

Anna Tamelo 336059

WMM lab 4.

Zadanie 1. Zrealizować operację barwnego obrazu cyfrowego



GaussNoise_Gaussian_3x3



GaussNoise_Gaussian_5x5



GaussNoise_Gaussian_7x7



GaussNoise_Median_3x3



GaussNoise_Median_5x5



GaussNoise_Median_7x7



ImpulseNoise_Gaussian_3x3



ImpulseNoise_Gaussian_5x5



ImpulseNoise_Gaussian_7x7



ImpulseNoise_Median_3x3



ImpulseNoise_Median_5x5



ImpulseNoise_Median_7x7

PSNR	3x3	5x5	7x7
Gauss_Gaussian	34.11	34.31	33.56
Gauss_Median	33.11	33.33	32.66
Impulse_Gaussian	31.28	32.46	32.56
Impulse_Median	38.24	35.09	33.61

Wpływ rozmiaru maski filtru na skuteczność usuwania szumu i na zniekształcenie obrazu:

- Filtr Gaussa:
Dla szumu gaussowskiego większy rozmiar maski może efektywniej wygładzić rozproszone zakłócenia, ale jednocześnie bardziej rozmywa detale (co może obniżyć PSNR). Z kolei dla szumu impulsowego filtr Gaussa rzadko bywa optymalny, jednak powiększanie maski również powoduje silniejsze rozmycie.
- Filtr medianowy:
Dla szumu gaussowskiego filtr medianowy działa wygładzająco, jednak na podstawie wartości PSNR widać, że jest on mniej dopasowany do rozkładu gaussowskiego niż filtr Gaussa. W przypadku szumu impulsowego filtr medianowy jest bardziej skuteczny, niż filtr Gaussa (w tabeli widać wysokie PSNR przy maskach 3x3 i 5x5). Przy większych maskach (7x7) może nadmiernie wygładzać obraz, co prowadzi do spadku PSNR.

Podsumowując, im większe maska, tym zazwyczaj mocniejsze tłumienie szumu, ale rośnie rozmycie krawędzi i spada ostrość. W efekcie przy zbyt dużej masce spada PSNR, bo traci się ważne detale obrazu.

Czy ocena subiektywna jest zgodna z wynikami PSNR?

Tak, ocena wizualna jest zgodna z wartościami PSNR - wyższe PSNR oznacza mniej zniekształceń, a więc wyraźniejszy i czystszy obraz.

Kod:

```
import cv2
import numpy as np

def calcPSNR(img1, img2):
    imax = 255.**2
    mse = ((img1.astype(np.float64) - img2.astype(np.float64))**2).sum() / img1.size
    if mse == 0:
        return 100
    return 10.0 * np.log10(imax / mse)

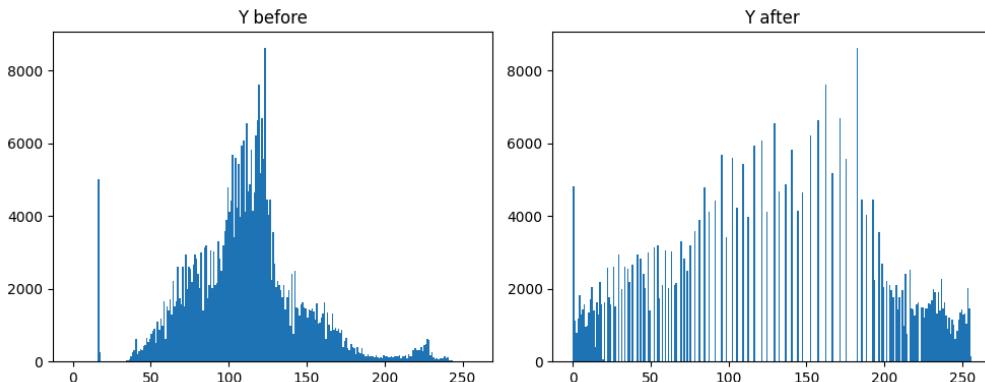
path = "./"
original = cv2.imread(path + "susie_col.png", cv2.IMREAD_COLOR)
img_gauss_noise = cv2.imread(path + "susie_col_noise.png", cv2.IMREAD_COLOR)
```

```



```

Zadanie 2. Zrealizować operację wyrównania histogramu dla obrazu barwnego.



Na podstawie powyższych histogramów widać, że przed wyrównaniem histogram kanału Y (jasności) jest skoncentrowany w okolicach wartości 90-120. Po wyrównaniu wartości pikseli w

kanale Y rozkładają się znacznie szerzej na całym przedziale 0-255, co przekłada się na większy kontrast i bardziej zrównoważoną jasność w obrazie końcowym. Po wyrównaniu histogramu najciemniejsze obszary stały jeszcze ciemniejsze, a jasne stały jeszcze jaśniejsze, przez co przybywa detali zarówno w cieniach, jak i w jasnych częściach. Subiektywnie obraz wygląda wyraźniej, ale nie "lepiej", ma większy kontrast i lepszą widoczność szczegółów.

Kod:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

image = cv2.imread("susie_col.png")
ycrcb = cv2.cvtColor(image, cv2.COLOR_BGR2YCrCb)
ycrcb[:, :, 0] = cv2.equalizeHist(ycrcb[:, :, 0])
image_eq = cv2.cvtColor(ycrcb, cv2.COLOR_YCrCb2BGR)
cv2.imwrite("susie_col_eq.png", image_eq)

orig_ycrcb = cv2.cvtColor(image, cv2.COLOR_BGR2YCrCb)
eq_ycrcb = cv2.cvtColor(image_eq, cv2.COLOR_BGR2YCrCb)

plt.figure(figsize=(10, 4))
plt.subplot(1, 2, 1)
plt.title("Y before")
plt.hist(orig_ycrcb[:, :, 0].ravel(), bins=256, range=(0, 256))
plt.subplot(1, 2, 2)
plt.title("Y after")
plt.hist(eq_ycrcb[:, :, 0].ravel(), bins=256, range=(0, 256))
plt.tight_layout()
plt.show()
```

Zadanie 3. Dokonać wyostrzenia obrazu korzystając z filtra Laplace'a do wyznaczenia wysokoczęstotliwościowych składowych obrazu.





weight = -0.4



weight = -0.6



weight = -0.8



weight = -1.0

Im większa wartość bezwzględna wagi, tym mocniejsze wyostrzanie (zarówno krawędzi, jak i ewentualnego szumu). Zbyt duża wartość prowadzi do przeostrzenia obrazu, silnych artefaktów oraz wzmacniania zakłóceń. Zbyt mała wartość powoduje, że efekt wyostrzania jest słabo widoczny. Dla podanego obrazu wartości wagi w okolicach -0.2 - -0.8 pozwalają uzyskać dobre, przyjemne dla oka wyniki.

Kod:

```
import cv2
import numpy as np

img_in = cv2.imread("susie_col.png", cv2.IMREAD_GRAYSCALE)
img.blur = cv2.GaussianBlur(img_in, (3, 3), 0)
lap = cv2.Laplacian(img.blur, cv2.CV_64F, ksize=3)

weights = [-0.2, -0.4, -0.6, -0.8, -1.0]

for w in weights:
    img_in_f = img_in.astype(np.float32)
    lap_f = lap.astype(np.float32)

    sharp_float = cv2.addWeighted(img_in_f, 1.0, lap_f, w, 0.0)
    sharp_8u = cv2.convertScaleAbs(sharp_float)

    cv2.imshow(f"Sharp (w={w})", sharp_8u)
    cv2.imwrite(f"susie_sharp_w{w}.png", sharp_8u)

cv2.waitKey(0)
cv2.destroyAllWindows()
```