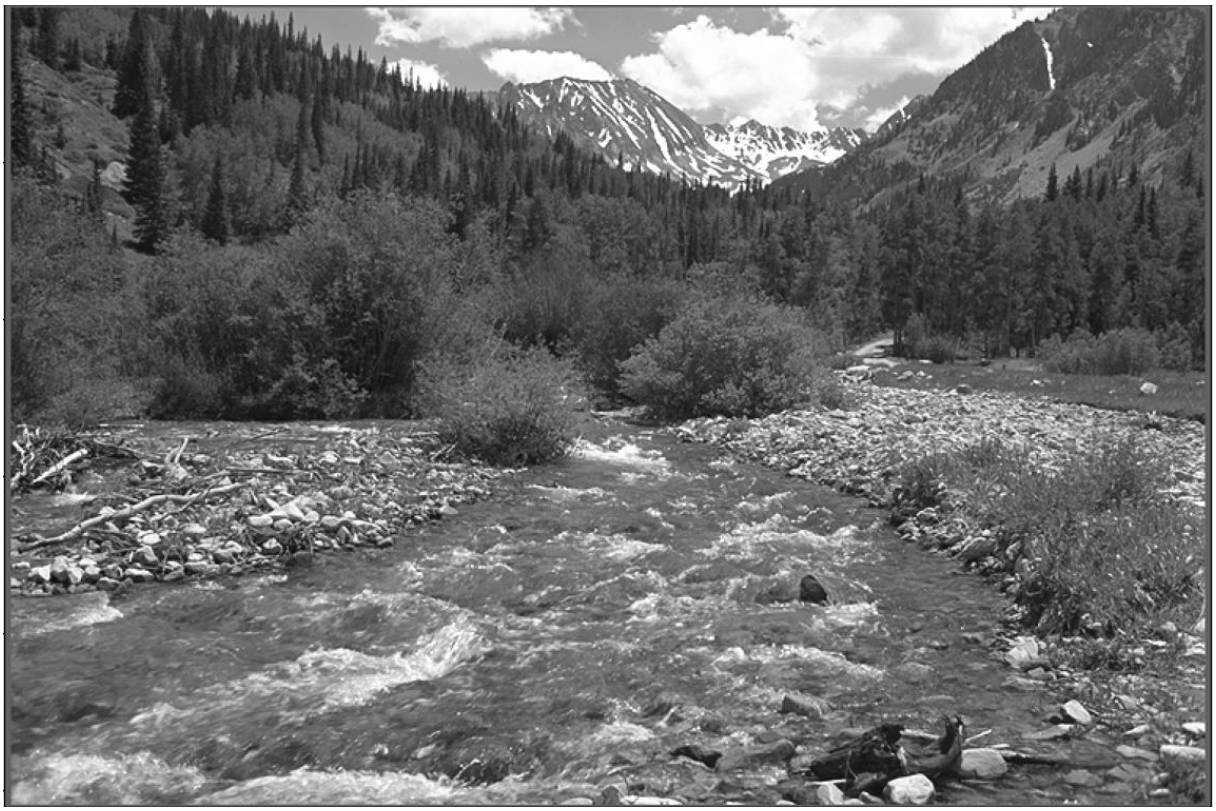
Sprawozdanie Laboratorium WMM - Statyczne właściwości obrazów

Mateusz Ostaszewski 325203

Dla obrazu monochromatycznego

Obraz wejściowy



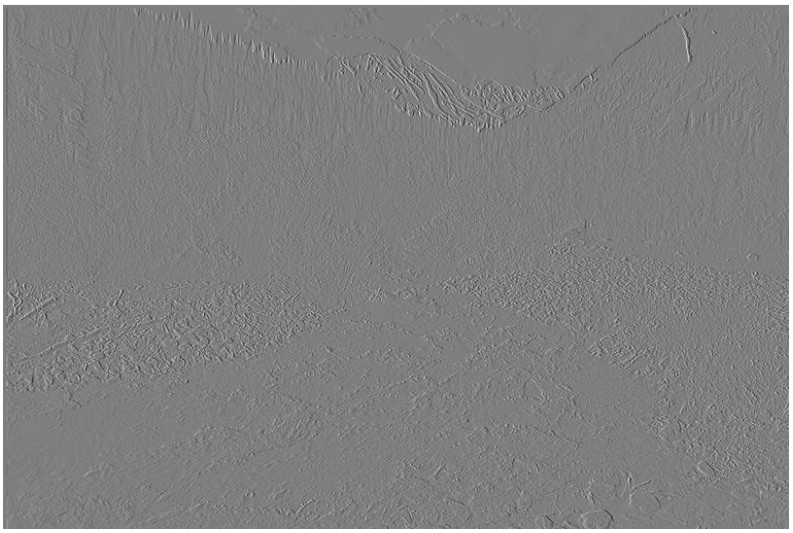
Zad1

Entropia obrazu wejściowego

Entropia = 7.4317

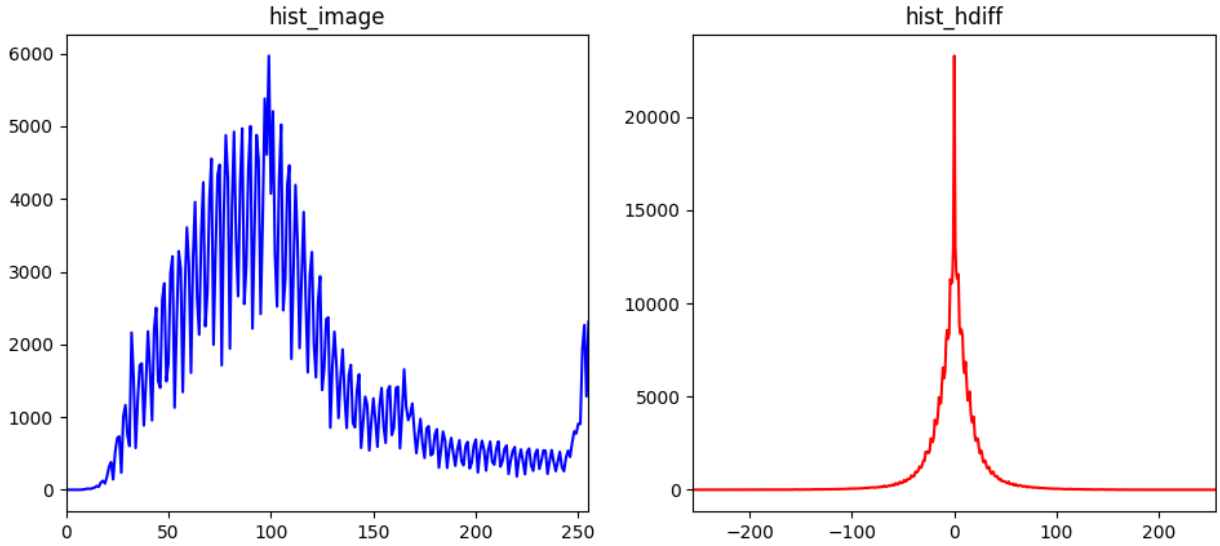
Zad2

Wyznaczam obraz różnicowy w taki sposób, że wartość każdego piksela zastępowana jest różnicą pomiędzy wartością tego piksela a wartością jego lewego sąsiada.



Na przeskalowanym obrazie różnicowym widać krawędzie.

Histogramy



Histogram obrazu oryginalnego ma bardziej rozproszone wartości, pokazując szerszy zakres intensywności pikseli, co sugeruje większą różnorodność w wartościach pikseli. Histogram obrazu różnicowego jest znacznie węższy i skoncentrowany wokół wartości 0, co wskazuje, że wiele pikseli ma małą różnicę w intensywności względem swoich sąsiadów.

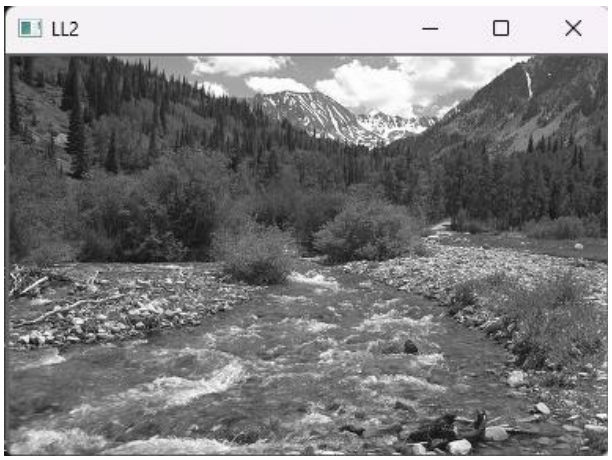
Entropia obrazu predykcją poziomą: 6.3391

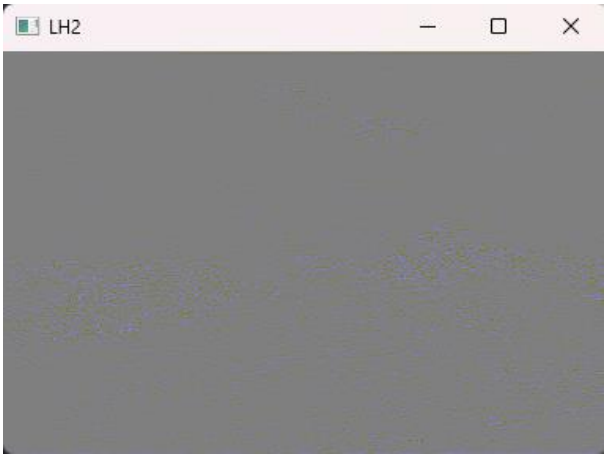
Entropia obrazu oryginalnego: 7.4317

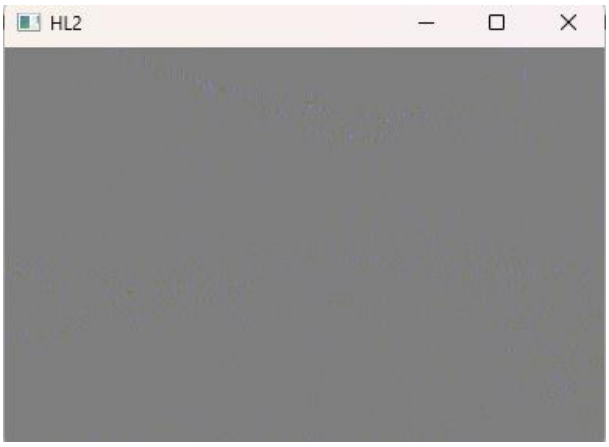
Entropia obrazu różnicowego jest znacząco niższa w porównaniu z entropią obrazu oryginalnego, ale obraz różnicowy dalej zawiera tą samą liczbę informacji.

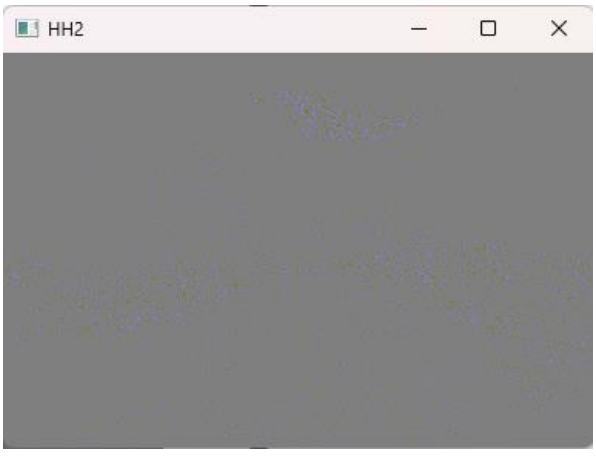
Zad3

Wyznaczam współczynniki DWT korzystając z funkcji zamieszczonej w skrypcie i wyświetlam poszczególne pasma.

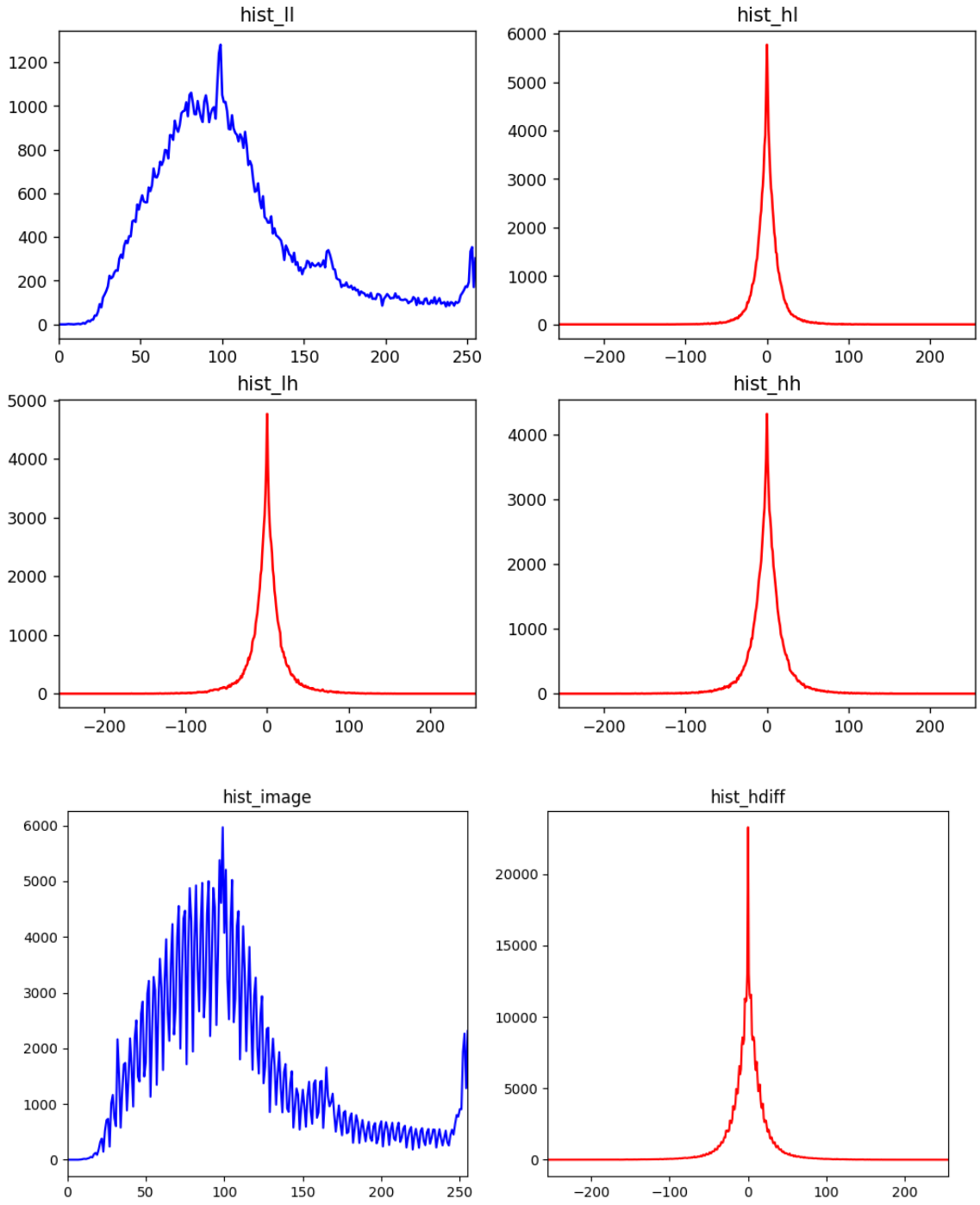








Histogramy



Entropie

Entropia (LL) = 7.4532
Entropia (LH) = 6.2555
Entropia (HL) = 5.8266
Entropia (HH) = 6.2754
Entropia_śr = 6.4527

Analiza Histogramów

Histogramy dla pasm LL (niskie częstotliwości), LH (poziome krawędzie), HL (pionowe krawędzie) i HH (diagonalne krawędzie) wykazują znaczące różnice w rozkładzie intensywności.

Pasmo LL, reprezentujące niskie częstotliwości, ma bardziej płaski histogram, sugerujący większe zróżnicowanie wartości, co jest typowe dla ogólnych trendów i wolnych zmian intensywności w obrazie.

Pasma LH, HL i HH mają histogramy skupione wokół wartości zero, co wskazuje na mniej aktywnych pikseli, ponieważ te pasma kodują informacje o krawędziach i detalach - wysokie częstotliwości, które są mniej powszechne w typowych obrazach.

Analiza Entropii

Entropia obrazu oryginalnego jest wyższa niż entropie pasm LH, HL i HH, ale niższa niż entropia pasma LL, co oznacza, że najwięcej informacji w obrazie znajduje się w pasmie niskich częstotliwości.

Wnioski

Wyniki te wskazują, że kompresja obrazu przy użyciu transformacji falkowej pozwala na skuteczne oddzielenie istotnych informacji o obrazie (niskie częstotliwości) od detali i szumu (wysokie częstotliwości), co może być wykorzystane do bardziej efektywnej kompresji danych.

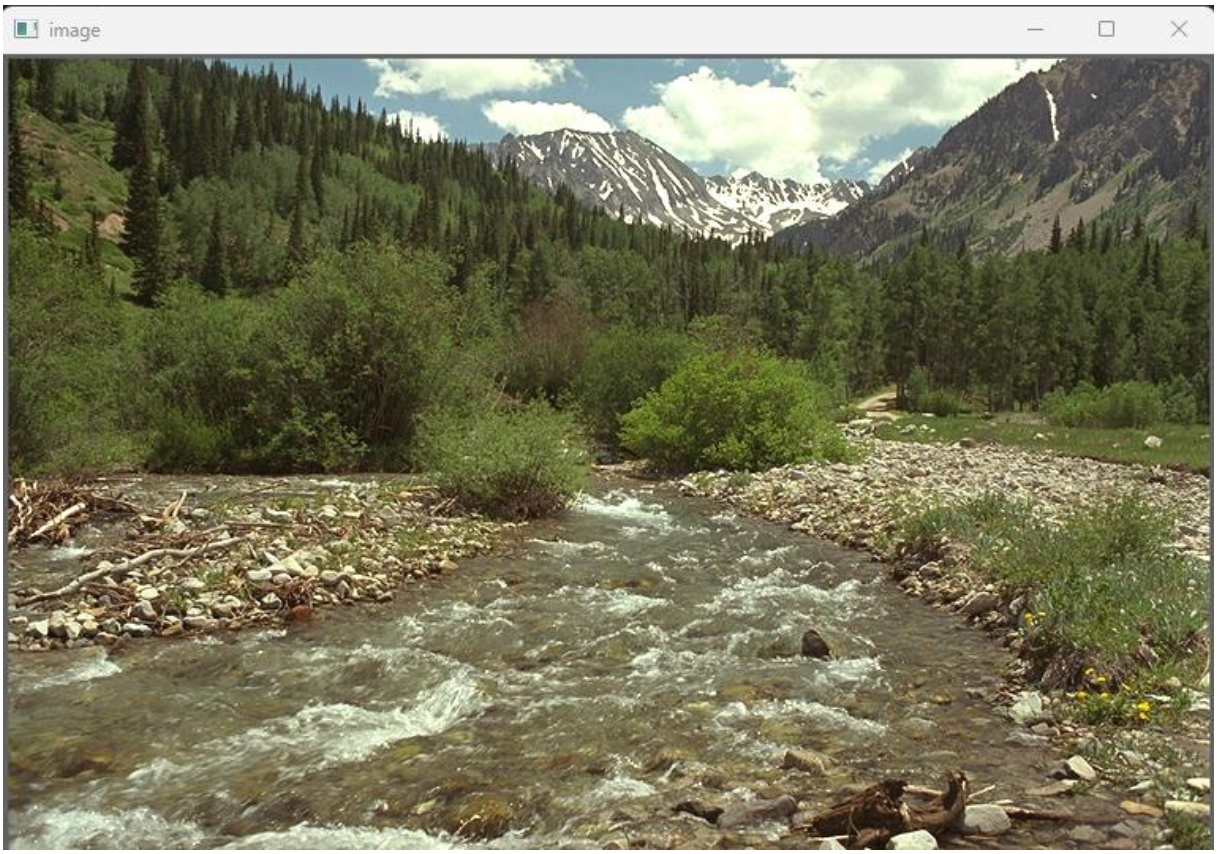
Zad4

bitrate: 6.2425
Entropia obrazu predykcją poziomą: 6.3391
Entropia obrazu oryginalnego: 7.4317
Entropia (LL) = 7.4532
Entropia (LH) = 6.2555
Entropia (HL) = 5.8266
Entropia (HH) = 6.2754
Entropia_śr = 6.4527

Bitrate jest mniejszy od wszystkich wyliczonych entropii poza HL. Nierówność Shannona, mówi nam, że średnia długość słowa kodowego nie może być mniejsza niż entropia źródła w przypadku kompresji bezstratnej. Kiedy bitrate jest mniejszy od entropii, nie oznacza to, że nierówność jest nieprawdziwa. Fakt, że bitrate jest mniejszy niż entropie, nie obala nierówności Shannona; pokazuje to, że kompresja obrazu (w tym przypadku PNG) jest efektywna(stosuje bardziej zaawansowane techniki niż zaprezentowane powyżej) i wykorzystuje własności obrazu pozwalające na skompresowanie go do rozmiaru mniejszego niż wynikałoby to z teoretycznej entropii.

Dla obrazu barwnego

Obraz Wejściowy

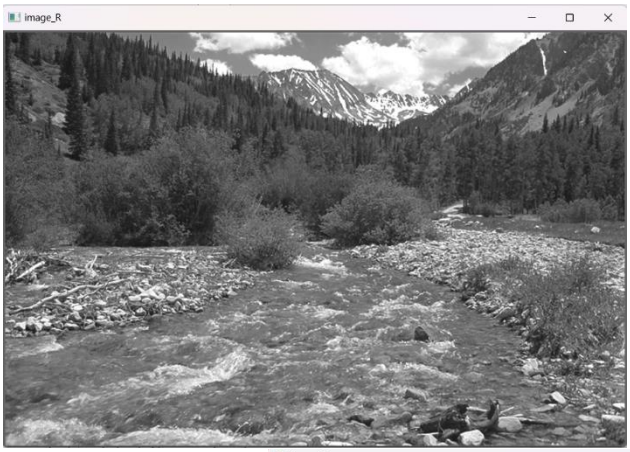


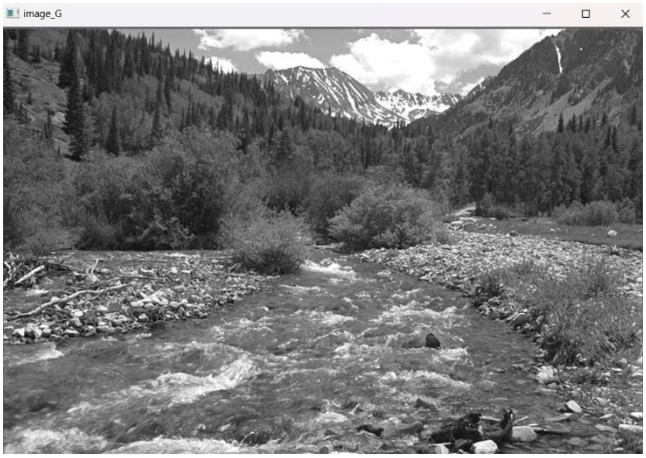
Zad5

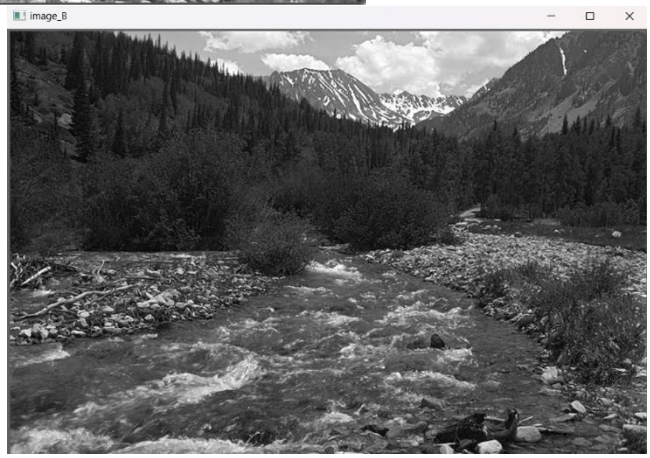
Entropia(R) = 7.4418
Entropia(G) = 7.4274
Entropia(B) = 7.4508
Entropia_śr = 7.4400

Zad6

Składowe dla RGB

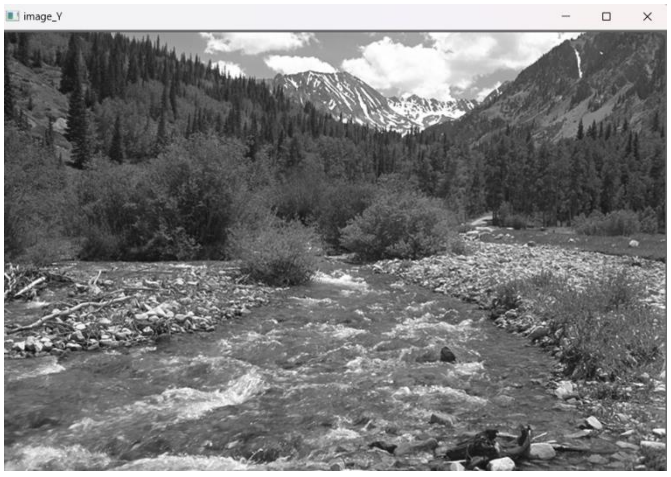


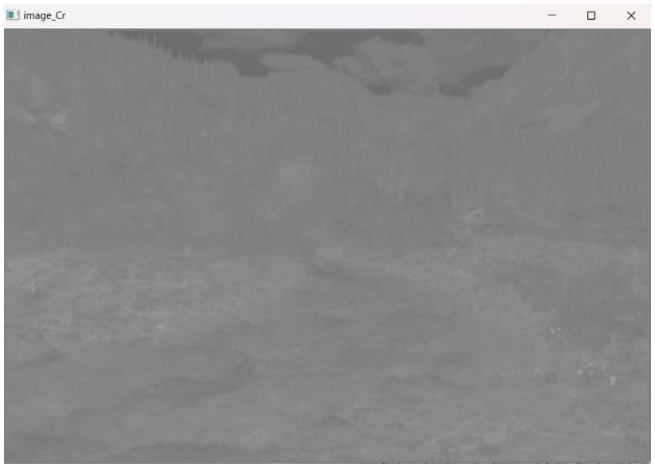




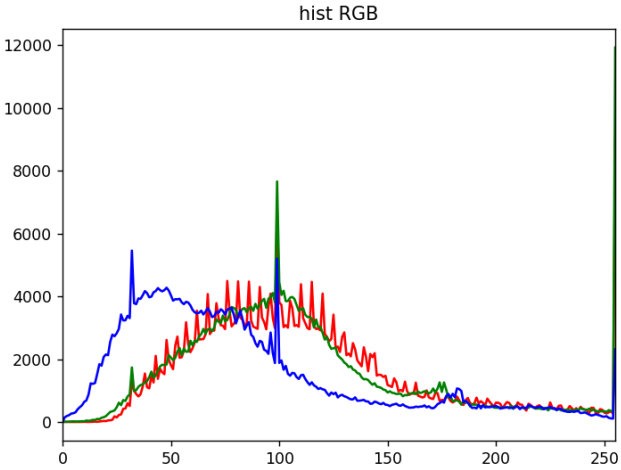
Dokonuje konwersji z RGB do YUV i obliczam entropię dla składowych YUV.

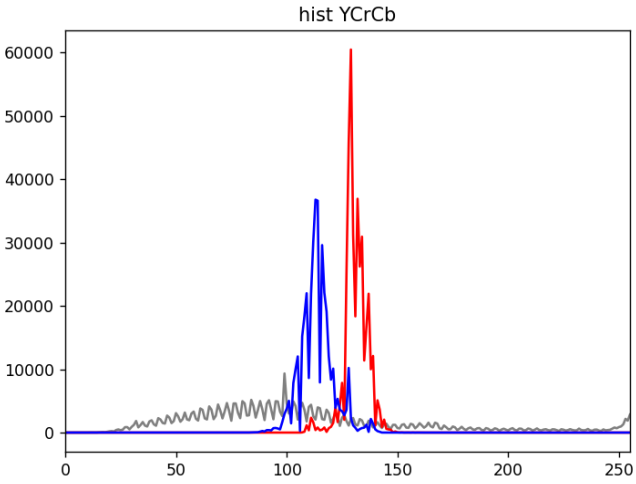
Składowe dla YUV





Histogramy





Y – szary
U – czerwony
V – niebieski

Histogramy RGB: Histogramy dla składowych RGB są rozłożone w szerokim zakresie wartości, co wskazuje na znaczną różnorodność kolorze obrazu.

Histogram YCbCr: Składowe Cb i Cr, które reprezentują chrominancję, mają węższy zakres wartości, co jest typowe, ponieważ te składowe przenoszą informacje o kolorze, który zwykle zmienia się mniej dramatycznie niż jasność.

Entropie

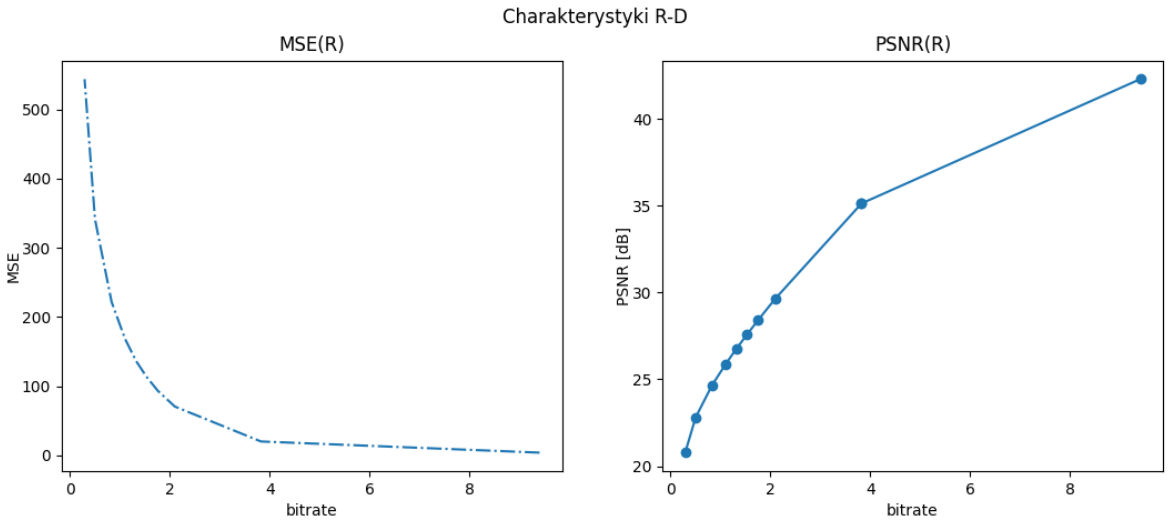
Entropia(Y) = 7.4259
Entropia(Cr) = 4.1125
Entropia(Cb) = 4.7098
Entropia_śr = 5.4161

Entropia składowej Y jest wyższa (7.4259) niż entropie składowych U (4.1125) i V (4.7098). Jest to spodziewane, ponieważ luminancja zazwyczaj zawiera więcej informacji o szczegółach obrazu. Składowe Cb i Cr mają niższą entropię, co

oznacza, że informacje, które przenoszą, są mniej złożone i bardziej przewidywalne. U i V, odpowiadają za kolory. Dlatego też zawierają znacznie mniej informacji od składowej Y.

Zad7

Wyznaczam zależność zniekształcenia D od przepływności R



Ocena

QUALITY	OCENA JAKOŚCI
10	bardzo zła
20	zła
30	zła
40	średnia
50	średnia
60	dobra
70	dobra
80	bardzo dobra
90	doskonała
100	doskonała

Bardzo zła (0, 10>
Zła (10, 30>
Średnia (40, 50>
Dobra (60, 70>
Bardzo dobra (70, 80>
Doskonała (90, 100>

Bitrate kolorowego obrazu PNG: 18.15
Najwyższy bitrate dla JPEG(quality = 100): 9.4

Kompresja JPEG(Quality = 100) w subiektywnym odbiorze nie obniża jakości obrazu a ma wyższy stopień kompresji.

Kod

Kod podstawowy

```
from pathlib import Path
import cv2
import matplotlib.pyplot as plt
import numpy as np
import os


def cv_imshow(img, img_title="image"):
    """
    Funkcja do wyświetlania obrazu w wykorzystaniem okna OpenCV.
    Wykonywane jest przeskalowanie obrazu z rzeczywistymi lub 16-bitowymi
całkowitoliczbowymi wartościami pikseli,
    żeby jedną funkcją wywietlać obrazy różnych typów.
    """
    # cv2.namedWindow(img_title, cv2.WINDOW_AUTOSIZE) # cv2.WINDOW_NORMAL

    if (img.dtype == np.float32) or (img.dtype == np.float64):
        img_ = img / 255
    elif img.dtype == np.int16:
        img_ = img * 128
    else:
        img_ = img
    cv2.imshow(img_title, img_)
    cv2.waitKey(0)


def calc_entropy(hist):
    pdf = hist / hist.sum()
    # entropy = -(pdf*np.log2(pdf)).sum() ### zapis na tablicach, ale problem
z '/0'
    entropy = -sum([x * np.log2(x) for x in pdf if x != 0])
    return entropy


def printi(img, img_title="image"):
    """Pomocnicza funkcja do wypisania informacji o obrazie."""
    print(
        f"{img_title}, wymiary: {img.shape}, typ danych: {img.dtype},
wartości: {img.min()} - {img.max()}"
    )
images_dir = Path("LAB5/monochrome")

img_paths = sorted(list(images_dir.glob("*.png")))

id_number = 325203
```

```
number_of_imgs = len(img_paths)
img_nr = id_number % number_of_imgs
print(number_of_imgs)

img = cv2.imread(str(img_paths[img_nr]))
```

Zad1

```
hist_image = cv2.calcHist([img], [0], None, [256], [0, 256])
hist_image = hist_image.flatten()
print(hist_image.sum(), img.shape[0]*img.shape[1]) ### dla sprawdzenia: suma
wartości histogramu powinna być równa liczbie pikseli w obrazie

H_image = calc_entropy(hist_image)
print(f"Entropia = {H_image:.4f}")
```

Zad2

```
"""
Predykcja w kierunku poziomym:
od wartości danego piksela odejmowana jest wartość piksela z lewej strony -
'lewego sąsiada' (operacje na kolumnach).
Operację taką można wykonać dla pikseli leżących w drugiej i kolejnych
kolumnach obrazu, z pominięciem skrajnie lewej kolumny.
"""
img_tmp1 = img[:, 1:]
img_tmp2 = img[:, :-1]

"""
W wyniku odejmowania pojawią się wartości ujemne - zakres wartości pikseli w
obrazie różnicowym to będzie [-255, 255],
dlatego trzeba zminić typ wartości pikseli, żeby zakres wartości nie
ograniczał się do [0, 255];
może to być np. cv2.CV_16S (odpowiednio np.int16 w NumPy), żeby pozostać w
domenie liczb całkowitych.
"""
image_hdiff = cv2.addWeighted(img_tmp1, 1, img_tmp2, -1, 0, dtype=cv2.CV_16S)
"""
image_hdiff ma o jedną kolumnę mniej - dla skrajnie lewej kolumny nie było
danych do odejmowania,
kolumnę tę można potraktować oddzielnie i 'połączyć' wyniki.
"""
image_hdiff_0 = cv2.addWeighted(img[:, 0], 1, 0, 0, -127,
dtype=cv2.CV_16S).reshape(
    512, 1, 3
)  # Zmiana wymiarów na 3D
image_hdiff = np.hstack((image_hdiff_0, image_hdiff))
printi(image_hdiff, "image_hdiff")

cv_imshow(
```

```
    image_hdiff, "image_hdiff"
)  ### zdefiniowana funkcja pomocnicza odpowiednio 'obsługuje' obrazy z 16-bitowymi wartościami

"""
cv2.calcHist() wymaga danych w formacie liczb całkowitych bez znaku (8- lub 16-bitowych) lub 32-bitowych liczb rzeczywistych,
dlatego wartości pikseli są przesuwane z zakresu [-255, 255] do [0, 510] (-> '+255')
oraz konwertowane na typ np.uint16 (-> astype()).
"""
image_tmp = (image_hdiff + 255).astype(np.uint16)
hist_hdiff = cv2.calcHist([image_tmp], [0], None, [511], [0, 511]).flatten()

hist_image = cv2.calcHist([img], [0], None, [256], [0, 256])
hist_image = hist_image.flatten()
H_image = calc_entropy(hist_image)

""" Wypisanie entropii """
H_hdiff = calc_entropy(hist_hdiff)
print(f"Entropia obrazu predykcją poziomą: {H_hdiff:.4f}")
print(f"Entropia obrazu oryginalnego: {H_image:.4f}")

""" Wyświetlenie histogramów z wykorzystaniem matplotlib.pyplot """
plt.figure()
plt.subplot(1, 2, 1)
plt.plot(hist_image, color="blue")
plt.title("hist_image")
plt.xlim([0, 255])
plt.subplot(1, 2, 2)
plt.plot(
    np.arange(-255, 256, 1), hist_hdiff, color="red"
)  ### jawne podane wartości 'x' i 'y', żeby zmienić opisy na osi poziomej
plt.title("hist_hdiff")
plt.xlim([-255, 255])
plt.show()
```

Zad3

```
def dwt(img):
    """
    Bardzo prosta i podstawowa implementacja, nie uwzględniająca efektywnych metod obliczania DWT
    i dopuszczająca pewne niedokładności.
    """
    maskL = np.array(
        [
            0.02674875741080976,
```

```
            -0.01686411844287795,
            -0.07822326652898785,
            0.2668641184428723,
            0.6029490182363579,
            0.2668641184428723,
            -0.07822326652898785,
            -0.01686411844287795,
            0.02674875741080976,
        ]
    )
    maskH = np.array(
        [
            0.09127176311424948,
            -0.05754352622849957,
            -0.5912717631142470,
            1.115087052456994,
            -0.5912717631142470,
            -0.05754352622849957,
            0.09127176311424948,
        ]
    )

    bandLL = cv2.sepFilter2D(img, -1, maskL, maskL)[::2, ::2]
    bandLH = cv2.sepFilter2D(img, cv2.CV_16S, maskL, maskH)[
        ::2, ::2
    ]  ### ze względu na filtrację górnoprzepustową -> wartości ujemne,
dlatego wynik 16-bitowy ze znakiem
    bandHL = cv2.sepFilter2D(img, cv2.CV_16S, maskH, maskL)[::2, ::2]
    bandHH = cv2.sepFilter2D(img, cv2.CV_16S, maskH, maskH)[::2, ::2]

    return bandLL, bandLH, bandHL, bandHH

ll, lh, hl, hh = dwt(img)
printi(ll, "LL")
printi(lh, "LH")
printi(hl, "HL")
printi(hh, "HH")

cv_imshow(ll, "LL2")
cv_imshow(cv2.multiply(lh, 2), "LH2") ### cv2.multiply() -> zwiększenie
kontrastu obrazów 'H', żeby lepiej uwidocznić
cv_imshow(cv2.multiply(hl, 2), "HL2")
cv_imshow(cv2.multiply(hh, 2), "HH2")

""" Entropia dla obrazów pasmowych """

hist_ll = cv2.calcHist([ll], [0], None, [256], [0, 256]).flatten()
```

```
hist_lh = cv2.calcHist([(lh+255).astype(np.uint16)], [0], None, [511], [0,
511]).flatten() ### zmiana zakresu wartości i typu danych ze względu na
cv2.calcHist() (jak wcześniej przy obrazach różnicowych)
hist_hl = cv2.calcHist([(hl+255).astype(np.uint16)], [0], None, [511], [0,
511]).flatten()
hist_hh = cv2.calcHist([(hh+255).astype(np.uint16)], [0], None, [511], [0,
511]).flatten()
H_ll = calc_entropy(hist_ll)
H_lh = calc_entropy(hist_lh)
H_hl = calc_entropy(hist_hl)
H_hh = calc_entropy(hist_hh)
print(f"Entropia(LL) = {H_ll:.4f} \nEntropia(LH) = {H_lh:.4f} \nEntropia(HL) =
{H_hl:.4f} \nEntropia(HH) = {H_hh:.4f} \nEntropia_śr =
{(H_ll+H_lh+H_hl+H_hh)/4:.4f}")

""" Wyświetlenie histogramów - jeden obraz z czterema pod-obrazami """
fig = plt.figure()
fig.set_figheight(fig.get_figheight()*2) ### zwiększenie rozmiarów okna
fig.set_figwidth(fig.get_figwidth()*2)
plt.subplot(2, 2, 1)
plt.plot(hist_ll, color="blue")
plt.title("hist_ll")
plt.xlim([0, 255])
plt.subplot(2, 2, 3)
plt.plot(np.arange(-255, 256, 1), hist_lh, color="red")
plt.title("hist_lh")
plt.xlim([-255, 255])
plt.subplot(2, 2, 2)
plt.plot(np.arange(-255, 256, 1), hist_hl, color="red")
plt.title("hist_hl")
plt.xlim([-255, 255])
plt.subplot(2, 2, 4)
plt.plot(np.arange(-255, 256, 1), hist_hh, color="red")
plt.title("hist_hh")
plt.xlim([-255, 255])
plt.show()
```

Zad4

```
bitrate = 8 * os.stat(str(img_paths[img_nr])).st_size / (img.shape[0] *
img.shape[1])
print(f"bitrate: {bitrate:.4f}")
```

Zad5

```
images_dir = Path("LAB5/color")

img_paths = sorted(list(images_dir.glob("*.png")))
```

```
id_number = 325203
number_of_imgs = len(img_paths)
img_nr = id_number % number_of_imgs

img = cv2.imread(str(img_paths[img_nr]))
cv_imshow(img)

printi(img, "image_col")

image_R = img[:, :, 2] ### cv2.imread() zwraca obrazy w formacie BGR
image_G = img[:, :, 1]
image_B = img[:, :, 0]

hist_R = cv2.calcHist([image_R], [0], None, [256], [0, 256]).flatten()
hist_G = cv2.calcHist([image_G], [0], None, [256], [0, 256]).flatten()
hist_B = cv2.calcHist([image_B], [0], None, [256], [0, 256]).flatten()

H_R = calc_entropy(hist_R)
H_G = calc_entropy(hist_G)
H_B = calc_entropy(hist_B)
print(f"Entropia(R) = {H_R:.4f} \nEntropia(G) = {H_G:.4f} \nEntropia(B) =
{H_B:.4f} \nEntropia_śr = {(H_R+H_G+H_B)/3:.4f}")

cv_imshow(image_R, "image_R")
cv_imshow(image_G, "image_G")
cv_imshow(image_B, "image_B")
plt.figure()
plt.plot(hist_R, color="red")
plt.plot(hist_G, color="green")
plt.plot(hist_B, color="blue")
plt.title("hist RGB")
plt.xlim([0, 255])
plt.show()
```

Zad6

```
image_YCrCb = cv2.cvtColor(img, cv2.COLOR_BGR2YCrCb) ### albo:
cv2.COLOR_BGR2YUV
printi(image_YCrCb, "image_YCrCb")

hist_Y = cv2.calcHist([image_YCrCb[:, :, 0]], [0], None, [256], [0,
256]).flatten()
hist_Cr = cv2.calcHist([image_YCrCb[:, :, 1]], [0], None, [256], [0,
256]).flatten()
hist_Cb = cv2.calcHist([image_YCrCb[:, :, 2]], [0], None, [256], [0,
256]).flatten()
```

```
H_Y = calc_entropy(hist_Y)
H_Cr = calc_entropy(hist_Cr)
H_Cb = calc_entropy(hist_Cb)
print(f"Entropia(Y) = {H_Y:.4f} \nEntropia(Cr) = {H_Cr:.4f} \nEntropia(Cb) =
{H_Cb:.4f} \nEntropia_śr = {(H_Y+H_Cr+H_Cb)/3:.4f}")

cv_imshow(image_YCrCb[:, :, 0], "image_Y")
cv_imshow(image_YCrCb[:, :, 1], "image_Cr")
cv_imshow(image_YCrCb[:, :, 2], "image_Cb")
plt.figure()
plt.plot(hist_Y, color="gray")
plt.plot(hist_Cr, color="red")
plt.plot(hist_Cb, color="blue")
plt.title("hist YCrCb")
plt.xlim([0, 255])
plt.show()
```

Zad7

```
def calc_mse_psnr(img1, img2):
    """Funkcja obliczająca MSE i PSNR dla różnicy podanych obrazów, zakładana
wartość pikseli z przedziału [0, 255]."""

    imax = 255.0**2  ### maksymalna wartość sygnału -> 255
    """
    W różnicy obrazów istotne są wartości ujemne, dlatego img1 konwertowany
jest do typu np.float64 (liczby rzeczywiste)
    aby nie ograniczać wyniku do przedziału [0, 255].
    """
    mse = (
        (img1.astype(np.float64) - img2) ** 2
    ).sum() / img1.size  ###img1.size - liczba elementów w img1,
==img1.shape[0]*img1.shape[1] dla obrazów mono,
==img1.shape[0]*img1.shape[1]*img1.shape[2] dla obrazów barwnych
    psnr = 10.0 * np.log10(imax / mse)
    return (mse, psnr)

print(8 * os.stat(str(img_paths[img_nr])).st_size / (img.shape[0] *
img.shape[1])) ### bitrate RGB w PNG


xx = []  ### tablica na wartości osi X -> bitrate
ym = []  ### tablica na wartości osi Y dla MSE
yp = []  ### tablica na wartości osi Y dla PSNR

for quality in [
    100,
    90,
```

```
    90,
    70,
    60,
    50,
    40,
    30,
    20,
    10,
]:  ### wartości dla parametru 'quality' należałoby dobrać tak, aby uzyskać 'gładkie' wykresy...
    out_file_name = f"LAB5/out_images/q{quality:03d}.jpg"
    """ Zapis do pliku w formacie .jpg z ustaloną 'jakością' """
    cv2.imwrite(out_file_name, img, (cv2.IMWRITE_JPEG_QUALITY, quality))
    """ Odczyt skompresowanego obrazu, policzenie bitrate'u i PSNR """
    image_compressed = cv2.imread(out_file_name, cv2.IMREAD_UNCHANGED)
    bitrate = (
        8 * os.stat(out_file_name).st_size / (img.shape[0] * img.shape[1])
    )  ### image.shape == image_compressed.shape
    mse, psnr = calc_mse_psnr(img, image_compressed)
    """ Zapamiętanie wyników do późniejszego wykorzystania """
    xx.append(bitrate)
    ym.append(mse)
    yp.append(psnr)

""" Narysowanie wykresów """
fig = plt.figure()
fig.set_figwidth(fig.get_figwidth() * 2)
plt.suptitle("Charakterystyki R-D")
plt.subplot(1, 2, 1)
plt.plot(xx, ym, "-.")
plt.title("MSE(R)")
plt.xlabel("bitrate")
plt.ylabel("MSE", labelpad=0)
plt.subplot(1, 2, 2)
plt.plot(xx, yp, "-o")
plt.title("PSNR(R)")
plt.xlabel("bitrate")
plt.ylabel("PSNR [dB]", labelpad=0)
plt.show()
print(max(xx)) ### MAX bitrate JPEG
cv2.waitKey(0)
```