Звіт до індивідуального завдання №2
Виконала Мерцало Ірина, студентка групи ПМІ-41

Використала те саме зображення, що й для першого індивідуального завдання.
Зображення: https://www.pexels.com/photo/person-eye-865711/
Його метадані такі:

METADATA2GO.com FREE ONLINE EXIF VIEWER

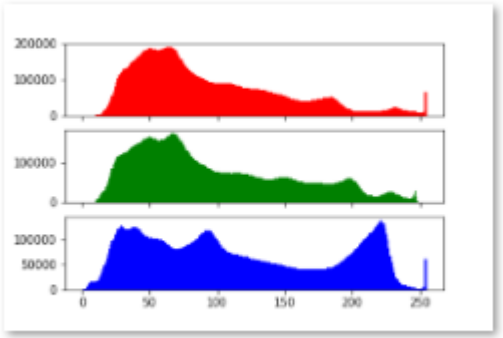
Metadata Info Of Your File

The following table contains all the exif data and metadata info we could extract from your file using our free online metadata

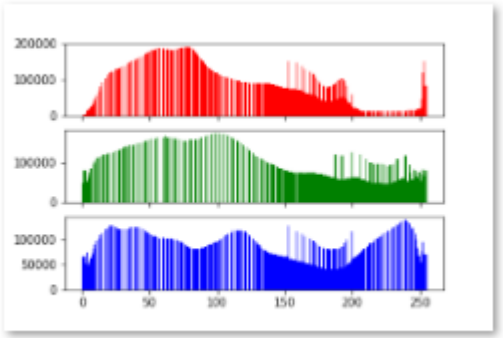
File Name	first_eye_2.bmp
File Size	51 MiB
File Type	BMP
File Type Extension	bmp
Mime Type	image/bmp
Bmp Version	Windows V5
Image Width	5184
Image Height	3456
Planes	1
Bit Depth	24
Compression	None
Image Length	53747712
Pixels Per Meter X	2834

Вихідне зображення BMP без стиснення важить 68.3 МБ і виглядає ось так:

Еквалізацію проводила по каналах кольорів, гістограма стало розріджена (нормалізувалася і розширилась по всьому діапазону). Гістограми до і після еквалізації виглядають так:



histogram_first.png



histogram_second.png

З цих гістограм ми можемо зрозуміти, що еквалізація відбулася добре, бо піки розширилися, а права частина гістограми піднялась.

Вихідне та отримане внаслідок еквалізації зображення виглядають так:

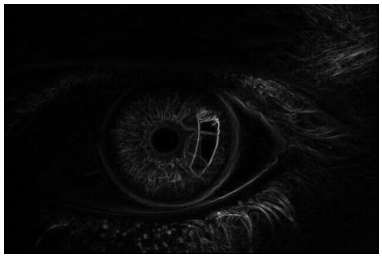
my_first_eye.bmp

my_second_eye.png

З порівняння цих зображень добре видно результат еквалізації, зображення стало світліше та вираніше. Особливо можна роздивитися результат на червоних венах на лівій частині білка ока, які були затемнені, а стали значно чіткіші та виразніші.

Внаслідок дії операторами фільтрації на зображення, отримала такі результати:

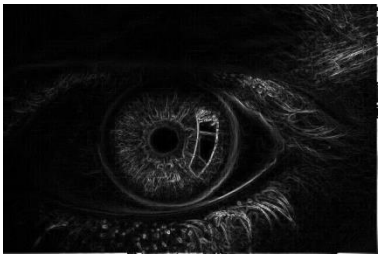
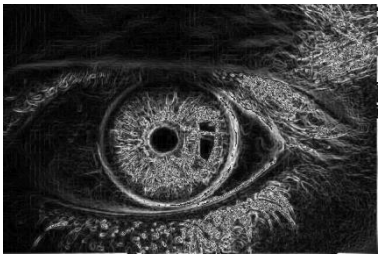
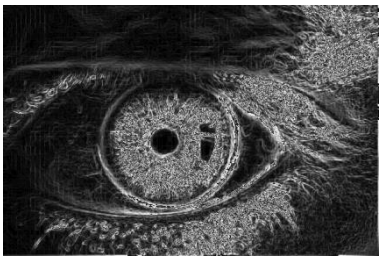
Previt_1 Roberts_1 Sobel_1


Тут можна побачити, що трохи гірший результат у моєму випадку є при застосуванні оператора робертса, зображення трохи розмитіше за інші. Тоді як після застосування двох інших операторів чітко виділилися лінії.

Також я пройшлася операторами по обрахованих на попередньому кроці пікселях, внаслідок чого отримала більш виразну відмінність між зображеннями.

Roberts_2 Previt_2 Sobel_2

Тут ми можемо побачити, що оператор Собеля дав занадто багато білого шуму, тоді як завдяки оператору Превіта ми можемо роздивитися лінії чіткіше. Оператор Робертса візуально більш красивий, адже залишив лише найбільш виразні лінії, але досить темний, тому про перевагу чи

недолік цього результату ми можемо говорити лише в контексті задач для яких будемо в подальшому використовувати це зображення.

Відповіді на зауваження, які виникли під час захисту:

1.Для еквалізації було використано ф-цію equalizeHist() бібліотеки openCV, попередньо були здійснені такі дії для її застосування (див.коментарі на скріні):

```
def histogram_equalization(img_in):

    # ділимо зображення на канали за кольором
    b,g,r = cv2.split(img_in)

    # Ф-цією flatten отримуємо копію масиву кожного кольору, зведеного в один вимір.
    # Ф-ція histogram - вбудована ф-ція в питру для побудови гістограми.
    # Параметрами вона приймає одновимірний масив (для цього ми тут застосували flatten), з якого буде будувати гістограму,
    # кількість стовпців та шкалу.
    h_b, bin_b = np.histogram(b.flatten(), 256, [0, 256])
    h_g, bin_g = np.histogram(g.flatten(), 256, [0, 256])
    h_r, bin_r = np.histogram(r.flatten(), 256, [0, 256])

    # Ф-ція cumsum повертає суму елементів вздовж заданої осі
    # Так ми дізнаємося загальну к-ть пікселів заданого кольору
    cdf_b = np.cumsum(h_b)
    cdf_g = np.cumsum(h_g)
    cdf_r = np.cumsum(h_r)

    # маскує кожен стовпець нулем (створює копію, де ми зможемо зберегти нашу нормалізовану гістограму)
    cdf_m_b = np.ma.masked_equal(cdf_b,0)
    # нормалізація мінімаксним методом, тут відбувається "розтягування" гістограми. Присвоюється попередньо підготовленій копії
    cdf_m_b = (cdf_m_b - cdf_m_b.min())*255/(cdf_m_b.max()-cdf_m_b.min())
    # копія "приводиться" у тип піксель(uint8) і заповнюється цими пікселями
    cdf_final_b = np.ma.filled(cdf_m_b,0).astype('uint8')

    cdf_m_g = np.ma.masked_equal(cdf_g,0)
    cdf_m_g = (cdf_m_g - cdf_m_g.min())*255/(cdf_m_g.max()-cdf_m_g.min())
    cdf_final_g = np.ma.filled(cdf_m_g,0).astype('uint8')

    cdf_m_r = np.ma.masked_equal(cdf_r,0)
    cdf_m_r = (cdf_m_r - cdf_m_r.min())*255/(cdf_m_r.max()-cdf_m_r.min())
    cdf_final_r = np.ma.filled(cdf_m_r,0).astype('uint8')

    # розтягує гістограми на весь проміжок
    equ_b = cv2.equalizeHist(b)
    equ_g = cv2.equalizeHist(g)
    equ_r = cv2.equalizeHist(r)
    equ = cv2.merge((equ_b, equ_g, equ_r))
    #print(equ)
    cv2.imwrite('my_second_eye.png', equ)

    # вертає нормалізовані значення з трьох каналів у одне зображення
    img_g = cdf_final_g[g]
    img_r = cdf_final_r[r]
    img_b = cdf_final_b[b]
    img_out = cv2.merge((img_b, img_g, img_r))

    return img_out
```

2. Під час реалізації операторів (i-1, j-1) береться для того, щоб значення яке було в 1,1 у вихідній матриці записати в 0, 0 у новому масиві. Це робиться тому, що матриця бере попередні і наступні пікселі, а для 0, 0 нема попереднього так само як для n, m нема наступного.

```
if(operator == 'sobel' or operator == 'previt'):
    for channel in range(size[2]):
        for i in range(1, size[0] - 1):
            for j in range(1, size[1] - 1):
                horizontalGrad = (horizontal[0, 0] * img[i - 1, j - 1, channel]) + \
                                 (horizontal[0, 1] * img[i - 1, j, channel]) + \
                                 (horizontal[0, 2] * img[i - 1, j + 1, channel]) + \
                                 (horizontal[1, 0] * img[i, j - 1, channel]) + \
                                 (horizontal[1, 1] * img[i, j, channel]) + \
                                 (horizontal[1, 2] * img[i, j + 1, channel]) + \
                                 (horizontal[2, 0] * img[i + 1, j - 1, channel]) + \
                                 (horizontal[2, 1] * img[i + 1, j, channel]) + \
                                 (horizontal[2, 2] * img[i + 1, j + 1, channel])

                verticalGrad = (vertical[0, 0] * img[i - 1, j - 1, channel]) + \
                               (vertical[0, 1] * img[i - 1, j, channel]) + \
                               (vertical[0, 2] * img[i - 1, j + 1, channel]) + \
                               (vertical[1, 0] * img[i, j - 1, channel]) + \
                               (vertical[1, 1] * img[i, j, channel]) + \
                               (vertical[1, 2] * img[i, j + 1, channel]) + \
                               (vertical[2, 0] * img[i + 1, j - 1, channel]) + \
                               (vertical[2, 1] * img[i + 1, j, channel]) + \
                               (vertical[2, 2] * img[i + 1, j + 1, channel])

                mag = np.sqrt(pow(horizontalGrad, 2.0) + pow(verticalGrad, 2.0))
                img[i - 1, j - 1, channel] = mag
```

Висновок:

Таким чином, на основі зображення з оком, було простежено результат дії еквалізації, яка відбулася добре. Те, що вона відбулась добре ми можемо побачити по гістограмі (вона розрідилася, піди поширилися на області де їх не вистачало, а також правий край піднявся), саме зображення стало світлішим та виразнішим, а хороший результат дії еквалізації можна особливо простежити на венах, які знаходились в темному куті ока – їх також стало дуже чітко видно.

Внаслідок дії операторів масочної фільтрації Робертса, Превіта і Собеля на це зображення, було простежено, що оператор Робертса дає більш розмите зображення, тоді як Собеля і Превіта з досить схожим результатом дають більш чіткі лінії. Проте, якщо подіяти ними на матрицю утворену на попередньому кроці, то можемо побачити, що краще працює на цьому зображенні оператор Превіта, ніж оператор Собеля, а аналізувати оператор Робертса в такому випадку можна лише залежно від подальшої задачі використання зображення, бо він дає досить чіткі основні лінії, але водночас зображення дуже темне.