Instrukcja	**Zaawansowane przetwarzanie obrazów**
1	**Temat**: Wprowadzenie do biblioteki OpenCV Przygotował: mgr inż. Tomasz Michno

1 Wstęp

1.1 OpenCV - krótki wstęp

OpenCV (Open Source Computer Vision) jest otwartą biblioteką dla języka C/C++, służącą do przetwarzania obrazu (zwłaszcza wideo) w czasie rzeczywistym. Dostępna jest na większość systemów operacyjnych. Oprócz języka C (i C++), OpenCV pozwala na programowanie (za pomocą nakładek - wrapperów) w językach takich jak: C#, Python, Ruby i Java.
Biblioteka umożliwia m.in.:

- obsługę popularnych plików graficznych, kamer oraz zapis obrazu i wideo do pliku
- rozpoznawanie obiektów
- wykonywanie standardowych operacji na obrazie (np. wykrywanie krawędzi, rozmycie itp.)
- wykonywanie przekształceń
- rozpoznawanie gestów
- śledzenie ruchu

Oprócz wymienionych funkcji biblioteka posiada własny system tworzenia okien, w których wyświetlany jest obraz lub wideo. Dzięki temu programista może skupić się na właściwej części aplikacji, uzyskując wieloplatformowy kod.

UWAGA! Od wersji OpenCV 4.0 interfejs dla języka C nie jest już wspierany, jeśli chcemy go używać należy zainstalować dowolną wersję wcześniejszą (3.x lub 2.x). Poniżej przedstawię najbardziej podstawowe operacje dla interfejsów: C, C++ i Python.

1.2 OpenCV - Wczytywanie obrazów

1.2.1 Interfejs C

Plik: przyklad1.cpp -Interfejs z języka C

```
1  #include <opencv/cv.h>
2  #include <opencv/highgui.h>
3
4  int main(){
5    cvNamedWindow( "Przyklad 1", CV_WINDOW_AUTOSIZE );
6    IplImage* image = cvLoadImage("obrazek.jpg");
7    if(image==NULL){
8      perror("Wystapil blad przy wczytywaniu obrazka obrazek.jpg!")
           ;
9      cvDestroyWindow( "Przyklad 1" );
10     return 1;
11   }
12   cvShowImage("Przyklad 1", image);
13   cvWaitKey(0);
14   cvReleaseImage( &image );
15   cvDestroyWindow( "Przyklad 1" );
16   return 0;
17 }
```

Listing 1: Wczytywanie i wyświetlanie plików graficznych - interfejs C

Listing znajdujący się powyżej (Listing 1) zawiera pełny kod programu, który wczytuje z pliku obraz oraz wyświetla go w oknie używając interfejsu dla języka C.
Linijki 1 i 2 zawierają pliki nagłówkowe, które należy dołączyć do każdego programu.
W linijce 5 tworzone jest nowe okno o nazwie „Przyklad 1” za pomocą funkcji **cvNamedWindow**():

```
int cvNamedWindow(const char* name, int flags)
```

gdzie:
name - nazwa okna
flags - flagi (obecnie możliwa jest tylko flaga CV_WINDOW_AUTOSIZE)

Linijka 6 odpowiada za utworzenie i wczytanie obrazka (przechowywany za pomocą wskaźnika typu IplImage), za pomocą funkcji **cvLoadImage**:

```
IplImage* cvLoadImage( const char* filename, int iscolor=CV_LOAD_IMAGE_COLOR
)
```

gdzie:
filename - ścieżka do pliku
iscolor - informuje, jak wczytać kolory obrazka (możliwe opcje to: *CV_LOAD_IMAGE_COLOR* - obrazek kolorowy o 3 kanałach, *CV_LOAD_IMAGE_GRAYSCALE* - odcienie

szarości, *CV_LOAD_IMAGE_UNCHANGED* - bez zmian - jak w pliku)

W linijce 12 wyświetlany jest obraz w utworzonym oknie za pomocą funkcji cvShowImage o dwóch parametrach: nazwie okna i wskaźniku na zmienną typu IplImage. Funkcja cvWaitKey (jedynym parametrem jest czas przez jaki funkcja ma czekać na naciśnięcie klawisza; wartość <=0 oznacza nieskończoność) w linii 13 odpowiada za zatrzymanie programu, aż do naciśnięcia klawisza ESC.

Następnie przy wychodzeniu z programu (linie 10 i 11) należy zwolnić pamięć przydzieloną do obrazka (funkcja cvReleaseImage()) oraz usunąć utworzone okno (cvDestroyWindow()).

1.2.2 Interfejs C++

Plik: przyklad1.cpp -Interfejs z języka C++

```
1  #include <opencv2/core.hpp>
2  #include <opencv2/highgui.hpp>
3  using namespace cv;
4
5  int main(){
6
7    namedWindow( "Przyklad 1", WINDOW_AUTOSIZE );      // utworzenie
       okna o nazwie Przyklad 1
8    Mat image= imread("obrazek.jpg");
9    if (!image.data) {
10     perror("Wystapil blad przy wczytywaniu obrazka obrazek.jpg!")
         ;
11     destroyWindow( "Przyklad 1" );           // zniszczenie okna
12     return 1;
13   }
14   imshow("Przyklad 1", image);          // wyswietlenie obrazka w
       oknie o nazwie Przyklad 1
15   waitKey(0);                // wstrzymanie programu do nacisniecia
       klawisza ESC
16   destroyWindow( "Przyklad 1" );           // zniszczenie okna
17   return 0;
18 }
```

Listing 2: Wczytywanie i wyświetlanie plików graficznych - interfejs C++

Listing znajdujący się powyżej (Listing 2) zawiera pełny kod programu, który wczytuje z pliku obraz oraz wyświetla go w oknie używając interfejsu dla języka C++. Jest on znacznie wygodniejszy w użyciu oraz nie wymaga

tworzenia dynamicznie wskaźnika na typ iplImage (co jest znacznie bezpieczniejsze).
Linijki 1 i 2 zawierają pliki nagłówkowe, które należy dołączyć do każdego programu.
W linijce 7 tworzone jest nowe okno o nazwie „Przyklad 1" za pomocą funkcji **namedWindow**():
`int namedWindow(const String & name, int flags)`
gdzie:
name - nazwa okna
flags - flagi, np.:
WINDOW_NORMAL - umożliwia zmianę rozmiaru okna,
WINDOW_AUTOSIZE(Default) - automatycznie ustawia rozmiar okna,
WINDOW_FULLSCREEN – okno odpalane w trybie pełnoekranowym).

Linijka 8 odpowiada za utworzenie i wczytanie obrazka (przechowywany za pomocą zmiennej typu Mat), za pomocą funkcji **imread**:
`Mat imread(const String & filename, int flags = IMREAD_COLOR)`
gdzie:
filename - ścieżka do pliku
iscolor - informuje, jak wczytać kolory obrazka (możliwe opcje to:
IMREAD_COLOR - obrazek kolorowy o 3 kanałach,
IMREAD_GRAYSCALE - odcienie szarości,
IMREAD_UNCHANGED - bez zmian - jak w pliku).

W linijce 14 wyświetlany jest obraz w utworzonym oknie za pomocą funkcji imshow o dwóch parametrach: nazwie okna i zmienną typu Mat. Funkcja waitKey (jedynym parametrem jest czas przez jaki funkcja ma czekać na naciśnięcie klawisza; wartość <=0 oznacza nieskończoność) w linii 15 odpowiada za zatrzymanie programu, aż do naciśnięcia klawisza ESC.

Następnie przy wychodzeniu z programu (linia 16) należy usunąć utworzone okno (destroyWindow()).

1.2.3 Interfejs Python

Plik: przyklad1.cpp -Interfejs z języka Python

```
1 import cv2 as cv
2 import sys
3 img = cv.imread("obrazek.jpg")
4 if img is None:
5     sys.exit("Nie mozna wczytac obrazka")
6 cv.imshow("Okno obrazek.jpg", img)
```

```
7 k = cv.waitKey(0)
```

Listing 3: Wczytywanie i wyświetlanie plików graficznych - interfejs Python

Listing znajdujący się powyżej (Listing 3) zawiera pełny kod programu, który wczytuje z pliku obraz oraz wyświetla go w oknie używając interfejsu dla języka Python. Mimo różnic pomiędzy językami interfejs ten jest bardzo zbliżony do interfejsu C++.
Linijki 1 i 2 zawierają importy bibliotek, które należy dołączyć do każdego programu.
Linijka 3 odpowiada za utworzenie i wczytanie obrazka, za pomocą funkcji **imread**.

W linijce 6 wyświetlany jest obraz w utworzonym oknie za pomocą funkcji imshow o dwóch parametrach: nazwie okna i obrazka. Funkcja waitKey (jedynym parametrem jest czas przez jaki funkcja ma czekać na naciśnięcie klawisza; wartość <=0 oznacza nieskończoność) w linii 7 odpowiada za zatrzymanie programu, aż do naciśnięcia klawisza ESC.

1.3 OpenCV - Proste przekształcenia obrazów

Plik: przyklad2.cpp
Biblioteka OpenCV pozwala na dokonywanie przekształceń m.in. takich jak: rozmywanie, wyostrzanie, transformacje, wykrywanie krawędzi oraz rysowanie figur geometrycznych.
Wszystkie operacje przedstawione są w dokumentacji OpenCV - `https://docs.opencv.org/3.4/d5/d98/tutorial_mat_operations.html`. Na przykładzie zostanie pokazane jedynie rozmywanie i rozpoznawanie krawędzi.

```
1  IplImage* image2 = cvCreateImage( cvGetSize(image), image->depth,
       image->nChannels );
2  if(image2==NULL){perror("Blad przy tworzeniu obrazka"); return
       1;}
3  cvSmooth( image, image2, CV_GAUSSIAN, 15, 5 );
4
5  IplImage* imageBW = cvCreateImage(cvGetSize(image2), IPL_DEPTH_8U
       , 1);
6  IplImage* edgesImage = cvCreateImage(cvGetSize(image2),
       IPL_DEPTH_8U, 1);
7  if(imageBW==NULL || edgesImage==NULL){perror("Blad przy tworzeniu
       obrazka"); return 1;}
8
9  cvCvtColor(image2, imageBW, CV_BGR2GRAY);
10 cvCanny( imageBW, edgesImage, 0.3, 0.8);
```

Listing 4: Przekształcenia graficzne - C

```
#include <opencv2/core.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/imgproc.hpp>
using namespace cv;

int main(){

  namedWindow( "Przyklad 2 - okno 1 - obraz pierwotny",
      WINDOW_AUTOSIZE );       // utworzenie okna o nazwie
      Przyklad 2 - okno 1 - obraz pierwotny
  Mat image= imread("systemyMultimedialne.jpg");
  if (!image.data) {
    perror("Wystapil blad przy wczytywaniu obrazka
        systemyMultimedialne.jpg!");
    destroyWindow( "Przyklad 2 - okno 1 - obraz pierwotny" );
              // zniszczenie okna
    return 1;
  }
  imshow("Przyklad 2 - okno 1 - obraz pierwotny", image);
        // wyswietlenie obrazka w oknie o nazwie PPrzyklad 2 -
      okno 1 - obraz pierwotny

  namedWindow("Przyklad 2 - okno 2 - blur", WINDOW_AUTOSIZE );
  Mat image2(image.rows,image.cols, CV_8UC3, Scalar(0,0,0));//
      utworzenie nowego obrazka o rozmiarze obrazka image (
      najpierw podajemy liczbe wierszy, potem kolumn)

  if(!image2.data){
    perror("Wystapil blad przy tworzeniu obrazu przechowujacego
        przeksztalcony obraz!");
    destroyWindow( "Przyklad 2 - okno 1 - obraz pierwotny" );
              // zniszczenie okna
    destroyWindow( "Przyklad 2 - okno 2 - blur" );
    return 1;
  }

  GaussianBlur( image, image2, Size( 15, 15 ), 0, 0 );  //
      zastosowanie rozmycia gaussowskiego do image, wynik
      zostanie zapamietany w image2
  imshow("Przyklad 2 - okno 2 - blur", image2);

  //====== wykrywanie krawedzi: ======
  // w celu wykrycia krawedzi funkcja cvCanny trzeba
      przekonwertowac obraz do odcieni szarosci
  Mat imageBW(image2.rows,image2.cols, CV_8UC1, Scalar(0,0,0));
      // utworzenie obrazu w odcieniach szarosci
```

```
33    Mat edgesImage(image2.rows,image2.cols, CV_8UC1, Scalar(0,0,0))
          ;
34    if(!imageBW.data || !edgesImage.data){
35      perror("Wystapil blad przy Tworzeniu obrazu przechowujacego
            przeksztalcony obraz!");
36      destroyWindow( "Przyklad 2 - okno 1 - obraz pierwotny" );
                // zniszczenie okna
37      destroyWindow( "Przyklad 2 - okno 2 - blur" );
38      return 1;
39    }
40    cvtColor(image2, imageBW, CV_BGR2GRAY);  // skopiowanie obrazu
          kolorowego do obrazu z odcieniami szarosci (konwersja na
          odcienie szarosci)
41    Canny( imageBW, edgesImage, 0.3, 0.8);  // wykrycie krawedzi
42    //=================================
43    namedWindow( "Przyklad 2 - okno 3 - krawedzie",
          CV_WINDOW_AUTOSIZE );
44
45    imshow("Przyklad 2 - okno 3 - krawedzie", edgesImage);
46
47    waitKey(0);                // wstrzymanie programu do nacisniecia
          klawisza ESC
48    destroyWindow( "Przyklad 2 - okno 1 - obraz pierwotny" );
            // zniszczenie okna
49    destroyWindow( "Przyklad 2 - okno 2 - blur" );
50    return 0;
51  }
```

Listing 5: Przekształcenia graficzne - C++

```
1  import cv2 as cv
2  import sys
3  import numpy as np
4
5  image = cv.imread(cv.samples.findFile("systemyMultimedialne.jpg")
       )
6  if image is None:
7      sys.exit("Nie mozna wczytac obrazka")
8  cv.imshow("Okno systemyMultimedialne.jpg", image)
9
10 cv.namedWindow("Przyklad 2 - okno 2 - blur", cv.WINDOW_AUTOSIZE )
11
12 image2 = cv.GaussianBlur(image,(5,5),0) # zastosowanie rozmycia
       gaussowskiego do image, wynik zostanie zapamietany w image2
13 cv.imshow("Przyklad 2 - okno 2 - blur", image2)
14
15 #======= wykrywanie krawedzi: =======
16 # w celu wykrycia krawedzi funkcja cvCanny trzeba przekonwertowac
       obraz do odcieni szarosci
```

```
17 imageBW = cv.cvtColor(image2, cv.COLOR_BGR2GRAY) # skopiowanie
       obrazu kolorowego do obrazu z odcieniami szarosci (konwersja
       na odcienie szarosci)
18 edgesImage = cv.Canny(imageBW, 0.3, 0.8, 3) #  wykrycie krawedzi
19 #=================================
20 cv.namedWindow( "Przyklad 2 - okno 3 - krawedzie", cv.
       WINDOW_AUTOSIZE )
21
22 cv.imshow("Przyklad 2 - okno 3 - krawedzie", edgesImage)
23
24 k = cv.waitKey(0)
```

Listing 6: Przekształcenia graficzne - Python

Większość przekształceń graficznych w OpenCV wymaga, aby stworzyć kopię obrazka do którego zostanie zapisany wynik. W tym celu należy go utworzyć za pomocą funkcji cvCreateImage.

Rozmywanie można wykonać poprzez: interfejs C: cvSmooth(), C++/Python: GaussianBlur().

1.4 Operacje na pikselach

W celu odczytania wartości piksela w punkcie należy skorzystać z funkcji
Interfejs C:

```
CvScalar cvGet2D( CvArr* image, int y, int x);
```

gdzie:
image - obrazek źródłowy
y, x - współrzędne y i x punktu, z którego będzie pobierany piksel
Typ CvScalar posiada jedynie pole o nazwie val, będące tablicą czteroelementową (gdzie kolejno są zapisane wartości r (indeks 0), g (indeks 1) i b (indeks 2)).
Interfejs C++:
obraz o jednym kanale:

```
Scalar intensity = img.at<uchar>(y, x);
```

obraz o 3 kanałach:

```
Vec3b intensity = img.at<Vec3b>(y, x);
uchar blue = intensity.val[0];
uchar green = intensity.val[1];
uchar red = intensity.val[2];
```

Interfejs Python:
do pojedynczego piksela możemy uzyskać dostęp tak jak do elementu tablicy numpy: `img[row, column]`

W celu ustawienia wartości piksela w punkcie należy skorzystać z funkcji:

Interfejs C:

```
void cvSet2D( CvArr* image, int y, int x, CvScalar point);
```

gdzie:
image - obrazek źródłowy
y, x - współrzędne y i x punktu, z którego będzie pobierany piksel
point - zmienna typu CvScalar, zawiera informacje o kolorze piksela, który ma zostać wstawiony na obrazie we współrzędnych x,y
Interfejs C++:
obraz o jednym kanale:

```
image.at<uchar>(y,x) = color;
```

obraz o 3 kanałach:

```
image.at<Vec3b>(Point(x,y)) = color;
```

Interfejs Python:
do pojedynczego piksela możemy uzyskać dostęp tak jak do elementu tablicy numpy: `img[row, column]`

1.5 Dokumentacja biblioteki OpenCV

Pod poniższymi adresami można znaleźć opisy przetwarzania z użyciem poszczególnych interfejsów:
- `https://docs.opencv.org/3.4.17/df/d4e/group__imgproc__c.html`
- `https://docs.opencv.org/3.4.17/d7/dbd/group__imgproc.html`
- `https://docs.opencv.org/3.4/d2/d96/tutorial_py_table_of_contents_imgproc.html`

2 Zadania

1. Stwórz nowy obraz w programie (bez wczytywania z pliku), a następnie narysuj na nim używając funkcji graficznych dowolny rysunek. Obrazek należy wyświetlić.

2. Wczytaj dowolny obrazek z dysku, a następnie go wyświetl.

3. Do wczytanego obrazka dodaj minimum 4 dowolne przekształcenia (np. wyostrzanie, inwersja kolorów itp.).

4. Dodaj do powyższego programu funkcję realizującą odbicie lustrzane (poziome lub pionowe - do wyboru) bloku pikseli z użyciem funkcji działających bezpośrednio na pikselach.