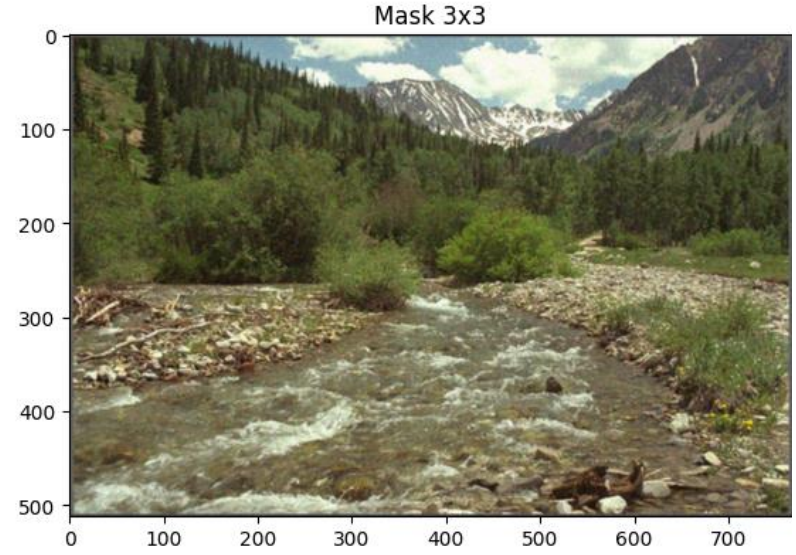
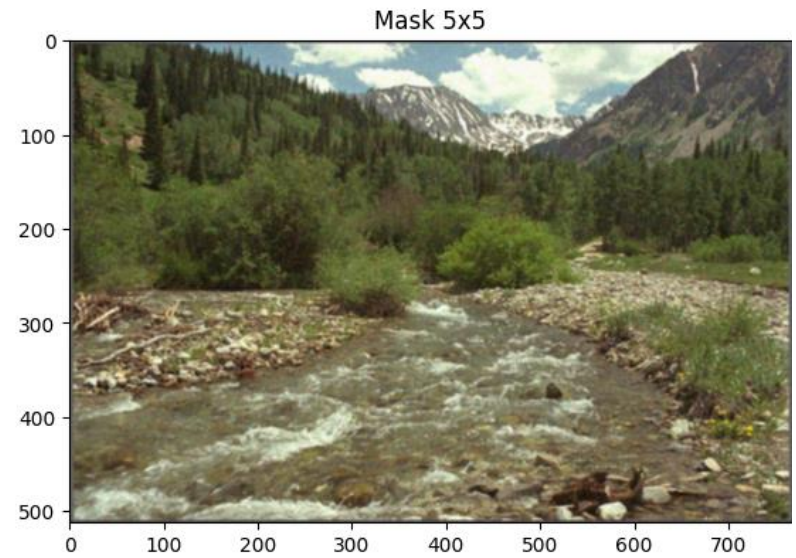
Sprawozdanie Laboratorium WMM - Podstawowe przetwarzanie obrazów

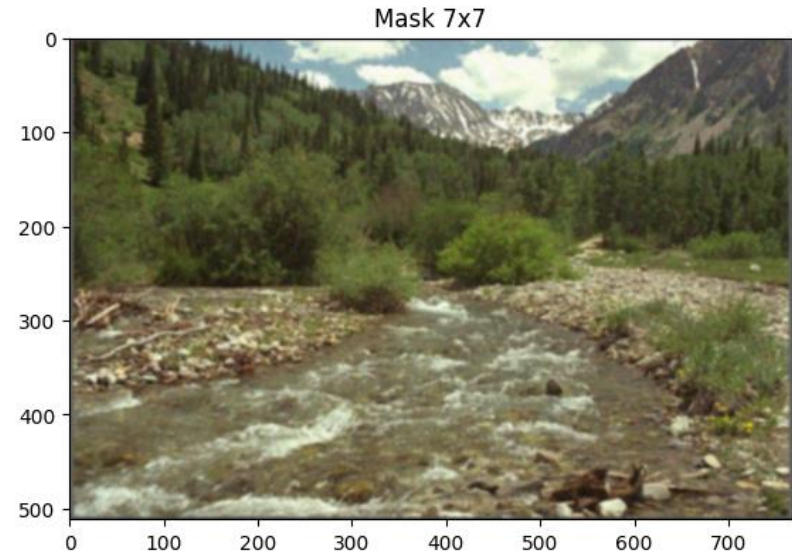
Mateusz Ostaszewski 325203

ZAD1

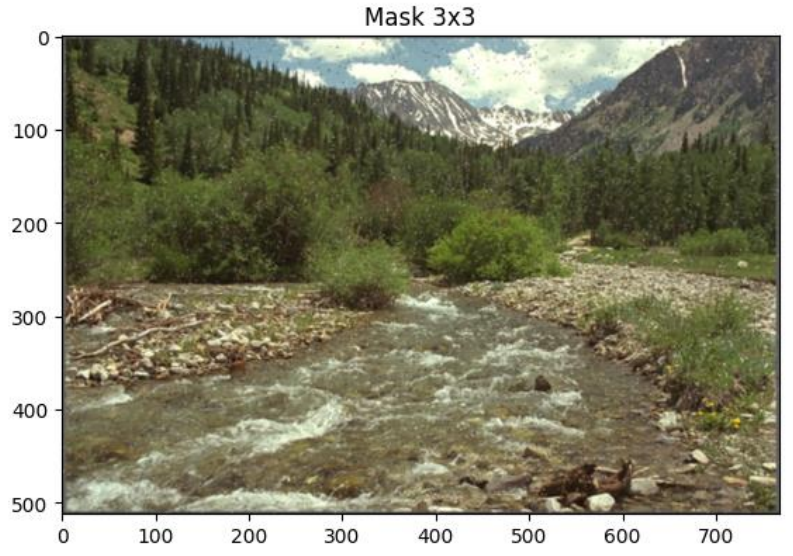
Filtracja Gaussa obrazu zaszumionego szumem gaussowskim

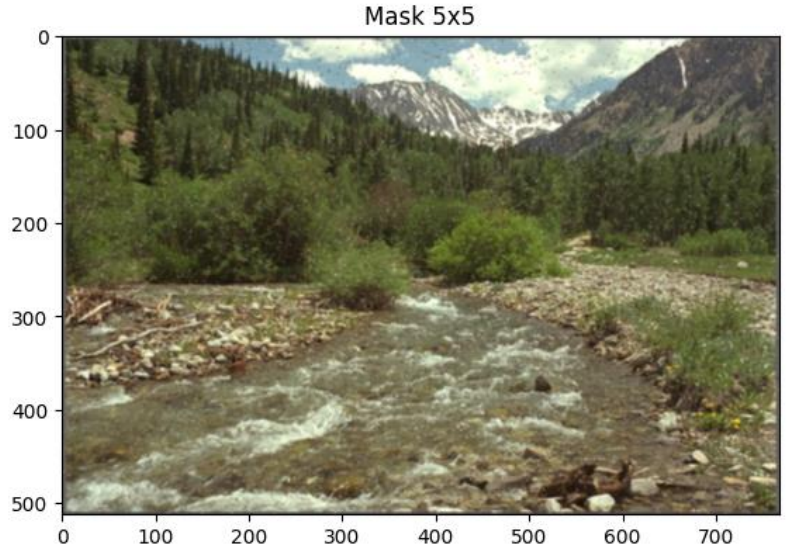


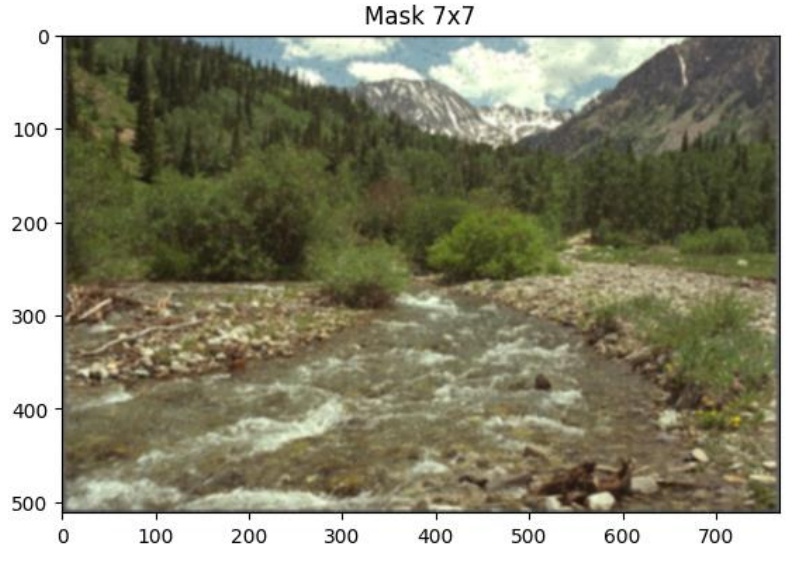




Filtracja Gaussa obrazu zaszumionego szumem impulsowym

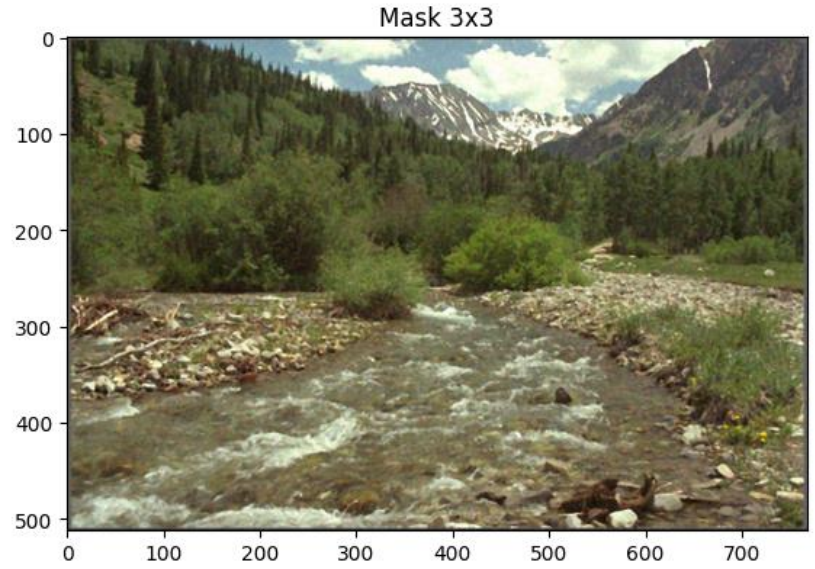


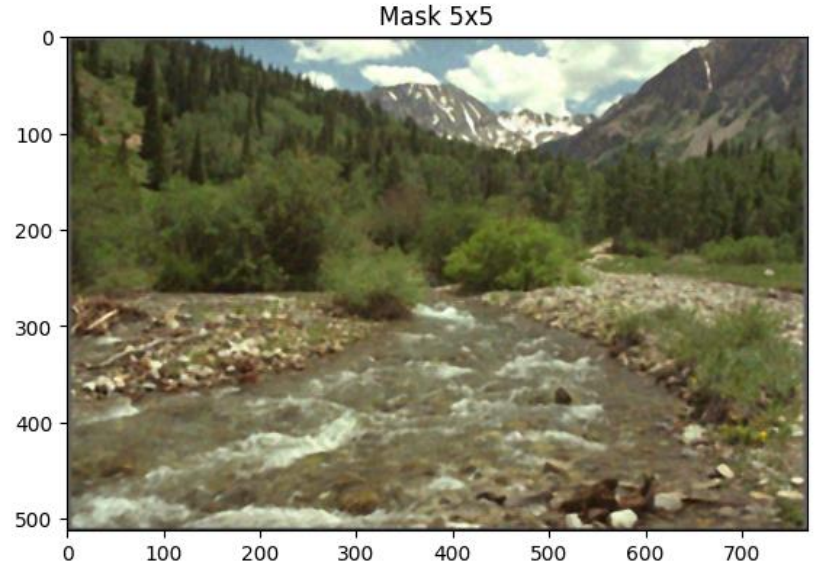


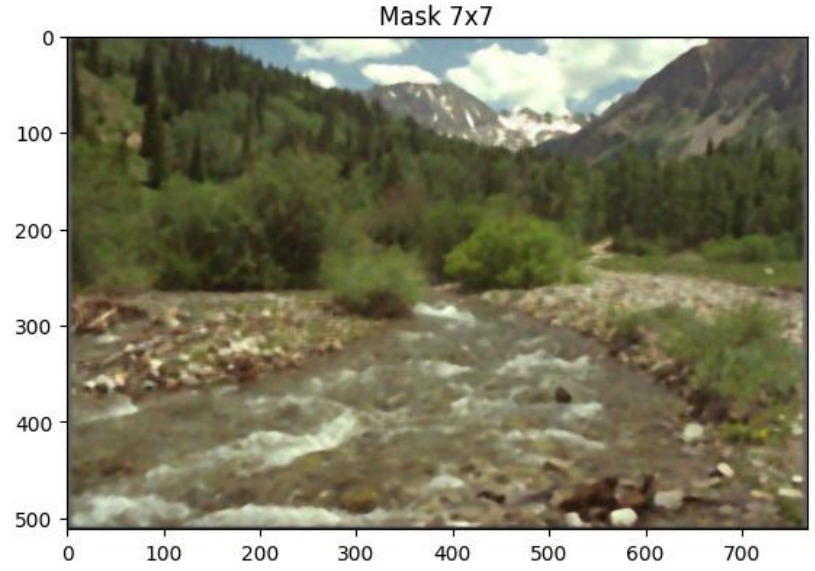


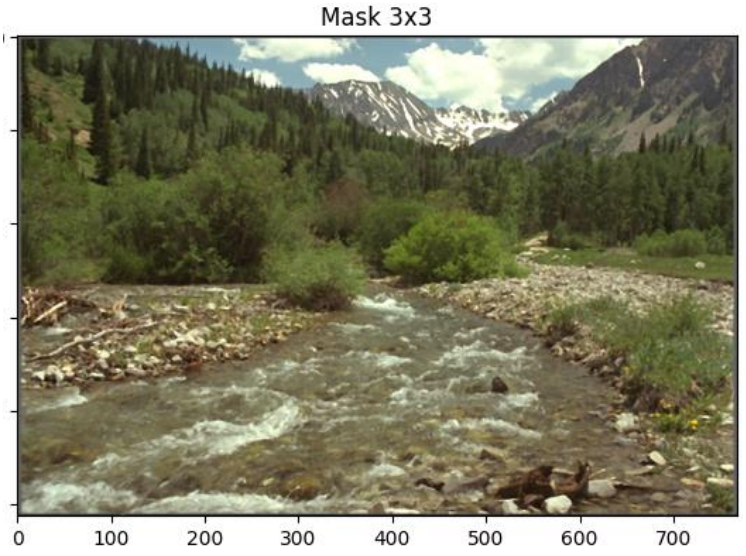
Filtracja medianowa obrazu zaszumionego szumem impoulsowym

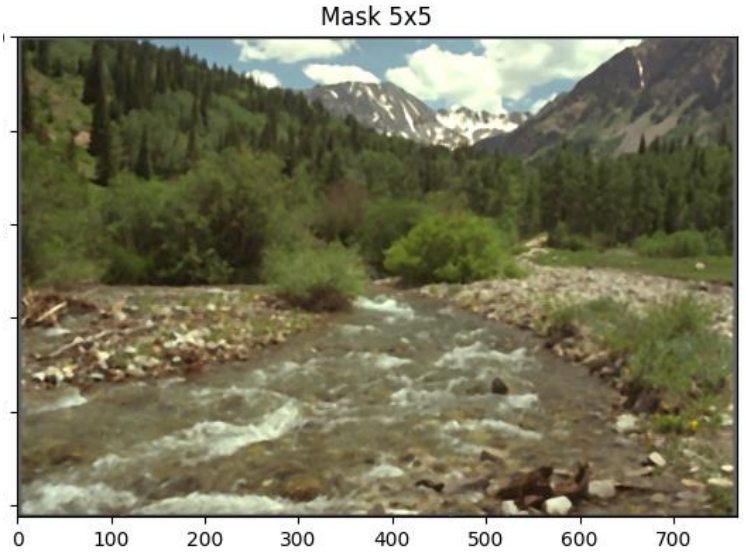
Filtracja medianowa obrazu zaszumionego szumem gaussowskim











Wnioski

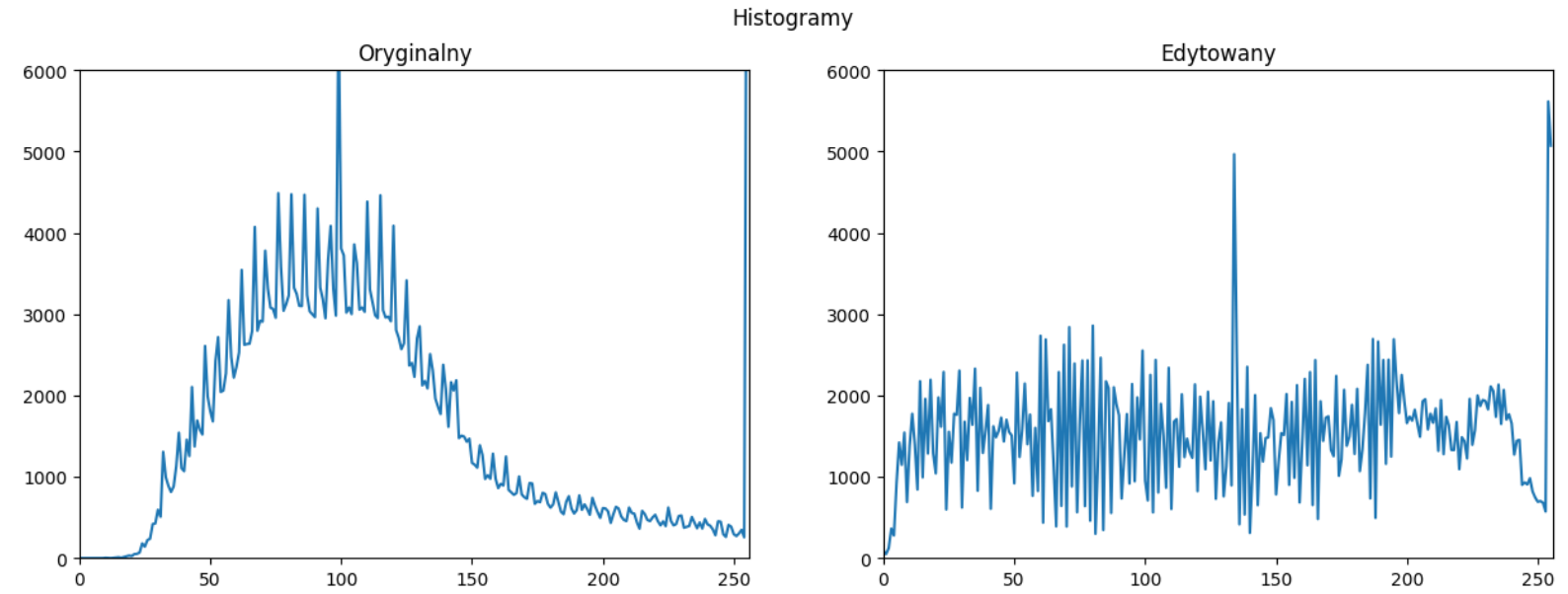
Z powyższej tabeli wynika, że najlepiej się sprawdził filtr gaussowski z maską 3x3, w tej konfiguracji uzyskujemy najwyższy wskaźnik PSNR, co oznacza że moc sygnału jest znacznie większa niż moc szumu, obraz jest mniej zniekształcony i bliższy oryginalnemu. W mojej subiektywnej opinii również jest to zdjęcie o najlepszej jakości.

Wpływ rozmiaru maski:

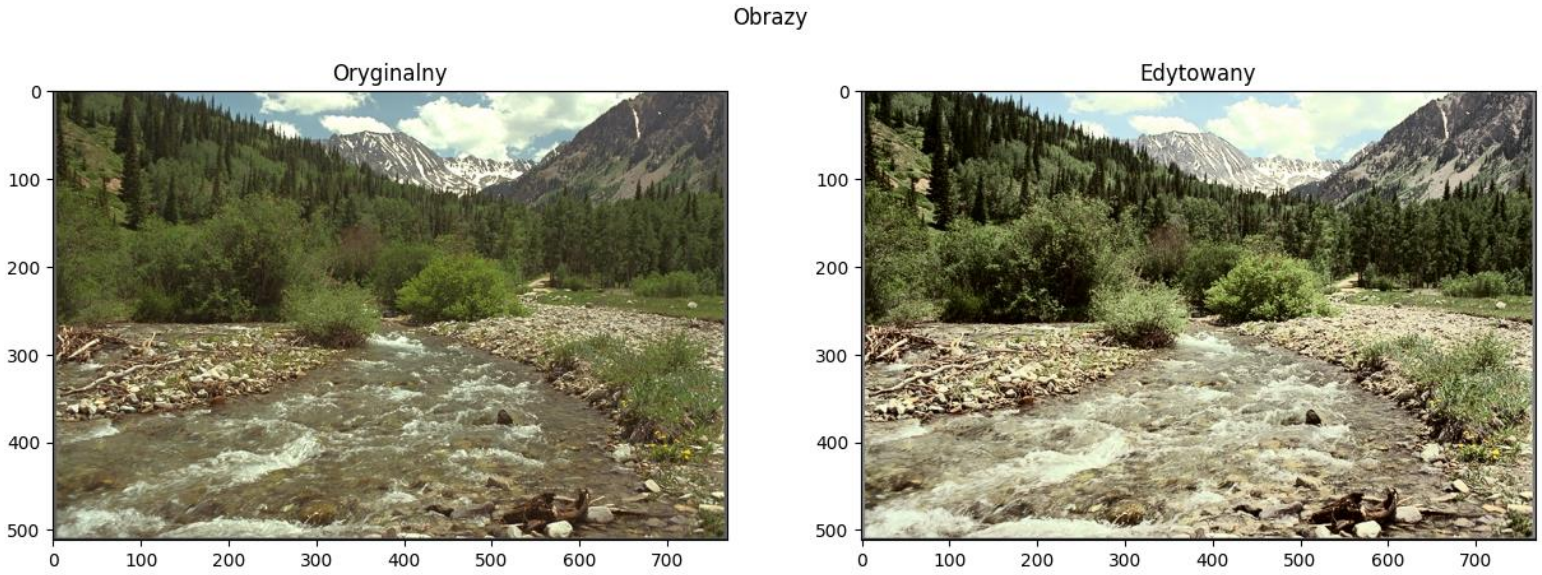
- Skuteczność filtracji: Większa maska filtru (np. 7x7 zamiast 3x3) będzie bardziej skuteczna w redukcji szumów, ponieważ bierze pod uwagę większą ilość pikseli sąsiednich. To może prowadzić do lepszego usunięcia szumów, ale kosztem utraty szczegółów obrazu.

- Zniekształcenie obrazu: Większa maska filtru może prowadzić do większego zniekształcenia obrazu, ponieważ filtracja jest bardziej "agresywna". Może to prowadzić do efektu rozmycia, ponieważ wysokie częstotliwości (szczegóły) mogą być zredukowane.

Zad2



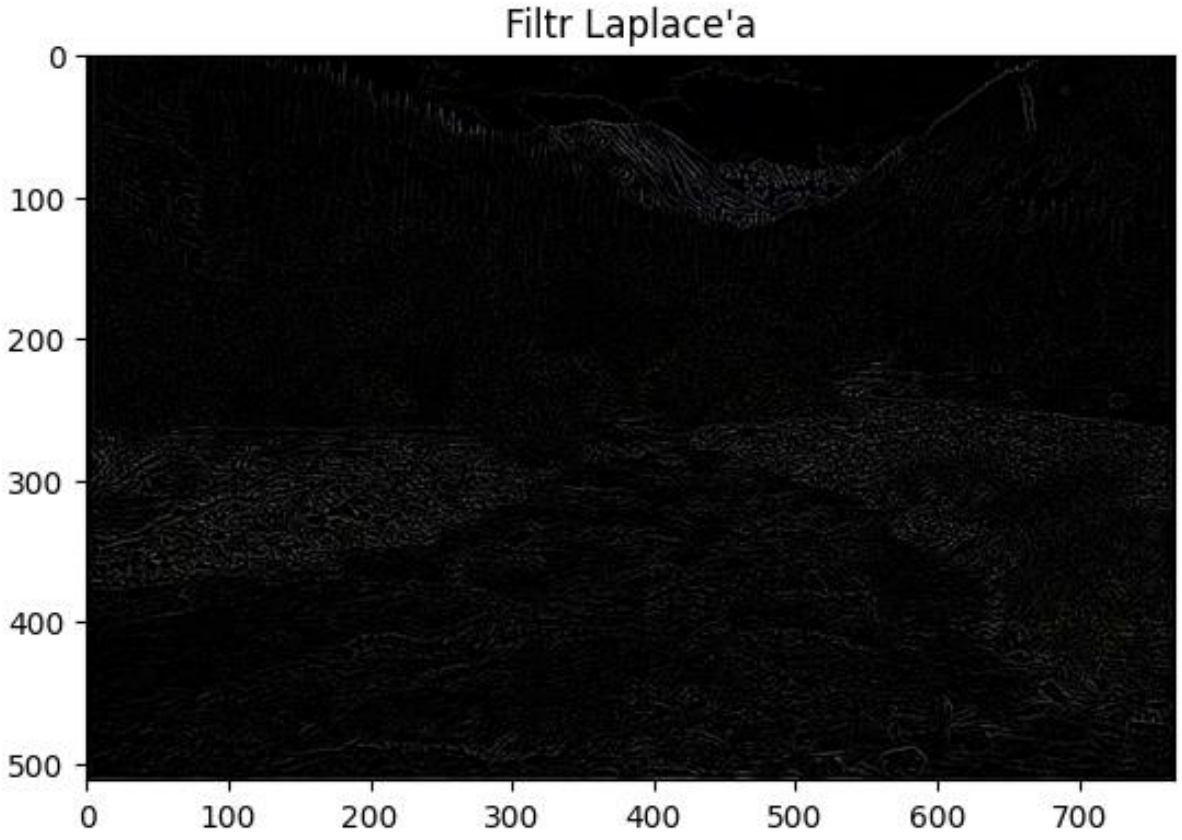
Histogram został względnie wyrównany

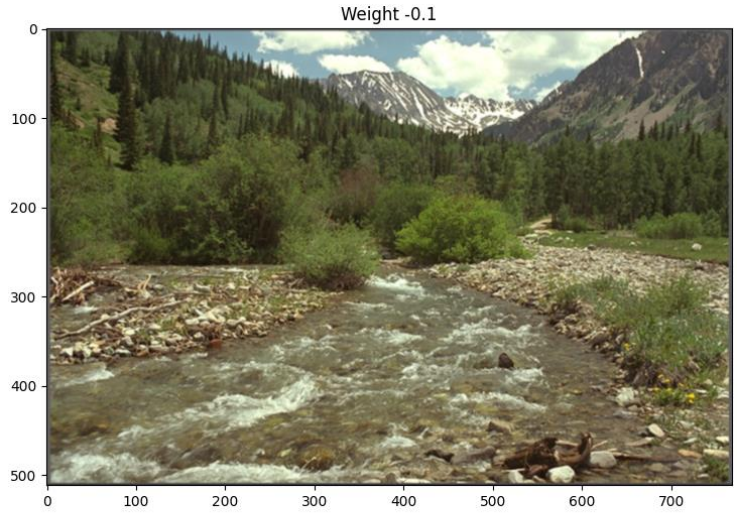


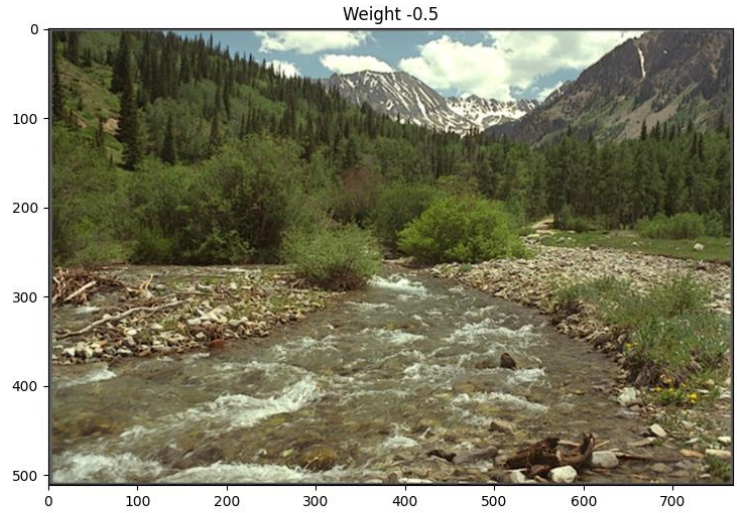
Wnioski

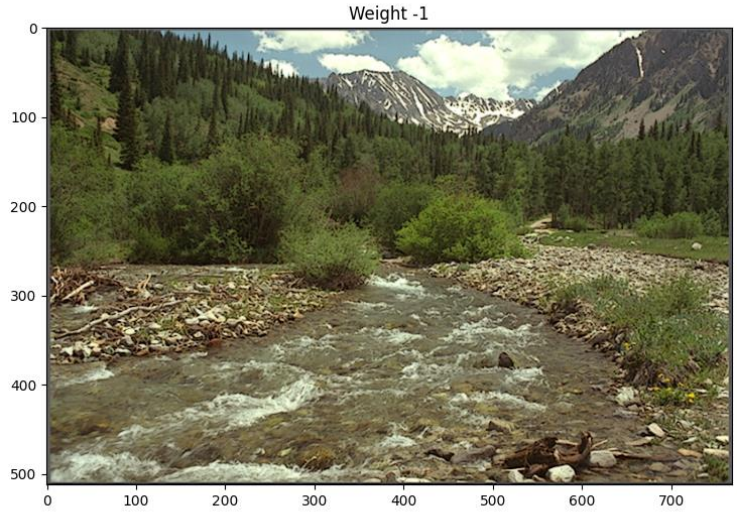
Podczas przeglądania wyników edycji obrazu, zauważyłem znaczącą poprawę jego jakości. Moje subiektywne odczucia sugerują, że manipulacje przeprowadzone na obrazie przyniosły pozytywne efekty, dodając mu estetycznej wartości i poprawiając ogólne wrażenie wizualne.

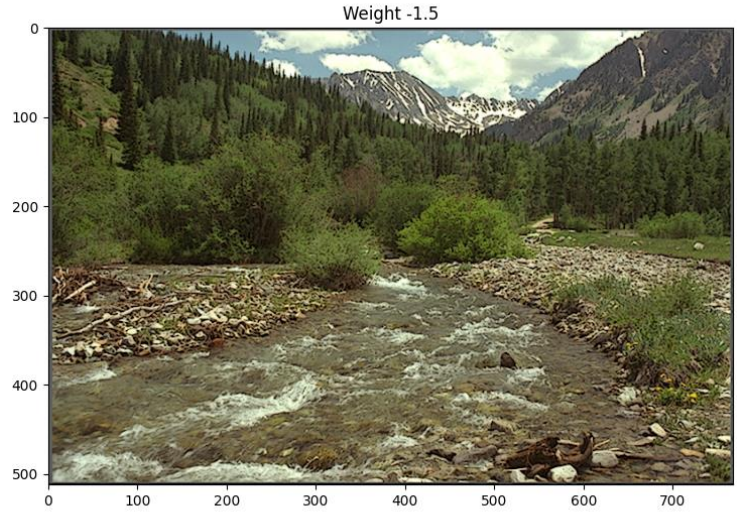
Zad3

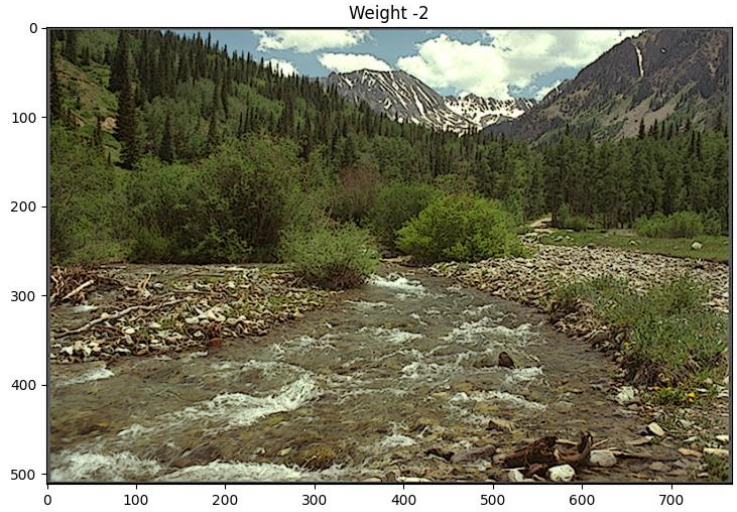


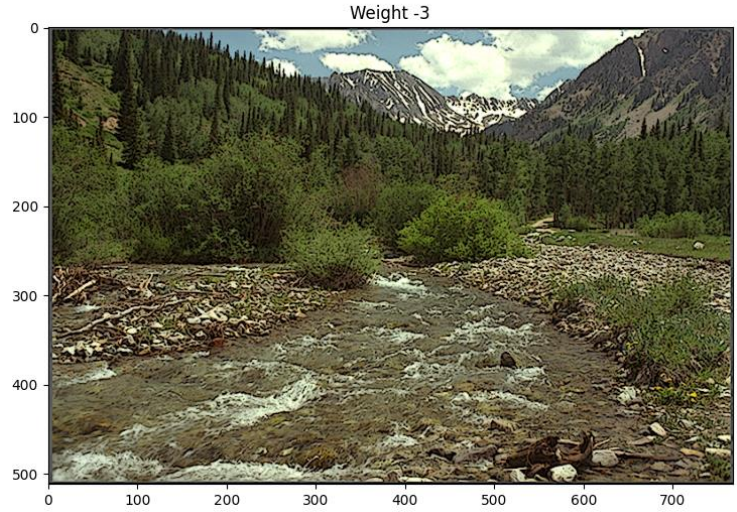












Wnioski

Przy zastosowaniu wag z zakresu od -1.5 do -0.5, uzyskujemy estetycznie atrakcyjne i przyjemne dla oka rezultaty. Wartości wag -3 i -2 intensyfikują kontrast do stopnia, który może negatywnie wpływać na jakość obrazu, czyniąc go mniej atrakcyjnym. Z kolei, wyostrzenie obrazu przy użyciu wagi -0.1 jest subtelne i nie wprowadza znaczących zmian w porównaniu do obrazu oryginalnego.

KOD

Importy

```
from pathlib import Path
import cv2
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import math
```

[50] ✓ 0.0s

Otworzenie zadanych obrazów

```
images_dir = Path("obrazy_testowe")
oryginal_dir = images_dir / "color"
inoise1_dir = images_dir / "color_inoise1"
inoise2_dir = images_dir / "color_inoise2"
noise_dir = images_dir / "color_noise"

oryginal_imgs = sorted(list(oryginal_dir.glob("*.png")))
inoise1_imgs = sorted(list(inoise1_dir.glob("*.png")))
inoise2_imgs = sorted(list(inoise2_dir.glob("*.png")))
noise_imgs = sorted(list(noise_dir.glob("*.png")))

id_number = 325203
number_of_imgs = len(oryginal_imgs)
img_nr = id_number % number_of_imgs
print(number_of_imgs)


oryginal_img = cv2.cvtColor(cv2.imread(str(oryginal_imgs[img_nr])), cv2.COLOR_BGR2RGB)
inoise1_img = cv2.cvtColor(cv2.imread(str(inoise1_imgs[img_nr])), cv2.COLOR_BGR2RGB)
inoise2_img = cv2.cvtColor(cv2.imread(str(inoise2_imgs[img_nr])), cv2.COLOR_BGR2RGB)
gaus_noise_img = cv2.cvtColor(cv2.imread(str(noise_imgs[img_nr])), cv2.COLOR_BGR2RGB)

plt.figure(figsize=(15, 10))
plt.subplot(2, 2, 1)
plt.title('Oryginal')
plt.imshow(oryginal_img)
plt.subplot(2, 2, 2)
plt.title('Gaus Noise')
plt.imshow(gaus_noise_img)
plt.subplot(2, 2, 3)
plt.title('Impuls Noise1')
plt.imshow(inoise1_img)
plt.subplot(2, 2, 4)
plt.title('Impuls Noise2')
plt.imshow(inoise2_img)
```

ZAD1

```python
def calcPSNR(img1, img2):
    imax = 255.**2  ### zakładana wartość pikseli z przedziału [0, 255]
    ##### w różnicy obrazów istotne są wartości ujemne, dlatego img1 konwertowan
    mse = ((img1.astype(np.float64)-img2)**2).sum()/img1.size  ### img1.size - l
    return 10.0*np.log10(imax/mse)

psnrs = [[],[],[],[]]

masks = [3, 5, 7]
```

✓ 0.0s

```python
plt.figure(figsize=(10, 15))
plt.suptitle('Filtracja Gaussa obrazu zaszumionego szumem gaussowskim')
for idx, i in enumerate(masks):
    gblur_img = cv2.GaussianBlur(gaus_noise_img, (i, i), 0)
    psnrs[0].append(calcPSNR(oryginal_img, gblur_img))
    plt.subplot(3, 1, idx+1)
    plt.title(f'Mask {i}x{i}')
    plt.imshow(gblur_img)
```

```python
plt.figure(figsize=(10, 15))
plt.suptitle('Filtracja Gaussa obrazu zaszumionego szumem impulsowym')
for idx, i in enumerate(masks):
    gblur_img = cv2.GaussianBlur(inoise1_img, (i, i), 0)
    psnrs[1].append(calcPSNR(oryginal_img, gblur_img))
    plt.subplot(3, 1, idx+1)
    plt.title(f'Mask {i}x{i}')
    plt.imshow(gblur_img)
```

✓ 3.5s

```python
plt.figure(figsize=(10, 15))
plt.suptitle('Filtracja medianowa obrazu zaszumionego szumem impoulsowym')
for idx, i in enumerate(masks):
    mblur_img = cv2.medianBlur(inoise1_img, i)
    psnrs[3].append(calcPSNR(oryginal_img, mblur_img))
    plt.subplot(3, 1, idx+1)
    plt.title(f'Mask {i}x{i}')
    plt.imshow(mblur_img)
```

✓ 2.8s

```python
plt.figure(figsize=(10, 15))
plt.suptitle('Filtracja medianowa obrazu zaszumionego szumem gaussowskim')
for idx, i in enumerate(masks):
    mblur_img = cv2.medianBlur(gaus_noise_img, i)
    psnrs[2].append(calcPSNR(oryginal_img, mblur_img))
    plt.subplot(3, 1, idx+1)
    plt.title(f'Mask {i}x{i}')
    plt.imshow(mblur_img)
```

```python
# Nazwy filtrów i szumów
index = ['filtr gaussowski, szum gaussowski',
         'filtr gaussowski, szum impulsowy',
         'filtr medianowy, szum gaussowski',
         'filtr medianowy, szum impulsowy']

# Rozmiary masek
columns = ['3x3', '5x5', '7x7']

# Tworzenie DataFrame
df = pd.DataFrame(psnrs, index=index, columns=columns)

# Wyświetlanie DataFrame
print(df)
```

Zad2

Histogram dla orginalnego zdjęcia

```
original_histogram = cv2.calcHist([oryginal_img], [0], None, [256], [0, 256])
original_histogram = original_histogram.flatten()
```

✓ 0.0s

Konwersja i wyrównanie histogramu

```
coverted_img = cv2.cvtColor((oryginal_img), cv2.COLOR_RGB2YUV)
coverted_img[:, :, 0] = cv2.equalizeHist(coverted_img[:, :, 0]) # Y - odpowiedzi
equalized_img = cv2.cvtColor(coverted_img, cv2.COLOR_YUV2RGB)

equalized_histogram = cv2.calcHist([equalized_img], [0], None, [256], [0, 256])
equalized_histogram = equalized_histogram.flatten()
```

✓ 0.0s

```
cv2.imwrite("equalized_img.png", equalized_img)
```

✓ 0.0s

```
True
```

Porównanie histogramów

```
plt.figure(figsize=(15, 5))
plt.suptitle("Histogramy")
plt.subplot(1, 2, 1)
plt.title("Oryginalny")
plt.xlim([0, 256])
plt.ylim([0, 6000])
plt.plot(original_histogram)
plt.subplot(1, 2, 2)
plt.title("Edytowany")
plt.xlim([0, 256])
plt.ylim([0, 6000])
plt.plot(equalized_histogram)
```

Porównanie zdjęć

```
plt.figure(figsize=(15, 5))
plt.suptitle("Obrazy")
plt.subplot(1, 2, 1)
plt.title("Oryginalny")
plt.imshow(oryginal_img)
plt.subplot(1, 2, 2)
plt.title("Edytowany")
plt.imshow(equalized_img)
```

✓ 2.2s

Zad 3

Wstępne wygładzenie obrazu

```
img = cv2.GaussianBlur(oryginal_img, (3, 3), 0)
```

✓ 0.0s

Wygenerowanie obrazu Laplace'a

```
laplacian_img = cv2.Laplacian(img, cv2.CV_8U)
plt.title("Filtr Laplace'a")
plt.imshow(laplacian_img)
```

✓ 0.3s

Wyostrzanie

```
weights = [-0.1, -0.5, -1, -1.5, -2, -3]
plt.figure(figsize=(20, 20))
for i, weight in enumerate(weights):
    result_img = cv2.addWeighted(img, 1, laplacian_img, weight, 0)
    plt.subplot(math.ceil(len(weights) / 2), 2, i+1)
    plt.title(f'Weight {weight}')
    plt.imshow(result_img)
```

✓ 5.9s