

Raport 5-6

Wiktor Krasiński ISI gr.2

Zad. 1.

a)

Poniżej kod do programu `rysuj_histogram_RGB(obraz)`:

```
def rysuj_histogram_RGB(obraz):
    hist = obraz.histogram()
    plt.figure(figsize=(16, 16))

    plt.subplot(*args: 2, 2, 1)
    plt.title("kanal R")
    plt.bar(range(256), hist[:256], color='r', alpha=0.5)

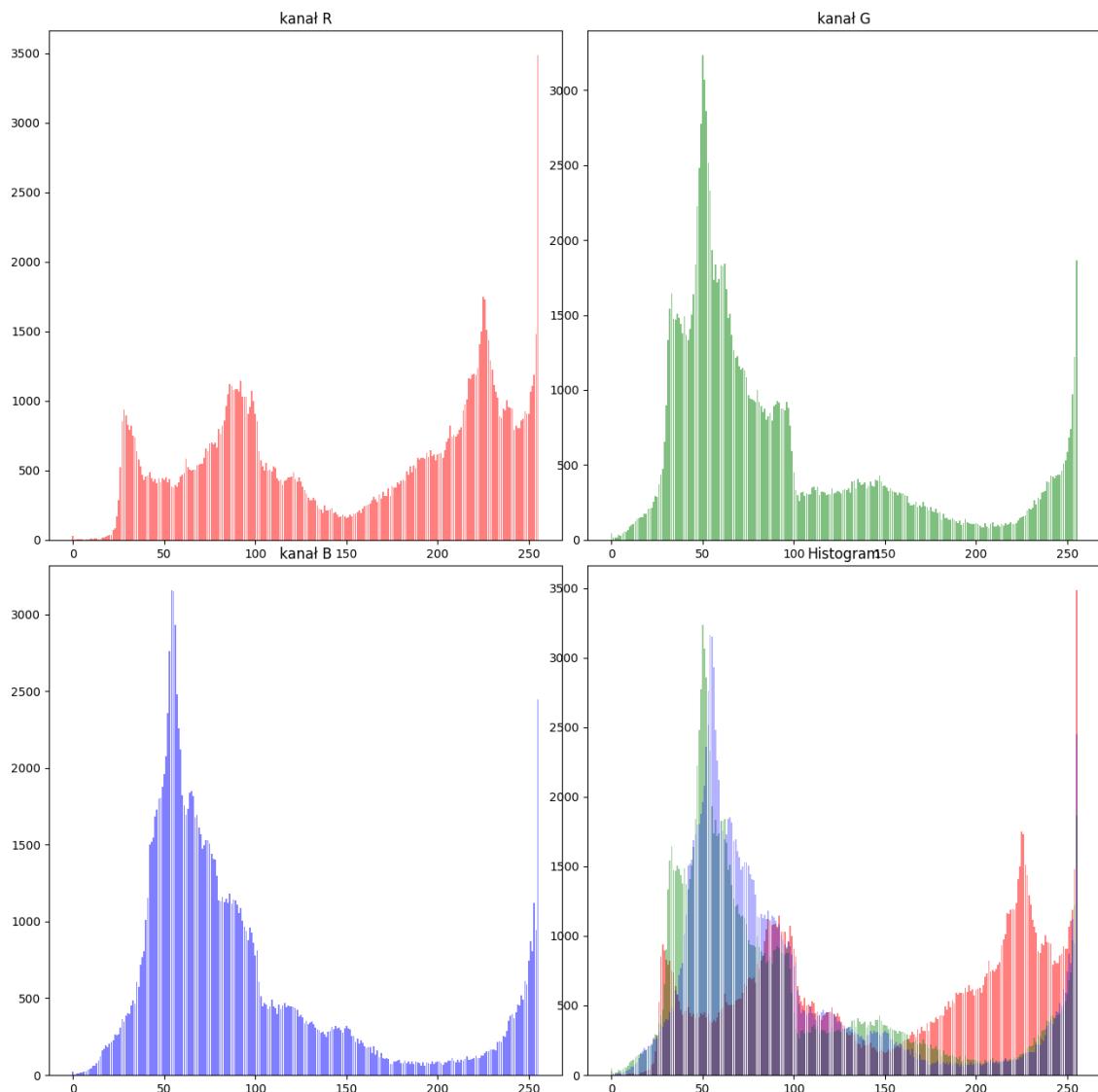
    plt.subplot(*args: 2, 2, 2)
    plt.title("kanal G")
    plt.bar(range(256), hist[256:2 * 256], color='g', alpha=0.5)

    plt.subplot(*args: 2, 2, 3)
    plt.title("kanal B")
    plt.bar(range(256), hist[2 * 256:], color='b', alpha=0.5)

    plt.subplot(*args: 2, 2, 4)
    plt.title("Histogram")
    plt.bar(range(256), hist[:256], color='r', alpha=0.5)
    plt.bar(range(256), hist[256:2 * 256], color='g', alpha=0.4)
    plt.bar(range(256), hist[2 * 256:], color='b', alpha=0.3)
    plt.subplots_adjust(wspace=0.05, hspace=0.05)
    plt.savefig("zad1a")

im = Image.open("robert_lewandowski.png")
rysuj_histogram_RGB(im)
```

Po wykonaniu kodu, otrzymujemy histogram (zad1a.png) wszystkich kanałów obrazu na diagramach plt:



b)

```
zad1b = im.histogram()
print("Kanał R: ", zad1b[155])
print("Kanał G: ", zad1b[155 + 256])
print("Kanał B: ", zad1b[155 + 2 * 256])
```

Po wykonaniu otrzymujemy ilość pikseli o wartości 155 na kanałach RGB:

b)

```
Kanał R: 186
Kanał G: 322
Kanał B: 218
```

c)

```
def zlicz_piksele(obraz, kolor):
    piksele = obraz.load()
    szer, wys = obraz.size
    licznik = 0

    for y in range(wys):
        for x in range(szer):
            if list(piksele[x, y]) == kolor:
                licznik += 1
    return licznik

zad1c = zlicz_piksele(im, kolor: [155,155,155])
print("Liczba pikseli o wartości [155,155,155]: ", zad1c)
```

Po wykonaniu otrzymujemy ilość pikseli o podanej wartości [155,155,155] w obrazie im:

c)

```
Liczba pikseli o wartości [155,155,155]: 1
```

Zad 2.

Kod do pokazania statystyk obrazów (im.png) oraz (im_jpg.jpg)

a)

```
im_jpg = Image.open("robert_lewandowski.jpg")

print("Statystyki obrazu im: \n")
statystyki(im)
print("\nStatystyki obrazu im_jpg: \n")
statystyki(im_jpg)
```

Po wykonaniu otrzymujemy dane statystyki:

Statystyki obrazu im:

```
extrema [(0, 255), (0, 255), (0, 255)]
count [147740, 147740, 147740]
mean [152.7109381345607, 94.23620549614188, 95.64481521591986]
rms [169.8480534766245, 114.4382747547689, 114.1576165305666]
median [167, 68, 73]
stddev [74.34870976594459, 64.92808562199333, 62.32199238031819]
```

Statystyki obrazu im_jpg:

```
extrema [(0, 255), (0, 255), (0, 255)]
count [147740, 147740, 147740]
mean [152.55543522404224, 94.20839312305401, 95.68431027480709]
rms [169.64203905872236, 114.15463993816668, 113.97375546658256]
median [166, 68, 73]
stddev [74.19744334951264, 64.46751495586433, 61.923579534703904]
```

Różnią się ponieważ kiedy używamy formatu JPG następuje kompresja stratna, więc tracimy część danych obrazu.

- Ekstrema się nie zmieniły
- Średnia się lekko różni, największa różnica w kanale czerwonym (około 0.2)
- RMS lekko spadł (około 0.2 – 0.3)
- Mediana przesunęła się o 1 wartość w kanale czerwonym
- Mniejsze odchylenie standardowe (lekkie rozmycie obrazu)

b)

```
roznica = ImageChops.difference(im, im_jpg)
roznica.show()
statystyki(roznica)
```

Po wykonaniu otrzymujemy:

```
extrema [(0, 47), (0, 22), (0, 34)]
count [147740, 147740, 147740]
mean [2.004873426289427, 1.3452890212535535, 1.7225328279409773]
rms [3.5951191285778803, 2.211460543750269, 2.7078107519976515]
median [1, 1, 1]
stddev [2.9841856666811246, 1.7552080178311886, 2.0892868939663947]
```

c)

```
im_jpg3 = Image.open("robert_lewandowski3.jpg")
print("Statystyki obrazu im: \n")
statystyki(im)
print("\nStatystyki obrazu im_jpg3: \n")
statystyki(im_jpg3)|
```

Statystyki obrazu im:

```
extrema [(0, 255), (0, 255), (0, 255)]
count [147740, 147740, 147740]
mean [152.7109381345607, 94.23620549614188, 95.64481521591986]
rms [169.8480534766245, 114.4382747547689, 114.1576165305666]
median [167, 68, 73]
stddev [74.34870976594459, 64.92808562199333, 62.32199238031819]
```

Statystyki obrazu im_jpg3:

```
extrema [(0, 255), (0, 255), (0, 255)]
count [147740, 147740, 147740]
mean [152.2110396642751, 94.1983213753892, 95.65370921889806]
rms [169.20208905498558, 113.83803393157189, 113.68331838776278]
median [165, 69, 74]
stddev [73.89956931465672, 63.920061165995506, 61.435045310636355]
```

Obrazy się różnią jeszcze bardziej, ponieważ kompresja stratna, za każdym razem jak zapisujemy w formacie JPG, to obraz traci część danych.

im.png vs im.jpg3

- **Ekstrema się nie zmieniły**
- **Średnia się dalej różni, najwięcej w kanale czerwonym o 0.5**
- **RMS spadł o wiele mocniej (o około 0.6 w każdym kanale)**
- **Medianą przesunęła się o 2 wartości w kanale czerwonym, 1 w zielonym i 1 w niebieskim**
- **Odchylenie standardowe różni się o wiele bardziej niż w przypadku im_jpg, aż o 1 wartość w kanale zielonym, oznacza większe rozmycie obrazu**

Zad 3.

a)

```
arr = np.asarray(im)
t_r = arr[:, :, 0]
t_g = arr[:, :, 1]
t_b = arr[:, :, 2]

im_r = Image.fromarray(t_r)
im_g = Image.fromarray(t_g)
im_b = Image.fromarray(t_b)
```

b)

```
im1 = Image.merge(mode: "RGB", bands: (im_r, im_g, im_b))
diff = ImageChops.difference(im, im1)
```

c)

```
plt.figure(figsize=(16, 16))

plt.subplot(*args: 2, 2, 1)
plt.title("Obraz im")
plt.axis('off')
plt.imshow(im)

plt.subplot(*args: 2, 2, 2)
plt.title("Obraz im1")
plt.axis('off')
plt.imshow(im1)

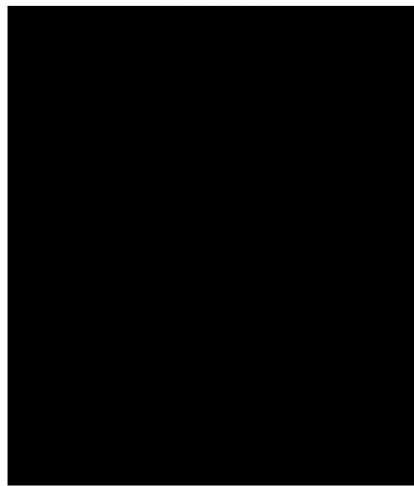
plt.subplot(*args: 2, 2, 3)
plt.title("Różnica obrazów")
plt.axis('off')
plt.imshow(diff)
plt.subplots_adjust(wspace=0.05, hspace=0.05)

# plt.savefig('fig1.png')
plt.show()
```

Po wykonaniu kodu otrzymujemy figurę (fig1.png) z obrazami:



Różnica obrazów



Statystyki diff:

```
extrema [(0, 0), (0, 0), (0, 0)]
count [147740, 147740, 147740]
mean [0.0, 0.0, 0.0]
rms [0.0, 0.0, 0.0]
median [0, 0, 0]
stddev [0.0, 0.0, 0.0]
```

d)

Nie widać różnicy w obrazach. Otrzymujemy różnice 0, co oznacza że obrazy są takie same.

Zad 4.

Poniżej kod do funkcji `mieszaj_kanaly(obraz)`

```
def mieszaj_kanaly(obraz): 1 usage
    r, g, b = obraz.split()

    nr = Image.fromarray(255 - np.array(r))
    ng = Image.fromarray(255 - np.array(g))
    nb = Image.fromarray(255 - np.array(b))

    kanal_lista = [r, g, b, nr, ng, nb]
    nowe_kanaly = [random.choice(kanal_lista) for _ in range(3)]
    return Image.merge(mode: "RGB", nowe_kanaly)
```

a)

Po wykonaniu kodu otrzymujemy obraz (`mix.png`):



b)

Poniżej kod do funkcji rozpoznaj_mix(obraz, mix)

```
def rozpoznaj_mix(obraz, mix): 1 usage
    r, g, b = obraz.split()

    nr = Image.fromarray(255 - np.array(r))
    ng = Image.fromarray(255 - np.array(g))
    nb = Image.fromarray(255 - np.array(b))

    kanal_dict = {
        "r": np.array(r),
        "g": np.array(g),
        "b": np.array(b),
        "nr": np.array(nr),
        "ng": np.array(ng),
        "nb": np.array(nb)
    }

    mr, mg, mb = mix.split()
    mix_kanaly = [np.array(mr), np.array(mg), np.array(mb)]

    wynik = []
    for mk in mix_kanaly:
        znaleziono = False
        for nazwa, kanal in kanal_dict.items():
            if np.array_equal(mk, kanal):
                wynik.append(nazwa)
                znaleziono = True
                break
        if not znaleziono:
            wynik.append("nieznany")

    return wynik
```

Po wykonaniu kodu otrzymujemy (dla obrazu mix.png z podpunktu a):

Kanał w mix [R, G, B] pochodzi z kanału:
['ng', 'g', 'nb']

Zad. 5.

Poniżej kod do sprawdzenia trybu obrazu:

```
im = Image.open('beksinski.png')
print("Tryb obrazu 'beksinski.png': ", im.mode)
im1 = Image.open('beksinski1.png')
print("Tryb obrazu 'beksinski1.png': ", im1.mode)
💡
r, g, b = im1.split()
```

Po wykonaniu kodu otrzymujemy błąd:

```
File "C:\Users\Wiktor\WprDoGrafMasz\lab5\main.py", line 227, in <module>
    r, g, b = im1.split()
    ^^^^^^^^
ValueError: too many values to unpack (expected 3)
```

Nasze polecenie `.split()` nie działa, ponieważ obraz „beksinski1” ma 4 kanały (RGBA) a my podajemy tylko 3 zmienne (RGB).

Zad. 6.

Poniżej kod do funkcji **ocen_czy_identyczne(obraz 1, obraz2)**

```
def ocen_czy_identyczne(obraz1, obraz2): 3 usages
    if obraz1.mode != obraz2.mode:
        return "obrazy nie są identyczne, bo mają różne tryby"

    if obraz1.size != obraz2.size:
        return "obrazy nie są identyczne, bo mają różne wymiary"

    arr1 = np.array(obraz1)
    arr2 = np.array(obraz2)

    if np.array_equal(arr1, arr2):
        return "obrazy identyczne"
    else:
        return "obrazy nie są identyczne, bo wartości pikseli są różne"

beksinski = Image.open("beksinski.png")
beksinski1 = Image.open("beksinski1.png")
beksinski2 = Image.open("beksinski2.png")
beksinski3 = Image.open("beksinski3.png")

print(ocen_czy_identyczne(beksinski, beksinski1))
print(ocen_czy_identyczne(beksinski, beksinski2))
print(ocen_czy_identyczne(beksinski, beksinski3))
```

Po wykonaniu kodu otrzymujemy:

```
obrazy nie są identyczne, bo mają różne tryby
obrazy nie są identyczne, bo mają różne wymiary
obrazy nie są identyczne, bo wartości pikseli są różne
```

Zad. 7.

Poniżej kod do funkcji pokaz_roznice(obraz_wejsciowy)

a)

```
def pokaz_roznice(obraz_wejsciowy): 1 usage
    arr = np.asarray(obraz_wejsciowy)
    if len(arr.shape) == 3:
        wynik = np.zeros_like(arr)

        for kanal in range(arr.shape[2]):
            max_val = arr[:, :, kanal].max()
            if max_val > 0:
                wynik[:, :, kanal] = (arr[:, :, kanal] / max_val) * 255
            else:
                wynik[:, :, kanal] = arr[:, :, kanal]

        wynik = wynik.astype(np.uint8)
        return Image.fromarray(wynik)

    else:
        max_val = arr.max()
        if max_val > 0:
            arr = (arr / max_val) * 255

        arr = arr.astype(np.uint8)
        return Image.fromarray(arr)

im_jpg3 = Image.open("im.jpg3.jpg")
```

b)

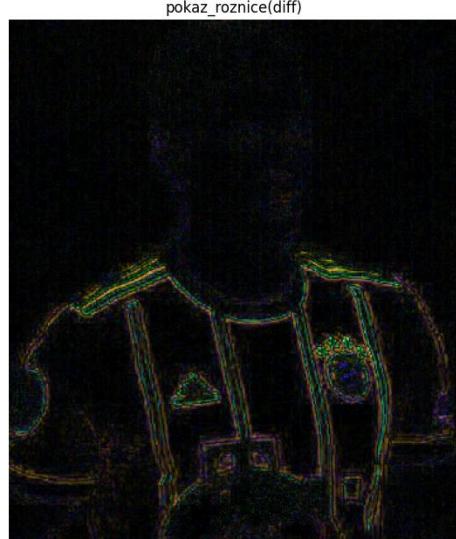
Poniżej kod do utworzenia obrazu diff:

```
im = Image.open("im.png")
diff = ImageChops.difference(im, im_jpg3)

pokaz_diff = pokaz_roznice(diff)
```

c)

Po wykonaniu kodu otrzymujemy figurę (fig2.png) z otrzymanymi obrazami:



Zad. 8.

Poniżej kod do funkcji `wstaw_inicjaly(obraz_bazowy, obraz_wstawiany, m, n, kolor)`

```
def wstaw_inicjaly(obraz_bazowy, obraz_wstawiany, m, n, kolor): 3 usages
    tab = np.asarray(obraz_bazowy, dtype=np.uint8)
    tab_start = np.copy(tab)
    tab_end = np.asarray(obraz_wstawiany)

    h_ins, w_ins = tab_end.shape
    h_baz, w_baz, dim = tab_start.shape

    m_start = max(0, m)
    n_start = max(0, n)

    m_end = min(w_baz, m + w_ins)
    n_end = min(h_baz, n + h_ins)

    for wysokosc in range(n_start, n_end):
        for szerokosc in range(m_start, m_end):
            if tab_end[wysokosc - n, szerokosc - m] == 0:
                tab_start[wysokosc, szerokosc] = kolor

    return Image.fromarray(tab_start)

inicjaly = Image.open("inicjaly.bmp")
wstawione_inicjaly = wstaw_inicjaly(im, inicjaly, m: 270, n: 0, kolor: [0, 255, 255])
wstawione_inicjaly = wstaw_inicjaly(wstawione_inicjaly, inicjaly, m: 0, n: 355, kolor: [255, 255, 0])
wstawione_inicjaly = wstaw_inicjaly(wstawione_inicjaly, inicjaly, m: 300, n: 200, kolor: [0, 255, 0])

#wstawione_inicjaly.save("obraz_inicjaly.png")
```

a) Po wykonaniu funkcji na obrazie (im.png) otrzymujemy (obraz_inicjaly.png):



Zad. 9.

a)

```
def odkoduj(obraz1, obraz2): 1 usage
    arr1 = np.asarray(obraz1)
    arr2 = np.asarray(obraz2)

    difference = np.any(arr1 != arr2, axis=2)
    diff_image_array = difference.astype(np.uint8) * 255

    diff_image = Image.fromarray(diff_image_array)

    return diff_image
```

b)

Po wykonaniu funkcji `odkoduj()` na obrazach `jesien.jpg` oraz `zakodowany1.bmp`, otrzymujemy (`kod2.bmp`):



B R A W O !!!

Wynikiem naszej funkcji `odkoduj(obraz1, obraz2)` jest obraz `kod.bmp`