Instrukcja laboratoryjna **7**	**Zaawansowane przetwarzanie obrazów**
	Temat: Rozpoznawanie kodów kreskowych i 2D
	Przygotował: mgr inż. Tomasz Michno

Wstęp teoretyczny

Jedną z bibliotek służących do rozpoznawania kodów kreskowych w języku C/C++ jest ZBar(http://zbar.sourceforge.net/)[1]. Rozpoznaje ona kody 1D oraz kody 2D QR w następujący sposób:

1. Utworzenie skanera kodów QR
2. Utworzenie obrazu "surowego" – zawierającego bajty kolorów (odcienie szarości)
3. Przeskanowanie obrazu
4. odczyt w pętli wszystkich odnalezionych kodów kreskowych (np. wyświetlenie użytkownikowi programu)
5. usunięcie pamięci przydzielonej dla biblioteki

Konfiguracja biblioteki dla IDE (domyślna instalacja ZBar):

- dodać katalog *C:\Program Files (x86)\ZBar\include* do listy ścieżek katalogów zawierających pliki nagłówke (CB: Build Options-> Search directories)
- dodać plik *C:\Program Files (x86)\ZBar\lib\libzbar-0.lib* dla linkera (CB: Bulid options-> Linker settings)
- skopiować wszystkie pliki dll zkatalogu *C:\Program Files (x86)\ZBar\bin* do katalogu z plikiem wykonywalnym tworzonego programu

Tworzenie programu (w języku C++ - plik z rozszerzeniem *.cpp):

dodanie pliku nagłówkowego:

```
#include <zbar.h>
```

dodanie namespace dla biblioteki:

```
using namespace zbar;
```

Utworzenie i skonfigurowanie skanera kodów kreskowych:

```
   ImageScanner scanner;
    scanner.set_config(ZBAR_NONE, ZBAR_CFG_ENABLE, 1);
```

Utworzenie binarnego obrazka dla skanera:

```
   Image image(width, height, "Y800", raw, width * height);
```

raw – tablica typu char zawierająca kolory na obrazie (UWAGA! Odcienie szarości – jeden element tablicy – jeden piksel),

1 Dla języków C# i Java oraz systemu operacyjnego Android warto wypróbować bibliotekę Zxing - http://zxing.org/

width, height – wymiary obrazu

Zeskanowanie obrazka:

```
int n = scanner.scan(image);
```

metoda scan zwraca liczbę rozpoznanych kodów kreskowych

Wyświetlenie rozpoznanych danych z kodów kreskowych – przejście w pętli for:

```
    // extract results
    for(Image::SymbolIterator symbol = image.symbol_begin();
        symbol != image.symbol_end();
        ++symbol) {
        // wyświetlenie odczytanych danych zkodów kresowych:
        cout << "decoded " << symbol->get_type_name()
             << " symbol \"" << symbol->get_data() << '"' << endl;
    }
```

metoda get_data() zwraca dane odczytane z kodu kreskowego (np. adres URL), metoda get_type_name() informuje jakiego rodzaju dane zostały odczytane.

Na końcu należy usunąć pamięć przydzieloną dla biblioteki:

```
image.set_data(NULL, 0);
```

Biblioteka wymaga, aby jako obrazek wejściowy do skanowania przesłać tablicę bajtów typu char zawierącą piksele. W celu połączenie bibliteki OpenCV z ZBar można wykorzystać atrybut imageData obiektu typu IplImage (należy pamiętać, aby był on w odcieniach szarości!). Przykładowa konwersja obrazka odczytanego przez OpenCV (nazwa zmiennej: imageOpenCV) wraz w konwersją na odcienie szarości:

```
  IplImage *im_gray = cvCreateImage(cvGetSize(imageOpenCV),IPL_DEPTH_8U,1); // utworzenie obrazka w odcieniach szarości
  cvCvtColor(imageOpenCV,im_gray,CV_RGB2GRAY); // konwersja obrazu RGB na odcienie szarości
```

```
  int width = imageOpenCV->width;    // szerokość obrazka
  int height= imageOpenCV->height;   // wysokość obrazka
  char *raw = (char*)imageOpenCV->imageData; // tablica typu char, którą przekażemy do image(width, height, "Y800", raw, width * height)
```

Przy odczycie pliku z dysku możliwa jest również jego automatyczna konwersja na odcienie szarości:

```
IplImage* im_gray = cvLoadImage("image.jpg",CV_LOAD_IMAGE_GRAYSCALE);
```

Zadanie

Napisz program, który korzystając z biblioteki Zbar będzie rozpoznawał kody kreskowe QR z plików. Następnie dodaj do niego obsługę kamery. Przetestuj poprawność rozpoznawania kodów przez biblitekę przy różnych stopniach degradacji kodu.