ŁUKASZ STANISZEWSKI

WPROWADZENIE DO MULTIMEDIÓW

SPRAWOZDANIE Z LABORATORIUM 5 – STATYSTYCZNE WŁAŚCIWOŚCI OBRAZÓW



# **Przygotowanie**

Koniecznym było pozyskanie odpowiednich plików do wykonania zadania. W wyniku działania , pobrano pliki: **boat2\_mono.png** oraz **boat2\_col.png**.

# **Przepływność i entropia obrazu mono**

## **KOD ŹRÓDŁOWY**

*kod 2.1 – pełne rozwiązanie punktu II*

**def calc\_bitrate\_and\_entropy(image, path):**

**bitrate = 8 \* os.stat(path).st\_size / (image.shape[0] \* image.shape[1])**

**print(f"bitrate: {bitrate:.4f} bpp") # obliczanie przeplywnosci / bitrate**

**hist\_image = cv2.calcHist([image], [0], None, [256], [0, 256])**

**hist\_image = hist\_image.flatten() # otrzymanie histogramu**

**H\_image = calc\_entropy(hist\_image) # obliczanie entropii**

**print(f"H(image) = {H\_image:.4f}")**

**def calc\_entropy(hist):**

**pdf = hist / hist.sum()  # normalizacja histogramu -> rozkład prawdopodobieństwa**

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

**return entropy**

**def task12():**

**imageBW = cv2.imread("boat2\_mono.png", cv2.IMREAD\_UNCHANGED)**

**calc\_bitrate\_and\_entropy(imageBW, "boat2\_mono.png")**

## **ROZWIĄZANIE**

W wyniku wykonania przedstawionych wyżej funkcji, otrzymano **przepływność/bitrate** (liczbę bitów przypadającą na 1 piksel) dla obrazu skompresowanego **PNG**, a także obliczono **entropię obrazu** (H) i porównano z wyliczoną **przepływnością**. Wynikiem jest:

*WYNIK 2.1 – OBLICZONY BITRATE I ENTROPIA DLA OBRAZU MONO PNG*

**bitrate: 5.0293 bpp**

**H(image) = 6.8484**

Można tu zauważyć, że **entropia obrazu jest większa od jego bitrate’u/przepływności**. Fakt ten jednak **nie oznacza**, że zależność mówiąca, że **średnia długość kodu przedrostkowego nie może być mniejsza niż entropia jest fałszywa**. Jest to spowodowane faktem, że **zależność ta dotyczy kodów przedrostkowych**, a zastosowana w zadaniu **kompresja PNG** jest **kompresją** **bezstratną** i opiera się na **idei kodowania słownikowego** (stąd zależność może być tu łamana). Poprzez zastosowanie kompresji PNG bit-rate obrazu oryginalnego zmalał przy jednoczesnym zachowaniu się prawdopodobieństw wartości pikseli (kompresja bezstratna – zachowanie entropii), przez co zależność przestała być spełniona, ale nie dotyczy tu już kodu przedrostkowego.

# **Obraz różnicowy**

## **KOD ŹRÓDŁOWY**

**def get\_differential\_image(image):**

**img\_tmp1 = image[:, 1:]   # kolumny od 1 do ostatniej**

**img\_tmp2 = image[:, :-1]  # kolumny od 0 do przedostatniej**

**image\_hdiff = cv2.addWeighted(img\_tmp1, 1, img\_tmp2, -1, 0, dtype=cv2.CV\_16S) # predykcja w poziomie**

**image\_hdiff\_0 = cv2.addWeighted(image[:, 0], 1, 0, 0, -127, dtype=cv2.CV\_16S)  # od****0 kolumny odejmowane 127**

**image\_hdiff = np.hstack((image\_hdiff\_0, image\_hdiff))  # połączenie tablic w kierunku poziomym**

**cv\_imshow(image\_hdiff, "image\_hdiff")****# funkcja pomocnicza do wyświetlania obrazów**

**return image\_hdiff**

**def task3():**

**imageBW = cv2.imread("boat2\_mono.png", cv2.IMREAD\_UNCHANGED)**

**image\_differential = get\_differential\_image(imageBW) # otrzymanie obrazu różnicowego i jego wyświetlenie**

**compare\_hist\_entropy(imageBW, image\_differential) # funkcja do wyswietlania histogramow i entropii**

*KOD 3.1 – OBLICZENIE OBRAZU RÓŻNICOWEGO I JEGO WYŚWIETLENIE*

## **ROZWIĄZANIE**

*kod 3.2 – funkcja wyświetlająca histogramy i entropie obrazu oryginalnego i różnicowego*

**def compare\_hist\_entropy(image1, image2):**

**image\_tmp = (image2 + 255).astype(np.uint16) # calcHist() wymaga liczb całkowitych bez znaku, odp. skalowanie**

**hist\_hdiff = cv2.calcHist([image\_tmp], [0], None, [511], [0, 511]).flatten() # histogram obrazu różnicowego**

**hist\_image = cv2.calcHist([image1], [0], None, [256], [0, 256]).flatten() # histogram obrazu oryginalnego**

**print(f"Entropy original image: {calc\_entropy(hist\_image):.4f}")** **# obliczanie entropii**

**print(f"Entropy differential image: {calc\_entropy(hist\_hdiff):.4f}")**

**plt.figure() # tworzenie zestawu histogramów**

**plt.subplot(121)**

**plt.plot(hist\_image, color="blue")**

**plt.title("Histogram original image")**

**plt.xlim([0, 255])**

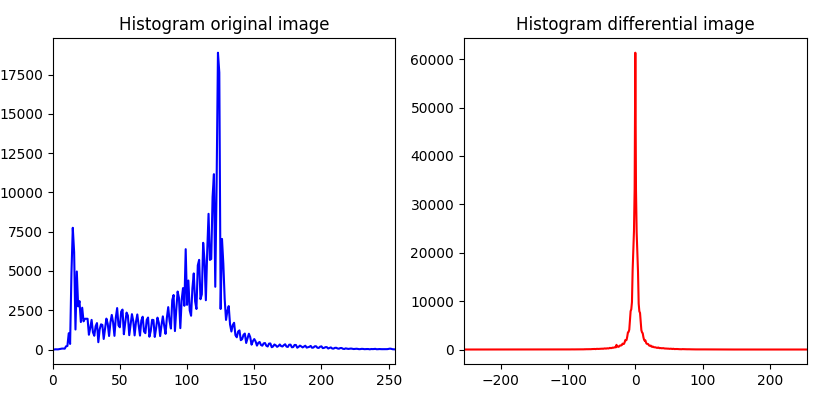
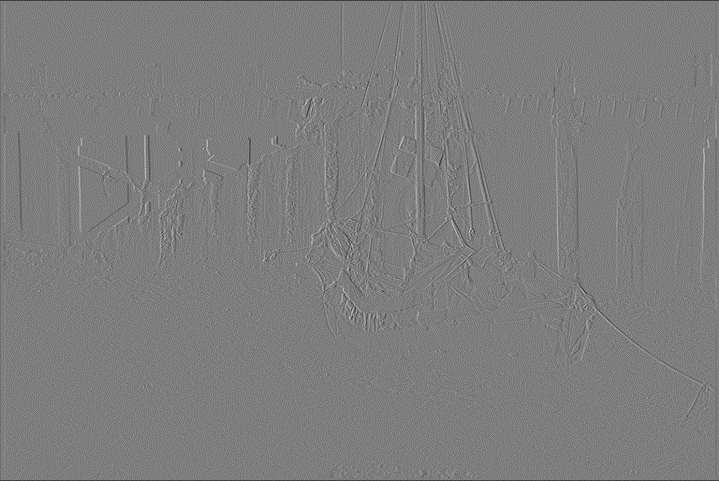
**plt.subplot(122)**

**plt.plot(np.arange(-255, 256, 1), hist\_hdiff, color="red") # dziedzina w różnicowym to <-255, 255>**

**plt.title("Histogram differential image")**

**plt.xlim([-255, 255])**

**plt.show()**

Na początku koniecznym było wyznaczenie **obrazu różnicowego** (z kodowaniem różnicowym / predykcyjnym **poziomym**), gdzie dla pierwszego piksela w wierszu konieczne było przyjęcie wartości sąsiada jako **127**. Następnie **wyświetlono obraz różnicowy**. Na końcu **wyznaczono histogram dla obrazu różnicowego i oryginalnego**, oraz **entropię** obu **obrazów**.

*RYS. 3.2 - PORÓWNANIE HISTOGRAMU OBRAZU ORYGINALNEGO Z HISTOGRAMEM OBRAZU RÓŻNICOWEGO*

*RYS. 3.1 - OBRAZ RÓŻNICOWY*

WYNIK 3.1 – ENTROPIA OBRAZU ORYGINALNEGO I OBRAZU RÓZNICOWEGO

**Entropy original image: 6.8484**

**Entropy differential image: 5.0033**

* **Wygląd obrazu różnicowego bardzo oddaje jego histogram**. Na obrazie oryginalnym mono dominuje bardzo **dużo pikseli mających sąsiednie piksele o tym samym kolorze** (całe niebo, większość wody, w miejscach tych dominuje wartość piksela o **wartości 127,5** – **peak na histogramie obrazu oryginalnego**), dlatego też **różnice pomiędzy tymi pikselami będą bardzo bliskie 0**, co oddaje **peak na histogramie obrazu różnicowego**. Z **histogramu obrazu oryginalnego** można wyczytać, że występuje na obrazie oryginalnym **całkiem dużo pikseli bliskich koloru czarnego** (bliskich wartości 0), najczęściej na obrazie oryginalnym mono są to piksele pod mostem, natomiast **na obrazie różnicowym** zauważyć można **krawędzie na obrazie** – te **piksele** **posiadały** w obrazie oryginalnym **lewych sąsiadów o innej wartości**, świadczą o tym **na histogramie obrazu różnicowego** **wartości dla argumentów wokół 0**, jednak są one dużo mniej znaczące pod względem ilości w porównaniu z liczebnością różnicy pikseli wynoszącej 0.
* **Na obrazie różnicowym dominuje kolor szary** i **jego odcienie** (dlatego, że **wszystkie wartości różnicy są w bardzo bliskiej odległości od środka zakresu dopuszczalnych wartości pikseli** i po odpowiednim przeskalowaniu **środek ten odpowiada kolorowi szaremu**).
* W związku z tak **dużą dominacją koloru szarego na obrazie różnicowym**, jego **średnia informacji będzie znacznie mniejsza niż dla obrazu oryginalnego**, co oczywiście odbije się też na jego **entropię**, która będzie **znacznie mniejsza niż entropia obrazu oryginalnego mono (prawdopodobieństwa** pojedynczych **wartości pikseli** znaczącosię **zwiększają)**.

# **Współczynniki DWT**

## **KOD ŹRÓDŁOWY**

**def dwt(img):       # funkcja wyznaczająca pasma dwt (uproszczona) i je zwracająca**

**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]**

**# w filtracji górnoprzepustowej wartości ujemne, więc wynik 16-bitowy ze znakiem**

**bandLH = cv2.sepFilter2D(img, cv2.CV\_16S, maskL, maskH)[::2, ::2]**

**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 show\_hist\_bands(hist\_ll, hist\_lh, hist\_hl, hist\_hh):**

**# funkcja przedstawia podane histogramy jako subplot**

**fig = plt.figure()**

**# zwiększenie rozmiarów okna**

**fig.set\_figheight(fig.get\_figheight() \* 2)**

**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()**

**cv2.destroyAllWindows()**

**def task4():**

**imageBW = cv2.imread("boat2\_mono.png", cv2.IMREAD\_UNCHANGED)**

**# wyznaczenie pasm z użyciem transformacji DWT**

**ll, lh, hl, hh = dwt(imageBW)**

**# wyświetlenie pasm**

**cv\_imshow(ll, "LL2")**

**cv\_imshow(cv2.multiply(lh, 2), "LH2")**

**cv\_imshow(cv2.multiply(hl, 2), "HL2")**

**cv\_imshow(cv2.multiply(hh, 2), "HH2")**

**# obliczenie histogramów z odpowiednim skalowaniem i konwersją dla calcHist()**

**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()**

**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()**

**# liczenie entropi**

**H\_ll = calc\_entropy(hist\_ll)**

**H\_lh = calc\_entropy(hist\_lh)**

**H\_hl = calc\_entropy(hist\_hl)**

**H\_hh = calc\_entropy(hist\_hh)**

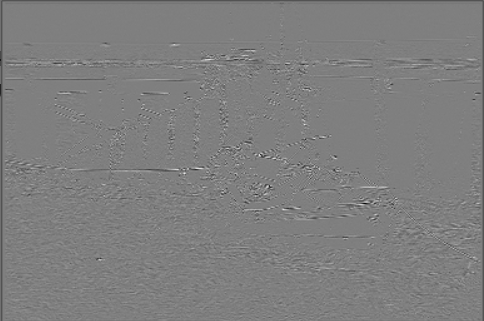
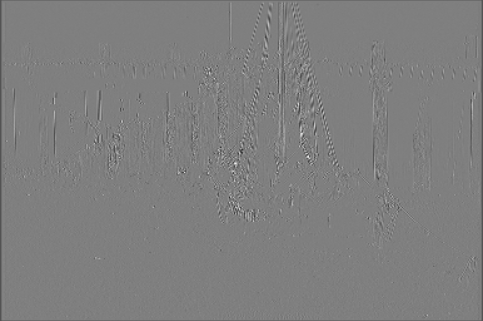
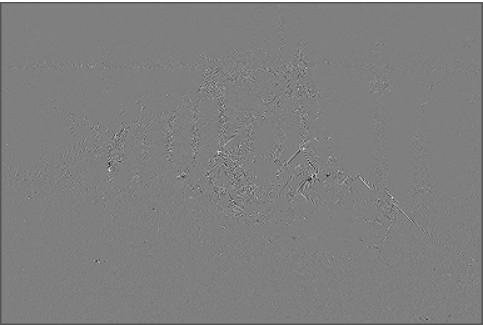
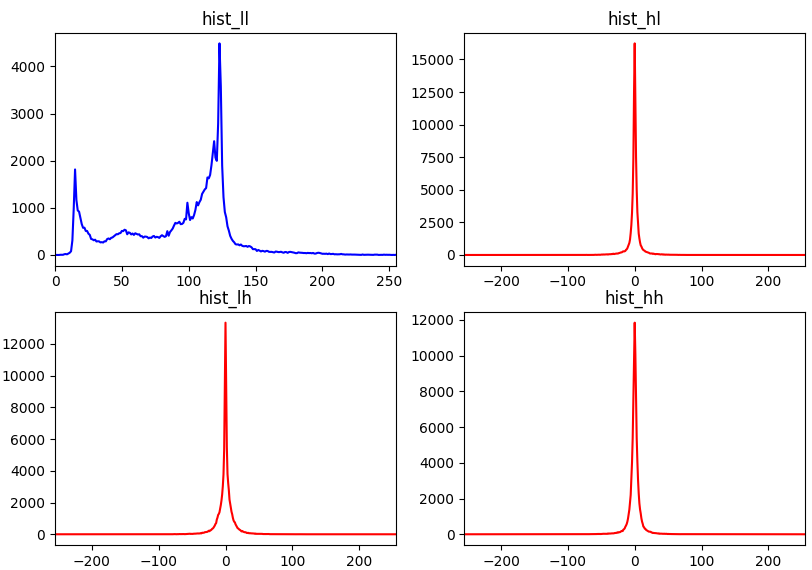
**print(f"H(LL) = {H\_ll:.4f} \nH(LH) = {H\_lh:.4f} \nH(HL) = {H\_hl:.4f} \nH(HH) = {H\_hh:.4f} \n")**

**# zestawienie histogramów**

**show\_hist\_bands(hist\_ll, hist\_lh, hist\_hl, hist\_hh)**

*KOD 4.1 – LICZENIE DWT, WYŚWIETLENIE PASM, ICH ENTROPII I HISTOGRAMÓW*

## **ROZWIĄZANIE**

Na początku koniecznym było **wyznaczenie współczynników DWT** (z wykorzystaniem funkcji ze skryptu) oraz **wyświetlenie poszczególnych pasm**. Na końcu **wyznaczono histogramy** i **entropie** dla wszystkich **pasm**.

**H(LL) = 6.8753**

**H(LH) = 5.0958**

**H(HL) = 4.4986**

**H(HH) = 4.7002**

*WYNIK 4.1 – ZESTAWIENIE ENTROPII DLA POSZCZEGÓLNYCH PASM*

*RYS 4.5 – ZESTAWIENIE HISTOGRAMÓW*

*RYS 4.4 – PASMO HH*

*RYS 4.3 – PASMO HL*

*RYS 4.2 – PASMO LH*

*RYS 4.1 – PASMO LL*

* **Pasmo LL** pod względem zarówno **wyglądu**, **histogramu** jak i **entropii** najbardziej **wyróżnia się spośród wszystkich otrzymanych pasm** . Widać na podstawie tego, że **dolne pasmo zawiera najwięcej energii** i **przypomina oryginał** (ponieważ **na niskich częstotliwościach są szczegóły**). Nastąpiło tutaj **zmniejszenie rozdzielczości**, ale **wydzielono obraz podobny do oryginału**. O **podobieństwie świadczy także histogram dla pasma LL** (który jest całkiem **podobny do histogramu obrazu oryginalnego mono)**, a także **bardzo podobne do siebie entropie pasma LL i obrazu**.
* **Pasma LH, HL oraz HH są do siebie całkiem podobne**, zarówno pod względem **entropii**, **histogramu** jak i **wyglądu**. **Posiadają** one **znacznie mniejszą energię** niż w przypadku **pasma LL**. **Przypominają** one (pod względem samego **wyglądu**, **histogramu** i **entropii**) **obraz różnicowy** (przedstawione na rys.: 3.1 i 3.2). Tutaj również **dominuje kolor szary**, zgodne jest to z postaciami **histogramów** (posiadają **peak** w tym samym miejscu, **po środku** dopuszczalnych wartości). **Same histogramy dla tych pasm różnią się maksymalną wartością samego peak’u**.

# **Entropia RGB oraz YUV**

## **KOD ŹRÓDŁOWY**

*KOD 5.1 – OTRZYMANIE HISTOGRAMÓW I ENTROPII DLA SKŁADOWYCH RGB I YUV*

**def showHistograms(histograms, hist\_names):**

**# funkcja dla podanych histogramów wyświetla je obok siebie**

**subplots = [131, 132, 133]**

**plt.figure()**

**for hist, name, sub in zip(histograms, hist\_names, subplots):**

**plt.subplot(sub)**

**plt.plot(hist)**

**plt.title(name)**

**plt.show()**

**def task56():**

**image\_col = cv2.imread("boat2\_col.png")**

**# wyznacznie obrazów dla poszczególnych składowych**

**image\_R = image\_col[:, :, 2]**

**image\_G = image\_col[:, :, 1]**

**image\_B = image\_col[:, :, 0]**

**# histogramy dla każdej składowej**

**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()**

**# entropie dla każdej składowej**

**H\_R = calc\_entropy(hist\_R)**

**H\_G = calc\_entropy(hist\_G)**

**H\_B = calc\_entropy(hist\_B)**

**print(f"H(R) = {H\_R:.4f} \nH(G) = {H\_G:.4f} \nH(B) = {H\_B:.4f}\n")**

**# konwersja do YUV**

**image\_YUV = cv2.cvtColor(image\_col, cv2.COLOR\_BGR2YUV)**

**# histogramy dla składowych**

**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()**

**#entropie dla składowych**

**H\_Y = calc\_entropy(hist\_Y)**

**H\_U = calc\_entropy(hist\_U)**

**H\_V = calc\_entropy(hist\_V)**

**print(f"H(Y) = {H\_Y:.4f} \nH(U) = {H\_U:.4f} \nH(V) = {H\_V:.4f}\n")**

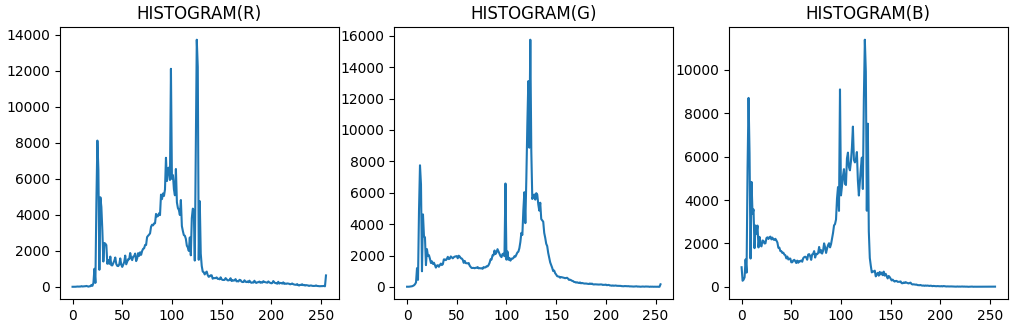
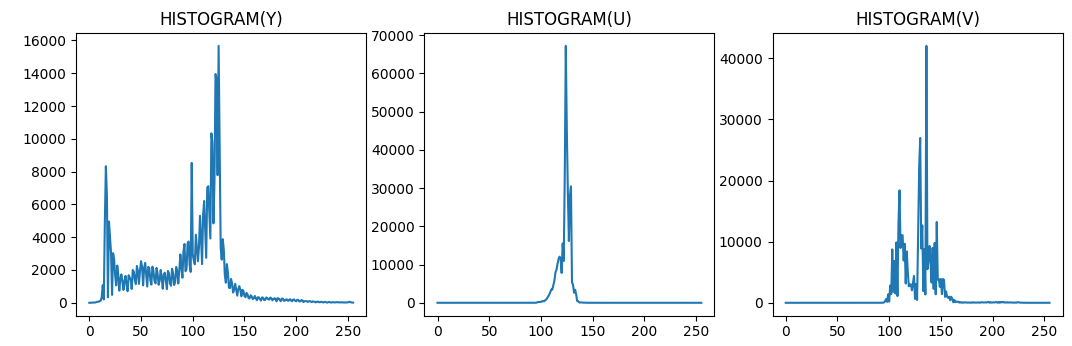
**# wyświetlenie histogramów wszystkich składowych**

**showHistograms([hist\_R, hist\_G, hist\_B], ["HISTOGRAM(R)", "HISTOGRAM(G)", "HISTOGRAM(B)"])**

**showHistograms([hist\_Y, hist\_U, hist\_V], ["HISTOGRAM(Y)", "HISTOGRAM(U)", "HISTOGRAM(V)"])**

## **ROZWIĄZANIE**

Na początku koniecznym było **wyznaczenie poszczególnych składowych RGB**, wykonania na ich podstawie **histogramów** i policzenie **entropii**. W następnym kroku wykonano **przekształcenie obrazu barwnego do przestrzeni barw YUV**, wyznaczenie **składowych**, **histogramów** na ich podstawie, policzenie **entropii składowych YUV**, a na końcu **wyświetlono** **uzyskane entropie i histogramy**.



*RYS 5.2 – HISTOGRAMY DLA SKŁADOWYCH YUV*

*WYNIK 5.1 – ENTROPIE DLA POSZCZEGÓLNYCH SKŁADOWYCH RGB I YUV*

**H(R) = 6.9182  
H(G) = 7.0234  
H(B) = 6.9199  
  
H(Y) = 6.8474  
H(U) = 4.2524  
H(V) = 5.4024**

*RYS 5.1 – HISTOGRAMY DLA SKŁADOWYCH RGB*

* Jeśli chodzi o **histogramy dla składowych RGB**, **żaden z nich nie wyróżnia się na tle pozostałych**. Świadczy o tym również **entropia dla poszczególnych składowych**, składowe są do siebie bardzo zbliżone pod jej względem, tzn. że **każda ze składowych posiada bardzo podobną średnią informację niesioną przez siebie** i **w każdej ze składowych skupiona jest podobna energia**.
* **Różnice można zauważyć natomiast w przypadku składowych YUV**, gdzie **histogram dla składowej Y jest znacznie bardziej rozszerzony niż histogramy dla składowych U i V** (gdzie występuje skupienie w okolicy środkowej dopuszczalnej wartości pikseli). Podobnie jest w przypadku **entropii** – **widać znaczącą różnicę między entropia dla poszczególnych składowych**. **Najmniej energii skupionej jest w składowej V**, **więcej w U**, **a najwięcej w składowej Y (składowej luminancji)**. **Właściwość** ta jest **wykorzystywana w kompresji**. Z powodu, że mamy **zróżnicowaną energię w poszczególnych składowych**, **składowe,** w których skupione jest **mniej energii**, mogą być **reprezentowane i obserwowane na mniejszej liczbie bitów**.

# **Koder PNG a koder JPEG**

## **KOD ŹRÓDŁOWY**

*KOD 6.1 – WYZNACZONA ZALEŻNOŚĆ ZNIEKSZTAŁCENIA D OD PRZEPŁYWNOŚCI R DLA KODERA JPEG*

**def calc\_mse\_psnr(img1, img2):**

**# Funkcja liczy MSE i PSNR dla podanych obrazów, zał: piksele z przedziału [0, 255]**

**imax = 255. \*\* 2  # maksymalna wartość sygnału -> 255**

**# Istotne wartości ujemne, dlatego img1 konwertowany do np.float64 (liczby rzeczywiste)**

**mse = ((img1.astype(np.float64) - img2) \*\* 2).sum() / img1.size**

**psnr = 10.0 \* np.log10(imax / mse)**

**return mse, psnr**

**def task7():**

**image\_col = cv2.imread("boat2\_col.png", cv2.IMREAD\_UNCHANGED)**

**xx = []  # tablica na bitrate**

**ym = []  # tablica na MSE**

**yp = []  # tablica na PSNR**

**for quality in [100, 90, 80, 70, 60, 50, 40, 30, 20, 10, 0]:**

**out\_file\_name = f"out\_image\_q{quality:03d}.jpg"**

**# Zapis do pliku w formacie .jpg z ustaloną jakością**

**cv2.imwrite(out\_file\_name, image\_col, (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\_col.shape[0] \* image\_col.shape[1])**

**mse, psnr = calc\_mse\_psnr(image\_col, image\_compressed)**

**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()**

**# liczenie stopnia kompresji**

**size\_png = os.stat("boat2\_col.png").st\_size**

**size\_jpg\_100 = os.stat("out\_image\_q100.jpg").st\_size**

**size\_jpg\_10 = os.stat("out\_image\_q010.jpg").st\_size**

**print("~~STOPNIE KOMPRESJI:")**

**print(f"Stopień kompresji przy quality=100: {size\_png / size\_jpg\_100:.4f}")**

**print(f"Stopień kompresji przy quality=10: {size\_png / size\_jpg\_10:.4f}")**

**# liczenie przepływności**

**print("~~PRZEPŁYWNOŚCI:")**

**image\_col = cv2.imread("boat2\_col.png", cv2.IMREAD\_UNCHANGED)**

**bitrate\_col = 8 \* os.stat("boat2\_col.png").st\_size / (image\_col.shape[0] \* image\_col.shape[1])**

**image\_bw = cv2.imread("boat2\_mono.png", cv2.IMREAD\_UNCHANGED)**

**bitrate\_bw = 8 \* os.stat("boat2\_mono.png").st\_size / (image\_bw.shape[0] \* image\_bw.shape[1])**

**image\_jpg = cv2.imread("out\_image\_q100.jpg", cv2.IMREAD\_UNCHANGED)**

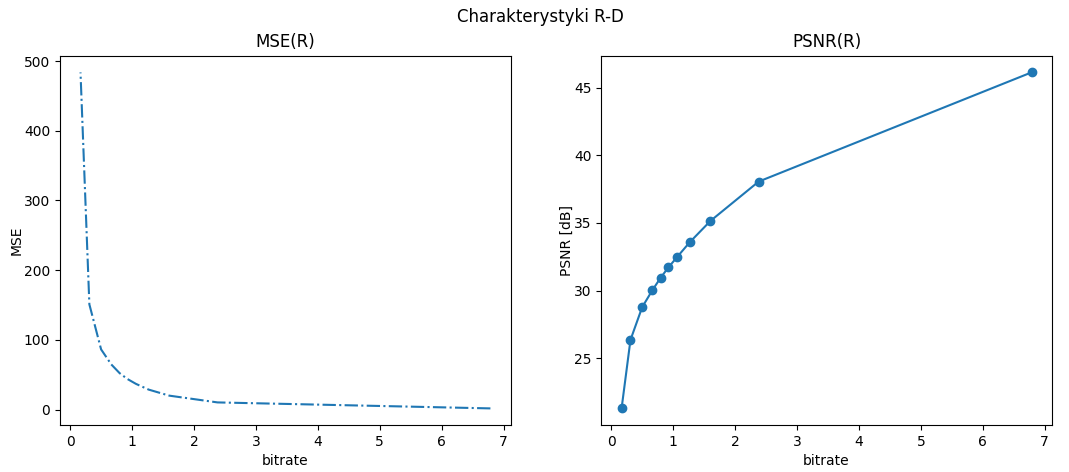
**bitrate\_jpg = 8 \* os.stat("out\_image\_q100.jpg").st\_size / (image\_jpg.shape[0] \* image\_jpg.shape[1])**

**print(f"Przepływność dla RGB PNG: {bitrate\_col} bpp")**

**print(f"Przepływność dla MONO PNG: {bitrate\_bw} bpp")**

**print(f"Przepływność dla RGB JPG: {bitrate\_jpg} bpp")**

## **ROZWIĄZANIE**

W tym zadaniu konieczne jest wyznaczenie **zależności zniekształcenia D od przepływności R** w postaci **krzywej R-D dla kodera JPEG** z miarami zniekształceń **PSNR** oraz **MSE**. W tym celu zostały dobrane tak parametry **quality**, żeby **uzyskane wykresy krzywej były gładkie**. Następnie **zobrazowano wykresy** tej zależności (dla każdej miary osobno), a także **oceniono subiektywnie obrazy zrekonstruowane**.

*RYS. 6.1 – CHARAKTERYSTYKI R-D DLA MIARY MSE ORAZ PSNR*

*RYS 6.3 – OBRAZ W WYNIKU KOMPRESJI JPEG DLA QUALITY=10*

*RYS 6.2 – OBRAZ W WYNIKU KOMPRESJI JPEG DLA QUALITY=100*

Można zauważyć, że **wraz ze zmniejszaniem się jakości kompresji** (i maleniem rozmiaru pliku kompresowanego), **maleje jakość obrazów wyjściowych** (dla jakości **100** obraz jest **doskonałej jakości**, jednak dla jakości **10** na obrazie widać **zniekształcenia** w postaci m.in. **„zpikselowanych” fragmentów np. nieba**). Jeśli chodzi o wykresy to widać, że **wraz ze zwiększaniem się przepływności w bitach na piksel** (zwiększaniem jakości kompresji), **logarytmicznie** **wzrasta** **szczytowy stosunek sygnału do szumu PSNR** (skala decybelowa), natomiast **błąd średniokwadratowy MSE** **maleje** **wymiernie** (ze względu na coraz większe rozmiary obrazu w bajtach). Dodatkowo została stworzona tabela zestawiająca subiektywne oceny obrazów zrekonstruowanych do zakresów ich przepływności.

*TABELA 6.1 – ZESTAWIENIE SUBIEKTYWNEJ OCENY JAKOŚCI DLA DANYCH PRZEPŁYWNOŚCI OBRAZÓW SKOMPRESOWANYCH*

|  |  |
| --- | --- |
| **BITRATE [BITS PER PIXEL]** | **SUBIEKTYWNIE JAKOŚĆ** |
|  | *JAKOŚĆ NIEAKCEPTOWALNA* |
|  | *JAKOŚĆ ZŁA* |
|  | *JAKOŚĆ ŚREDNIA* |
|  | *JAKOŚĆ DOBRA* |
|  | *JAKOŚĆ BARDZO DOBRA* |
|  | *JAKOŚĆ DOSKONAŁA* |

Na końcu zostały zestawione **stopnie kompresji JPEG** z **jakością 100** oraz z **jakością 10**, a także **porównane przepływności dla obrazów**: **oryginalnego kolorowego kompresowanego PNG**, **oryginalnego monochromatycznego PNG**, a także **kolorowego kompresowanego JPEG z jakością 100**. Sam **stopień** **kompresji** liczony był z użyciem wzoru: .

WYNIK 6.1 – WYLICZONY STOPIEŃ KOMPRESJI DLA JAKOŚCI 10 ORAZ 100 WRAZ Z OBLICZONYMI PRZEPŁYWNOŚCIAMI

**~~STOPNIE KOMPRESJI:**

**Stopień kompresji przy quality=100: 1.9093**

**Stopień kompresji przy quality=10: 41.8653**

**~~PRZEPŁYWNOŚCI:**

**Przepływność dla RGB PNG: 12.9577 bpp**

**Przepływność dla MONO PNG: 5.0293 bpp**

**Przepływność dla RGB JPG: 6.7866 bpp**

Jak widać, **kompresowanie przy użyciu kodera JPEG z jakością = 10**, gwarantuje **40 razy większy stopień kompresji** niż w przypadku użyciu **kodera JPEG z jakością = 100**, jednak **należy pamiętać** o tym, że **niska jakość** może znacząco wpłynąć na **liczbę zniekształceń w obrazie wyjściowym**. Użycie kodera JPEG z najlepszą jakością zagwarantowało dwukrotnie lepszy stopień kompresji niż użycie samego kodera PNG. Mówi o tym również fakt, że **przepływność bitowa dla obrazu barwnego skompresowanego koderem PNG jest znacznie większa od przepływności dla obrazu skompresowanego koderem JPEG** (niemal dwukrotnie większa). Jeśli chodzi o **przepływność bitową dla obrazu monochromatycznego**, to jest ona **najmniejsza spośród wszystkich trzech**, co wynika z **tylko jednej składowej opisującej kolor**, jednak nie jest ona znacznie mniejsza niż dla kolorowego obrazu skompresowanego koderem JPEG.