Raport 9

Wiktor Krasiński ISI gr. 2

Zad. 1.

Funkcja filtruj(obraz, kernel, scale), która na podstawie podanej tablicy kernel wykonuje konwolucję, a następnie dzieli przez skalę scale

Kod do funkcji:

```
def filtruj(obraz, kernel, scale):  4 usages
    size = int(len(kernel)**0.5)
    size_tuple = (size, size)
    kernel_filter = ImageFilter.Kernel(size_tuple, kernel, scale=scale, offset=0)
    obraz_po_filter = obraz.filter(kernel_filter)
    return obraz_po_filter
```

Zad. 2.

a)

Stosujemy filtr BLUR do obrazu (obraz.png)

```
obraz_blur_2a = obraz.filter(ImageFilter.BLUR)
```

b)

Pobieramy informacje o filtrze BLUR

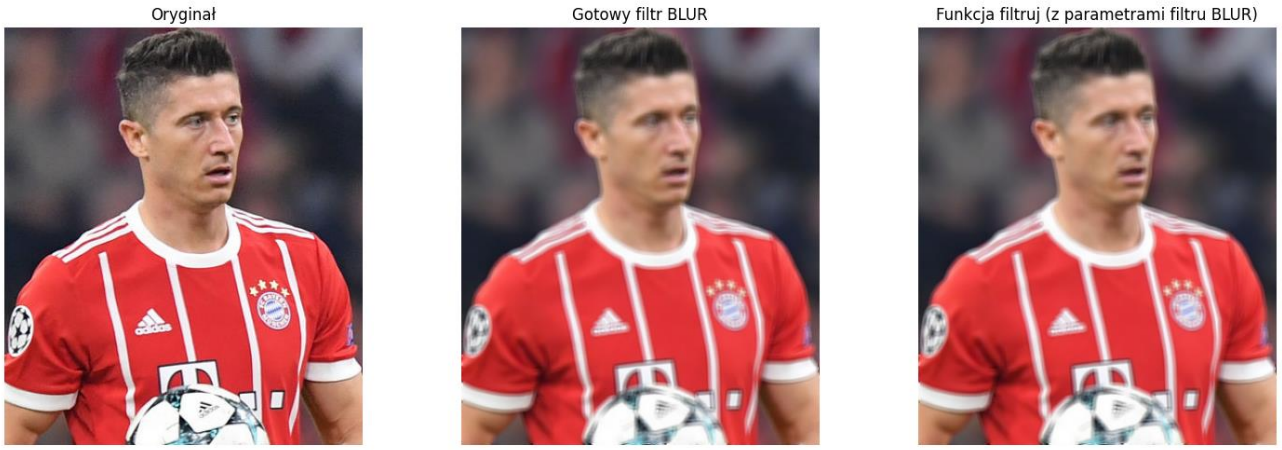
```
Informacje filtru BLUR:
((5, 5), 16, 0, (1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1))
```

Wstawiamy je jako parametry funkcji filtruj i stosujemy funkcję do obrazu (obraz.png)

```
print("Informacje filtru BLUR:")
print(ImageFilter.BLUR.filterargs)
obraz_blur_2b = filtruj(obraz,  kernel: (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),  scale: 16)
```

c)

Porównanie obrazów (diagram plt zad2.png):



Obrazy wyglądają niemalże identycznie, są tak samo rozmyte, ponieważ zaimplementowaliśmy jądro odpowiadające standardowemu filtrowi BLUR

Zad. 3.

a)

Stosujemy filtr CONTOUR do obrazu (obraz.png):

```
obraz_contour_3a = obraz.filter(ImageFilter.CONTOUR)
```

b)

Pobieramy informacje o filtrze CONTOUR

```
Informacje filtru CONTOUR:
((3, 3), 1, 255, (-1, -1, -1, -1, 8, -1, -1, -1, -1))
```

Wstawiamy je jako parametry funkcji filtruj i stosujemy funkcję do obrazu (obraz.png)

```
print("Informacje filtru CONTOUR:")
print(ImageFilter.CONTOUR.filterargs)
obraz_contour_3b = filtruj(obraz, kernel: (-1, -1, -1, -1, 8, -1, -1, -1, -1), scale: 1)
```

c)

Porównanie obrazów (diagram plt zad3.png):

Oba obrazy wykrywają kontury i krawędzie, jeden obraz z gotowym filtrem CONTOUR ma białe tło i czarne kontury, a za to nasza funkcja z parametrami filtru CONTOUR ma czarne tło i białe kontury.

Różnica wizualna wynika z tego, że fabryczny filtr CONTOUR ma dodatkową operację dodawania stałej wartości offset, za to nasza funkcja ma ustawienie standardowe offsetu równego 0.

Zad. 4.

Konwertujemy obraz na tryb ‘L’ przy użyciu .convert

```
obraz_L = obraz.convert('L')
```

a)

Stosujemy filtr EMBOSS na tym przekonwertowanym obrazie (obraz_L):

```
obraz_emboss = obraz_L.filter(ImageFilter.EMBOSS)
```

b)

Pobieramy informacje o filtrze EMBOSS

```
Informacje filtru EMBOSS:
((3, 3), 1, 128, (-1, 0, 0, 0, 1, 0, 0, 0, 0))
```

Zmieniamy wartość listy kernel dla SOBEL1 oraz SOBEL2 i stosujemy filtr

```
print("Informacje filtru EMBOSS:")
print(ImageFilter.EMBOSS.filterargs)

obraz_sobel_4b_1 = filtruj(obraz_L, kernel: (-1, 0, 1, -2, 0, 2, -1, 0, 1), scale: 1)
obraz_sobel_4b_2 = filtruj(obraz_L, kernel: (-1, -2, -1, 0, 0, 0, 1, 2, 1), scale: 1)
```

c)

Na diagramie plt (fig2.png) umieszczamy wszystkie otrzymane obrazy z punktów a) i b)

Filr EMBOSS: wyróżnia nam krawędzie poprzez cieniowanie.

Filtr SOBEL1: akcentuje krawędzie pionowe, jasne linie oznaczają krawędzie orientacji pionowej (inaczej zmiana jasności w poziomie), czarne tło

Filtr SOBEL2: akcentuje krawędzie poziomie, jasne linie oznaczają krawędzie orientacji poziomej czyli zmiana jasności w poziomie, również widać czarne tło

Porównanie **SOBEL vs EMBOSS:** Filtr EMBOSS daje efekt cienia (jaśniejsza i ciemniejsza strona krawędzi), podczas gdy filtr SOBEL daje nam czystą detekcję krawędzi w danej orientacji (poziomej lub pionowej)