Algorytmy Przetwarzania Obrazów

Operacje na obrazach (III)

WYKŁAD 3
Dla studiów niestacjonarnych 2022/2023

Dr hab. Anna Korzyńska, prof. IBIB PAN

Operacje na obrazach

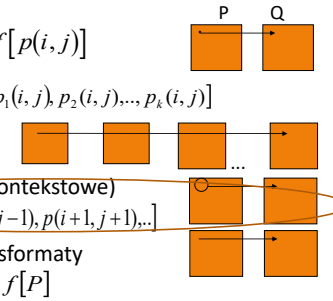
- Operacje punktowe (jednopunktowe):

Jednoargumentowe

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

Wieloargumentowe

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



- 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

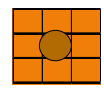
```
For  i=0 to X do:
        Begin.
         For  j=0 to Y do
                Begin.
                fnew(i,j):= F(f(i,j))
                End.
        End.
```

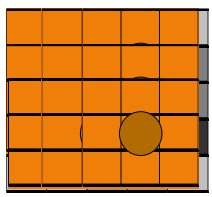
F(f1(i,j),f2(i,j))

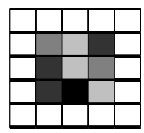
OPERACJE SĄSIEDZTWA

Są to operacje, w których na wartość zadanego piksela obrazu wynikowego ***q*** o współrzędnych *(i,j)* mają wpływ wartości pikseli **pewnego otoczenia** piksela obrazu pierwotnego ***p*** o współrzędnych *(i,j)*

Proces liczenia operacji sąsiedztwa







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.
```

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

- **OpenCV** (*Open source computer vision*) to biblioteka funkcji programowania w dziedzinie zwanej wizja komputerową (computer vision) zawierające zoptymalizowane algorytmy wykorzystywane w wizji komputerowej czasu rzeczywistego. Powstała w firmie Intel, przeszła na własność Willow Garage a potem firmy Itseez (która została wykupiona przez Intel).
- Jest dostosowana do **różnych języków programowanie (cross-platform)**
- **Jest darmowa** według licencji: **open-source BSD license**.

C++, Python, Java and MATLAB pod systemy operacyjne: Windows, Linux, Android and Mac OS.
Zawiera ponad 500 procedur i ponad 5000 funkcji.
Jest napisana w C++.

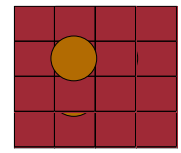
Laboratorium nr. 3

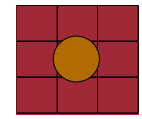
Zadanie 2

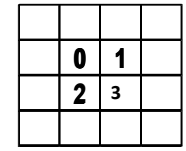
Opracowanie algorytmu i uruchomienie aplikacji realizującej uniwersalną operację medianową opartą na otoczeniu 3x3, 5x5, 7x7, 9x9 zadawanym w sposób interaktywny (wybór z list, przesuwanie baru lub wpisanie w przygotowane pole). Zastosować powyższych metod uzupełniania brzegowych pikselach obrazu, dając użytkownikowi możliwość wyboru, jak w zadaniu 1 z lab 3.

.

Mediana i pozostałe filtry statystyczne







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

0, 0, 0, 0, 0, 1, 1, 2, 3 mediana=0, min=0, max=3, najbardziej prawdopodobna=0

0, 0, 0, 0, 1, 2, 3, 3, 3 mediana=1, min=0, max=3, najbardziej prawdopodobna=0

0, 0, 1, 1, 2, 2, 3, 3, 3 mediana=2, min=0, max=3, najbardziej prawdopodobna=3

0, 1, 1, 2, 3, 3, 3, 3,3 mediana=3, min=0, max=3, najbardziej prawdopodobna=3

8

Filtracja medianowa

scr – obraz źródłowy;
ksize – rozmiar otoczenia mediany
dst – obraz wynikowy

- Python:
  cv2.medianBlur(src, ksize[, dst]) → dst
- C++
  void medianBlur(InputArray **src**, OutputArray **dst**, int **ksize**)
- JS
  Cv.MedianBlur(src, dst, ksize);
- Java
  Imgproc.medianBlur(src, dst, 5);

Funkcja działa na obrazach monochromatycznych i kolorowych

Laboratorium nr. 3

Zadanie 1

Implementacji detekcji krawędzi operatorami opartymi na maskach Sobela i Prewitta oraz operatorem Cannyego. Implementacja dla obrazów monochromatycznych.

Metoda specjalnego gradientu i konstrukcji operatorów krawędziujących

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.

Operatory proste: **Robertsa**, **Sobela**, **Prewitta,**
Operator złożony: **Canny**.

11

Maski konwolucyjne do konstrukcji operatorów krawędziowych

Roberts

1	0
0	-1

G_x

0	-1
1	0

G_y

Sobel:

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

G_x

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

G_y

Prewitt:

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

G_x

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

G_y

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

12

Metoda Robertsa

f_0	f_1	f_2
f_3	f_4	f_5
f_6	f_7	f_8

(i,j)

$$R(i,j) = \sqrt{(f_4 - f_8)^2 + (f_7 - f_5)^2}; \qquad \alpha = -\frac{\pi}{4} + tg^{-1}\left(\frac{f_7 - f_5}{f_4 - f_8}\right)$$

R(i,j) - specjalny gradient w punkcie (i,j)
α - kierunek gradientu intensywności.

Metoda Sobela

dwie składowe gradientu:

$$S_x = (f_2 + 2f_5 + f_8) - (f_0 + 2f_3 + f_6)$$
$$S_y = (f_6 + 2f_7 + f_8) - (f_0 + 2f_1 + f_2)$$
$$S(x,y) = \sqrt{S_x^2 + S_y^2}$$

13

Filtry kierunkowe Sobela

Wschód				Południowy wschód				Południe				Południowy zachód		
-1	0	1		-2	-1	0		-1	-2	-1		0	-2	-1
-2	0	2		-1	0	1		0	0	0		1	0	-1
-1	0	1		0	1	2		1	2	1		1	2	0
E				SE				S				SW		

W				NW				N				NE		
1	0	-1		2	1	0		1	2	1		0	1	2
2	0	-2		1	0	-1		0	0	0		-1	0	1
1	0	-1		0	-1	-2		-1	-2	-1		-2	-1	0
Zachód				Północny zachód				Północ (N)				Północny wschód		

14

Operatory krawędziowania

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

dokładnej $G = \sqrt{Gx^2 + Gy^2}$

przybliżonej $G = |Gx| + |Gy|$

Robertsa

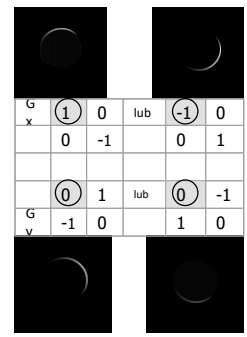
Gx	1	0	lub	-1	0
	0	-1		0	1
	0	1	lub	0	-1
Gy	-1	0		1	0

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

15

Filtry i operatory Robertsa



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

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

16

Operator Sobela

dx – rząd pochodne po x
dy – rząd pochodne po y

Python:

```
dst =   cv.Sobel(src, ddepth, dx, dy[, dst[, ksize[, scale[, delta[,
                 borderType]]]]]
                 )
```

C++

```
void cv::Sobel  (InputArray     src,
                 OutputArray    dst,
                 int            ddepth,
                 int            dx,
                 int            dy,
                 int            ksize = 3,
                 double         scale = 1,
                 double         delta = 0,
                 int            borderType = BORDER_DEFAULT
                 )
```

Operator oparty na maskach Prewitta

Filtry kierunkowe Prewitta

Wschód				Południowy wschód				Południe				Południowy zachód		
-1	0	1		-1	-1	0		-1	-1	-1		0	-1	-1
-1	0	1		-1	0	1		0	0	0		1	0	-1
-1	0	1		0	1	1		1	1	1		1	1	0
E				SE				S				SW		

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

W				NW				N				NE		
1	0	-1		1	1	0		1	1	1		0	1	1
1	0	-1		1	0	-1		0	0	0		-1	0	1
1	0	-1		0	-1	-1		-1	-1	-1		-1	-1	0
Zachód				Północny zachód				Północ (N)				Północny wschód		

Ten operator należy opracować indywidualnie

18

Operator Canny-ego

Wielostopniowy algorytm detekcji krawędzi zaproponowany przez twórcę teorii *Computational theory of edge detection* **John F. Canny** w 1986 r. Jego działanie oparte jest o znane detektory krawędzi (Robertsa, Sobela / Prewitta) i **progowanie i śledzenia z histerezą**, które optymalizuje wynik wyrzucając krawędzie rozmyte, nachylone pod kątami niewiele różniącymi się od wcześniej wykrytych, zapobiega przerwaniu krawędzi w miejscach utraty kontrastu, itp..

1986. *A computational approach to edge detection*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, 1986, pp. 679–698

Etapy działania operatora Canny-ego

- Zamiana obrazu kolorowego do monochromatycznego w odcieniach szarości
- Rozmycie filtrem gaussowskim o zadanym parametrze: sigma (δ)
- Obliczenie gradientu Sobele/Prewitta i kierunku gradientu według Robertsa
- Lokalna selekcja i wytłumienie gradientów mniejszych niż maksymalne w obrazie gradientu w celu otrzymania cienkiej linii
- Dokonanie podwójnego progowania według zadanych parametrów Tmin (i Tmax):
  - <Tmin - brak krawędzi,
  - >Tmax – silne krawędzie
  - >=Tmin i <=Tmax – krawędzie słabe
- Wykonanie śledzenia krawędzi opartego na histerezie w celu ich uciąglenia: słabe krawędzi będące przedłużeniem silnych są dołączane, pozostałe są oznaczane do czyszczenia
- Wyczyszczenie krawędzi

Implementacje

threshold1 – Tmin
threshold2 – Tmax
L2gradient – true
operator ze wzoru dokładnego;
false
operator ze wzoru przybliżonego

Język C

```
void cv::Canny(InputArray  image,
               OutputArray edges,
               double      threshold1,
               double      threshold2,
               int         kernelSize = 3,
               bool        L2gradient = false
               )
```

- blur(src_gray, detected_edges, Size(3,3));
- Canny(detected_edges, detected_edges, lowThreshold, lowThreshold*ratio, kernel_size);
  - *highThreshold*: rekomendacja autora 3* lowThreshold
  - *kernelsize*: 3 (rozmiar filtów Sobel/Prewitta)

Python

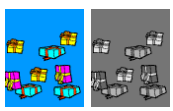
```
Edges = cv.Canny(image, threshold1, threshold2[, edges[, karneleSize[, L2gradient]]])
```

19

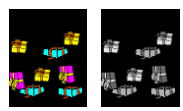
Segmentacja obrazu

Segmentacja

Wyodrębnienie spośród wybranych fragmentów tych, które stanowią obiekt zainteresowania (ang. ROI) ze względu na cel analizy obrazu.



Maska/Mapa



Najbardziej skomplikowane algorytmy

23

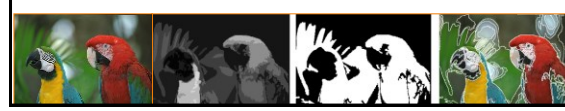
Segmentacja

- Segmentacja to podział obrazu na rozłączne (nienakładające się) fragmenty.
- Segmentacja jest powiązana z semantyką (znaczeniem i rozumieniem) obrazu.

bywa rozumiana dwojako:

– Jako podział na jednorodne rejony, które składają się na znaną hierarchię lub strukturę

– Jako podział na to, co nas interesuje z punktu widzenia celu przetwarzania, pozostałe nieinteresujące obiekty i tło



24

Klasyfikacja metod segmentacji

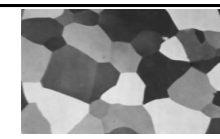
Segmentacja może być zarówno operacją kontekstową (sąsiedztwa) jak i niekontekstową (punktową), ale najczęściej jest kombinacją metod kontekstowych i punktowych.

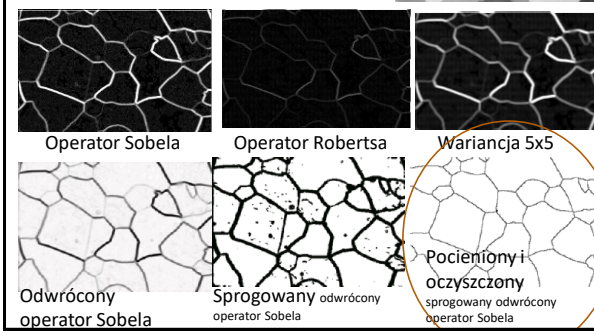
Jeśli metoda:

- Ignoruje zależności między pikslami i klasyfikuje je na podstawie globalnej cechy, np. wartości poziomu szarości – progowania, to jest metodą punktową
- Wykorzystuje zależności między pikslami, np.: podobieństwo wartości poziomu szarości – „dziel i łącz", to jest metodą kontekstową

25

Segmentacja to proces wieloetapowy





26

Rodzaje segmentacji

(około 1000 algorytmów segmentacji)

- Poszukiwanie nieciągłości poziomów szarości, koloru, tekstury, własności spektralnych tekstury, czyli poszukiwanie krawędzi, a w konsekwencji wskazanie wnętrza obiektu (np. z wykorzystaniem histogramu dwuwymiarowego),
- Maksymalne obszary wykazujące podobieństwo w kolorze, odcieniu szarości, teksturze (progowanie, klasteryzacja, analiza tekstury, itp.)
- Wododziały (ang. *watershed transform)* i inne operacje morfologii matematycznej,
- Podziały przeszukujące obszar obrazu (split and merge, drzewa czwórkowe, itp)
- Dopasowywanie konturów np. metoda aktywnego konturu, (ang. *Active contour*), snake czy Level-set
- Metody sztucznej inteligencji oparte na głębokich sieciach neuronowych (Unet, AlexNet, CNN)

Uwaga

W wyniku stosowania obszarowych metod segmentacji uzyskuje się **zawsze** obszary zamknięte (granice obszarów są ciągłe). Jest to zaleta w porównaniu np. z metodami *detekcji krawędzi*, które na ogół **nie zapewniają** ciągłości wykrytych krawędzi.

27

Laboratorium 5

Zadanie1

Opracować algorytm i uruchomić aplikację realizującą segmentację obrazów 3 metodami:

- Ręcznie (interakcyjnie) wyznaczony próg (*thresholding*)
- Progowanie adaptacyjne (*adaptive thresholding*)
- Progowanie metodą Otsu

28

Ogólna teoria progowania

$$T=T(x, y, p(x, y), n(x, y))$$

x i y - współrzędne w obrazie;
p(x,y) – poziom szarości lub kolor w punkcie x, y
n(x,y) – jakieś lokalne własności punku x,y względem jego sąsiedztwa.

Globalne progowanie: T=T(p(x, y))

Lokalne progowanie: T=T(p(x, y), n(x, y))

progowanie zależne od tego czy w sąsiedztwie punktu występuje krawędź (skok jasności) czy też obszar jednorodny albo zależy od lokalnego kontrastu.

Adaptacyjne (dynamiczne) progowanie:

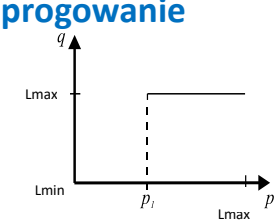
$$T=T(x, y, p(x, y))$$

progowanie, w którym próg jest inny w różnych częściach obrazu – zależnie od występowania cieni, nierównomierności oświetlenia, itp..

Podstawowe progowanie

Operacja redukuje wartości poziomów szarości do dwóch wartości: Lmax i Lmin.

Ma jeden arbitralnie wybierany przez użytkownika parametr zwany progiem (*ang*. **threshold values** (p1); *pol.* próg) – czyli graniczna wartość poziomu jasności powyżej której przypisywana jest Lmax (*ang.* white; *pol.* biel), a dla niej i wszystkich wartości poniżej przypisywana jest wartość Lmin (ang. black; *pol.* czerń).



$$q = \begin{cases} L_{\min} & \text{dla } p \le p_1 \\ L_{\max} & \text{dla } p > p_1 \end{cases}$$

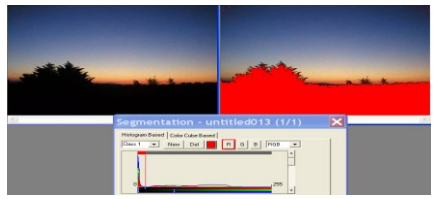
30

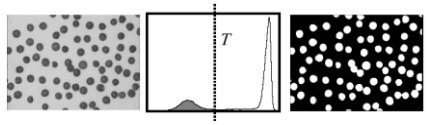
Przykład progowania

Progowanie dobrze segmentuje tylko wtedy, gdy:

-istnieje rozdzielność poziomów szarości lub kolorów obiektu i tła

-gdy „dolinka" progu jest głęboka (najlepiej zerowa)





Inne warianty operacji progowania

- Progowanie z binaryzacją: Lmin=0 and Lmax=1

$$q = \begin{cases} 0 & \text{dla } p \le p_1 \\ 1 & \text{dla } p > p_1 \end{cases}$$

- Progowanie odwrotne

$$q = \begin{cases} L_{\min} & \text{dla } p \ge p_1 \\ L_{\max} & \text{dla } p < p_1 \end{cases}$$

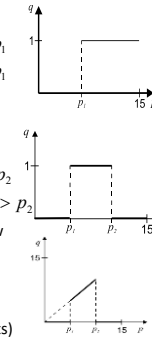
- Progowanie z dwoma progami (dwa parametr: p1 and p2)

$$q = \begin{cases} L_{\max} & \text{dla } p_1 \le p \le p_2 \\ L_{\min} & \text{dla } p < p_1 \text{ lub } p > p_2 \end{cases}$$

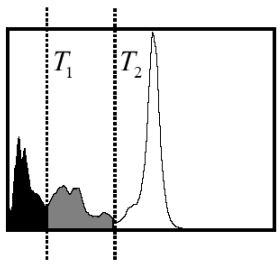
- Progowanie z dwoma progami z pozostawieniem wartości w przedziale

$$q = \begin{cases} p & \text{dla } p_1 \le p \le p_2 \\ 0 & \text{dla } p < p_1,\ p > p_2 \end{cases}$$

- Adaptive thresholding (p1=f(neighborhood(i,j)) i,j-coordinats)
- Recursive thresholding
- Hierarchical thresholding (pyramidal, scalable)



Progowanie z dwoma progami

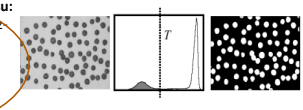


33

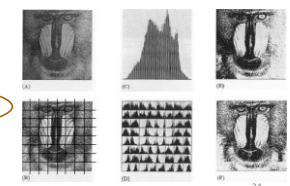
Wybór progu do operacji progowania

- **Globalny ustalony próg:**
  - Arbitralnie wyznaczony przez użytkownika
  - Automatycznie wyszukiwana w histogramie najniższa wartość w dolince
  - Próg globalny wyznaczany metodą **Otsu:**

Metoda sprawdza wartość błędu wewnątrz i/lub międzygrupowego we wszystkich możliwych położeniach progu 0 <t <225 i wybiera ten, który minimalizuje błąd wewnątrzgrupowy lub maksymalizuje błąd międzygrupowy.



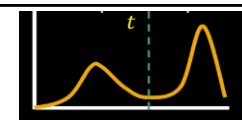
- **Próg o zmiennej wartości zależnej od położenie** na obrazie:
  - **adaptacyjna** (zależny od wartości jasności w sąsiedztwie)
  - **dynamiczny** (zależny od pozycji w obrazie)



34

Jak automatycznie wyznaczyć wartość progu

Progowanie z progiem wyznaczonym metodą Otsu



Metoda wyznaczania optymalnego progu zaproponowana przez Nobuyuki Otsu w 1979 roku – algorytm oparty na histogramie.

Algorytm sprawdza każdy z możliwych progów 0<t<225 poszukując takiego, który daje minimalną wariancję wewnątrz grupową lub maksymalizuje wariancją międzygrupową

Minimalizacja wewnątrzklasowa – odpowiada minimalzacji wariancji wewnątrzkalsowejj

$$\sigma^2 = \underbrace{\sigma_w^2(t)}_{\text{Within-class}} + \underbrace{q_1(t)[1-q_1(t)][\mu_1(t)-\mu_2(t)]^2}_{\text{Between-class}}$$

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

$$q_1(t) = \sum_{i=1}^{t} P(i) \qquad q_2(t) = \sum_{i=t+1}^{I} P(i)$$

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)} \qquad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{q_2(t)}$$

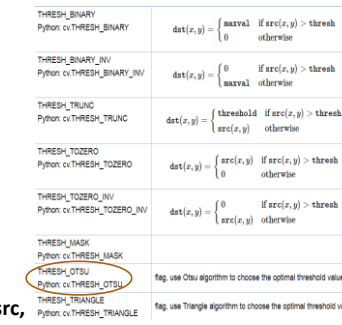
$$\sigma_1^2(t) = \sum_{i=1}^{t} [i-\mu_1(t)]^2 \frac{P(i)}{q_1(t)} \qquad \sigma_2^2(t) = \sum_{i=t+1}^{I} [i-\mu_2(t)]^2 \frac{P(i)}{q_2(t)}$$

OpenCV

- threshold()

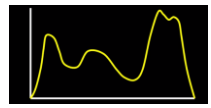
Double cv::threshold(
InputArray src,
OutputArray dst,
double thresh,
double maxval,
int type)

Python:
retval, dst=cv.threshold(src, thresh, maxval, type[, dst])



CV.THRESH_BINARY | CV.THRESH_OTSU

Progowanie z dwoma progami wyznaczonymi metodą Otsu

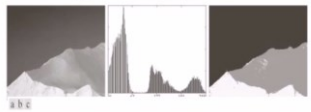


Multiple Thresholds

- Otsu's method can be generalized to find multiple thresholds
- In the case of K classes, we maximize the between-class variance

$$\sigma_B^2 = \sum_{k=1}^{K} P_k(m_k - m_G)^2$$

where $P_k = \sum_{i \in C_k} p_i \qquad m_k = \frac{1}{P_k}\sum_{i \in C_k} i\,p_i$



a b c

FIGURE 10.45 (a) Image of iceberg. (b) Histogram. (c) Image segmented into three regions using dual Otsu thresholds. (Original image courtesy of NOAA.)

Progowanie adaptacyjne

a b c

FIGURE 10.49 (a) Text image corrupted by spot shading. (b) Result of global thresholding using Otsu's method. (c) Result of local thresholding using moving averages.

To progowanie za zmiennym progiem wybieranym dla każdego otocznie oddzielnie na podstawie:

1. Średniej z otoczenia
2. Średniej ważonej według współczynników dobranych z zastosowaniem filtru gausowskiego

OpenCV

adaptiveThreshold (src, dst, maxValue, adaptiveMethod, thresholdType, blockSize, C)

src – An object of the class **Mat** representing the source (input) image.
dst – An object of the class **Mat** representing the destination (output) image.
maxValue – A variable of double type representing the value that is to be given if pixel value is more than the threshold value.
adaptiveMethod – A variable of integer the type representing the adaptive method to be used. This will be either of the following two values
ADAPTIVE_THRESH_MEAN_C – threshold value is the mean of neighborhood area.
ADAPTIVE_THRESH_GAUSSIAN_C – threshold value is the weighted sum of neighborhood values where weights are a Gaussian window.
thresholdType – A variable of integer type representing the type of threshold to be used.
blockSize – A variable of the integer type representing size of the pixelneighborhood used to calculate the threshold value.
C – A variable of double type representing the constant used in the both methods (subtracted from the mean or weighted mean).

42

Pomoc - przykłady

- https://docs.opencv.org/3.4/dc/dd3/tutorial_gausian_median_blur_bilateral_filter.html
- https://docs.opencv.org/3.4/db/d8e/tutorial_threshold.html
- https://docs.opencv.org/3.4/d4/dbd/tutorial_filter_2d.html
- https://docs.opencv.org/3.4/d2/d2c/tutorial_sobel_derivatives.html
- https://docs.opencv.org/3.4/d5/db5/tutorial_laplace_operator.html
- https://docs.opencv.org/3.4/da/d5c/tutorial_canny_detector.html
- https://opencv-java-tutorials.readthedocs.io/en/latest/index.html
- https://docs.opencv.org/4.5.3/d4/d86/group__imgproc__filter.html

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**).