Szymon Dyszewski
Numer albumu: 310625

Sprawozdanie z laboratorium nr 5 z przedmiotu WMM

Zdjęcie nr 17

Szymon Dyszewski
Numer albumu: 310625

Przepływność i entropia

```
def bitratecalculate(image_path):
    bitrate = 8*os.stat(image_path).st_size/(mono_img.shape[0]*mono_img.shape[1])
    print(f"Bitrate: {bitrate:.4f}")
bitratecalculate(mono_path)


def enthropycalculate(hist):
    pdf = hist/hist.sum() ### normalizacja histogramu -> rozkład prawdopodobieństwa;
    # 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
hist_mono = cv2.calcHist([mono_img], [0], None, [256], [0, 256]).flatten()
entropy_mono = enthropycalculate(hist_mono)
print(f"Entropy: {entropy_mono:.4f}")
```

Bitrate: 5.1694

Entropy: 7.1887

Czy przepływność mniejsza od entropii oznacza, że zależność: $l_{śr} \geq H$ jest nieprawdziwa?

Nie. Format PNG jest tworzony na podstawie kompresji wykorzysującej skorelowanie pikseli, a powyższa zależność zakłada brak takiej korelacji.

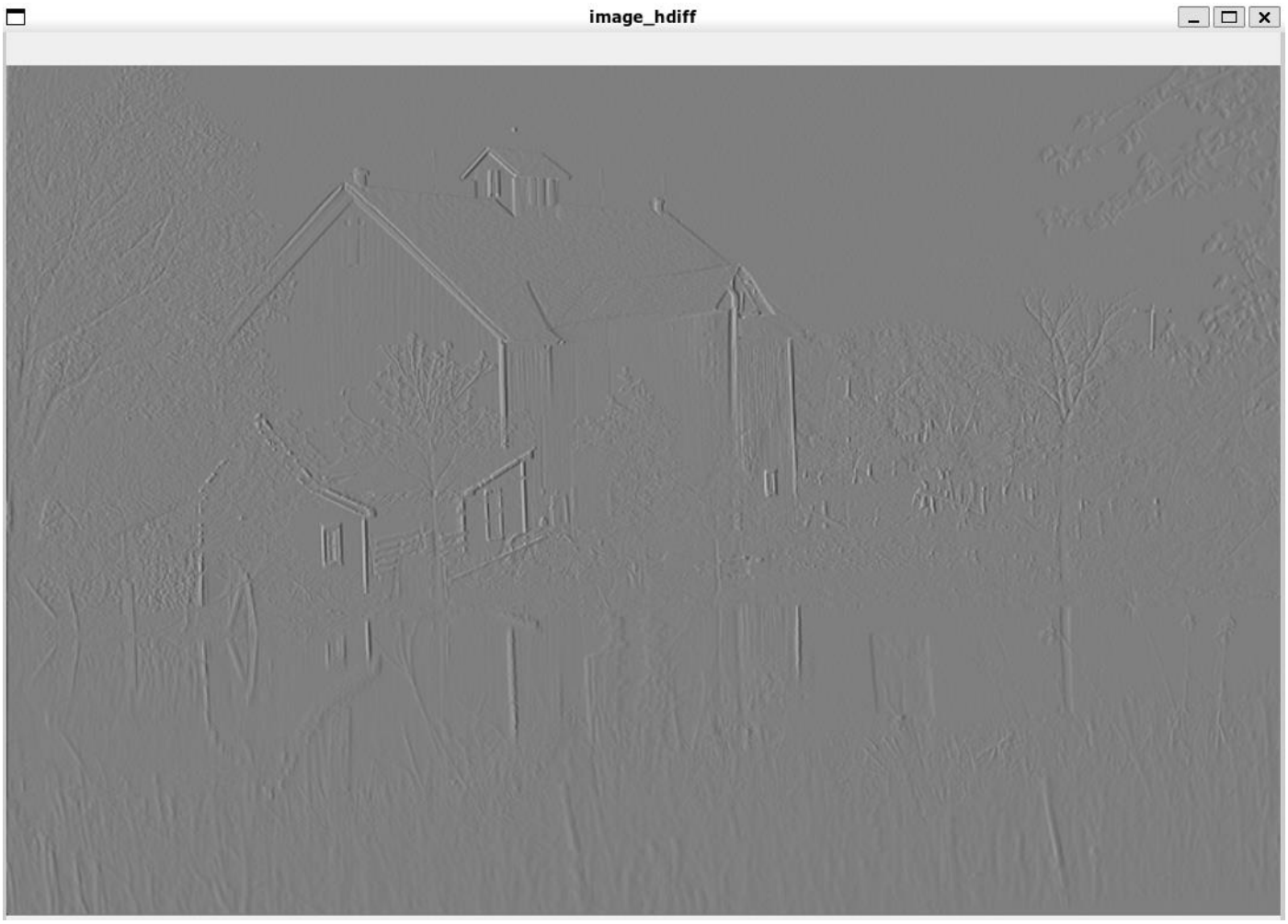
Obraz różnicowy

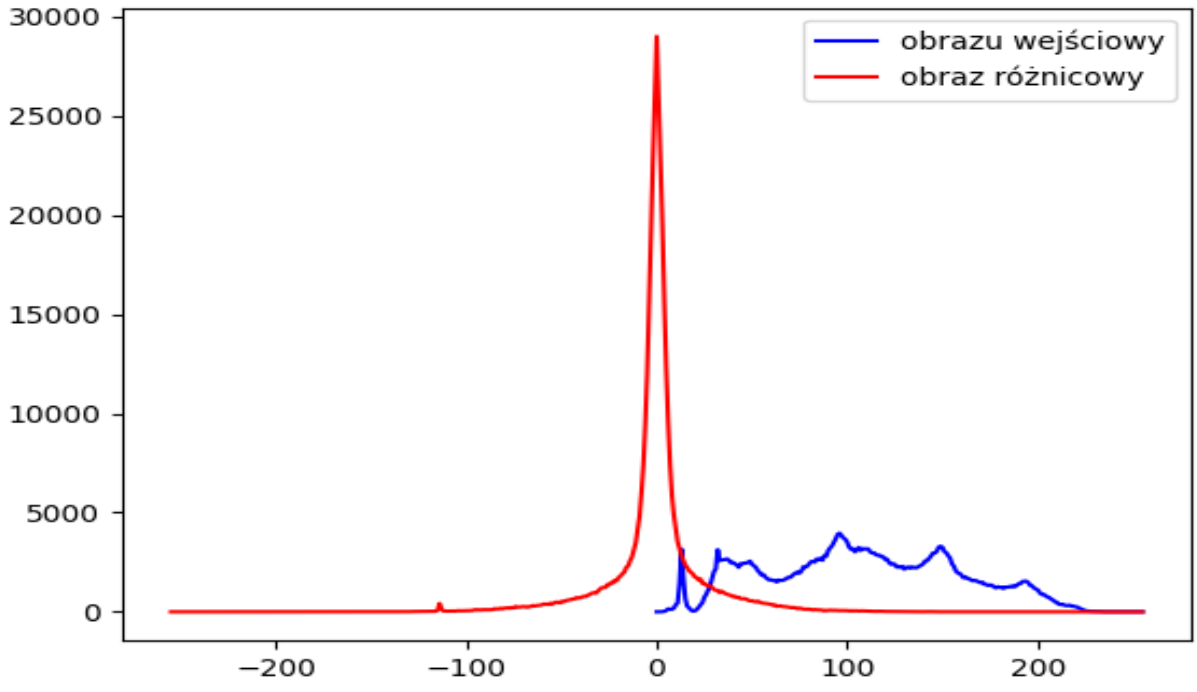
```
def hdiff():
    img_tmp1 = mono_img[:, 1:]
    img_tmp2 = mono_img[:, :-1]
    image_hdiff = cv2.addWeighted(img_tmp1, 1, img_tmp2, -1, 0, dtype=cv2.CV_16S)
    image_hdiff_0 = cv2.addWeighted(mono_img[:, 0], 1, 0, 0, -127, dtype=cv2.CV_16S) ### od 'zerowej' kolumny
    image_hdiff = np.hstack((image_hdiff_0, image_hdiff)) ### połączenie tablic w kierunku poziomym, czyli 'ko
    printi(image_hdiff, "image_hdiff")
    cv_imshow(image_hdiff, "image_hdiff")
    image_hdiff = cv2.calcHist([(image_hdiff + 255).astype(np.uint16)], [0], None, [511], [0, 511]).flatten()
    plt.plot(np.arange(0, 256), hist_mono, color="blue", label="obrazu wejściowy")
    plt.plot(np.arange(-255, 256), image_hdiff, color="red", label="obraz różnicowy")
    plt.legend()
    plt.gcf().set_dpi(150)
    plt.show()
    entropy_hdiff = enthropycalculate(image_hdiff)
    print(f"Entropy: mono {entropy_mono:.4f} Entropy: hdiff {entropy_hdiff:.4f}")
hdiff()
```

Entropy: mono 7.1887 Entropy: hdiff 5.2565

Entropia obrazu różnicowego jest niższa od entropii obrazu pierwotnego, ponieważ obraz różnicowy posiada o wiele mniejszy zakres informacji niż wejściowy. Widać to na histogramie, gdzie zdecydowana większość pikseli obrazu różnicowego znajduje się w okolicy zera.

Szymon Dyszewski
Numer albumu: 310625



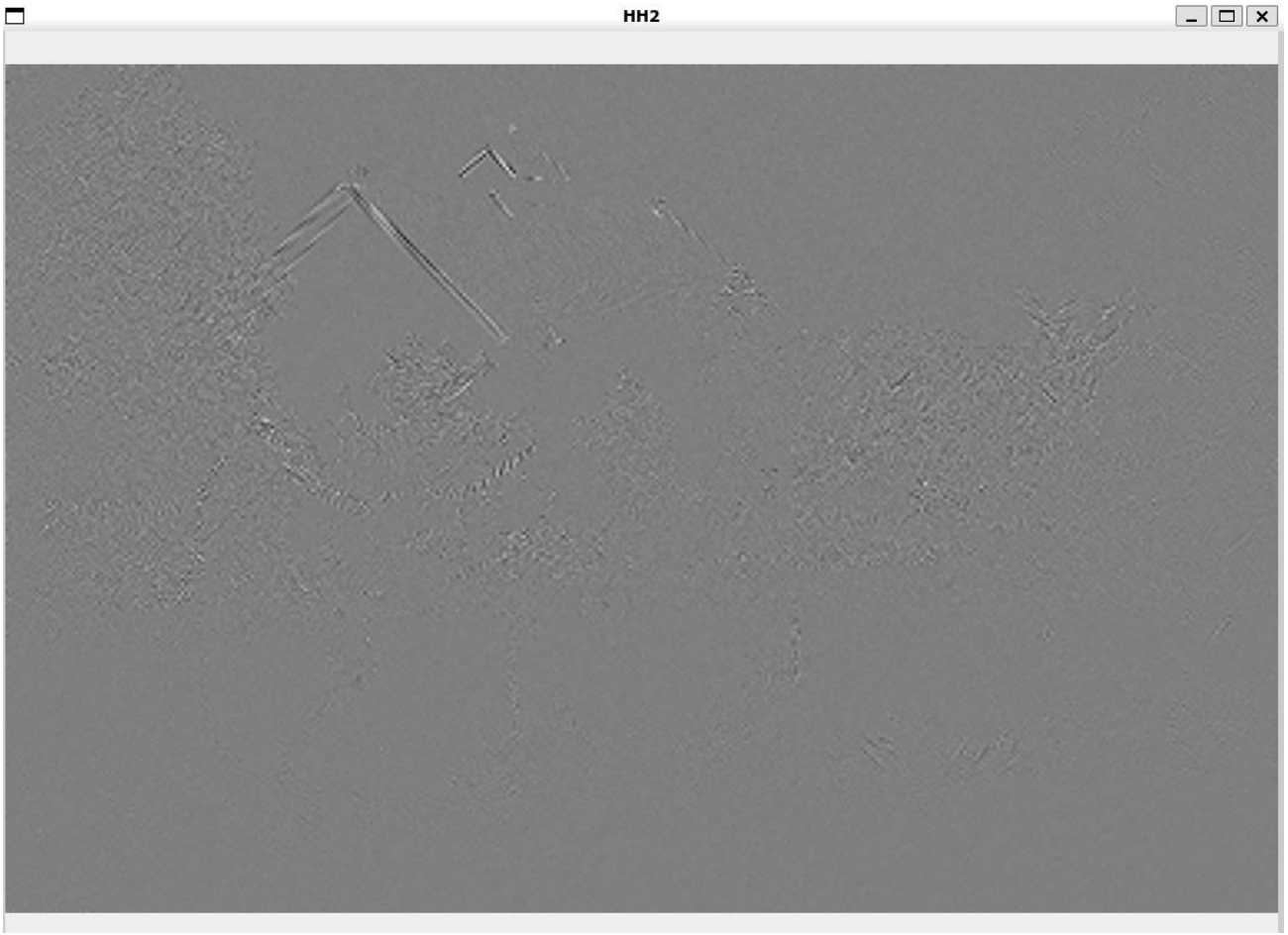


Szymon Dyszewski
Numer albumu: 310625

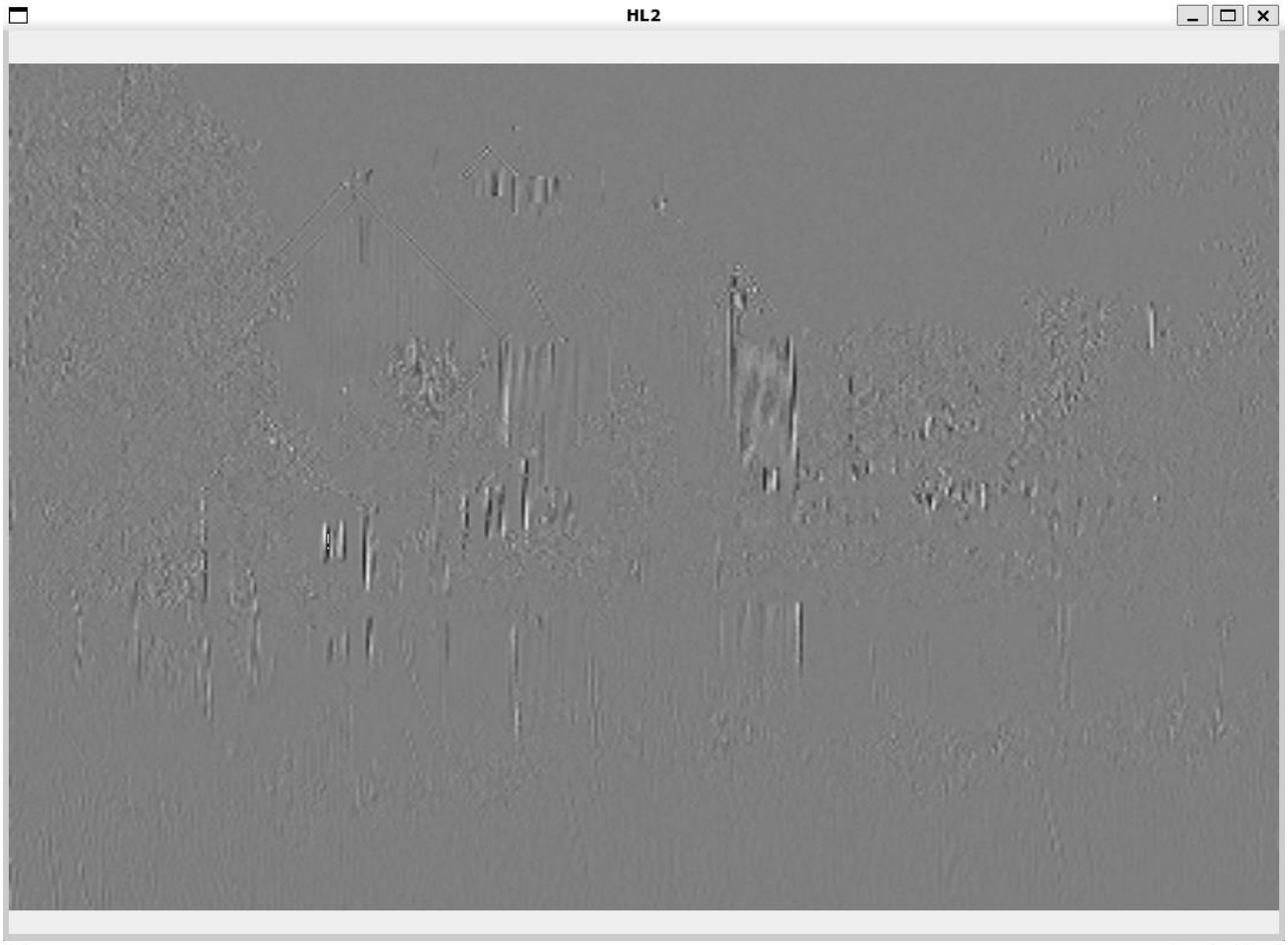
Histogram pokazuje, że zdecydowana większość pikseli obrazu różnicowego znajduje się w okolicy zera, a więc sąsiadujące ze sobą piksele nie różnią się od siebie znacznie. Natomiast obraz pierwotny wykorzystuje całe dostępne spektrum, a jego histogram wykorzystuje wszystkie wartości od 0 do 255.

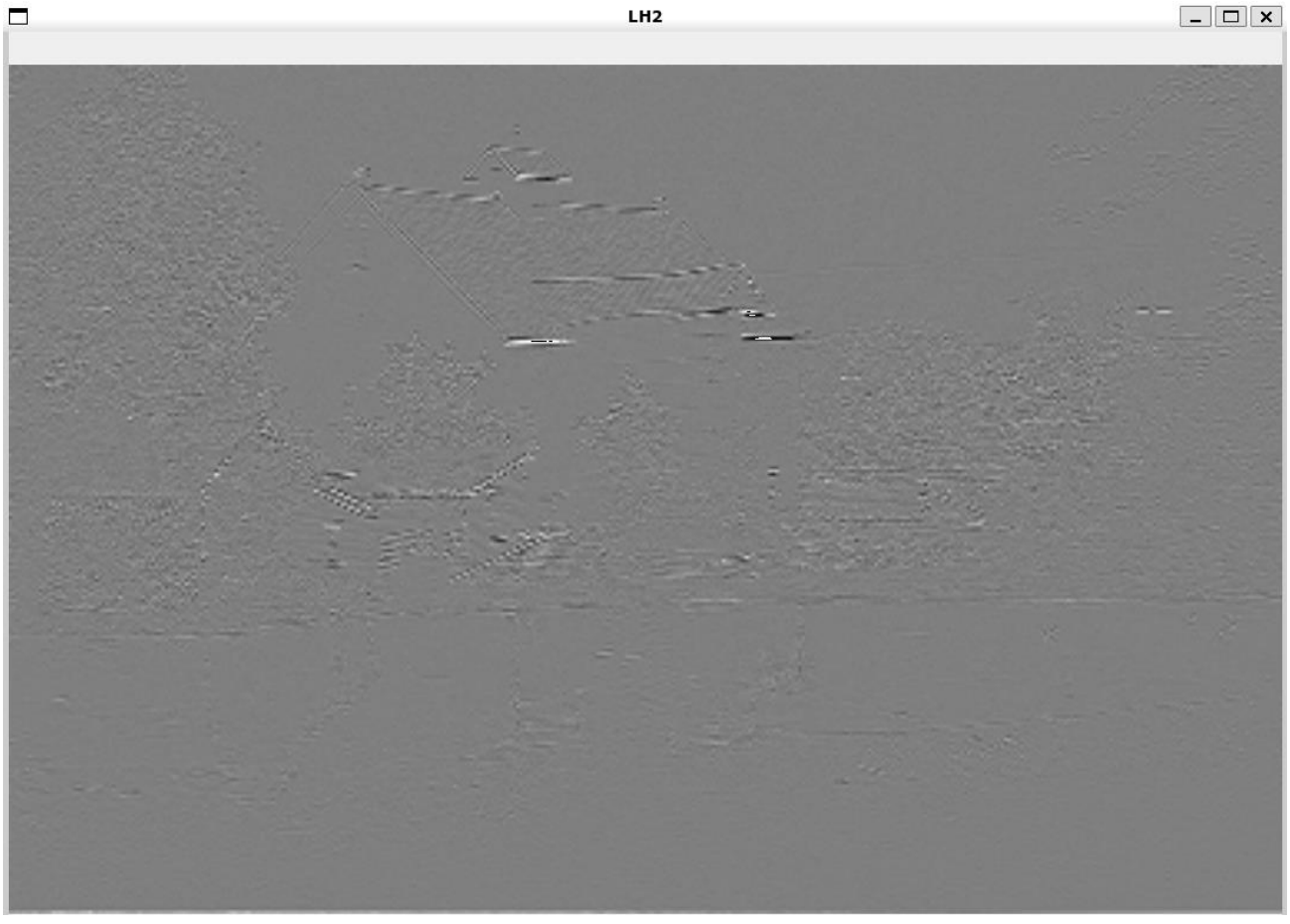
Transformacja falkowa obrazu

```python
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])

    ll = cv2.sepFilter2D(img,         -1, maskL, maskL)[::2, ::2]
    lh = cv2.sepFilter2D(img, cv2.CV_16S, maskL, maskH)[::2, ::2] ### ze względu na filtrację górnoprzepustową -> wartości ujemne, dla
    hl = cv2.sepFilter2D(img, cv2.CV_16S, maskH, maskL)[::2, ::2]
    hh = cv2.sepFilter2D(img, cv2.CV_16S, maskH, maskH)[::2, ::2]

    cv_imshow(ll, 'LL2')
    cv_imshow(cv2.multiply(hh, 2), 'HH2')
    cv_imshow(cv2.multiply(lh, 2), 'LH2')
    cv_imshow(cv2.multiply(hl, 2), 'HL2')

    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 dany
    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 = enthropycalculate(hist_ll)
    H_lh = enthropycalculate(hist_lh)
    H_hl = enthropycalculate(hist_hl)
    H_hh = enthropycalculate(hist_hh)
    print(f"H(LL) = {H_ll:.4f} \nH(LH) = {H_lh:.4f} \nH(HL) = {H_hl:.4f} \nH(HH) = {H_hh:.4f} \nH_śr = {(H_ll+H_lh+H_hl+H_hh)/4:.4f}")
```

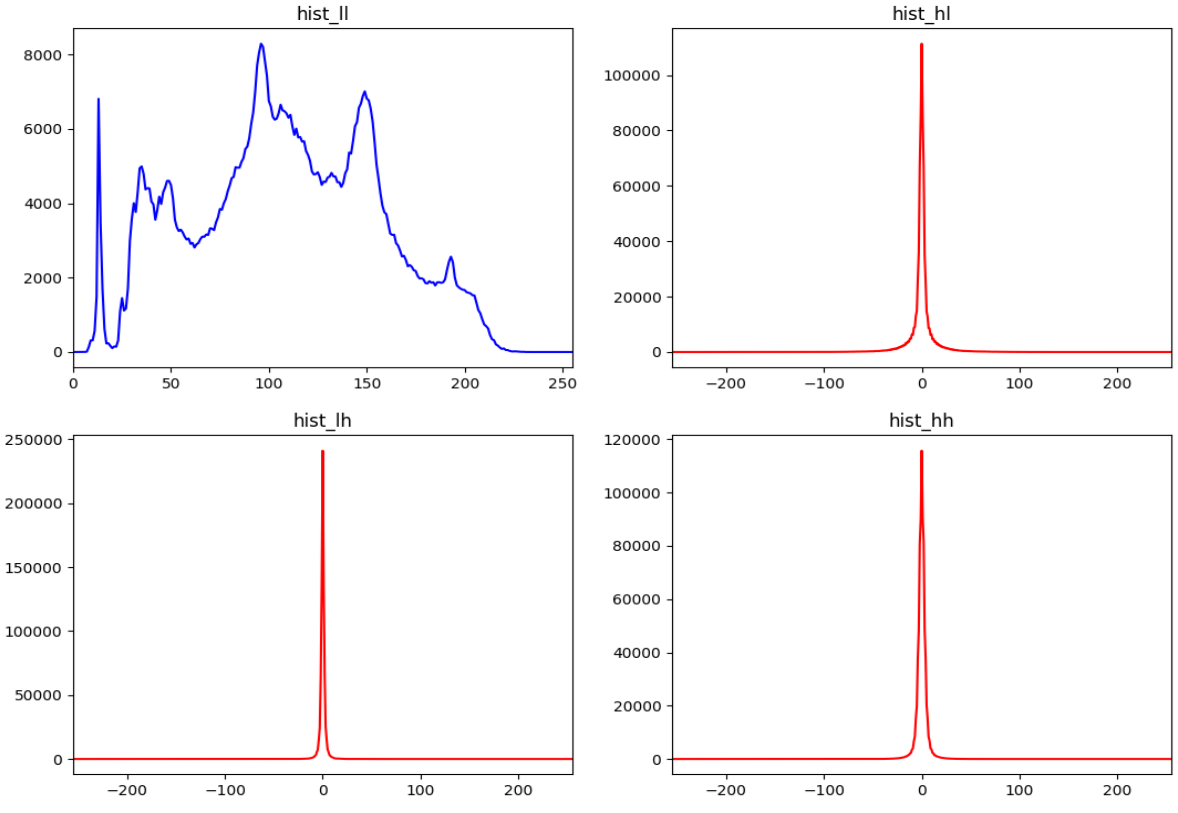


Szymon Dyszewski
Numer albumu: 310625





Szymon Dyszewski
Numer albumu: 310625



Szymon Dyszewski
Numer albumu: 310625

H(LL) = 7.2055

H(LH) = 4.6222

H(HL) = 4.7691

H(HH) = 4.7711

Porównać wyniki (histogram, entropia) uzyskane dla poszczególnych pasm między sobą (czy któreś się wyróżniają i dlaczego?) oraz z wynikami uzyskanymi dla obrazu oryginalnego i obrazu różnicowego.

Pasmo LL – dwukrotne zastosowanie filtru dolnoprzepustowego wyróżnia się na tle pozostałych pasm. Nie tylko wizualnie, ale histogram i entropia również odstają od pozostałych pasm. LL jest bardzo zbliżone do obrazu pierwotnego. Zastosowanie filtru górnoprzepustowego (LH, HL, HH) zbliżyło entropię i histogram do obrazu różnicowego. Wyjątkowość pasma LL bierze się z faktu, że filtr dolnoprzepustowy pozostawia detale obrazu.

Obraz barwny

```python
def entrophycol(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 = enthropycalculate(hist_R)
    H_G = enthropycalculate(hist_G)
    H_B = enthropycalculate(hist_B)
    print(f"H(R) = {H_R:.4f} \nH(G) = {H_G:.4f} \nH(B) = {H_B:.4f} \nH_śr = {(H_R+H_G+H_B)/3:.4f}")
```

H(R) = 7.1427

H(G) = 7.2771

H(B) = 7.1341

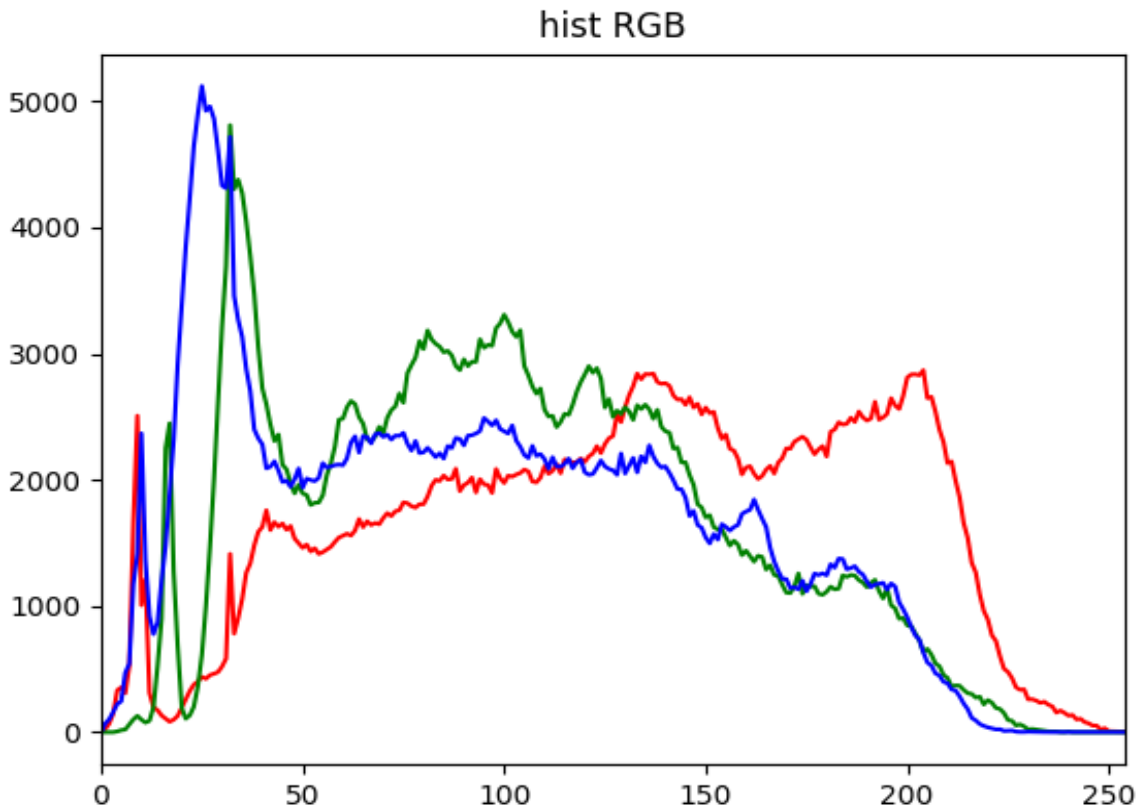
Szymon Dyszewski
Numer albumu: 310625

Image B:

Image G:

Szymon Dyszewski
Numer albumu: 310625

Image R:



Szymon Dyszewski
Numer albumu: 310625

Porównać wyniki uzyskane dla poszczególnych składowych.

Histogram składowych RGB przedstawia równomierny rozkład informacji, a ich entropie są zbliżone. Jedynie kolor czerwony posiada delikatne przesunięcie na prawą część histogramu.

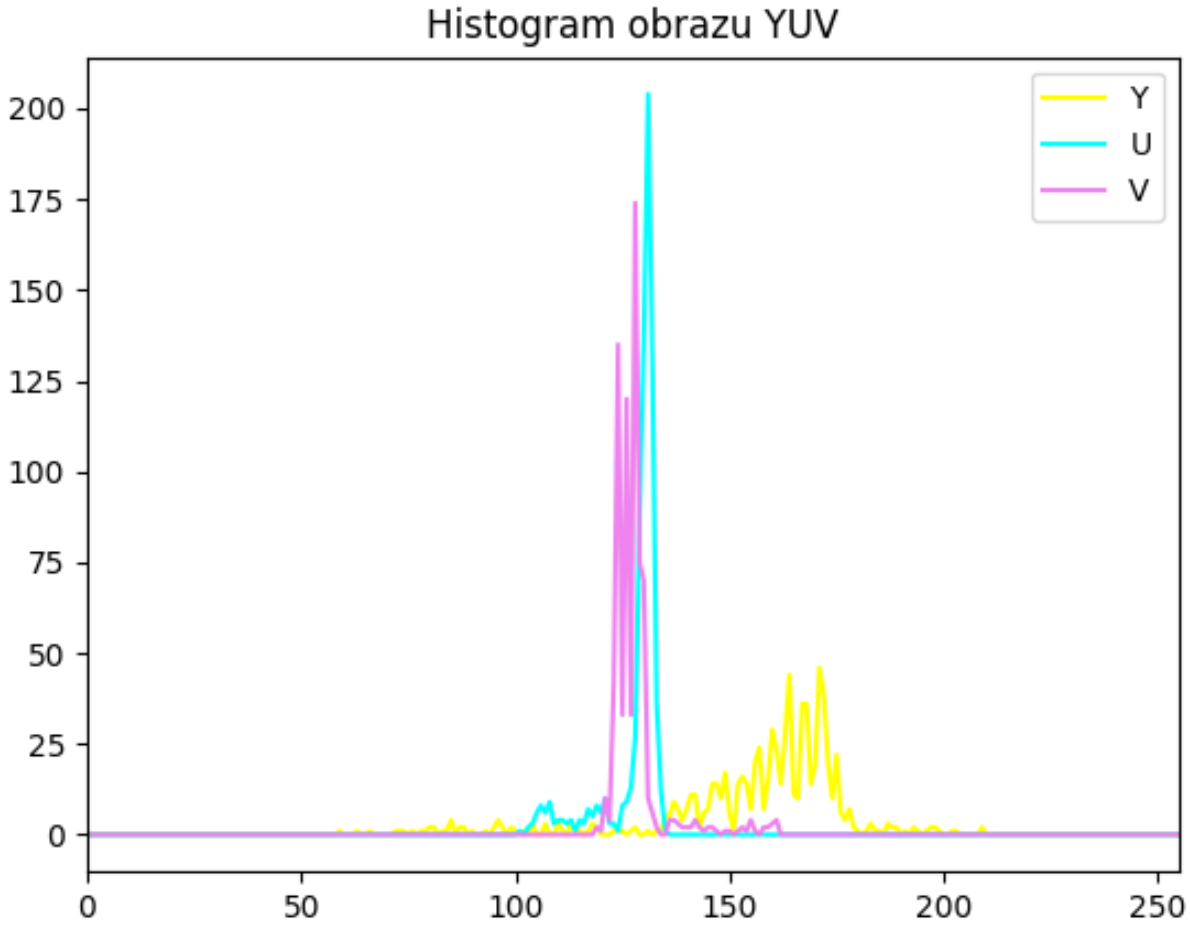
```
def BGR2YUV(img):
    image_yuv = cv2.cvtColor(img, cv2.COLOR_BGR2YUV)
    hist_Y = calcHist(image_yuv[...,0], [0], None, [256], [0, 256]).flatten()
    hist_U = calcHist(image_yuv[...,1], [0], None, [256], [0, 256]).flatten()
    hist_V = calcHist(image_yuv[...,2], [0], None, [256], [0, 256]).flatten()

    H_Y = enthropycalculate(hist_Y)
    H_U = enthropycalculate(hist_U)
    H_V = enthropycalculate(hist_V)
    print(f"H_Y = {H_Y:.4f} \nH_U = {H_U:.4f} \nH_V = {H_V:.4f} \nH_śr = {(H_Y+H_U+H_V)/3:.4f}")
```

H_Y = 5.7371

H_U = 3.4401

H_V = 3.4699

H_śr = 4.2157



Szymon Dyszewski
Numer albumu: 310625

Czy dla składowych UV entropia jest mniejsza? Z czego ta mniejsza wartość może wynikać?

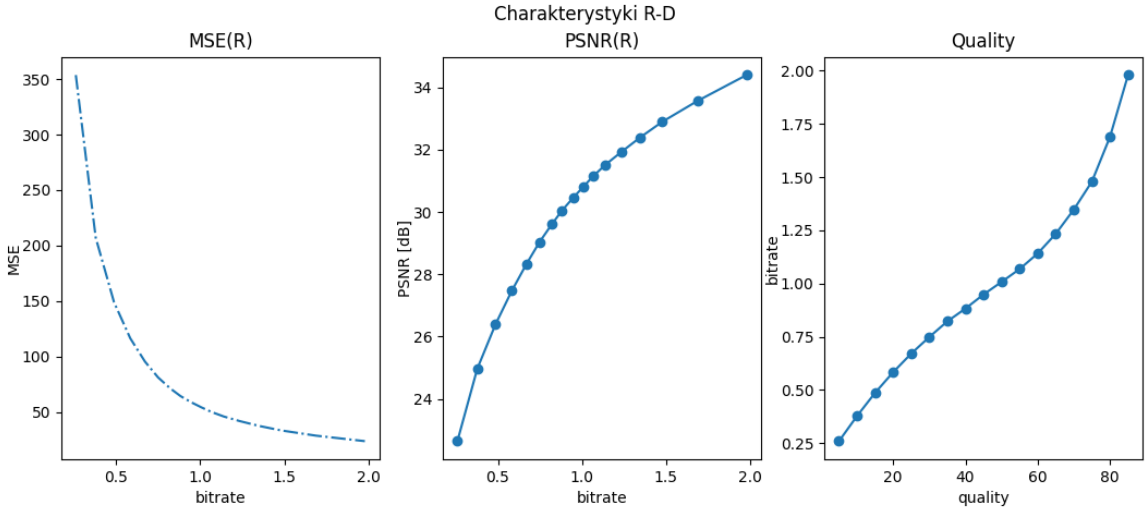
Tak. Składowa Y zajmuje znacznie większe spektrum wartości, a U i V skumulowane są pośrodku skali przez co ich entropia jest mniejsza.

Kompresja JPEG

```python
def calc_mse_psnr(img1, img2):
    """ Funkcja obliczająca MSE i PSNR dla różnicy podanych ob

    imax = 255.**2 ### maksymalna wartość sygnału -> 255
    """
    W różnicy obrazów istotne są wartości ujemne, dlatego img1
    aby nie ograniczać wyniku do przedziału [0, 255].
    """
    mse = ((img1.astype(np.float64)-img2)**2).sum()/img1.size
    psnr = 10.0*np.log10(imax/mse)
    return (mse, psnr)
```

```python
image = cv2.imread(color_path, cv2.IMREAD_UNCHANGED)
xx = [] ### tablica na wartości osi X -> bitrate
ym = [] ### tablica na wartości osi Y dla MSE
yp = [] ### tablica na wartości osi Y dla PSNR
Quality = [x for x in range(5, 86, 5)]

for quality in Quality: ### wartości dla parametru 'quality' należałoby dobrać
    out_file_name = f"./images/out_image_q{quality:03d}.jpg"
    """ Zapis do pliku w formacie .jpg z ustaloną 'jakością' """
    cv2.imwrite(out_file_name, image, (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/(image.shape[0]*image.shape[1])
    mse, psnr = calc_mse_psnr(image, image_compressed)
    """ Zapamiętanie wyników do późniejszego wykorzystania """
    xx.append(bitrate)
    ym.append(mse)
    yp.append(psnr)
```

Szymon Dyszewski
Numer albumu: 310625



Jakość 5:

Szymon Dyszewski
Numer albumu: 310625

Jakość 15:

Jakość 25:

Szymon Dyszewski
Numer albumu: 310625

Jakość 30:

Jakość 50:

Szymon Dyszewski
Numer albumu: 310625

Jakość 75:

Jakość 85:

Szymon Dyszewski
Numer albumu: 310625

Dokonać subiektywnej oceny obrazów zrekonstruowanych (według własnej skali ocen, np.: jakość doskonała, bardzo dobra, dobra, średnia, kiepska, zła, bardzo zła, itp., lub: zniekształcenia niewidoczne, lekko widoczne, widoczne, bardzo widoczne, nie do przyjęcia, itp.) i zamieścić te oceny w sprawozdaniu (niekoniecznie dla każdego obrazu wynikowego osobno, raczej 'zgrupować' oceny dla pewnych zakresów przepływności).

Oceny obrazów dokonano w następującej skali:

- 1: niedopuszczalna jakość
- 2: dopuszczalna jakość
- 3: przeciętna jakość
- 4: dobra jakość
- 5: bardzo dobra jakość
- 6: jakość doskonała

Obraz wejściowy jest w niskiej rozdzielczości, wpływa to negatywnie na odbiór zdjęcia.

zakres ‘quality’	**ocena**
od 0.0 do 0.2	1
od 0.2 do 0.3	2
od 0.3 do 0.5	3
od 0.5 do 0.6	4
od 0.6 do 0.85	5
od 0.85 do 1	6

Porównać stopnie kompresji uzyskiwane dla kodera JPEG ze stopniem kompresji uzyskanym dla kodera PNG (pamiętając, że w pierwszej części laboratorium wykorzystywany był monochromatyczny obraz PNG, a kompresja JPEG była wykonywana dla obrazu barwnego; ewentualnie wyliczyć przepływność bitową dla obrazu barwnego skompresowanego koderem PNG).

```
png_bitrate = 8 * os.stat(color_path).st_size / color_img.size
print(f"Bitrate of color PNG: {png_bitrate:.4f}")
```

Bitrate of color PNG: 4.7671

Kompresja PNG w przeciwieństwie do JPEG jest kompresją bezstratną. Dzięki temu przepływność kompresji PNG jest zdecydowanie lepsza, kosztem od 2. do 13. razy większego rozmiaru pliku niż dla kompresji JPEG.