Laboratorium 5

Bartosz Psik

Numer indeksu : 325 211

Wstępne infomracje

Wybieram obraz testowy na podstawie numeru indeksu, 325211 % 36 = 23 stąd wybieram 23 obraz od góry numerując od 0 i otrzymuję mobile_col.png

Porównanie obrazu kolorowego oraz monochromatycznego

```
show_images(color_img, mono_img)
```

Obraz monochromatyczny

Wyliczam entorpię dla obrazu monochromatycznego

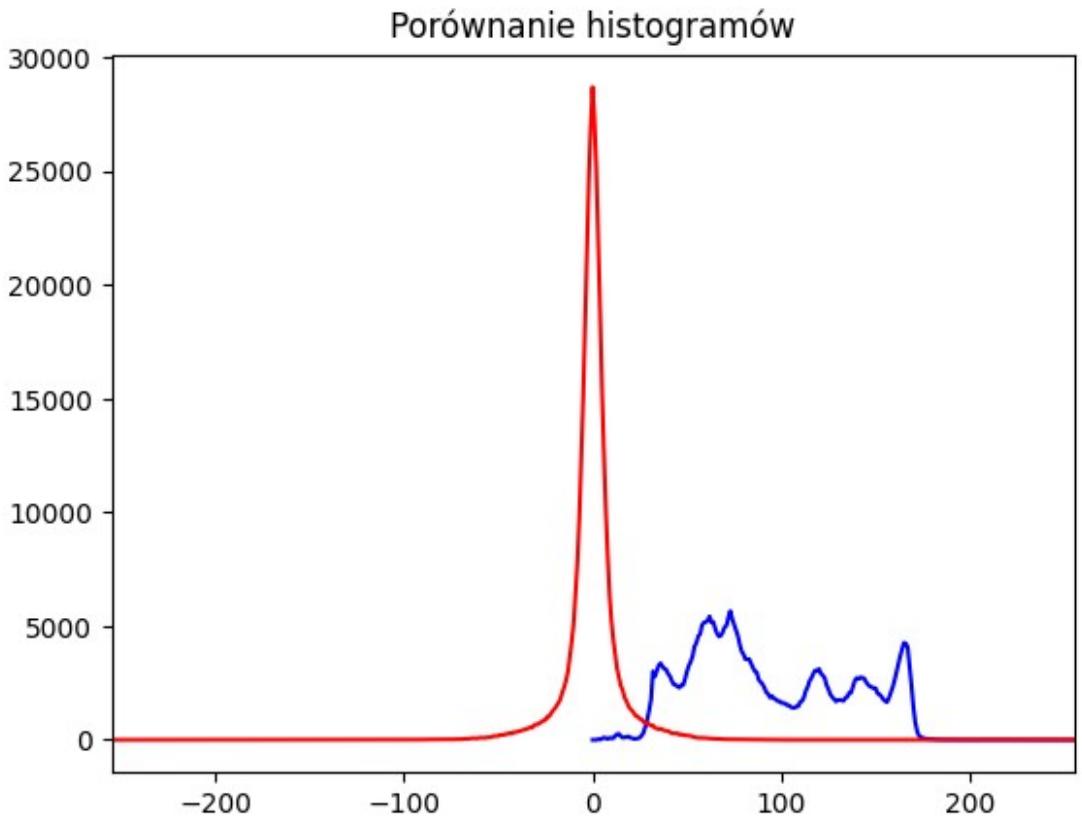
```
mono_entropy(mono_img)
```

```
Entropia obrazu monochromatycznego:  [7.1105666]
```

Wyliczam obraz różnicowy oraz porównuje histogramy oraz entoropie obrazu różnicowego do obrazu monochormochromatycznego

```
show_diff_img()
```


```
show_histograms()
```




```
compare_entropy()

Porównanie entropii obrazu różnicowego i oryginalnego
Entropia obrazu różnicowego:  5.331626374303596
Entropia obrazu:  7.110565223556478
```

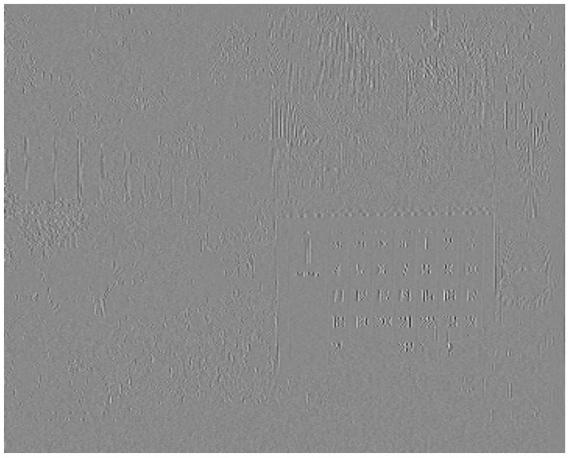
Wyświetlam pasma po transformacji falkowa

```
dwt_images()

LL, wymiary: (288, 360), typ danych: uint8, wartości: 5 - 225
LH, wymiary: (288, 360), typ danych: int16, wartości: -115 - 111
HL, wymiary: (288, 360), typ danych: int16, wartości: -96 - 81
HH, wymiary: (288, 360), typ danych: int16, wartości: -72 - 73
```

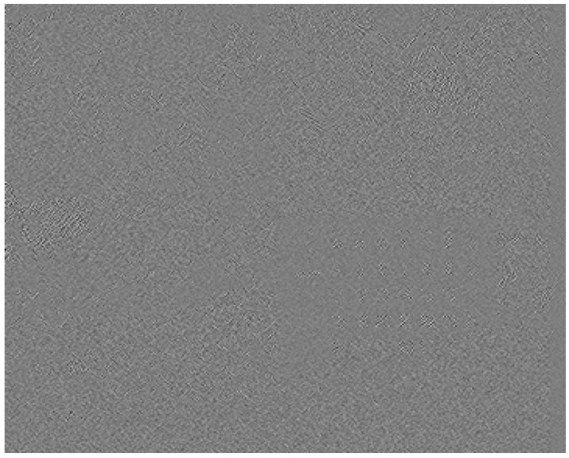
LL

LH

HL



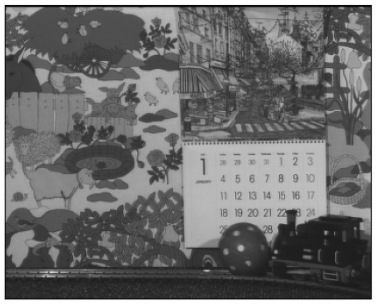
HH

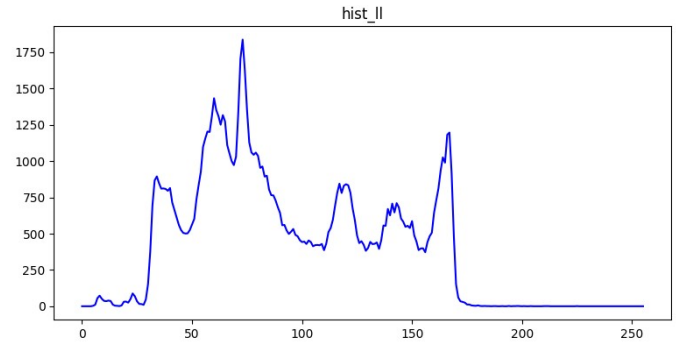


Histogramy poszczególnych pasm

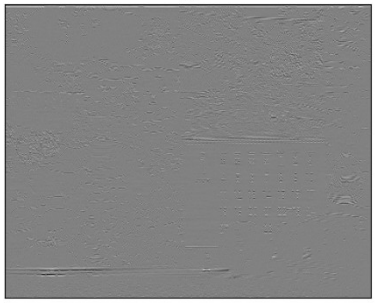
```
show_dwt_hist()
```

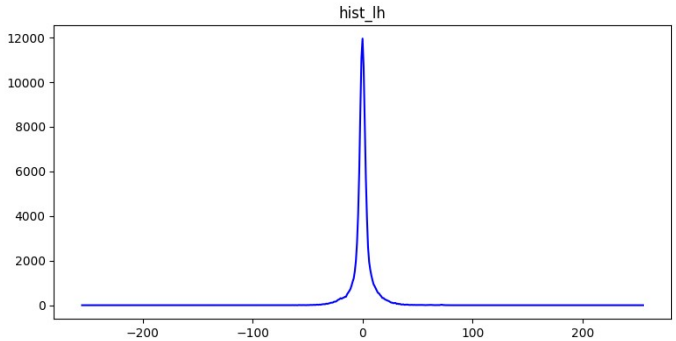
ll



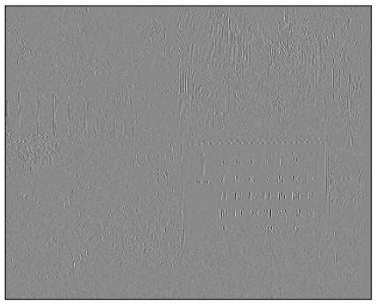


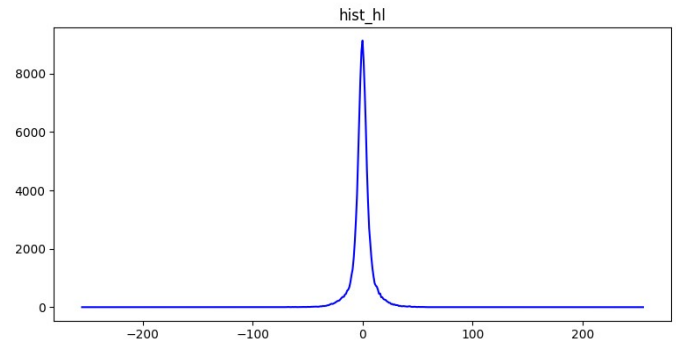
lh



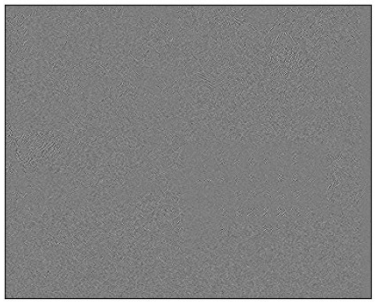


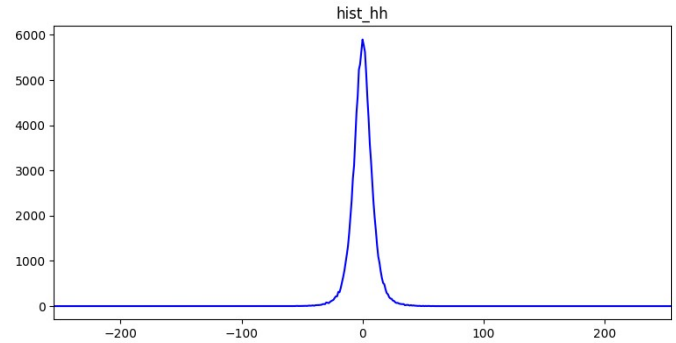
hl





hh





Entorpie poszczególnych pasm

```
show_dwt_hist(entropy=True)

Entropia dla obrazu ll: 7.069824178732233
Entropia dla obrazu lh: 4.71567396701721
Entropia dla obrazu hl: 4.858693136615329
Entropia dla obrazu hh: 5.136381754797185
```

```
Porównanie entropii obrazu różnicowego i oryginalnego
Entropia obrazu różnicowego:  5.331626374303596
Entropia obrazu:  7.110565223556478
```

Przepływowość obrazu monochromatycznego

```
calc_bitrate(mono)
```

```
5.378954475308642
```

Wnioski

Obraz różnicowy

- Na histogramie obrazu różnicowego widać, że większość wartości jest bliska 0, Wynika to z tego, że różnice między sąsiednimi pikselami są niewielkie
- Na histogramie obrazu wejściwoego widać, pełen przedział wartości i ich rozłożenie ze względu na nią
- Porównując entorpie obrazów różnicowego i monochromatycznego widać, że ta dla obrazu różnicowego jest mniejsza. Może to sugerować, że obraz różnicowy jest bardziej jednolity, ma podobniejsze wartości pikseli oraz ma mniej szczegółów. W skócie można poweidzieć, że ejst mniej złożony niż obraz oryginalny
- Jednak obraz różnicowy niesie tyle samo informacji, ponieważ jesteśmy w stanie odtworzyć z niego obraz oryginalny. Ta informacja jest po prostu zdekorelowana
- Entropia jest mniejsza, bo różnicowy niesie informację tam gdzie sąsiednie piksele mają różne wartości, a w różnicowym będą bardziej zbliżone do siebie

Transformacja falkowa

- Po transfomacie falkowa i porównaniu wyników, można stwierdzić, że najbardziej zbliżony do obrazku oryginalnego jest pasmo LL (filtr dolnoprzepustowy użyty dwa razy). Zarówno jego histogram jak i entorpia jest poodbna do obrazu oryginalnenego
- Natomiast obrazki, na których użyto filtru górnoprzepustowego chociaż raz, są bardziej zbliżone do obrazku różnicowego. Zarówno ich histogramy oraz wartości entropii przyjmują podobne wartości. Na histogramie widać duże zagęszczenie przy wartościach 0
- Entropia jest mniejsza, bo sąsiednie piksele mają zbilżoną wartość do siebie, zwłaszcza dla HH, HL , LH

Porównanie przepływowwości do entorpii

- Przepływowość naszego obrazu monochromatycznego wyniosła około 5.38. Ta wartość jest mniejsza niż entropia obrazku monochromatycznego oraz entorpia tego obrazu w pasmie LL. Jest natomiast większa od wartości entorpii dla obrazów, w których chociaż raz wykorzystano filtr górnoprzepustowy : LH, HH, HL
- Jednak przepływność mniejsza od entropii nie oznacza, że zależność $l_{śr} \geq H$ jest nieprawdziwa. W przypadku obrazów w formacie PNG, korzystamy z korelacji między pikselami, co pozwala na bardziej efektywne kodowanie informacji. Przepływność nie

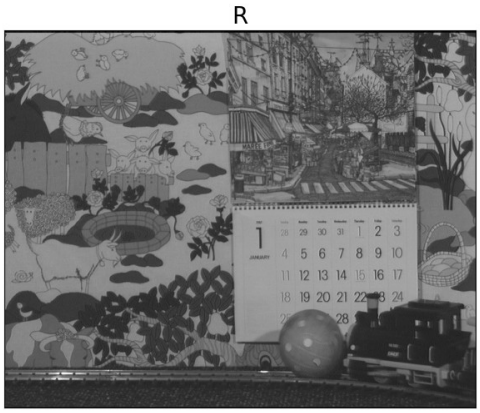
uwzględnia specyfiki kodowania. Entropia natomiast niekoniecznie odzwierciedla efektywność kompresji.

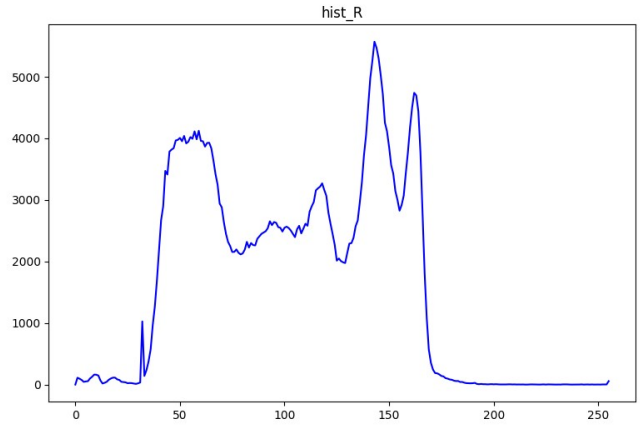
- Przepływowość mniejsza od entropii nie oznacza, że zależność lśr ≥ H jest nieprawdziwa. Korzystając z formatu PNG korzystamy z korelacji między pikselami, dzięki czemu możemu kodować informacje w bardziej efektywny sposób. Nasza przepływowość nie uwzględna w sobie sposobu , tak samo jak entorpia nie musi odzwierciedlać efektywności kompresji

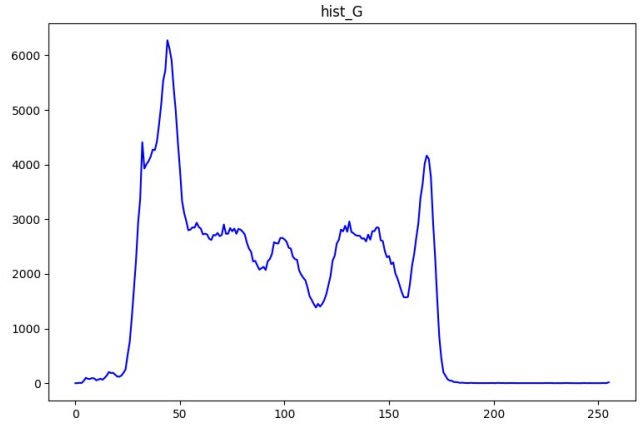
Obraz kolorowy

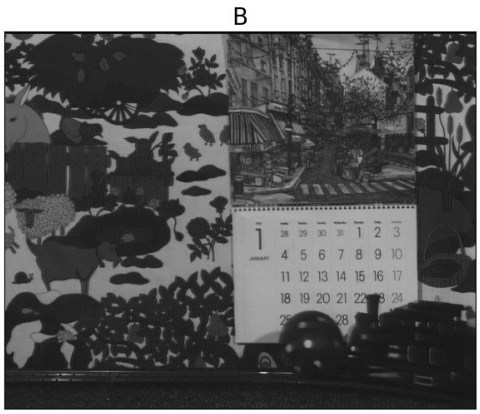
Wyliczenie entropii dla poszczególnych barw RGB oraz przedstawienie ich histogramów

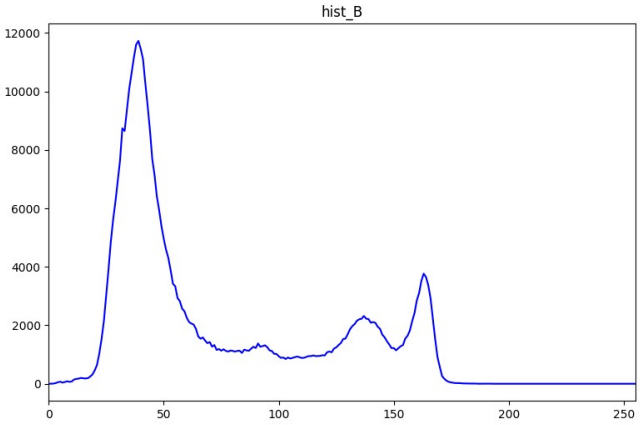
```
show_rgb_enropy()

Entropia dla kanału R: 7.078820897222613
Entropia dla kanału G: 7.189142757291847
Entropia dla kanału B: 6.748955090872187

rgb_histograms()
```







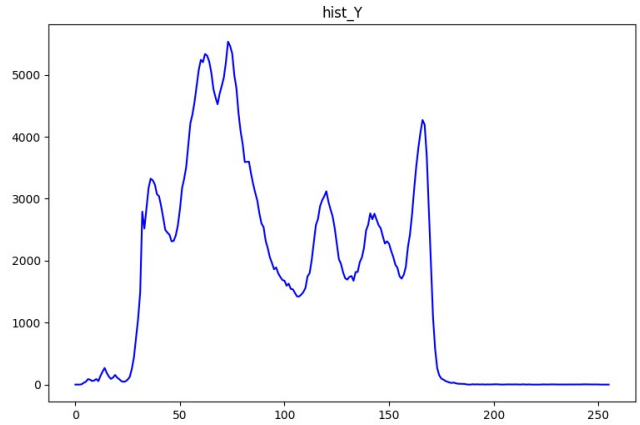




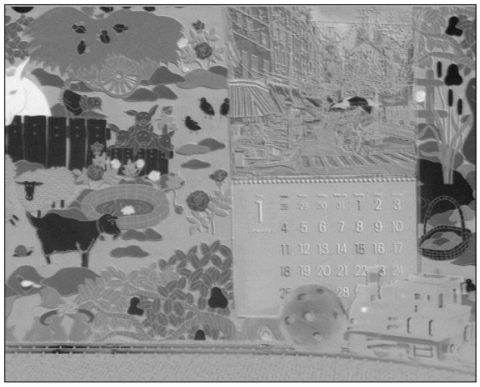
Konwersja do przestrzeni YUV i również wylizcenie entorpii i histogramów dla poszczególnych składowych

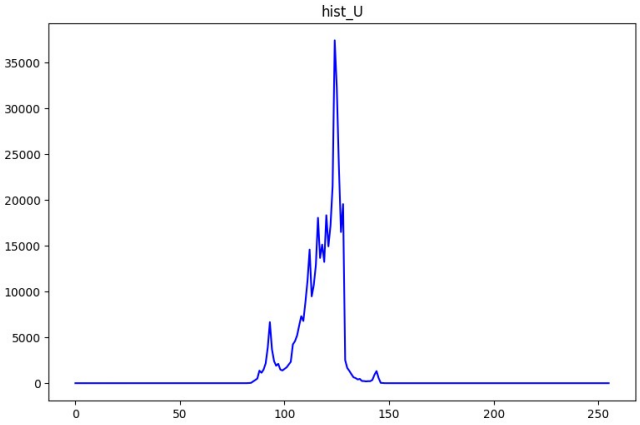
```
yuv_entropy()

Entropia dla kanału Y: 7.11153374890273
Entropia dla kanału U: 5.01073758869461
Entropia dla kanału V: 5.669846977420093

yuv_histograms()
```

Y

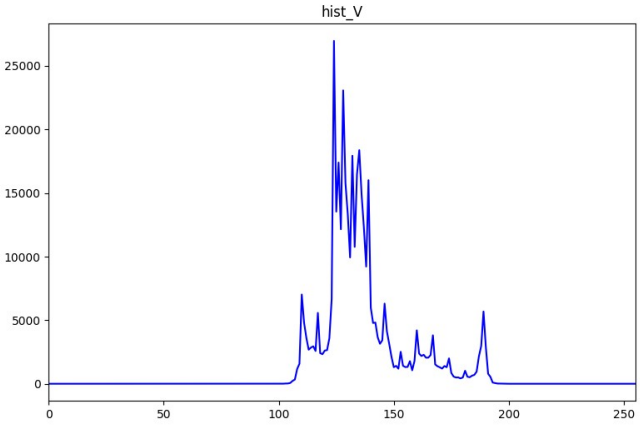


U



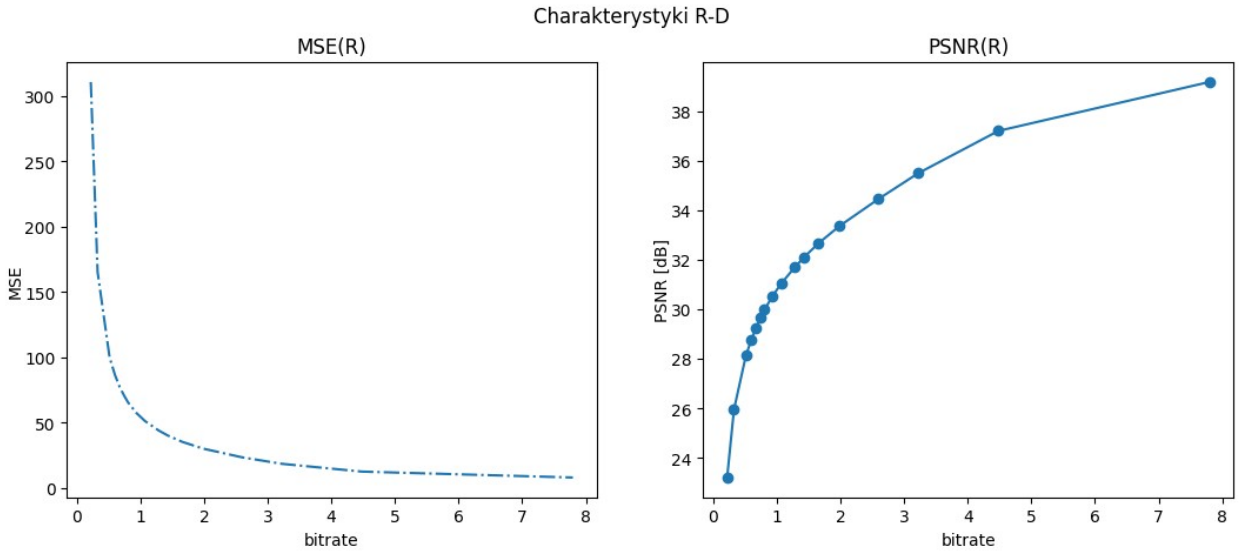


V



Charakterystyka RD

```
show_RD()
```



Porównanie jakości obrazów dla różnych wartości jakości

```
compare_quality()
```

Jakość: 10, Przepływność: 0.33

Jakość: 20, Przepływność: 0.52

Jakość: 30, Przepływność: 0.67

Jakość: 50, Przepływność: 0.93

Jakość: 60, Przepływność: 1.07

Jakość: 75, Przepływność: 1.43

```
compare_bitrate()

Przepływność dla obrazu jpeg/jpeg_q5.jpg: 0.2199266975308642 bpp
Przepływność dla obrazu jpeg/jpeg_q10.jpg: 0.3272569444444444 bpp
Przepływność dla obrazu jpeg/jpeg_q20.jpg: 0.5152199074074074 bpp
Przepływność dla obrazu jpeg/jpeg_q25.jpg: 0.5994405864197531 bpp
Przepływność dla obrazu jpeg/jpeg_q30.jpg: 0.674903549382716 bpp
Przepływność dla obrazu jpeg/jpeg_q35.jpg: 0.7487268518518518 bpp
Przepływność dla obrazu jpeg/jpeg_q40.jpg: 0.8077739197530864 bpp
Przepływność dla obrazu jpeg/jpeg_q50.jpg: 0.9344907407407408 bpp
Przepływność dla obrazu jpeg/jpeg_q60.jpg: 1.071855709876543 bpp
Przepływność dla obrazu jpeg/jpeg_q70.jpg: 1.2867476851851851 bpp
Przepływność dla obrazu jpeg/jpeg_q75.jpg: 1.431230709876543 bpp
Przepływność dla obrazu jpeg/jpeg_q80.jpg: 1.6563850308641976 bpp
Przepływność dla obrazu jpeg/jpeg_q85.jpg: 1.9847029320987655 bpp
Przepływność dla obrazu jpeg/jpeg_q90.jpg: 2.598688271604938 bpp
Przepływność dla obrazu jpeg/jpeg_q93.jpg: 3.2237654320987654 bpp
Przepływność dla obrazu jpeg/jpeg_q96.jpg: 4.482600308641976 bpp
Przepływność dla obrazu jpeg/jpeg_q100.jpg: 7.795756172839506 bpp
Przepływność dla obrazu color/mobile_col.png: 15.141415895061728 bpp
```

Wnioski

RGB i **YUV**

- Wszystkie wartości entorpii dla poszczególnych barw mają wartości zbliżone do siebie i do entorpii oryginalnego obrazka. Najabrdziej widać do na przykładzie składowej R, która jest największa oraz najbliższa tym wartościom
- Na histogramach składowych RGB widać, że wartości są w miare rozproszone na całą szerokość, co oznacza że wartości pikseli dla skłądowych w RGB są zróżnicowane. Widać, to zwłaszcza jak porównamy te wartości do składowcyh YUV, gdzie widać duże zbicie, zwłaszcza dla składowych U i V, które skupiają się głównie w jednym obszarze
- Porównując entropie można zauważyć, że te dla YUV są mniejsze od tych w RGB. Tylko wartość składowej Y jest zbliżona. Można z tego wywnioskować, że informacyjna wartość niesiona w YUV jest bardziej skoncentrowana a w wartościach chrominancji jest niesionych mniej informacji niż w wartościach luminancji. Chrominancja ma mniejsze zróżnicowanie niż luminancja, dlatego entorpia dla U i V jest mniejsza niż dla Y

Porównanie JPEG do PNG

- Na pierwszy rzut oka, patrząc na wartości przepływności obrazów JPEG oraz obrazu PNG widać, że nawet dla ajlepszej jakości obrazu JPEG ta wartość jest znacznie mniejsza niż dla obrazu PNG. W naszym przypadku jest to wartość prawei 2x mniejsza
- Wynika to z tego, że obrazy PNG korzystają z kompresji bezstratnej, której nie mamy w obrazach JPEG
- JPEG natomiast oferuje nam mniejszy rozmiar, ale przez mniejszą przepływowość można na nim zauważyć różne zniekształcenia, które znacząco mogą wpływnąć na jakość obrazka

- Dla JPEG zwiększając jakość obrazu wzrasta również przepustowość, co skutkuje wzrostem jakości obrazu, poniważ zwiększa się ilość bitów przypadających na jedn piksel obrazu
- Dla jakości mniejszych niż 30 obrazy są najgorszej jakości. Bardzo mocno widać na nich różne zniekształcenia, przez co przestają być one ostre, a są wręcz rozmyte. Jest to bardzo zła jakość obrazu
- Dla jakości z zakresu 50 - 75 widać poprawę jakości. Trzeba się przyjrzeć obrazkowi żeby zobaczyć jakieś zniekształcenia. Oczywiście im bardziej powiększymy obraz tym bardziej będzie dostrzegalna słaba jakość obrazku. Obszary zniekształcone są mniejsze na pierwszy rzut oka. Obrazy z tej jakości można nazwać dobrymi
- Dając największą jakość czyli 90+ można powiedzieć, że obrazki są ładne. Ich przepustowość jest w miarę wysoka. Można dostrzec szczegóły obrazków i nie widać pojeduyńczych pikseli jak przy jakości mniejszej niż 30. Takie obrazki należą do grupy bardzo dobrej jakości, jednak dalej poziomem jakości nie dorównują obrazkom w formacie PNG

Przydatne funkcje oraz biblioteki

```
# użyte biblioteki
import numpy as np
import os
import cv2
import matplotlib.pyplot as plt

# wyliczenie obrazka
list_of_images = os.listdir("color")
number = 325211%36
name = list_of_images[number]
print("Nazwa obrazu :", name, "\nNumer obrazu: ", number)


Nazwa obrazu : mobile_col.png
Numer obrazu:  23

# ładowanie obrazków -> zakładamy że folder z obrazkami, znajduje się
w tym samym folderze co skrypt
png_name = "mobile"

color = "color/" + png_name + "_col.png"
mono = "monochrome/" + png_name + "_mono.png"

# przydatne funckje
def load_img(path_to_file):
    img = cv2.imread(path_to_file, cv2.IMREAD_COLOR)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    return img
```

```
def load_mono_img(path_to_file):
    img = cv2.imread(path_to_file, cv2.IMREAD_GRAYSCALE)

    return img

# liczenie entropi
def calc_entropy(hist):

    pdf = hist/hist.sum()
    entropy = -sum([x*np.log2(x) for x in pdf if x != 0])
    return entropy

# funckja do wyświetlania obrazu
def plt_imshow(img, img_title="image"):
    if img.dtype == np.int32:
        img = (img + 255) // 2

    plt.figure()
    plt.title(img_title)
    plt.imshow(img, cmap="gray", vmin=0, vmax=255)
    plt.xticks([]), plt.yticks([])
    plt.show()

def normalize(img):
    return ((img / max(img.max(), -img.min())+ 1.0) *
127.5).astype(np.uint8)

# obrazki
color_img = load_img(color)
mono_img = load_mono_img(mono)
```

Kod do obrazu monochromatycznego

```
# pokazanie obrazków
def show_images(color, monochorome):
    plt.figure(figsize=(20, 20))
    plt.subplot(1, 2, 1)
    plt.imshow(color, cmap="gray")
    plt.title("Originalny")
    plt.axis("off")

    plt.subplot(1, 2, 2)
    plt.imshow(monochorome, cmap="gray",vmin=0, vmax=255)
    plt.title("Monochoamtyczny")
    plt.axis("off")

    plt.show()
```

```python
# liczenie entropii
def mono_entropy(img):
        hist = cv2.calcHist([img], [0], None, [256], [0, 256])
        mono_entropy = calc_entropy(hist)
        print("Entropia obrazu monochromatycznego: ", mono_entropy)

mono_entropy(mono_img)
mono_entropy(color_img)
```

```
Entropia obrazu monochromatycznego:  [7.1105666]
Entropia obrazu monochromatycznego:  [7.0788207]
```

```python
# liczenie obrazu różnicowego
def calc_hdiff(img_mon):
    img_hdiff_mono = cv2.addWeighted(img_mon[:, 0], 1, 0, 0, -127,
dtype=cv2.CV_16S)
    img_hdiff_mono = cv2.addWeighted(img_mon[:, 1:], 1, img_mon[:, :-
1], -1, 0, dtype=cv2.CV_16S)


    return img_hdiff_mono

# pokazanie obrazu różnicowego
def show_diff_img():
    img_diff = calc_hdiff(mono_img)
    plt.figure()
    plt.imshow(img_diff, cmap="gray")
    plt.title("Obraz różnicowy")
    plt.axis("off")
    plt.show()

# porównanie histogramów
def show_histograms():
    mono_hdiff = calc_hdiff(mono_img)
    mono_hist = cv2.calcHist([mono_img], [0], None, [256], [0,
256]).flatten()

    image_tmp = (mono_hdiff+255).astype(np.uint16)
    hist_hdiff = cv2.calcHist([image_tmp], [0], None, [511], [0,
511]).flatten()


    plt.figure()
    plt.plot(mono_hist, color="blue")
    plt.title("Porównanie histogramów")
    plt.xlim([0, 255])
    plt.plot(np.arange(-255, 256, 1), hist_hdiff, color="red")
    plt.xlim([-255, 255])
    plt.show()
```

```
# porónanie entorpii różnicowego i oryginalnego
def compare_entropy():
    mono_hdiff = calc_hdiff(mono_img)
    mono_hist = cv2.calcHist([mono_img], [0], None, [256], [0,
256]).flatten()
    mono_entropy = calc_entropy(mono_hist)

    image_tmp = (mono_hdiff+255).astype(np.uint16)
    hist_hdiff = cv2.calcHist([image_tmp], [0], None, [511], [0,
511]).flatten()
    diff_entropy = calc_entropy(hist_hdiff)



    print("Porównanie entropii obrazu różnicowego i oryginalnego")
    print("Entropia obrazu różnicowego: ", diff_entropy)
    print("Entropia obrazu: ", mono_entropy)

# DWT
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()}")

def dwt(img):

    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


def dwt_images():

    image = cv2.imread(mono, cv2.IMREAD_GRAYSCALE)

    ll, lh, hl, hh = dwt(image)
```

```
    printi(ll, "LL")
    printi(lh, "LH")
    printi(hl, "HL")
    printi(hh, "HH")

    # w celu poprawienia widoczności, zwiększamy kontrast obrazów 'H'
    lh = cv2.multiply(lh, 2)
    hl = cv2.multiply(hl, 2)
    hh = cv2.multiply(hh, 2)

    fig, axs = plt.subplots(2, 2, figsize=(20, 20))
    for ax in axs.flat:
        ax.axis("off")

    axs[0, 0].imshow(ll, cmap="gray")
    axs[0, 0].set_title("LL", fontsize=20)
    axs[0, 1].imshow(lh, cmap="gray")
    axs[0, 1].set_title("LH", fontsize=20)
    axs[1, 0].imshow(hl, cmap="gray")
    axs[1, 0].set_title("HL", fontsize=20)
    axs[1, 1].imshow(hh, cmap="gray")
    axs[1, 1].set_title("HH", fontsize=20)
    plt.show()

# wyliczenie histogramów
def show_dwt_hist(entropy=False):
    image = cv2.imread(mono, cv2.IMREAD_GRAYSCALE)
    ll, lh, hl, hh = dwt(image)

    # entropy dla obrazów po DWT
    img = [(ll, "ll"), (lh, "lh"), (hl, "hl"), (hh, "hh")]

    # histogramy
    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()

    histograms = [hist_ll, hist_lh, hist_hl, hist_hh]

    if not entropy:
        # wyświtlenie obrazu po DWT
        fig, axs = plt.subplots(4, 2, figsize=(20, 20))
        for i in range(4):
            axs[i,0].imshow(img[i][0], cmap="gray")
```

```
            axs[i,0].set_title(img[i][1], fontsize=20)
            axs[i][0].set_xticks([])
            axs[i][0].set_yticks([])

        # wyświetlenie histogramów
        for i in range(4):
            if i == 0:
                axs[i, 1].plot(histograms[i], color="blue")
                plt.xlim([0, 255])
            else:
                axs[i, 1].plot(np.arange(-255, 256, 1), histograms[i],
color="blue")
                plt.xlim([-255, 255])
            axs[i, 1].set_title(f"hist_{img[i][1]}")
        plt.show()

    if entropy :
        entropies = [calc_entropy(h) for h in histograms]

        for i in range(4):
            print(f"Entropia dla obrazu {img[i][1]}: {entropies[i]}")

        compare_entropy()
```

Kod do obrazu kolorowego

```
# on już jest w RGB
image = load_img(color)

image_R = image[:, :, 0] # wybieramy czerwony kanał
image_G = image[:, :, 1] # wybieramy zielony kanał
image_B = image[:, :, 2] # wybieramy niebieski kanał

# obliczamy histogramy dla każdego kanału

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()

histograms_rgb = [hist_R, hist_G, hist_B]

# obliczamy entropie dla każdego kanału
r_entropy = calc_entropy(hist_R)
g_entropy = calc_entropy(hist_G)
b_entropy = calc_entropy(hist_B)

entropies_rgb = [(r_entropy, "R"), (g_entropy, "G"), (b_entropy, "B")]

# funckja do pokazywania entorpii
```

```python
def show_rgb_enropy():
    for e in entropies_rgb:
        print(f"Entropia dla kanału {e[1]}: {e[0]}")

# pokazywanie histogramów rgb
def rgb_histograms():
    fig, axs = plt.subplots(3, 2, figsize=(20, 20))
    for i in range(3):
        axs[i, 0].imshow(image[:, :, i], cmap="gray")
        axs[i, 0].set_title(entropies_rgb[i][1], fontsize=20)
        axs[i, 0].set_xticks([])
        axs[i, 0].set_yticks([])

    for i in range(3):
        axs[i, 1].plot(histograms_rgb[i], color="blue")
        plt.xlim([0, 255])
        axs[i, 1].set_title(f"hist_{entropies_rgb[i][1]}")

# konwersja do YUV
image_YUV = cv2.cvtColor(image, cv2.COLOR_RGB2YUV)

hist_Y = cv2.calcHist([image_YUV[:, :, 0]], [0], None, [256], [0,
256]).flatten()
hist_U = cv2.calcHist([image_YUV[:, :, 1]], [0], None, [256], [0,
256]).flatten()
hist_V = cv2.calcHist([image_YUV[:, :, 2]], [0], None, [256], [0,
256]).flatten()
histograms_yuv = [hist_Y, hist_U, hist_V]

entropy_Y = calc_entropy(hist_Y)
entropy_U = calc_entropy(hist_U)
entropy_V = calc_entropy(hist_V)

entropies_yuv = [(entropy_Y,"Y"), (entropy_U, "U"), (entropy_V, "V")]

def yuv_entropy():
    for e in entropies_yuv:
        print(f"Entropia dla kanału {e[1]}: {e[0]}")

def yuv_histograms():
    fig, axs = plt.subplots(3, 2, figsize=(20, 20))
    for i in range(3):
        axs[i, 0].imshow(image_YUV[:, :, i], cmap="gray")
        axs[i, 0].set_title(entropies_yuv[i][1], fontsize=20)
        axs[i, 0].set_xticks([])
        axs[i, 0].set_yticks([])

    for i in range(3):
        axs[i, 1].plot(histograms_yuv[i], color="blue")
```

```
        plt.xlim([0, 255])
        axs[i, 1].set_title(f"hist_{entropies_yuv[i][1]}")

# charakterystyka RD
def calc_mse_psnr(img1, img2):
    imax = 255.**2 ### maksymalna wartość sygnału -> 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)


def show_RD():
    xx = [] ### tablica na wartości osi X -> bitrate
    ym = [] ### tablica na wartości osi Y dla MSE
    yp = [] ### tablica na wartości osi Y dla PSNR

    qualities = [5, 10, 20, 25, 30, 35, 40, 50, 60, 70, 75, 80, 85,
90, 93, 96, 100]

    color_img = load_img(color)

    for quality in qualities:
        out_file = f"jpeg/jpeg_q{quality}.jpg"
        cv2.imwrite(out_file, color_img, [cv2.IMWRITE_JPEG_QUALITY,
quality])

        image_compressed = cv2.imread(out_file, cv2.IMREAD_COLOR)
        img_bitrate =
8*os.stat(out_file).st_size/(image_compressed.shape[0]*image_compresse
d.shape[1])
        mse, psnr = calc_mse_psnr(color_img, image_compressed)

        xx.append(img_bitrate)
        ym.append(mse)
        yp.append(psnr)

    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()

# porównanie jakości
def compare_quality():
    rated_imgs = [10, 20, 30, 50, 60, 75, 90, 100]

    fig, axs = plt.subplots(len(rated_imgs)//2, 2, figsize=(10, 20))

    for idx, quality in enumerate(rated_imgs):
        src = f"jpeg/jpeg_q{quality}.jpg"
        img = cv2.imread(src, cv2.IMREAD_COLOR)
        img_bitrate =
8*os.stat(src).st_size/(img.shape[0]*img.shape[1])


        cur_axis = axs[idx//2][idx%2]
        cur_axis.imshow(img)
        cur_axis.set_title(f"Jakość: {quality}, Przepływność:
{img_bitrate:.2f}")
        cur_axis.set_xticks([])
        cur_axis.set_yticks([])

    plt.gcf().set_dpi(250)
    plt.show()

def calc_bitrate(img_name):
    file_stats = os.stat(img_name)
    file_size = file_stats.st_size  # Access the file size attribute
    bitrate = 8 * file_size / (image.shape[0] * image.shape[1])
    return bitrate

def compare_bitrate():
    qualities = [5, 10, 20, 25, 30, 35, 40, 50, 60, 70, 75, 80, 85,
90, 93, 96, 100]
    # Porównanie przeplywowości dla różnych jakości
    for quality in qualities:
        out_file = f"jpeg/jpeg_q{quality}.jpg"
        cv2.imwrite(out_file, color_img, [cv2.IMWRITE_JPEG_QUALITY,
quality])

        image_compressed = cv2.imread(out_file, cv2.IMREAD_COLOR)
        img_bitrate =
8*os.stat(out_file).st_size/(image_compressed.shape[0]*image_compresse
d.shape[1])
        print(f"Przepływność dla obrazu {out_file}: {img_bitrate}
bpp")
```

```
    bitrate = calc_bitrate(color)
    print(f"Przepływność dla obrazu {color}: {bitrate} bpp")
```