| Instrukcja | **Zaawansowane przetwarzanie obrazów** |
|---|---|
| **3** | **Temat**: Obsługa kamer, stosowanie filtrów, rozpoznawanie wzorców w obrazie w czasie rzeczywistym<br>Przygotował: mgr inż. Tomasz Michno |

# 1 Wstęp

```
1  CvCapture* capture = cvCreateCameraCapture(0);
2  if(capture==NULL){perror("Blad."); return 1;}
3  IplImage* frame;
4
5  while(1) {
6      frame = cvQueryFrame( capture );
7      if( !frame ) break;
8
9      cvShowImage( "Przyklad - kamera", frame );
10
11     char c = cvWaitKey(33);
12     if( c == 27 ) break;
13 }
14
15 cvReleaseCapture( &capture );
```

Listing 1: Obsługa kamery - odczyt i wyświetlanie klatek w oknie

Obsługa obrazu z kamery w bibliotece OpenCV jest bardzo zbliżona do obsługi zwykłych obrazów. Jednocześnie możliwe jest obsługiwanie wielu kamer. Programista tworzy urządzenie, które będzie odczytywało ramki (klatki, funkcja cvCreateCameraCapture, linia 1), a następnie w pętli pobiera kolejne klatki z użyciem funkcji cvQueryFrame (linia 5). Wszystkie ramki są typu IplImage, co pozwala na przetwarzanie ich w identyczny sposób, co zwykłe obrazy.

`CvCapture* cvCreateCameraCapture( int index )`
gdzie:
*index* - numer kamery, która ma być obsługiwana; w przypadku posiadania tylko jednej kamery lub w celu wybrania dowolnej można użyć wartości -1

`IplImage* cvQueryFrame( CvCapture* capture )`
gdzie:
*capture* - wskaźnik na zmienną typu CvCapture, utworzoną za pomocą cvCreateCameraCapture

W listingu 1 warto zwrócić uwagę na sposób obsługi klawiatury (linie 10 i 11) - w linii 10 odczytujemy kod ASCII wciśniętego klawisza, a następnie w linii 11 sprawdzamy jego kod. W przypadku wciśnięcia klawisza ESC kończymy pętlę.
Na koniec należy zwolnić zasoby przydzielone urządzeniu rejestrującemu za pomocą funkcji cvReleaseCapture (linia 14).

Biblioteka OpenCV posiada zaimplementowane funkcje służące do rozpoznawania wzorców. Najprostszą z nich jest cvMatchTemplate(), niestety posiada ona poważną wadę którą jest czas wykonania. Z tego względu warto skorzystać z funkcji FastMatchTemplate dostępnej dla języka C++ pod adresem `http://opencv.willowgarage.com/wiki/FastMatchTemplate`. Jest ona bardzo szybka oraz pozwala wykrywać wiele wystąpień dane wzorca w jednym wykonaniu. Wymaga dodania pliku nagłówkowego vector.

```
bool FastMatchTemplate( const IplImage& source,
const IplImage& target,
vector<CvPoint>* foundPointsList,
vector<double>* confidencesList,
int matchPercentage = 70,
bool findMultipleTargets = true,
int numMaxima = 5,
int numDownPyrs = 2,
int searchExpansion = 15 );
```

Najważniejsze parametry:
*source, target* - obrazek źródłowy oraz wzorzec
*foundPointsList* - wektor zawierający znalezione punkty, w których znajduje się wzorzec
*confidencesList* - wektor zawierający stopnień pewności dla każdego znalezionego wystąpienia (0-100)
*matchPercentage* - minimalny poziom rozpoznania wzorca
*findMultipleTargets* - wartość true oznacza wyszukiwanie wielu wystąpień

```
1 IplImage* marker = cvLoadImage("marker.png");
2 vector<CvPoint> foundPointsList;
3 vector<double>  confidencesList;
4 if( !FastMatchTemplate( *image,
5                          *marker,
6                          &foundPointsList,
7                          &confidencesList, 50, true, 15 ) ){
```

```
8     printf( "\nERROR: Fast match template failed.\n" );
9     return 3;
10  }
11  for(int i=0; i<foundPointsList.size(); i++){
12      cout<<"x="<<foundPointsList[i].x<<", y="<<foundPointsList[i].
            y<<endl;
13      cvCircle(image, foundPointsList[i], 64, CV_RGB(255, 0, 0 ) );
14  }
```

Listing 2: Rozpoznawanie wzorców

W Listingu 2 używana jest również funkcja cvCircle rysująca okrąg:

`void cvCircle(CvArr* img, CvPoint center, int radius, CvScalar color, int thickness=1, int lineType=8, int shift=0)`
gdzie:
*img* - obrazek na którym zostanie narysowany okrąg
*center* - środek okręgu
*radius* - promień okręgu
*color* - kolor linii (można użyć funkcji CV_RGB(r,g,b))
*thickness* - grubość linii (jeśli wartość ujemna, okrąg jest wypełniany)
*lineType* - typ linii ( dozwolone typy: 8, 4, CV_AA )
*shift* - liczba bitów ułamkowych w punkcie środka okręgu i promieniu

# 2 Zadania

1. Napisz program, który będzie wyświetlał obraz z kamery na ekranie.
2. Do powyższego programu dodaj 4 dowolne przekształcenia, wyświetlając je w osobnych oknach.
3. W osobnym oknie odwróć fragment obrazu lub cały obraz z użyciem bezpośredniego dostępu do pikseli (np. podobnie jak w instrukcji nr 1).
4. Napisz program, który będzie rozpoznawał dowolną liczbę 4 wzorców (uproszczony rysunek) na obrazie wczytanym z dysku. W miejscu rozpoznania powinien być rysowany obrazek w pełnej jakości (nieuproszczony), za pomocą cvGet2D() i cvSet2D().