Krzysztof Krupicki – lab 9

Wybierz obraz.png w trybie RGB (może być obraz z poprzednich ćwiczeń) i wczytaj jako obraz 1. Napisz funkcję filtruj(obraz, kernel, scale), która na podstawie podanej tablicy (lub listy) kernel wykonuje konwolucję (suma ważona) a następnie dzieli przez skalę scale.

```
def filtruj(obraz, kernel, scale):
    obraz_t = np.array(obraz)
    kernel_h, kernel_w = len(kernel), len(kernel[0])
    offset_y = kernel_h // 2
    offset_x = kernel_w // 2

    if obraz_t.ndim == 2:
        h, w = obraz_t.shape
        wynik = np.zeros((h, w), dtype=np.int32)
        for y in range(offset_y, h - offset_y):
            for x in range(offset_x, w - offset_x):
                sum_value = 0
                for ky in range(kernel_h):
                    for kx in range(kernel_w):
                        sum_value += int(obraz_t[y + ky - offset_y, x + kx - offset_x]) * kernel[ky][kx]
                sum_value /= scale
                wynik[y, x] = np.clip(sum_value, 0, 255)
    else:
        h, w, d = obraz_t.shape
        wynik = np.zeros_like(obraz_t, dtype=np.int32)
        for y in range(offset_y, h - offset_y):
            for x in range(offset_x, w - offset_x):
                for c in range(d):
                    sum_value = 0
                    for ky in range(kernel_h):
                        for kx in range(kernel_w):
                            sum_value += int(obraz_t[y + ky - offset_y, x + kx - offset_x, c]) * kernel[ky][kx]
                    sum_value /= scale
                    wynik[y, x, c] = np.clip(sum_value, 0, 255)

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

2. Filtr BLUR

   a. Zastosuj filtr BLUR do swojego obrazu.

   b. Pobierz informacje o filtrze BLUR, wstaw je jako parametry funkcji filtruj. Zastosuj do obrazu.

   c. Porównaj obrazy z a. i b.

```
obraz_blur = obraz.filter(ImageFilter.BLUR)
print(f"Parametry BLUR {ImageFilter.BLUR.filterargs}")
obraz_filtruj_blur = filtruj(
    obraz,
    [
    [1, 1, 1, 1, 1],
    [1, 0, 0, 0, 1],
    [1, 0, 0, 0, 1],
    [1, 0, 0, 0, 1],
    [1, 1, 1, 1, 1]
    ],
    16,
)
```

```
Parametry BLUR ((5, 5), 16, 0, (1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1))
```

Krzysztof Krupicki – lab 9

obraz BLUR

obraz filtruj() BLUR

Krzysztof Krupicki – lab 9

3. Filtr CONTOUR

a. Zastosuj filtr CONTOUR do swojego obrazu.

b. Pobierz informacje o filtrze CONTOUR, wstaw je jako parametry funkcji filtruj. Zastosuj do obrazu.

c. Porównaj obrazy z a. i b.

```
obraz_contour = obraz.filter(ImageFilter.CONTOUR)
print(f"Parametry CONTOUR {ImageFilter.CONTOUR.filterargs}")
obraz_filtruj_contour = filtruj(
    obraz,
    [
    [-1, -1, -1],
    [-1, 8, -1],
    [-1, -1, -1],
    ],
    1,
)
```

```
Parametry CONTOUR ((3, 3), 1, 255, (-1, -1, -1, -1, 8, -1, -1, -1, -1))
```

obraz CONTOUR

obraz filtruj() CONTOUR

Krzysztof Krupicki – lab 9

4. SOBEL, podobnie jak Emboss wyróżnia krawędzie. Przekonwertuj swój obraz na tryb ‘L’ (obraz.convert(‘L’)). Na tym obrazie:

   a. Zastosuj filtr EMBOSS

   b. Pobierz informacje o filtrze EMBOSS a następnie zmień zawartość listy kernel.

      i. SOBEL1: (-1, 0, 1, -2, 0, 2, -1, 0, 1). Zastosuj filtr

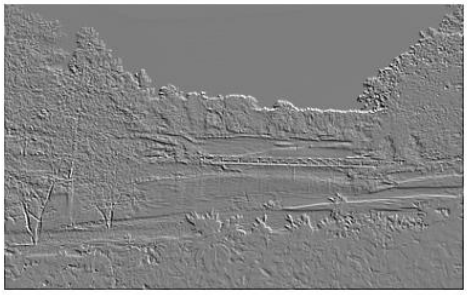
      ii. SOBEL2: (-1, -2, -1, 0, 0, 0, 1, 2, 1). Zastosuj filtr

   c. Na diagramie plt (fig2.png) umieść obraz otrzymany po konwersji na L oraz obrazy z punktów a. i b. Napisz jakie widzisz różnice między powyższymi obrazami.

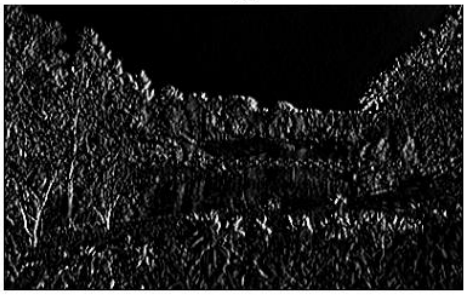
```python
obraz_L = obraz.convert("L")
obraz_emboss = obraz_L.filter(ImageFilter.EMBOSS)
print(f"Parametry EMBOSS {ImageFilter.EMBOSS.filterargs}")
obraz_filtruj_sobel1 = filtruj(
    obraz_L,
    [
        [-1, 0, 1],
        [-2, 0, 2],
        [-1, 0, 1],
    ],
    1,
)

obraz_filtruj_sobel2 = filtruj(
    obraz_L,
    [
        [-1, -2, -1],
        [0, 0, 0],
        [1, 2, 1],
    ],
    1,
)
```

obraz oryginal L

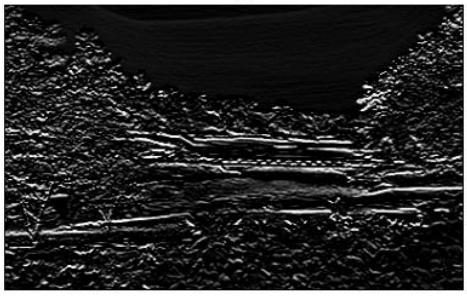
obraz EMBOSS



obraz filtruj() SOBEL1



obraz filtruj() SOBEL2



Krzysztof Krupicki – lab 9

Obraz z filtru EMBOSS daje efekt obrazu wykutego na kamieniu, wyrytego w drewnie. Tracimy informacje o jasności obiektów, ale otrzymujemy informacje o zmianach tekstury. Widać lepiej krawędzie obiektów.
Obraz z filtru SOBEL1 daje efekt wykrywania krawędzi pionowo, stają się one bardzo jasne i wyraźne, elementy poziome są bardzo mało widoczne.
Obraz z filtru SOBEL2 analogicznie do filtru SOBEL1, lecz otrzymujemy krawędzie poziome.