# Systemy multimedialne

Daniel Brzezicki (bd46477)

Kwantyzacja obrazu i dithering – 28.03.2022

## Zadanie 1

Funkcja colorFit działającą dla obrazu kolorowego i skali odcieni szarości, która dla podanego przez was koloru piksela zwróci najbliższy mu kolor z podanej palety kolorów. Najlepiej żeby funkcja przyjmowała na wejściu dwa parametry:

- wartość koloru
- paletę kolorów w formie tabeli Nx1 - dla obrazów w skali odcieni szarości lub Nx3 dla obrazów w RGB, gdzie N to ilość kolorów w palecie.

Funkcja colorfit

```
def colorFit(colors, palette):
    return palette[np.argmin(np.linalg.norm(palette-colors,axis=1))]
```

- Funkcje realizujące dithering (0,2 pkt):
  - Losowy (dla obrazów binarnych)
  - Zorganizowany dla co najmniej M2 (4x4)
  - Floyd–Steinberga

Dithering losowy:

```python
def randomDithering(sourceIm):
    sourceIm = checkImage(sourceIm)
    image = sourceIm.copy()
    layer=1
    if len(image.shape) < 3:
        height, width = image.shape[:2]
    else:
        height, width, layer = image.shape[:3]

    for y in range(width-1):
        for x in range(height-1):
            if(random.random()<image[x,y]):
                image[x,y]=1
            else:
                image[x,y]=0

    return image
```

Dithering zorganizowany:

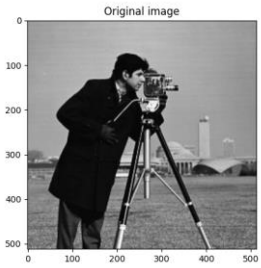
```python
def organizedDithering(sourceIm, palette):
    sourceIm = checkImage(sourceIm)

    image= sourceIm.copy()
    height, width = image.shape[:2]


    M = np.array([
        [0, 8, 2, 10],
        [12, 4, 14, 6],
        [3, 11, 1, 9],
        [15, 7, 13, 5]
        ])

    n = M.shape[0]
    mPre = (M+1)/n**2-0.5

    for y in range(width):
        for x in range(height):
            image[x,y]=colorFit(image[x,y] + mPre[y%n,x%n], palette)

    return image
```

Dithering Floyd’a-Steinberga:

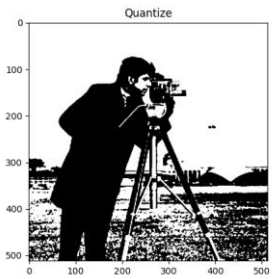
```python
def floydDithering(sourceIm, palette):
    sourceIm = checkImage(sourceIm)

    image= sourceIm.copy()
    width, height = image.shape[:2]

    for x in range(width-1):
        for y in range(height-1):
            oldpixel = image[x,y].copy()
            newpixel = colorFit(oldpixel, palette)
            image[x,y] = newpixel
            quant_error = oldpixel - newpixel
            image[x + 1,y    ] = np.clip(image[x + 1,y    ] + quant_error * 7 / 16,0,1)
            image[x - 1,y + 1] = np.clip(image[x - 1,y + 1] + quant_error * 3 / 16,0,1)
            image[x    ,y + 1] = np.clip(image[x    ,y + 1] + quant_error * 5 / 16,0,1)
            image[x + 1,y + 1] = np.clip(image[x + 1,y + 1] + quant_error * 1 / 16,0,1)

    return image
```

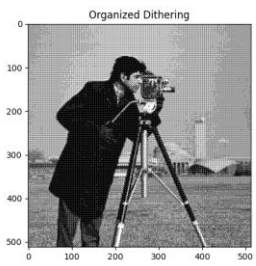
# Zadanie 2

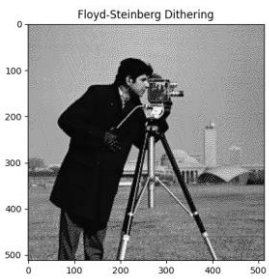
1. Porównać działanie algorytmów oraz czystej kwantyzacji:
   - Dla danych w skali odcieni szarości zapisanych na 1, 2 oraz 4 bitach.
   - Dla danych kolorowych wykorzystując podane wcześniej w instrukcji palety kolorów.

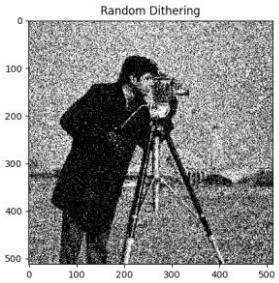



Paleta jednobitowa

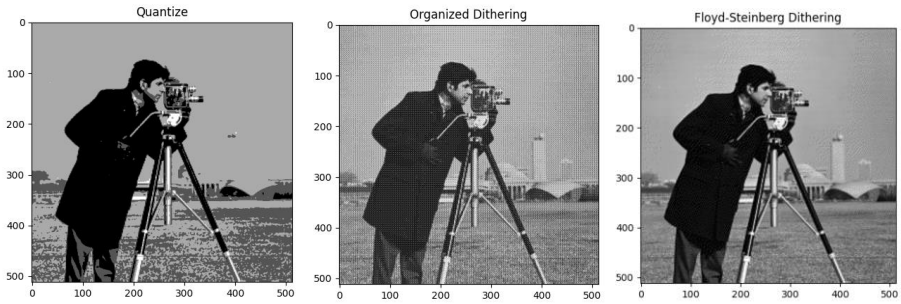


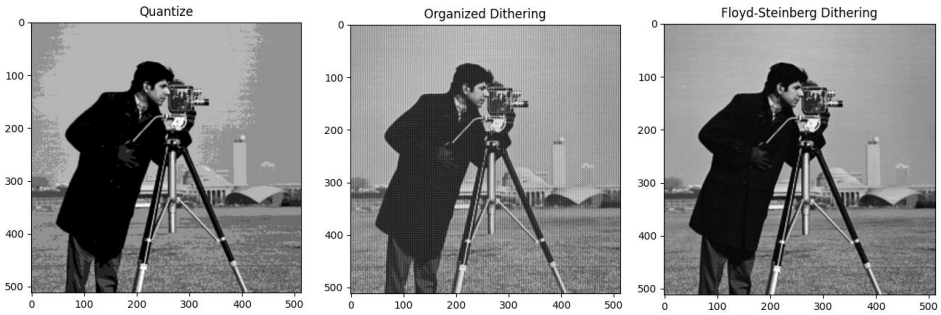




Paleta dwubitowa

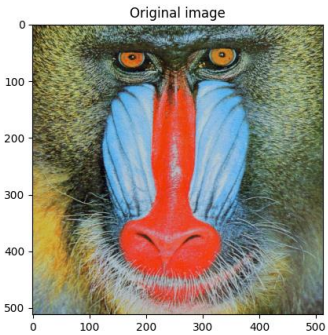


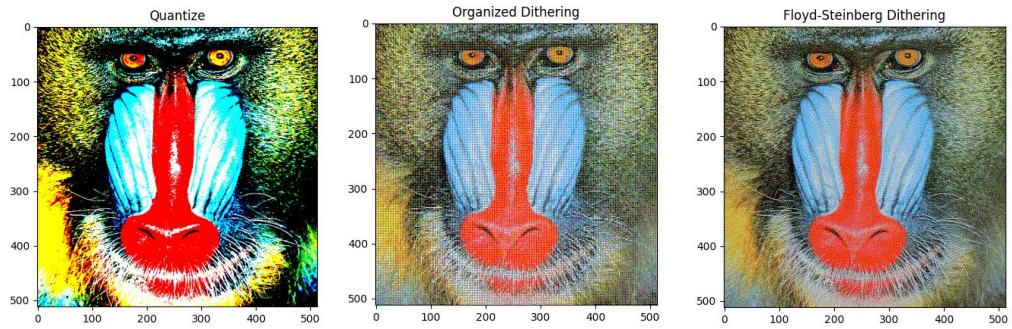

Paleta czterobitowa



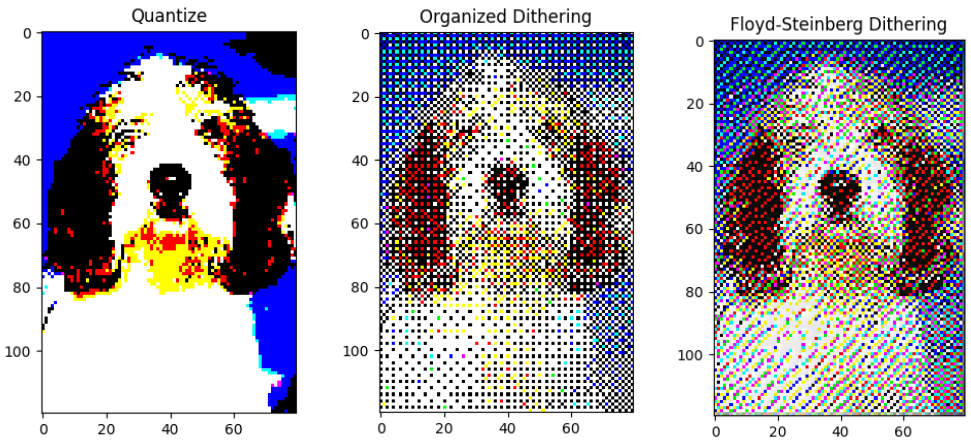

Analizując kolejno zmiany przy paletach o różnych bitach najlepiej wypada dithering Floyd'a Steinberg'a, obraz jest najbardziej zbliżony do oryginału. Dithering zorganizowany pozostawia wiele analogicznie ułożonych pikseli.  Kwantyzacja przy palecie 1 bitowej, znacznie zaburza wygląd całego obrazu, czego nie można już powiedzieć o kwantyzacji dwu i cztero-bitowej, gdzie kolory zostały artystycznie zachowane.

Paleta 8 bitowa










Wybrane do tego zostały dwa obrazy o różnych rozdzielczościach, jak widać na przykładzie ilustracji powyższych ilustracji, zmiany przy ditheringu zaczynają się zacierać, gdy mamy do czynienia z większymi obrazami. Zupełnie przeciwnie jest przy obrazach o niskiej rozdzielczości, gdzie dithering bardzo psuje jakość obrazu.

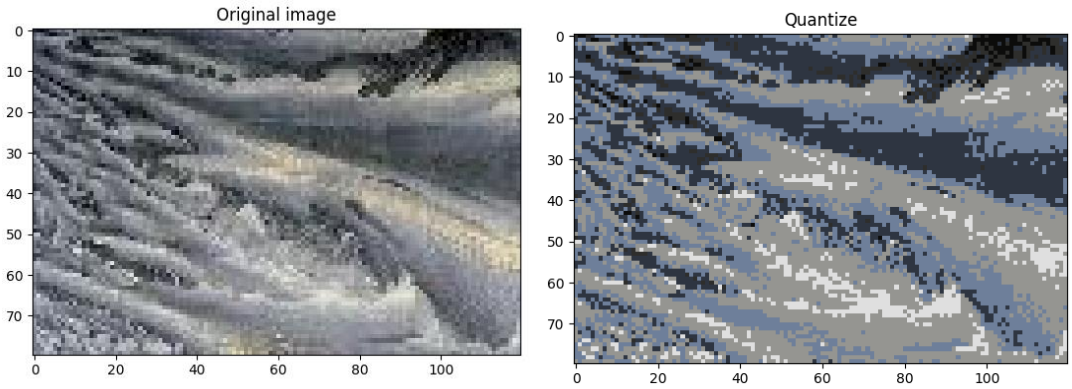
Kwantyzacja na podobnym poziomie, bez większych różnic jak w pozostałych przypadkach.

# Zadanie 3

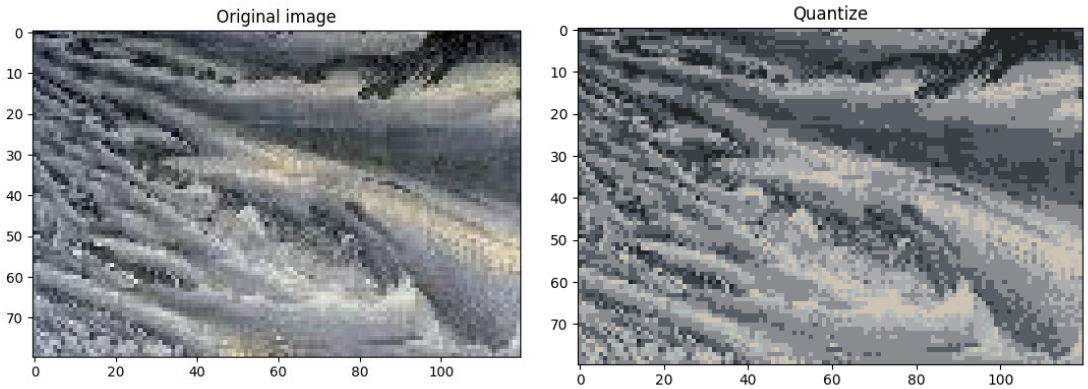
3. Dla ***jednego*** wybranego obrazu kolorowego z załączonych spróbować dobrać jak **najmniejszą** paletę (max 24 kolorów), która zadziała bez użycia żadnego z algorytmów ditheringu. (0,1 pkt)

Własna paleta barw

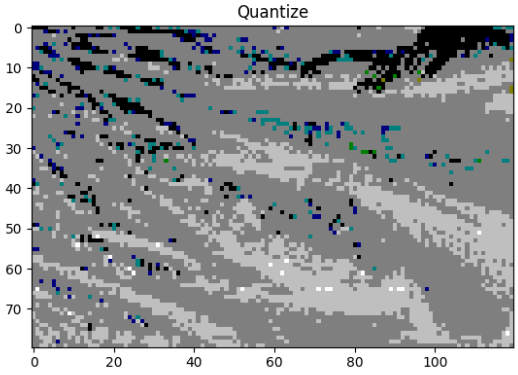
Pierwsza próba z dobraniem kolorów “na oko”




Druga próba z dobraniem kolorów wykorzystując pobrane kolory z obrazu (funkcja color picker)




Kwantyzacja z paletą 16 kolorów z zdania wyżej.



Precyzyjnie dobrane kolory dają bardzo dobre rezultaty. Na przykładzie z dobranymi kolorami przez color picker obraz wydaje się być bardzo zbliżony do oryginału, czego nie można powiedzieć o ostatnim przykładzie, z paletą 16 bitową.

Słowem zakończenia, patrząc na porównanie aspektu czasowego do jakości, najlepszą metodą ditheringu będzie dithering zorganizowany. Jeśli jednak nie zależy nam tak na czasie, lecz na jakości to dithering Floyda-Stainberga pokazał najlepsze rezulataty. Dithering losowy, zdaje się być najprostszą metodą i jednocześnie najmniej satysfakcjonującą. Kwantyzacja zaś jest bardzo dobra w połączeniu z poprawnie dobraną paletą kolorów.