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 поперднього 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 світлі або в темряві вплине на чіткість зображення, тому можливо необхідновикористовувати фільтри зображень, щоб змінити зображення згідно власних потреб. Аналогічно, зображення може бути розмитим або зашумленим, яке потребувати 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:



**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:

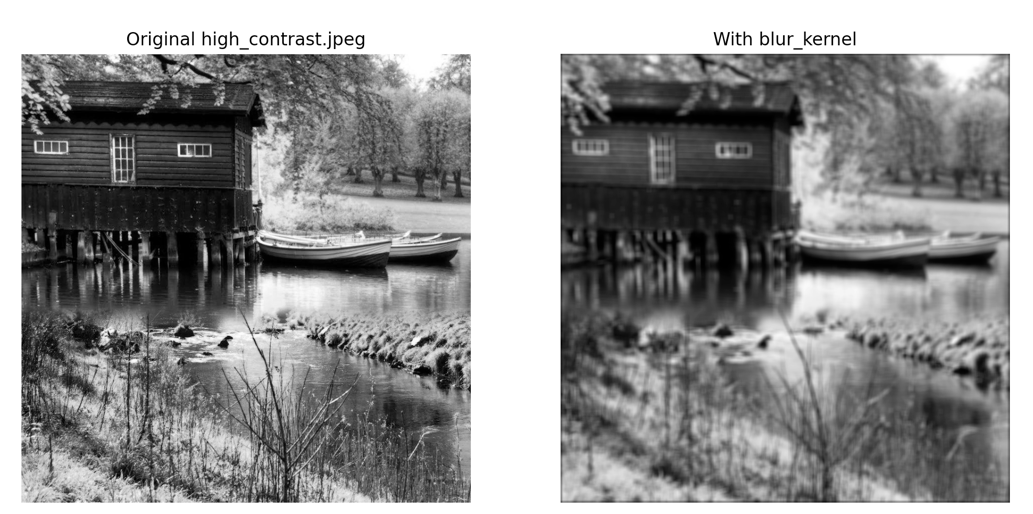


Рис. 1 Високо-контрастне зображення з ядром blur.

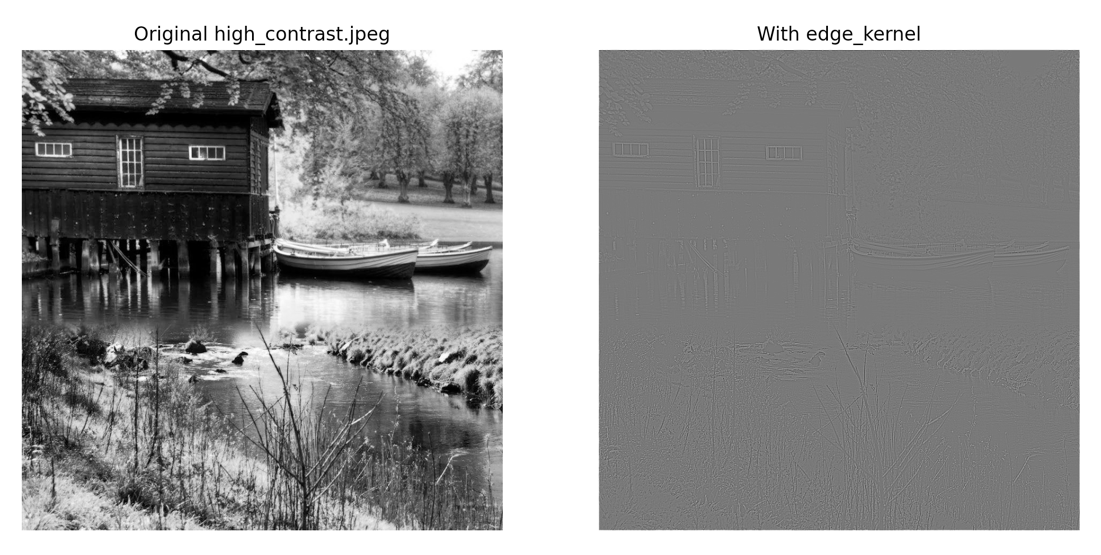


Рис. 2 Високо-контрастне зображення з ядром edge.

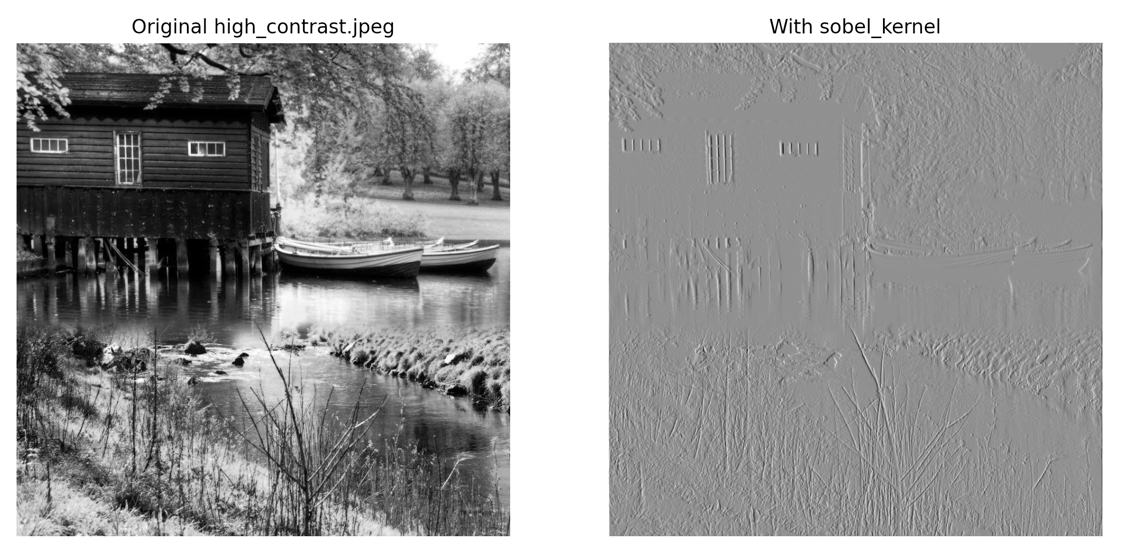


Рис. 3 Високо-контрастне зображення з ядром sobel.

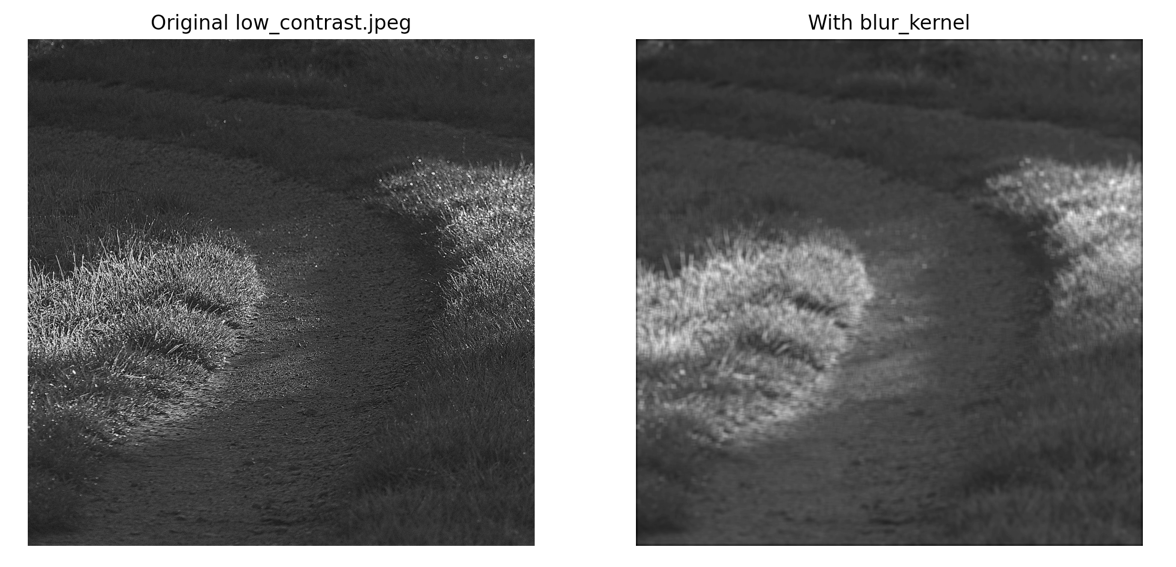


Рис. 4 Низько-контрастне зображення з ядром blur.

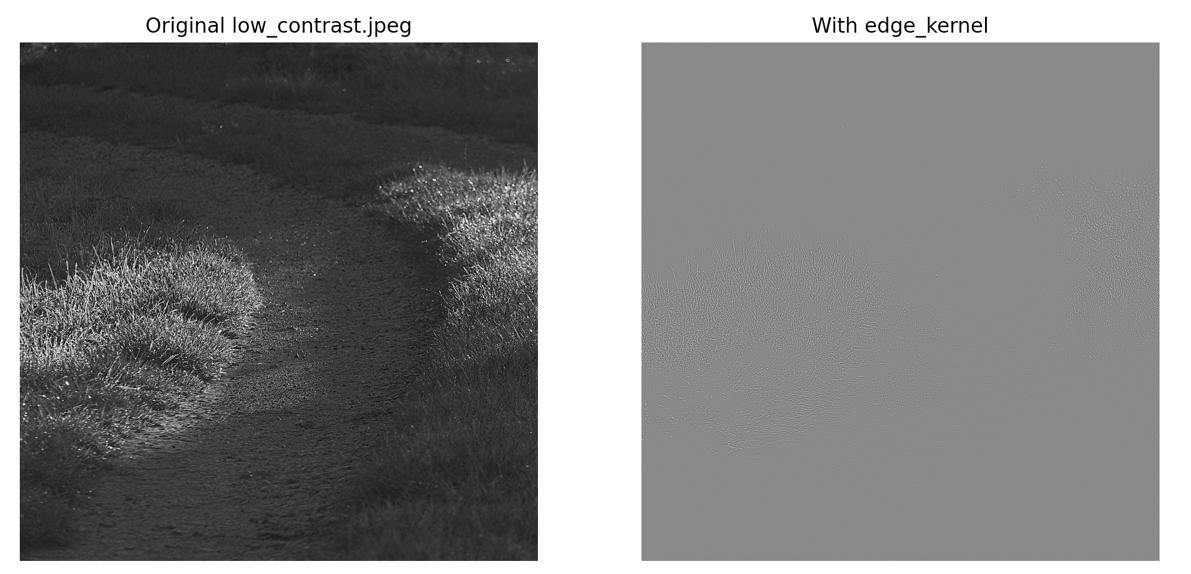


Рис. 5 Низько-контрастне зображення з ядром edge.

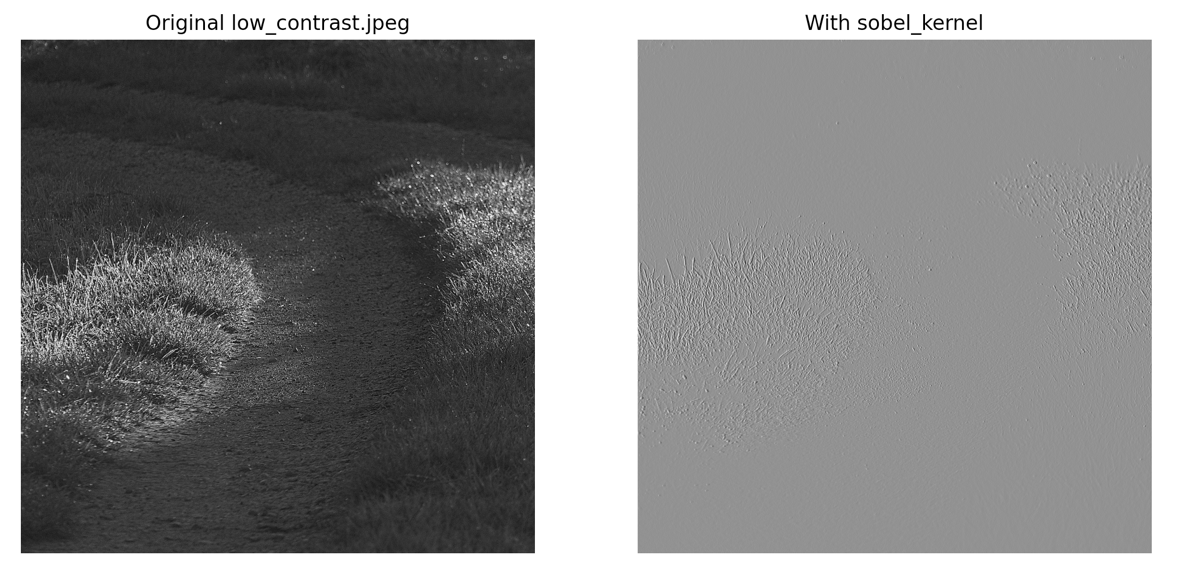


Рис. 6 Низько-контрастне зображення з ядром sobel.

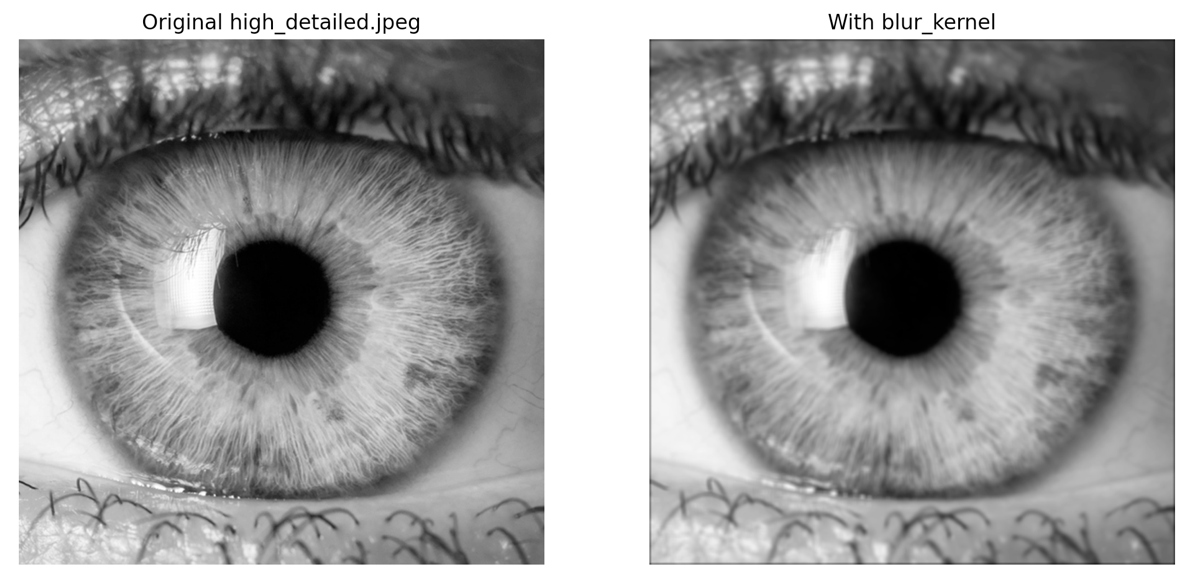


Рис. 7 Високо-детальне зображення з ядром blur.

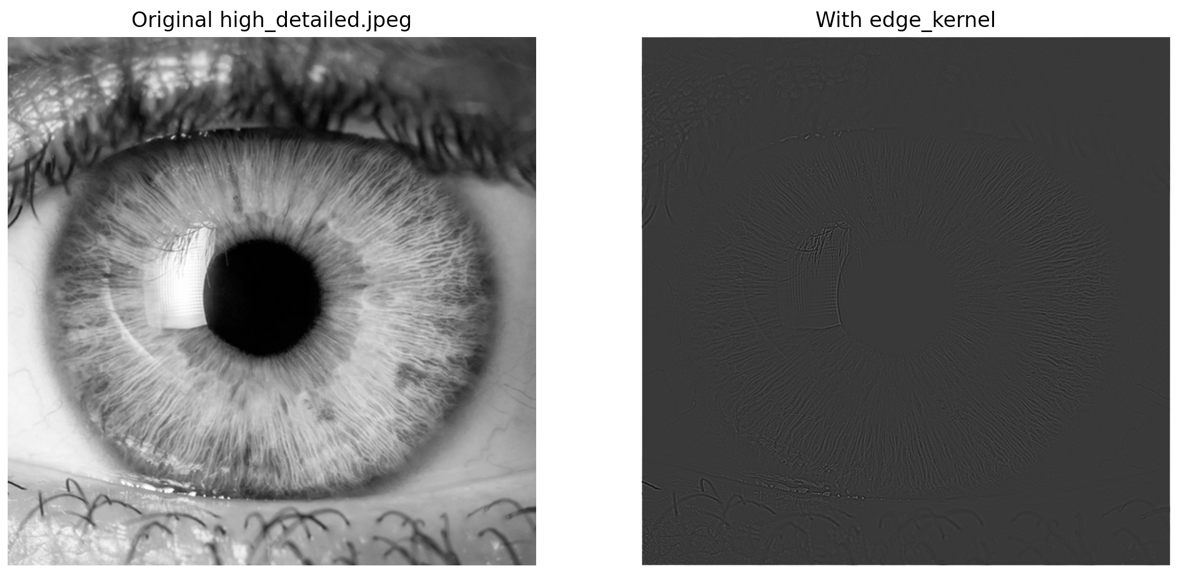


Рис. 8 Високо-детальне зображення з ядром edge.

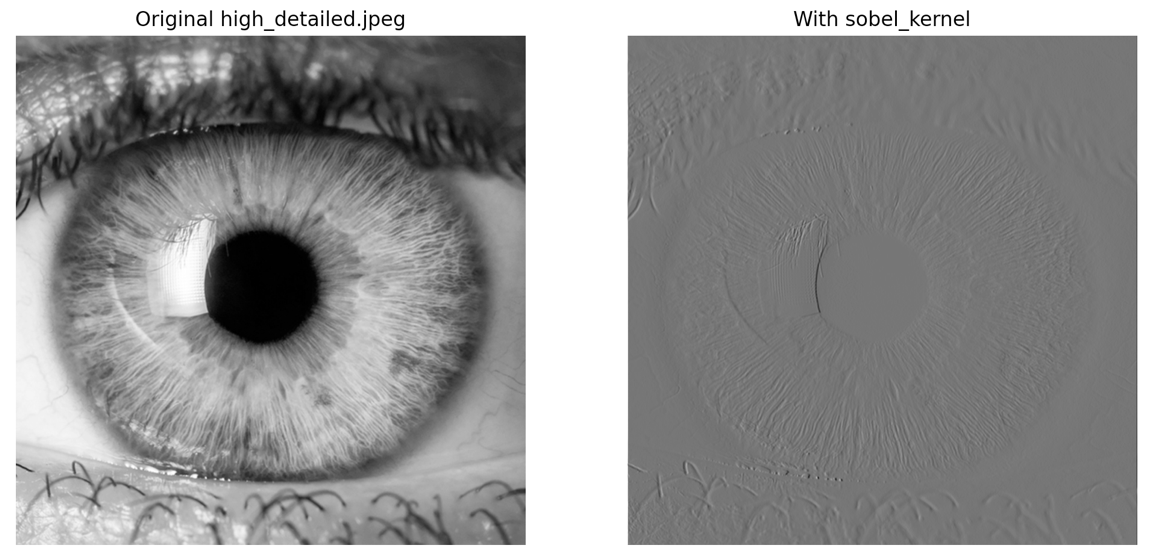


Рис. 9 Високо-детальне зображення з ядром sobel.

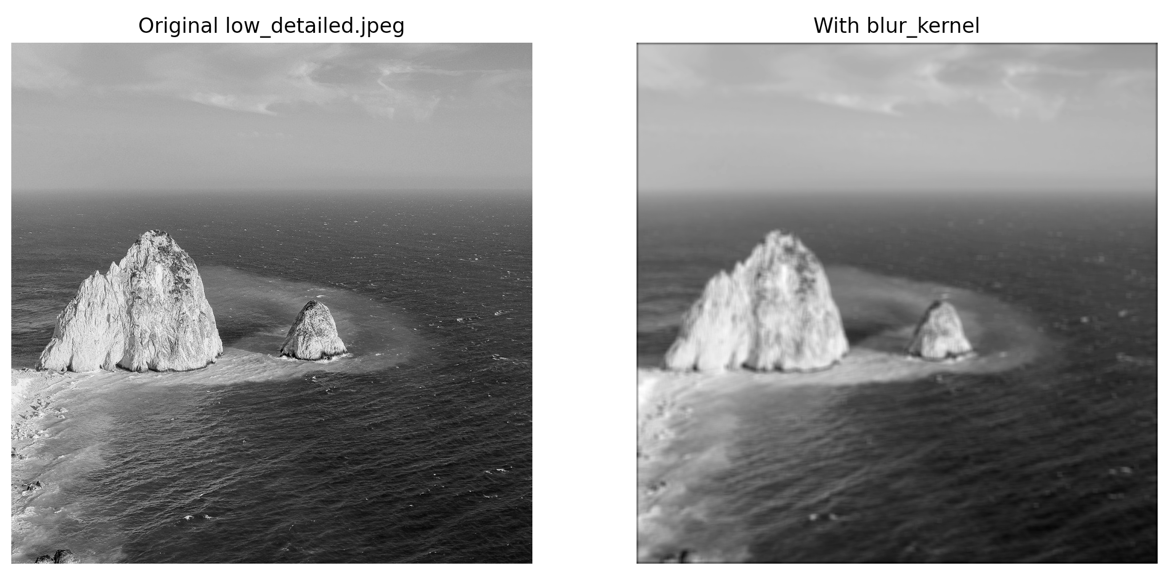


Рис. 10 Мало-детальне зображення з ядром blur.

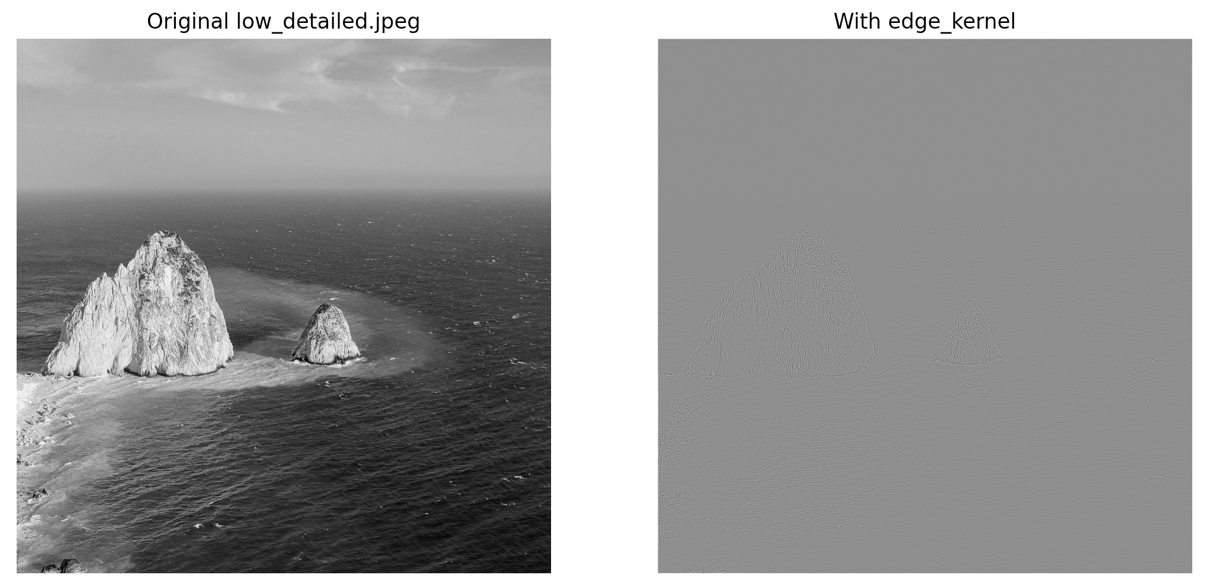


Рис. 11 Мало-детальне зображення з ядром edge.

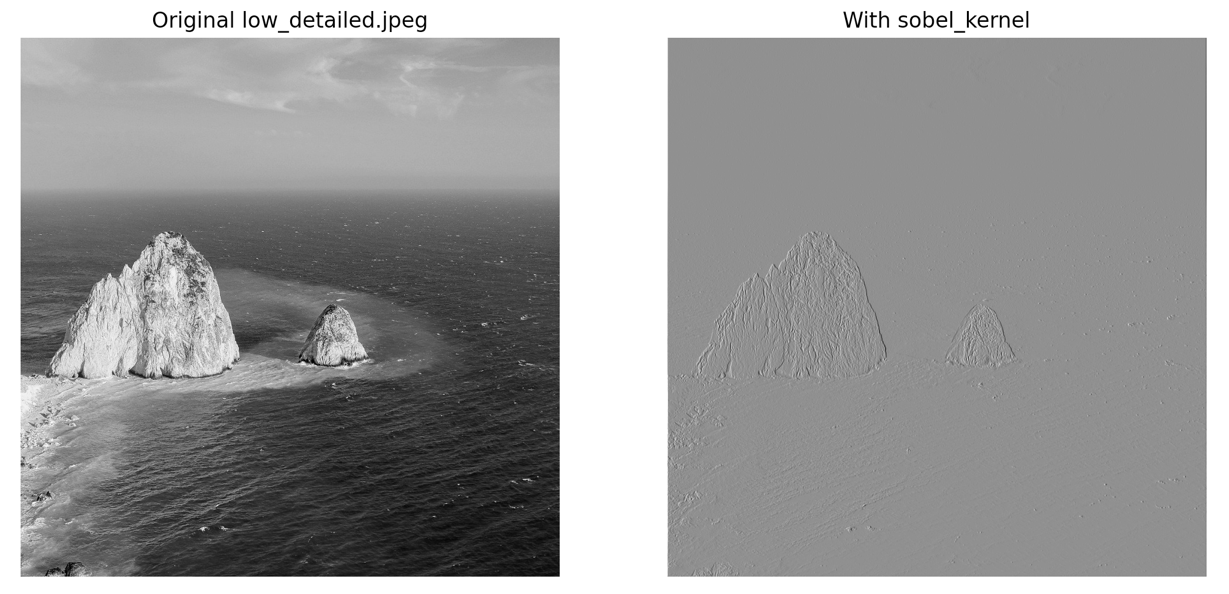


Рис. 12 Мало-детальне зображення з ядром sobel.

**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 зображенні. Також ядро типу sobel спрацювало трошки краще ніж edge на контрастних зображеннях та на мало детальному.