Algorytmy Przetwarzania Obrazów

Operacje na obrazach (II)

WYKŁAD 3
Dla studiów niestacjonarnych 2019/2020

Dr hab. Anna Korzyńska, prof. IBIB PAN

Operacje na obrazach

- Operacje punktowe (jednopunktowe):

  Jednoargumentowe

  $$[q(i,j)] = f[p(i,j)]$$

  Wieloargumentowe

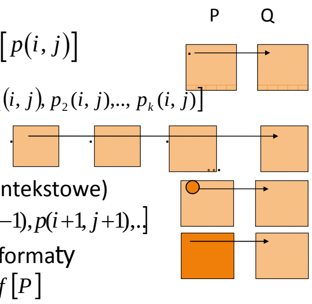
  $$[q(i,j)] = f[p_1(i,j), p_2(i,j),\ldots, p_k(i,j)]$$

- Operacje sąsiedztwa (kontekstowe)

  $$[q(i,j)] = f[p(i,j), p(i-1,j-1), p(i+1,j+1),..]$$

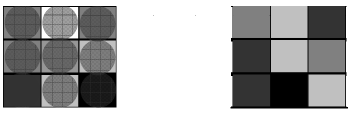
- Operacje globalne transformaty

  $$[q(i,j)] = f[P]$$



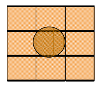
Operacje punktowe (lokalne, jednopunktowe)

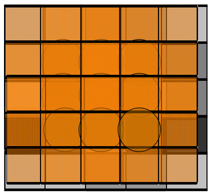
Do operacji punktowych; realizacja funkcji F

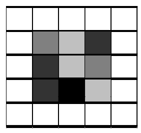


```
For i=0 to X do:
    Begin.
    For j=0 to Y do
        Begin.
        fnew(i,j):= F(f(i,j))
        End.
    End.
```


Proces liczenia operacji sąsiedztwa







Wynik operacji zależy od wielkości maski, ale głównie od funkcji zdefiniowanej na punkcie i jego otoczeniu.

Do operacji sąsiedztwa F dla maski 3x3

```
For i=2 to X-2 do:
    Begin.
    For j=2 to X-2 do
        Begin.
        f new(i,j):= F( f(i-1, j-1), f(i-1, j),
        f(i-1, j+1), f(i, j-1), f(i,j) f(i, j+1),
        f(i+1, j-1), f(i+1, j), f(i+1, j+1))
        End.
    End.
```


Funkcjonalny podział operacji sąsiedztwa:

- operacje *wygładzania*
- operacje *wyostrzania*

Operacje wygładzania stanowią praktyczną realizację *filtracji dolnoprzepustowej* (FD) i dzielą się na operacje filtracji ***liniowej* i *nieliniowej*.**

Operacje filtracji nieliniowej dzielą się na operacje filtracji ***logicznej*** i ***medianowej***.

Operacje wyostrzania stanowią praktyczną realizację *filtracji górnoprzepustowej* (FG) i dzielą się na operacje filtracji ***gradientowej*** i ***laplasjanowej***

Przykłady macierzy wag i masek operacji filtracji liniowej

Macierz wag

1/10	1/10	1/10
1/10	2/10	1/10
1/10	1/10	1/10

1/16	2/16	1/16
2/16	4/16	2/16
1/16	2/16	1/16

Maska filtracji dolnoprzepustowej

1	1	1
1	2	1
1	1	1

K = 1/10

1	2	1
2	4	2
1	2	1

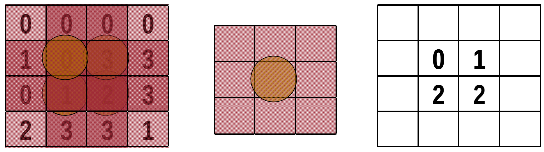
K = 1/16

Filtracja nieliniowa; wygładzanie medianowe

Mediana - wartość **środkowa** (w sensie położenia w ciągu wartości uporządkowanych)

Usuwanie zakłóceń **bez rozmywania krawędzi** (por. metodę filtracji liniowej)

Mediana i pozostałe filtry statystyczne



1, 2, 2, 4, 5, 6, 7, 8, 9

0, 0, 0, <u>0</u>, 1, 1, 2, 3 mediana=0, min=0, max=3, najbardziej prawdopodobna=0

0, 0, 0, 0, <u>1</u>, 2, 3, 3, 3 mediana=1, min=0, max=3, najbardziej prawdopodobna=0

0, 0, 1, 1, <u>2</u>, 2, 3, 3, 3 mediana=2, min=0, max=3, najbardziej prawdopodobna=3

0, 0, 1, 1, <u>2</u>, 2, 3, 3, 3 mediana=2, min=0, max=3, najbardziej prawdopodobna=3

Operacje wyostrzania

Metoda: konwolucja + maska *filtracji górnoprzepustowej(FG).*

W wyostrzaniu stosuje się metody numeryczne aproksymujące pochodną.

<u>Zadanie wyostrzania</u>:
- podkreślenie na obrazie konturów obiektów
- podkreślenie na obrazie punktów informatywnych (np. wierzchołki dla wielokątów, zakończenia, skrzyżowania, rozgałęzienia linii dla rysunków technicznych, wykresów lub pisma).

Model zadania wyostrzania: wydobycie i uwypuklenie krawędzi obiektu.

Cyfrowa wersja gradientu

Pochodna pionowa G_x funkcji f(x,y)

$$G_x \stackrel{def}{=} [f(x+1,y-1)+2f(x+1,y)+f(x+1,y+1)] - [f(x-1,y-1)+2f(x-1,y)+f(x-1,y+1)]$$

maska:

	y-1	*y*	*y+1*
x-1	-1	-2	-1
x	0	0	0
x+1	1	2	1

Pochodna pozioma G_y funkcji f(x,y)

$$G_y \stackrel{def}{=} [f(x-1,y+1)+2f(x,y+1)+f(x+1,y+1)] - [f(x-1,y-1)+2f(x,y-1)+f(x+1,y-1)]$$

$$G(x,y) = \sqrt{G_x^2 + G_y^2}$$

maska:

	y-1	*y*	*y+1*
x-1	-1	0	1
x	-2	0	2
x+1	-1	0	1

Cyfrowa wersja laplasjanu

$L(x, y) = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4 f(x, y)]$

maska:

	y-1	*y*	*y+1*
x-1	0	1	0
x	1	-4	1
x+1	0	1	0

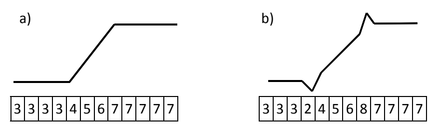
Własności:
Gradient: wrażliwy na intensywność zmiany; używany tylko do detekcji krawędzi;
Laplasjan: podaje dodatkową informację o położeniu piksela względem krawędzi (po jasnej czy po ciemnej stronie).

<u>Uwaga:</u> Dla operacji wyostrzania współczynnik maski **K=1**

Krawędź obrazu widoczna w przekroju (xz)

a) obraz pierwotny
b) obraz wynikowy po obróbce laplasjanowej i po dodaniu w sposób ważony jasności odpowiednich pikseli
c) maska laplasjanu

Rezultat ***uwypuklenie*** (wzmacnianie) krawędzi (edge enhancement)



Wartość punktów

c)

-1	2	-1

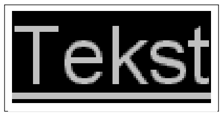
Wyznaczanie i wzmacnianie konturach obiektów

Wyostrzenie - **złożenie** obrazów:
- wejściowego,
- po operacji zadanej laplasjanem, następnie przeskalowanie stopni szarości (jak w operacjach jednopunktowych).

a)			b)			c)			d)			e)		
0	-1	0	-1	-1	-1	1	-2	1	-1	-1	-1	0	-1	0
-1	4	-1	-1	8	-1	-2	4	-2	-1	9	-1	-1	5	-1
0	-1	0	-1	-1	-1	1	-2	1	-1	-1	-1	0	-1	0

13

Laplasjany

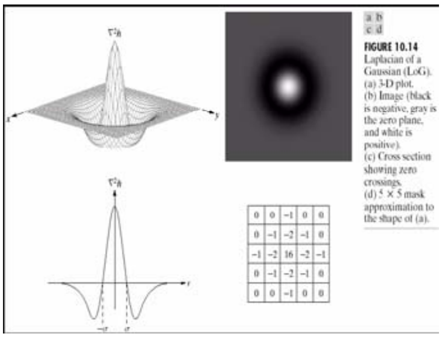


Dyskretna forma drugiej pochodnej

0	-1	0		-1	-1	-1		1	-2	1		-1	-1	-1		0	-1	0
-1	4	-1		-1	8	-1		-2	4	-2		-1	9	-1		-1	5	-1
0	-1	0		-1	-1	-1		1	-2	1		-1	-1	-1		0	-1	0

14

Laplasjan filtra gaussowskiego



15

Metoda specjalnego gradientu

Stosowana w przypadkach, gdy metody filtracji górnoprzepustowej (FG) powodują wzmocnienie zakłóceń w obszarach leżących wewnątrz konturu.

Zasada
Krawędź uznana jest za istniejącą, jeśli wartość gradientu intensywności w pewnych punktach przekracza ustalony próg. Metody aproksymacji: **Robertsa**, **Sobela**, **Prewitta**.

Oznaczenia pikseli:

f_0	f_1	f_2
f_3	f_4	f_5
f_6	f_7	f_8

(i,j)

16

Maski konwolucyjne kierunkowego krawędziwania

Roberts

G_x			G_y	
1	0		0	-1
0	-1		1	0

Sobel:

G_x				G_y		
-1	0	1		-1	-2	-1
-2	0	2		0	0	0
-1	0	1		1	2	1

Prewitt:

G_x				G_y		
1	0	-1		1	1	1
1	0	-1		0	0	0
1	0	-1		-1	-1	-1

Zadanie 3

17

Operatory krawędziowania

Konstruowane przez nieliniową kombinacje dwóch prostopadłych kierunków gradientu (liniowych transformacji)

dokładnej **przybliżonej**

$$G = \sqrt{Gx^2 + Gy^2} \qquad G = |Gx| + |Gy|$$

Robertsa

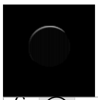
Gx	1	0	lub	-1	0
	0	-1		0	1
	0	1	lub	0	-1
Gy	-1	0		1	0

Sobela

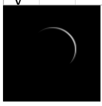
	1	0	-1		-1	0	1
Gx	2	0	-2	lub	-2	0	2
	1	0	-1		-1	0	1
	1	2	1		-1	-2	-1
Gy	0	0	0	lub	0	0	0
	-1	-2	-1		1	2	1

18

Filtry i operatory Robertsa




Gx	1	0	lub	-1	0
	0	-1		0	1
	0	1	lub	0	-1
Gy	-1	0		1	0




$$G = \sqrt{Gx^2 + Gy^2}$$

Filtry kierunkowe rzeźbiące

Wschód				Południowy wschód				Południe	Południowy zachód				
-1	0	1		-1	-1	0		-1	-1	-1		0	-1	-1
-1	1	1		-1	1	1		0	1	0		1	1	-1
-1	0	1		0	1	1		1	1	1		1	1	0

E  SE  S  SW 

W  NW  N  NE 

1	0	-1		1	1	0		1	1	1		0	1	1
1	1	-1		1	1	-1		0	1	0		-1	1	1
1	0	-1		0	-1	-1		-1	-1	-1		-1	-1	0
Zachód				Północny zachód				Północ (N)				Północny wschód		

Metody operacji na pikselach wchodzących w skład skrajnych kolumn i wierszy

1. Pozostawienie wartości pikseli bez zmian
2. Wartości pikseli są nieokreślone (xxxxxxxxxx)
3. Nadanie pikselom wartości arbitralnie zadanych przez użytkownika (np. same wartości „0", „15", „10" itd.
4. Operacje z zastosowaniem kolumn i wierszy pomocniczych (zdublowanie (powielenie) skrajnych wierszy i kolumn)
5. Operacje z wykorzystaniem pikseli z **istniejącego** sąsiedztwa.
   - Lewa skrajna kolumna (oprócz pikseli górnego i dolnego rogu) – kierunki 0,1,2,6,7,
   - Lewa skrajna kolumna piksel w górnym rogu – kierunki 0, 6,7,
   - Lewa skrajna kolumna (piksel w dolnym rogu) – kierunki 0,1,2,
   - Prawa skrajna kolumna (oprócz pikseli górnego i dolnego rogu) – kierunki 2,3,4,5,6,
   - Prawa skrajna kolumna piksel w górnym rogu – kierunki 4,5,6,
   - Prawa skrajna kolumna (piksel w dolnym rogu) – kierunki 2,3,4,
   - Górny skrajny wiersz (oprócz pikseli z lewego i prawego rogu) – kierunki 4,5,6,7,0
   - Dolny skrajny wiersz (oprócz pikseli z lewego i prawego rogu) – kierunki 0,1,2,3,4.

Metody skalowania tablic obrazów wynikowych

Cel skalowania: sprowadzanie wartości pikseli do zakresu **[0, *(M-1)*]**

Metoda proporcjonalna

$$g'(x,y) = \frac{g(x,y) - g(x,y)_{\min}}{g(x,y)_{\max} - g(x,y)_{\min}} \cdot (M-1)$$

Własność:
Równomierne przeskalowanie wszystkich pikseli obrazu.
Końcowy efekt: obraz z zakresu [0, (*M-1*)]

Metoda trójwartościowa

$$g'(x,y) = \begin{cases} 0 \text{ dla } g(x,y) < 0 \\ E[(M-1)/2] \text{ dla } g(x,y) = 0 \\ M-1 \text{ dla } g(x,y) > 0 \end{cases}$$

Zastosowanie
obrazy o jednolitym tle i dobrze widocznych obiektach - np. obrazy binarne. Efekt: czarno-biała krawędź na szarym tle.

Metoda obcinająca

$$g'(x,y) = \begin{cases} 0 \text{ dla } g(x,y) < 0 \\ g(x,y) \text{ dla } 0 \le g(x,y) \le M-1 \\ M-1 \text{ dla } g(x,y) > M-1 \end{cases}$$

Biblioteka OpenCV

It has C++, Python, Java and MATLAB interfaces
and supports Windows, Linux, Android and Mac OS.
There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++.

OpenCV + Python

- Instalacja:
  - pip install numpy (niezbędne do działania openCV ze względu na obliczenia macierzowe obrazów)
  - pip install opencv-python
- Wczytanie:
  - import cv2 as cv
- Test:
  - print(cv.__version__)

OpenCV.js

- Example for synchronous loading:

```
<script src="opencv.js"
type="text/javascript"></script>
```

- Po załadowaniu jest gotowy do użycia:

```
imgElement.onload = function() {
      let mat = cv.imread(imgElement);
      cv.imshow('canvasOutput', mat);
      mat.delete();
};
```

OpenCV in C++

- Opcja 1:
  - #include *"opencv2/core/core.hpp"*
  - cv::Mat H = cv::findHomography(points1, points2, CV_RANSAC, 5);
- Opcja 2:
  - #include *"opencv2/core/core.hpp"*
  - **using namespace** cv;
  - Mat H = findHomography(points1, points2, CV_RANSAC, 5);

Wygładzanie $K = \frac{1}{\texttt{ksize.width*ksize.height}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 1 \\ \cdots \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}$

- Python:

  cv2.blur(src, ksize[, dst[, anchor[, borderType]]]) → dst
- C++

  void blur(InputArray **src**, OutputArray **dst**, Size **ksize**, Point **anchor**=Point(-1,-1), int **borderType**=BORDER_DEFAULT)
- JS

  cv.blur(src, dst, ksize, anchor, cv.BORDER_DEFAULT);

Zadanie 1

Laplasjan $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

- Python:

  cv2.Laplacian(src, ddepth[, dst[, ksize[, scale[, delta[, borderType]]]]]) → dst
- C++

  void Laplacian(InputArray **src**, OutputArray **dst**, int **ddepth**, int **ksize**=1, double **scale**=1, double **delta**=0, int **borderType**=BORDER_DEFAULT)
- JS

  cv2.Laplacian(src, dst, cv.CV_8U, ksize, scale, 0, cv.BORDER_DEFAULT);

Zadanie 1

Filtracja medianowa

- Python:

  cv2.medianBlur(src, ksize[, dst]) → dst
- C++

  void medianBlur(InputArray **src**, OutputArray **dst**, int **ksize**)
- JS

  cv.medianBlur(src, dst, ksize);

Zadanie 3

Ogólnie – filtr 2D - argumenty

- Argumenty wejściowe:
  - src – obraz wejściowy
  - ddepth – głębokość obrazu wyjściowego (ddepth = -1 dla zachowania wartości z obrazu wejściowego)
  - kernel – jądro przekształcenia
  - anchor – punkt zaczepienia jądra przekształcenia
  - delta – (opcjonalnie) stała dodana do wartości obrazu
  - dst – obraz wyjściowy
  - borderType – określa postępowanie z pikselami brzegowymi
  - scale – (opcjonalne) określenie liczby przez którą mnożymy w celu przeskalownaia

  cv2.filter2D(src, ddepth, kernel[, dst[, anchor[, delta[, borderType]]]]) → dst

Ogólnie – filtr 2D

Argumenty wejściowe: obraz, rozmiar otoczenia

- Python:

  cv2.filter2D(src, ddepth, kernel[, dst[, anchor[, delta[, borderType]]]]) → dst
- C++

  void filter2D(InputArray **src**, OutputArray **dst**, int **ddepth**, InputArray **kernel**, Point **anchor**=Point(-1,-1), double **delta**=0, int **borderType**=BORDER_DEFAULT)
- JS

  cv.filter2D(src, dst, cv.CV_8U, cv.Mat.eye(3, 3, cv.CV_32FC1), anchor, 0, cv.BORDER_DEFAULT);

Wartości brzegowe

- Wiele funkcji (klas) w OpenCV przyjmuje jako argument wejściowy zmienną określającą jak należy postępować z pikselami brzegowymi
- Typowo jest to parametr **borderType**

```
/* Various border types,
           image boundaries are denoted with '|'

 * BORDER_REPLICATE: aaaaaa|abcdefgh|hhhhhhh
 * BORDER_REFLECT: fedcba|abcdefgh|hgfedcb
 * BORDER_REFLECT_101: gfedcb|abcdefgh|gfedcba
 * BORDER_WRAP: cdefgh|abcdefgh|abcdefg
 * BORDER_CONSTANT: iiiiii|abcdefgh|iiiiiii
                           with some specified 'i' */
```

Pomoc - przykłady

- https://docs.opencv.org/3.4/dc/dd3/tutorial_gausian_median_blur_bilateral_filter.htmlhttps://docs.opencv.org/3.4/dc/dd3/tutorial_gausian_median_blur_bilateral_filter.html
-
- https://docs.opencv.org/3.4/db/d8e/tutorial_threshold.htmlhttps://docs.opencv.org/3.4/db/d8e/tutorial_threshold.html
-
- https://docs.opencv.org/3.4/d4/dbd/tutorial_filter_2d.htmlhttps://docs.opencv.org/3.4/d4/dbd/tutorial_filter_2d.html
-
- https://docs.opencv.org/3.4/d2/d2c/tutorial_sobel_derivatives.htmlhttps://docs.opencv.org/3.4/d2/d2c/tutorial_sobel_derivatives.html
-
- https://docs.opencv.org/3.4/d5/db5/tutorial_laplace_operator.htmlhttps://docs.opencv.org/3.4/d5/db5/tutorial_laplace_operator.html
-
- https://docs.opencv.org/3.4/da/d5c/tutorial_canny_detector.htmlhttps://docs.opencv.org/3.4/da/d5c/tutorial_canny_detector.html

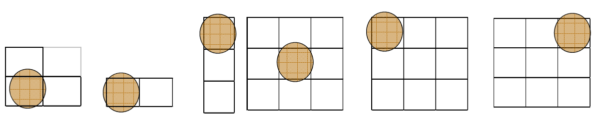
Operacje morfologii matematycznej

Operacje morfologii matematycznej na obrazach

Operacje pozwalające na budowanie złożonych operacji, pozwalających na analizę kształtu i wzajemnego położenia obiektów.

Fundamentalne pojęcie: **element strukturalnym (strukturujący)**

– podzbiór obrazu z wyróżnionym punktem, zwanym często punktem centralnym



36

Operacje morfologii matematycznej na obrazach

– w elemencie strukturalnym występują następujące symbole:
- **1** element wskazuje piksel zapalony tzn. wartość obiektu w masce binarnej
- **0** element wskazuje piksel wytłumiony tzn. wartość tła w masce binarnej
- **X** element wskazuje dowolną wartość tzn. wartość tła lub obiektu w masce binarnej

Przekształcenia polegają na zmianie intensywności lub pozostawieniu intensywności punktu przykrytego przez punkt centralny elementu strukturalnego w zależności od spełnienia warunków logicznych.

Operacje morfologiczne przekształcają tylko część punktów obrazu

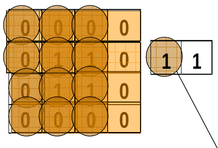
37

Operacje morfologiczne

1. Element strukturalny jest przemieszczany po wszystkich punktach obrazu tak, że punkt centralny elementu strukturalnego jest nakładany na kolejne punkty w kolejnych wierszach,
2. W każdym położeniu elementu sprawdza się, czy rzeczywista konfiguracja punktów jest zgodna (koincydentna) ze wzorcem zawartym w elemencie strukturalnym zakodowanym symbolami 1, 0, X
3. W przypadku wykrycia zgodności jest wykonywana operacja związana z filtrem, a w przeciwnym przypadku wartość występująca w obrazie pierwotnym jest przepisywana.

38

Operacje morfologiczne



0	0	0	
0	1	0	
0	1	0	
0	0	0	

Jeśli punkt otoczenia jest wygaszony (równy wartości tła - 0) przy zapalonym (większym od tła - 1) elemencie centralnym, wynik zostaje wygaszany, a w przeciwnym wypadku zostawiamy poprzednią wartość

39

Podstawowe operacje morfologii matematycznej

0-zgaszony; 1-zapalony; X-o dowolnej wartości.

- Erozja

$$q(i,j) = \min_{i_n, j_m \in B(i,j)} (p(i_n, j_m))$$

1	1	1
1	1	1
1	1	1

- Dylatacja (dylacja) negatyw erozji

$$q(i,j) = \max_{i_n, j_m \in B(i,j)} (p(i_n, j_m))$$

X	X	X
X	0	X
X	X	X

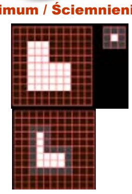
B(i,j) element strukturalny z punktem centralnym o współrzędnych (i,j)

Dylatacja jest operacją dualną do erozji i na odwrót

40

Przykłady operacji erozji

Minimum / Ściemnienie



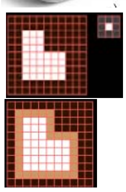
O małym (2x2)/dużym(6x6) kwadratowym elemencie strukturalnym

O wertykalnym/horyzontalnym elemencie strukturalnym

Praca dyplomowa Szymona Mireckiego

41

Przykłady operacji dylacji na obrazach w skali szarości



O małym (2x2)/dużym(6x6) kwadratowym elemencie strukturalnym

O wertykalnym/horyzontalnym elemencie strukturalnym

Maksimum / Rozjaśnianie

Praca dyplomowa Szymona Mireckiego

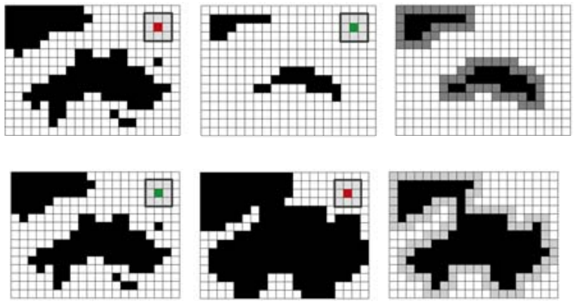
42

Inne operacje morfologii matematycznej

- Otwarcie (=Erozja+Dylacja)
- Zamknięcie[domknięcie] (=Dylacja+Erozja)
- Detekcja ekstremów Top Hat (=Zamknięcie-Obraz=Obraz-Otwarcie)
- Gradient morfologiczny (= Otwarcie+Zamknięcie)
- Wygładzanie morfologiczne (=Dylacja–Erozja)
- Pocienianie
- Pogrubianie
- Szkieletyzacja (znalezienie szkieletu czyli punktów obiektu równoodległych od jej brzegów)
- Odcinanie gałęzi (artefaktów z nieregularności obiektów szkieletyzowanych)
- Detekcja centroidów (punktów centralnych obiektu)
- Dylatacja bez styków (SKIZ ang. Skeleton by influece zone)
- Erozja warunkowa
- Rekonstrukcja (wygłodzanie obszru, czyszczenie brzegów, zalewanie dziur)
- Automediana

43

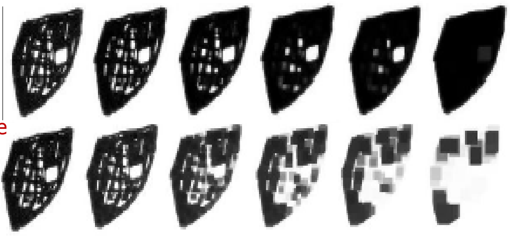
Otwarcie i zamknięcie



Operacje otwarcia i zamknięcia na obrazach szaroodcieniowych

Otwarcie

Zamknięcie



Element strukturalny: 3x3 5x5 9x9 13x13 21x21

Komplementarność operacji

Praca dyplomowa Szymona Mireckiego

45

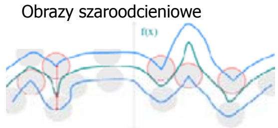
Otwarcie i zamknięcia – interpretacja geometryczna

Obrazy binarne

Otwarcie: toczenie elementu strukturalnego od wewnątrz

Zamknięcie: toczenie elementu strukturalnego od zewnątrz

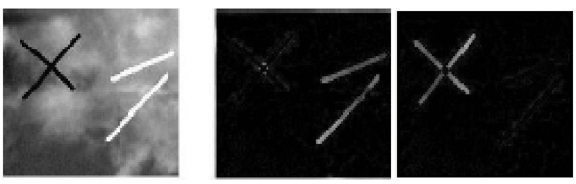
Obrazy szaroodcieniowe



Otwarcie/Zamknięcie: toczenie kulki/piłki czyli elementu strukturalnego od dołu/góry linii profilu dla obrazów w skali szarości

46

Top Hat

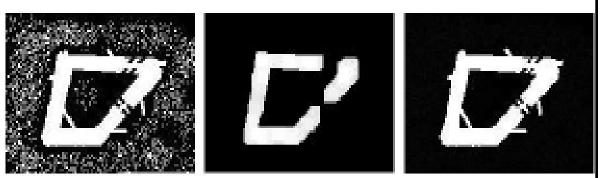


White Top Hat

Black Top Hat

47

Rekonstrukcja

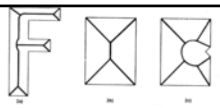


Otwarcie

Rekonstrukcja

48

Algorytmy szkieletyzacji



Umożliwiają upraszczanie obiektów na obrazach prowadząc do zastąpienia obriktu jego szkieletem

Szkielet odzwierciedla podstawowe własności topologiczne figury, a jego dalsza analiza może zostać wykorzystana do:

- klasyfikacji figur ze względu na kształt,
- wyznaczania orientacji figur podłużnych,
- określania linii środkowej szerszych linii,
- rozdzielanie złączonych obiektów.

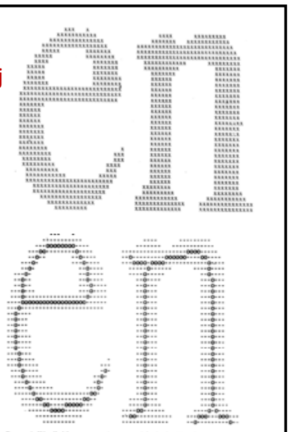
Ścienianie jest potrzebne, aby odtworzyć liniową strukturę obrazu wejściowego nie niszcząc jego spójności.

Matematyczna definicja ścieniania na płaszczyźnie analogowej:

- *DEFINICJA 1: Niech R będzie zbiorem punktów na płaszczyźnie, B jego brzegiem, a P punktem należącym do R. Najbliższym sąsiadem punktu P na brzegu B jest punkt M należący do B taki, że nie istnieje inny punkt należący do B, którego odległość od punktu P jest mniejsza od odległości PM. Jeżeli punkt P ma więcej niż jednego najbliższego sąsiada, to P nazywamy punktem szkieletowym zbioru R. Zbiór wszystkich punktów szkieletowych jest szkieletem lub osią środkową zbioru R.*

Szkielet figury, to zbiór wszystkich punktów równoodległych od co najmniej dwóch brzegów.

DEFINICJA 2.
Szkieletem zbioru R elementów obrazu cyfrowego jest zbiór wyznaczony w następujący sposób.
W zbiorze R określa się:
- potencjalnie szkieletowe (otoczone punktami obiektu) lub szkieletowe (stanowiące krzywą lub prostą ciągnącą się w dowolnym kierunku)
oraz
- konturowe (w otoczeniu jest poziom jasności tła 0)
elementy obrazu.
Następnie usuwa się wszystkie konturowe elementy obrazu, które nie są szkieletowymi i z tak otrzymanym zbiorem R rekurencyjnie powtarzamy procedurę aż do uzyskania zbioru zawierającego jedynie szkieletowe elementy obrazu.



Rys. 4. Wynik klasycznego algorytmu ścieniania. Usunięto elementy obrazu oznac „–", a szkieletowe elementy obrazu oznaczono przez „0"

Wyznaczanie szkieletu binarnego polega na wielokrotnym stosowaniu (często naprzemiennych, z różnymi elementami strukturalnymi) operacji pocieniania – do momentu, aż kolejne operacje nie wpływają na wygląd obrazu wynikowego. W tym celu można stosować różne zestawy elementów strukturalnych. Przykładem adekwatnego zestawu jest 8 elementów otrzymanych w wyniku obrotów następujących elementów strukturalnych:

$$\begin{bmatrix} 0 & 0 & 0 \\ z & 1 & z \\ 1 & 1 & 1 \end{bmatrix} \text{ oraz } \begin{bmatrix} z & 0 & 0 \\ 1 & 1 & 0 \\ z & 1 & z \end{bmatrix} \text{ o kąty } 0^\circ, 90^\circ, 180^\circ \text{ i } 270^\circ.$$

Algorytm 1 (klasyczny algorytm ścieniania)**:**

Oznaczenia: I oznacza obraz wejściowy. P oznacza zbiór wzorców sąsiedztwa szkieletowych elementów obrazu wraz z obróconym o 90° pierwszym wzorcem i obróconym o 90° , 180° i 270° drugim wzorcem. Znacznik *remain* z wartością *true* wskazuje, że nieszkieletowe elementy obrazu mogą pozostać. Znacznik *skel* z wartością *true* wskazuje, że sąsiedztwo elementu obrazu odpowiada jednemu ze wzorców zbioru P. Jedynka/zero we wzorcu odpowiada niezerowemu/zerowemu elementowi w sąsiedztwie.

1. Podstaw *true* jako wartość znacznika *remain* .
2. While *remain* = *true* **do** kroki 3-12.
Begin
3. Podstaw *false* jako wartość znacznika *remain* . {nie dokonano żadnej zmiany}
4. **For** j = 0,2,4 i 6 **do** kroki 5-12.
Begin
5. **For** dla wszystkich elementów p obrazu I **do** kroki 6-10.
Begin
6. **If** p = 1 **and if** jego j-sąsiad = 0 **then do** kroki 7-10.
Begin
7. Podstaw *false* jako wartość znacznika *skel.*
8. **For** wszystkich wzorców P **do** krok 9.
Begin
9. If *sąsiedztwo* p odpowiada wzorcowi P **then** podstaw *true* jako wartość *skel* i wyjdź z pętli
End.
10. If *skel =true* **then** podstaw 2 jako wartość p {szkieletowy element obrazu} **else** podstaw 3 jako wartość p {usuwamy element obrazu}i podstaw *true* jako wartość *remain.*
End.
End.
11. **For** wszystkich elementów p obrazu I **do** krok 12.
Begin
12. **If** p = 3, **then** podstaw jako p wartość 0.
End.
End.
End.
13. Koniec algorytmu.

$$\begin{bmatrix} 0 & 0 & 0 \\ z & 1 & z \\ 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} z & 0 & 0 \\ 1 & 1 & 0 \\ z & 1 & z \end{bmatrix}$$

$$0^\circ, 90^\circ, 180^\circ \text{ i } 270^\circ$$

a)

	A	A	A			A	A	A	
	0	P	0			A	P	0	
	B	B	B			A	0	2	

b)

A	A	C
0	2	2+
B	B	C

Rys. 3. Wzorce sąsiedztwa powtarzalnych elementów obrazu. a) Co najmniej jeden z każdej grupy elementów obrazu oznaczonych przez A lub B musi być niezerowy. b) Co najmniej jeden z elementów obrazu oznaczony przez C musi być niezerowy. Jeżeli obydwa elementy obrazu oznaczone przez C są niezerowe, to wartości elementów obrazu oznaczonych przez A i B mogą być dowolne. W przeciwnym przypadku co najmniej jeden z elementów każdej pary oznaczonej przez A lub B musi być niezerowy

Uwagi ogólne dotyczące algorytmów

- Dyskusja nad algorytmami obejmuje m.in. zagadnienie złożoności obliczeniowej.
- Podając algorytm próbujemy określić czas i pamięć potrzebne do jego wykonania.
- Typowy błąd: mylenie złożoności obliczeniowej i programowej.
- Generalnie, długość programu realizującego algorytm ma mało wspólnego z szybkością wykonania, a nawet z wymaganą wielkością pamięci.
- Jeśli jakaś relacja istnieje, to jest ona wręcz odwrotna.
- Algorytmy „złożone" są zwykle szybsze niż „proste". Np. program dla FFT (nierekurencyjny) jest dłuższy i bardziej złożony niż program realizujący wzór sumy dla transformaty. Jednak wykonuje się znacznie szybciej. Podobnych przykładów dostarczają algorytmy sortowania.
- Często atrakcyjniejsze wydaje się użycie rekurencyjnej formy algorytmu, jako dużo krótszej niż nierekurencyjna, a liczba operacji w obu formach jest taka sama. W takich przypadkach należy pamiętać o kosztach wywołań rekurencyjnych, potrzebie przechowywania wartości rejestrów w pamięci itp. Jeśli liczba wywołań jest mała w porównaniu z liczbą innych operacji, to ich koszt może być opłacalny z powodu prostoty programu. W innych przypadkach algorytmy w formie nierekurencyjnej dają programy wydajniejsze.
- O ile **prostota programowania** może wydawać się atrakcyjna programiście, który jest ograniczony czasem i planuje uruchomienie programu z niewielką ilością danych, o tyle jest **szkodliwa** w przypadkach zastosowań użytkowych przy dużych zbiorach danych.

Materiał:

- M.Doros, Przetwarzanie obrazów, skrypt WSISIZ
- Materiały wykładowe POBZ z zeszłego roku na UBIKu
- T.Pavlidis, Grafika i Przetwarzanie Obrazów, WNT Warszawa 1987.
- **I.Pitas,** Digital image processing, algorithms and applications, John Wiley &Sons, Inc. 2000, **pp. 162-166** (**w katalogu …*APOZ\\Materialy* na UBIKu**).

Omówienie tematów projektów

- **Materiał w katalogach na UBIKu:**

…\\ *APOZ\\Materialy*

…\\ *APOZ\\2019-2020\\Projekty*