**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД**
**ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ**
**КАФЕДРА ПРИКЛАДНОЇ МАТЕМАТИКИ І ІНФОРМАТИКИ**

# МЕТОДИЧНІ ВКАЗІВКИ І ЗАВДАННЯ
## до ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ ЗА КУРСОМ «КОМП'ЮТЕРНА ОБРОБКА ЗОБРАЖЕНЬ»

для студентів спеціальності
122 Комп'ютерні науки
всіх форм навчання

**ЛУЦЬК**
2024

УДК 004.9(072)

М 54

Методичні вказівки і завдання до виконання лабораторних робіт за курсом «Комп'ютерна обробка зображень» для суд. спец. 122 Комп'ютерні науки всіх форм навчання [Електронний ресурс] / уклад. : Є.О. Башков, О.В. Александрова. –Луцьк : ДонНТУ, 2024. – 63 с.

Методичні вказівки і завдання до виконання лабораторних за курсом «Комп'ютерна обробка зображень» призначені для закріплення знань студентів, що навчаються за спеціальністю 122 Комп'ютерні науки галузі знань 12 Інформаційні технології, одержуваних при вивченні лекційного матеріалу.

Комплекс лабораторних робіт дисципліни «Сучасні методи обробки візуальної інформації» виконується відповідно робочої програми, передбачає послідовне виконання завдань по кожній лабораторної роботі та має на меті відпрацювання навичок і вмінь щодо розробки алгоритмів перетворень цифрових зображень, їх програмної реалізації та візуалізації результатів для декількох типів обробки: геометричні перетворення, лінійна та нелінійна фільтрація, морфологічні операції.

При виконанні лабораторних робіт необхідно застосовувати викладені в методичних вказівках теоретичні положення для вирішення поставлених задач. Кожному студенту необхідно виконати індивідуальне завдання, варіанти яких наведено в методичних вказівках. Таким чином, самостійна робота студентів дозволить розвити навички розв'язання практичних задач обробки зображень. За кожною темою наводяться основні теоретичні відомості, контрольні запитання і змістовні завдання, що дозволяють опанувати на практиці вивчений матеріал.

Укладачі: **Є.О. Башков**, професор кафедри ПМІ, д.т.н., професор;
**О.В. Александрова**, асистент кафедри ПМІ.

Рецензент: **Ковальов С.О**., доцент кафедри ЕТ, к.т.н., доцент.

Відповідальний за випуск: **Маслова Н.О**., в.о.завідувача кафедри ПМІ.

Затверджено навчально-методичним відділом ДонНТУ,
протокол № 7 від 25.04. 2023 року

Розглянуто на засіданні кафедри Прикладної математики і інформатики»
протокол № 4 від 18.04.2023 року

**ЗМІСТ**

# ВСТУП

Дисципліна «Сучасні методи обробки візуальної інформації» має на меті надання цілісного представлення щодо базових методів обробки цифрових зображень та підготовка майбутнього фахівця до розробки і застосування сучасних технологій комп'ютерної обробки зображень при організації та розробці програмного забезпечення комп'ютерів, комп'ютерних систем та мереж.

Наданий комплекс лабораторних робіт дисципліни «Сучасні методи обробки візуальної інформації» передбачає послідовне виконання завдань по кожній лабораторній роботі та має на меті відпрацювання навичок і вмінь щодо розробки алгоритмів перетворень цифрових зображень, їх програмної реалізації та візуалізації результатів для декількох типів обробки: геометричні перетворення, лінійна та нелінійна фільтрація, морфологічні операції.

Самостійна робота студентів передбачає підготовку до лабораторних робіт, ознайомлення з рекомендованою літературою з наданого переліку, а також підготовку та оформлення залікового звіту за результатами виконання лабораторних робіт.

Кожна лабораторних робота орієнтована на групу родинних алгоритмів для виконання визначеного типу обробки. Поряд з теоретичним матеріалом у кожному розділі наведені приклади фрагментів коду та результати їх виконання.

Для виконання лабораторних завдань рекомендовано використовувати мову **Python** з встановленими додатковими пакетами (***NumPy***, ***SciKit-Imge, Matplotlib***). Послання на відповідні ресурси наведені в переліку рекомендованих програмних продуктів.

***Остаточний вибір мови та середовища програмування залишається за студентом.***

# 1. ЦИФРОВЕ ЗОБРАЖЕННЯ ТА ЙОГО ПРЕДСТАВЛЕННЯ

## 1.1 Загальна інформація

Довільне цифрове зображення у растровій формі складається з неподільних елементів – пікселів (від англ. pixel – **pic**ture **el**ement – елемент зображення). Зазвичай зображення надається як прямокутна матриця **пікселів**, кожен з котрих зберігає числову інформацію щодо свого кольору. Матриця складається з ***M*** стовпчиків і ***N*** рядків. Стовпчики нумеруються зліва – направо від ***0*** до ***M-1***, а рядки нумеруються зверху до низу від ***0*** до ***N - 1***, відповідно. Початок системи координат в лівому верхньому куті (рис. 1).

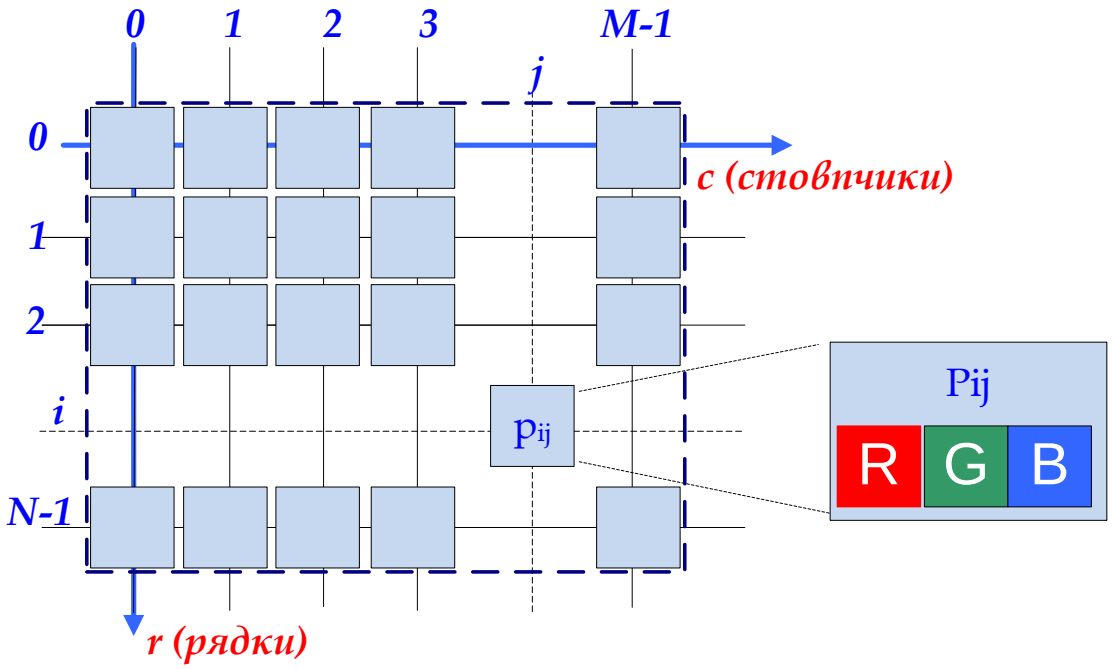



***Рисунок 1-1 Представлення цифрового зображення. Нумерація пікселів***

**Розмір** растрового цифрового зображення визначається парою чисел: кількість пікселів у стовпчику помножену на кількість пікселів в рядку **N x M**. Зазвичай загальний розмір зображення вказується в **МЕГАПІКСЕЛЯХ (Мр, Мп)**, при цьому **1Мп = 1000000 пікселів.**

Важливо розуміти, що **цифрове растрове зображення не має ніяких фізичних вимірів – це тільки набір чисел.** Тільки коли цифрове зображення відтворюєтеся за допомогою деякого пристрою «з'являються» реальні фізичні розміри.

Представлення кольорів в цифрових зображеннях базується на деякому **математичному** опису кольорів – так званій **колірній моделі**. Кольорова модель це представлення довільного кольору у вигляді набору **трійки** чисел, які визначають колірні координати (колірні компоненти). Тобто значення трьох колірних компонент однозначно визначають колір. В практиці роботи з цифровими зображеннями застосовується декілька кольорових моделей, конкретний тип якої залежить, в першу чергу, від пристрою, що використовується для відтворення зображення.

Найбільш поширеною в цифрових технологіях є **R G B** колірна модель, що реалізована в таких пристроях як монітори, цифрові проектори, сканери, цифрові фотоапарати. **R G B** це абревіатура англійських слів **R**ed, **G**reen, **B**lue — червоний, зелений, синій, відповідно. Типове кодування **R G B** кольорових компонент використовує три числа, що належать деякому діапазону, за правило від 0 до деякого максимального значення $2^N - 1$ (формат цілих чисел). Тут ***N*** – кількість біт двійкового представлення числа.

Зображення, що обробляються, з точки зору їх кольорового представлення, класифікуються наступним чином.

**Повнокольорове R G B зображення (True Color)** використовує всі можливі комбінації трьох кольорових компонент. Якщо кожна кольорова компонента представлена 8-біт двійковим цілим, тобто кожна кольорова компонента може приймати значення від 0 до $2^8 - 1 = 255$, то повнокольорове цифрове зображення може мати $2^{24} - 1 = 16\,777\,215$ різних кольорів.

**Монохромне зображення** - зображення, що містить світло одного конкретного кольору і сприймається як один кольоровий відтінок. Для **R G B** зображень це зображення, для якого дві компоненти мають у всіх пікселів мають **незмінні** значення, а третя компонента може змінюватися від 0 до максимального значення. Для ***N = 8*** монохромне зображення має 255 рівнів одного кольорового відтінку.

**Ахроматичне зображення (**безбарвне, сіре, **grayscale) -** зображення, що формується через змішування трьох кольорових компонент у рівних прапорцях. Тобто всі три кольорові компоненти зображення мають однакові значення: **R=G=B** для всіх пікселів зображення. Таким чином відтінок сірого може змінюватися від чорного (всі компоненти нульові) до білого (всі компоненти максимальні).

**Бінарне зображення (bitmp) -** зображення, у якому присутні тільки пікселі з двома різними комбінаціями **R G B** компонент. Зображення складається з пікселів двох кольорів.

**Чорно-біле зображення –** бінарне зображення, у якому присутні тільки пікселі з двома різними комбінаціями **R G B** компонент: **R = G = B = 0** (чорне) та **R = G = B = $2^N$-1** (біле).

Таким чином цифрове зображення ***I*** можна представити як матрицю

$$\boldsymbol{I} \Leftrightarrow \|I(i,j)\|, \qquad i = 1,2,\dots,N-1;\ j = 1,2,\dots,M-1,$$

де кожен елемент $I_{i,j}$ для кольорового зображення має три компоненти

$$I_{i,j} = \left[R_{i,j}\ ,\ G_{i,j}\ ,\ B_{i,j}\right],$$

а для ахроматичного зображення

$$I_{i,j} = \left[L_{i,j}\ ,\ L_{i,j}\ ,\ L_{i,j}\right].$$

## 1.2 Цифрові зображення у пакеті *SciKit-Imge*

Зображення у пакеті ***SciKit-Imge*** представлені масивами **ndarray** пакету ***NumPy***. Тобто усі загальні маніпуляції з зображенням можливо виконувати штатними засобами ***NumPy***.

Тобто, довільне кольорове зображення обробляється як 3-вимірний масив **ndarray** з наступною структурою:

- перший вимір - рядки зображення від 0 до ***N-1*** (з верху зображення до низу), де ***N*** - кількість рядків,

- другий вимір - колонки зображення від 0 до ***M-1*** (зліва направо), де ***M*** - кількість колонок,
- третій вимір - 3 кольори: **R**, **G**, **B** відповідно, задані у форматі **np.uint8** (байт на колір, кожна кольорова компонента змінюється в межах від 0 до 255 ) або **np.float32** (кожна кольорова компонента змінюється в межах від 0.0 до 1.0 ).

Приведемо типові прийоми роботи з зображенням.

### 1.2.1 Завантаження бібліотек

На початку програми потрібно завантажити пакети та модулі, що використовуються

```
import numpy as np
import skimage.io as io
import matplotlib.pyplot as plt
```

### 1.2.2 Завантаження файлу зображення

Для завантаження зображення з файлу рекомендуємо використовувати функцію **skimage.io.imread()** з пакету ***SciKit-Image***.

Наприклад, потрібно завантажити зображення з файлу **c:/test_image_in.jpg** (повне ім'я файлу) до змінної **test_image**

```
filename = 'c:/test_image_in.jpg'
test_image = io.imread(filename)
```

### 1.2.3 Зберігання зображення до файлу

Для зберігання зображення у файлі рекомендуємо використовувати функцію **skimage.io.imsave()** з пакету ***SciKit-Image***.

Наприклад, потрібно зберегти зображення представлене змінною **my_image** до файлу **c:/test_image_out.jpg** (повне ім'я файлу)

```
out_filename = 'c:/test_image_out.jpg'
io.imsave(out_filename,my_image_)
```

### 1.2.4 Вивід зображення на екран

Для виводу зображення на екран, рекомендуємо використовувати можливості пакету ***Matplotlib***.

Наприклад, потрібно вивести зображення представлене змінною **image_out** на екран дисплею.

```
plt.title('IMAGE')
plt.imshow(image_out)
```

Важлива примітка щодо використання ***Matplotlib.*** Функція **Matplotlib.imshow(image)** за замовчуванням відображає тільки три канальні (**R, G, B**) зображення. Тобто, всі кольори R, G, B в масиві **image** повинні бути представлені в одному з двох форматів:

- як масив елементів np.uint8 (кожний колір приймає значення від 0 до 255)
- як масив елементів np.float32 (кожний колір приймає значення від 0.0 до 1.0)

# 2. Лабораторна робота №1. Основи роботи з зображеннями. Найпростіші операції з зображеннями.

**Лабораторна робота №1 орієнтована на** оволодіння студентами базовими прийомами роботи із зображеннями та їх перетворень, що застосовуються у пакетах ***SciKit-Imge, Matplotlib, NumPy***.

**Мета лабораторної роботи:**

- Ознайомити студентів з базовими прийомами обробки зображень, що застосовуються у пакеті ***SciKit-Imge;***
- Оволодіти алгоритмами найпростішої обробки зображень за допомогою ***NumPy***;
- Придбати практичні навички у використанні команд пакетів ***SciKit-Imge*** та ***Matplotlib*** для зчитування, запису та відображення зображень;
- Розвинути у студентів вміння проводити самооцінку виконання лабораторної роботи №1.

У разі успішного виконання лабораторної роботи студент буде здатен обирати та використовувати програмні продукти для роботи із зображеннями та виконувати найпростіші операції обробки цифрових зображень.

Успішне засвоєння матеріалів та виконання лабораторної роботи буде сприяти формуванню у студента наступних соціальних навичок та вмінь: бажання постійно вчитися, вміння дотримуватися регламентних рамок, оцінювати строки проведення та виконання роботи, обґрунтувати той чи інший спосіб діяльності при виконанні практичних завдань. Форми прояву: прагнення вирішувати поточні проблеми самостійно, бажання пробувати щось нове, йти на розумний ризик при прийнятті рішень, застосовувати елементи самоконтролю і самооцінки при виконанні роботи, вміння планувати свій час.

## 2.1 Завдання до лабораторної роботи №1

В процесі виконання лабораторної роботи студент повинен створити програмний додаток, що виконує наступні дії:

1. Зчитує оригінальне зображення у відповідності за варіантом завдання (номер наведено в таблиці 1). Виводить на екран дисплею завантажене зображення. **Примітка:** Ім'я зображення за номером та місце збереження файлу наведені в достатку 1.

2. Визначає та виводить структуру та розмір оригінального зображення. Виводить значення кольорових компонент заданого [***i, j***] пікселю.

3. Формує **монохромне** зображення з відповідного каналу оригінального зображення. Виводить отримане зображення на екран дисплею. Зберігає у файл в форматі JPG.

4. Формує **ахроматичне** зображення з оригінального зображення. Виводить отримане зображення на екран дисплею. Зберігає у файл в форматі BMP.

5. Формує **перетворене** зображення за допомогою перетворення оригінального зображення відповідно до варіанту завдання. Виводить перетворене зображання на екран дисплею. Зберігає у файл в форматі JPG.

Таблиця 1

Варіанти завдання до лабораторної роботи 1

| Вар. № | № зображення | i, j пікселю | Канал кольори | Перетворення |
|---|---|---|---|---|
| 1 | 11 | 10, 15 | R | Горизонтальне симетричне відображення |
| 2 | 12 | 20, 15 | G | Вертикальне симетричне відображення |
| 3 | 13 | 30, 15 | B | Поворот на +90 градусів |
| 4 | 14 | 40, 15 | R | Поворот на -90 градусів |
| 5 | 15 | 50, 15 | G | Змістити зображення по горизонталі на 100 пікселів |
| 6 | 16 | 60, 15 | B | Змістити зображення по вертикалі на 200 пікселів |
| 7 | 17 | 70, 15 | R | Виділити ліву половину зображення |
| 8 | 18 | 80, 15 | G | Виділити праву половину зображення |
| 9 | 19 | 90, 15 | B | Виділити нижню половину зображення |
| 10 | 20 | 100, 15 | R | Виділити верхню половину зображення |
| 11 | 21 | 10, 5 | R | Горизонтальне симетричне відображення |
| 12 | 22 | 10, 35 | G | Вертикальне симетричне відображення |
| 13 | 23 | 20, 45 | B | Поворот на +90 градусів |
| 14 | 24 | 30, 55 | R | Поворот на -90 градусів |
| 16 | 25 | 40, 65 | G | Змістити зображення по вертикалі на 200 пікселів |
| 17 | 26 | 10, 75 | B | Виділити нижню половину зображення |
| 18 | 27 | 20, 85 | R | Виділити верхню половину зображення |
| 19 | 28 | 30, 95 | G | Виділити ліву половину зображення |
| 20 | 29 | 40, 105 | B | Виділити праву половину зображення |

## 2.2 Підготовка до лабораторної роботи №1

До найпростіших геометричних перетворень зображень відносяться:

- Обертання зображень на заданий кут (±90 градусів);
- Кадрування, тобто вибірка прямокутної частин зображення;
- Гортання: перевертання зображення горизонтально або вертикально.

Такі перетворення виконуються за допомогою відповідної перенумерації пікселів в оригінальному зображенні. Якщо, наприклад, виконується операція кадрування, спочатку необхідно визначити область інтересу (RoI ) - частина зображення, яке необхідно виділити. Для цього задаються координати (індекси пікселів) лівого верхнього кута RoI та її висота і ширина. Потім створюється порожнє зображення відповідного розміру, до якого послідовно переносяться значення пікселів RoI.

## 2.3 Контрольні питання для допуску до лабораторної роботи №1

Дайте визначення цифрового зображення.

Дайте визначення повнокольорового зображення.

Дайте визначення монохромного зображення.

Дайте визначення ахроматичного зображення.

Дайте визначення бінарного зображення.

## 2.4 Методичні вказівки до виконання лабораторної роботи № 1

### 2.4.1 До завдання 1.1. Завантажити зображення.

По-перше треба скачати необхідне а варіантом зображення (див. Додаток 1) з депозитарію на GitHub за посиланням http://githab

По-друге, завантажити зображення до програми за допомогою фрагменту коду відповідно 1.2.1

По-третє, відобразити зображення за допомогою фрагменту коду з 1.2.4.

### 2.4.2 До завдання 1.2. Структура, розмір, колір пікселя

Структуру зображення за змінною **image** можна визначити наступим чином:

```
rows_num = image.shape[0] ## кількість рядків
clms_num = image.shape[1] ## кількість колонок
pix_num = rows_num*clms_num ## кількість пікселів
```

Розмір зображення за змінною **image** можна визначити як:

```
image_size = image.size ## розмір
```

Колір пікселя зображення **image** з координатами ***i, j*** можна визначити як:

```
color_R = image[i,j,0]
color_G = image[i,j,1]
color_B = image[i,j,2]
```

### 2.4.3 До завдання 1.3. Монохромне зображення

Перетворення оригінального зображення до монохромного з визначеним за варіантом завдання кольоровим відтінком (наприклад ***B***) можна виконати за наступним кодом:

```
## Визначення монохромного зображення
image_out = np.zeros((rows_num,clms_num,3), dtype=np.uint8)
## Only BLUE channel
image_out[:, :, 0] = 0
```

```
image_out[:, :, 1] = 0
image_out[:, :, 2] = test_im[:, :, 2]
```

### 2.4.4 До завдання 1.4. Ахроматичне зображення

Для перетворення оригінального зображення до ахроматичного з урахуванням чутливості зору до «чистих» кольорів необхідно для всіх пікселів кольорові компоненти змішати за наступною пропорцію

$$R' = G' = B' = 0.299R + 0.587G + 0.114B.$$

Код цього перетворення:

```
## Визначення монохромного зображення
image_gray = np.zeros((rows_num,clms_num,3),dtype=np.uint8)
for i in  range (rows_num):
   for j in  range (clms_num):
      image_gray[i,j,:] = 0.299*test_im[i,j,0]+
      0.587*test_im[i,j,1]+0.114*test_im[i,j,2]
```

### 2.4.5 До завдання 1.5. Перетворення зображення

Найпростіші перетворення зображення зводиться до роботи з індексами масиву пікселів. У якості прикладу наведемо код, який виконує наступне перетворення. Оригінальне кольорове зображення **image** з структурою 900 рядків, 900 колонок. Необхідно сформувати нове зображення, як частину оригінального (450*450) та повернутого на 90 градусів.

```
image_crop_ = np.zeros((450,450,3), dtype=np.uint8)
for i in range (450):
   for j in range (450):
      image_crop_[i,j,:] = test_im[j, 450-i,:]
```

У додатку 2 наведено приклад повної програми до лабораторної роботи 1 та отримані зображення.

# 3. Лабораторна робота №2. Точкові методи препарування зображень

**Лабораторна робота №2 орієнтована на** оволодіння студентами базовими алгоритмами точкового (по піксельного) перетворення зображень.

**Мета лабораторної роботи:**

- Оволодіти алгоритмом перетворення ахроматичного зображення до бінарного (чорно-білого) із заданим порогом бінарізації**;**
- Навчитися виконувати операцію зрізу яскравості над ахроматичним зображенням;
- Засвоїти алгоритм отримання негативного зображення із ахроматичного.

У разі успішного виконання лабораторної роботи студент буде здатен обирати та використовувати програмні продукти для роботи із зображеннями та виконувати типові операції обробки цифрових зображень.

Успішне засвоєння матеріалів та виконання лабораторної роботи буде сприяти формуванню у студента наступних соціальних навичок та вмінь: бажання постійно вчитися, вміння дотримуватися регламентних рамок, оцінювати строки проведення та виконання роботи, обґрунтувати той чи інший спосіб діяльності при виконанні практичних завдань. Форми прояву: прагнення вирішувати поточні проблеми самостійно, бажання пробувати щось нове, йти на розумний ризик при прийнятті рішень, застосовувати елементи самоконтролю і самооцінки при виконанні роботи, вміння планувати свій час.

## 3.1 Завдання до лабораторної роботи № 2.

В процесі виконання лабораторної роботи студент повинен створити програмний додаток, що виконує наступні дії:

1. Зчитує оригінальне зображення у відповідності за варіантом завдання (номер зображення в таблиці 2). Виводить на екран дисплею завантажене зображення. Завдання виконується аналогічно лабораторній роботі №1. **Примітка:** Ім’я зображення за номером та місце збереження файлу наведені в достатку 1.

2. Формує **ахроматичне** зображення з оригінального. ***Рекомендація***: створити функцію перетворення кольорового зображення до ахроматичного на базі коду з лабораторної роботи №1.

3. Формує **бінарне (чорно-біле ) зображення** з ахроматичного, значення порогу (threshold) за варіантом з таблиці 2. Відображає на екрані дисплею порівняння ахроматичного та бінарного зображень.

4. Формує **зріз зображення** з ахроматичного, значення порогів (L_low, L_high) за варіантом з таблиці 2. Відображає на екрані дисплею порівняння ахроматичного зображення та зображення зрізу.

5. Формує **негативне зображення** з ахроматичного Відображає на екрані дисплею порівняння ахроматичного зображення та негативного зображення.

Таблиця 2

Варіанти завдання до лабораторної роботи 2

| Вар. № | № зображення | threshold | L_Low | L_High |
|---|---|---|---|---|
| 1 | 4 | 140 | 100 | 200 |
| 2 | 5 | 130 | 110 | 190 |
| 3 | 6 | 120 | 120 | 180 |
| 4 | 7 | 110 | 130 | 170 |
| 5 | 8 | 100 | 140 | 160 |
| 6 | 9 | 150 | 105 | 215 |
| 7 | 10 | 160 | 115 | 205 |
| 8 | 11 | 170 | 125 | 195 |
| 9 | 12 | 180 | 135 | 185 |
| 10 | 13 | 190 | 145 | 175 |
| 11 | 14 | 200 | 150 | 165 |
| 12 | 15 | 105 | 100 | 200 |
| 13 | 16 | 115 | 110 | 190 |
| 14 | 17 | 125 | 120 | 180 |
| 16 | 18 | 135 | 130 | 170 |
| 17 | 19 | 145 | 140 | 160 |
| 18 | 20 | 155 | 105 | 215 |
| 19 | 1 | 165 | 115 | 205 |
| 20 | 2 | 185 | 125 | 195 |

## 3.2 Підготовка до лабораторної роботи №2

Завдання лабораторної роботи орієнтовані на обробку ахроматичних зображень, які створюються із зображень за варіантом за допомогою коду із лабораторної роботи 1. При цьому кожен піксель ахроматичного зображення має однакові **R, G, B** значення кольорових координат, що змінюються від 0 до 255. Наді будемо позначати це значення як $\boldsymbol{L}$ – інтенсивність пікселю (також будемо вживати термін – яскравість пікселю).

Точкові методи перетворення ахроматичних цифрових зображень базуються на тому, що для кожного пікселя вихідного зображення виконується деяка зміна його інтенсивності (яскравості). Тобто яскравість кожного пікселю перераховується відповідно до деякої функції перетворення:

$$L'_{i,j} = F\left(L_{i,j}\right), \qquad i = 0,1,2,\dots,N-1; j = 0,1,2,\dots,M-1.$$

Перетворення до бінарного (чорно- білого) зображення виконується за допомогою функції

$$L'_{i,j} = \begin{cases} 0, & if\ L_{i,j} < L_{bin} \\ 255, & if\ L_{i,j} \geq L_{bin} \end{cases}, \tag{3.1}$$

де $L_{bin}$ – поріг (threshold) бінаризації. Графічне представлення функції перетворення до чорно-білого зображення наведена на рис. 3.1.

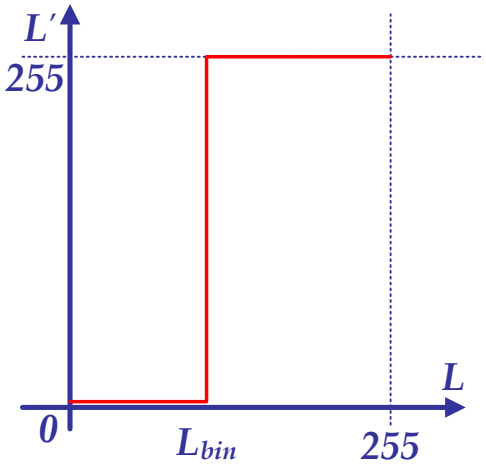



***Рисунок 3-1 Функція бінаризації зображення***

Так званий зріз яскравіти зображення виконується за допомогою функції

$$L'_{i,j} = \begin{cases} 0, & if\ (L_{i,j} < L_{low}\ or\ L_{i,j} \geq L_{high}) \\ L_{i,j}, & otherwise \end{cases}, \tag{3.2}$$

де $L_{low}$ - мінімальний поріг можливої яскравості, $L_{high}$ - максимальний поріг можливої яскравості. Графічне представлення функції створення зрізу яскравості зображення наведено на рис. 3.2.

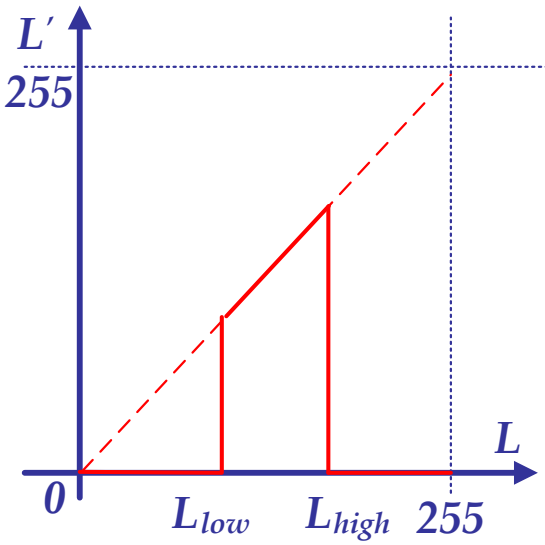



***Рисунок 3-2 Функція зрізу яскравості зображення***

Створення негативного зображення виконується за залежністю, вигляд якої наведено на рис. 3.3.

$$L'_{i,j} = 255 - L_{i,j}. \tag{3.3}$$

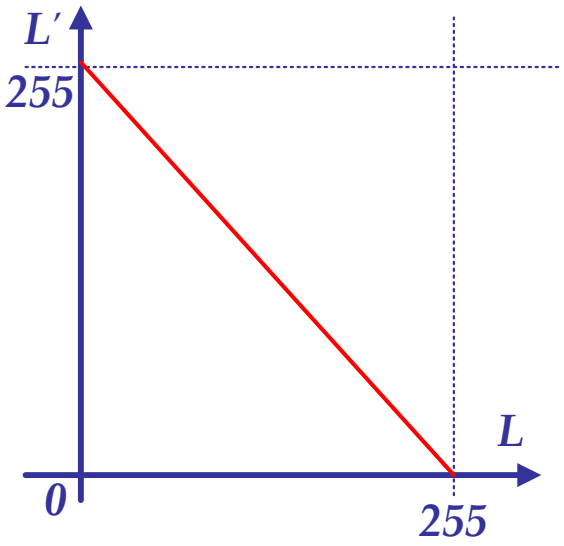



***Рисунок 3-3 Функція створення негативного ахроматичного зображення***

***Важливо*** – всі вище вказані перетворення виконуються для кожного пікселю зображення.

## 3.3 Контрольні питання для допуску до лабораторної роботи №2

Надайте визначення ахроматичного зображення.

Надайте визначення бінарного зображення.

Визначте функцію перетворення ахроматичного зображення до бінарного.

Визначте функцію зрізу яскравості.

Визначте функцію перетворення ахроматичного зображення до негативного ахроматичного.

## 3.4 Методичні вказівки до виконання лабораторної роботи № 2

### 3.4.1 Вивід двох зображень для порівняння

Для візуального порівняння двох зображень необхідно відтворити їх на дисплеї за допомогою наступного коду. Наприклад, є два зображення imag1 та image2.

```
import matplotlib.pyplot as plt
… … … … …
fig, axes = plt.subplots(1,2,figsize=(16,8))
ax = axes.ravel()
ax[0].imshow(image1)
ax[0].set_title("Ім’я першого зображення")
ax[1].imshow(image2)
ax[1].set_title("Ім’я другого зображення")
plt.show()
```

### 3.4.2 До завдань 2.1 та 2.2. Завантаження та створення ахроматичного зображення.

Завантаження та вивід на екран дисплею оригінального зображення виконується відповідно 1.2.1 та 2.4.1.

Для створення ахроматичного зображення рекомендовано відповідний фрагмент коду з лабораторної роботи №1 оформити у вигляді функції та застосувати її для завантаженого оригінального зображення.

### 3.4.3 До завдання 2.3. Чорно-біле зображення.

Відповідно до виразу 3.1 чорно-біле зображення з визначеним $L_{bin}$ можна створити за допомогою наступного коду

```
```
Lbin = . . . # значення за завданням
rows_num = image.shape[0] #кількість рядків зображення
clms_num = image.shape[1] #кількість стовпчиків зображення
# Визначення масиву бінарного зображення (! Цілком чорного)
bin = np.zeros((rows_num, clms_num, 3), dtype=np.uint8)
Lbin =   # вказуємо потрібний поріг бінаризації
for i in  range (rows_num):
    for j in  range (clms_num):
        if image [i, j, 0] > Lbin:
                bin[i, j, :] = 255
```
```

Для візуальне порівняння ахроматичного та бінарного (чорно-білого) зображень використовуємо код з 3.4.1. Приклад отриманих зображень наведені на рис. 3.4.

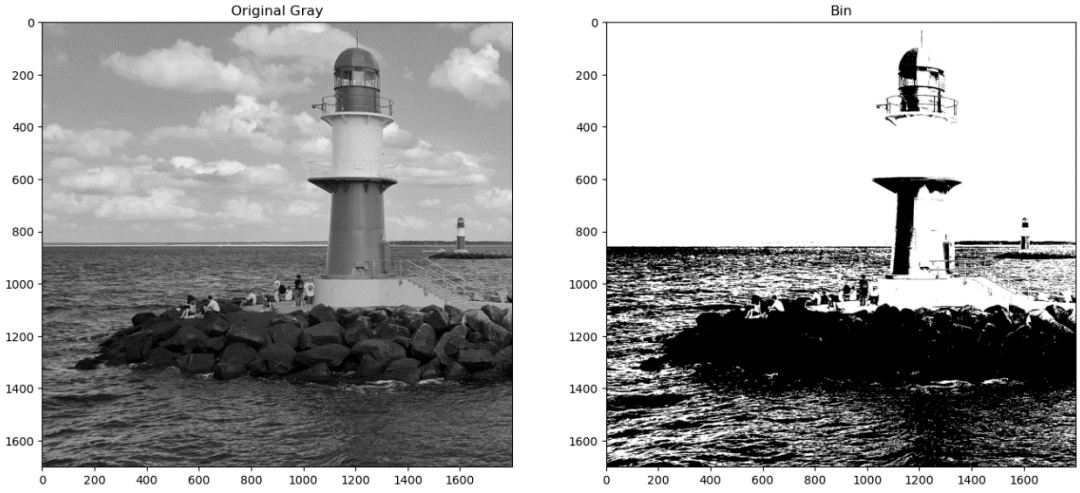



*Рисунок 3-4 Порівняння монохромного та бінарного зображення (threshold = 100)*

### 3.4.4 До завдання 2.4. Зріз яскравості.

Відповідно до виразу 3.2 зображення зрізу яскравості ахроматичного зображення із визначеними $L_{low}$ та $L_{high}$ можна отримати за допомогою наступного коду

```
```
rows_num = image.shape[0] #кількість рядків
clms_num = image.shape[1] #кількість стовпчиків
```
```

```
## Визначення повної копії ахроматичного зображення
slice = image.copy()
L_low =    # вказуємо ніжній поріг
L_high =   # вказуємо верхній поріг
for i in range (rows_num):
    for j in  range (clms_num):
        if image[i,j,0]<l_low or image[i,j,0] > l_high:
                slice[i,j,:] = 0
```

Для візуальне порівняння ахроматичного та чорно-білого зображень використовуємо код з 3.4.1. Приклад отриманих зображень наведені на рис. 3.5.

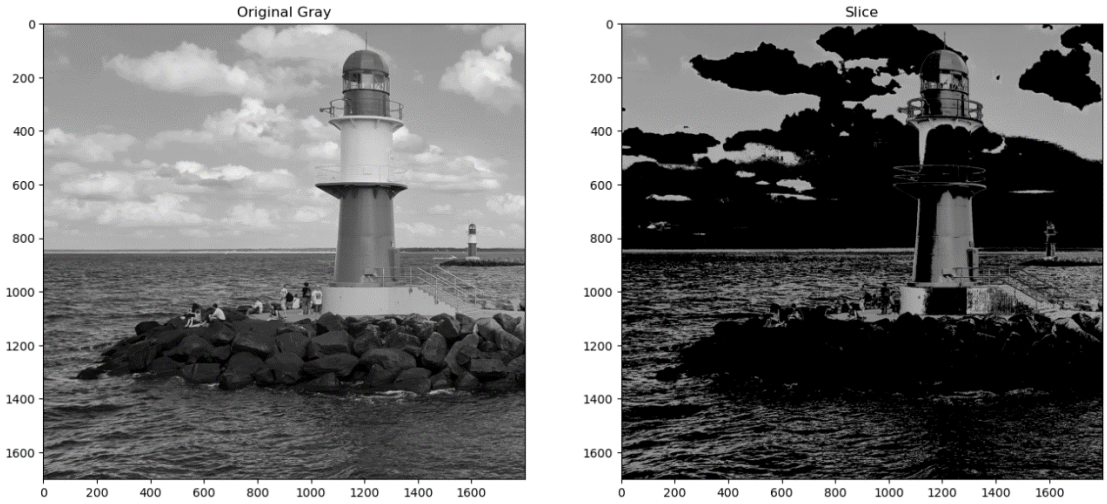



***Рисунок 3-5 Порівняння монохромного зображення та зрізу для наступних параметрів: ніжній поріг L_low = 100, верхній поріг L_high = 175.***

### 3.4.5 До завдання 2.5. Негативне зображення.

Звертаючись до виразу 3.3 негативне зображення відносно визначеного ахроматичного зображення можна отримати за допомогою наступного коду:

```
```
rows_num = image.shape[0] #кількість рядків
clms_num = image.shape[1] #кількість стовпчиків
## Визначення масиву НЕГАТИВНОГО зображення
neg = np.zeros((rows_num, clms_num, 3), dtype=np.uint8)
for i in  range (rows_num):
    for j in  range (clms_num):
        # Neg image
        neg [i,j,:] = 255 - image[i,j,0]
```

Приклад візуального порівняння ахроматичного та негативного зображень наведено на рис. 3.6.
```

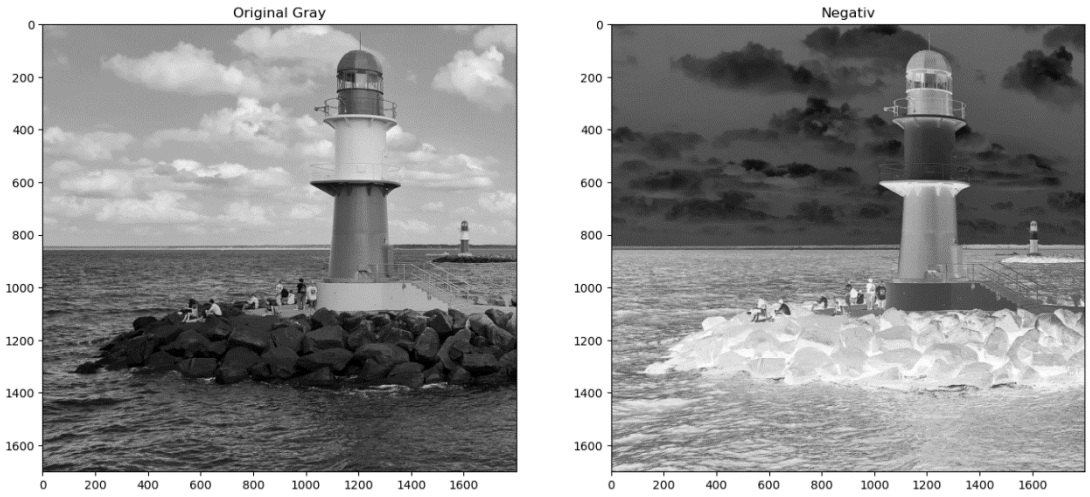



***Рисунок 3-6 Порівняння монохромного та негативного зображень.***

# 4. Лабораторна робота №3. Гістограма зображення. Керування яскравістю та контрастом зображення.

**Практична робота №3 орієнтована на** оволодіння студентами базовими алгоритмами побудови та роботи з гістограмами зображень.

**Мета лабораторної роботи:**

- Засвоїти поняття яскравості та контрасту зображення, методам обчислення середньої яскравості та контрасту зображення.
- Засвоїти поняття гістограми зображення, оволодіти алгоритмом створення та відображення гістограми, обчислення параметрів гістограми та прийомами її аналізу.
- Навчитися виконувати операцію зміщення яскравості ахроматичного зображення;
- Засвоїти алгоритм нормалізації ахроматичного зображення .

У разі успішного виконання лабораторної роботи студент буде здатен обирати та використовувати програмні продукти для роботи із зображеннями та виконувати типові операції обробки цифрових зображень.

Успішне засвоєння матеріалів та виконання лабораторної роботи буде сприяти формуванню у студента наступних соціальних навичок та вмінь: бажання постійно вчитися, вміння дотримуватися регламентних рамок, оцінювати строки проведення та виконання роботи, обґрунтувати той чи інший спосіб діяльності при виконанні практичних завдань. Форми прояву: прагнення вирішувати поточні проблеми самостійно, бажання пробувати щось нове, йти на розумний ризик при прийнятті рішень, застосовувати елементи самоконтролю і самооцінки при виконанні роботи, вміння планувати свій час.

## 4.1 Завдання до лабораторної роботи № 3.

В процесі виконання лабораторної роботи студент повинен створити програмний додаток, що виконує наступні дії:

1. Зчитує оригінальне зображення у відповідності за варіантом завдання (номер зображення дивись табл. 3). Виводить на екран дисплею завантажене зображення. Завдання виконується аналогічно лабораторній роботі №1. **Примітка:** Ім’я зображення за номером та місце збереження файлу наведені в достатку 1.

2. Формує **ахроматичне** зображення з оригінального. Обчислює середню яскравість та контраст зображення. ***Рекомендація***: створити функцію перетворення кольорового зображення до ахроматичного на базі коду з лабораторної роботи №1.

3. Формує **гістограму** отриманого ахроматичного зображення. Відображає гістограму на екрані дисплею. Визначає мінімальну та максимальну яскравість на зображенні та кількість пікселів з мінімальної та максимальною яскравістю.

4. Формує ахроматине зображення з зміщеною яскравістю. Значення зміщення L_shift наведено за варіантом у таблиці 3. Відображає на екрані дисплею порівняння оригінального ахроматичного зображення та зображення із зміщеною яскравістю. Обчислює середню яскравість та контраст зображення із зміщеною яскравістю.

***Зробити висновок*** щодо порівняння отриманих значень із відповідними для оригінального зображення.

5. Сформує **гістограму** зображення із зміщеною яскравістю. Відображає на екрані дисплею порівняння гістограм оригінального зображення та зображення із зміщеною яскравістю. Визначає мінімальну та максимальну яскравість на зображенні із зміщеною яскравістю та кількість пікселів з мінімальної та максимальною яскравістю.

6. Виконує **нормалізацію** гістограми оригінального зображення. Формує нормалізоване зображення. Відображає на екрані дисплею порівняння оригінального ахроматичного зображення та нормалізованого зображення. Обчислює середню яскравість та контраст нормалізованого зображення

***Зробити висновок*** щодо порівняння отриманих значень із відповідними для оригінального зображення.

7. Формує **гістограму** нормалізованого зображення. Відображає на екрані дисплею порівняння гістограм оригінального зображення та нормалізованого. Визначає мінімальну та максимальну яскравість на нормалізованому зображенні та кількість пікселів з мінімальної та максимальною яскравістю.

Таблиця 3

Варіанти завдання до лабораторної роботи 3

| Вар. № | № зображення | L_Shift |
|---|---|---|
| 1 | 10 | 10 |
| 2 | 11 | 15 |
| 3 | 12 | 20 |
| 4 | 13 | 25 |
| 5 | 14 | 30 |
| 6 | 15 | 25 |
| 7 | 16 | 25 |
| 8 | 17 | 15 |
| 9 | 18 | 10 |
| 10 | 19 | 5 |
| 11 | 20 | -10 |
| 12 | 1 | -15 |
| 13 | 2 | -20 |
| 14 | 3 | -25 |
| 16 | 4 | -30 |
| 17 | 5 | -25 |
| 18 | 6 | -25 |
| 19 | 7 | -15 |
| 20 | 8 | -10 |

## 4.2 Підготовка до лабораторної роботи №3

Основні завдання лабораторної роботи пов’язані з поняттями яскравості та контрасту зображень та гістограми зображень.

***Яскравість*** зображення – інтегроване візуальне сприйняття наглядачем значущих рівнів інтенсивності випромінювання пікселів зображення, свідоцтво переважання певного рівня яскравості. Типова оцінка яскравості ахроматичного цифрового зображення розміром $N \times M$ обчислюється як середня яскравість всіх пікселів зображення:

$$\overline{L} = \frac{1}{M * N} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (L_{i,j}) \quad (4.1)$$

**Контрастність** ахроматичного зображення це безрозмірна величина, що характеризує різницю яскравості точок (пікселів) зображення. За спрощенням – це є візуальне сприйняття співвідношення яскравості найсвітлішої та найтемнішої частин зображення. Чисельна оцінка контрастності ***C*** цифрового ахроматичного зображення зазвичай виконується як **середньоквадратична контрастність** - стандартне відхилення яскравості пікселя $L_{i,j}$від середньої яскравості $\overline{L}$ растрового зображення розмірами *N* × M*:*

$$C = \frac{1}{M * N} \sqrt{\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (L_{i,j}^2 - \overline{L})} \quad (4.2)$$

**Гістограма зображення** - це графічне відображення статистичного розподілу пікселів цифрового зображення з різною яскравістю, в якому по горизонтальній осі представлена яскравість, а по вертикалі – абсолютне (або відносне) число пікселів з конкретним значенням яскравості. Гістограма для ахроматичного зображення з кількістю рівнів яскравості **256** будується як одновимірний масив з 256 елементів, ***k***-й елемент котрого містить кількість пікселів, яскравість яких дорівнює ***k*** та відображається як діаграма розподілу кількості пікселів на зображенні з однаковою яскравістю. На рис. 4.1 наведено приклад зображення та відповідна зображення.

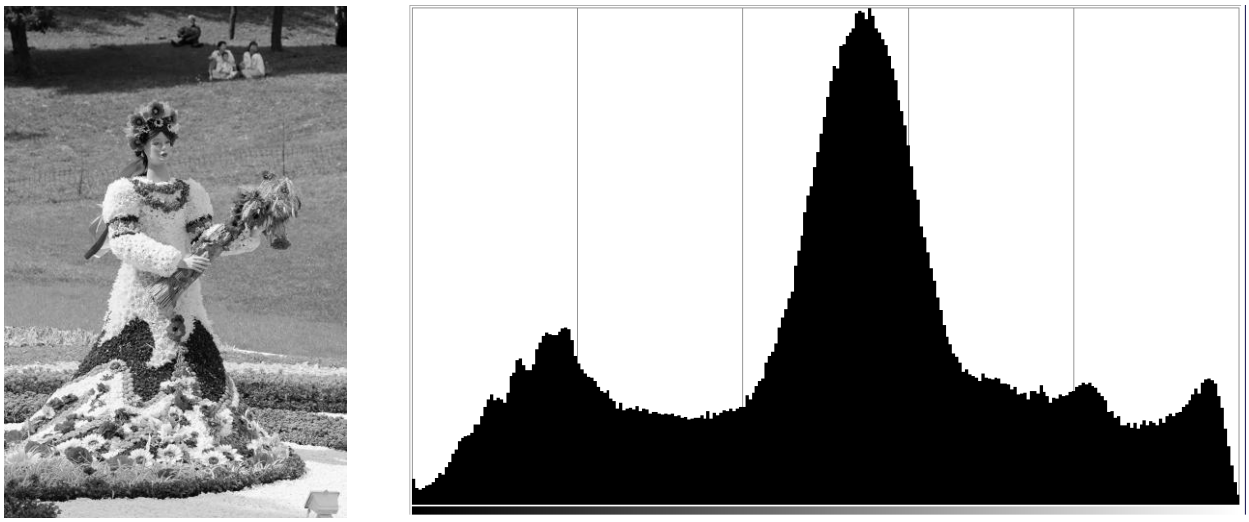


***Рисунок 4-1 Зображення та гістограма зображення.***

Аналізуючи гістограму, можна отримати загальне уявлення про «правильність» розподілу яскравості, контрасту, оцінити необхідну корекцію зображення. Наприклад,

- вузька гістограма поблизу центру діапазону яскравості - зображення з низьким контрастом;

- ненульові рівні гістограми покривають широку частину діапазону яскравості, тобто розподіл близький до рівномірного - високо контрастне зображення.

На цих властивостях побудовані різноманітні методи візуального поліпшення зображень (розтягування яскравості, нормалізація, еквілізація). Розглянемо метод еквілізації гістограми, що підвищує контрастність зображення.

Позначмо загальну кількість пікселів зображення як ***NM*** **=** ***M*** х ***N*** та **Bins** – кількість рівнів яскравості**.** Гістограма зображення визначає імовірність $p_k$ появи на зображенні пікселя яскравість якого дорівнює ***k***

$$p_k = \frac{n_k}{NM}\ , \tag{4.3}$$

де $n_k$- кількість пікселів з рівнем яскравості ***k.***

З іншого боку при рівномірному розподілі яскравістей на кожен рівень яскравості повинно припадати

$$n_{awer} = \frac{NM}{Bins} \text{ пікселів.}$$

Тобто необхідно перетворити «випадковий» розподіл яскравостей пікселів похідного зображення в розподіл за рівномірним законом.

З точки зору теорії ймовірності, функція кумулятивного розподілу інтенсивностей визначається як

$$S_k = \sum_{j=0}^{k} (p_k),\ k = 0,1, \dots, Bin - 1 \tag{4.4}$$

За допомогою лінеаризації цієї функції визначаються всі можливі значення яскравості в приблизно однаковій кількості для «еквівалентної» гістограми

$$p_k^{new} = round\left(\frac{S_k - S_{min}}{MN - S_{min}}(L-2)\right) + 1 \tag{4.5}$$

## 4.3 Контрольні питання для допуску до лабораторної роботи №3

Визначте функцію обчислення яскравості зображення.

Визначте функцію обчислення контрасту зображення.

Надайте визначення гістограми зображення.

Поясніть підходи щодо якісного аналізу зображення за допомогою гістограми.

Поясніть процедуру еквілізації гістограми.

## 4.4 Методичні вказівки до виконання лабораторної роботи № 3

### 4.4.1 До завдань 3.1 та 3.2. Завантаження та створення ахроматичного зображення.

Завантаження та вивід на екран дисплею оригінального зображення виконується відповідно 1.2.1 та 2.4.1.

Для створення ахроматичного зображення можна використати функцію, створену за завданням 2.2. та застосувати її для завантаженого оригінального зображення.

Розрахунок середньої яскравості та контрасту ахроматичного зображення, наприклад з ім'ям **ahromimg,** можна виконати наступним чином:

```
```
L_averg = np.average(ahromimg[:,:,0])
L_std = np.std(ahromimg [:,:,0])
```
```

### 4.4.2 До завдання 3.3. Обчислення гістограми.

Гістограма формується як масив, який у якості елементів має кількість пікселів однакової яскравості. Далі приклад для зображення **ahromimg** із 256 рівнями яскравості.

```
```
histoarray = np.zeros((256), dtype=np.uint32)
    ##Формування гістограми
    for i in range (ahromimg.shape[0]):
        for j in range (ahromimg.shape[1]):
            histoarray[ahromimg[i,j,0]] += 1
```
```

Відобразити зображення та відповідну гістограму можна наступним чином:

```
```
fig, axes = plt.subplots(1, 2, figsize=(10, 5))
ax = axes.ravel()
ax[0].imshow(ahromimg)
ax[0].set_title("Original Gray")
pix_index = np.arange(256)
ax[1].bar(pix_index, histoarray)
ax[1].set_title('Image Histogram')
plt.show()
```
```

Приклад отриманих зображень дивись рис. 4.2.

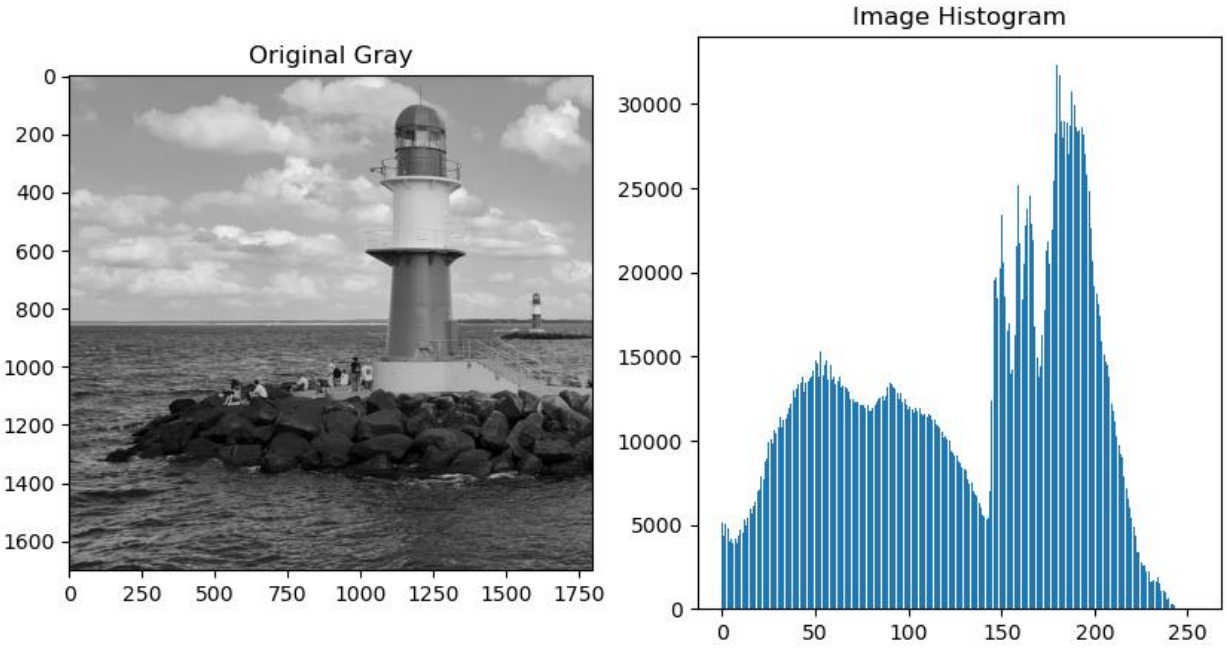



***Рисунок 4-2 Оригінальне ахроматичне зображення та його гістограма. Середня яскравість = 128,1 . Середнє квадратична контрастність = 61,.8***

### 4.4.3 До завдань 3.4, 3.5. Зміщення яскравості.

Для визначеного зображення **ahromimg** та заданого зміщення **L_bias** розрахувати зображення із зміщеною яскравістю **im_shift** можна за допомогою наступного коду:

```
rows_num = ahromimg.shape[0]
clms_num = ahromimg.shape[1]
im_shift = np.zeros((rows_num,clms_num,3),dtype=np.uint8)
for i in range (rows_num):
    for j in range (clms_num):
        L = np.int32(ahromimg[i,j,0] + L_bias)
        if L < 0:
            im_shift[i,j,:] = np.int8(0)
        elif L>bins-1:
            im_shift[i,j,:] = np.int8(255)
        else :
            im_shift [i,j,:] = np.int8(L)
```

Розрахунок яскравості та контрасту для **im_shift** аналогічно 4.4.1. Розрахунок гістограми та її відображення аналогічно 4.4.2.

Приклад отриманих зображень дивись рис. 4.3, 4.4.

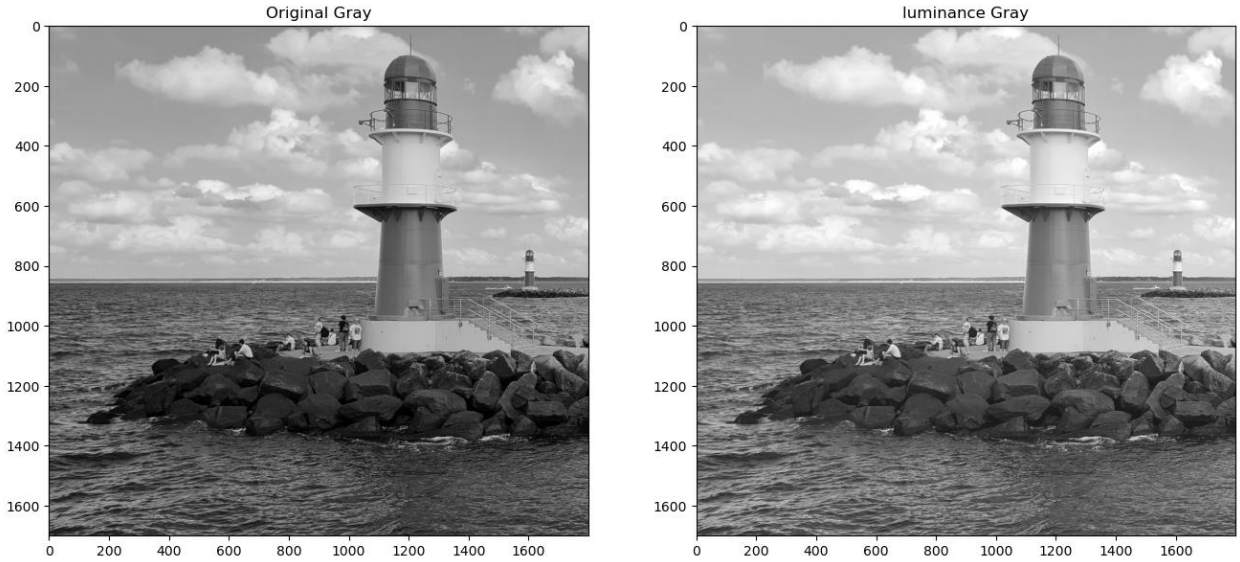



***Рисунок 4-3 Оригінальне ахроматичне зображення та зображення із зміщеною інтенсивністю.***

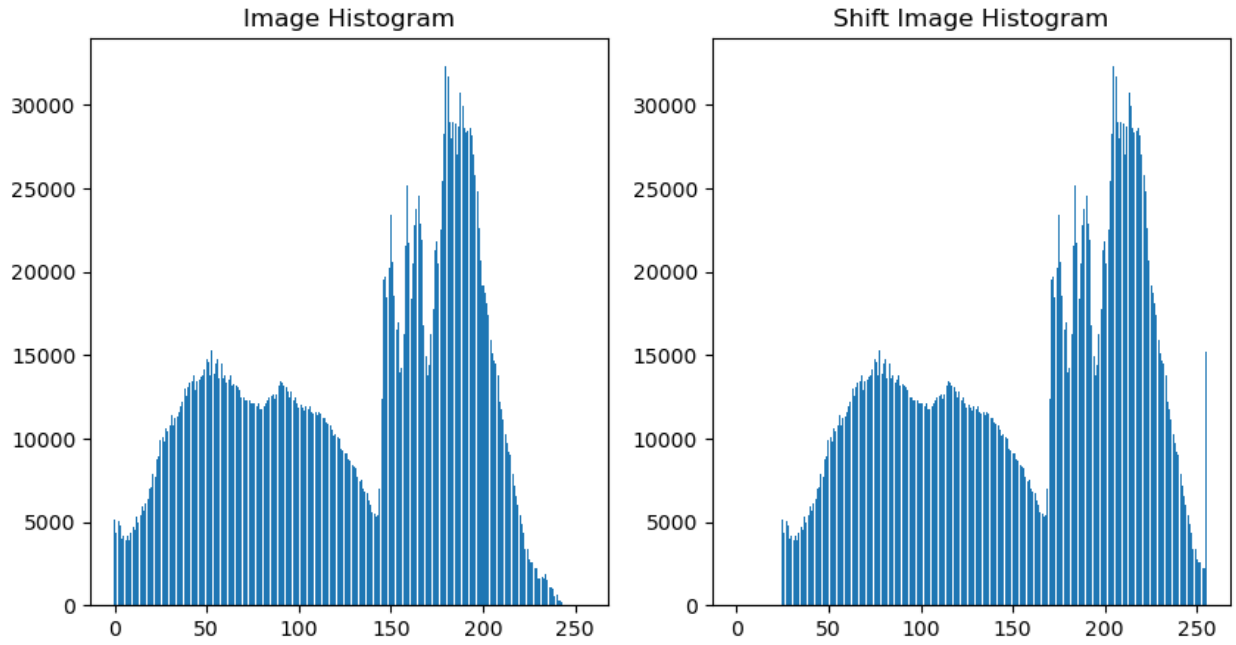



***Рисунок 4-4 Порівняння гістограм оригінального ахроматичного зображення та гістограми зображення із зміщеною інтенсивністю. Середня яскравість = 153,1 . Середнє квадратична контрастність = 61,.3.***

### 4.4.4 До завдання 3.6. Еквілізація гістограми.

Попередньо визначені: кількість пікселів в зображенні pix_num = N*M та кількість рівнів яскравості **bins** = 256. Також обчислена гістограма оригінального зображення **histoarray.** В наступному коді обчислюються нормалізована гістограма **norm_histo** за (4.3), комулятивна гістограма **cum_histo** за (4.4) та перетворене зображення **norm_image** за (4.5).

```
Norm_histo = np.zeros((256), dtype=np.float32)
## Розраховуємо гістограму та нормуємо до максимального значення 255
norm_histo[:] = 255.0* histoarray[:]/pix_num
```

```
norm_sum = np.sum(norm_histo)

# Кумулятивна гістограма
cum_histo = np.zeros ((256), dtype=np.uint8)
for i in  range (bins):
    Cum_histo[i] = np.uint8(np.sum(norm_histo [0:i])+0.5)

# Еквілізоване зображення
norm_image = np.zeros((rows_num, clms_num,3), dtype=np.uint8)
for i in range (rows_num):
    for j in range (clms_num):
        # New equilized image
        norm_image[i,j,:] = Cum_histo[ahromimg[i, j, 0]]
```

Візуальне порівняння оригінального та еквілізованого зображень виконується стандартним чином.  Аналогічно виконується візуальне порівняння гістограм.

Приклад отриманих зображень дивись рис. 4.5, 4.6.

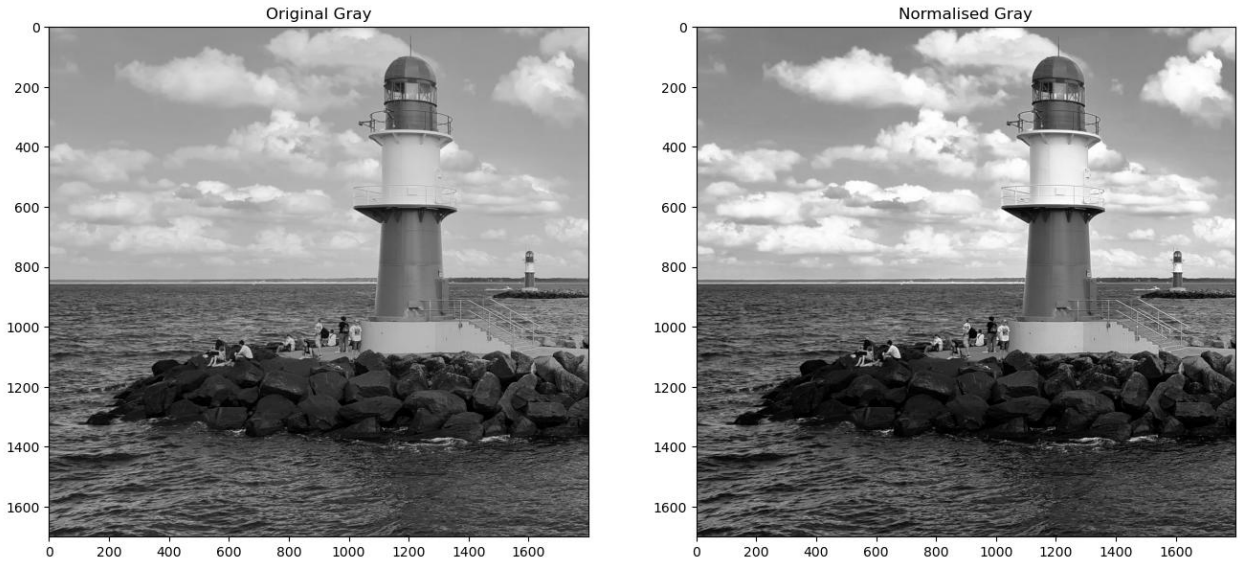


***Рисунок 4-5 Оригінального ахроматичного зображення та зображення із нормалізованою гістограмою.***

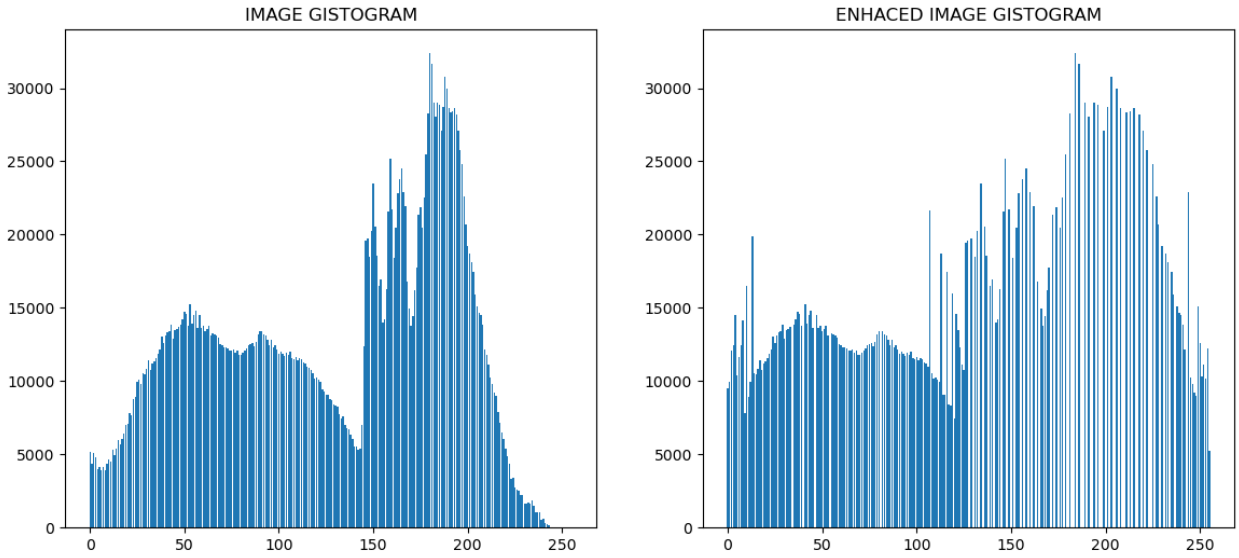



***Рисунок 4-6 Гістограми оригінального ахроматичного зображення та зображення із нормалізованою гістограмою. Середня яскравість = 126,1 . Контраст = 73,.4***

***Примітка:*** порівняйте значення середньої яскравості та контрасту для оригінального зображення, зображення із зміщеною яскравістю та нормалізованого зображення.

# 5. Лабораторна робота №4. Просторова обробка зображень. Лінійна фільтрація. Розмивання зображень.

**Практична робота №4 орієнтована на** оволодіння студентами базовими алгоритмами лінійної фільтрації зображень.

**Мета лабораторної роботи:**

- Засвоїти загальне поняття фільтрації зображення, математичного опису дискретної згортки та способу завдання околу фільтру.
- Оволодіти методом розмивання зображень за допомогою середньоарифметичного фільтру.
- Оволодіти методом розмивання зображень ї за допомогою фільтру Гауса.

У разі успішного виконання лабораторної роботи студент буде здатен обирати та використовувати програмні продукти для роботи із зображеннями та виконувати типові операції обробки цифрових зображень.

Успішне засвоєння матеріалів та виконання лабораторної роботи буде сприяти формуванню у студента наступних соціальних навичок та вмінь: бажання постійно вчитися, вміння дотримуватися регламентних рамок, оцінювати строки проведення та виконання роботи, обґрунтувати той чи інший спосіб діяльності при виконанні практичних завдань. Форми прояву: прагнення вирішувати поточні проблеми самостійно, бажання пробувати щось нове, йти на розумний ризик при прийнятті рішень, застосовувати елементи самоконтролю і самооцінки при виконанні роботи, вміння планувати свій час.

## 5.1 Завдання до лабораторної роботи № 4.

В процесі виконання лабораторної роботи студент повинен створити програмний додаток, що виконує наступні дії:

1. Зчитує оригінальне зображення у відповідності за варіантом завдання (номер зображення дивись табл. 4). Формує ахроматичне зображення з оригінального. Формує гістограму отриманого ахроматичного зображення. На екрані дисплею відображає оригінальне зображення, відповідне ахроматичне зображення та його гістограму.

2. За допомогою середньоарифметичного фільтру з радіусом околу $\boldsymbol{s = 1}$ формує перший варіант розмитого кольорового зображення. Формує ахроматичне зображення першого варіанту розмитого. Формує гістограму отриманого ахроматичного зображення. На екрані дисплею відображає оригінальне зображення, перший варіант розмитого та порівняння їх гістограм.

3. Розраховує коефіцієнти ядра фільтру Гауса для радіуса околу $\boldsymbol{s = 2}$ та радіусу розмивання $\boldsymbol{\sigma}$, що заданий в таблиці варіантів (Табл. 4).

4. За допомогою фільтру Гаусу з обчисленим ядром формує другий варіант розмитого кольорового зображення. Формує ахроматичне зображення другого варіанту. Формує гістограму отриманого ахроматичного зображення. На екрані дисплею відображає оригінальне зображення, другий варіант розмитого та порівняння їх гістограм.

5. Із зображень оригінального та розмитих за першим та другим варіантом виділяє фрагмент зображення із визначеними параметрами (табл. 4) та відображає їх порівняння на екрані дисплею.

За результатами роботи програми необхідно порівняти отримані зображення і зробити висновки щодо якості розмивання.

Таблиця 4

Варіанти завдання до лабораторної роботи 4

| Вар. № | № зображення | σ | Центр вікна I,j | Розмір прямокутного вікна |
|---|---|---|---|---|
| 1 | 21 | 1 | 100, 200 | 100 |
| 2 | 22 | 0,9 | 120, 200 | 120 |
| 3 | 23 | 0,8 | 140, 200 | 140 |
| 4 | 24 | 0,7 | 160, 200 | 160 |
| 5 | 25 | 0,6 | 180, 200 | 180 |
| 6 | 26 | 0,5 | 200, 200 | 200 |
| 7 | 27 | 0,4 | 200, 100 | 100 |
| 8 | 28 | 1,1 | 200, 120 | 120 |
| 9 | 29 | 1,2 | 200, 140 | 140 |
| 10 | 30 | 1,3 | 200, 160 | 160 |
| 11 | 19 | 1,2 | 200, 180 | 180 |
| 12 | 18 | 1,1 | 200, 200 | 200 |
| 13 | 17 | 1,0 | 100, 200 | 100 |
| 14 | 16 | 0,7 | 110, 100 | 120 |
| 16 | 15 | 0,6 | 120, 120 | 140 |
| 17 | 14 | 0,5 | 140, 140 | 160 |
| 18 | 13 | 0,4 | 160, 160 | 180 |
| 19 | 12 | 1,1 | 180, 180 | 200 |
| 20 | 11 | 1,2 | 200, 100 | 180 |

## 5.2 Підготовка до лабораторної роботи №4

### 5.2.1 Просторове перетворення зображень. Загальне визначення

Операцію просторове перетворення зображення (ахроматичних) в загальному сенсі можна визначити як

$$\boldsymbol{I}^{(2)} = \mathbb{F}(\boldsymbol{I}^{(1)}),$$

де інтенсивність $L_{i,j}$ кожного пікселя перетвореного (вихідного) зображення $\boldsymbol{I}^{(2)}$ обчислюється за допомогою, в деякому розумінні, функції, що інтегрує (згортує) значення інтенсивності пікселів що належать визначеному околу пікселя ***i,j*** початкового $\boldsymbol{I}^{(1)}$ зображення:

$$L_{i,j}^{(2)} = \boldsymbol{F}\left(L_{k,q}^{(1)}\right), \quad \forall k,q \in \text{окіл } i,j.$$

Операція просторового перетворення ілюструються рисунком 5.1. Таким чином яскравість (інтенсивність) кожного пікселю перераховується відповідно до деякої функції перетворення над визначеним околом.

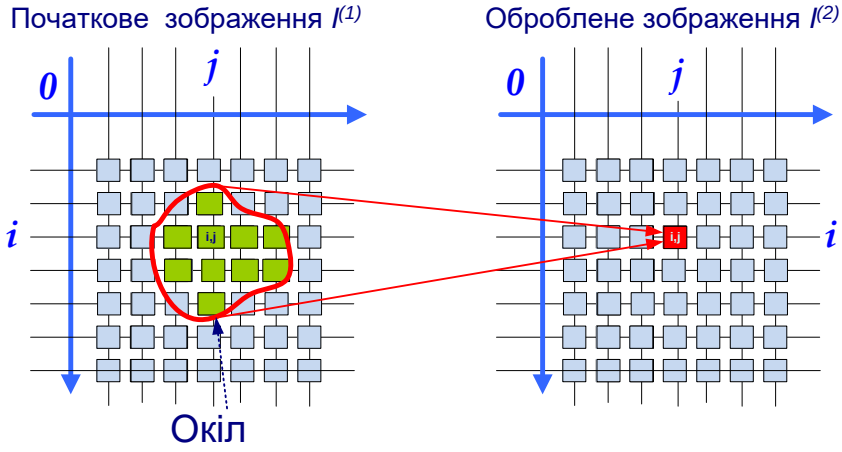



***Рисунок 5-1 Просторове перетворення зображення***

Надалі в практичних роботах розглядаються лише неказуальні квадратні околи ***S***, тобто околи, де обидві координати (номер рядка ***k*** і номер стовпця ***q***) всіх пікселів околу ***i,j*** - го пікселю похідного зображення задовольняють умові

$$|k-i| \le s, |q-j| \le s, \ \forall k,q \in \text{окіл } i,j,$$

де ***s*** – розмір (радіус) околу ***S***. Наприклад, якщо ***s* = 1,** то індекси ***k, q*** околу будуть приймати значення **-1, 0, +1** і, відповідно, індекси пікселів початкового зображення буть приймати значення (рис. 5.2):

| | | |
|---|---|---|
| ***I - 1, j - 1*** | ***i - 1, j*** | ***i - 1, j + 1*** |
| ***i, j - 1*** | ***i, j*** | ***i, j + 1*** |
| ***i + 1, j - 1*** | ***i +1, j*** | ***i + 1, j + 1*** |

***Рисунок 5-2 Індексація пікселів квадратного казуального околу одиничного радіусу.***

В залежності від вигляду функції $\boldsymbol{F}(\ \ )$, просторові перетворення підрозділяються на лінійну та нелінійну фільтрацію.

### 5.2.2 Лінійна фільтрація.

Лінійне фільтрування зображень може бути математично представлене як операція згортки між вхідним зображенням $\boldsymbol{I}^{(1)}$, яке представлене як матриця $\left\|L^{(1)}(i,j)\right\|$, та ядром фільтру, яке представлене як матриця $\|f(k,q)\|$. Яскравість кожного пікселя вихідного зображення $\boldsymbol{I}^{(2)}$, яке представлене як матриця $\left\|L^{(2)}(i,j)\right\|$, обчислюється наступним чином:

$$L_{i,j}^{(2)} = \boldsymbol{F}\left(L_{k,q}^{(1)}\right) = \frac{1}{D}\sum_{k=-s}^{s}\sum_{q=-s}^{s} L_{i-k,j-q}^{(1)} * f_{k,q},$$

де $\|f(k,q)\|$ - матриця коефіцієнтів фільтру (або матриця згортки фільтру, або ядро фільтру).

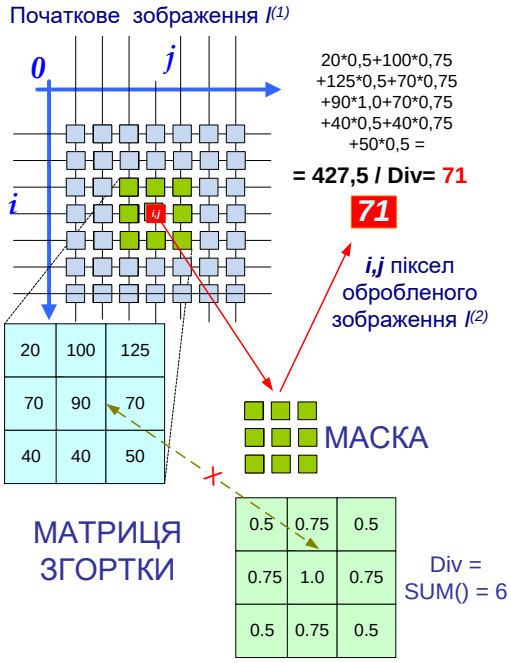



*Рисунок 5-3 Лінійна фільтрація*

Операцію згортки проілюстровано рис. 5.2. На рисунку окіл (маска) ***S*** має радіус 1, тобто матриця коефіцієнтів 3*3. Фільтрація виконується послідовно для кожного ***i,j*** пікселя початкового зображення. На основі значень яскравості ***i,j*** пікселя та його околу розраховується яскравість ***i,j*** пікселя результуючого зображення. В прикладі:

$$L_{i,j}^{(2)} = \frac{1}{6}(0{,}5L_{i-1,j-1}^{(1)} + 0{,}75L_{i-1,j}^{(1)} + 0{,}5L_{i-1,j+1}^{(1)} + 0{,}75L_{i,j-1}^{(1)} + L_{i,j}^{(1)} + 0{,}75L_{i,j+1}^{(1)} + (0{,}5L_{i+1,j-1}^{(1)}P + 0{,}75L_{i+1,j}^{(1)} + 0{,}5L_{i+1,j+1}^{(1)}$$

З урахуванням яскравості пікселів, що наведені у прикладі, значення пікселя фільтрованого зображення буде **71**. Зверніть увагу, що параметр ***D*** функції згортки в нашому прикладі дорівнює **6**, що є сумою усіх коефіцієнтів ядра фільтру.

Приклад демонструє, що коефіцієнти ядра фільтру визначають, скільки кожен навколишній піксель околу впливає на результуюче значення. Змінюючи значення коефіцієнтів в ядрі фільтру, можна застосовувати різні лінійні операції фільтрування до вхідного зображення.

### 5.2.3 Середньоарифметичні фільтри розмивання

Найпростіший лінійний фільтр є **середньоарифметичний фільтр розмивання**, який усереднює значення яскравості пікселів околу

$$L_{i,j}^{(2)} = \boldsymbol{F}\left(L_{k,q}^{(1)}\right) = \frac{1}{D}\sum_{k=-s}^{s}\sum_{q=-s}^{s} L_{i-k,j-q}^{(1)}.$$

Тобто всі коефіцієнти такого фільтру **рівні одиниці**, параметр ***D*** для околу радіусом 1 дорівнює **9**, а для околу радіусом 2 дорівнює **25**.

Більш якісними лінійними фільтрами розмивання є фільтри зваженого середнього (фільтри Гаусу), що використовують дискретну апроксимацію функції Гауса

$$L_{i,j}^{(2)} = \frac{1}{2\pi\sigma^2}\sum_{k=-s}^{s}\sum_{q=-s}^{s} L_{i-k,j-q}^{(1)}\, e^{-\frac{d^2}{2\sigma^2}}$$

де **σ** – радіус розмивання, $s \approx 3\sigma$, $d = \sqrt{k^2+q^2}$.

У таких фільтрах, кожен навколишній піксель впливає на фінальний результат з різною вагою, яку відображає ядро фільтру. Ступінь впливу визначається квадратом відстані до центрального пікселя. На рис. 5.4 наведено типове ядро фільтру Гауса для $\sigma = 1$, $s = 2$.

| | | | | |
|---|---|---|---|---|
| *0,003* | *0,013* | *0,023* | *0,013* | *0,003* |
| *0,016* | *0,059* | *0,097* | *0,059* | *0,013* |
| *0,023* | *0,097* | *0,159* | *0,097* | *0,023* |
| *0,013* | *0,059* | *0,097* | *0,059* | *0,013* |
| *0,003* | *0,013* | *0,023* | *0,013* | *0,003* |

***Рисунок 5-4 Ядро зваженого середнього фільтру Гауса***

Обчислення коефіцієнтів фільтру Гауса зазвичай виконується при визначених значеннях радіусу околу та радіусу розмивання. В лабораторній роботі будемо вважати заданим радіус околу $s = 2$, значення $\sigma$ – обирати за варіантом завдання.

## 5.3 Контрольні питання для допуску до лабораторної роботи № 4

Поясніть поняття околу пікселя та надайте формальне визначення неказуального квадратного околу.

Поясніть операцію дискретної згортки.

Визначте середньоарифметичний фільтр розмивання.

Визначте фільтр Гауса розмивання.

## 5.4 Методичні вказівки до виконання лабораторної роботи № 4

### 5.4.1 До завдання 4.1. Завантаження, створення ахроматичного зображення та гістограми.

Завантаження та вивід на екран дисплею оригінального зображення виконується відповідно 1.2.1 та 2.4.1. Для створення ахроматичного зображення можна використати функцію, створену за завданням 2.2. та застосувати її до завантаженого оригінального зображення. Гістограму можна обчислити та відобразити аналогічно завданню 3.3.

### 5.4.2 До завдання 4.2. Середньоарифметичний фільтр.

В наступному прикладі визначені розмір фільтру 3 Х 3, оригінальне зображення **test_im** вихідне ображення **filtr_im**.

```
# Визначення маски фильтру розмиття
L = 3; mask_row = L; mask_clm = L
# Визначення файлу перетвореного зображення
filtr_im = np.zeros((rows_num,clms_num, 3),dtype=np.uint32)
for i in range(1,(rows_num-1),1):
    for j in range(1,(clms_num-1),1):
        filtr_im[i,j,:] = 0
        for l in range(mask_row):
            for k in range(mask_clm):
                filtr_im [i,j,:] += test_im [i-(1-k),j-(1-l),:]
        filtr_im[i,j,:] = np.uint8(filtr_im[i,j,:]/9)
```

Візуальне порівняння оригінального та розмитого за допомогою середньоарифметичного фільтру зображень наведено на рис. 5.5.

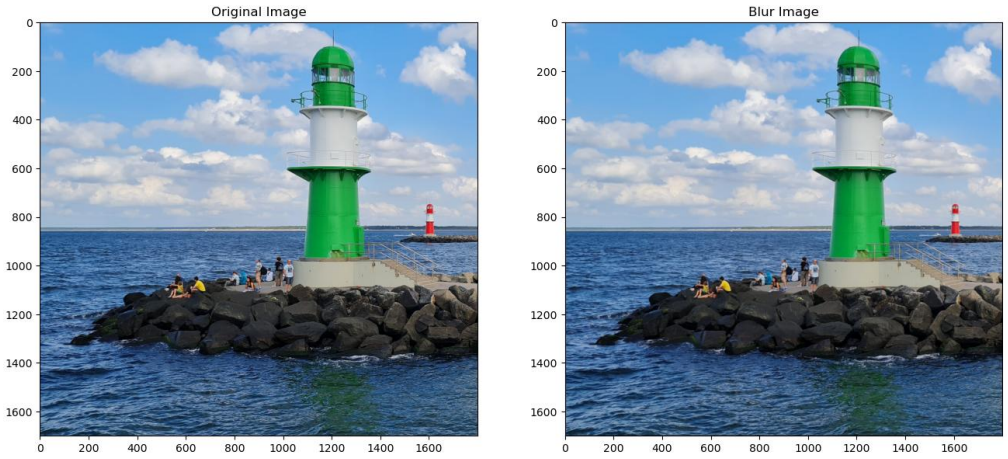



***Рисунок 5-5 Порівняння оригінального та розмитого зображень***

Для більш наглядного візуального ефекту фільтрації рекомендовано виділити невелике вікно в отриманих зображеннях. На рис. 5.6 наведено центральні частини зображень (вікно з координатами верхнього лівого кута [800, 800] та розміром 400 пікселів.


***Рисунок 5-6 Збільшена центральна частина оригінального та розмитого зображень***

Зверніть увагу на явно виражений ефект розмиття.

### 5.4.3 До завдання 4.3. Розрахунок ядра фільтру Гауса

Розрахунок коефіцієнтів ядра фільтру Гауса (**mask_koeff**) заданих радіуса околу ***s*** **= 2** та радіусу розмивання **sigma** та розміру маски **L** можна виконати за наступним кодом:

```
D = 2*np.pi*sigma*sigma
kn = - 1/(2*sigma*sigma)
mask_row = L
mask_clm = L
mask_koeff=np.zeros((mask_row,mask_clm,1), dtype=np.float32)
for k in range(mask_row):
    i = k - S
    for q in range (mask_clm):
        j = q - S
        mask_koeff [k, q] = 1/D*np.exp(kn*((i*i)+(j*j)))
```

Для перевірки розрахунку в якості тесту використовуйте приклад рис. 5.4.

### 5.4.4 До завдання 4.4. Фільтр Гауса

Отримання розмитого за допомогою фільтру Гауса зображення **filtr_im_Gaus** можна виконати наступним чином:

```
filtr_im_Gaus = np.zeros((rows_num,clms_num,3),dtype=np.uint8)

for i in range(2,(rows_num-2),1):
    for j in range(2,(clms_num-2),1):
        for l in range (mask_row):
            for k in range (mask_clm):
            filtr_im_Gaus[i,j,:]+=np.uint8(mask_koeff[l,k]*
            test_im[i-(2-k), j-(2-l),:])
```

Аналогічно середньоарифметичному фільтру рекомендовано вирізати і порівняти центральну частину зображення (рис.  5.7).


***Рисунок 5-7 Збільшена центральна частина оригінального та розмитого за Гаусом зображень***

### 5.4.5 До завдання 4.5. Порівняння  фільтрованих зображень

Для порівняння  зображень рекомендовано вивести на екран вікна зображень та з другого боку обчислити гістограми (практична робота 3) оригінального та фільтрованих зображень та відобразити їх на екрані дисплею. Приклад порівняння гістограм наведено на рис. 5-8.

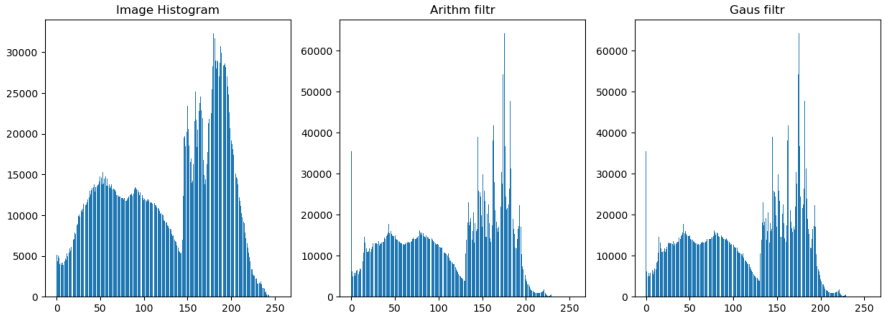



***Рисунок 5-8 Порівняння гістограм***

Аналіз зображень та відповідних гістограм розмитих (згладжених) зображень показує, що фільтр Гауса менш «пошкоджує» контури силуетів об’єктів на зображенні.

# 6. Лабораторна робота №5. Просторова обробка зображень. Лінійна фільтрація. Посилення різкості.

**Практична робота №5 орієнтована** на подальше оволодіння студентами базовими алгоритмами лінійної фільтрації зображень.

**Мета лабораторної роботи:**

- Засвоїти математичне обґрунтування побудови фільтру посилення різкості областей зображення.
- Засвоїти обчислення параметрів операції дискретної згортки Лапласа.
- Оволодіти методом посилення різкості зображень за допомогою фільтру Лапласа.

У разі успішного виконання лабораторної роботи студент буде здатен обирати та використовувати програмні продукти для роботи із зображеннями та виконувати типові операції обробки цифрових зображень.

Успішне засвоєння матеріалів та виконання лабораторної роботи буде сприяти формуванню у студента наступних соціальних навичок та вмінь: бажання постійно вчитися, вміння дотримуватися регламентних рамок, оцінювати строки проведення та виконання роботи, обґрунтувати той чи інший спосіб діяльності при виконанні практичних завдань. Форми прояву: прагнення вирішувати поточні проблеми самостійно, бажання пробувати щось нове, йти на розумний ризик при прийнятті рішень, застосовувати елементи самоконтролю і самооцінки при виконанні роботи, вміння планувати свій час.

## 6.1 Завдання до лабораторної роботи № 5.

В процесі виконання лабораторної роботи студент повинен створити програмний додаток, що виконує наступні дії:

1. Зчитує оригінальне зображення у відповідності за варіантом завдання (номер зображення дивись табл. 5). Формує ахроматичне зображення з оригінального. Формує гістограму отриманого ахроматичного зображення. На екрані дисплею відображає оригінальне зображення, відповідне ахроматичне зображення та його гістограму.

2. Розраховує коефіцієнти ядра фільтру Лапласа для визначеного у таблиці варіантів (Табл. 5) значення параметра $\boldsymbol{b}$.

3. За допомогою розрахованого ядра фільтру формує ахроматичне зображення з підсиленою різкістю. Формує гістограму отриманого ахроматичного зображення. На екрані дисплею відображає оригінальне зображення, зображення з підвищеною різкістю та порівняння їх гістограм.

4. Розраховує коефіцієнти ядра LoG фільтру визначеного у таблиці варіантів (Табл. 5) значення параметра $\boldsymbol{a}$.

5. За допомогою розрахованого ядра LoG фільтру формує ахроматичне зображення з підсиленою різкістю. Формує гістограму отриманого ахроматичного зображення. На екрані дисплею відображає оригінальне зображення, зображення з підвищеною за LoG фільтром різкістю та порівняння їх гістограм.

За результатами роботи програми необхідно порівняти отримані зображення і зробити висновки щодо якості підсилення різкості.

Таблиця 5

Варіанти завдання до лабораторної роботи 5

| Вар. № | № зображення | b | a |
|---|---|---|---|
| 1 | 21 | 1 | 0,1 |
| 2 | 22 | 0,9 | 0,2 |
| 3 | 23 | 0,8 | 0,3 |
| 4 | 24 | 0,7 | 0,4 |
| 5 | 25 | 0,6 | 0,5 |
| 6 | 26 | 0,5 | 0,6 |
| 7 | 27 | 0,4 | 0,7 |
| 8 | 28 | 1,1 | 0,8 |
| 9 | 29 | 1,2 | 0,9 |
| 10 | 30 | 1,3 | 1,0 |
| 11 | 19 | 1,2 | 0,9 |
| 12 | 18 | 1,1 | 0,8 |
| 13 | 17 | 1,0 | 0,7 |
| 14 | 16 | 0,7 | 0,6 |
| 16 | 15 | 0,6 | 0,5 |
| 17 | 14 | 0,5 | 0,4 |
| 18 | 13 | 0,4 | 0,3 |
| 19 | 12 | 1,1 | 0,2 |
| 20 | 11 | 1,2 | 0,1 |

## 6.2 Підготовка до лабораторної роботи №5

### 6.2.1 Діскрений фільтр Лапласа

Підвищення чіткості ахроматичних зображень відчувається при візуальному підкресленні меж фрагментів зображення, які мають однакову (або мало змінну) яскравість. Тобто необхідно визначати пікселі, в околицях яких яскравість суттєво змінюється та, відповідно, контур фрагментів.

Більшість методів виділення контурних ліній, що відокремлюють сусідні фрагменти, базується на обчисленні часткових похідних від функції яскравості. На рисунку 6.1 наведена одномірна ілюстрація визначення границі фрагменту зображення. Розглянемо зміну яскравості вдовж деякої незалежної змінної (рядок пікселів). В околі точки $\bar{x}$ яскравість $I(x)$ змінюється, що визначає межу між темною та світлою частинами зображення. Перша похідна в цій точці набуває свого максимуму, при цьому друга похідна в лівій частині околу $\bar{x}$ приймає позитивні значення , а в правій – негативні. А в $\bar{x}$ набуває нульового значення. Якщо з функції $I(x)$ **відняти** значення другої похідної з деяким ваговим коефіцієнтом ***b***, результуюча крива

$$\check{I}(x) = I(x) - b * \frac{\partial^2 I(x)}{\partial x^2}$$

буде мати більш «круті» фронти, що візуально буде сприйматися як підкреслення межі в $\bar{x}$.

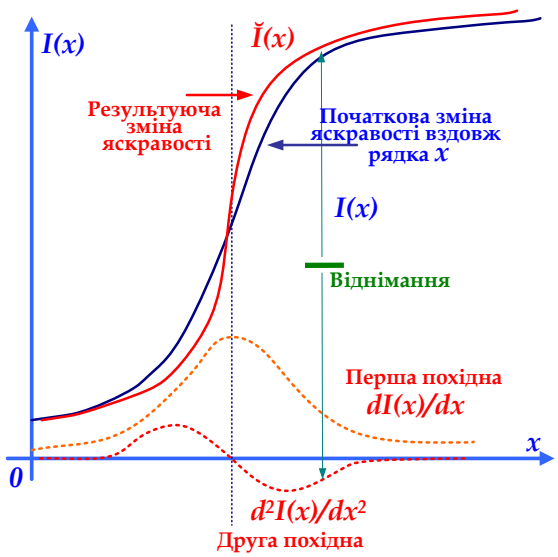



***Рисунок 6-1 Ілюстрація підкреслення межі фрагменту зображення***

Оскільки яскравість зображення є функцією двох змінних (**x, y**), градієнт функції яскравості в кожній точці визначається як двовимірний вектор:

$$G[I(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix}, \qquad G_x = \frac{\partial I(x,y)}{\partial x},\ G_y = \frac{\partial I(x,y)}{\partial y},$$

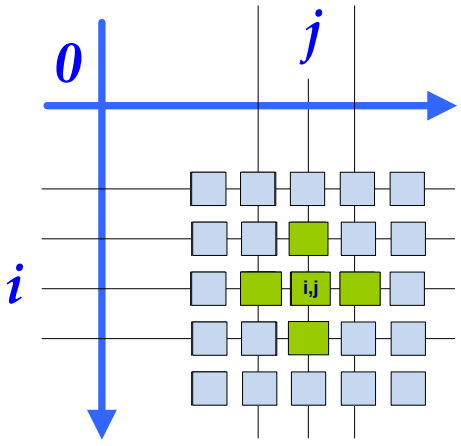
а друга похідна може бути представлена як матриця

$$\Delta[I(x,y)] = \begin{bmatrix} \frac{\partial^2 I(x,y)}{\partial x^2} & \frac{\partial^2 I(x,y)}{\partial x \partial y} \\ \frac{\partial^2 I(x,y)}{\partial y \partial x} & \frac{\partial^2 I(x,y)}{\partial y^2} \end{bmatrix} \qquad (6.1)$$

При обробці зображень використовуються деяка дискретна апроксимація (6.1) за допомогою скінчених різниць, що визначає так званий дискретний Лапласіан. При використанні найпростішого п'яти точкового шаблону (рис. 6.2), обчислення другої похідної для ***i , j*** пікселю можна виконати за виразом

$$\Delta[I(i,j)] = I(i-1,j) + I(i+1,j) + I(i,j-1) + I(i,j+1) - 4\,I(i,j), \qquad (6.2)$$

що відповідає ядру фільтра з околом радіусом 1 (розмір 3 X 3):




| | | |
|---|---|---|
| ***0*** | ***1*** | 0 |
| ***1*** | ***-4*** | ***1*** |
| ***0*** | ***1*** | 0 |

***Рисунок 6-2 П'яти точковий шаблон***

Зверніть увагу, що дана апроксимація не включає діагональні пікселі, що приводить до суттєвого посилення шуму на зображенні. Для зменшення цього явища необхідна більш

стала та ізотропна апроксимація, що включає діагоналі (дев'яти точковий шаблон), наприклад:

| | | |
|---|---|---|
| ***0,25*** | ***0,5*** | ***0,25*** |
| ***0,5*** | ***-3*** | ***0,5*** |
| ***0,25*** | ***0,5*** | ***0,25*** |

Загалом, з урахуванням знаку та значення ваги **b** розрахунок коефіцієнтів ядра дискретного лапласіана розміром 3 X 3 рекомендовано розраховувати за виразами:

| п'яти точковий шаблон (ядро #1) | | | дев'яти точковий шаблон (ядро #2) | | | дев'яти точковий шаблон (ядро #3) | | | (6.3) |
|---|---|---|---|---|---|---|---|---|---|
| ***0*** | ***-b*** | 0 | ***-b*** | ***-b*** | ***-b*** | ***b*** | ***-2b*** | ***b*** | |
| ***-b*** | ***(4b+1)*** | ***-b*** | ***-b*** | ***(8b+1)*** | ***-b*** | ***-2b*** | ***(4b+1)*** | ***-2b*** | |
| ***0*** | ***-b*** | 0 | ***-b*** | ***-b*** | ***-b*** | ***b*** | ***-2b*** | ***b*** | |

Використання фільтру Лапласа має відомий недолік. Крім підсилення різкості (виділення границь фрагментів) фільтр також суттєво підвищує шуми на зображення. Для запобігання цьому явищу використовується комбінація фільтрів Гауса та Лапласа.

### 6.2.2 LoG фільтр

Для ослаблення впливу шуму результат виділення перепадів яскравості оператором Лапласа рекомендується зробити згладжування шумів за допомогою фільтру Гауса. Послідовну обробку зображення цими двома фільтрами можна замінити обробкою комбінованим фільтром, який є результатом застосування оператора Лапласа до гауссіана і називається LoG-фільтром:

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right)e^{-\frac{x^2+y^2}{2\sigma^2}}.$$

Для найпростішого випадку (маска розміром 3×3 та σ =1) рекомендовано обчислювати коефіцієнти ядра LoG фільтру як

$$\frac{1}{1+a}\begin{bmatrix} -a & a-1 & -a \\ a-1 & a+4 & a-1 \\ -a & a-1 & -a \end{bmatrix} \quad (6.4)$$

де **a** – параметр в межах [0, 1].

## 6.3 Контрольні питання для допуску до лабораторної роботи № 5

Поясніть поняття підвищення різкості зображень (виділення контурів) та використання похідних яскравості для цього.

Надайте визначення фільтру Лапласа.

Надайте визначення LoG фільтру.

## 6.4 Методичні вказівки до виконання лабораторної роботи № 5

### 6.4.1 До завдання 5.1. Завантаження, створення ахроматичного зображення та гістограми.

Завантаження та вивід на екран дисплею оригінального зображення виконується відповідно 1.2.1 та 2.4.1. Для створення ахроматичного зображення можна використати функцію, створену за завданням 2.2. та застосувати її до завантаженого оригінального зображення. Гістограму можна обчислити та відобразити аналогічно завданню 3.3.

### 6.4.2 До завдання 5.2. Розрахунок коефіцієнтів ядра фільтру Лапласа.

При виконання завдання треба розрахувати три ядра за виразами (6.3). В наступному фрагменті наведено приклад розрахунку ядра 1 фільтру Лапласа.

```
# Визначення маски фильтру підвищення різкості
mask_row = 3
mask_clm = 3
mask_sharp_1 = np.zeros ( (mask_row, mask_clm, 1), dtype =
np.float32)
print ('MASK SHAPE', mask_sharp_1.shape, 'MASK SIZE',
mask_row*mask_clm)
# ВЗНАЧЕННЯ ВАГИ
b = .5
# пятиточкова маска (ядро 1)
mask_sharp_1[0,1] = -b
mask_sharp_1[1,0] = mask_sharp_1[1,2] = -b
mask_sharp_1[2,1] = -b
mask_sharp_1[1,1] = 4*b+1
```

### 6.4.3 До завдання 5.3. Підвищення чіткості за Лапласом та візуалізація результатів.

Дане завдання передбачає обробку одного й того ж зображення трьома варіантами фільтру Лапласу. Наступний фрагмент коду демонструє використання mask_1.

```
filtr_im_gray_1 = np.zeros((rows, clms, 3), dtype = np.float32)
pixel = np.zeros(3, dtype = np.float32)

for i in  range (1, (rows-1), 1):
    for j in  range (1, (clms-1), 1):
        pixel[:] = 0
        for l in range (mask_row):
            for k in range (mask_clm):
                pixel+=mask_sharp_1[l,k]*test_im_gray[i-(1-
                k), j-(1-l), : ]
            filtr_im_gray_1 [i, j, :] = pixel [:]
```

Візуальне порівняння результатів можна виконати за кодом

```
fig, axes = plt.subplots(1, 4, figsize=(16, 4))
ax = axes.ravel()
ax[0].set_title("Original Gray")
```

```
ax[0].imshow(test_im_gray)
ax[1].set_title('MASK 1')
ax[1].imshow(filtr_im_gray_1)
ax[2].set_title('MASK 2')
ax[2].imshow(filtr_im_gray_2)
ax[3].set_title('MASK 3')
ax[3].imshow(filtr_im_gray_3)
plt.show()
```

Приклад порівняння отриманих зображень наведено на рис. 6.3.

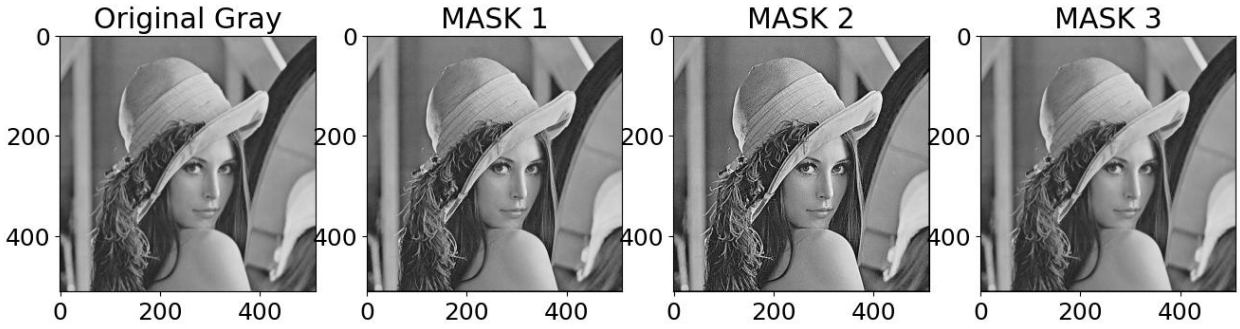



***Рисунок 6-3 Порівняння зображень перетворених варіантами фільтру Лапласа***

Також для більш наглядного відображення ефекту фільтрації рекомендовано виділити невелике вікно. На рис. 6.4 наведено частина зображення (вікно з координатами верхнього лівого кута [20, 150] та розміром 100 пікселів).

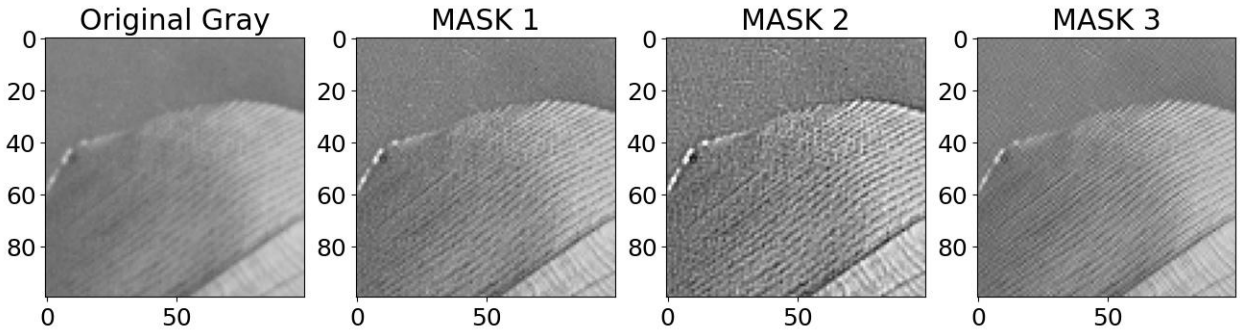



***Рисунок 6-4 Порівняння фрагменту зображень перетворених варіантами фільтру Лапласа***

Порівняння гістограм зображень можна виконати аналогічно 5.4.5.

### 6.4.4 До завдання 5.4. Розрахунок коефіцієнтів ядра LoG фільтру.

Виконується аналогічно 6.4.2 з урахуванням виразу (6.4).

### 6.4.5 До завдання 5.5. Підвищення чіткості за LoG фільтром та візуалізація результату.

Виконується аналогічно 6.4.3 для маски за виразом (6.4).

Результати порівняння вхідного зображення, фільтрованого за Лапласом з першим ядром та LoG фільтром наведені на рис. 6.5.

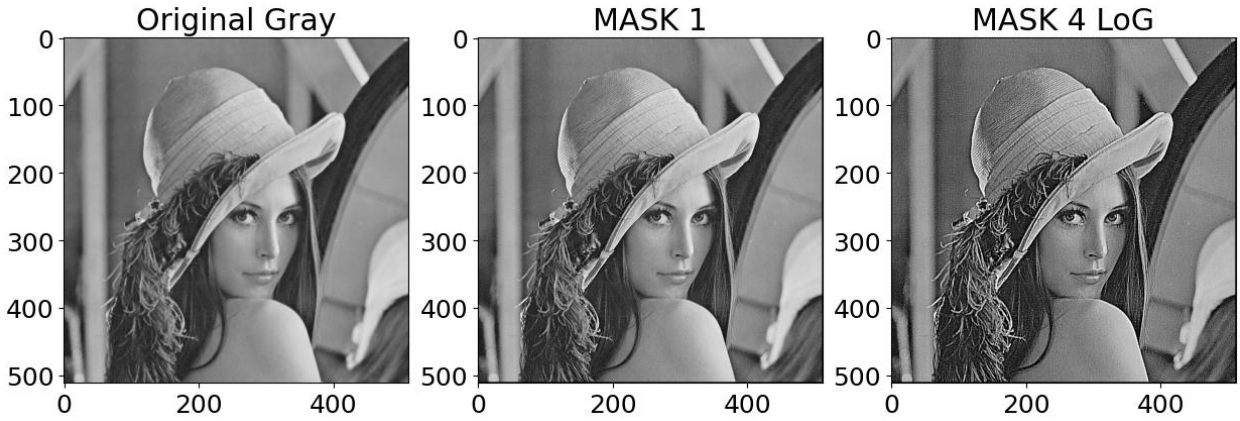



***Рисунок 6-5 Порівняння зображень фільтром Лапласа (ядро 1) та LoG фільтром***

На рис. 6.6 наведено порівняння частин зображення перетворених за фільтрами Лапласа та LoG.

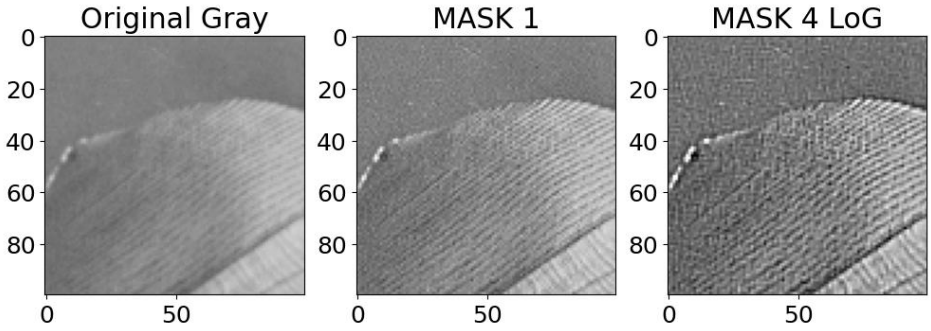



***Рисунок 6-6 Порівняння зображень фільтром Лапласа (ядро 1) та LoG фільтром***

# 7. Лабораторна робота №6. Просторова обробка зображень. Нелінійна фільтрація. Медіанні фільтри

**Практична робота №6 орієнтована** на оволодіння студентами базовими алгоритмами нелінійної фільтрації зображень.

**Мета лабораторної роботи:**

- Засвоїти математичне обґрунтування побудови медіанного фільтру усунення імпульсного шуму на зображеннях.
- Засвоїти математичне обґрунтування побудови адаптивного медіанного фільтру усунення імпульсного шуму на зображеннях.
- Оволодіти алгоритмами медіанної та адаптивної фільтрації зображень з одно полярним та біполярним шумом.

У разі успішного виконання лабораторної роботи студент буде здатен обирати та використовувати програмні продукти для роботи із зображеннями та виконувати типові операції обробки цифрових зображень.

Успішне засвоєння матеріалів та виконання лабораторної роботи буде сприяти формуванню у студента наступних соціальних навичок та вмінь: бажання постійно вчитися, вміння дотримуватися регламентних рамок, оцінювати строки проведення та виконання роботи, обґрунтувати той чи інший спосіб діяльності при виконанні практичних завдань. Форми прояву: прагнення вирішувати поточні проблеми самостійно, бажання пробувати щось нове, йти на розумний ризик при прийнятті рішень, застосовувати елементи самоконтролю і самооцінки при виконанні роботи, вміння планувати свій час.

## 7.1 Завдання до лабораторної роботи № 6.

В процесі виконання лабораторної роботи студент повинен створити програмний додаток, що виконує наступні дії:

1. Зчитує оригінальне зображення у відповідності за варіантом завдання (номер зображення дивись табл. 6). Формує копію зображення, спотвореного одно полярним імпульсним шумом. Кількість імпульсів визначено в завдання. Формує ахроматичне спотворене зображення та його гістограму.

2. Виконує фільтрацію кольорового зображення із визначеним шумом за допомогою середньоарифметичного фільтру (розмір маски див. табл. 6).

3. Формує ахроматичне зображення та його гістограму фільтрованого за пунктом 2 зображення. На екрані дисплею відображає кольорові спотворене та фільтроване зображення, ахроматичні спотворене та фільтроване зображення та їх гістограми.

4. Виконує фільтрацію зображення із **однополярним** шумом за допомогою медіанного фільтру (розмір маски, тип маски див. табл. 6).

5. Формує ахроматичне зображення та його гістограму фільтрованого за пунктом 4 зображення. На екрані дисплею відображає кольорові спотворене та фільтроване зображення, ахроматичні спотворене та фільтроване зображення та їх гістограми.

6. Формує копію оригінального зображення, спотвореного **біполярним** імпульсним шумом. Кількість імпульсів визначено в завдання. Формує ахроматичне спотворене зображення та його гістограму.

7. Виконує фільтрацію зображення з біполярним шумом за допомогою медіанного та адаптивного медіанного фільтру (розмір маски, тип маски див. табл. 6).

8. Формує ахроматичне зображення та його гістограму фільтрованих за пунктом 7 зображень. На екрані дисплею відображає кольорові спотворене та фільтроване зображення, ахроматичні спотворене та фільтровані зображення та їх гістограми.

За результатами роботи програми необхідно порівняти отримані зображення і зробити висновки щодо якості усунення імпульсного шуму.

Таблиця 6

Варіанти завдання до лабораторної роботи 6

| Вар. № | № зображення | Кількість імпульсів | Номер маски Рис. 7.1 |
|---|---|---|---|
| 1 | 17 | 1000 | 1 |
| 2 | 16 | 600 | 2 |
| 3 | 15 | 800 | 3 |
| 4 | 14 | 500 | 4 |
| 5 | 13 | 700 | 5 |
| 6 | 12 | 900 | 1 |
| 7 | 11 | 1000 | 2 |
| 8 | 7 | 600 | 3 |
| 9 | 6 | 800 | 4 |
| 10 | 5 | 500 | 5 |
| 11 | 4 | 700 | 1 |
| 12 | 3 | 900 | 2 |
| 13 | 2 | 1000 | 3 |
| 14 | 1 | 600 | 4 |
| 16 | 21 | 800 | 5 |
| 17 | 22 | 500 | 4 |
| 18 | 23 | 700 | 3 |
| 19 | 24 | 900 | 2 |
| 20 | 25 | 1000 | 1 |

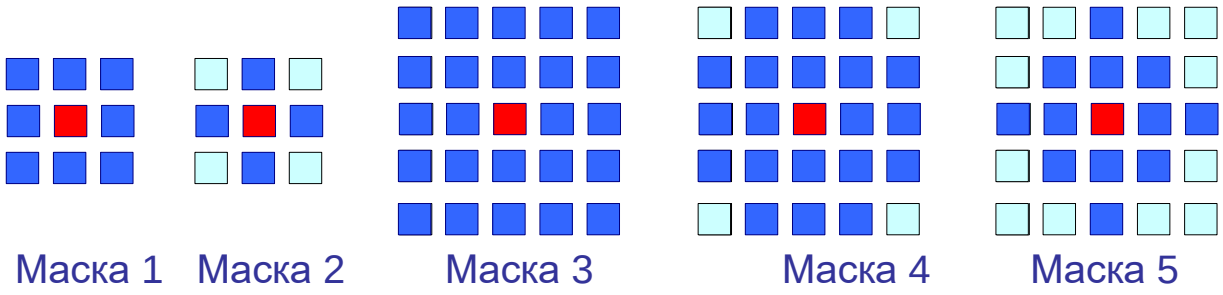



***Рисунок 7-1 Формат маски до індивідуального завдання***

## 7.2 Підготовка до лабораторної роботи №6

### 7.2.1 Медіанний фільтр

Медіанний фільтр є один з видів нелінійних цифрових фільтрів, що широко використовується в цифровій обробці зображень для зменшення рівня шуму.

У загальному вигляді медіанний фільтр можна описати наступним чином. Колір (яскравість для ахроматичних зображень) поточного пікселя замінюється медіаною кольорів всіх пікселів зображення, що потрапили у вікно фільтру. Тобто колір (яскравість) поточного пікселя в фільтрованому зображенні дорівнює кольору середньому по порядку пікселю, що впорядковані по зростанню (спаданню). Формально для вікна типу «хрест» розміром 3 X 3 (Маска 2 на рис. 7.1) $L'_{i,j}$ - яскравість ***i,j*** – го пікселя фільтрованого зображення обчислюється як

$$L'_{i,j} = \boldsymbol{med}(L_{i,j}, L_{i-1,j}, L_{i+1,j}, L_{i,j-1}, L_{i,j+1}) \qquad (7.1)$$

Приклад розрахунку за 7.1 наведено на рис. 7.2.

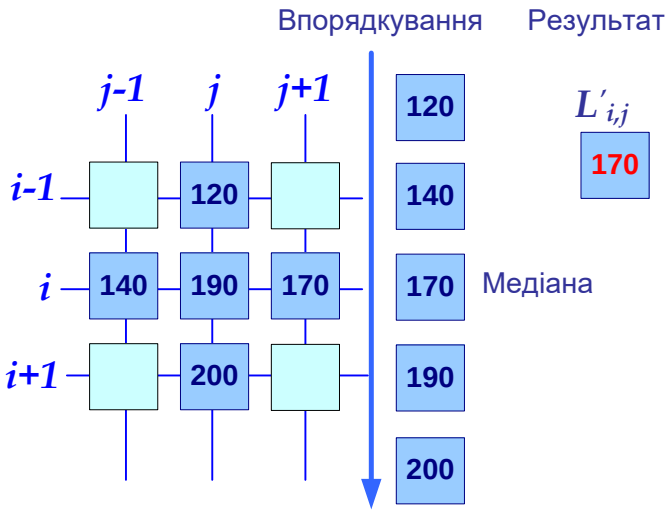



***Рисунок 7-2 Визначення яскравості пікселя за медіанним фільтром***

Вважається, що медіанна фільтрація є ефективною процедурою обробки зображень, що спотворені одно полярними імпульсними завадами. Фільтр видаляє із зображення фрагменти з розмірами, меншими ніж половина розміру вікна фільтра, і мало спотворює або майже не спотворює інші ділянки оригінального зображення.

### 7.2.2 Адаптивний медіанний фільтр

Для видалення біполярних імпульсних завад та згладжування шумів інших типів використовується адаптивний медіанний фільтр. В роботі розглянемо спрощену версію такого фільтру. В цьому випадку, розрахунок відгуку адаптивного медіанного фільтру можна визначити наступним алгоритмом.

Для визначеного вікна фільтру обчислюються (формули надані для вікна «хрест» розміром 3 X 3):

$$Lmin = min(L_{i,j}, L_{i-1,j}, L_{i+1,j}, L_{i,j-1}, L_{i,j+1}) \quad (7.2)$$

$$Lmed = med(L_{i,j}, L_{i-1,j}, L_{i+1,j}, L_{i,j-1}, L_{i,j+1}) \quad (7.3)$$

$$Lmax = max(L_{i,j}, L_{i-1,j}, L_{i+1,j}, L_{i,j-1}, L_{i,j+1}) \quad (7.4)$$

Значення $L'_{i,j}$ яскравості ***i,j*** – го пікселя фільтрованого зображення визначається як

$$L'_{i,j} = \begin{cases} L_{i,j}, & Lmin < L_{i,j} < Lmax \\ Lmed, & L_{i,j} = Lmin \\ Lmed, & L_{i,j} = Lmax \end{cases} \quad (7.5)$$

Приклад розрахунку за 7.2 – 7.5 наведено на рис. 7.3.

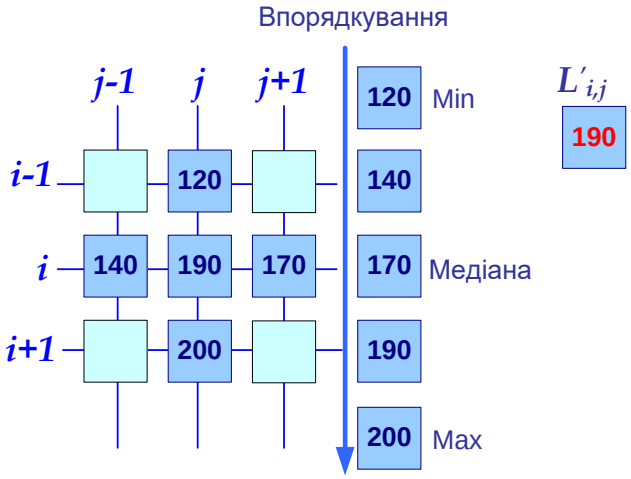



***Рисунок 7-3 Визначення яскравості пікселя за адаптивним медіанним фільтром***

Слід зауважити, що в повній версії адаптованого медіанного фільтру також збільшується розмір вікна, якщо $\boldsymbol{Lmed = Lmax}$ або $\boldsymbol{Lmed = Lmin}$.

Адаптивний медіанний фільтр допомагає «знищити» імпульсні завади та зберігає деталі в частинах зображення що не спотворені імпульсними завадами.

## 7.3 Контрольні питання для допуску до лабораторної роботи № 6

Поясніть алгоритм роботи медіанного фільтру та його властивості щодо усунення завад на зображеннях.

Поясніть алгоритм роботи адаптивного медіанного фільтру та його властивості щодо усунення завад на зображеннях.

## 7.4 Методичні вказівки до виконання лабораторної роботи № 6

Пункти 1, 3, 5, 8 завдання виконуються аналогічно попереднім практичним роботам.

Пункт 2 завдання (середньоарифметичний фільтр) повторює завдання 2 роботи 5 з урахування розміру та форми маски.

### 7.4.1 До завдань 6.4 та 6.6. Додавання однополярної та біполярної імпульсних завад.

Для спотворення оригінального зображення випадковими імпульсними завадами по-перше необхідно завантажити модуль генерації випадкових чисел

```
from numpy.random import Generator, MT19937
```

Надалі за допомогою цього генератора створюється необхідна кількість «випадкових» пікселів, які встановлюються до білого «соль» або чорного «перець».

```
rg = Generator(MT19937(12345))
impuls_num = 512  # КІЛЬКІСТЬ СПОТВРЕНИХ ПІКСЕЛІВ
test_im_noise = test_im.copy()
for i in range(1,impuls_num):
    test_im_noise[np.int32(rg.random()*rows_num), # сіль
                  np.int32(rg.random()*clms_num),:] = 0
    test_im_noise[np.int32(rg.random()*rows_num), # перець
                  np.int32(rg.random()*clms_num),:] = 255
```

Приклад фрагменту спотвореного однопоярною завадою зображення наведено на рис. 7.4.

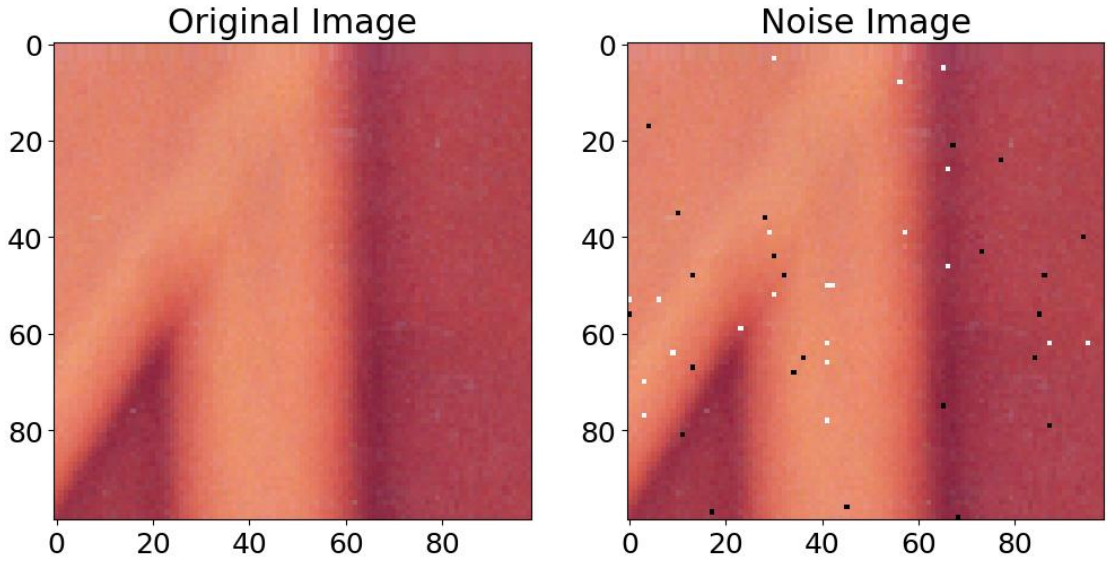



*Рисунок 7-4 Зображення з однополярними імпульсами*

Біполярні імпульсні завади можна додати наступним чином

```
impuls_num = 512   # КІЛЬКІСТЬ СПОТВРЕНИХ ПІКСЕЛІВ / 2
test_im_noise = test_im.copy()
for i in range(1,impuls_num):
    i_n=np.int32(rg.random()*(rows_num-1))
    j_n=np.int32(rg.random()*(clms_num-1))
    test_im_noise[i_n,j_n,:] = 255
    test_im_noise[i_n+1,j_n+1,:] = 0
    test_im_noise[i_n-1,j_n-1,:] = 0
```

Приклад фрагменту спотвореного біполярною завадою зображення наведено на рис. 7.

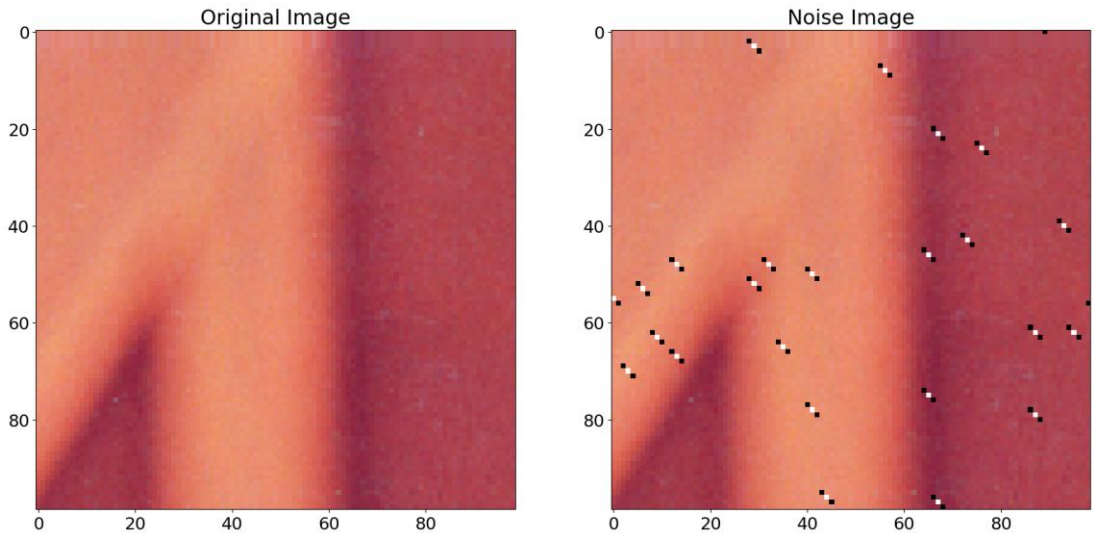



*Рисунок 7-5 Зображення з біполярними імпульсами*

### 7.4.2 До завдання 6.4. Медіанна фільтрація

Дане завдання передбачає обробку спотвореного зображення медіанним фільтром із заданою маскою. Нижче наведено відповідний фрагмент коду фільтрації з використанням mask_2.

```
# Визначення файлу фільтрованого зображення
filtr_im_noise=np.zeros((rows_num,clms_num,3),dtype=np.uint8)
# Визначення масиву пікселів для обчислення медіани
pixels = np.zeros((5, 3), dtype=np.uint8)
#Піксель із значеннями кольорів медіани
pix = np.zeros(3, dtype=np.uint8)

for i in  range (1, (rows_num -1), 1):
    for j in  range (1, (clms_num-1), 1):
        # Обираємо пікселі з «хреста»
        pixels[0,:] =  filtr_im [i,j,:] #центр
        pixels[1,:] =  filtr_im [i-1,j,:] #зліва
        pixels[2,:] =  filtr_im [i+1,j,:] #справа
        pixels[3,:] =  filtr_im [i,j-1,:] #вище
        pixels[4,:] =  filtr_im [i,j+1,:] #нижче
        # Обчислюємо медіани кольорів R,G,B
        pix[0] = np.median(pixels[:,0])
        pix[1] = np.median(pixels[:,1])
        pix[2] = np.median(pixels[:,2])
        filtr_im_noise[i,j,:] = pix[:]
```

Приклад фрагменту зображення, обробленого медіанним фільтром наведено на рис. 7.6.

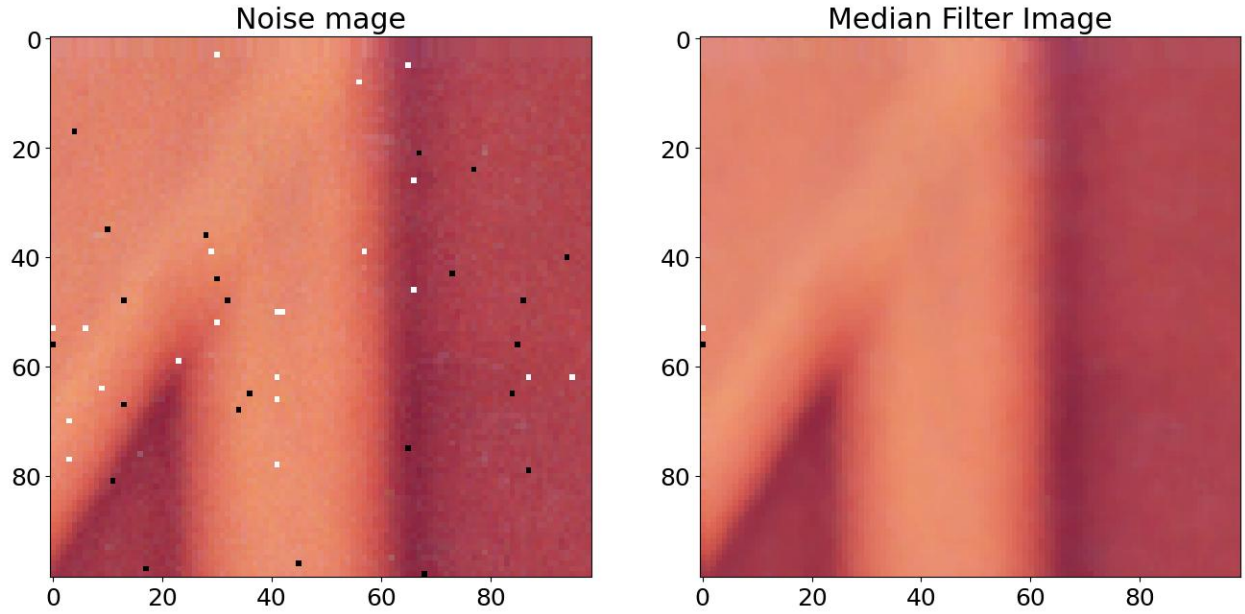



***Рисунок 7-6 Порівняння фрагментів спотвореного однополярним шумом та обробленого медіанним фільтром зображень***

### 7.4.3 До завдання 6.1. Адаптивна медіанна фільтрація

Нижче надано фрагмент коду адаптивної медіанної фільтрації з використанням mask_2.

```
filtr_im_noise=np.zeros((rows_num,clms_num,3),dtype=np.uint8)
pixels = np.zeros((5, 3), dtype=np.uint8)
pix_med = np.zeros(3, dtype=np.uint8)
#Піксель із значеннями кольорів медіани
pix_med=np.zeros(3,dtype=np.uint8)
#Піксель із значеннями кольорів мінімуму
pix_min=np.zeros(3,dtype=np.uint8)
#Піксель із значеннями кольорів максимуму
pix_max=np.zeros(3, dtype=np.uint8) #

for i in  range (1, (rows_num -1), 1):
    for j in  range (1, (clms_num-1), 1):
        pixels[0,:] = test_im_noise[i, j,   :]
        pixels[1,:] = test_im_noise[i-1,j,  :]
        pixels[2,:] = test_im_noise[i+1,j,  :]
        pixels[3,:] = test_im_noise[i  ,j-1,:]
        pixels[4,:] = test_im_noise[i  ,j+1,:]
        for k in range (3):
            pix_med[k] = np.median(pixels[:,k])
            pix_min[k] = np.min(pixels[:,k])
            pix_max[k] = np.max(pixels[:,k])
        filtr_im_noise[i,j,:] = test_im_noise[i,j,:]
        for q in range (0,3):
            if filtr_im_noise[i,j,q]==pix_min[q]:
                filtr_im_noise[i,j,q] = pix_med[q]
            if filtr_im_noise[i,j,q]==pix_max[q]:
                filtr_im_noise[i,j,q] = pix_med[q]
```

Приклад фрагменту зображення, обробленого адаптивним  медіанним фільтром наведено на рис. 7.7

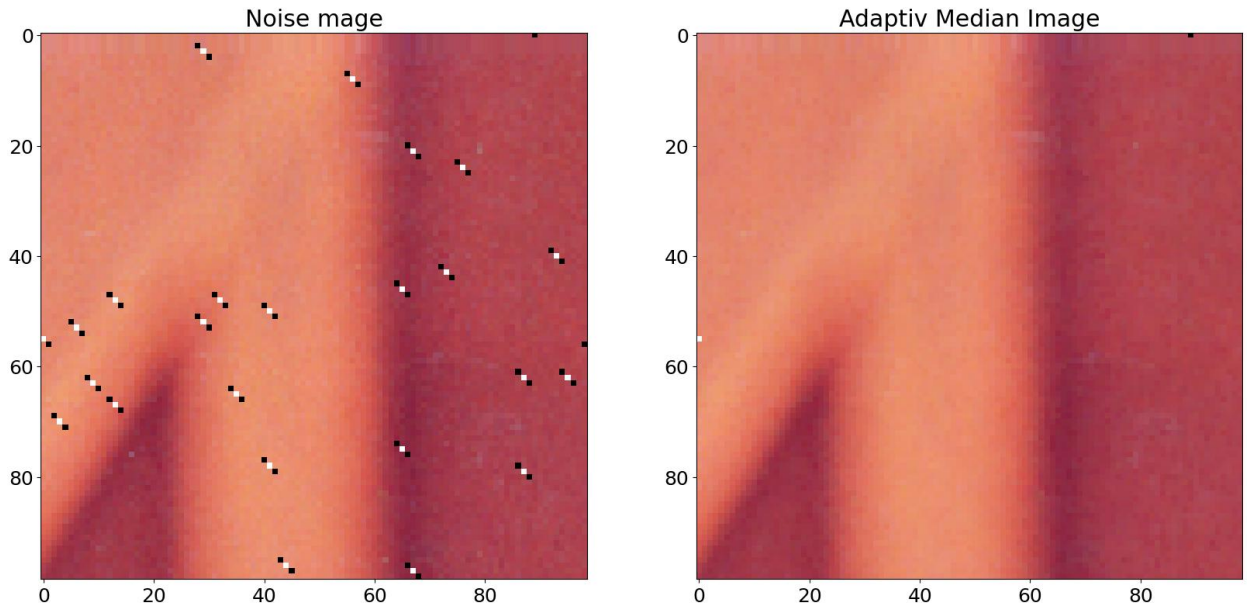



***Рисунок 7-7 Порівняння фрагментів спотвореного біполярним шумом  та обробленого адаптивним медіанним фільтром зображень***

# 8. Лабораторна робота №7. Просторова обробка зображень. Нелінійна фільтрація. Градієнтні фільтри.

**Практична робота №7 орієнтована** на подальше оволодіння студентами базовими алгоритмами нелінійної фільтрації зображень.

**Мета лабораторної роботи:**

- Засвоїти математичне обґрунтування побудови градієнтних фільтрів визначення перепадів яскравості та виявлення кордонів об’єктів на зображеннях.
- Оволодіти алгоритмом побудови та використання градієнтного оператора Робертса виявлення кордонів.
- Оволодіти алгоритмом побудови та використання градієнтного оператора Превітта виявлення кордонів
- Оволодіти алгоритмом побудови та використання градієнтного оператора Собеля виявлення кордонів.

У разі успішного виконання лабораторної роботи студент буде здатен обирати та використовувати програмні продукти для роботи із зображеннями та виконувати типові операції обробки цифрових зображень.

Успішне засвоєння матеріалів та виконання лабораторної роботи буде сприяти формуванню у студента наступних соціальних навичок та вмінь: бажання постійно вчитися, вміння дотримуватися регламентних рамок, оцінювати строки проведення та виконання роботи, обґрунтувати той чи інший спосіб діяльності при виконанні практичних завдань. Форми прояву: прагнення вирішувати поточні проблеми самостійно, бажання пробувати щось нове, йти на розумний ризик при прийнятті рішень, застосовувати елементи самоконтролю і самооцінки при виконанні роботи, вміння планувати свій час.

## 8.1 Завдання до лабораторної роботи № 7.

В процесі виконання лабораторної роботи студент повинен створити програмний додаток, що виконує наступні дії:

1. Зчитує оригінальне зображення у відповідності за варіантом завдання (номер зображення дивись табл. 7). Формує відповідне ахроматичне зображення.

2. Виконує операцію виявлення кордонів об’єктів на зображенні за допомогою градієнтного оператора Робертса. Відображає оригінальне ахроматичне зображення та перетворене оператором Робертса зображення.

3. Виконує операцію виявлення кордонів об’єктів на зображенні за допомогою градієнтного оператора Превітта. Відображає оригінальне ахроматичне зображення та перетворене оператором Превіта зображення.

4. Виконує операцію виявлення кордонів об’єктів на зображенні за допомогою градієнтного оператора Собеля. Відображає оригінальне ахроматичне зображення та перетворене оператором Собеля зображення.

5. На екрані дисплею відображає три перетворених зображення, що отримані за пунктами 2 - 4 завдання.

За результатами роботи програми необхідно порівняти отримані зображення і зробити висновки щодо якості виявлення кордонів об’єктів на зображенні.

Таблиця 7

Варіанти завдання до лабораторної роботи 7

| Вар. № | № зображення |
|---|---|
| 1 | 19 |
| 2 | 20 |
| 3 | 13 |
| 4 | 12 |
| 5 | 11 |
| 6 | 20 |
| 7 | 21 |
| 8 | 22 |
| 9 | 23 |
| 10 | 24 |
| 11 | 25 |
| 12 | 11 |
| 13 | 12 |
| 14 | 13 |
| 16 | 14 |
| 17 | 15 |
| 18 | 16 |
| 19 | 17 |
| 20 | 18 |

## 8.2 Підготовка до лабораторної роботи №7

Одним з найбільш простих та поширених підходів щодо виявлення кордонів є просторове диференціювання функції яскравості, що виконується за деякою дискретною апроксимацією просторового градієнту функції яскравості.

Як визначено в 6.2 яскравість ***I*** довільного зображення є функцією двох змінних (***x, y***), а градієнт функції яскравості в кожній точці визначається як двовимірний вектор:

$$G[I(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix}, \qquad G_x = \frac{\partial I(x,y)}{\partial x},\ G_y = \frac{\partial I(x,y)}{\partial y}. \tag{8.1}$$

Для визначення суттєвого перепаду яскравості на кордоні деякого об'єкту зображення визначальну роль грає модуль (довжина) цього вектору, яка визначає швидкість зміни яскравості. Обчислити довжину градієнту можна за виразом

$$|G| = \sqrt{G_x^2 + G_y^2}. \tag{8.2}$$

Необхідно також відмітити, що для обчислення градієнту не обов'язково брати похідні по ***x*** та ***y.*** Можна обрати два довільних взаємо перпендикулярних напрямки.

Для позначення контуру на зображенні обчислений за (8.2) модуль градієнту порівнюється з деяким порогом (threshold) для виконання наступної бінаризації зображення. Будемо використовувати білий піксель $L'_{i,j} = [255,255,255]$ для позначення кордонів об'єкту на зображенні.

### 8.2.1 Фільтр (оператор) Робертса

Дискретний оператор Робертса використовує маску розміром 2 X 2 для приблизного обчислення компонент градієнта в двох взаємо перпендикулярних напрямках (повернутих щодо системи координат ***0xy*** на 45 градусів). Ядра фільтру Робертса мають вигляд

| | |
|---|---|
| ***-1*** | ***0*** |
| ***0*** | ***1*** |

| | |
|---|---|
| ***0*** | ***-1*** |
| ***1*** | ***0*** |

Для кожного ***i, j*** пікселя зображення обчислюються компоненти градієнту

$$G_1 = I(i+1,j) - I(i,j+1), \qquad G_2 = I(i+1,j) - I(i,j+1)$$

та модуль градієнту аналогічно (8.2).

Яскравість пікселя перетвореного зображення визначається відповідно співвідношення

$$L'_{i,j} = \begin{cases} 0, |G| \le \gamma \\ 255, |G| > \gamma \end{cases}, \tag{8.3}$$

де γ – поріг бінарізації.

### 8.2.2 Фільтр (оператор) Превітта

Дискретний оператор Превітта Робертса використовує маску розміром 3 X 3 для приблизного обчислення компонент градієнта в двох взаємо перпендикулярних напрямках (вдовж координатних вісей). Ядра фільтру Превітта мають вигляд

| | | |
|---|---|---|
| ***-1*** | ***0*** | ***1*** |
| ***-1*** | ***0*** | ***1*** |
| ***-1*** | ***0*** | ***1*** |

| | | |
|---|---|---|
| ***-1*** | ***-1*** | ***-1*** |
| ***0*** | ***0*** | ***0*** |
| ***1*** | ***1*** | ***1*** |

Для кожного ***i, j*** пікселя зображення обчислюються компоненти градієнту

$$G_x = I(i-1,j+1) + I(i,j+1) + I(i+1,j+1) - I(i-1,j-1) - I(i,j-1) - I(i+1,j-1), \tag{8.4}$$

$$G_y = I(i+1,j-1) + I(i+1,j) + I(i+1,j+1) - I(i-1,j-1) - I(i-1,j) - I(i-1,j+1)$$

та модуль градієнту за (8.2). Надалі, за (8.3) визначаються пікселі, що належать кордонам об'єктів на зображенні.

### 8.2.3 Фільтр (оператор) Собеля

Дискретний оператор Собеля використовує маску розміром 3 X 3 для приблизного обчислення компонент градієнта в двох взаємо перпендикулярних напрямках (вдовж координатних вісей). Ядра фільтру Собеля мають вигляд

| | | |
|---|---|---|
| ***-1*** | ***0*** | ***1*** |
| ***-2*** | ***0*** | ***2*** |

| | | |
|---|---|---|
| ***-1*** | ***-2*** | ***-1*** |
| ***0*** | ***0*** | ***0*** |

| | | |
|---|---|---|
| *-1* | *0* | *1* |

| | | |
|---|---|---|
| *1* | *2* | *1* |

Слід відмітити, що оператор Собеля збільшує вплив яскравості сусідніх пікселів строго по вертикалі та горизонталі. Це дає змогу декілька зменшити шуми на фільтрованому зображені із визначеними контурами.

Дещо змінюються співвідношення для обчислення компоненти градієнту

$$G_x = I(i-1,j+1) + 2I(i,j+1) + I(i+1,j+1) - I(i-1,j-1) - 2I(i,j-1) - I(i+1,j-1), \quad (8.5)$$

$$G_y = I(i+1,j-1) + 2I(i+1,j) + I(i+1,j+1) - I(i-1,j-1) - 2I(i-1,j) - I(i-1,j+1).$$

Надалі модуль градієнту та визначення пікселі, що належить кордонам об’єктів виконується відповідно (8.2), (8.3).

## 8.3 Контрольні питання для допуску до лабораторної роботи № 7

Поясніть алгоритм побудови та використання градієнтного оператора Робертса для виявлення кордонів об’єктів на зображенні.

Поясніть алгоритм побудови та використання градієнтного оператора Превітта для виявлення кордонів об’єктів на зображенні.

Поясніть алгоритм побудови та використання градієнтного оператора Собеля для виявлення кордонів об’єктів на зображенні.

## 8.4 Методичні вказівки до виконання лабораторної роботи № 7

Пункт 1 завдання виконуються аналогічно попереднім практичним роботам.

### 8.4.1 До завдання 7.2. Оператор Робертса

Дане завдання передбачає обробку ахроматичного зображення фільтром з оператором Робертса та його бінарізацію. Нижче наведено відповідний фрагмент коду фільтрації. Оригінальне ахроматичне зображення надано файлом **test_im_gray.**

```
# Визначення параметрів маски
L = 2; mask_row = L; mask_clm = L
Gamma = 50 # Поріг бінаризації
# Перетворення файлу зображення до формату int32
test_im_gray_flt = np.int32(test_im_gray)
# Визначення файлу перетвореного зображення
filtr_im=np.zeros((rows_num,clms_num,3), dtype=np.uint32)

for i in  range (1, (rows_num-1), 1):
    for j in  range (1, (clms_num-1), 1):
        GrX=test_im_gray_flt[i+1,j+1,0]- \
            test_im_gray_flt[i,j,0]
        GrY=test_im_gray_flt[i+1,j,0]- \
             test_im_gray_flt[i,j+1,0]
        Gr = np.sqrt(GrX*GrX+GrY*GrY)
        if Gr >= Gamma: filtr_im_g1 [i,j,:] = 255
```

Зверніть увагу на перетворення файлів зображення до формату **int32** що необхідно для усунення можливості переповнення при розрахунку компонент вектору градієнту. Відтворення оригінального та перетвореного зображень відбувається звичайним чином. На рис.8.1 наведено приклад порівняння оригінального зображення та зображення з визначеними кордонами за допомогою фільтру Робертса.

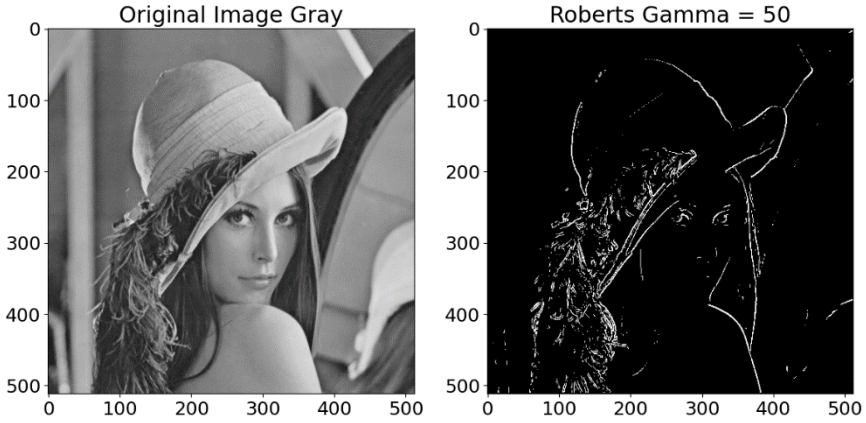



***Рисунок 8-1 Результат фільтрації за Робертсом (γ=50)***

### 8.4.2 До завдання 7.3. Оператор Превітта

Дане завдання передбачає обробку ахроматичного зображення **test_im_gray.**фільтром з оператором Превітта та його подальшу бінарізацію. Нижче наведено відповідний фрагмент коду.

```
# Визначення параметрів маски
L = 3; mask_row = L; mask_clm = L
## Визначення файлу перевореного зображення
Gamma = 50
test_im_gray_flt = np.int32(test_im_gray)
filtr_im_ = np.zeros((rows_num,clms_num,3), dtype=np.uint32)

for i in  range (1, (rows_num-1), 1):
    for j in  range (1, (clms_num-1), 1):
        GrX=test_im_gray_flt[i-1,j+1,0]\
       +test_im_gray_flt[i,j+1,0]+test_im_gray_flt[i+1,j+1,0]\
       -test_im_gray_flt[i-1,j-1,0]-test_im_gray_flt[i,j-1,0]\
       -test_im_gray_flt[i+1,j-1,0]
        GrY = test_im_gray_flt[i+1,j-1,0]\
       +test_im_gray_flt[i+1,j,0]+test_im_gray_flt[i+1,j+1,0]\
       -test_im_gray_flt[i-1,j-1,0]-test_im_gray_flt[i-1,j,0]\
       -test_im_gray_flt[i-1,j+1,0]
        Gr = np.sqrt(GrX*GrX+GrY*GrY)
        if Gr >= Gamma: filtr_im_g1 [i,j,:] = 255
```

На рис.8.2 наведено приклад порівняння оригінального зображення та зображення з визначеними кордонами за допомогою фільтру Превітта.

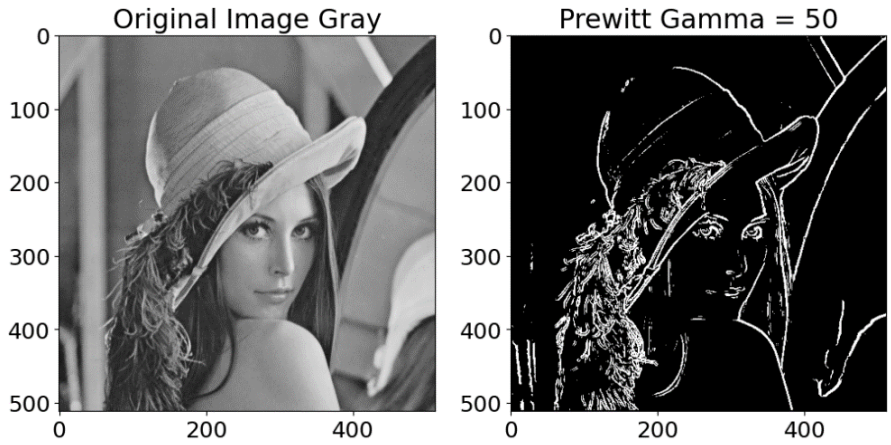



***Рисунок 8-2 Результат фільтрації за Превіттом (γ=50)***

### 8.4.3 До завдання 7.4. Оператор Собеля

Виконується аналогічно завданню 3. При цьому враховуються зміни у співвідношеннях обчислення компонент градієнту відповідно (8.5). Нижче наведено відповідний фрагмент коду обчислення компонент градієнту (тіло циклу).

```
GrX=test_im_gray_flt[i-1,j+1,0]\
+2*est_im_gray_flt[i,j+1,0]+test_im_gray_flt[i+1,j+1,0]\
-test_im_gray_flt[i-1,j-1,0]-2*test_im_gray_flt[i,j-1,0]\
-test_im_gray_flt[i+1,j-1,0]

GrY = test_im_gray_flt[i+1,j-1,0]\
+2*test_im_gray_flt[i+1,j,0]+test_im_gray_flt[i+1,j+1,0]\
-test_im_gray_flt[i-1,j-1,0]-2*test_im_gray_flt[i-1,j,0]\
-test_im_gray_flt[i-1,j+1,0]
```

На рис.8.3 наведено приклад порівняння оригінального зображення та зображення з визначеними кордонами за допомогою фільтру Собеля.

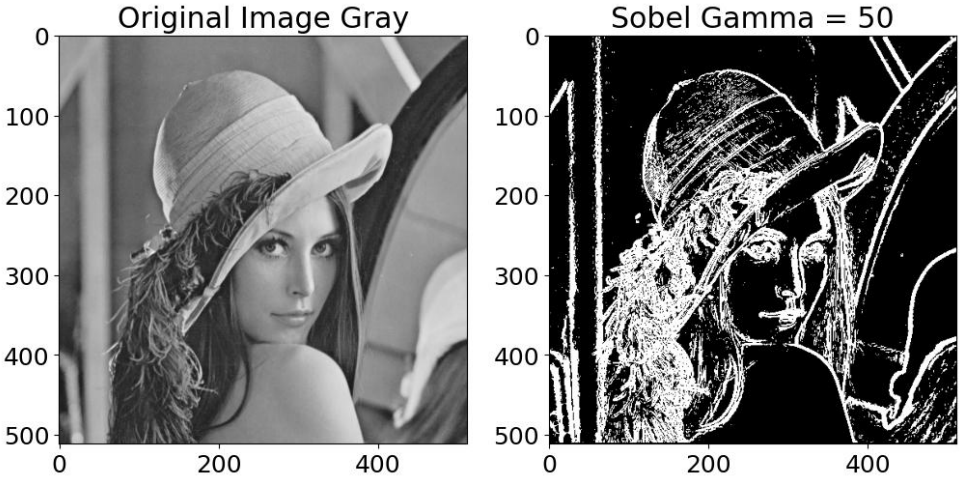



***Рисунок 8-3 Результат фільтрації за Собелем (γ=50)***

## Список рекомендованої літератури

1. Нейроподібні методи, алгоритми та структури обробки зображень у реальному часі : монографія / Ю. М. Рашкевич, Р. О. Ткаченко, І. Г. Цмоць та ін. ; М-во освіти і науки України, Нац. ун-т «Львів. політехніка». — Львів : Вид-во Львів. політехніки, 2014. — 256 с. : іл.

2. Творошенко, І. С. Конспект лекцій з дисципліни «Цифрова обробка зображень» (для студентів 5 курсу денної та заочної форм навчання спеціальності 7.08010105 – Геоінформаційні системи та технології) / І. С. Творошенко ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2015. – 75 с.

3. Методичні вказівки для виконання практичних та самостійної робіт з навчальної дисципліни «Цифрова обробка зображень» (для студентів 4 курсу денної форми навчання напряму 6.080101 – Геодезія, картографія та землеустрій) / Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова ; уклад. І. С. Творошенко. – Харків : ХНУМГ ім. О. М. Бекетова, 2016. – 55 с.

4. Журавчак, Л. М. Програмування комп'ютерної графіки та мультимедійні засоби : навч. посіб. / Л. М. Журавчак, О. М. Левченко. – Львів : Вид-во Львівської політехніки, 2019. – 274 с. : іл.

5. Цифрова обробка зображень Система дистанційного навчання “MOODLE” [Електронний ресурс]. – Режим доступу : *https://eln.stu.cn.ua/course/view.php?id=4352* . - (дата звернення: 03.12.2022).

6. GNU Octave [Електронний ресурс]. – Режим доступу : *https://www.gnu.org/software/octave/index* . - (дата звернення: 03.12.2022).

# РЕКОМЕНДОВАНІ ПРОГРАМНІ ПРОДУКТИ

- **Python** [Електронний ресурс]. – Режим доступу: *https://www.python.org/downloads/.*
- **IDE Anaconda** [Електронний ресурс]. – Режим доступу: *https://anaconda.org/*
- **Microsoft Visual Studio** [Електронний ресурс.] – Режим доступу: *https://visualstudio.microsoft.com.*
- **NumPY** [Електронний ресурс]. – Режим доступу: *https://numpy.org/.*
- **Matplotlib:** Visualization with Python [Електронний ресурс]. – Режим доступу: *https://matplotlib.org/.*
- **SciKit-Imge** [Електронний ресурс]. – Режим доступу: *https://scikit-image.org/.*

# Додаток 1. Цифрові зображення відповідно варіантам.

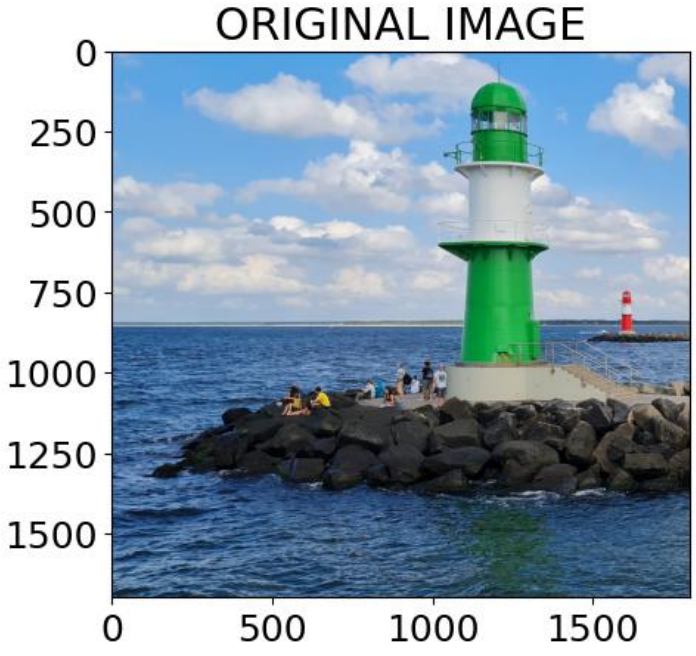
Зображення знаходяться в теці на GitHUB. Доступ за посиланням:

*https://github.com/eabshkvprof/2023_DIP/*

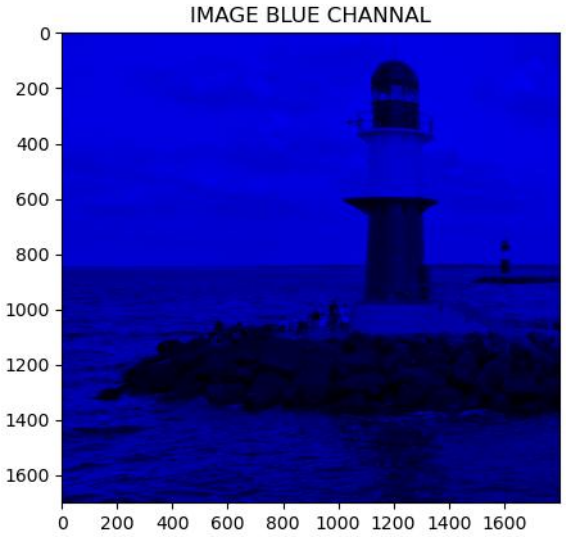
| Номер зображення | Файл Зображення |
|---|---|
| 1 | COCO_000012774.jpg |
| 2 | COCO_000015322.jpg |
| 3 | COCO_000025391.jpg |
| 4 | COCO_000028877.jpg |
| 5 | COCO_000040672.jpg |
| 6 | COCO_000049940.jpg |
| 7 | COCO_000104571.jpg |
| 8 | COCO_000112142.jpg |
| 9 | COCO_000117357.jpg |
| 11 | COCO_000130857.jpg |
| 12 | COCO_000170013.jpg |
| 13 | COCO_000173164.jpg |
| 14 | COCO_000176388.jpg |
| 24 | COCO_000180783.jpg |
| 15 | COCO_000186732.jpg |
| 16 | COCO_000201853.jpg |
| 17 | COCO_000224540.jpg |
| 18 | COCO_000253228.jpg |
| 19 | COCO_000277143.jpg |
| 20 | COCO_000286630.jpg |
| 21 | COCO_000362930.jpg |
| 22 | COCO_000366557.jpg |
| 23 | COCO_000386098.jpg |
| 24 | COCO_000390755.jpg |
| 25 | COCO_000411389.jpg |
| 26 | COCO_000495977.jpg |
| 27 | COCO_000575689.jpg |
| 28 | COCO_000580970.jpg |

# Додаток 2. Приклад коду до завдання 1

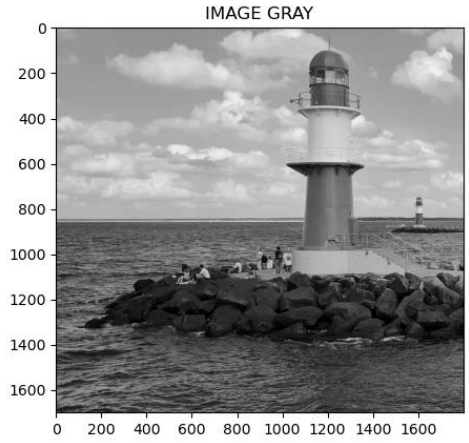
```
# Завантаження пакетів
import numpy as np
import skimage.io as io
import matplotlib.pyplot as plt
plt.rcParams['font.size']=10  #Розмір символів для графічного
виводу

#Завантаження файлу зображення
Filename = 'Повний шлях до файлу зображення'
test_im = io.imread(filename)
#Визначення структури та розміру зображення
print('IMAGE SHAPE', test_im.shape, 'IMAGE SIZE', test_im.size)
rows_num = test_im.shape[0] # кількість рядків
clms_num = test_im.shape[1] # кількість колонок
pix_num = rows_num*clms_num # кількість пікселів
bins = 256 # кількість рівнів яскравості
print('ROWS NUMBER',rows_num,'CLMS NUMBER',clms_num,'PIX NUMBER,
                                              pix_num, 'Bins',bins)
i = 100
j = 100
print ('PIXEL ',i,j,'COLORS R =',test_im[i,j,0],
                G = ,test_im[i,j,1],'B = ',test_im[i,j,2])
#Вивід оригінального зображення на екран
plt.title('ORIGINAL IMAGE')
plt.imshow(test_im)
plt.show()
```



```
#Визначення монохромного зображення
image_out = np.zeros ( (rows_num , clms_num,3), dtype=np.uint8)
#Only BLUE channel
image_out  [:, :, 0] =  0
image_out  [:, :, 1] =  0
image_out  [:, :, 2] =  test_im [:, :, 2]
plt.figure(figsize=(5, 5)) #Розмір на екрані
plt.title('IMAGE BLUE CHANNAL')
plt.imshow(image_out)
plt.show()
```




```
#Визначення ахроматичного зображення
image_gray = np.zeros ((rows_num , clms_num,3), dtype=np.uint8)
for i in  range (rows_num):
    for j in  range (clms_num):
        # Gray image
        image_gray [i, j, :] = 0.299*test_im [i, j,0]
               +0.587*test_im [i, j, 1]+0.114*test_im [ i, j, 2]
#Вивід перетвореного зображення на екран
plt.figure(figsize=(5, 5))
plt.title('IMAGE GRAY')
plt.imshow(image_gray)
plt.show()
```



```
# Визначення перетвореного зображення
image_transf = np.zeros ( (rows_num , rows_num,3), dtype=np.uint8)
for i in  range (rows_num):
    for j in  range (rows_num):
        # Gray image
        image_transf[i, j,:] = test_im[j+500, i,:]
#Вивід перетвореного зображення на екран
plt.title('TRANSFORM IMAGE')
plt.imshow(image_transf)
plt.show()
```