Sprawozdanie z Laboratorium nr 4 WMM

Podstawowe przetwarzanie obrazów – Piotr Szmurlo (303785)

Zdjęcie oryginalne:

Obraz zaszumiony szumem gaussa:

Obraz zaszumiony szumem impulsowym:

Skrypt ładujący obrazy:

```
#color
data_dir_color = os.path.join('obrazy_testowe', 'color')
images = sorted(os.listdir(data_dir_color))
img_path_color = os.path.join(data_dir_color, images[303785%len(images)])
image_color = cv2.imread(img_path_color)

#color_inoise1
data_dir_inoise1 = os.path.join('obrazy_testowe', 'color_inoise1')
images = sorted(os.listdir(data_dir_inoise1))
img_path_inoise1 = os.path.join(data_dir_inoise1, images[303785%len(images)])
image_inoise1 = cv2.imread(img_path_inoise1)

#color_noise gauss
data_dir_noise = os.path.join('obrazy_testowe', 'color_noise')
images = sorted(os.listdir(data_dir_noise))
img_path_noise = os.path.join(data_dir_noise, images[303785%len(images)])
image_noise = cv2.imread(img_path_noise)
```

Zadanie 1

Do filtracji obrazu korzystam z funkcji zawartych w bibliotece OpenCV: *cv2.GaussianBlur i cv2.medianBlur*. Do obliczenia PSNR używam dostarczonej z zadaniem funkcji *calcPSNR*.

Obrazy zaszumione szumem gaussowskim, przetworzone filtrem wygładzającym gaussowskim:

3x3:

5x5:

7x7:

Obrazy zaszumione szumem impulsowym, przetworzone filtrem wygładzającym gaussowskim:

3x3:

5x5:

7x7:

Obrazy zaszumione szumem gaussowskim, przetworzone filtrem wygładzającym medianowym:

3x3:

5x5:

7x7:

Obrazy zaszumione szumem impulsowym, przetworzone filtrem wygładzającym medianowym:

3x3:

5x5:

7x7:

Tabela z wartościami PSNR:

	Szum gauss; filtr gaussowski	Szum impulsowy; filtr gaussowski	Szum gauss; filtr medianowy	Szum impulsowy; filtr medianowy
3x3	27.15	25.28	25.68	26.24
5x5	25.42	24.62	23.28	23.35
7x7	24.08	23.71	22.43	22.41

Na podstawie powyższej tabeli można wywnioskować, że filtr Gaussa daje lepsze wyniki dla szumu gaussowskiego, a filtr medianowy daje wyniki lepsze dla szumu impulsowego. Najlepszą rekonstrukcję obrazu według tabeli PSNR dał filtr gaussowski 3x3 na szumie gausowskim; warto jednak nadmienić, że wartości PSNR nie zawsze odpowiadają subiektywnym opiniom i tak jest w tym przypadku – moim zdaniem ten obrazek nie wygląda najlepiej. Im rozmiar maski jest większy, tym mocniej obraz jest zniekształcony.

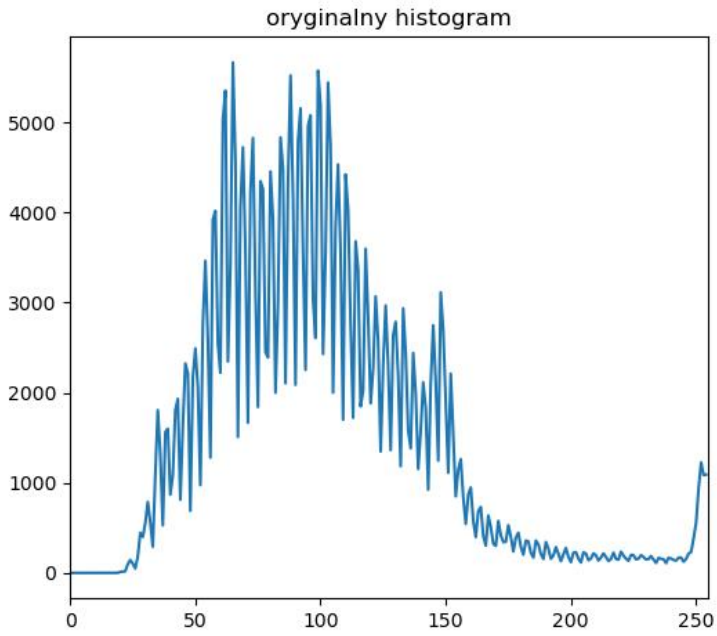
Skrypt do rozwiązania zadania:

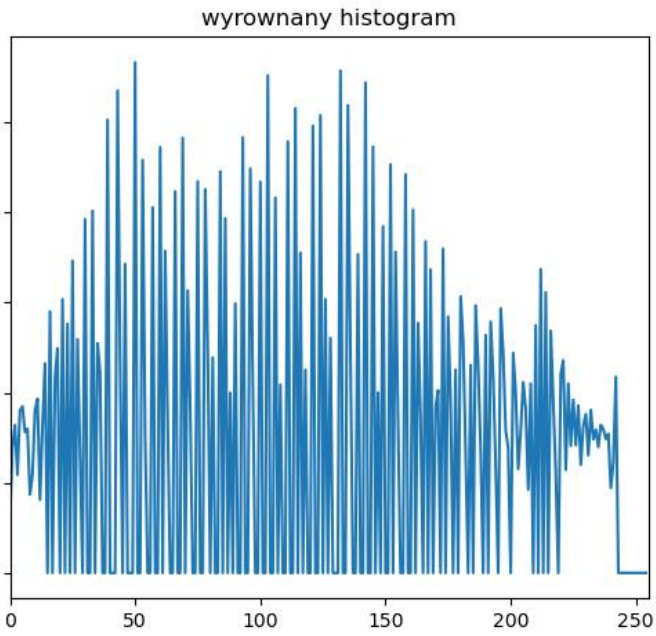
```
for i in (3, 5, 7): # przetwarzanie obrazu filtrem Gaussa zaszumionego szumem gaussowskim
    g_gauss_blurred_img = cv2.GaussianBlur(image_noise, (i, i), 0)
    psnr[0].append(calcPSNR(image_color, g_gauss_blurred_img))
    cv2.imwrite(f"out_gauss_gauss_{i}x{i}.png", g_gauss_blurred_img)

for i in (3, 5, 7): # przetwarzanie obrazu filtrem Gaussa zaszumionego szumem impulsowym
    i_gauss_blurred_img = cv2.GaussianBlur(image_inoise1, (i, i), 0)
    psnr[1].append(calcPSNR(image_color, i_gauss_blurred_img))
    cv2.imwrite(f"out_inoise1_gauss_{i}x{i}.png", i_gauss_blurred_img)

for i in (3, 5, 7): # przetwarzanie obrazu filtrem medianowym zaszumionego szumem gaussowskim
    g_median_blurred_img = cv2.medianBlur(image_noise, i)
    psnr[2].append(calcPSNR(image_color, g_median_blurred_img))
    cv2.imwrite(f"out_gauss_median_{i}x{i}.png", g_median_blurred_img)

for i in (3, 5, 7): # przetwarzanie obrazu filtrem medianowym zaszumionego szumem impulsowym
    i_median_blurred_img = cv2.medianBlur(image_inoise1, i)
    psnr[3].append(calcPSNR(image_color, i_median_blurred_img))
    cv2.imwrite(f"out_inoise1_median_{i}x{i}.png", i_median_blurred_img)
```

Zadanie 2

Do rozwiązania zadania użyłem funkcji z OpenCV: cv2.cvtColor cv2.calcHist i cv2.equalizeHist.

Przestrzeń barw RGB zmieniłem na YUV i wyrównałem histogram dla składowej Y, a następnie powróciłem do przestrzeni barw RGB.





Obraz przed wyrównaniem histogramu:

Obraz po wyrównaniu histogramu:

Obraz po wyrównaniu histogramu subiektywnie jest *lepszej* jakości.

Skrypt do rozwiązania zadania:

```
image_equalized = cv2.cvtColor(image_color, cv2.COLOR_BGR2YUV)
original_histogram = cv2.calcHist([image_equalized], [0], None, [255], [0, 255])
image_equalized[:, :, 0] = cv2.equalizeHist(image_equalized[:, :, 0])
equalized_histogram = cv2.calcHist([image_equalized], [0], None, [255], [0, 255])
image_equalized = cv2.cvtColor(image_equalized, cv2.COLOR_YUV2BGR)

fig, axs = plt.subplots(1, 2, sharey=True, tight_layout=True)
axs[0].plot(original_histogram.flatten())
axs[1].plot(equalized_histogram.flatten())
axs[0].set_xlim(0, 255)
axs[1].set_xlim(0, 255)
axs[0].set_title("oryginalny histogram")
axs[1].set_title("wyrownany histogram")
plt.show()
cv2.imwrite("equalized_image.png", image_equalized)
```

Zadanie 3

Do rozwiązania zadania skorzystałem z funkcji zawartych w OpenCV: cv2.Laplacian i cv2.addWeighted.

Obraz wygenerowany przez filtr Laplace’a:

Wynikowe obrazu po zsumowaniu krawędzi z wagami *w*:

Obraz dla w=0.5

Obraz dla w=1

Obraz dla w=1.5

Obraz dla w=2

Szumu nie zostały uwydatnione, więc stosowanie filtru wygładzającego nie było konieczne.

Waga *w* odpowiada za uwydatnienie krągwędzi (im większa, tym bardziej widoczne).

Dla wagi *w* z przedziału od ok. 0.5 do mniej niż 1 otrzymane obrazy są dobre i przyjemne dla oka. Dla większych wag krawędzie są zbyt przerysowane i obraz wygląda nienaturalnie.

Skrypt do rozwiązania zadania:

```
laplacian_image = cv2.Laplacian(image_color, cv2.CV_8U)
cv2.imwrite("laplacian_image.png", laplacian_image)
for id, weight in enumerate([0.5, 1.0, 1.5, 2.0]):
    result = cv2.addWeighted(image_color, 1, laplacian_image, -weight, 0)
    cv2.imwrite(f"laplacian_image_w{id}.png", result)
```