Piotr Chłystek 226100
Michał Chojnacki 225936

Data oddania sprawozdania: 18.12.2017 r.
Termin zajęć: Poniedziałek 7:30-10:15, TP

Urządzenia peryferyjne

Ćwiczenie 12

Obsługa kamery USB

Prowadzący:
dr inż. Jan Nikodem

1. **Cel ćwiczenia**

   Celem ćwiczenia było napisanie programu, który wykryje urządzenia typu kamera, podłączone do komputera, by następnie móc wyświetlić i przetwarzać w aplikacji obrazy, przechwytywane z dwóch, wybranych przez użytkownika urządzeń.

2. **Wstęp teoretyczny**

   Kamera cyfrowa jest urządzeniem, które realizuje przekształcenie rejestrowanego obrazu do postaci sygnału cyfrowego. Najczęściej kamera jest podłączana przy pomocy łącza USB, jednak coraz częściej spotyka się kamery wbudowane w monitor lub obudowę laptopa. Przy pomocy obiektywu, filmowany obraz jest rzutowany na matrycę, najczęściej typu CCD (Charge coupled devices) lub CMOS (Complemetary metal-oxide semiconductor), która jest rozwiązaniem tańszym, oferującym szybszą formę odczytu (można odczytać dowolną liczbę pikseli, w dowolnej kolejności, w matrycy CCD trzeba odczytać całą zawartość), rozdzielczość obrazu 640x480 pikseli oraz 30 klatek na sekundę. Matryca posiada informacje na temat oświetlenia obrazu, nie posiada jednak informacji o kolorach. Do uzyskania kolorowego obrazu używa się filtrów w formie mozaiki w kolorach RGB (red, green, blue – kolejno czerwony, zielony i niebieski). Najczęściej stosowany w urządzeniach cyfrowych do rejestracji obrazu jest filtr Bayera. Uzyskiwany jest efekt „zielonej szachownicy', ponieważ zawiera 50% zielonych, 25% czerwonych i 25% niebieskich filtrów elementarnych , ponieważ ludzkie oko jest najwrażliwsze na zielony kolor. Dzięki algorytmom demozaikującym (np. Interpolacji Bayera), analizującym sąsiednie piksele otrzymujemy kolorowy, wyraźny obraz. Kamery typu USB, takie jak te, użyte do wykonania ćwiczenia mają powszechne zastosowanie w komunikacji, jednak coraz częściej używa się ich w ramach poprawy bezpieczeństwa a nawet w medycynie, przy przeprowadzaniu operacji chirurgicznych.

3. **Opis programu**

   Program został napisany w technologii C#, z wykorzystaniem open source'owego framework'u AForge.NET, zaprojektowanego specjalnie z myślą o przetwarzaniu obrazów, sieciach neuronowych, robotyce i tym podobnych gałęziach nauk technicznych. Biblioteki:

```
using AForge;
using AForge.Video.DirectShow;
using AForge.Video;               //przetwarzanie video
using Accord.Video.VFW;           //biblioteki Accord, zamienne z bibliotekami
//AForge, które nie zostały prawidłowo załączone w projekcie
using Accord.Video.FFMPEG;
using AForge.Imaging.Filters;     //przetwarzanie obrazu, filtry
```

   Lista atrybutów:

```
 FilterInfoCollection videoDevicesList; //obiekt klasy FilterInfoCollection,
 //pozwalający wylistować wyszukane urządzenia
 VideoCaptureDevice cameraOne;    //obiekty typu VideoCapture, przechwytujące
 //obraz z kamer
 VideoCaptureDevice cameraTwo;
 int brightess1 = 0; //zmienne okreslajace parametry takie jak jasność,
       //kontrast i nasycenie kolorów dla obu kamery
 int contrast1 = 0;
 int saturation1 = 0;
 int brightess2 = 0;
 int contrast2 = 0;
 int saturation2 = 0;
 bool isRecording1 = false; //zmienne boolowskie określające czy obraz z kamery
       //1 lub 2 jest nagrywany
 bool isRecording2 = false;
VideoFileWriter writer; //obiekt klasy VideoFileWriter to zapisu filmu
```

```
Funkcja, wykrywająca urządzenia
 private void button_searchDev_Click(object sender, EventArgs e)
 {
   videoDevicesList = new FilterInfoCollection(FilterCategory.VideoInputDevice);
     foreach (FilterInfo videoDevice in videoDevicesList)
     {
         cmbDevList1.Items.Add(videoDevice.Name);
         cmbDevList2.Items.Add(videoDevice.Name);
     }
 }
```

Ustawienie bitmapy w celu aktualizacji obrazu, pbCam1 jest obiektem typu PictureBox, utworzonym w oknie głównym. W tej funkcji następuje również ustawienie parametrów obrazu (jasność, kontrast i nasycenie kolorów)

```
 private void CameraOne_NewFrame(object sender, NewFrameEventArgs eventArgs)
 {
     Bitmap bitmap1 = (Bitmap) eventArgs.Frame.Clone();
     BrightnessCorrection br = new BrightnessCorrection(brightess1);
     ContrastCorrection cr = new ContrastCorrection(contrast1);
     SaturationCorrection sr = new SaturationCorrection(saturation1);
     bitmap1 = br.Apply((Bitmap)bitmap1.Clone());
     bitmap1 = cr.Apply((Bitmap)bitmap1.Clone());
     bitmap1 = sr.Apply((Bitmap)bitmap1.Clone());

     pbCam1.Image = bitmap1;

     if (isRecording1)
     {
         writer.WriteVideoFrame(bitmap1);
     }
 }
```

Funkcje, do rozpoczęcia przechwytywania i zakończenia przechwytywania obrazu z kamery, by utworzyć obiekt typu VideoCaptureDevice do przechwytywania obrazu, należy przekazać MonikerString, który jest wskaźnikiem na urządzenie, jako argument

```
 private void buttonSsCam1_Click(object sender, EventArgs e)
 {
       cameraOne = new
VideoCaptureDevice(videoDevicesList[cmbDevList1.SelectedIndex].MonikerString);
     cameraOne.NewFrame += new NewFrameEventHandler(CameraOne_NewFrame);
     cameraOne.Start();
 }

 private void buttonCamOneStop_Click(object sender, EventArgs e)
 {
     cameraOne.Stop();
  }
```

Funkcja, zapisująca zrzut ekranu do pliku. Przechwytuje obraz i otwiera okno dialogowe, służące do zapisu obrazu

```
 private void buttonPictureCamOne_Click(object sender, EventArgs e)
 {
     buttonCamOneStop_Click(sender, e);
     Bitmap picture = (Bitmap) pbCam1.Image;
     saveFileDialog.Filter = "Bitmap Image|*.bmp";
     saveFileDialog.Title = "Save an Image File";
     saveFileDialog.ShowDialog();
     System.IO.FileStream fs = (System.IO.FileStream)
             saveFileDialog.OpenFile();
```

```
        picture.Save(fs, System.Drawing.Imaging.ImageFormat.Bmp);
        fs.Close();
    }
```

Funkcje, zmieniające parametry obrazu, przy pomocy suwaków, wyświetlonych w oknie głównym aplikacji.

```
private void jasnosc1TrackBar_Scroll(object sender, EventArgs e)
{
        if(cameraOne.IsRunning)
        brightess1 = jasnosc1TrackBar.Value;

}
private void kontrast1TrackBar_Scroll(object sender, EventArgs e)
{
        if(cameraOne.IsRunning)
        contrast1 = kontrast1TrackBar.Value;
}

private void nasycenie1TrackBar_Scroll(object sender, EventArgs e)
{
        if (cameraOne.IsRunning)
        saturation1 = nasycenie1TrackBar.Value;
}
```

Funkcje do rozpoczęcia i zakończenia nagrywania filmu z kamery 1, niedziałające poprawnie. Po wciśnięciu przycisku nagrywania, w jego miejscu powinien pojawić się przycisk, zatrzymujący nagrywanie, po wciśnięciu którego otwiera się okno dialogowe do zapisu filmu w formacie pliku .avi

```
private void buttonRecordingCamOne_Click(object sender, EventArgs e)
{
    if (cameraOne.IsRunning)
    {
        buttonRecordingCamOne.Enabled = false;
        buttonStopRecordingCamOne.Enabled = true;
        try
        {
            isRecording1 = true;
            writer = new VideoFileWriter();
            writer.Open("video.avi", pbCam1.Image.Width, pbCam1.Image.Height,
             30, VideoCodec.MPEG4);
        }
        catch
        {}
    }
}
private void buttonStopRecordingCamOne_Click(object sender, EventArgs e)
{
    if (cameraOne.IsRunning)
    {
        isRecording1 = false;
        writer.Close();

        buttonRecordingCamOne.Enabled = true;
        buttonStopRecordingCamOne.Enabled = false;
        saveFileDialog.Filter = "Avi Files (*.avi)|*.avi";
        saveFileDialog.Title = "Save a Video File";
```

```
            saveFileDialog.ShowDialog();
            System.IO.FileStream fs =
                 (System.IO.FileStream)saveFileDialog.OpenFile();
        }
    }
```

4. Wnioski

Biblioteki napisane w ramach frameworku AForge udostępniają szeroką funkcjonalność użytkownikom, chcącym zajmować się przetwarzaniem obrazu. Program napisany na zajęciach działa poprawnie, pomijając aspekt zapisu filmu do pliku avi. Nie udało się nam również procedury wykrywającej ruch.