Inżynieria obrazów

Przekształcenia morfologiczne binarne

Marcin Lasak 272886

Spis treści

1. Wstęp teoretyczny

1.1 Morfologia matematyczna

Morfologia matematyczna jest metodą analizy struktury obiektów bazującą na zagadnieniach teorii mnogości. Obrazy postrzegane są jako zbiory punktów przestrzeni o dowolnym wymiarze. Dla obrazów binarnych przyjmuje się, że punkty należące do obiektu przyjmują wartość „1", natomiast piksele tła – wartość „0". Każdy obraz posiada tzw. punkt centralny o współrzędnych (0,0) (dla przestrzeni dwuwymiarowej).

Podstawową zasadą morfologii matematycznej jest użycie do analizy obrazu, innego, zazwyczaj dużo mniejszego obrazu zwanego elementem strukturalnym. Jest on swoistego rodzaju wzorcem, którego szukamy w przetwarzanym obrazie. Stanowi on geometryczny parametr przekształceń morfologii matematycznej. Posiada on swój własny punkt centralny, względem którego określanie są współrzędne punktów należących do niego[1].

Wyróżniamy dwie podstawowe operacje morfologii matematycznej: erozję oraz dylatację.

1.2 Dodawanie Minkowskiego

Dla zrozumienia operacji erozji oraz dylatacji, kluczowe jest wprowadzenie działania dodawania Minkowskiego. Jest to działanie na zbiorach, które można określać jako translację elementów zbioru A o zbiór wektorów B[2].

$$A \oplus B = \{a + b : a \in A \wedge b \in B\}$$

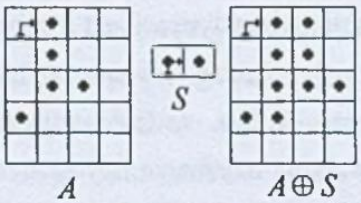
1.3 Dylatacja

Dylatacja (dylacja), zwana także rozszerzaniem, jest zastosowaniem do zbioru punktów obrazu sumy Minkowskiego. Polega na wykonaniu dodawania Minkowskiego na zbiorze punktów obrazu oraz elemencie strukturalnym. Możemy zdefiniować więc dylatację w sposób następujący: niech $A, S \in E^2$, gdzie A – zbiór punktów obrazu, S – element strukturalny, E^2 – dyskretny zbiór nadrzędny. Dylatację określimy wtedy następującym wzorem:

$$A \oplus S = \{c \in E^n : c = a + s \wedge a \in A \wedge s \in S\}$$

Przykład 2.1.1

Operacja dylatacji zbioru $A = \{(0,1),(1,1),(2,1),(2,2),(3,0)\}$ poprzez element strukturalny $S = \{(0,0),(0,1)\}$ pokazana jest na rysunku rys. 1.



Rys. 1. Przykład dylatacji
Fig. 1. An example of dilation

Rysunek 1 Przykład operacji dylatacji

[1] Stańczyk Urszula. *Morfologia matematyczna – pojęcia podstawowe.* Studia Informatica. 2000, Vol. 21, nr 3, s. 27-53

[2] https://faculty.sites.iastate.edu/jia/files/inline-files/15.%20Minkowski%20sum.pdf

Operacja ta skutkuje zwiększeniem rozmiarów obiektów. Znikają detale; zostają wypełnione wszelkie „dziury”, obiekty położone blisko siebie zostają połączone. Element strukturalny określa kierunek rozrostu[3].

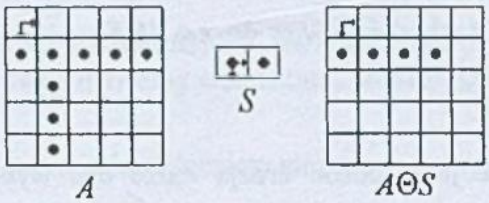
1.4 Erozja

Erozja, zwana również zwężaniem (kurczeniem, redukowaniem), jest zastosowaniem działania dualnego do sumy Minkowskiego, tj. różnicy Minkowskiego. Definiujemy ją jako dopełnienie sumy Minkowskiego dopełnienia A oraz odbicia B względem początku układu współrzędnych (-B). Oznacza to, że różnicą Minkowskiego zbioru A i B, będzie zbiór wszystkich $a \in A$, dla których $x + s \in A$, dla dowolnego $b \in B$. Zatem erozję możemy zdefiniować w sposób następujący: niech $A, S \in E^2$, gdzie A – zbiór punktów obrazu, S – element strukturalny, E^2 – dyskretny zbiór nadrzędny. Erozję określimy wtedy następującym wzorem:

$$A \ominus S = \{x \in E^2 : \forall (s \in S) x + s \in A\}$$

Przykład 2.2.1

Operacja erozji zbioru $A = \{(1,0),(1,1),(1,2),(1,3),(1,4),(2,1),(3,1),(4,1)\}$ poprzez element strukturalny $S = \{(0,0),(0,1)\}$, $A \Theta S = \{(1,0),(1,1),(1,2),(1,3)\}$ pokazana jest na rysunku 7.



Rys. 7. Przykład operacji erozji
Fig. 7. An example of erosion

Rysunek 2 Przykład operacji erozji

Operacja erozji eliminuje bądź wygładza drobne szczegóły. Pomniejsza sylwetki obiektów i niweluje szum. Element strukturalny wpływa na kierunek zwężania.

Operacje dylatacji i erozji zostały zdefiniowane dla przestrzeni dwuwymiarowych, lecz możemy je oczywiście rozszerzyć na dowolne dyskretne przestrzenie n-wymiarowe.

1.5 Otwieranie i zamykanie

Operacje dylatacji oraz erozji pozwalają nam zdefiniować operacje otwierania i zamykania (domykania).

Otwarcie definiujemy jako wykonanie operacji dylatacji na obrazie poddanemu operacji erozji.

Domknięcie definiujemy jako wykonanie operacji erozji na obrazie poddanemu operacji dylatacji.

[3] https://kcir.pwr.edu.pl/~jr/cpois/wyklady/wyklad10.pdf

1.6 Gradient morfologiczny

Pojęcie gradientu morfologicznego nie jest jednoznaczne. Generalnie określa ono operatory wzmacniające zmiany pikseli w zadanym sąsiedztwie. W celu określenia gradientu korzystamy z trzech różnych kombinacji operatorów:

- różnica arytmetyczna między operatorem ekstensywnym φ, a operatorem anty-ekstensywnym ψ: $\varphi(f) - \psi(f)$
- różnica arytmetyczna między operatorem ekstensywnym φ, a oryginalną funkcją f: $\varphi(f) - f$
- różnica arytmetyczna między oryginalną funkcją f, a operatorem anty-ekstensywnym ψ: $f - \psi(f)$

Operator nazywamy ekstensywnym wtedy i tylko wtedy, gdy:

$$\forall(f)\ \varphi(f) > f$$

Operator nazywamy anty-ekstensywnym wtedy i tylko wtedy, gdy:

$$\forall(f)\ \psi(f) < f$$

Tak zdefiniowane operacje pozwolą nam na określenie gradientu morfologicznego na obrazie przy operacjach dylatacji i erozji[4].

W naszym przypadku operatorem ekstensywnym jest dylatacja, a operatorem anty-ekstensywnym – erozja.

Dla obrazów cyfrowych typowo wykorzystuje się gradient pełny określony jako różnica pomiędzy operatorem ekstensywnym, a antyekstensywnym.

Dwie pozostałe różnice określa się kolejno mianem gradientu zewnętrznego (g^+) i wewnętrznego (g^-). Gradient pełny (opisany wyżej) jest sumą gradientów częściowych.

$$g(f) = g^+(f) + g^-(f) = (\varphi(f) - f) + \big(f - \psi(f)\big) = \varphi(f) - \psi(f)$$

1.7 Morfologiczny operator Laplace'a

Jedna z klasycznych metod detekcji konturów, polega na analizie zerowych przecięć (0-crossings) dyskretnego operatora Laplace'a. Morfologiczny operator Laplace'a definiujemy w następujący sposób[5]:

$$\Delta_{\square} = (\varphi(f) - f) - \big(f - \psi(f)\big) = \varphi(f) + \psi(f) - 2f$$

Widzimy, że morfologiczny operator Laplace'a jest różnicą pomiędzy gradientem zewnętrznym, a wewnętrznym.

---

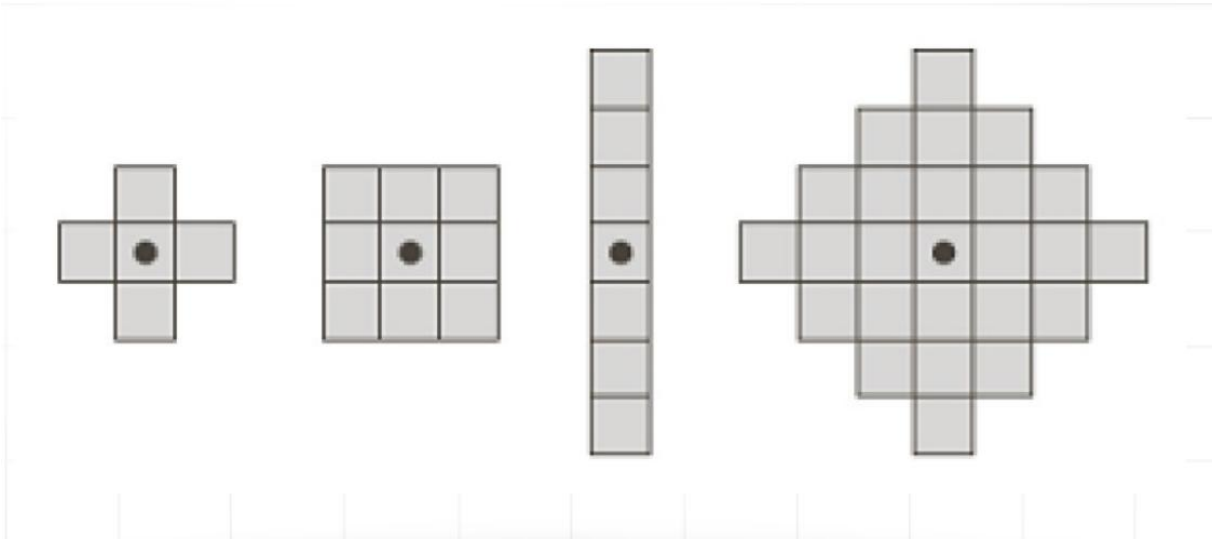
[4] Rivest Jean-F., Soille Pierre, Beucher Serge. *Morphological gradients.* Journal of Electronic Imaging. 1993, Vol. 2, nr 4, s. 326-336

[5] Lê Duy Huỳnh, Xu Yongchao, Géraud Thierry. *Morphological Hierarchical Image Decomposition Based on Laplacian 0-Crossings*. Lecture Notes in Computer Science, 2017, Vol. 10225, s. 159–171

2. Zadania

W następujących zadaniach będziemy używać języka Python oraz biblioteki OpenCV, umożliwiającą wykonywanie operacji morfologicznych. Dodatkowo używana będzie biblioteka numpy – do obliczeń i tworzenia macierzy (dwuwymiarowe tablice numpy) oraz Pillow – do tworzenia obrazów z tablic numpy.

Poniższe zadania wykorzystują elementy strukturalne zadane rysunkiem nr 3.



Rysunek 3

Dlatego też, przed właściwą implementacją rozwiązań zadań, zaimplementowano funkcje generujące zadane elementy strukturalne. Rysunek nr 4 prezentuje implementacje powyższych elementów strukturalnych w kolejności od lewej do prawej.

Funkcje te zwracają macierze reprezentujące element strukturalny, jednakże bez określenia punktu centralnego (anchor). Aby ułatwić wybór macierzy elementu strukturalnego oraz określić punkt centralny, zaimplementowano funkcję choose_kernel() zadaną rysunkiem nr 5. W zależności od wybranego numeru elementu strukturalnego, zwraca ona jego macierz oraz współrzędne punktu centralnego.

Funkcje te oraz rozwiązania zadań dane są plikiem lab6.py. Plik main.py zawiera wywołania funkcji wykonujących określone zadanie. Na podstawie pliku main.py powstał plik wykonywalny main.exe z wykorzystaniem narzędzia PyInstaller.

lab6.py

```python
def kernel1():
    kernel = np.array([[0, 1, 0],
                       [1, 1, 1],
                       [0, 1, 0]], dtype=np.uint8)
    return kernel

def kernel2():
    kernel = np.array([[1, 1, 1],
                       [1, 1, 1],
                       [1, 1, 1]], dtype=np.uint8)
    return kernel

def kernel3():
    kernel  = np.array([[1],[1], [1],[1],[1], [1],[1]], dtype=np.uint8)
    return kernel


def kernel4():
    kernel = np.array([[0, 0, 0, 1, 0, 0, 0],
                       [0, 0, 1, 1, 1, 0, 0],
                       [0, 1, 1, 1, 1, 1, 0],
                       [1, 1, 1, 1, 1, 1, 1],
                       [0, 1, 1, 1, 1, 1, 0],
                       [0, 0, 1, 1, 1, 0, 0],
                       [0, 0, 0, 1, 0, 0, 0]
                         ], dtype=np.uint8)
    return kernel
```

Rysunek 4

lab6.py

```python
def choose_kernel(kernel_no):
    kernel = np.array([0])
    anchor = (0, 0)

    if kernel_no == 1:
        kernel = kernel1()
        anchor = (1, 1)

    if kernel_no == 2:
        kernel = kernel2()
        anchor = (1, 1)

    if kernel_no == 3:
        kernel = kernel3()
        anchor = (0, 4)

    if kernel_no == 4:
        kernel = kernel4()
        anchor = (4, 4)

    return kernel, anchor
```

Rysunek 5

2.1 Erozja

W tym zadaniu, należało wykonać operację erozji na obrazie test2.png (rysunek 6) z wykorzystaniem zadanych elementów strukturalnych. Dodatkowo należało obliczyć różnice między obrazem oryginalnym, a zerodowanym.

Rysunek 6

W tym celu zaimplementowano funkcję erode() (rysunek 7) przyjmującą jako argument nazwę pliku graficznego oraz numer używanego elementu strukturalnego.

lab6.py

```
def erode(name,kernel_no):

    image = cv2.imread(name, cv2.IMREAD_GRAYSCALE)
    _, binary = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)

    kernel, anchor = choose_kernel(kernel_no)

    eroded = cv2.erode(binary, kernel,anchor=anchor,iterations=1)

    difference = cv2.subtract(binary, eroded)

    img = Image.fromarray(eroded)
    img.save(name+"_eroded"+str(kernel_no)+".png")

    img2 = Image.fromarray(difference)
    img2.save(name+"_difference"+str(kernel_no)+".png")

    return eroded
```

Rysunek 7

Funkcja ta wczytuje zadany obraz w skali szarości i przekształca go w obraz binarny. Następnie pobiera dane o elemencie strukturalnym, aby wykonać operację erozji oraz obliczyć gradient

wewnętrzny obrazu. Wynikowe tablice zostają zapisane ze stosownymi nazwami. Funkcja ta zwraca macierz obrazu zerodowanego, co będzie wykorzystane w kolejnych zadaniach.

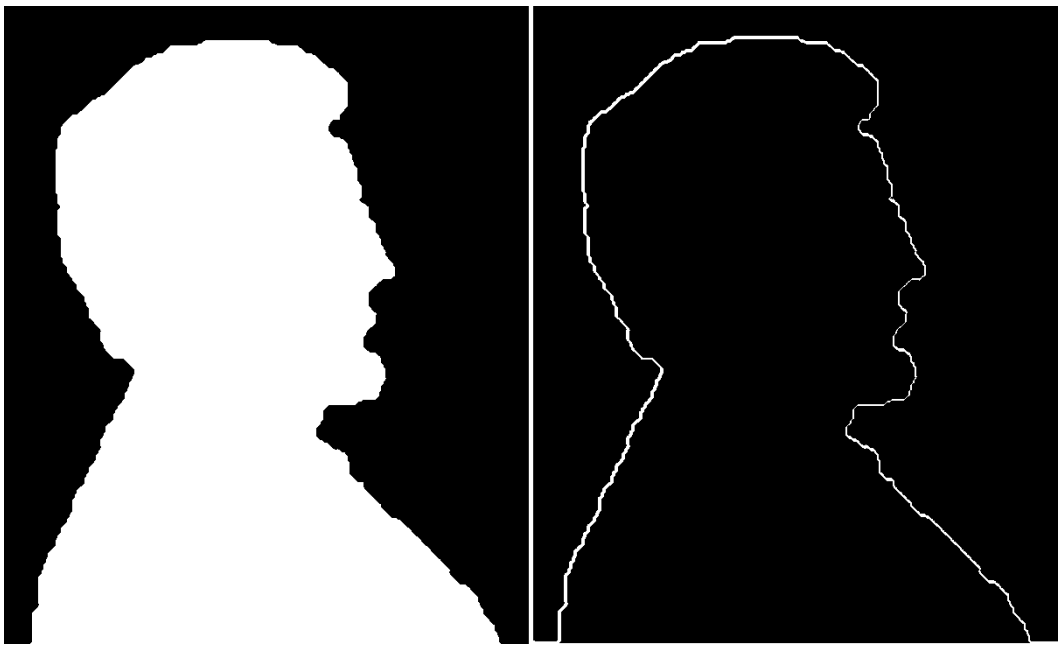
Następnie zaimplementowano funkcję zad1(), która wywołuje erode() dla każdego z czterech elementów strukturalnych. Funkcja ta zostaje wywołana po wybraniu opcji nr 1 w programie main.exe.

Wyniki działania programu dane są rysunkami 8-11 – po lewej obraz zerodowany, po prawej gradient wewnętrzny.

Rysunek 8

Rysunek 9

Rysunek 10



Rysunek 11

Zauważmy nieregularny kształt gradientu wewnętrznego dla elementu strukturalnego nr 3 (rysunek 10). Dla wszystkich innych elementów strukturalnych, obserwujemy dość zrównoważone zmiany, z uwagi na symetrię macierzy w płaszczyźnie pionowej i poziomej. W przypadku elementu strukturalnego nr 3, jego wysokość jest większa od szerokości, stąd nieregularne zmiany na rysunku nr 10 (zwężanie następuje tylko w kierunku wertykalnym).

2.2 Dylatacja

W kolejnym zadaniu należało przeprowadzić operację dylatacji (dylacji). W tym celu zaimplementowano funkcję dilate() daną rysunkiem nr 12. Funkcja ta jest stworzona analogicznie do funkcji erode() z pominięciem obliczania gradientu składowego.

lab6.py

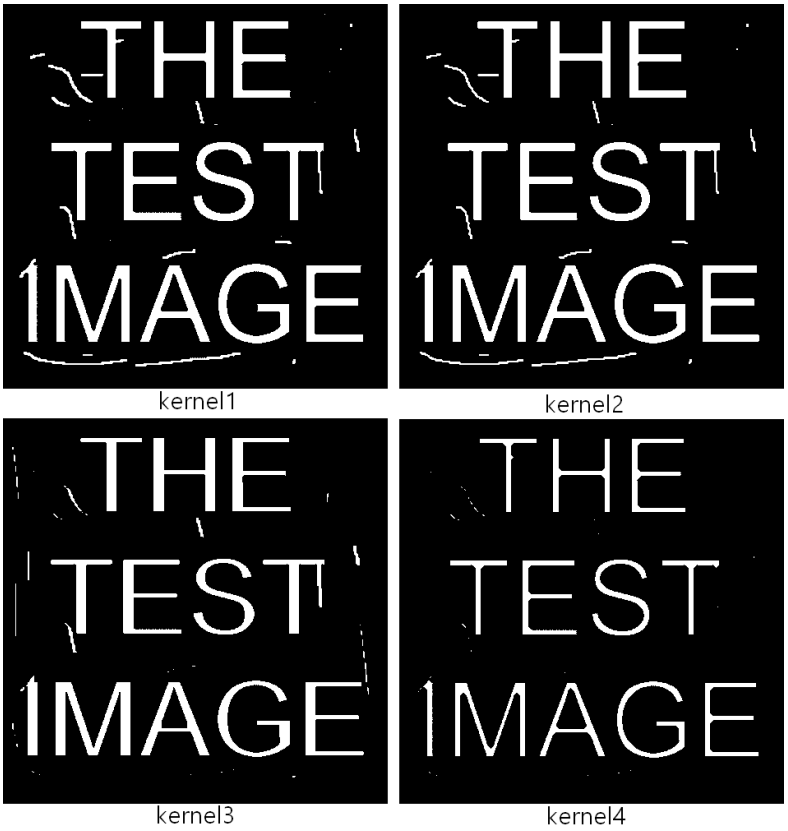
```
def dilate(name,kernel_no):

    image = cv2.imread(name, cv2.IMREAD_GRAYSCALE)
    _, binary = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)

    kernel, anchor = choose_kernel(kernel_no)

    dilated = cv2.dilate(binary, kernel,anchor=anchor, iterations=1)

    img = Image.fromarray(dilated)
    img.save(name+"_dilated"+str(kernel_no)+".png")

    return dilated
```

Rysunek 12

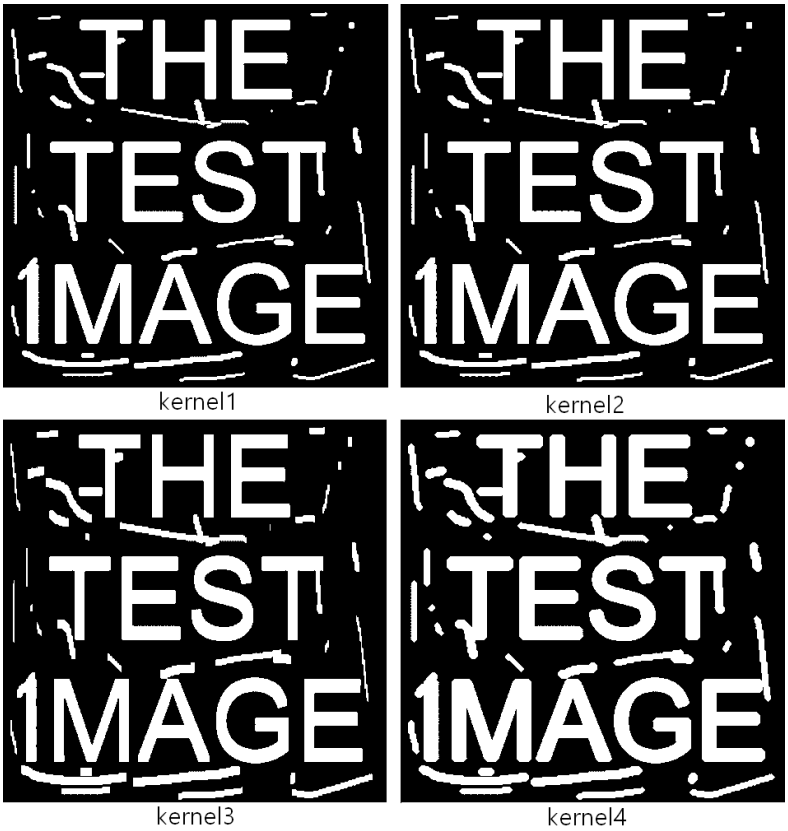
Następnie zaimplementowano funkcję zad2(), która wykonuje operacje erozji i dylacji dla każdego z czterech elementów strukturalnych na obrazie test1.png (rysunek 13). Wybranie opcji nr 2 w programie main.exe skutkuje wywołaniem tej funkcji.

Rysunek 13

Wyniki działania programu dane są rysunkami 14 i 15.



Rysunek 14



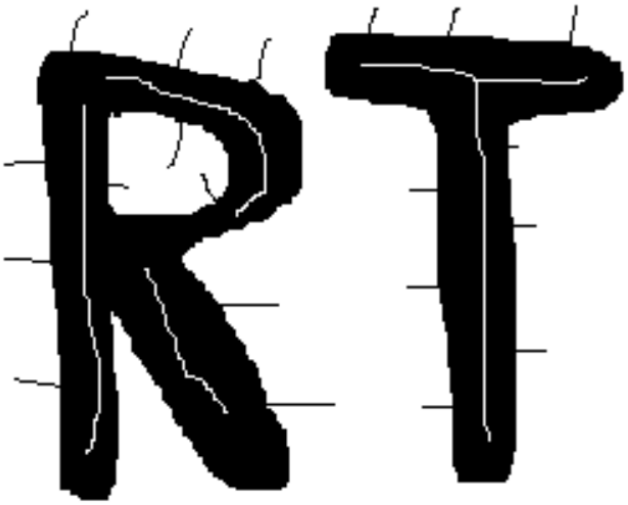
Rysunek 15

Na tych obrazach dużo lepiej widać, skąd potoczne nazwy „zwężanie' i „rozszerzanie". Zgodnie z zadaną definicją operacji ekstensywnej i anty-ekstensywnej, w pierwszym przypadku (rysunek 15) rozszerzamy zbiór ($\forall(f)\ \varphi(f) > f$), a w drugim (rysunek 14) – ograniczamy ($\forall(f)\ \psi(f) < f$).

Zauważmy znów pojawiającą się „schodkowość" w wynikach operacji z wykorzystaniem elementu strukturalnego nr 3. Widzimy dobrze, że operacja erozji i dylatacji jest przeprowadzana w kierunku wertykalnym. Efekt dla linii o nachyleniu większym od 45 stopni jest podobny do próby rysowania linii algorytmem naiwnym (bez wykorzystania algorytmu Bresenhama).

2.3 Domknięcie

Kolejne zadanie polegało na zaimplementowaniu operacji domknięcia oraz wykonanie jej dla zadanych elementów strukturalnych na obrazie test3.png (rysunek 16).



Rysunek 16

Funkcja close() implementująca domknięcie dana jest rysunkiem nr 17. Wykorzystano w niej zaimplementowane wcześniej funkcje erode() i dilate().

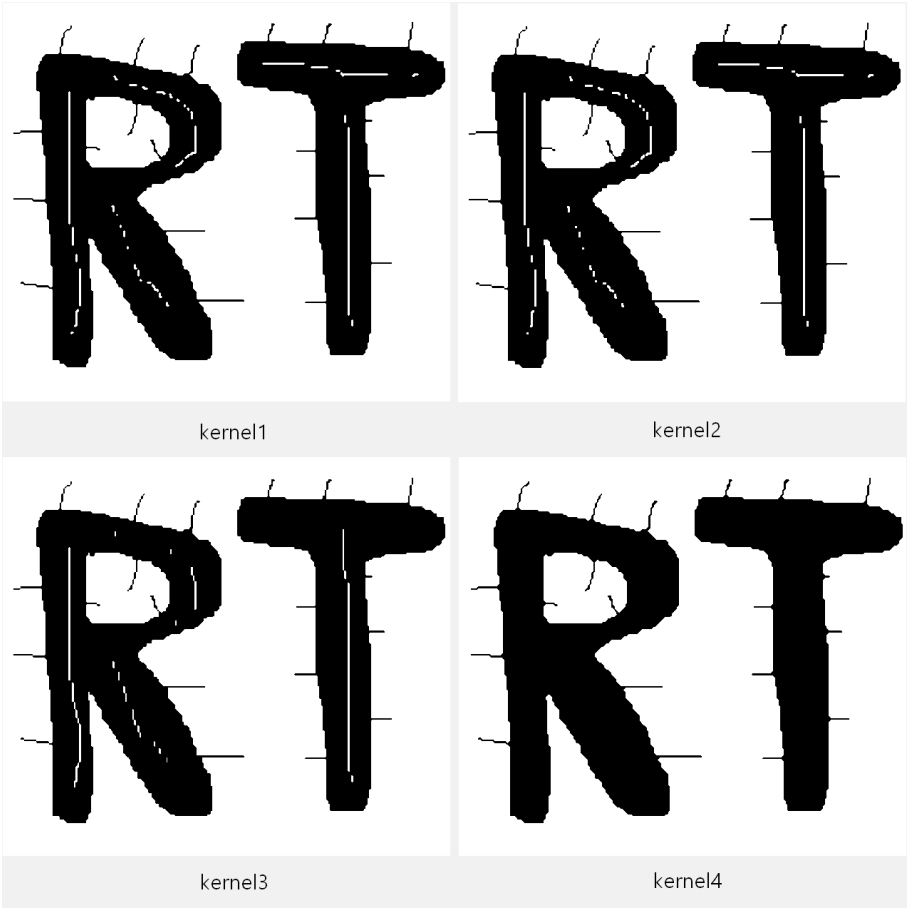
lab6.py

```
def close(name,kernel_no):
    erode(name,kernel_no)
    dilate(name+"_eroded"+str(kernel_no)+".png",kernel_no)
```

Rysunek 17

Następnie utworzono funkcję zad3(), która wywołuje funkcję close() dla obrazu test3.png dla każdego z zadanych elementów strukturalnych. Wybranie opcji nr 3 w programie main.exe skutkuje wywołaniem tej funkcji.

Wynik działania programu dany jest rysunkiem nr 18.



Rysunek 18

2.4 Otwarcie

W tym zadaniu należało zaimplementować i wykonać na obrazie test3.png (rysunek 16) operację otwarcia. Utworzono zatem funkcję open(), wykorzystującą zaimplementowane wcześniej funkcje dilate() i erode(). Funkcja ta dana jest rysunkiem nr 19.

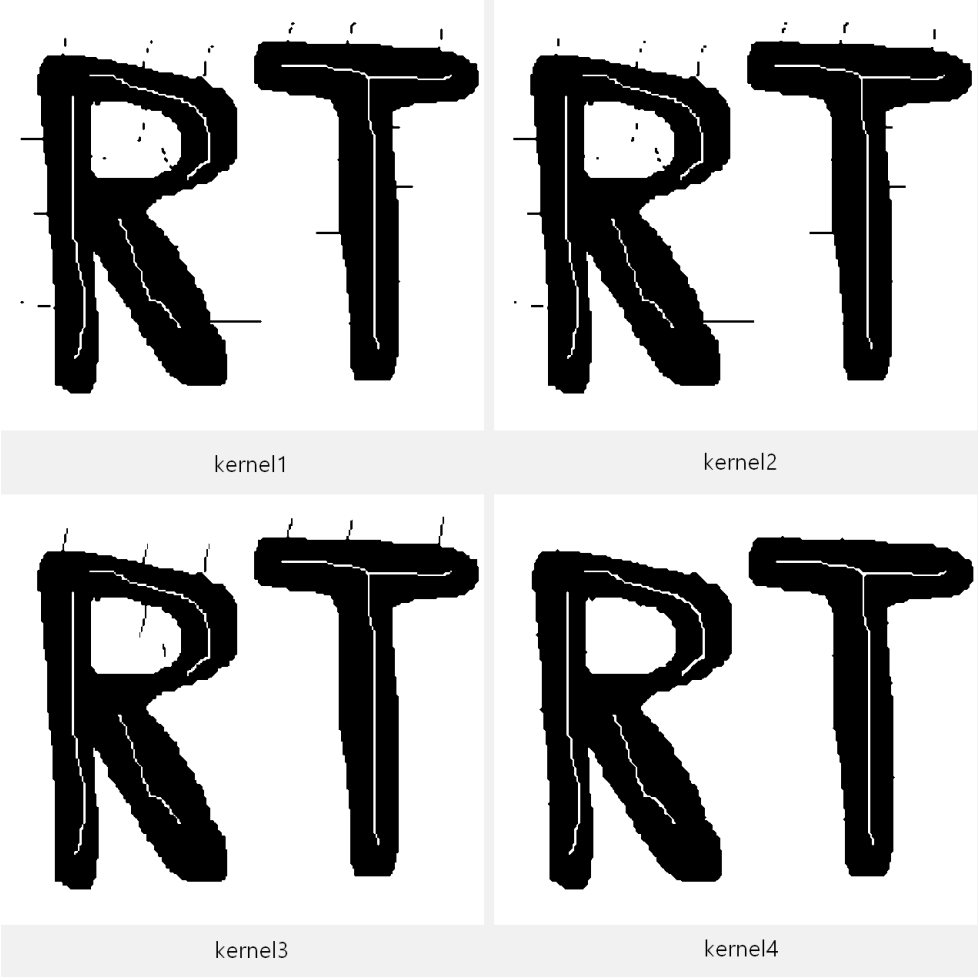
lab6.py

```
def open(name,kernel_no):
    dilate(name, kernel_no)
    erode(name+"_dilated"+str(kernel_no)+".png",kernel_no)
```

Rysunek 19

Następnie zaimplementowano funkcję zad4() wykonującą operację otwarcia dla każdego z zadanych elementów strukturalnych. Wybranie opcji nr 4 w programie main.exe skutkuje wywołaniem tej funkcji.

Wynik działania programu dany jest rysunkiem nr 20.



Rysunek 20

2.5 Gradient i operator Laplace'a

Ostatnie zadanie polegało na zaimplementowaniu wyznaczania gradientu oraz laplasjanu dla zadanego obrazu. W tym celu zaimplementowano funkcję gradient_laplacian() daną rysunkiem nr 21.

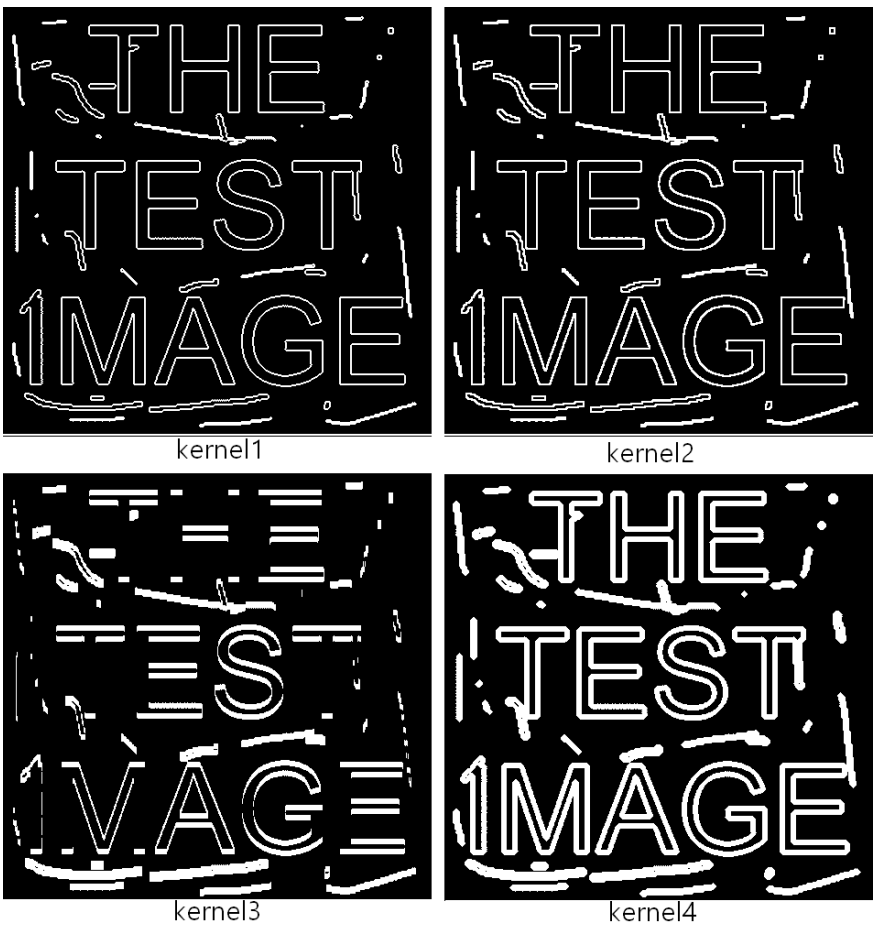
Rozpoczyna się ona analogicznie do dilate() i erode(); wczytujemy obraz w skali szarości i przekształcamy go do postaci binarnej. Następnie wykonujemy operację dylatacji i erozji i przechowujemy tablice wynikowe tych operacji. W przypadku gradientu, wystarczy odjąć wynik erozji od wyniku dylatacji, wg wzoru z punktu 1.6. W przypadku operatora Laplace'a, zmieniono typ danych, aby zachować poprawność obliczeń (przykładowo 0+0-2*255 już skutkuje wyjściem poza zakres uint8). Wpierw zmieniono typ z całkowitoliczbowego ośmiobitowego na całkowitoliczbowy szesnastobitowy, a następie obliczono laplasjan zgodnie ze wzorem z punktu 1.7. Wynik zapisano ponownie w uint8. Gradient oraz laplasjan zapisywane są jako obrazy png.

lab6.py

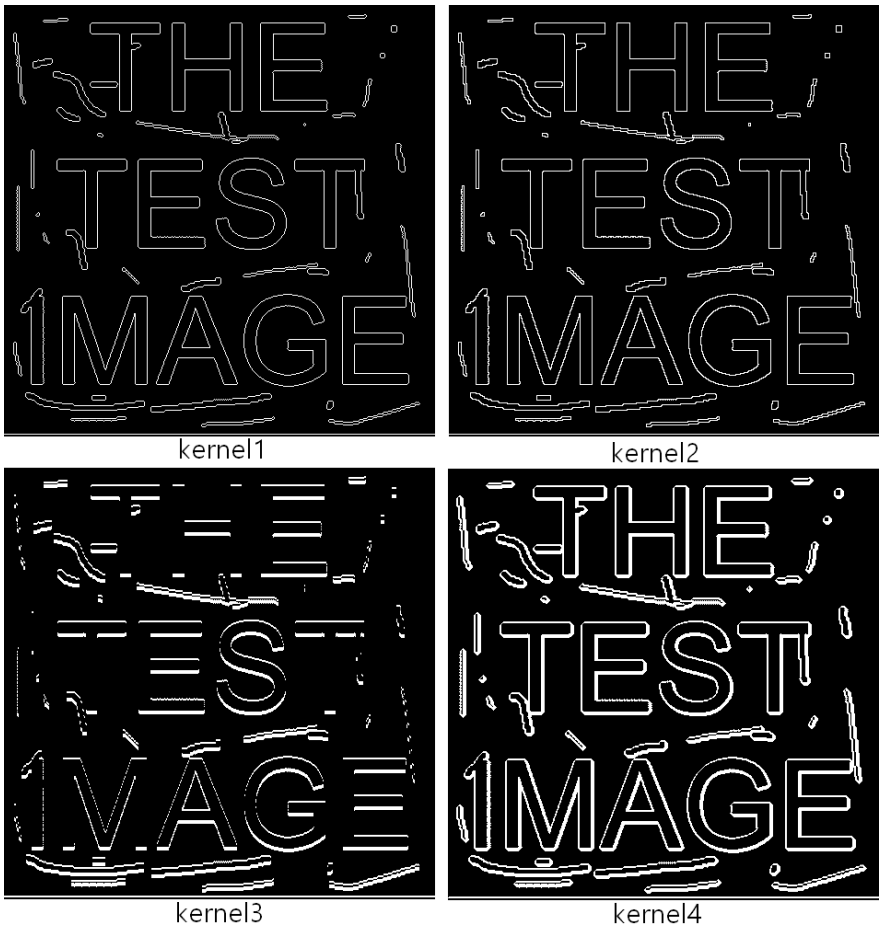
```python
def gradient_laplacian(name, kernel_no):
    image = cv2.imread(name, cv2.IMREAD_GRAYSCALE)
    _, binary = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)

    dilation = dilate(name, kernel_no)
    erosion = erode(name, kernel_no)

    gradient = cv2.subtract(dilation, erosion)

    binary_16 = np.int16(binary)
    laplacian = np.clip(dilation.astype(np.int16) + erosion.astype(np.int16) - 2 *
    binary_16, 0, 255).astype(np.uint8)

    img = Image.fromarray(gradient)
    img.save(name+"_gradient"+str(kernel_no)+".png")

    img2 = Image.fromarray(laplacian)
    img2.save(name+"_laplacian"+str(kernel_no)+".png")
```

Rysunek 21

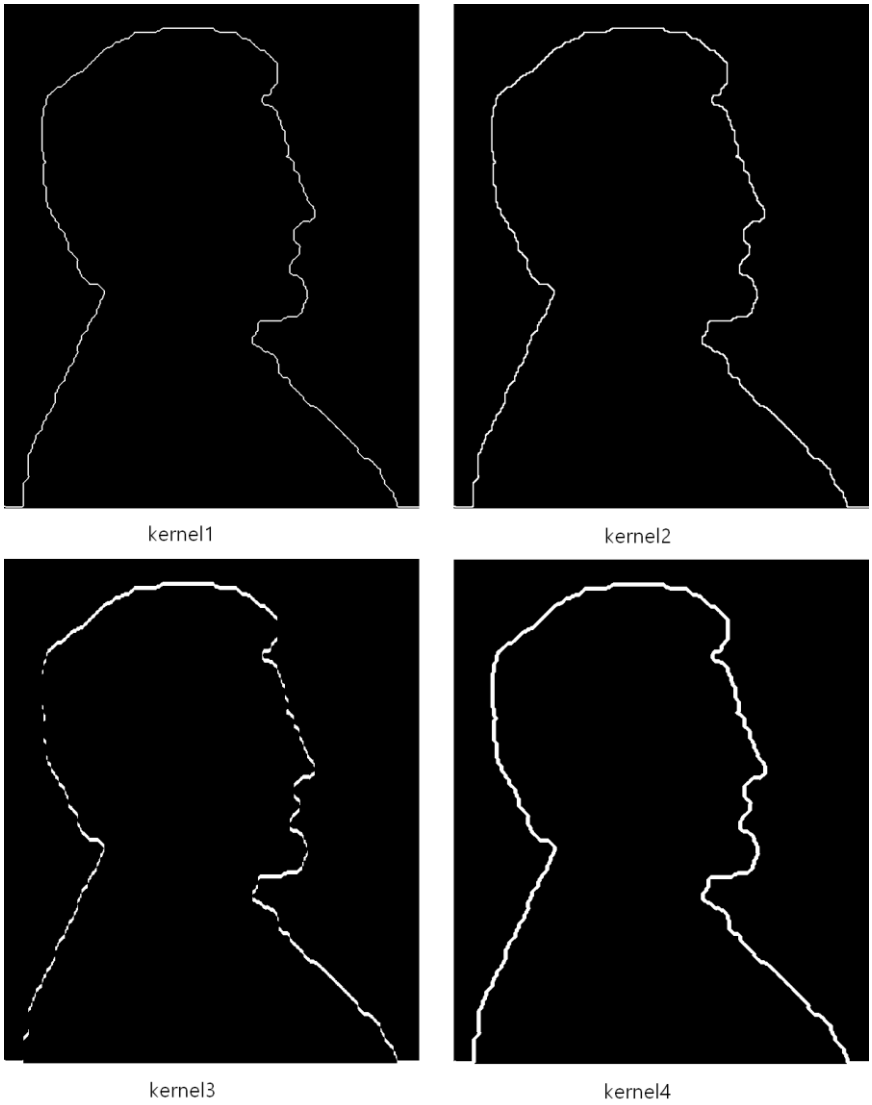
Gradient oraz laplasjan wyznaczono dla obrazów test1.png i test2.png (rysunki 13 i 6) dla każdego z zadanych elementów strukturalnych. Funkcja zad5() wywołuje gradient_laplacian() dla tej konfiguracji operacji. Wybór opcji nr 5 w programie main.exe skutkuje wywołaniem tej funkcji. Wyniki działania programu dane są rysunkami 22-25.



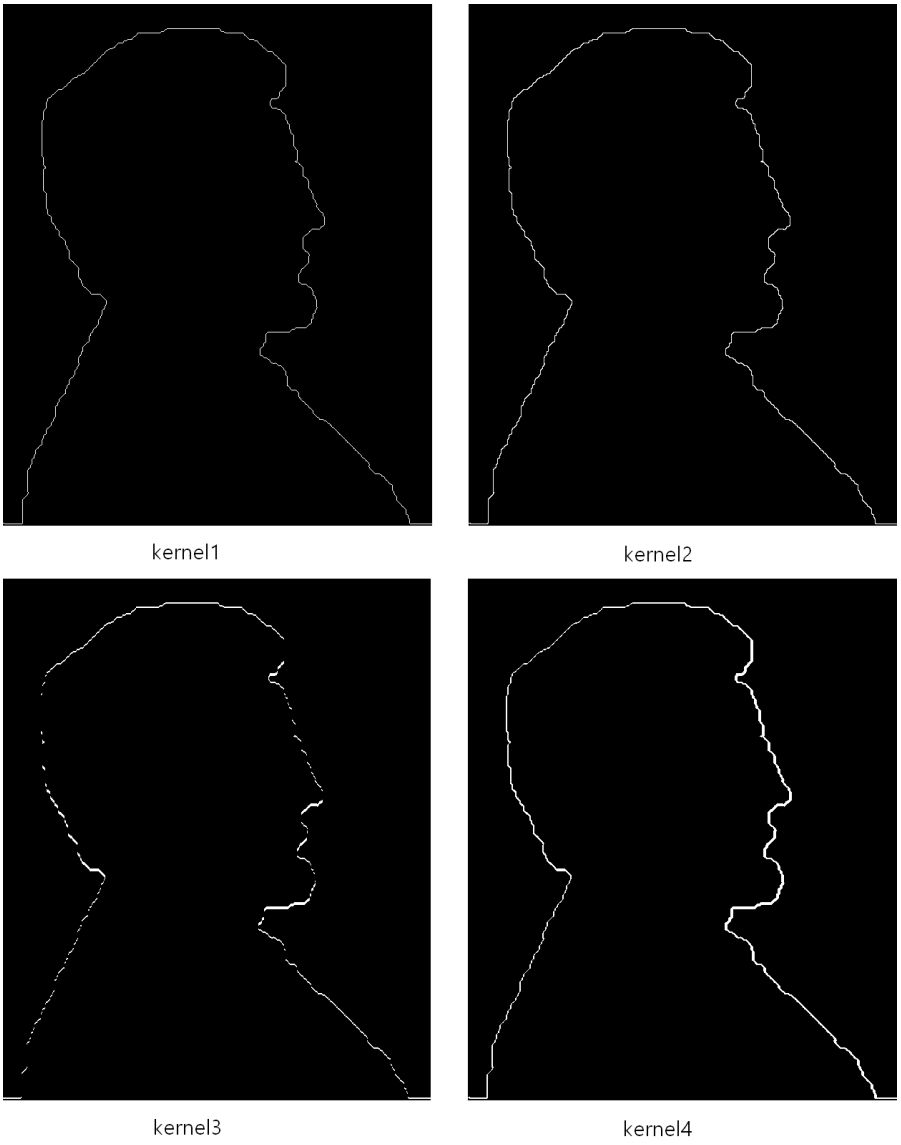
Rysunek 22 Gradient



Rysunek 23 Laplasjan



Rysunek 24 Gradient



Rysunek 25 Laplasjan

Wyniki pozwalają nam na sformułowanie dwóch obserwacji. Po pierwsze, operacje te fantastycznie sprawdzają się jako detektor krawędzi, jako alternatywa dla filtru górnoprzepustowego dla obrazów binarnych. Zwłaszcza, że przy filtrze górnoprzepustowym kontury nie zawsze będą tak wyraźne jak w przypadku powyższych operacji:

„Gradienty morfologiczne zapewniają dokładniejsze wyznaczenie konturów na obrazach binarnych niż standardowe filtry górnoprzepustowe, których krawędzie mogą wydawać się rozmyte lub mniej wyraźne.[6]”

Po drugie, w dalszym ciągu obserwujemy wpływ elementu strukturalnego na kierunek działania. Na rysunkach nr 22 i 23 dla elementu strukturalnego nr 3 zauważamy zniknięcie linii pionowych w literach. Element 1×7 to bardzo “wyostrzony” filtr pod kątem pionowych struktur

[6] Rivest Jean-F., Soille Pierre, Beucher Serge. *Morphological gradients.* Journal of Electronic Imaging. 1993, Vol. 2, nr 4, s. 326-336

o minimalnej wysokości 7 pikseli. Stąd cienkie lub krótsze pionowe kreski zostają usunięte w wyniku erozji.

3. Struktura projektu

Rozwiązania zadań zostały zaimplementowane w pliku lab6.py. Interfejs tekstowy do wykonywania tych zadań został zaimplementowany w pliku main.py. Za pomocą narzędzia PyInstaller utworzono plik wykonywalny main.exe na podstawie pliku main.py. Plik ten znajduje się w katalogu dist wraz z plikami graficznymi niezbędnymi do prawidłowego działania programu.

Źródła

Stańczyk, Urszula. Morfologia matematyczna – pojęcia podstawowe. Studia Informatica, 2000, Vol. 21, nr 3, s. 27–53.

Rivest, Jean-F.; Soille, Pierre; Beucher, Serge. Morphological gradients. Journal of Electronic Imaging, 1993, Vol. 2, nr 4, s. 326–336.

Lê, Duy Huỳnh; Xu, Yongchao; Géraud, Thierry. Morphological Hierarchical Image Decomposition Based on Laplacian 0-Crossings. Lecture Notes in Computer Science, 2017, Vol. 10225, s. 159–171.

Jia, Xiaoli. Minkowski Sum. [online] Iowa State University. https://faculty.sites.iastate.edu/jia/files/inline-files/15.%20Minkowski%20sum.pdf

Rutkowski, Jarosław. Wykład 10 – Morfologia matematyczna. [online] Politechnika Wrocławska. https://kcir.pwr.edu.pl/~jr/cpois/wyklady/wyklad10.pdf