Politechnika Wrocławska

Wydział Informatyki i Telekomunikacji

Urządzenia peryferyjne

Kamera internetowa

Ćwiczenie nr 12

Prowadzący:

dr inż. Jan Nikodem

Grupa:

Piątek 13:15 TP

Autorzy:

Daniel Glazer, 252743

Paweł Helisz, 252779

Wrocław 28.01.2022

Spis treści

1 Cel ćwiczenia **3**

2 Działanie kamery **3**

3 Kod aplikacji **4**

3.1 Wyświetlanie podglądu obrazu 4

3.2 Zmienianie ustawień kamery 4

3.3 Zapisywanie obrazu w postaci zdjęcia 5

3.4 Zapisywanie obrazu w postaci filmu 5

3.5 Zoom 5

3.6 Detekcja ruchu 6

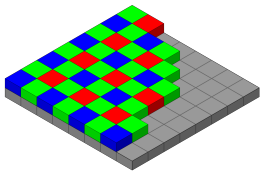
1 Cel ćwiczenia

Celem ćwiczenia było napisanie aplikacji obsługującej kamerę internetową. Aplikacja miała realizować takie funkcje jak:

- wyświetlanie podglądu obrazu z kamery
- zmienianie parametrów kamery
- zapisywanie obrazu z kamery w formacie **.jpg*
- zapisywanie obrazu w postaci filmu w formacie **.avi*
- detekcje ruchu

2 Działanie kamery

Kamera cyfrowa jest to urządzenie, które realizuje przekształcenie rejestrowanego obrazu do postaci sygnału cyfrowego. Najczęściej kamera jest podłączona przy pomocy łącza USB, ale potyka się również 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). Matryca posiada informacje na temat oświetlenia obrazu, nie posiada jednak informacji o kolorach. Do uzyskania kolorowego obrazu używa się filtrów, które umieszczane są na matrycy pomiędzy źródłem światła a matrycą. Filtry takie istnieją w formie mozaiki w kolorach RGB. RGB jest to model kolorów, w którym światło czerwone, zielone i niebieskie jest sumowane na różne sposoby w celu odtworzenia szerokiej gamy kolorów. Przykładem takiego filtru, który jest zarazem najczęściej wykorzystywanym filtrem jest filtr Bayera. Uzyskiwany jest na nim efekt tzw. zielonej szachownicy wynika to z faktu, że filtr zawiera w stosunku 1:2:1 filtrów elementarnych. Gdzie pierwsza jedynka oznacza kolor czerwony, dwójka kolor zielony, a ostatnia kolor niebieski. Dwukrotnie większa liczba zielonych filtrów jest rezultatem tego, że oko człowieka jest najbardziej wrażliwe na kolor zielony.



Rysunek 1: Przykład filtru Bayera na tablicy pikseli

Aby utworzyć z tej kombinacji trójkolorowych filtrów, wielobarwny i realistyczny obraz, korzysta się z metody zwanej interpolacją Bayera. Polega ona na tym, że oszacowuje natężenie światła, które zostało zmierzone przez sąsiednie piksele. Natężenie światła padające na piksel przekłada się na nasycenie prezentowanego przez niego koloru. Po złożeniu ze sobą trzech podstawowych kolorów filtra dadzą one dowolny inny kolor tworząc wielobarwny obraz. W czasach teraźniejszych kamery internetowe, mają powszechne zastosowanie w komunikacji, jednak coraz częściej używa się ich w ramach poprawy bezpieczeństwa a nawet w medycynie, przy przeprowadzaniu operacji chirurgicznych.

3 Kod aplikacji

3.1 Wyświetlanie podglądu obrazu

Aby umożliwić podgląd obrazu z kamery USB należało wyszukać dostępne urządzenia. Następnym krokiem było wykorzystanie biblioteki *AForge.Video* do utworzenia obiektu *VideoCaptureDevice*, który pozwala obsługiwać przechwytywany przez kamerę obraz.

```
1     //ustawienie filtra na przeszukiwanie urzadzen video
2     videoDevices = new FilterInfoCollection(FilterCategory.VideoInputDevice);
3     foreach (FilterInfo device in videoDevices)
4         {
5             comboBoxDevices.Items.Add(device.Name); //dodanie urzadzen na liste
6         }
7     camera = new VideoCaptureDevice();
8     if(comboBoxDevices.Items.Count!=0)
9         comboBoxDevices.SelectedIndex = 0;
```

Następnie utworzono bitmapę do przechowywania klatek obrazu. Następnie została ona wyświetlona na ekranie wykorzystując *pictureBoxOutput.Image*.

```
1     var bitmap = (Bitmap)eventArgs.Frame.Clone();
2     pictureBoxOutput.Image = bitmap;
```

Kolejnym krokiem było stworzenie metody łączącej uzyskane klatki obrazu.

```
1     private void startPreview()
2         {
3             if (camera.IsRunning)
4             {
5                 camera.Stop();
6                 pictureBoxOutput.Image = null;
7                 pictureBoxOutput.Invalidate();
8             }
9             else
10            {
11            camera = new VideoCaptureDevice(videoDevices[comboBoxDevices.SelectedIndex].MonikerString);
12            camera.NewFrame += videoSource_NewFrame; //laczenie klatek obrazu
13            camera.Start(); //wlaczenie przechwytywania kolejnego obrazu
14            }
15        }
```

3.2 Zmienianie ustawień kamery

Aby zmienić ustawienia kamery wykorzystano w tym celu systemowe okno właściwości urządzeń video.

```
1     camera.DisplayPropertyPage(Handle);
```

3.3 Zapisywanie obrazu w postaci zdjęcia

W celu zapisania obrazu z kamery w postaci zdjęcia należało skopiować aktualną bitmapę, podać nazwę pliku, rozszerzenie (jpg) oraz ścieżkę do katalogu.

```
1    private void captureImage()
2        {
3            Bitmap varBmp = new Bitmap(pictureBoxOutput.Image);
4            Bitmap current = (Bitmap)varBmp.Clone();
5            string filepath = Environment.CurrentDirectory;
6            string fileName = System.IO.Path.Combine(filepath, @"name.jpg");
7            current.Save(fileName);
8            current.Dispose();
9        }
```

3.4 Zapisywanie obrazu w postaci filmu

Do zapisu obrazu w postaci filmu wykorzystano bibliotekę *Accord.Video.FFMPEG*. Metoda Open() pozwoliła na stworzenie pliku .avi. Klatki obrazu zostały złączone wykorzystując metodę WriteVideoFrame(), która jako argument przyjmuje aktualną bitmapę.

```
1    writer = new VideoFileWriter();
2    writer.Open("video.avi", pictureBoxOutput.Image.Width, pictureBoxOutput.Image.Height, 60,
     VideoCodec.Default);
3    writer.WriteVideoFrame(currentFrame);
```

3.5 Zoom

Przybliżanie obrazu wymagało skalowanie bitmapy o wartości odczytywane z suwaka. Wykorzystano w tym celu bibliotekę *System.Drawing*.

```
1    Bitmap tmpImage1 = new Bitmap(bitmap.Width, bitmap.Height);
2    Graphics g = Graphics.FromImage(tmpImage1);
3    Rectangle dstRect = new Rectangle(0, 0, tmpImage1.Width, tmpImage1.Height);
4    Rectangle srcRect;
5
6    int width = bitmap.Width / zoomValue;
7    int height = bitmap.Height / zoomValue;
8    int left = bitmap.Width / 2 - (width / 2);
9    int top = bitmap.Height / 2 - (height / 2);
10
11   srcRect = new Rectangle(left, top, width, height);
12   pictureBoxOutput.CreateGraphics().DrawImage(bitmap, dstRect, srcRect, GraphicsUnit.Pixel);
```

3.6 Detekcja ruchu

Detekcja ruchu została zrealizowana dzięki bibliotece *AForge.Vision.Motion*. Porównuje ona klatki obrazu i zaznacza ruch jako czerwone piksele na ekranie.

```
1     detector = new MotionDetector(new TwoFramesDifferenceDetector(), new MotionBorderHighlighting());
2     detectionLevel = 0;
3
4      private void videoSourcePlayer1_NewFrame(object sender, ref Bitmap image)
5         {
6             detectionLevel = detector.ProcessFrame(image);
7         }
```