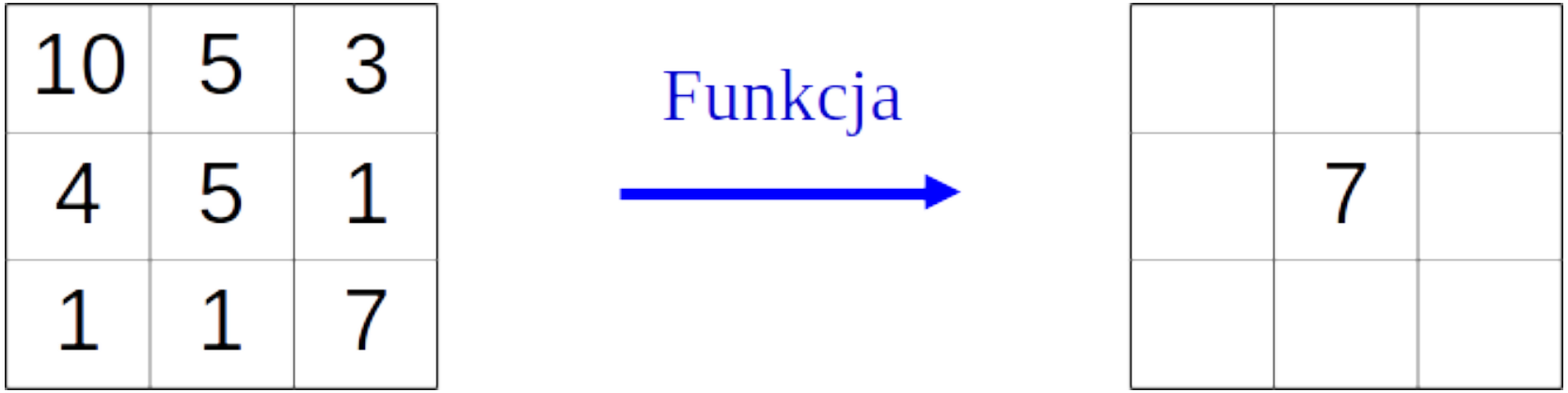
Wizja Maszynowa

Przetwarzanie obrazu - filtrowanie wykład 3

Adam Szmigielski
aszmigie@pjwstk.edu.pl
materials: $ftp(public) : //aszmigie/WM$

Filtr obrazu

Modyfikuje piksele obrazu w oparciu o piksele otoczenia



Filtracja liniowa

10	5	3
4	5	1
1	1	7

$\otimes$

0	0	0
0	0.5	0
0	1.0	0.5

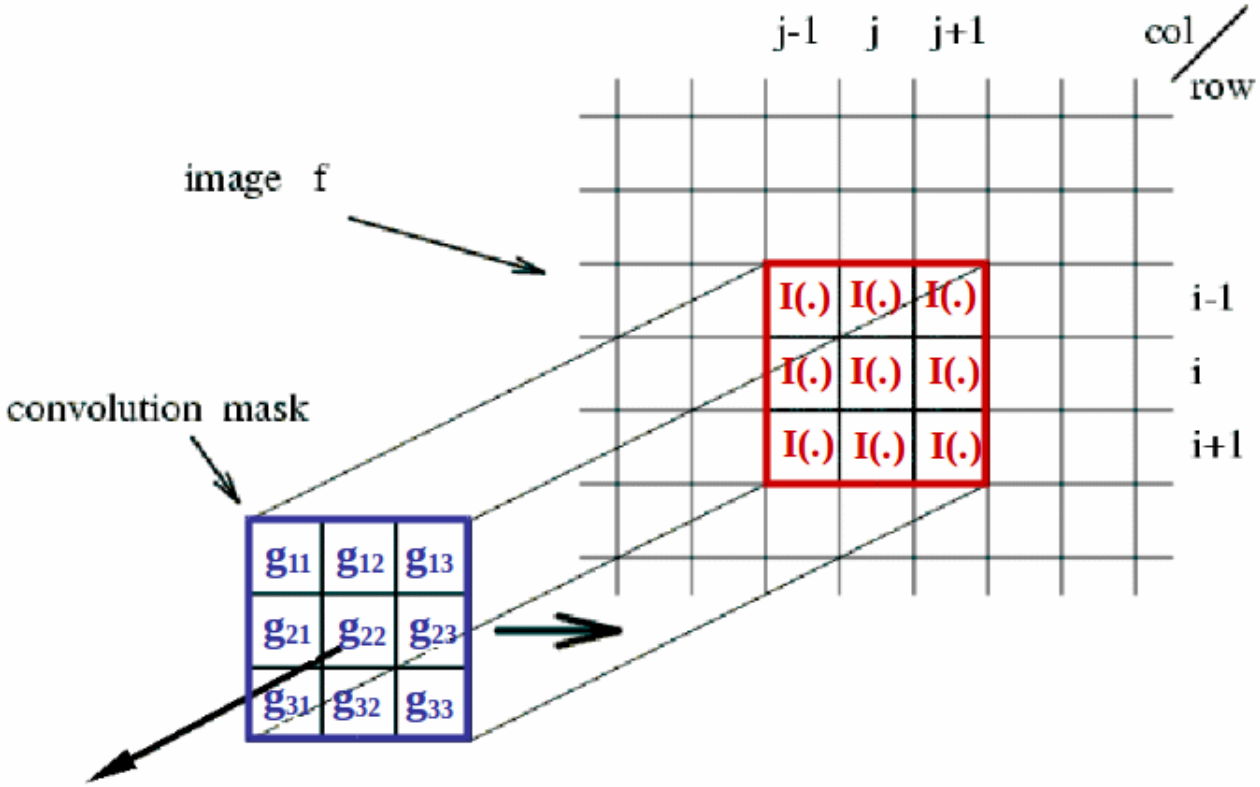
Jądro splotu

$=$

	7	

- liniowa jest najprostsza i najbardziej użyteczna,
- Zastępuje każdy piksel liniową kombinacją sąsiadów,
- Funkcja (sposób) dla kombinacji liniowej nazywana jest *jądrem splotu*.

Filtr liniowy - splot



$$f(i,j) = g_{11}\, I(i-1,j-1) + g_{12}\, I(i-1,j) + g_{13}\, I(i-1,j+1) + g_{21}\, I(i,j-1) + g_{22}\, I(i,j) + g_{23}\, I(i,j+1) + g_{31}\, I(i+1,j-1) + g_{32}\, I(i+1,j) + g_{33}\, I(i+1,j+1)$$

Operator sąsiedztwa jako filtr liniowy

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

$f(x,y)$

*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

$h(x,y)$

=

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

$g(x,y)$

- Obraz po lewej stronie jest spleciony z filtrem pośrodku, aby uzyskać obraz po prawej stronie.
- Jasnoniebieskie piksele wskazują sąsiedztwo źródłowe jasnozielonego piksela docelowego.
- Wartość piksela wyjściowego jest określana jako ważona suma wartości

pikseli wejściowych w małym sąsiedztwie N:

$$g(i,j) = \sum_{k,l} f(i+k, j+l)h(k,l).$$

- Wartości wag *jądro* lub *maska* $h(k,l)$ są często nazywane *współczynnikami filtru*. Powyższy **operator korelacji** można zapisać jako

  $$g = f \otimes h.$$

  Typowym wariantem tej formuły jest:

  $$g(i,j) = \sum_{k,l} f(i-k, j-l)h(k,l) = \sum_{k,l} f(k,l)h(i-k, j-l),$$

  gdzie znak przesunięć w f zostało odwrócone, nazywa się to **operatorem splotu**,

  $$g = f * h$$

  i h nazywana jest wtedy *funkcją odpowiedzi impulsowej* - ciągłą wersję splotu można zapisać jako: $g(x) = \int f(x-u)h(u)du$.

Korelacja i splot jako mnożenie macierz-wektor

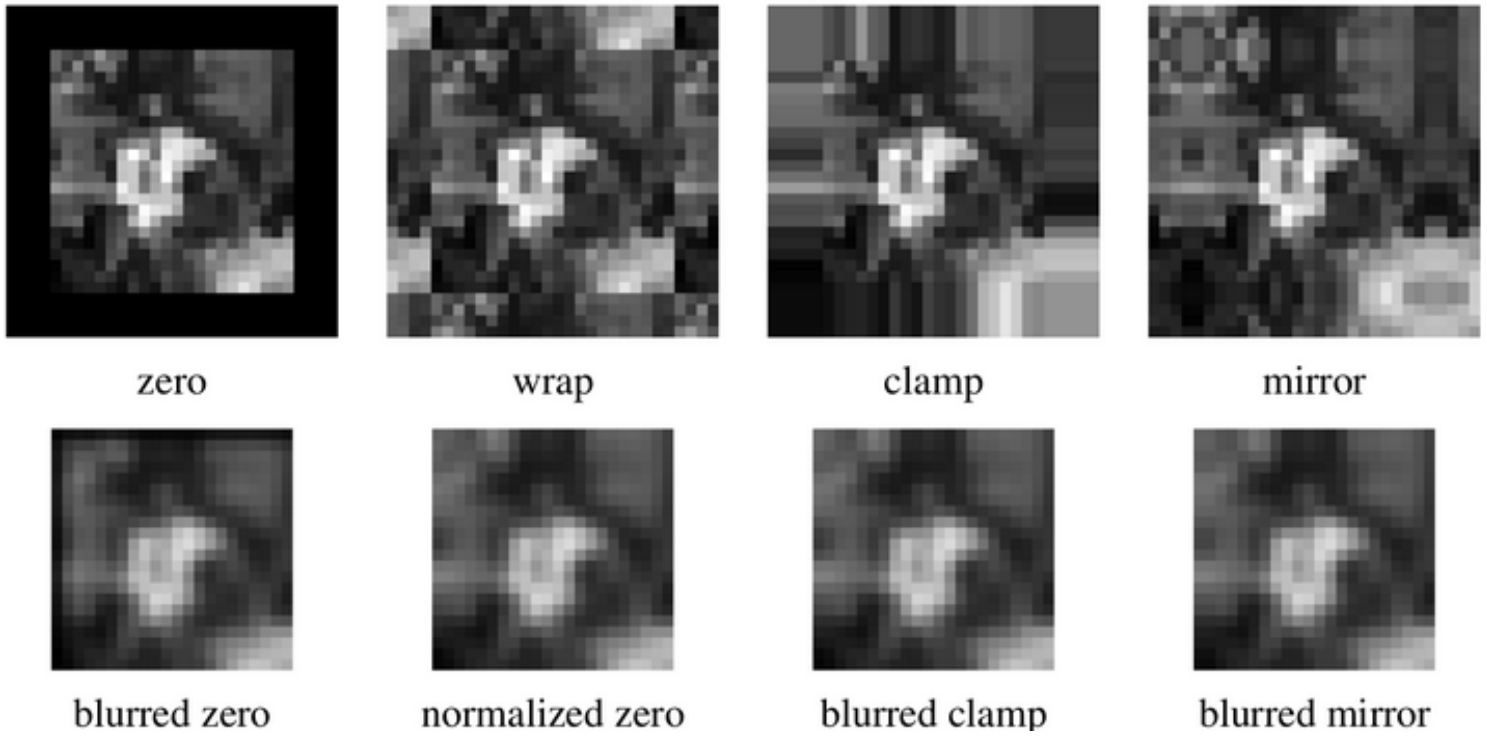
$$\boxed{72}\boxed{88}\boxed{62}\boxed{52}\boxed{37} * \boxed{1/4}\boxed{1/2}\boxed{1/4} \quad \Leftrightarrow \quad \frac{1}{4}\begin{bmatrix} 2 & 1 & . & . & . \\ 1 & 2 & 1 & . & . \\ . & 1 & 2 & 1 & . \\ . & . & 1 & 2 & 1 \\ . & . & . & 1 & 2 \end{bmatrix} \begin{bmatrix} 72 \\ 88 \\ 62 \\ 52 \\ 37 \end{bmatrix}$$

- Jednowymiarowy splot można przedstawić w postaci macierzowo-wektorowej.
- *Correlation* i *convolution* można zapisać jako mnożenie macierzy przez wektor, jeśli najpierw przekonwertujemy dwuwymiarowe obrazy $f(i, j)$ i $g(i, j)$ na wektory uporządkowane w rastrze f i g:

$$g = Hf,$$

gdzie (rzadka) macierz H zawiera jądra splotu.

Dopełnienie (efekty obramowania)



- Wynik korelacji jest mniejszy niż oryginalny obraz, co może nie być pożądane w wielu aplikacjach.
- Dzieje się tak, ponieważ sąsiedztwa typowych operacji korelacji i splotów wykraczają poza granice obrazu

- Opracowano szereg różnych trybów dopełniania lub rozszerzania dla operacji w sąsiedztwie:
  - *zero*: ustaw wszystkie piksele poza obrazem źródłowym na 0 (dobry wybór dla wycinania zdjęcia);
  - *stały (kolor obramowania)*: ustaw wszystkie piksele poza obrazem źródłowym na określone obramowanie wartość;
  - *zacisk (replikacja lub zaciskanie do krawędzi)*: powtarzanie pikseli krawędzi w nieskończoność;
  - *(cykliczne) zawijanie (powtarzanie lub kafelek)*: zapętlenie “ wokół ” obrazu w konfiguracji “ toroidalnej ”;
  - *lustro*: odbija piksele wzdłuż krawędzi obrazu;
  - *rozbudowa*: rozszerza sygnał przez odjęcie lustrzanej wersji sygnału od wartości piksela krawędzi.

Oddzielne filtrowanie

$\frac{1}{K^2}$
$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & 1 & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

$\frac{1}{16}$
$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$\frac{1}{256}$
$$\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

$\frac{1}{8}$
$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$\frac{1}{4}$
$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

$\frac{1}{K}$ $\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}$ $\frac{1}{4}$ $\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$ $\frac{1}{16}$ $\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix}$ $\frac{1}{2}$ $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ $\frac{1}{2}$ $\begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$

(a) box, $K = 5$ (b) bilinear (c) "Gaussian" (d) Sobel (e) corner

- Proces wykonywania splotu wymaga operacji K^2 (mnożenie i dodawanie) na piksel, gdzie K jest rozmiarem
- W wielu przypadkach operację tę można znacznie przyspieszyć, wykonując najpierw jednowymiarowy splot poziomy, a następnie jednowymiarowy splot pionowy, co wymaga łącznie $2K$ operacji na piksel.

- Jądro splotu, dla którego jest to możliwe, jest nazywane rozłącznym.
- Dwuwymiarowe jądro K odpowiadające kolejnemu splotowi z poziomym jądrem h i pionowym jądrem v jest iloczynem zewnętrznym dwóch jąder,

$$K = vh^T$$

- Na projektowanie jąder splotu do zastosowań komputerowych często wpływa ich rozdzielność.
- Często można to zrobić poprzez inspekcję lub przyjrzenie się analitycznej postaci jądra
- Bardziej bezpośrednią metodą jest potraktowanie jądra jako macierzy K i przyjęcie jej rozkładu na wartości osobliwe (SVD),

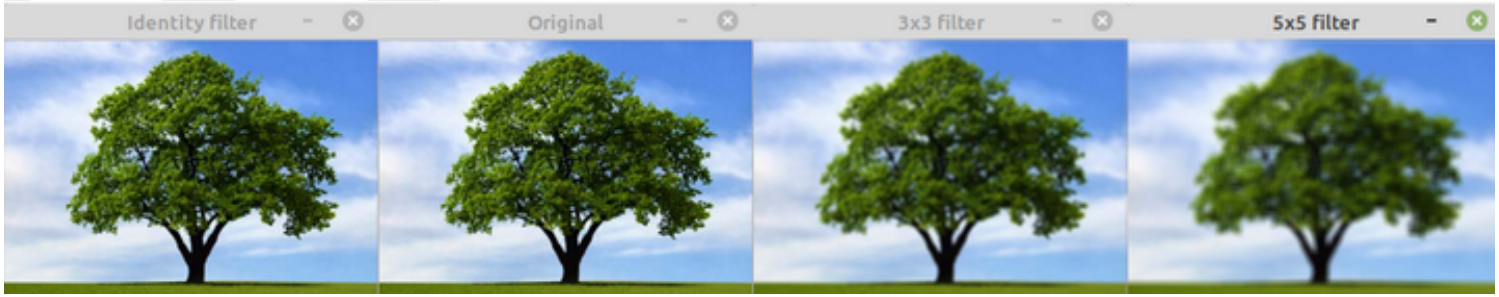
Filtr medianowy

- Filtr mediany wybiera wartość mediany z otoczenia każdego piksela,
- Wartości mediany można obliczyć w oczekiwanym czasie liniowym,
- Ponieważ wartość szumu zwykle leży daleko poza prawdziwymi wartościami w sąsiedztwie, filtr mediany jest w stanie odfiltrować takie złe piksele.
- Inną możliwością jest obliczenie ważonej mediany - jest to równoważne z minimalizacją ważonej funkcji celu:

$$\sum_{k,l} w(k,l)|f(i+k,j+l) - g(i,j)|^p,$$

gdzie $g(i,j)$ jest pożądaną wartością wyjściową, a $p = 1$ dla ważonej mediany. Wartość $p = 2$ jest zwykłą średnią ważoną, która jest równoważna korelacji.

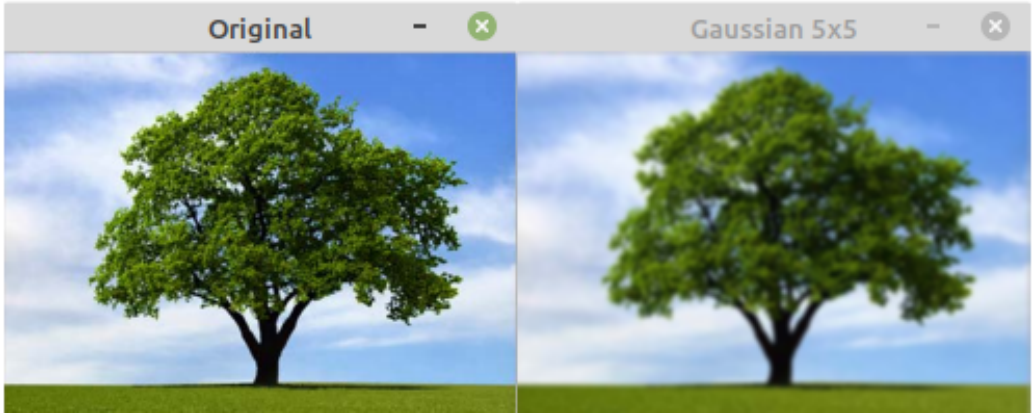
Konwolucja 2D (filtrowanie obrazu)- Python




```
import cv2
import numpy as np

img = cv2.imread('images/tree_input.png', cv2.IMREAD_COLOR)

kernel_identity = np.array([[0,0,0], [0,1,0], [0,0,0]])
kernel_3x3 = np.ones((3,3), np.float32) / 9.0
kernel_5x5 = np.ones((5,5), np.float32) / 25.0

cv2.imshow('Original', img)
output = cv2.filter2D(img, -1, kernel_identity)
cv2.imshow('Identity filter', output)
output = cv2.filter2D(img, -1, kernel_3x3)
cv2.imshow('3x3 filter', output)
output = cv2.filter2D(img, -1, kernel_5x5)
cv2.imshow('5x5 filter', output)
cv2.waitKey(0)
```

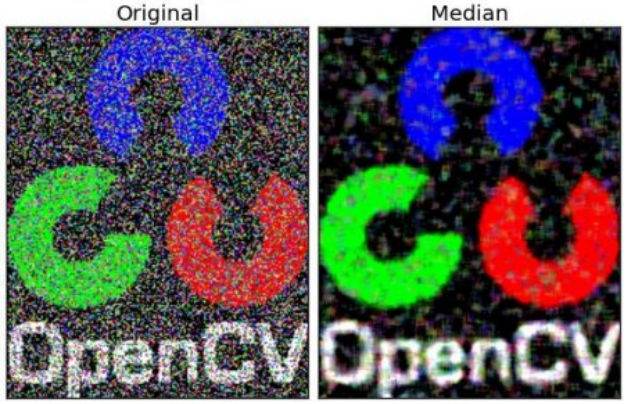
Rozmycie obrazu (wygładzanie obrazu)

- **Uśrednianie -** Odbywa się to poprzez splatanie obrazu ze znormalizowanym filtrem pudełkowym, np $K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
- **Rozmycie gaussowskie -** zamiast filtru pudełkowego używane jest jądro Gaussa.



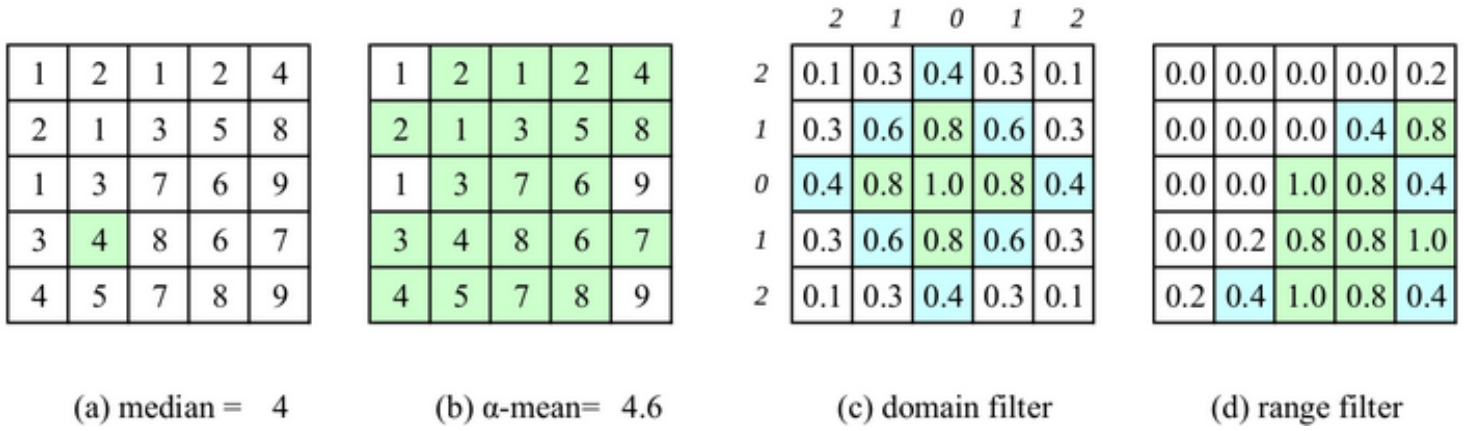

```
blur = cv2.GaussianBlur(img,(5,5),0)
```

- **Median Blurring -** pobiera medianę wszystkich pikseli pod obszarem jądra, a element centralny jest zastępowany tą medianą. Jest to bardzo skuteczne w przypadku szumu “soli i pieprzu” na obrazie.




```
median = cv2.medianBlur(img,5)
```

Filtrowanie nieliniowe



- W filtrze liniowym piksel wyjściowy jest ważoną sumą pewnej liczby pikseli wejściowych.
- W wielu przypadkach lepszą wydajność można jednak uzyskać stosując nieliniową kombinację sąsiednich pikseli.

Filtrowanie dwustronne

- W filtrze dwustronnym wartość wyjściowego piksela zależy od ważonej kombinacji sąsiednich wartości pikseli

$$g(i,j) = \frac{\sum_{k,l} f(k,l) w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$$

Współczynnik wagowy $w(i,j,k,l)$ zależy od produktu jądra domeny,

$$d(i,j,k,l) = exp(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2})$$

oraz jądro zakresu zależnego od danych:

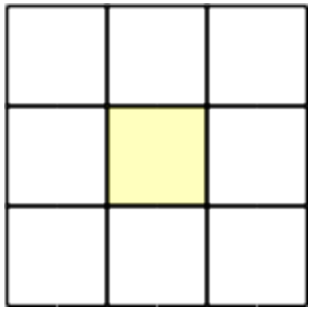
$$d(i,j,k,l) = exp(-\frac{||f(i,j) - f(k,l)||^2}{2\sigma_r^2})$$

Po pomnożeniu razem dają one zależną od danych dwustronną funkcję wagi:

$$d(i,j,k,l) = exp(-\frac{(i-k)^2+(j-l)^2}{2\sigma_d^2} - \frac{||f(i,j)-f(k,l)||^2}{2\sigma_r^2})$$

Filtrowanie dwustronne $cv.bilateralFilter()$ jest bardzo skuteczny w usuwaniu szumów przy jednoczesnym zachowaniu ostrych krawędzi. Ale działanie jest wolniejsze w porównaniu z innymi filtrami.

Operacje morfologiczne



- Podstawową koncepcją transformacji morfologicznej jest tzw. Element strukturalny obrazu,
- To jest pewna część obrazu (podzbiór elementów) z jednym punktem - centralnym punktem,
- Element konstrukcyjny (okrąg z promieniem jednostkowym) na siatce kwadratowej.
- Element strukturyzujący może mieć dowolny kształt, od prostego filtra pudełkowego 3×3 do bardziej skomplikowanych struktur dysków.

Element strukturyzujący - Python

Możemy stworzyć elementy strukturyzujące. W niektórych przypadkach można użyć jąder o kształcie eliptycznym lub okrągłym. W tym celu OpenCV ma funkcję, *cv.getStructuringElement*().

```
# Rectangular Kernel
>>> cv.getStructuringElement(cv.MORPH_RECT,(5,5))
array([[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]], dtype=uint8)
# Elliptical Kernel
>>> cv.getStructuringElement(cv.MORPH_ELLIPSE,(5,5))
array([[0, 0, 1, 0, 0],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [0, 0, 1, 0, 0]], dtype=uint8)
# Cross-shaped Kernel
>>> cv.getStructuringElement(cv.MORPH_CROSS,(5,5))
array([[0, 0, 1, 0, 0],
       [0, 0, 1, 0, 0],
       [1, 1, 1, 1, 1],
       [0, 0, 1, 0, 0],
       [0, 0, 1, 0, 0]], dtype=uint8)
```

Ogólny algorytm transformacji morfologicznej

- Element strukturalny jest przesuwany po całym obrazie i dla każdego punktu obrazu dokonywane jest porównanie punktów obrazu i elementu strukturalnego,
- W każdym punkcie obrazu sprawdzane jest, czy rzeczywista konfiguracja pikseli obrazu w pobliżu tego punktu jest zgodna z elementem konstrukcyjnym modelu,
- W przypadku stwierdzenia zgodności między wzorem pikselowym obrazu i szablonem elementu konstrukcyjnego, na badanym punkcie wykonywana jest pewna operacja.

Operacje morfologiczne

- **Erozja** - zerodowana figura to zbiór wszystkich środków okręgów o promieniu r, które są w całości zawarte w obszarze X
- **Dylacja** - figura dylacji to zbiór środków wszystkich okręgów B, dla których co najmniej jeden punkt pokrywa się z dowolnym punktem początkowej figury
- **Otwarcie** - polega na toczeniu kółkiem B po wewnętrznej stronie krawędzi figury i odrzucaniu wszystkich punktów, do których koło nie może dotrzeć.
- **Zamknięcie** - polega na toczeniu kółka B po zewnętrznej krawędzi figury i dodaniu do niej wszystkich punktów, do których koło nie może dotrzeć.

Morfologia - przykłady

Najpopularniejsze operacje na obrazach binarnych nazywane są operacjami morfologicznymi, ponieważ zmieniają one kształt leżących pod nimi obiektów binarnych.

(a) (b) (c) (d) (e) (f)

- Morfologia obrazu binarnego: (a) obraz oryginalny; (b) dylatacja; c) erozja; d) większość; (e) otwarcie; f) zamknięcie. Elementem strukturyzującym we wszystkich przykładach jest kwadrat 5×5. Efektem większości są subtelne zaokrąglenia ostrych rogów. Otwarcie nie eliminuje kropki, ponieważ nie jest wystarczająco szeroka.

Erozja

Podstawowa idea erozji jest taka, jak tylko erozja gleby, która powoduje erozję granic obiektu pierwszego planu.

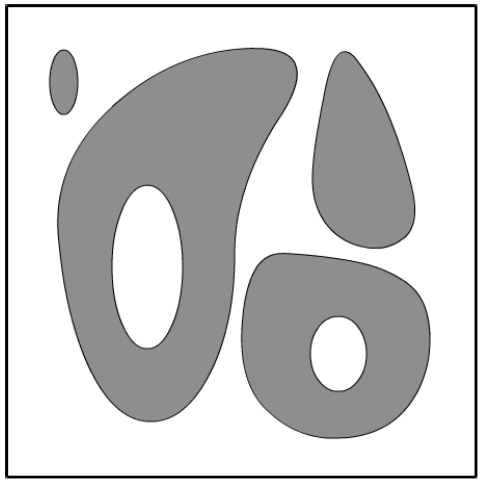


figura przed erozją

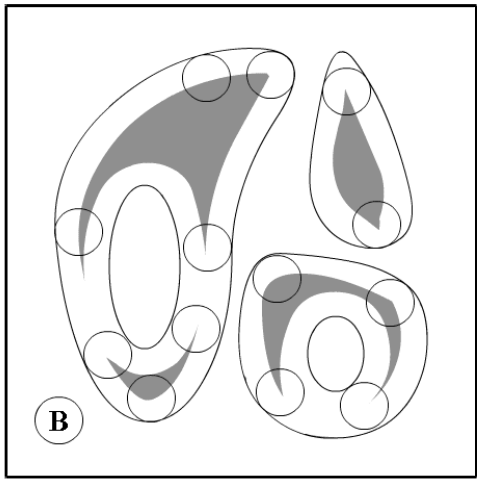


figura po erozji

```
import cv2 as cv
import numpy as np
img = cv.imread('j.png',0)
kernel = np.ones((5,5),np.uint8)
erosion = cv.erode(img,kernel,iterations = 1))
```

Dylacja

Jest to przeciwieństwo erozji. Zwiększa obszar obrazu lub zwiększa rozmiar obiektu na pierwszym planie

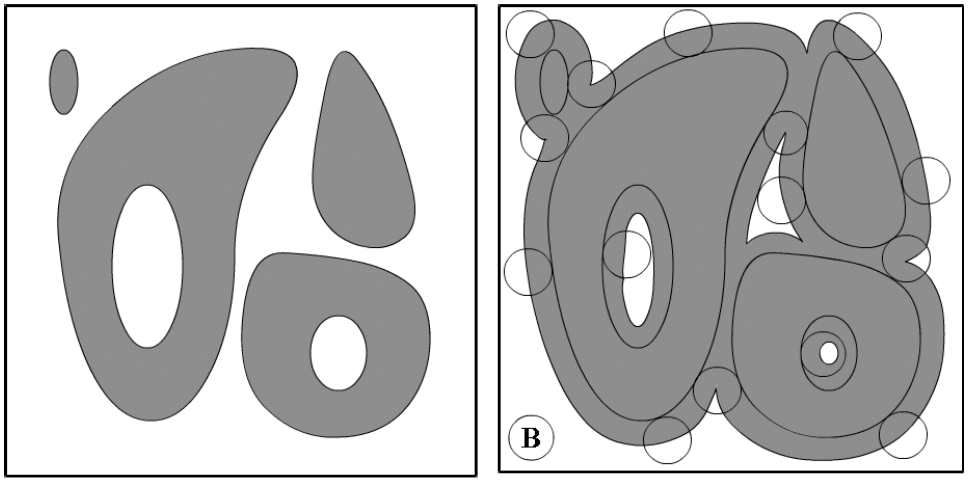


Figura przed dylatacją

figura po dylatacji

```
dilation = cv.dilate(img,kernel,iterations = 1)
```

Otwarcie

Otwarcie to tylko inna nazwa erozji, po której następuje dylatacja. Jest to przydatne w usuwaniu szumów.

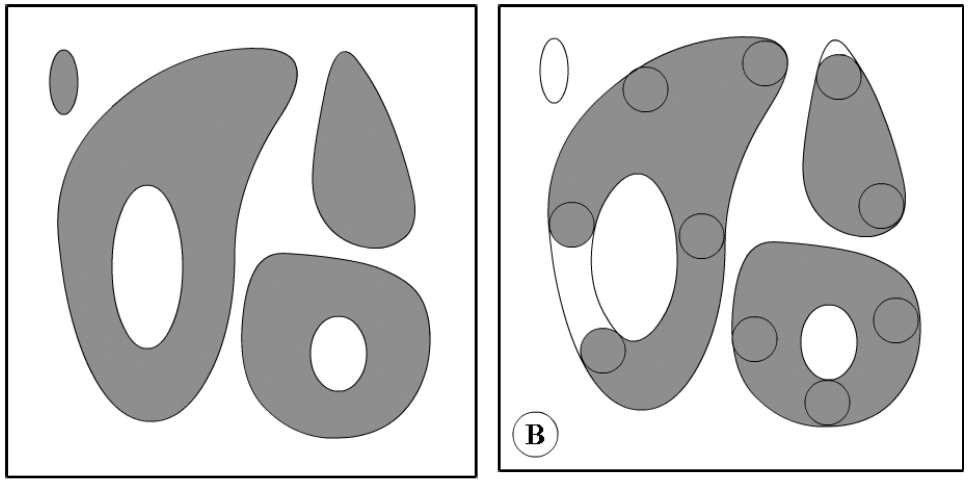


figura przed otwarciem

figura po otwarcia

- Otwarcie = Erozja + Dylacja: $O(x) = D(E(x))$

```
opening = cv.morphologyEx(img, cv.MORPH_OPEN, kernel)
```

Zamknięcie

Zamknięcie jest odwrotnością otwierania, rozszerzania, po którym następuje erozja. Służy do zamykania małych otworów wewnątrz obiektów pierwszego planu lub małych punktów na obiekcie.

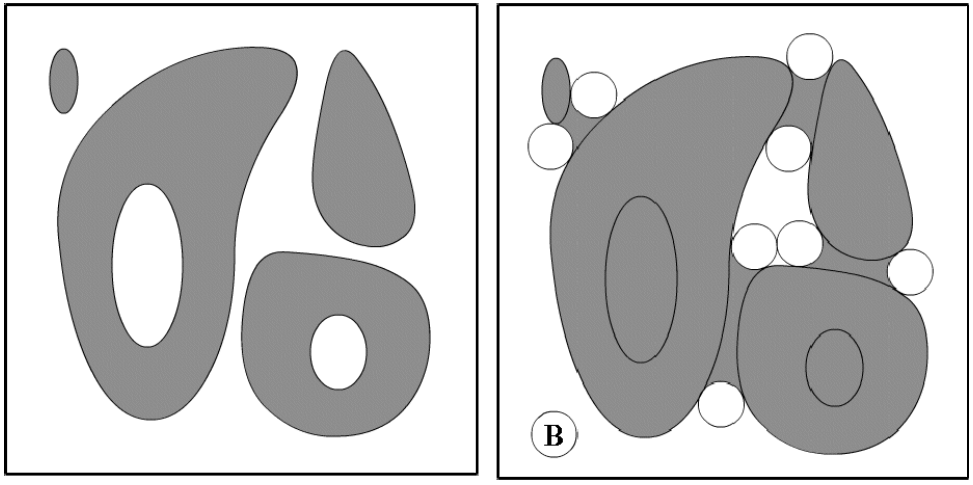


figura przed zamknięciem figura po zamknięciu

- Otwarcie = Dylacja + Erozja: $C(x) = E(D(x))$

```
closing = cv.morphologyEx(img, cv.MORPH_CLOSE, kernel)
```

Gradient morfologiczny

To jest różnica między dylatacją a erozją obrazu.

Wynik będzie wyglądał jak kontur obiektu

- Gradient = Dylacja - Erozja: $G(x) = D(x) - E(x)$

```
gradient = cv.morphologyEx(img, cv.MORPH_GRADIENT, kernel)
```

Cylinder ang. Top Hat

Jest to różnica między obrazem wejściowym a otwarciem obrazu. Poniższy przykład jest zrobiony dla jądra 9×9.

- TopHat = Obraz - Otwarcie: $TH(x) = x - O(x)$

```
tophat = cv.morphologyEx(img, cv.MORPH_TOPHAT, kernel)
```

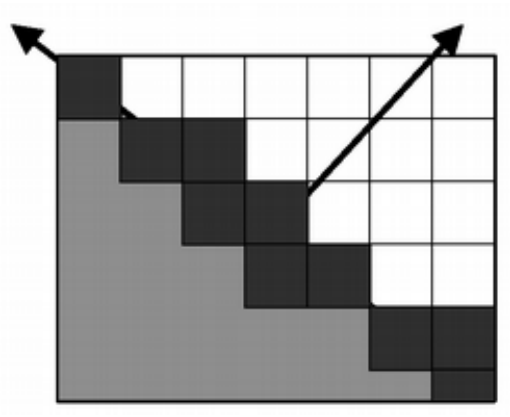
Czarny kapelusz ang. Black Hat

Jest to różnica między zamknięciem obrazu wejściowego i obrazu wejściowego.

- BlackHat = Obraz - Otwarcie: $BH(x) = C(x) - x$

```
blackhat = cv.morphologyEx(img, cv.MORPH_BLACKHAT, kernel)
```

Normalna do krawędzi



- Rozmiar krawędzi: $S = \sqrt{dx^2 + dy^2}$
- Detekcja krawędzi: $\alpha = arctan(\frac{dy}{dx})$

Operator Sobela

$S_1 =$

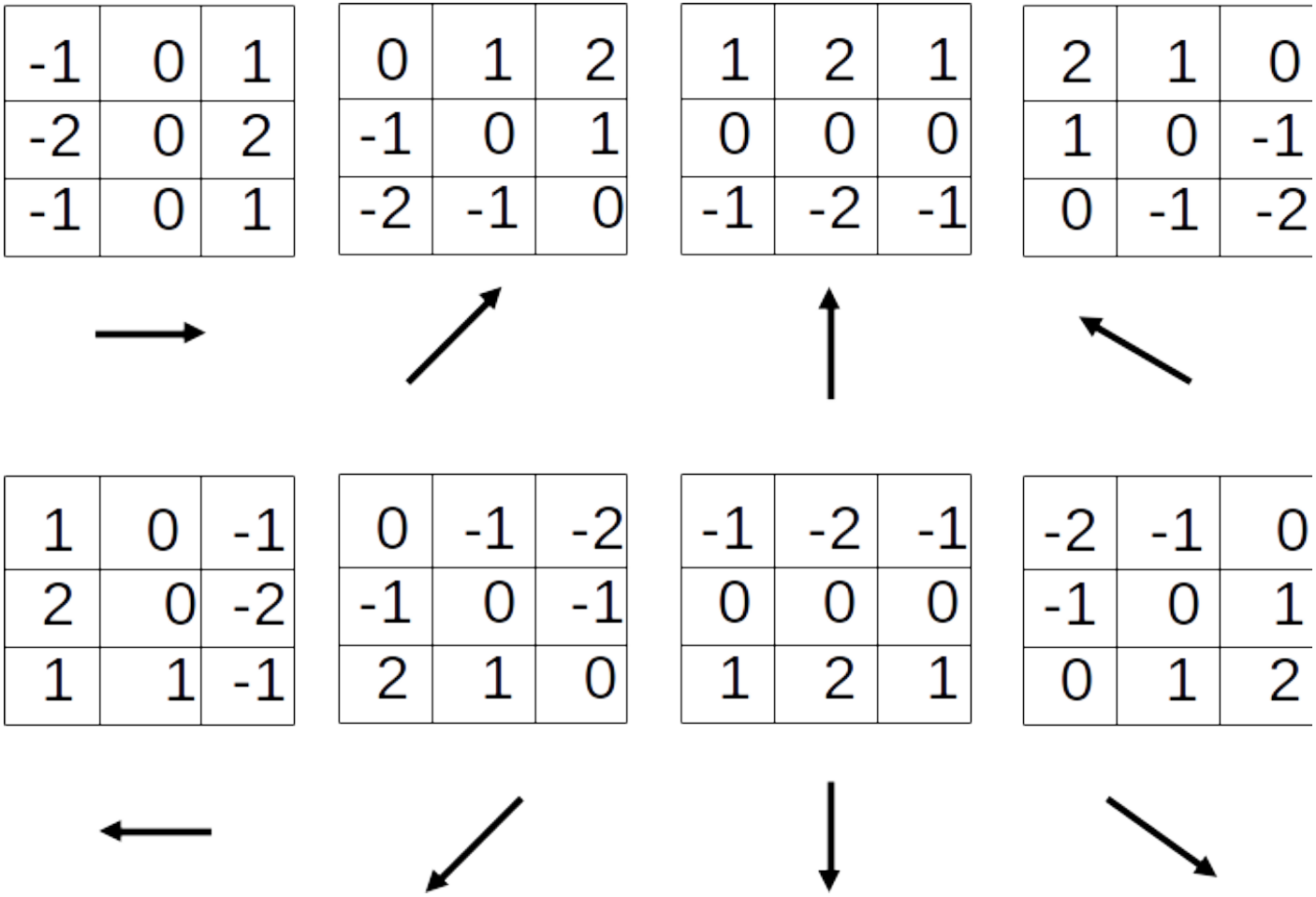
-1	-2	-1
0	0	0
1	2	1

$S_2 =$

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

- Rozmiar krawędzi: $\sqrt{S_1^2 + S_2^2}$
- Detekcja krawędzi: $arctan(\frac{S_1}{S_2})$

Różne kierunki filtracji brzegowej



Inne filtry liniowe

1	1	1
1	-2	1
-1	-1	-1

Prewitt 1

5	5	5
-3	0	-3
-3	-3	-3

Kirsch

-1	$-\sqrt{2}$	-1
0	0	0
1	$\sqrt{2}$	1

Frei & Chen

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

Prewitt 2

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

Sobel

- You can design your own linear filter.

Wykrywanie krawędzi Canny

Etapy filtracji:

1. etap 1 - wygładzanie,
2. etap 2 - obliczenie gradientu,
3. etap 3 - usuń maksymalną liczbę pikseli.

Wykrywanie narożników

Narożnik w koncepcji “ostro zakrzywionej krawędzi”, przecięcie 2 krawędzi, linie

- Pierwsze algorytmy wykrywające narożniki:
  - wykrywanie krawędzi,
  - śledzący wyodrębnione krawędzie i wykrywający nagłą zmianę ich kierunku,
- Nowsza generacja algorytmów - krzywizna o dużym gradiencie,
- Zalecane jest wcześniejsze wygładzenie obrazu.

Odległość

Transformacja odległości jest przydatna przy szybkim obliczaniu odległości do krzywej lub zbioru punktów.

Dwa powszechnie używane wskaźniki:

- Blokowy

$$d_1(k,l) = |k| + |l|$$

- Metryka Manhattan

$$d_2(k,l) = \sqrt{(k^2 + l^2)}$$

Powiązane elementy

Inną użyteczną operacją na pół-globalnym obrazie jest znajdowanie połączonych komponentów, które są zdefiniowane jako regiony sąsiednich pikseli, które mają tę samą wartość wejściową lub etykietę.

Takie statystyki obejmują dla każdego regionu R:

- obszar (liczba pikseli),
- obwód (liczba pikseli granicznych),
- centroid (średnie wartości xiy),
- momenty drugiego rzędu.

$$M = \sum_{(x,y)\in R} \begin{bmatrix} x-\overline{x} \\ y-\overline{y} \end{bmatrix} \cdot \begin{bmatrix} x-\overline{x} & y-\overline{y} \end{bmatrix}$$

Z którego można obliczyć orientację i długości głównych i pomocniczych osi za pomocą analizy wartości własnych.