ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Звіт до індивідуального завдання №2

з предмету

«Обробка зображень та мультимедіа»

Виконав:

студент групи ПМі-43

Ваврикович Михайло

Викладач:

доц. Гутік Олег

2023

Для виконання цієї роботи я завантажила зображення з глибиною кольору 24 біт на піксель.

Для виконання наступних кроків, використовував IDE VS Code та бібліотеку OpenCV, яка надає набір інструментів для обробки зображень.

1. Спершу я відкриваю зображення та розділяю його на окремі канали (синій, зелений та червоний)

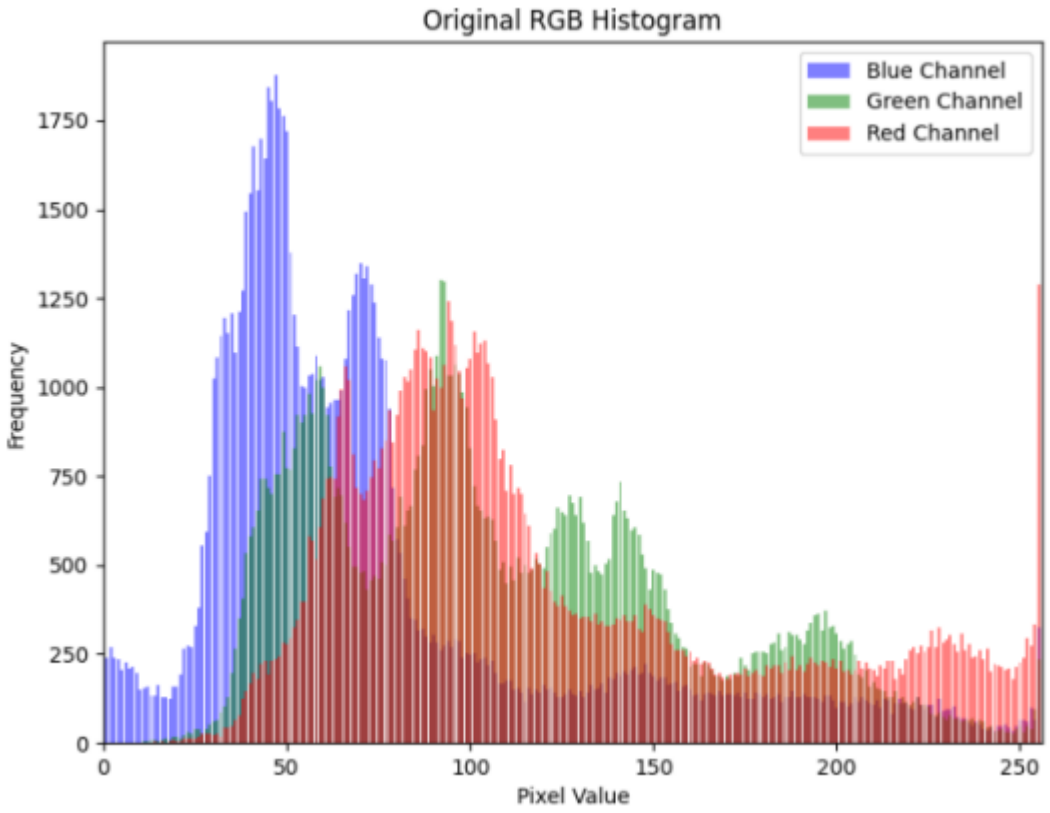
```
image = cv2.imread(image_path)
blue_channel, green_channel, red_channel = cv2.split(image)
```

Далі для кожного каналу обчислюю гістограми використовуючи функцію calcHist()

```
hist_blue = cv2.calcHist([blue_channel], [0], None, [256], [0, 256])
hist_green = cv2.calcHist([green_channel], [0], None, [256], [0, 256])
hist_red = cv2.calcHist([red_channel], [0], None, [256], [0, 256])
```

Першим параметром передаю канал, другим параметром передаю список індексів каналів, використовую [0], так як передаю лише один канал, третій параметр це область обчислення гістограми, None означає всю область, далі передаю кількість бітів та діапазон значень пікселів

Гістограму створюю за допомогою бібліотеки matplotlib. Для кожного каналу задаю його колір.



Видно, що тут переважає синій колір, і вона не є рівномірною.

2. Для проведення еквалізації використовую функцію equalizeHist і за таким ж принципом обчислюю гістограму.

```
blue_equalized = cv2.equalizeHist(blue_channel)
green_equalized = cv2.equalizeHist(green_channel
red_equalized = cv2.equalizeHist(red_channel)
```

```
hist_blue = cv2.calcHist([blue_equalized], [0], None, [256], [0, 256])
hist_green = cv2.calcHist([green_equalized], [0], None, [256], [0, 256])
hist_red = cv2.calcHist([red_equalized], [0], None, [256], [0, 256])
```

Отримав таку гістограму та еквалізоване зображення:

В результаті еквалізації гістограми покращується контрастність і розподіл кольорів.

3. Виконав процедури масочної фільтрації за допомогою наступних операторів:

Оператор Робертса:

Завантажую зображення у сірих відтінках:

```
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
```

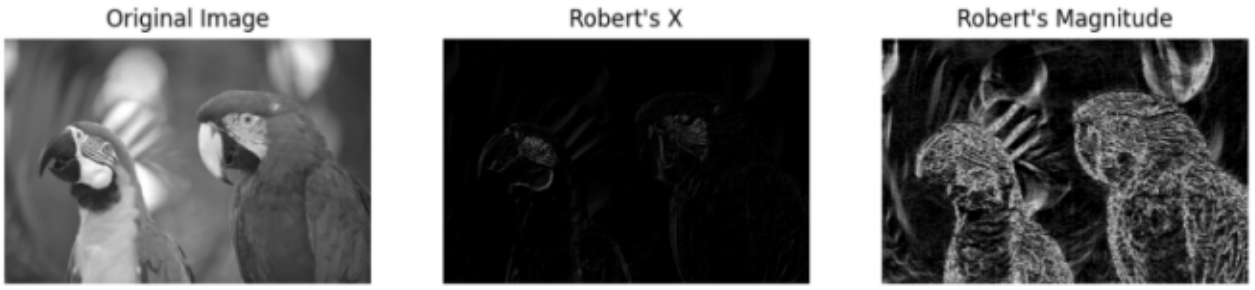
Визначаю матрицю оператора:

```
roberts_mask_x = np.array([[1, 0], [0, -1]], dtype=np.float32)
roberts_mask_y = np.array([[0, 1], [-1, 0]], dtype=np.float32)
```

Застосовую це до свого зображення:

```
roberts_x = cv2.filter2D(image, -1, roberts_mask_x)
roberts_y = cv2.filter2D(image, -1, roberts_mask_y)
```

І отримую такий результат:



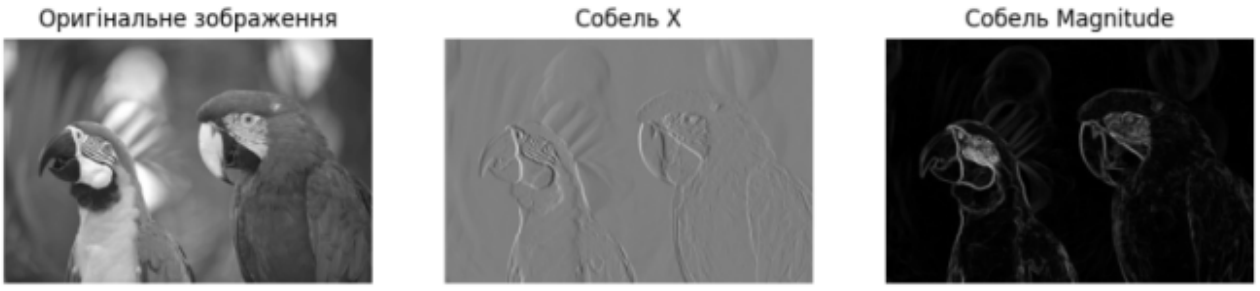
Цей оператор використовується для виявлення границь об’єктів на зображенні. Він вказує на горизонтальні та вертикальні границі.

Такі ж дії проводжу і для інших операторів.

Оператор Превіта:

Оператор Превіта також використовується для виявлення границь об’єктів на зображенні. Але він складніший ніж оператор Робертса і визначає границі у вісьмох напрямках. Визначає градієнт у різних напрямка

Оператор Собела:



Визначає градієнт у горизонтальному та вертикальному напрямках. Використовується для фільтрації зображення.

Для порівняння можемо побачити що для цього зображення оператор Превіта виділив межі найкраще, а Собела найгірше. Але оператор Робертса має найкращу швидкодію, тому що там використовується найменша матриця і треба виконувати менше операцій.