

## Algorytmy Przetwarzania Obrazów

### Operacje na obrazach (II)

WYKŁAD 2  
Dla studiów stacjonarnych 2022/2023

Dr hab. Anna Korzyńska, prof. IBIB PAN

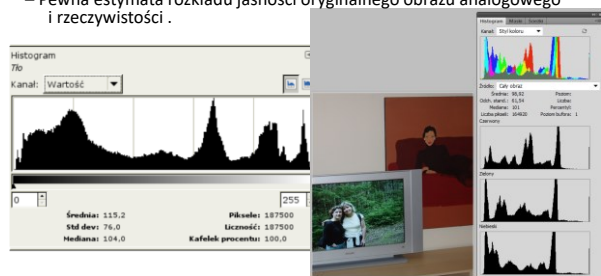
## Powtórzenie

### Manipulacje histogramem

### Histogram definicja

Histogram to wykres słupkowy przedstawiający ilość pikseli o każdej potencjalnej wartości występującej w obrazie.

- Statystyka odzwierciedlająca rozkład jasności punktów w obrazie.
- Pewna estymata rozkładu jasności oryginalnego obrazu analogowego i rzeczywistości.



### Operacje na histogramach

Obrazy bardzo często zawierają elementy, które są trudne do zauważenia głównie dlatego, że obiekty są mało zróżnicowane w stosunku do otoczenia.

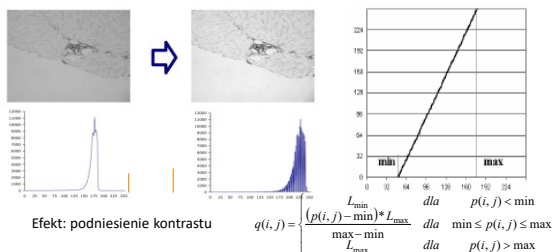
Operacje na histogramach:

1. Rozciąganie: z zakresu p1-p2 do zakresu q3-q4, (w szczególności gdy q3=0, q4=Lmax – rozciąganie do pełnego dostępnego zakresu poziomów szarości z i bez przepięnienia na poziomie 5% ilości pikseli).
2. Wyrównanie selektywne typu equalizacja

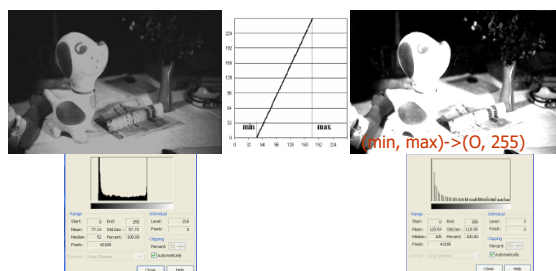
### Rozciąganie histogramu

Obrazy w których nieefektywnie wykorzystujemy pełną dynamikę odcieni i barw dostępną w danym zakresie

$$I(x, y) = I(x, y) - \min - \frac{255}{\max - \min}$$



### Liniowe rozciąganie histogramu - przykład



Dane: P(z)-obraz; h histogram, zakres rozciągnięcia Lmin=0 i Lmax=;

```
min=0; max=L; sem_min=false;
sem_max=false; hr(Z) i F(z) -
wypełnić zerami
```

For Z↑ do:  
Begin.  
IF Z<min then hr(Z)=lmin  
else IF Z>max then hr(Z)=lmax  
else hr(Z)=zaokrągl do  
całkowitych(  
((Z-min)\*Lmax)/  
(max-min)  
)  
End.  
For wszystkich elementów f obrazu do:  
Begin.  
F(z)=hr(P(z))  
End.

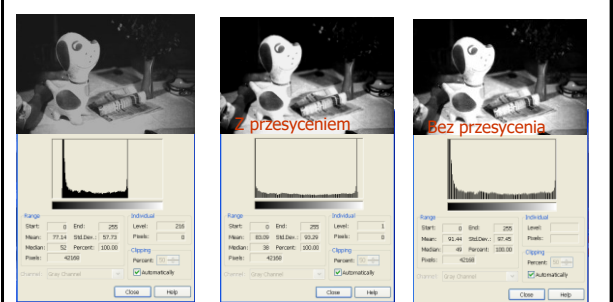
End.

For Z↓ do:  
Begin.  
If (sem\_max=false) then  
if h(Z) ≠ 0 then  
Begin.  
sem\_min=true  
min=Z  
End.  
End.  
If (sem\_min=false) then  
if h(Z) ≠ 0 then  
Begin.  
sem\_max=true  
max=Z  
End.  
End.

End.


$g(i, j) = \begin{cases} \frac{L_{\min}}{p(i, j) - \min} * L_{\max} & \text{dla } p(i, j) < \min \\ \frac{\max - \min}{L_{\max}} & \text{dla } \min \leq p(i, j) \leq \max \\ \frac{L_{\max}}{p(i, j) - \max} & \text{dla } p(i, j) > \max \end{cases}$

### Liniowe rozciąganie histogramu z i bez przesycenia



5

### Ni Contrast Enhancement



### Wyrównywanie histogramu przez equlzację

Do takiego wyrównanie histogramu wykorzystujemy tablicę LUT i dystrybuantę empiryczną czyli dystrybuantę prawdopodobieństwa wylosowanie piksela o określonej wartości jasności (odpowiednik histogramu skumulowanego)



### Przykład - wyrównania histogramu

Il.w = 4; il.k = 5;  
liczba pikseli = 20  
lmin = 0; lmax = 5;  
M = 6

histogram

2	5	0	3	9	1
0	1	2	3	4	5

dystrybuanta histogramu

2	7	7	10	19	20
0	1	2	3	4	5

D[i]/rozmiar obrazu=dystrybuanta prawdopodobieństwa wystąpienia c=i

2/20	7/20	7/20	10/20	19/20	20/20
0	1	2	3	4	5

wyliczamy LUT(i) - przekształcenie

0	1	2	3	4	5
0	1	2	3	4	5

histogram

2	5	3	0	9	1
0	1	2	3	4	5

$D[i] = (H_0 + H_1 + \dots + H_i) / \text{sum}$   
 $D_0$  – pierwsza niezerowa wartość = 2/20  
 $LUT[i] = ((D[i] - D_0) / (1 - D_0)) * (M - 1)$   
 $= ((20 * D[i] / 20 - 2 / 20) * 20 / 18)$

### Operacje na obrazach

## Operacje na obrazach

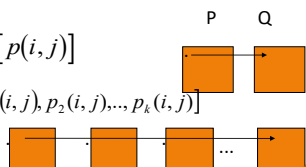
### ➤ Operacje punktowe (jednopunktowe):

Jednoargumentowe

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

Wieloargumentowe

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

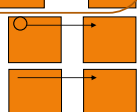


### ➤ Operacje sąsiedztwa (kontekstowe)

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

### ➤ Operacje globalne transformaty

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



## Operacje punktowe (lokalne, jednopunktowe)

Oprogramowanie:

- negacji,
- progowania,
- progowania z zachowaniem poziomów szarości,
- Progowanie z dwoma progami

## Operacje punktowe



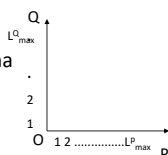
Definiowane przez:

- Definicję funkcji; z jawnie postawionymi warunkami logicznymi

- Wykres funkcji we współrzędnych OXY; na osi OX są **wszystkie potencjalne** wartości poziomów szarości obrazu pierwotnego P, a na OY obrazu po przekształceniu Q;

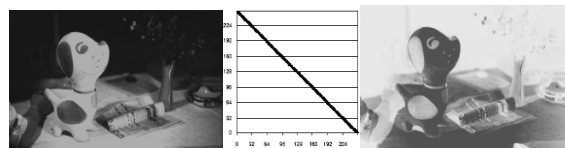
- Tablica przekodowań (LUT – Look Up Table).

0	1	2	....	$L_{p\_max}$	P
					Q



## Negacja

Negatyw obrazu



$$q(i, j) = L_{max} - p(i, j)$$

$$q(i, j) = \text{NOT } p(i, j)$$

Do prezentacji informacji zawartej w ciemnych tonach (cieniach) jeśli jasne tony są nieistotne

16

Dane: P(z)-obraz;

hr(Z) i F(z) - wypełnić zerami

For Z↑ do:

Begin.

hr(Z) = Lmax-Z

End.

For wszystkich elementów f obrazu do:

Begin.

F(z) = hr(P(z))

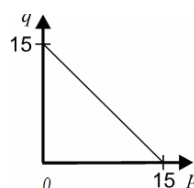
End.

## Operacja odwrotności (negacji)

$$q(i, j) = L_{max} - p(i, j) \text{ dla } L_{min} \leq p \leq L_{max}$$

Dla  $L_{min} = 0$ ,  $L_{max} = 15$  (czyli  $M=16$ ):  $q(i, j) = 15 - p(i, j)$

[q]



0	0	15	15	13
2	2	0	15	15
15	15	8	1	1
15	14	13	12	11
0	1	2	3	4

18

## Progowanie

Jest to taka wersja operacji zmniejszenia ilości poziomów szarości do dwóch, dla której istnieje możliwość arbitralnego wyboru wartości progu ( $p_1$ ) czyli szarości granicznej, od której przyporządkowujemy wyższy poziom szarości (najczęściej biel) oraz dla której i poniżej której przyporządkowujemy niższy próg szarości (najczęściej czerni).

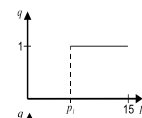
$$q = \begin{cases} L_{\min} & \text{dla } p \leq p_1 \\ L_{\max} & \text{dla } p > p_1 \end{cases}$$

19

## Różne typy progowania

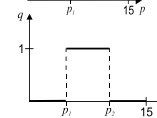
- Progowanie z pojedynczym progiem segmentacji

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



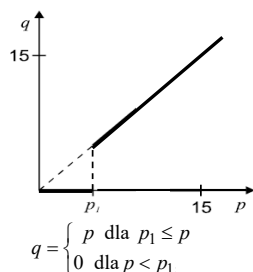
- Progowanie przedziałami

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



20

## Operacja progowania z jednym progiem i zachowaniem poziomów szarości



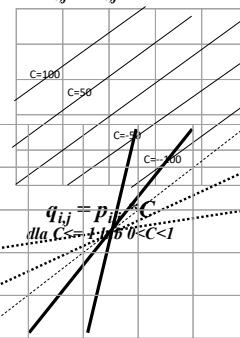
## Laboratorium 3

### Zadanie 1

Opracowanie algorytmu i uruchomienie funkcjonalności realizującej operacje punktowe wieloargumentowych:

- dodawania obrazów, z i bez wysycenia
- dodawanie, dzielenie i mnożenie obrazów przez liczbę całkowitą, z i bez wysycenia
- liczenia różnicy bezwzględnej obrazów.

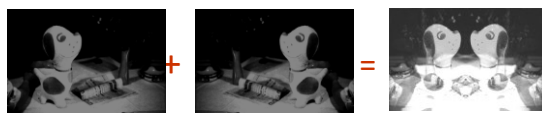
$$q_{ij} = p_{ij} \pm C$$



## Dodawanie obrazów

Przekroczenie zakresów poziomów szarości regulujemy:

- Wagami,
- Funkcją modulo,
- Skalowaniem wyniku – liniowe



Zastosowanie do:

- łączenia masek
- efekty nałożenia obiektów i przenikania

23

## Mnożenie obrazów

Przekroczenie zakresów poziomów szarości regulujemy:

- Wagami,
- Funkcją modulo,
- Skalowaniem wyniku

Zastosowanie:

- Kompozycje artystyczne
- Maskowanie binarną maską
- Jako element operacji rekonstrukcji obrazów



24

## Odejmowanie obrazów

- różnica



- różnica bezwzględna



- Przekroczenie zakresów poziomów szarości regulujemy:
- Wagami,
- Funkcją modulo,
- Skalowaniem wyniku - liniowe

Obraz po liniowym rozciągnięciu histogramu do podwojenia zakresu

Zastosowanie do:

1. Pokazania różnicy między obrazami, zwłaszcza w przypadku, gdy porównywane obrazy są nierozróżnialne wzrokowo
2. Angiografii różnicowej

25

## Laboratorium 3

### Zadanie 2

Opracowanie algorytmu i uruchomienie funkcjonalności realizującej operacje logiczne na obrazach monochromatycznych i binarnych:

- not
- and
- or
- xor.

Przy okazji proszę umożliwić użytkownikowi zmianę obrazów z maski binarnej na maskę zapisaną na 8 bitach i na odwrót (jeśli w wybranym środowisku i języku jest to możliwe). Proszę pamiętać o sprawdzeniu zgodności typów i rozmiarów obrazów stanowiących operandy.

Poziom jasności  $n$  jest zapisany w kodzie dwójkowym jako kombinacja ośmiu 0 i 1:

Czerń 00000000

Biel 11111111

127 01000001

Operacje logiczne:

NOT NOT(1)=0; NOT(0)=1

AND 1 AND 1=1; 0 AND 0=0; 1 AND 0=0; 0 AND 1=0

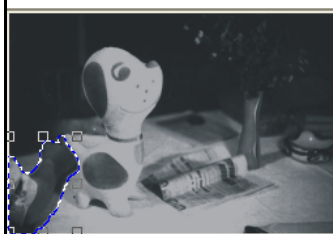
OR 1 OR 1=1; 0 OR 0=0; 1 OR 0=1; 0 OR 1=1

XOR 1 XOR 1=0; 0 XOR 0=0; 1 XOR 0=1; 0 XOR 1=1

W operacjach jednopunktowych dwuargumentowych logicznych na obrazach działania prowadzone są na odpowiednich pikselach obrazów stanowiących argumenty danej operacji.

W szczególności działania prowadzone są na bitach o tej samej wadze.

## Maska



28

## Operacja logiczna jednoargumentowa na obrazie NOT

$$q(i, j) = \text{NOT } p(i, j)$$

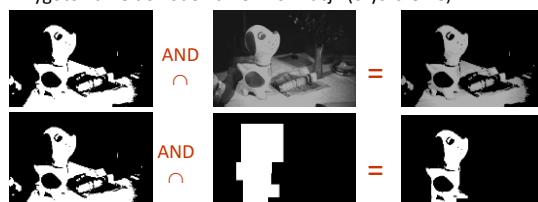
Równoważna negacji



29

## Operacja logiczna AND na obrazach

- Maskowanie, czyli selekcja fragmentów obrazów, zwanych ROI, (ang. *region of interest*) na podstawie binarnej maski
- Zacieśnianie maski
- Przygotowanie do kodowania informacji (czyszczenie)



30

## Operacja logiczna OR na obrazach

- Rozszerzanie maski
- Nakładanie informacji szyfrowanej



31

## Operacja logiczna XOR na obrazach

- Działa podobnie jak operacja OR



32

## Operacje na obrazach

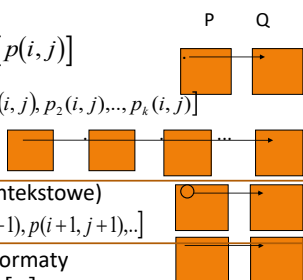
### ➤ Operacje punktowe (jednopunktowe):

Jednoargumentowe

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

Wieloargumentowe

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



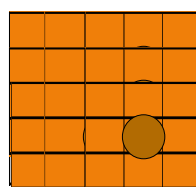
### ➤ Operacje sąsiedztwa (kontekstowe)

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

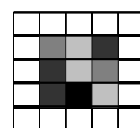
### ➤ Operacje globalne transformaty

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

## Proces liczenia operacji sąsiedztwa



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



Do operacji sąsiedztwa  $E$  dla maski  $3 \times 3$

For  $i=2$  to  $X-2$  do:

Begin.

For  $j=2$  to  $X-2$  do

Begin.

$f_{new}(i, j) := E(f(i-1, j-1), f(i-1, j), f(i-1, j+1), f(i, j-1), f(i, j), f(i, j+1), f(i+1, j-1), f(i+1, j), f(i+1, j+1))$

End.

End.

34

## Laboratorium 4

Proszę dołączyć bibliotekę OpenCV i korzystać z niej przygotowując poszczególne funkcjonalności.

## Biblioteka OpenCV

## Biblioteka OpenCV

- **OpenCV** (*Open source computer vision*) to biblioteka funkcji przystosowanych do zastosowań w komputerowej wizji, czyli do wykorzystania przy tworzeniu oprogramowania pracującego w trybie czasu rzeczywistego.
- Powstała dla firmy Intel, przeszła w ręce firmy Itseez którą następnie wchłonął Intel.
- Jest biblioteką darmową pod licencją **open-source BSD license**
- Zawiera ponad 500 algorytmów i około 5000 funkcji – bardzo efektywnych i szybkich
- Jest napisana w C++
- Obsługuje wiele języków programowania pod różnymi systemami operacyjnymi (C++, Python, Java and MATLAB oraz może pracować pod Windows, Linux, Android i Mac OS.).



## Implementacja liczenia histogramu w OpenC

```
#include <cv.h>
#include <highgui.h>

#pragma comment(lib, "cv210")
#pragma comment(lib, "highgui210")

IplImage* image = 0;
IplImage* imgHistogram = 0;
IplImage* gray = 0;

CvHistogram* hist;

int main(){
    image = cvLoadImage("xxxxxx.jpg",1);

    //określenie rozmiaru histogramu (ilości koszy)
    int size[] = {256};

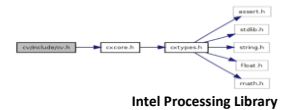
    //wartość kosza i jego znormalizowana wartość
    float val;
    int norm;

    //zakres jasności
    float range[] = { 0, 255 };
    float* ranges[] = { range };

    //wartość minimalna i maksymalna histogramu
    float min_val = 0, max_val = 0;

    //konwersja wejściowego obrazu kolorowego do 8 bitowego w skali szarości
    gray = cvCreateImage( cvGetSize(image), 8, 1 );
    cvCvtColor( image, gray, CV_BGR2GRAY );

    //tworzenie okien do pokazania rezultatu
```



Intel Processing Library

```
cvNamedWindow("original",1);
cvNamedWindow("gray",1);
cvNamedWindow("histogram",1);

//obrazy z których zostanie obliczony histogram
IplImage* images[] = { gray }; //tworzenie okien do pokazania rezultatu

// przygotowanie pustego histogramu
hist = cvCreateHist( 1, size, CV_HIST_ARRAY, ranges,1);
// obliczenie histogramu z obrazów
cvCalcHist( images, hist, 0, NULL);
// znalezienie minimalnej i maksymalnej wartości histo
cvGetMinMaxHistValue( hist, &min_val, &max_val);

// tworzenie 8bitowego obrazu do pokazania histogramu
imgHistogram = cvCreateImage(cvSize(256, 50),8,1);
// wypełnienie go białym kolorem
cvRectangle(imgHistogram, cvPoint(0,0), cvPoint(256,50),
CV_RGB(255,255,255),CV_FILLED);

//rysowanie histogramu jasności pikseli znormalizowanego od 0 do 50
for(int i=0; i < 256; i++){
    val = cvQueryHistValue_1D( hist, i);
    norm = cvRound(val*50/max_val);
    cvLine(imgHistogram,cvPoint(i,50), cvPoint(i,50-norm),CV_RGB(0,0,0));
}

//pokazanie okien z rezultatem
cvShowImage( "original", image );
cvShowImage( "gray", gray );
cvShowImage( "histogram", imgHistogram );
// zapis obrazu histogramu do pliku
cvSaveImage("histogram.jpg",gray);
cvWaitKey();

return 0;
}
```

```
* @function EqualizeHist_Demo.cpp
* @brief Demo code for equalizeHist function
* @author OpenCV team
```

#include "opencv2/imgcodecs.hpp"

#include "opencv2/highgui.hpp"

#include "opencv2/imgproc.hpp"

#include &lt;iostream&gt;

using namespace cv;

using namespace std;

/\*\*

\* @function main

\*/

int main( int argc, char\*\* argv )

{

// [Load image]

CommandLineParser parser( argc, argv, "@input | xxxx| input image" );

Mat src = imread( samples::findFile( parser.get&lt;String&gt;("input") ),

IMREAD\_COLOR );

if( src.empty() )

{

cout &lt;&lt; "Could not open or find the image!\n" &lt;&lt; endl;

cout &lt;&lt; "Usage: " &lt;&lt; argv[0] &lt;&lt; " &lt;input image&gt;" &lt;&lt; endl;

return -1;

}

// [Load image]

// [Convert to grayscale]

cvtColor( src, src, COLOR\_BGR2GRAY );

// [Convert to grayscale]

// [Apply Histogram Equalization]

Mat dst;

equalizeHist( src, dst );

// [Apply Histogram Equalization]

waitKey();

// [Display results]

imshow( "Source image", src );

imshow( "Equalized image", dst );

// [Wait until user exits the program]

waitKey();

// [Wait until user exits the program]

return 0;

OpenCV has a function to do this, **cv2.equalizeHist()**

## OpenCV dla 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

- Ładowanie biblioteki:

```
<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 dla 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);`

## OpenCV dla Java

- `import org.opencv.core.Core;`
- `import org.opencv.core.CvType;`
- `import org.opencv.core.Mat;`
- ulokowanie pliku `opencv-300.jar` w katalogu `\opencv\build\java`
- a biblioteki `opencv_java3xx.dll` library w katalogu: `\opencv\build\java\x64` (64-bitowy system) lub `\opencv\build\java\x86` (32-bitowy system).

## Laboratorium 4

### Zadanie 1

Opracowanie algorytmu i uruchomienie funkcjonalności realizującej operacje:

- wygładzania liniowego oparte na typowych maskach wygładzania (uśrednienie, uśrednienie z wagami, filtr gaussowski – przedstawione na wykładzie) przestawionych użytkownikowi jako maski do wyboru,

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

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

1	-2	1
-2	4	-2
1	-2	1

- wyostrzania liniowego oparte na 3 maskach laplasjanowych (podanych w wykładzie) przestawionych użytkownikowi maski do wyboru,

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

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

1	-2	1
-2	4	-2
1	-2	1

- kierunkowej detekcji krawędzi w oparciu o maski 8 kierunkowych masek Sobela (podstawowe 8 kierunków) przestawionych użytkownikowi do wyboru,

Wybór sposobu uzupełnienia marginesów/brzegów w operacjach sąsiedztwa według zasady wybranej spośród następujących zasad:  
wypełnienie ramki wybraną wartością stałą n narzuconą przez użytkownika:  
`BORDER_CONSTANT`  
wypełnienie wyniku wybraną wartością stałą n narzuconą przez użytkownika  
wyliczenie ramki według `BORDER_REFLECT`  
wyliczenie ramki według `BORDER_WRAP`

## Operacje wygładzania

$w_1$	$w_2$	$w_3$
$x-1, y-1$	$x-1, y$	$x-1, y+1$
$w_4$	$w_5$	$w_6$
$x, y-1$	$x, y$	$x, y+1$
$w_7$	$w_8$	$w_9$
$x+1, y-1$	$x+1, y$	$x+1, y+1$

↓ x

→ y

Podstawowe zadanie wygładzania: usuwanie zakłóceń z obrazu

Filtracja liniowa (metody *konwolucyjne*, tzn. uwzględniające pewne otoczenie przetwarzanego piksela):

$$g(x, y) = \sum_{k=1}^n w_k f_k(x, y)$$

$n$  - liczba punktów (pikseli) otoczenia wraz z pikselem przetwarzanym  
 $f(x, y)$  - wartość piksela o współrzędnych  $x, y$  obrazu pierwotnego  
 $g(x, y)$  - wartość piksela o współrzędnych  $x, y$  obrazu wynikowego  
 $w_k$  - waga  $k$ -tego piksela otoczenia

46

## Wygładzanie



Filtr jednokowy (o równych wagach)  
W openCV - normalized box filter

- 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

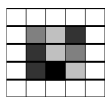
## Argumenty

Argumenty wejściowe:

- `src` – obraz wejściowy
- `ksize` – jądro przekształcenia
- `anchor` – punkt zaczepienia jądra przekształcenia
- `dst` – obraz wyjściowy
- `borderType` – określa postępowanie z pikselami brzegowymi



## Metody operacji na pikselach wchodzących w skład skrajnych kolumn i wierszy



1. Wartości pikseli są nieokreślone (xxxxxxxxxx)
2. Pozostawienie wartości pikseli bez zmian
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 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.

## 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' */
```

## Operacje wyostrzania i detekcji krawędzi

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

W wyostrzaniu stosuje się metody numeryczne  
aproksymujące pochodną.

Zadanie wyostrzania:

- podkreślenie na obrazie konturów obiektów
- podkreślenie na obrazie punktów informatywnych (np. wierzchołki dla wielokątów, zakończenia, skrzyżowania, rozgałęzienia linii dla rysunków technicznych, wykresów lub pisma).

**wyostrzania: wydobyć i uwypuklenie krawędzi obiektu.**

51

**Detekcja** (wykrywanie) krawędzi (edge detection)

Jest to technika segmentacji obrazu, polegająca na znajdowaniu pikseli krawędziowych przez sprawdzanie ich sąsiedztwa.

**Krawędź**

Zbiór pikseli na krzywej mający taką właściwość, że piksele w ich sąsiedztwie, lecz po przeciwnych stronach krzywej mają różne poziomy jasności.

**Cel detekcji**

znalezienie lokalnych nieciągłości w poziomach jasności obrazu oraz granic obiektów zawartych w obrazie.

52

## Cyfrowa wersja gradientu

Pochodna pionowa  $G_x$  funkcji  $f(x,y)$

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

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  $f(x,y)$

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

maska:

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

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

53

## Cyfrowa wersja laplasjanu

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

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).

**Uwaga:** Dla operacji wyostrzania współczynnik maski **K=1**

54

## Filtry uwypuklające krawędzie (w założonym kierunku lub ogólnie)

- Zmniejszają udział lub wręcz usuwają informację o obszarach jednorodnych, uwypuklając informację o krawędziach
- Suma współczynników maski równa 0
- Konieczność skalowania

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

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

1	-2	1
-2	4	-2
1	-2	1

55

## Laplasjany

Tekst

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



56

## Laplasjan/ogólny filtr

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

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

## 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 przeskalowania

## 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);`

## 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  $[0, (M-1)]$

60

Metoda trójkąciowa

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

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

Metoda obcinająca

$$g^o(x,y) = \begin{cases} 0 & \text{dla } g(x,y) < 0 \\ g(x,y) & \text{dla } 0 \leq g(x,y) \leq M-1 \\ M-1 & \text{dla } g(x,y) > M-1 \end{cases}$$

61

Projekty egzaminacyjne

nr.	Tytuł projektu dla APOZ 2022/2023	Nr indeksu studenta	Imię i nazwisko	Wzrost i waga
1	Segmentacja obrazu monochromatycznego/binarnego zawierających cyfry i ich rozpoznawanie z zastosowaniem porównania ze wzorcem			
2	Rozwinięcie opcji zamiany modeli koloru w obrazach kolorowych i przedstawienia linii profilu dla wszystkich wybranych modeli koloru			
3	Implementacje funkcji linii profilu wzdłuż wskazanej myśką prostej lub krzywej w obrazach monochromatycznych i kolorowych			
4	Segmentacja obrazu monochromatycznego/binarnego zawierającego znaki pisarskie i wyszukiwanie znaków o kształtach wypukłych np.: litery w tekście: D, O, itp..			
5	Segmentacja obrazu monochromatycznego/binarnego zawierającego znaki pisarskie i wyszukiwanie znaków o kształtach wklęsłych np.: *, C, itp..			
6	Segmentacja obiektów na podstawie zakresu intensywności wskazanego na wszystkich kanałach (dającego określony zakres kolorów) oraz implementacja algorytmu zliczania wysegmentowanych obiektów z mapy binarnej z użyciem stosu.			
7	Prezentacja obrazu w postaci wykresu 3D			
8	Oprogramowanie algorytmu projekcji wstecznej w obrazach i wykorzystanie tego narzędzia do segmentacji			
9	Oprogramowanie do segmentacji obrazów monochromatycznych barwionych tkanek z użyciem algorytmu wododziału.			
10	Implementacja funkcji detekcji całej twarzy i oczu za pomocą kaskadowego klasyfikatora opartego na cechach Haaara oraz implementacja rozpoznawania własnej twarzy			
11	Implementacja operacji erozji i dyfuzji działających na dowolnie zdefiniowanym elemencie strukturalnym/struturującym			

12	Przeniesienie oprogramowania stworzonego na zajęciach przedmiotu APO tak, aby działał dla systemu operacyjnego iOS			
13	Implementacja operacji wyliczenia transformaty odległościowej i wykorzystanie jej do rozdzielenia obiektów stykających się			
14	Segmentacja z obrazu monochromatycznego/binarnego zawierającego znaki pisarskie o różnych kształtach; poszukujemy wśród nich znaków o kształtach wklęsłych np.: *, C, itp..			
15	Segmentacja obrazu monochromatycznego/binarnego zawierających cyfry i ich rozpoznawanie z zastosowaniem sieci neuronowych			
16	Segmentacja obrazów kolorowych z wykorzystaniem klasteryzacji.			
17	Implementacja progowania obrazu prawdopodobieństwa przypasowania do zadanej tekstury przy użyciu wariacji jasności			
18	Implementacja narzędzia do tworzenia panoramy na podstawie serii zdjęć kolorowych			
19	Implementacja wybranych funkcjonalności w przestrzeni HSV			
20	Implementacja ekstrakcji linii pionowych i poziomych za pomocą operacji morfologicznych			
21	Segmentacja z obrazu monochromatycznego/binarnego obszarów zawierających obiekty o kształtach zawierających odcinki zbiegające po różnym kącie (rog) z wykorzystaniem morfologii matematycznej			
22	Implementacja operacji filtracji logicznych na obrazach binarnych; - rozwinięcie możliwości wyświetlania i zapisywania jako obraz uzyskiwanych na podstawie fragmentów histogramu			
23	Program prezentacji zasad przebiegu procesu wprowadzania i korekty zniekształceń radiometrycznych z wykorzystaniem obrazów			

24	Program prezentacji sposobu działania metody α-NN (α najbliższych sąsiadów) z wizualizacją przestrzeni cech przed i po fazie uczenia oraz w czasie klasyfikacji			
25	Implementacja operacji przenikania obrazów monochromatycznych			
26	Implementacja narzędzia do rozciągania i zawężania histogramu z jednoczesnym zastosowaniem funkcji logarytmicznej i odwrotnej funkcji logarytmicznej			
27	Implementacja operacji wyliczania otoczek wypukłej obiektu w obrazie binarnym			
28	Implementacja operacji wyliczenia średniej i średniej kroczącej (okienkowej) z serii obrazów			
29	Program ukrywania i odczytu obrazu w pliku graficznym			
30	Implementacja operacji rekonstrukcji morfologicznej			
31	Implementacja odszumiania przez uśredniania obrazów tego samego obiektu oraz implementacja operacji różnicy A-B i B-A			
32	Implementacja operacji regulowanego rozciągania zakresów poziomów jasności w obrazach monochromatycznych analogicznie do rozwiązania prezentowanego na wykładzie			
33	Implementacja operacji tworzenia histogramu dwuwymiarowego z obrazu monochromatycznego i jego matematycznego przekształcenia oraz rekonstrukcji obrazów z histogramu			
34	Segmentacja obrazu monochromatycznego zawierających emotikony z różnymi emocjami			

35	Segmentacja obrazu monochromatycznego zawierających drobne obrazy symboliczne obserwowane przez kamerę na taśmie produkcyjnej i otoczenie ich prostokątem dopasowanym do ich rozmiarów.			
36	Program do konstrukcji obrazów monochromatycznych z obrazów kolorowych według wskazań użytkownika co do konwersji koloru			
37	Implementacja wyrównania histogramu obrazu kolorowego wykorzystując dowolne lub wszystkie kanały kolorów w modelu L*a*b*			
38	Implementacja funkcji reprezentacji obrazu monochromatycznego w postaci obrazów o zawężonym zakresie poziomów szarości wyznaczonych według wskazań użytkownika			
39	Przeniesienie oprogramowania stworzonego na zajęciach tak, aby działał dla systemu operacyjnego ANDROID			
40	Segmentacja obrazu monochromatycznego zawierających cyfry i litery (jak w polskich tablicach rejestracyjnych samochodów).			
41	Segmentacja obrazu monochromatycznego zawierających drobne metalowe element obserwowane przez kamerę na taśmie produkcyjnej i otoczenie ich prostokątem dopasowanym do ich rozmiarów.			
42	Program do pseudokolorowania obrazów monochromatycznych według skali barw odpowiadającej tęczy			
43	Implementacja funkcji wykonywania wyrównania histogramu obrazu kolorowego wykorzystując HSV lub HSI			
44	Implementacja funkcji reprezentacji obrazu monochromatycznego w postaci ośmiu binarnych obrazów dla każdego bitu oddzielnie			
45	Segmentacja obrazów z wykorzystaniem klasteryzacji kolorów			

46	Ręczny podział obiektów skłajonych				
47	Rekonstrukcja obrazu z zakłóceniami w postaci jasnych pikseli lub linii				
48	Implementacja progowania obrazu prawdopodobieństwa przypasowania do zadanej tekstury przy użyciu filtrów Gabora				
49	Implementacja progowania obrazu prawdopodobieństwa przypasowania do zadanej tekstury przy użyciu metody LBP				
50	Implementacja progowania obrazu prawdopodobieństwa przypasowania do zadanej tekstury przy użyciu metody SIFT				
51	Implementacja funkcji detekcji całej twarzy i oczu lemurów za pomocą kaskadowego klasyfikatora opartego na cechach Haara				
52	Segmentacja z obrazu monochromatycznego obszarów zawierających obiekty o kształtach zawierających linie zbliżające pod różnym kątem (roggi) z wykorzystaniem metody Harris-Stephens'a.				
53	Segmentacja obrazu monochromatycznego przedstawiającego klocki lego obserwowane przez kamerę na taśmie produkcyjnej i otoczenie ich prostokątem dopasowanym do ich rozmiarów				
54	Segmentacja obrazu kolorowego przedstawiającego ziarna kawy, fasoli i soczewicy obserwowane przez kamerę na taśmie produkcyjnej i klasyfikację ziaren ze względu na kształt i kolor				
55	Segmentacja obrazu kolorowego przedstawiającego klocki lego obserwowane przez kamerę na taśmie produkcyjnej i klasyfikację klocków ze względu na kolor				
56	Implementacja funkcji linii profilu wzdłuż krzywej i łamanej zbudowanej na wskazanych przez użytkownika punktach.				
57	Program do wykonywania procesu wprowadzania i korekci zniekształceń radiometrycznych z wykorzystaniem ciemnego i jasnego obrazu odniesienia				

58	Oprogramowania do segmentacji obrazów monochromatycznych komórek w hodowli z użyciem algorytmu wodoróżała.				
59	Implementacje funkcji linii profilu wzdłuż okręgów i elips o zadanych parametrach i wskazanej lokalizacji środka figur.				
60	Przeniesienie oprogramowania stworzonego na zajęciach tak, aby działał dla systemu operacyjnego ANDROID				
61	Przeniesienie oprogramowania stworzonego na zajęciach tak, aby działał dla systemu operacyjnego IOS				
62	Implementacja narzędzia do tworzenia panoramy na postawie serii zdjęć monochromatycznych				
63	Oprogramowanie do wyszukiwania obiektów analogicznych do obiektów wskazanych na próbkach (małych zdjęciach przedstawiających poszukiwany obiekt)				
64	Wyszukiwanie i lokalizacja emotikonów umieszczanych w tekście.				
65	Program do pseudokolorowania obrazów monochromatycznych według zestawu barw występujących w mapach geograficznych				
66	Udoskonalenie oprogramowania przygotowanego na zajęciach przez dołożenie operacji zmniejszenia udziału szumu przez uśrednianie obrazów zebranych w takich samych warunkach oraz implementację progowania odwrotnego do wszystkich trzech progowań implementowanych na zajęciach				
67	Udoskonalenie oprogramowania przygotowanego na zajęciach przez implementację operacji przenikania obrazów monochromatycznych według skali podanej w procentach				
68	Implementacja operacji unsharpen mask				

69	Implementacja operacji tworzenia histogramu dwuwymiarowego z obrazu kolorowanego bazującego na kanałach oraz rekonstrukcji obrazów z histogramu				
70	Implementacja narzędzia do manipulacji widmem amplitudowym obrazu w celu modyfikacji udziału poszczególnych składowych widma				
71	Implementacja narzędzia do manipulacji widmem amplitudowym obrazu w celu likwidacji zakłóceń periodycznych				
72	Implementacja narzędzia do manipulacji widmem amplitudowym obrazu typu portret w celu tworzenia „uśmiechu Młony Lizy”				
73	Implementacja narzędzia do manipulacji widmem amplitudowym obrazu w celu wydzielenia wybranego zakresu częstotliwości				
74	Segmentacja obrazu kolorowego przedstawiającego klocki lego obserwowane przez kamerę na taśmie produkcyjnej i klasyfikację klocków ze względu na kolor				
75	Segmentacja obrazu monochromatycznego przedstawiającego ziarna kawy, fasoli i soczewicy obserwowane przez kamerę na taśmie produkcyjnej i klasyfikację klocków ze względu na poziom jasności				
76	Segmentacja obrazu monochromatycznego przedstawiającego różnej wielkości śruby obserwowane klocków ze względu na poziom jasności przez kamerę na taśmie produkcyjnej i ich klasyfikacja ze względu na rozmiar				
77	Segmentacja obrazu kolorowego przedstawiającego drewniane klocki obserwowane przez kamerę na taśmie produkcyjnej i ich klasyfikację klocków ze względu na rozmiar				
78	Segmentacja obrazu monochromatycznego przedstawiającego ziarna kawy, fasoli i soczewicy obserwowane przez kamerę na taśmie produkcyjnej i ich klasyfikację klocków ze względu na kształt				

### Uwagi ogólne dotyczące algorytmów

- Omawiając algorytm próbujemy określić czas i pamięć potrzebne do jego wykonania dlatego dyskusja nad algorytmami obejmuje m.in. zagadnienie złożoności obliczeniowej.
- 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:

- Materiały wykładowe POBD 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, (UBI w katalogu \APOD\Pitas).