1. Podstawy przetworzeń kontekstualnych

Operacje kontekstualne różnią się od operacji bezkontekstowych (inaczej: punktowych) tym, że wynikowa wartość piksela zależy nie tylko od źródłowej wartości poszczególnych pikseli, ale również od wartości otaczających je pikseli.

Tego typu operacje znajdują zastosowanie wszędzie tam, gdzie istotna jest analiza nie tylko spektralnych, ale także (a może przede wszystkim) przestrzennych cech obrazu, takich jak sąsiedztwo, kształt, tekstura, rozmiar obiektu itp. Do najczęstszych zastosowań operacji kontekstualnych zaliczamy usuwanie (tłumienie) szumu na obrazie oraz wykrywanie wybranych cech obrazu – jego charakterystycznych elementów, np. krawędzi lub narożników obiektów.

1.1. Wprowadzenie

Poniżej prezentujemy wybrane podstawy teoretyczne dotyczące operacji kontekstualnych, pomocne w wykonaniu ćwiczeń związanych z podstawami operacji kontekstualnych

1.1.1. Klasyfikacja operacji kontekstualnych

Istnieje przynajmniej kilka kryteriów klasyfikacji operacji kontekstualnych. Z punktu widzenia potencjalnych zastosowań, najważniejszy jest podział na operacje:

- dolnoprzepustowe (ang. *low-pass*),
- górnoprzepustowe (ang. *high-pass*).

Dolno- i górnoprzepustowość odnosi się do częstotliwości przestrzennej (ang. *spatial frequency*) obrazu. Według definicji PWN, częstotliwość przestrzenna to *liczba cykli jasność-ciemność we wzorcu na danym wymiarze odległości w przestrzeni wzrokowej*. Z punktu widzenia cyfrowych przetworzeń obrazów cyfrowych, możemy ją wyjaśnić jako zróżnicowanie wartości pikseli. Zatem dużą częstotliwość przestrzenną zaobserwujemy na obrazie o dużym zróżnicowaniu wartości pikseli, natomiast małą – na obrazie o małym zróżnicowaniu wartości pikseli. Operacje dolnoprzepustowe obniżają więc częstotliwość przestrzenną obrazu, zmniejszając różnice między wartościami pikseli położonych blisko siebie, tym bardziej, im większe są te różnice. Natomiast operacje górnoprzepustowe zwiększają częstotliwość przestrzenną lub wykrywają te elementy obrazu, które cechują się dużą częstotliwością przestrzenną (np. krawędzie obiektów). Z tego względu – usuwania, czy też odsiewania wybranych elementów obrazu – operacje kontekstualne nazywamy często filtrami. Choć warto pamiętać, że to określenie nie zawsze jest adekwatne.

Inny rodzaj podziału operacji kontekstualnych dotyczy sposobu realizacji przetworzenia – obliczenia wynikowej wartości piksela. Pod tym względem wyróżniamy operacje:

- arytmetyczne (konwolucyjne/splotowe),
- statystyczne (niearytmetyczne)

Operacje arytmetyczne realizują tzw. splot (ang. *convolution*), czy też mnożenie splotowe dwóch funkcji. Jedną z tych funkcji jest obraz (a w zasadzie jego fragment), natomiast drugą – tzw. maska splotu (filtru), czyli podzbiór obrazu, z przypisanymi wartościami wagowymi. Wykonanie operacji odbywa się poprzez przemnożenie odpowiadających sobie wartości – obrazu i maski splotu i zsumowanie tych iloczynów. Należy przy tym pamiętać, że w sytuacji, w której suma współczynników wagowych w masce splotu jest większa od 1 (ma to miejsce zazwyczaj w przypadku filtrów dolnoprzepustowych), wynik mnożenia splotowego należy jeszcze znormalizować – podzielić przez sumę współczynników wagowych.

Z kolei operacje statystyczne nie mają przypisanych wartości do poszczególnych elementów maski, jak w przypadku filtrów arytmetycznych. Obliczenie wynikowej wartości piksela odbywa się poprzez sortowanie wartości pikseli objętych maską filtru (np. rosnąco) i wybraniu na tej podstawie jednej wartości, np. środkowej, minimalnej, maksymalnej, najczęściej występującej itd.

Jeszcze inne kryterium podziału operacji kontekstualnych opiera się na ich strukturze. Pod tym kątem możemy wyróżnić operacje:

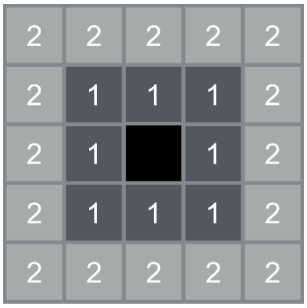
- proste,
- złożone.

Operacje proste to operacje polegające na pojedynczym przetworzeniu obrazu za pomocą odpowiedniej maski filtru. Z kolei operacje złożone mogą składać się z większej liczby przetworzeń różnymi (lub tymi samymi) maskami filtru. Mogą również składać się z różnego rodzaju (pojedynczych – prostych) operacji kontekstualnych, a także wykorzystywać innego rodzaju przetworzenia, np. arytmetyczne (odejmowanie, dodawanie, mnożenie, potęgowanie itd.).

1.1.2. Rodzaje masek filtrów

Maski filtru mogą przyjmować w zasadzie dowolny kształt i rozmiar (uwzględniając rastrową naturę obrazu cyfrowego). Jednak pomijając specyficzne zastosowania wymuszające *nietypowy* kształt maski, najczęściej zależy nam na takim kształcie, który nie będzie wyróżniał żadnego z kierunków na obrazie. W przestrzeni euklidesowej byłoby to koło. W przestrzeni (dyskretnej) obrazu rastrowego uzyskanie takiego kształtu (a w zasadzie jego odpowiednika) jest możliwe przy spełnieniu pewnych warunków. Warunki te wiążą się z systemami określania odległości między pikselami. Możemy wyróżnić 3 takie systemy.

Odległość typu *szachownica* (ang. *chessboard*) zakłada, że każdy piksel sąsiaduje z 8 położonymi w jego otoczeniu pikselami – na równych warunkach (czasem określamy to jako *8-sąsiedztwo*). Odległość kolejnych *warstw* pikseli w tym systemie ilustruje poniższy rysunek. Jeśli więc koło oznacza *zbiór wszystkich punktów płaszczyzny, których odległość od ustalonego punktu na tej płaszczyźnie, nazywanego środkiem koła, jest mniejsza lub równa długości promienia koła*, to w systemie odległości *szachownica* koło przyjmuje kształt kwadratu. Oczywiście, ten kształt będziemy określać jako *kwadrat* (ang. *square*). Rozmiar maski określimy natomiast długością *promienia* tego *pseudokoła*. Zatem na poniższej ilustracji widzimy 2 kwadraty: o rozmiarze 1 i 2. Warto jednak wiedzieć, że w różnych programach możemy się spotkać z innym określeniem rozmiaru, np. 3x3 piksele (rozmiar 1), 5x5 pikseli (rozmiar 2) itd.



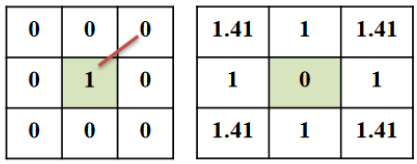
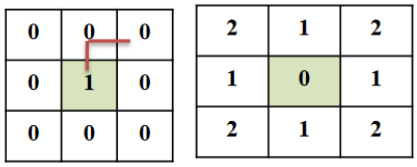
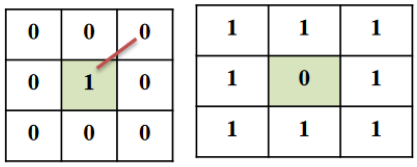
Odległość typu *miejskiego* (ang. *city-block, manhattan*) oznacza, że każdy piksel sąsiaduje z 4 pikselami (*4-sąsiedztwo*) położonymi po bokach oraz u góry i u dołu (wyłączone są piksele sąsiadujące narożnikami). Pokazuje to poniższa ilustracja. Kształt, który uzyskujemy w ten sposób nazywamy diamentem (ang. *diamond*). Widzimy tu *diamenty* o rozmiarze 1 i 2 (oraz fragmenty diamentów o rozmiarze 3 i 4).

4	3	2	3	4
3	2	1	2	3
2	1		1	2
3	2	1	2	3
4	3	2	3	4

Odległość pseudo-euklidesowa (ang. pseudo-euclidean) polega na obliczeniu odległości euklidesowej pomiędzy pikselami (ich środkami), jak na poniższej ilustracji a następnie zaokrągleniu do wartości całkowitych.

2,8	2,2	2	2,2	2,8
2,2	1,4	1	1,4	2,2
2	1		1	2
2,2	1,4	1	1,4	2,2
2,8	2,2	2	2,2	2,8

Kolejna ilustracja (poniżej) systematyzuje wiedzę na temat różnicy pomiędzy rodzajami masek filtrów.

Distance Metric	Description	Illustration
Euclidean P(x , y), q(s , t)	The Euclidean distance is the straight-line distance between two pixels. $D_e(p,q) = \sqrt{(x-s)^2 + (y-t)^2}$	
City Block P(x , y), q(s , t)	The city block distance metric measures the path between the pixels based on a 4-connected neighborhood. Pixels whose edges touch are 1 unit apart and pixels diagonally touching are 2 units apart. $D_4(p,q) = \|x-s\| + \|y-t\|$	
Chessboard P(x , y), q(s , t)	The chessboard distance metric measures the path between the pixels based on an 8-connected neighborhood. Pixels whose edges or corners touch are 1 unit apart. $D_8(p,q) = max(\|x-s\|, \|y-t\|)$	

źródło: https://uomustansiriyah.edu.iq/media/lectures/6/6_2020_05_20!01_45_22_AM.pdf

1.2. Operacje dolnoprzepustowe (low-pass filters (LPF)) i górnoprzepustowe (high-pass filters (HPF))

Filtry rozmywające, uśredniające mają właściwości wygładzające i redukujące szum, filtry krawędziowe i wyostrzające wydobywają z obrazu niedostrzegalne szczegóły a dzięki filtrom nieliniowym możliwa staje się rekonstrukcja częściowo uszkodzonego obrazu. Właściwości danego filtru a co za tym idzie efekt filtracji zależy od definicji maski, czyli jej rozmiaru oraz wartości poszczególnych jej elementów składowych. O ile wielkość maski decyduje o sile danego filtru to wartości maski decydują o klasie filtru.

Ogólnie można podzielić klasy filtru na rozmywające (dolnoprzepustowe) oraz wyostrzające (górnoprzepustowe).

Biblioteka OpecnCV zawiera funkcję *cv.filter2D()* przeznaczoną dla operacji konwolucją z filtrem (kernelem). Przykładowa operacja konwolucji z filtrem (kernelem) 5x5 wygląda następująco:

$$K = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

W wyniku tego przetworzenia nowa wartość piskela obliczana jest na podstawie średniej arytmetycznej.

Funkcja (1) przedstawia przykładowe wykorzystanie tej funkcji:

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt

#read image
image = cv2.imread('moje_zdjecie.png')

kernel = np.ones((5,5),np.float32)/25
dst = cv2.filter2D(img,-1,kernel)

plt.subplot(121),plt.imshow(img),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(dst),plt.title('Averaging')
plt.xticks([]), plt.yticks([])
plt.show()
```

(1)

1.2.1. Filtry dolnoprzepustowe (low-pass filters (LPF)

Filtry rozmywające to filtry dolnoprzepustowe uśredniające wartości otoczenia znajdującego się w zasięgu maski filtru. Ideą takiego filtru jest zastąpienie wartości każdego punktu w obrazie przez średnią intensywność jego sąsiedztwa zdefiniowanego przez maskę filtru. Filtr taki tłumi w sygnale składowe widma o wysokiej częstotliwości a pozostawia bez zmian niskie częstotliwości. Najbardziej oczywistym zastosowaniem takiego filtru jest redukcja przypadkowego szumu i wygładzenie obrazu. Jednakże, krawędzie, które są prawie zawsze pożądaną informacją w obrazie również mają ostre przejścia jasności, zatem jako obszar o wysokiej częstotliwości ulegną tłumieniu przez filtr, który spowoduje ich rozmycie. Jest to niewątpliwie niepożądany efekt działania filtru i zarazem jego największa wada.

Filtr uśredniający:

Najprostszy filtr rozmywający o rozmiarze 3 × 3 i współczynnikach równych 1 i wadze filtru równej 9 w wyniku działania zastępuje jasność punktu aktualnie przetwarzanego średnią arytmetyczną ze swojego 9-elementowego otoczenia. Tego typu filtr jedynkowy nazywa się filtrem uśredniającym lub pudełkowym (ang. box filter).

Maskę dla filtru uśredniającego dla rozmiaru 1 można przedstawić w następujący sposób:

$$K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Funkcja (2) przedstawia przykład operacji dla kernela rozmiaru 5x5:

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

#read image
image = cv2.imread('moje_zdjecie.png')

blur = cv2.blur(img,(5,5))

plt.subplot(121),plt.imshow(img),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(blur),plt.title('Blurred')
plt.xticks([]), plt.yticks([])
plt.show()
```

(2)

Filtr Gaussa:

Jeszcze lepszy efekt można osiągnąć aproksymując wartości maski filtru funkcją rozkładu Gaussa w dwuwymiarowej formie:

$$h(x, y) = e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

gdzie σ jest odchyleniem standardowym, x i y są wartościami całkowitymi, a współrzędna (0, 0) jest w środku filtru. Jego maskę można zbudować według poniższego schematu:

b^0	b^1	b^0
b^1	b^2	b^1
b^0	b^1	b^0

W zależności od wartości *b*, rzeczywiste współczynniki wagowe maski splotu będą różne. W przypadku, gdy *b=2*, maska będzie wyglądała następująco:

1	2	1
2	4	2
1	2	1

W przypadku masek o większych rozmiarach, w centralnym punkcie maski wartość *b* podnoszona jest do wyższej potęgi (3, przy rozmiarze 2, 4 przy rozmiarze 3 itd.)

Funkcja (3) przedstawia przykład wykorzystania filtracji Gaussa.

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

#read image
image = cv2.imread('moje_zdjecie.png')

blur = cv.GaussianBlur(img,(5,5),0)

plt.subplot(121),plt.imshow(img),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(blur),plt.title('Blurred')
plt.xticks([]), plt.yticks([])
```

(3)

```
plt.show()
```

Filtr medianowy

Jednym z najbardziej popularnych filtrów statystycznych jest filtr medianowy. Obliczenie wartości wynikowej przetworzenia odbywa się poprzez uszeregowanie wartości pikseli objętych maską filtru w porządku rosnącym, a następnie wybranie wartości środkowej (mediany).

Funkcja (4) przedstawia przykład wykorzystania filtracji medianowej.

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

#read image
image = cv2.imread('moje_zdjecie.png')

median = cv.medianBlur(img,5)

plt.subplot(121),plt.imshow(img),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(blur),plt.title('Blurred')
plt.xticks([]), plt.yticks([])
plt.show()
```

(4)

Filtr minimalny i maksymalny

Kolejnymi przykładami statystycznych filtrów dolnoprzepustowych są **filtr minimalny i maksymalny**. Są one dualne względem siebie. Jak łatwo się domyślić, wynik filtru minimalnego stanowi najmniejsza wartość spośród pikseli objętych maską, natomiast wynik filtru maksymalnego – największa wartość.

Filtr minimalny jest często nazywany filtrem kompresującym albo erozyjnym ponieważ efektem jego działania jest zmniejszenie globalnej jasności obrazu, jasne obiekty ulegną zmniejszeniu a powiększą się obiekty ciemne.

Filtr maksymalny bywa nazywany filtrem dekompresującym albo ekspansywnym gdyż w wyniku filtracji zwiększa globalnie jasność obrazu i analogicznie do filtru minimalnego tu powiększone zostaną obiekty jasne a zmniejszone ciemne.

Praca własna: utwórz funkcję dla filtru minimalnego i maksymalnego.

1.2.2. Filtry górnoprzepustowe (high-pass filters (HPF))

Filtry wyostrzające znacznie różnią się od poprzednich trzech rodzajów operacji. Celem filtrów wyostrzających jest bowiem wyostrzenie – pozorne – obrazu, podczas gdy pozostałe operacje mają za zadanie wskazanie miejsc o dużych częstotliwościach przestrzennych – krawędzi i innych istotnych szczegółów obrazu. Kolejność przedstawiania poszczególnych operacji oparta zostanie na logice wynikającej właśnie z powyższego podziału.

Inna ważna uwaga dotyczy podziału na operacje arytmetyczne oraz statystyczne. Większość „klasycznych” operacji górnoprzepustowych to operacje arytmetyczne. Z tego względu w pierwszej kolejności zostaną przedstawione przykłady operacji określonego typu realizujące mnożenie splotowe, natomiast w drugiej części, osobno, przedstawione zostaną przykłady operacji górnoprzepustowych statystycznych.

Istnieje kilka rodzajów operacji górnoprzepustowych, w zależności od sposobu realizacji operacji lub od jej celu. Możemy wyróżnić m.in.:

- operatory kierunkowe;
- filtry gradientowe – wykrywające krawędzie;

- laplasjany;
- filtry wyostrzające.

1.2.2.1. Operatory kierunkowe

Operatory kierunkowe nazywamy tak dlatego, że wykrywają krawędzie obiektów z różnym efektem, w zależności od orientacji krawędzi.

Operator Robertsa

Jednym z najprostszych operatorów jest operator *Robertsa*. Maska tego filtru może wyglądać następująco:

0	0	0
0	1	-1
0	0	0

Fakt, że współczynniki wagowe sumują się do *0* powoduje, że miejsca o zerowej częstotliwości przestrzennej będą przyjmowały wartości *0*. Z kolei miejsca o dużym zróżnicowaniu wartości pikseli, będą się charakteryzowały wynikami o dużych wartościach bezwzględnych.

Zwróćmy uwagę na fakt, że elementy maski splotu o wartościach *0* nie mają tak naprawdę znaczenia. Dlatego możemy je pominąć przy definiowaniu maski.

Zwróćmy też uwagę na fakt, że wraz z pojawieniem się współczynników wagowych o ujemnych wartościach, mogą się pojawić również ujemne wynikowe wartości pikseli. Takie wartości pikseli można, oczywiście, zapisać w obrazie cyfrowym (przy wyborze odpowiedniego typu pliku). Co innego jednak z wyświetleniem ich w trybie 8-bitowym – w tym przypadku rozwiązanie tego zagadnienia nie jest jednoznaczne.

Jednym ze sposobów „radzenia" sobie z tym zagadnieniem jest normalizacja, czyli rozciągnięcie histogramu (najczęściej liniowe) – od najmniejszej. W wyniku normalizacji, wartościom ujemnym zostaną przyporządkowane ciemne tony szarości, a wartościom dodatnim – jasne. Z kolei wartość *0* będzie przyjmować tony średnioszare.

Z kolei druga metoda – modułowa – polega na obliczeniu wartości bezwzględnych wszystkich pikseli i wyświetleniu ich zgodnie z nimi. W ten sposób, piksele o dużych wartościach bezwzględnych zostaną wyświetlone w jasnych tonach szarości, niezależnie od tego, czy wynikowa wartość piksela jest ujemna czy dodatnia. Biorąc pod uwagę, że filtry górnoprzepustowe mają wskazywać miejsca o dużej częstotliwości przestrzennej, korzystniejsza wydaje się metoda modułowa.

Funkcja (6) przedstawia przykład wykorzystania operatora Robertsa.

```
import cv2 as cv
import numpy as np
from scipy import ndimage


#read image
image = cv2.imread('moje_zdjecie.png', 0).astype('float64')

img/=255.0
vertical = ndimage.convolve( img, roberts_cross_v )
horizontal = ndimage.convolve( img, roberts_cross_h )

edged_img = np.sqrt( np.square(horizontal) + np.square(vertical))
edged_img=(np.absolute(edged_img))


cv2.imshow('Roberts', edged_img)
```

(6)

```
print(cv2.waitKey(0))
cv2.destroyAllWindows()
for i in range(2):
    cv2.waitKey(1)
```

Operator Prewitt

Operator *Robertsa* jest tylko jednym z przykładów tego typu operacji. Innym przykładem jest operator *Prewitta*. Operator $Prewitt_E$ możemy przedstawić w następujący sposób:

1	0	-1
1	0	-1
1	0	-1

Funkcja (7) przedstawia zastosowanie operatora Prewitta

```
import cv2 as cv
import numpy as np

#read image
image = cv2.imread('moje_zdjecie.png', 0)
image_gaussian = cv2.GaussianBlur(image,(3,3),0)

kernelx = np.array([[1,1,1],[0,0,0],[-1,-1,-1]])
kernely = np.array([[-1,0,1],[-1,0,1],[-1,0,1]])
img_prewittx = cv2.filter2D(img_gaussian, -1, kernelx)
img_prewitty = cv2.filter2D(img_gaussian, -1, kernely)

cv2.imshow("Prewitt X", img_prewittx)
cv2.imshow("Prewitt Y", img_prewitty)
cv2.imshow("Prewitt", img_prewittx + img_prewitty)

print(cv2.waitKey(0))
cv2.destroyAllWindows()
for i in range(2):
    cv2.waitKey(1)
```

(7)

1.2.2.2. Filtry gradientowe – wykrywające krawędzie

Najpopularniejszym filtrem tego typu jest filtr Sobela. Filtr Sobela jest operacją złożoną. Polega na wykonaniu – niezależnie – przetworzenia obrazu źródłowego za pomocą dwóch operatorów Sobela: $Sobel_S$ i $Sobel_E$. Takich, jak poniżej:

1	2	1
0	0	0
-1	-2	-1

1	0	-1
2	0	-2
1	0	-1

Następnie dwa obrazy wynikowe (nazwijmy je, odpowiednio S_S oraz S_E) są przetwarzane według poniżej formuły.

$$S = \sqrt{S_S^2 + S_E^2}$$

Zasada działania filtru opiera się na połączeniu efektów działania obydwu operatorów, z których każdy wykrywa krawędzie o innych orientacjach – z innym natężeniem.

Funkcja (9) przedstawia zastosowanie filtra Sobel

```
import cv2 as cv
import numpy as np

#read image
image = cv2.imread('moje_zdjecie.png', 0)
```

(9)

```
image_gaussian = cv2.GaussianBlur(image,(3,3),0)

img_sobelx = cv2.Sobel(img_gaussian,cv2.CV_8U,1,0,ksize=5)
img_sobely = cv2.Sobel(img_gaussian,cv2.CV_8U,0,1,ksize=5)
img_sobel = img_sobelx + img_sobely

cv2.imshow("Sobel X", img_sobelx)
cv2.imshow("Sobel Y", img_sobely)
cv2.imshow("Sobel", img_sobel)

print(cv2.waitKey(0))
cv2.destroyAllWindows()
for i in range(2):
    cv2.waitKey(1)
```

1.2.2.3. Filtry Laplace'a

Przeznaczeniem filtrów Laplace'a, czyli laplasjanów jest wykrywanie krawędzi – niezależnie od kierunku. Laplasjany z maską o rozmiarze *1* można przedstawić w jeden z dwóch sposobów:

0	-1	0
-1	4	-1
-1	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

Jak widać, suma współczynników wagowych jest równa *0*, jednak ich rozkład jest inna niż w przypadku operatorów kierunkowych. Na czym polega ta różnica? Z czego wynika to, że wartość na krawędzie będzie dodatnia lub ujemna – w przypadku operatorów kierunkowych i laplasjanów? (odpowiednie wnioski przydadzą nam się przy analizie działania filtrów wyostrzających).

Funkcja (10) przedstawia zastosowanie filtra Laplace

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

#read image
image = cv2.imread('moje_zdjecie.png', 0)
image_gaussian = cv2.GaussianBlur(image,(3,3),0)

laplacian = cv2.Laplacian(img,cv2.CV_64F)

plt.subplot(2,2,1),plt.imshow(img,cmap = 'gray')
plt.title('Original'), plt.xticks([]), plt.yticks([])
plt.subplot(2,2,2),plt.imshow(laplacian,cmap = 'gray')
plt.title('Laplacian'), plt.xticks([]), plt.yticks([])

plt.show()
```

(10)