Laboratorium 10

Detekcja krawędzi, progowanie, progowanie adaptacyjne, progowanie Otsu

Zadanie 1:

- Wczytaj obraz *chelsea* oraz uśrednij jego kanały barwne.
- Rozmyj obraz filtrem gaussowiskim o parametrze *sigma* równym 1. Wykorzystaj w tym celu gotową funkcję `gaussian_filter` z biblioteki `scipy`.
- Wykonaj detekcję krawędzi na dwa sposoby:

  warto zastosować funkcję `correlate` z `scipy`

  - Sposób 1: Filtracja operatorem **Laplacianu**:

$$L = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

  - Sposób 2: **Magnituda gradientu** na podstawie operatorów Prewitt:

$$P_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$P_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

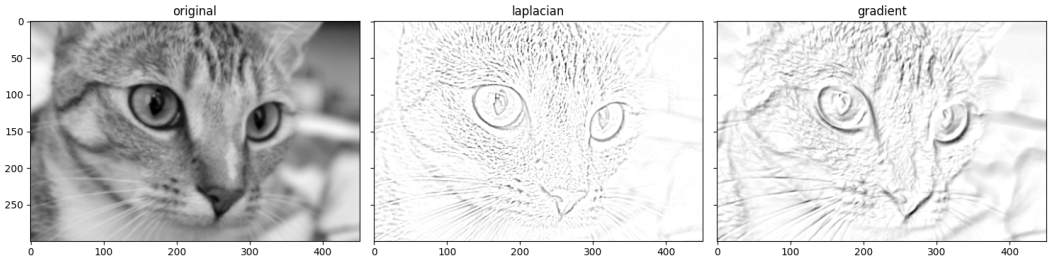
    Bezpośrednio po filtracji ustaw wartości pikseli ujemnych na 0.

    Magnitudę możesz obliczyć wyliczając sumę wartości bezwzględnych obrazów po filtracji za pomocą tych operatorów.

$$M = |G_x| + |G_y|$$

- Na wykresie przedstaw obraz oryginalny oraz efekt detekcji krawędzi (tutaj *cmapa binary*, zamiast *binary_r*).

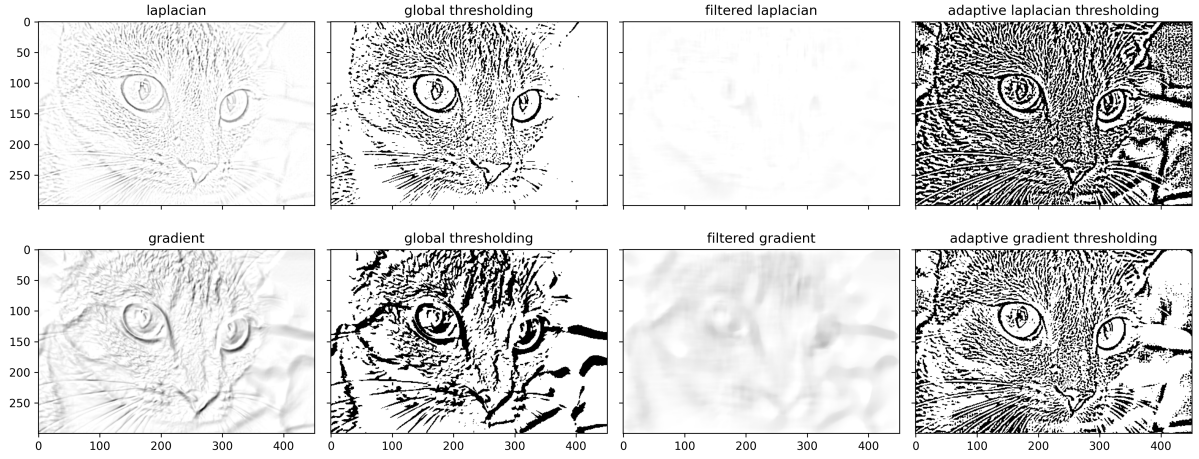
Efekt zadania 1:



Zadanie 2:

- Znormalizuj obrazy po filtracji do przedziału 0-1.
- Przygotuj wykres o czterech kolumnach i dwóch wierszach.
- W pierwszej kolumnie wykresu zaprezentuj wyniki z zadania pierwszego: efekt działania Laplacianu i magnitudę gradientu.
- W drugiej kolumnie zaprezentować wynik *progowania* obrazu dla globalnego progu równego 0.1.
- Przeprowadź *filtrację medianową* dla rozmiarów filtra 21 × 21. Warto posłużyć się gotową funkcją do tej filtracji z biblioteki `scipy`. Przedstaw efekt w trzeciej kolumnie wykresu.
- W ostatnim kroku zastusuj obraz po filtracji do **progowania adaptacyjnego** — punkt będzie uznawany za pozytywny, gdy jego wartość przekracza medianę w jego sąsiedztwie o wielkości 21 × 21.

Efekt zadania 2:



Zadanie 3:

W tym zadaniu trzeba samodzielnie zaimplementować ***progowanie Otsu*** dla obrazu po filtracji za pomocą Laplacianu (pierwsza komórka wykresu z zadania 2).

- Przygotuj wykres o dwóch wierszach i dwóch kolumnach.
- Przekształć obraz po filtracji do 8-bitowej formy cyfrowej (wartości całkowite [0,255]). Pokaż ten obraz w pierwszej komórce wykresu.
- W drugiej komórce wykresu pokaż histogram obrazu stosując skalę logarytmiczną na osi y (`.set_yscale('log')`).
- Wyznacz optymalny próg za pomocą metody *Otsu*:
  - Rozważ **wszystkie możliwe** wartości progu z zakresu [0,255]. Piksele poniżej rozważanego progu będą stanowić **klasę 0**, większe lub równe — **klasę 1**.
  - Dla każdej rozważanej wartości progu określ:
    - Liczbę pikseli w klasie 0 i klasie 1, następnie na tej podstawie **prawdopodobieństwo a priori** wystąpienia obiektu klasy: $P_k = \frac{|c_k|}{N}$ gdzie N oznacza liczbę pikseli obrazu a $|c_k|$ liczbę pikseli w danej klasie.

    - Średnią wartość intensywności w klasach m_k.
    - Średnią wartość intensywności całego obrazu m_G.
- Na podstawie tych wartości oblicz *wariancję pomiędzy klasami*:

$$v = P_0 \cdot (m_0 - m_G)^2 + P_1 \cdot (m_1 - m_G)^2$$

- Po obliczeniu wariancji dla wszystkich możliwych wartości progu znajdź optymalny próg — taki, dla którego wariancja jest maksymalna. Warto posłużyć się funkcją `argmax`.

  Uwaga: nie dla wszystkich progów da się obliczyć wariancję (efektem będzie NaN) — dla takich należy ustawić wartość wariancji na 0.
- W trzeciej komórce wykresu narysuj wykres wariancji dla każdego rozważonego progu. Czerwonym odcinkiem pionowym (funkcja `vlines`) zaznacz optymalny próg.
- W ostatniej komórce wykonaj pogowanie za pomocą optymalnego progu.

Efekt zadania 3: