Laboratorium 9

Przetwarzanie obrazu

OpenCV posiada szereg funkcji umożliwiających przetwarzanie obrazu. Stwórzmy dwie zmienne przechowujące obrazy. Do jednej załadujemy obraz zapisany na dysku. Druga będzie przechowywać efekt naszych zabiegów.

```
Mat before = imread("sharp.jpg", 1);
Mat after;
```

Kod poniżej przedstawia użycie wybranych funkcji służących do tzw. rozmycia obrazu (z ang. *blur*) stosując różne algorytmy przetwarzania obrazu:

```
blur(before, after, Size(10, 10), Point(-1,-1));
GaussianBlur(before, after, Size(11, 11), 0, 0);
medianBlur(before, after, 11);
```

O wszystkich możesz poczytać tutaj. Pierwszymi dwiema zmiennymi każdej z tych funkcji są oczywiście zmienne przechowujące kolejno obraz przed i po modyfikacji. Na szczególną uwagę zasługuje **trzecia zmienna**, która opisuje rozmiary okna wykorzystywanego przez te funkcje do filtracji obrazu.

Prześledź wpływ wielkości okna na uzyskiwany efekt.

Innymi, często wykorzystywanymi zabiegami stosowanymi w przetwarzaniu obrazu są dylatacja i erozja. Poniżej przedstawiono przykład ich użycia w OpenCV:

```
dilate(before, after, change, Point(-1, -1), 2, 1, 1);
erode(before, after, change, Point(-1, -1), 2, 1, 1);
```

Użycie tych funkcji wymaga zdefiniowania wcześniej osobnej zmiennej typu *Mat*, która mówi o rozmiarze okna przetwarzania obrazu:

```
int change_size = 5;
Mat change = getStructuringElement(MORPH_CROSS,
    Size(2*change_size + 1, 2*change_size + 1),
    Point(change_size, change_size));
```

Prześledź wpływ wielkości okna na uzyskiwany efekt.

Najpopularniejszymi cechami opisującymi obraz są jasność i kontrast. W OpenCV zmianę tych parametrów można zrealizować na dwa sposoby (przynajmniej). W zrozumieniu ich działania przydatna jest wiedza na temat funkcji opisującej transformację pojedynczego pixela:

$$g(i,j) = \alpha \cdot f(i,j) + \beta$$

gdzie:
f(i,j) – jeden pixel obrazu
α – zwykle nazywane *wzmocnieniem*, tutaj jest to po prostu *kontrast*
β – zwykle nazywany *zaburzeniem*, tutaj jest to po prostu *jasność*

Poniższy kod przedstawia jak iterując po wszystkich pixelach obrazu (i wszystkich składowych koloru pixela) ręcznie zmieniać wartość kontrastu i jasności obrazu:

```
double alpha = 5;
double beta = 50;
for( int y = 0; y < before.rows; y++ )
{
    for( int x = 0; x < before.cols; x++ )
    {
        for( int c = 0; c < 3; c++ )
        {
            after.at<Vec3b>(y,x)[c] =
            saturate_cast<uchar>(alpha*(before.at<Vec3b>(y,x)[c] ) + beta);
        }
    }
}
```

Niestety, gdy obraz jest duży, iteracja po wszystkich jego pixelach zajmuje pewien okres czasu. Obraz można jednak przeskalować do wybranej przez siebie wielkości. Poniższy kod pokazuje jak dowolny obraz przeskalować proporcjonalnie do obrazu wyjściowego, który nie przekracza 500 pixeli szerokości (wzdłuż osi X):

```
double imageCols = before.cols;
double imageRows = before.rows;
double maxCols = 500;
double fx = maxCols/imageCols;
Size dsize = Size(fx*imageCols, fx*imageRows);
resize(before, after, dsize, fx, fx, INTER_AREA);
```

Zmienna *dsize* mówi nam po prostu o wielkości obrazu wyjściowego.

Istnieje jednak szybsza funkcja, która może posłużyć do zmiany jasności i kontrastu obrazu. Jej użycie zaprezentowano poniżej:

```
double brightness = 30;
double contrast = 0.5;
before.convertTo(after, -1, contrast, brightness);
```

Prześledź wpływ parametrów alpha i beta, oraz brightness i contrast na uzyskiwany efekt.

A jak zmieni się jakość wykrywania twarzy (lista 8) po przetworzeniu obrazu którymś z przedstawionych algorytmów?