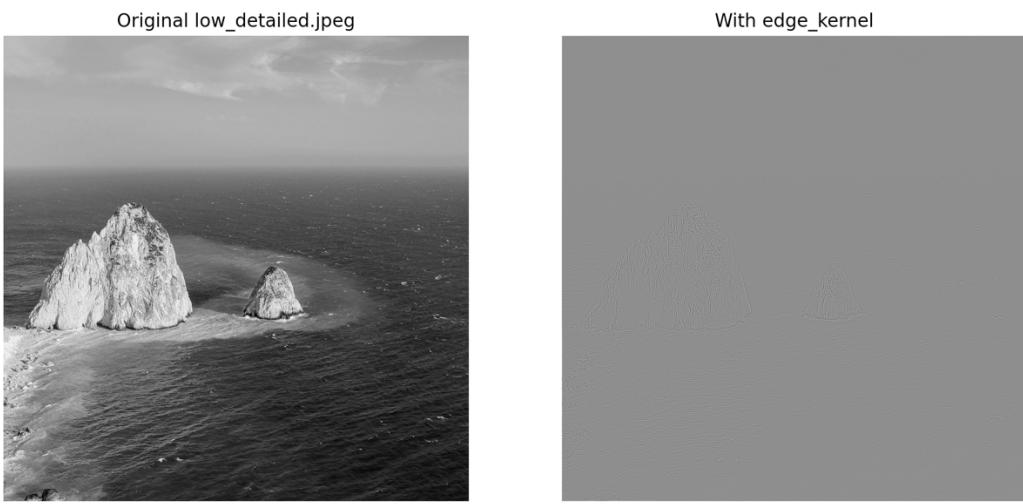


Рис. 11 Мало-детальне зображення з ядром edge.

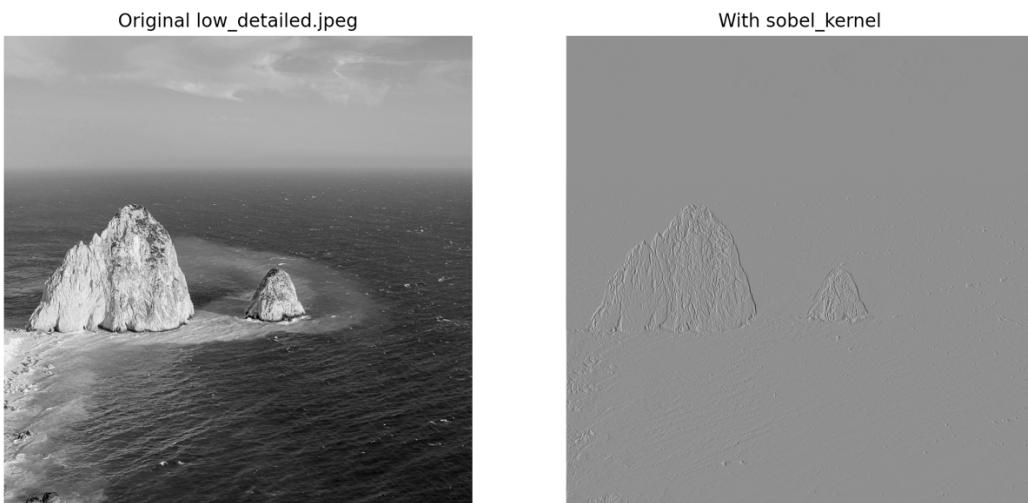


Рис. 12 Мало-детальне зображення з ядром sobel.

Висновок: виконавши дану роботу я вивчив просторову фільтрацію зображень, методи мінімізації шуму, морфології, виділення країв і границь та елементи бібліотеки OpenCV для розв'язання цих завдань.

Проглянувши результати можна сказати, що загалом 2D фільтрація з ядром типу blur однаково добре працює на кожному зображені. Ядро типу edge та sobel однаково добре працюють для виявлення границь(оскільки являються ядрами такого типу), особливо добре виявлення границь спрацювало на високо детальному зображені. Також ядро типу sobel

спрацювало трошки краще ніж edge на контрастних зображеннях та на мало детальному.