Urządzenia peryferyjne

Prowadzący zajęcia: dr inż. Jan Nikodem
Termin zajęć: Czwartek TN, godz. 14:10

Osoby wykonujące ćwiczenie: Mateusz Gawłowski, Bartosz Szymański	**Numer grupy:** 2
Tytuł ćwiczenia: Kamery cyfrowe	**Data wykonania ćwiczenia:** 21.12.2023

1. Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się z zasadami obsługi kamery/aparatu cyfrowego i napisanie aplikacji umożliwiającej zrobienie zdjęcia/filmu i zapisania/przesłania go na inne urządzenie oraz implementacja jednej z technik związanych z kamerami/aparatami cyfrowymi. Techniki te to: wykrywanie ruchu, udostępnianie filmu on-line w Internecie, tworzenie obrazów 3D, tworzenie fotografii HDR.

2. Wstęp teoretyczny

2.1 Budowa kamery

Matryca **CCD** stanowi płytę krzemową podzieloną na piksele, czyli kwadratowe obszary elektrycznie odizolowane od siebie. W trakcie rejestracji obrazu każdy piksel gromadzi ładunek proporcjonalny do intensywności padającego światła. Odczyt danych z pikseli w matrycy CCD odbywa się wzdłuż całych rzędów, co implikuje brak możliwości odczytu pojedynczych wartości pikseli. Następnie, zgromadzone napięcie przechodzi przez konwerter A/D, generując na wyjściu dane cyfrowe, które ostatecznie są zapisywane w pamięci kamery. Po zebraniu informacji z całego rzędu, dane o kolejnej grupie pikseli przesyłane są do kanału, kontynuując ten proces aż do uzyskania pełnych danych z całej matrycy. Rzędy pikseli przechodzą do kolejnego poziomu w każdym cyklu, aż do momentu, gdy dane z całej matrycy zostaną odczytane i zapisane w pamięci kamery. Przedstawione rozwiązanie uniemożliwia dostęp do pojedynczych pikseli przed zakończeniem procesu odczytu danych z całej matrycy i zapisania ich w pamięci kamery.

Działanie matrycy **CMOS** jest analogiczne do matrycy CCD, gdzie zgromadzony ładunek przekształcany jest w impuls elektryczny. Jednak różnice występują w procesie konwersji oraz w dalszej transmisji sygnału.

Podobnie jak matryce CCD, matryce CMOS gromadzą dane dotyczące intensywności światła w postaci ładunku. To, co wyróżnia tę technologię spośród sensorów CCD, to możliwość dostępu do każdego pojedynczego piksela o współrzędnych (x, y). W przeciwieństwie do sensorów CCD, gdzie występuje tylko jeden konwerter przekształcający zgromadzony ładunek na napięcie w całym układzie, matryca CMOS wyposażona jest w osobny element konwersji dla każdego piksela. W przypadku droższych i bardziej zaawansowanych modeli, pojedyncze piksele posiadają dedykowane przetworniki analogowo-cyfrowe (DAC). Takie rozwiązanie umożliwia zwiększenie prędkości

odczytu danych, co przekłada się na efektywność pracy kamery i systemu wizyjnego. Warto jednak zauważyć, że te dodatkowe elementy zajmują dodatkową przestrzeń na powierzchni matrycy. Dlatego sensory CMOS zazwyczaj mają mniejszą powierzchnię światłoczułą niż matryce CCD.

Rozdzielczość matrycy światłoczułej określa maksymalną liczbę pikseli, jaka może być wykorzystana do stworzenia zdjęcia – wyrażana jest ona w megapikselach.
Megapiksel to miara wskazująca na liczbę elementarnych punktów w matrycy, czyli pikseli. Prefiks "mega" oznacza miliony pikseli, które matryca jest w stanie odwzorować.
Rozdzielczość matrycy staje się zatem jednostką umożliwiającą określenie rzeczywistej wielkości obrazu tworzonego lub odwzorowywanego przy jej użyciu. Im wyższa wartość rozdzielczości, tym lepsze odwzorowanie detali widocznych na żywo.

2.2 Kolory w aparatach cyfrowych

Matryca światłoczuła w aparacie cyfrowym jest zdolna do rejestrowania jedynie intensywności światła, nie rozróżniając kolorów. **Tworzenie kolorów** jest możliwe dzięki układowi filtrów i procesorowi, które są integralnymi elementami aparatu cyfrowego. Cała matryca jest pokryta “szachownicą” mikroskopijnych filtrów, z których każdy znajduje się nad jednym elementem światłoczułym. Trzy podstawowe kolory - czerwony, zielony i niebieski - są rozmieszczone tak, że jeden kolor pokrywa jeden piksel. Każdy piksel rejestruje natężenie światła koloru odpowiadającego filtru, którym jest “przykryty”.
Rozkład tych kolorów nie jest jednak równomierny, co wynika z charakterystyki ludzkiego oka. Mówiąc ściśle, stosunek poszczególnych kolorów RGB wynosi 1:2:1, co oznacza, że na dwa piksele zielone przypada po jednym pikselu niebieskim i czerwonym.
W celu utworzenia z tej kombinacji trójkolorowych punktów, wielobarwny i realistyczny obraz, stosuje się metodę zwaną interpolacją Bayera. Metoda ta polega na oszacowaniu brakujących informacji dla poszczególnego koloru na podstawie natężenia światła zmierzonego przez sąsiednie piksele. Natężenie światła padające na piksel jest przekładane na nasycenie koloru prezentowanego przez ten piksel. Po złożeniu ze sobą trzech podstawowych kolorów, otrzymujemy dowolny inny kolor. Proces ten jest również nazywany de-mozaikowaniem.

Balans bieli to proces dostosowywania kolorów na zdjęciach lub w nagraniach wideo w taki sposób, aby białe obiekty wyglądały na prawdziwie białe, niezależnie od warunków oświetleniowych. W rzeczywistości światło może mieć różne temperatury barwowe, co wpływa na to, jak kolory są postrzegane. Na przykład światło słoneczne ma inną temperaturę barwową niż światło żarówek.

2.3 Zoom

Zoom cyfrowy to technika powiększania obrazu, która polega na zapisie proporcjonalnego wycinka obrazu z matrycy aparatu. W praktyce, po wykonaniu zdjęcia, obraz jest najpierw zapisywany na matrycy aparatu. Następnie, w zależności od wybranego powiększenia, aparat przy pomocy oprogramowania wycina centralny fragment zdjęcia o wielkości zgodnej z zadanym powiększeniem. Można to porównać z działaniem lupy, kiedy patrzymy na wydruk z powiększeniem i bez. Niestety, podobnie jak przy użyciu lupy, użycie zoomu cyfrowego powoduje ubytek na jakości obrazu. Dzieje się tak dlatego, że ostatecznie wycięty fragment musi mieć konkretny rozmiar, a więc jest rozciągany, aby nadrobić ubytek po wycięciu.
Właściwie taką samą operację możemy wykonać samodzielnie, kadrując zdjęcie w programie do obróbki na smartfonie lub komputerze. Takie przybliżenie jest pozorne. Nie otrzymasz dzięki niemu większej ilości szczegółów na zdjęciu.

Zoom optyczny to technika powiększania obrazu, która polega na zmianie ogniskowej obiektywu aparatu. W praktyce, zoom optyczny działa poprzez mechaniczne przesuwanie szklanych elementów w obiektywie.

Podczas korzystania ze zoomu optycznego, obiektyw utrzymuje stałe położenie płaszczyzny ostrości, niezależnie od ustawionej długości ogniskowej. Zmieniając ogniskową, fotograf może sprawić, że obiekty będą wydawały się większe i zmieszczą się w pełnej ramce zdjęcia.
W przeciwieństwie do zoomu cyfrowego, zoom optyczny działa w taki sposób, że powiększony obraz ma dokładnie tyle szczegółów, co cały obraz.

2.3 HDR

HDR (High Dynamic Range) to technologia, która zwiększa zakres rozpiętości pomiędzy ciemnymi i jasnymi tonami, co pozwala na uzyskanie żywszych kolorów i bardziej naturalnego obrazu. Efekt działania tego rozwiązania widoczny jest przede wszystkim w ciemnych scenach z jasnymi punktami.
W fotografii, technika HDR polega na wykonaniu kilku ekspozycji tego samego kadru, z których część jest niedoświetlona, a część prześwietlona. Pozwala ona otrzymać obraz sceny charakteryzującej się dużą rozpiętością tonalną.

2.4 Anaglif

Anaglify to obrazy, które dają złudzenie trójwymiaru podczas oglądania ich za pomocą specjalnych, najczęściej czerwono-cyjanowych okularów.
Tworzenie anaglifów w fotografii polega na nałożeniu na siebie dwóch zdjęć, wykonanych z lekkim poziomym przesunięciem, odpowiadającym obrazom dla lewego i prawego oka. Zdjęcia takie można uzyskać używając szyny nakładanej na statyw lub specjalnych aparatów fotograficznych o dwóch obiektywach.

2.4 Stabilizacja drgań

Stabilizacja optyczna - OIS działa poprzez zastosowanie ruchomego elementu wewnątrz obiektywu lub matrycy kamery, który jest w stanie kompensować drobne ruchy lub wibracje urządzenia. To pozwala na uzyskanie klarowniejszych i bardziej stabilnych obrazów, zwłaszcza w warunkach słabego oświetlenia lub przy dłuższych czasach naświetlania.
Stabilizacja cyfrowa - Stabilizacja obrazu odbywa się w trakcie przetwarzania danych z matrycy kamery. Procesor obrazu monitoruje informacje z przetwornika w czasie rzeczywistym, a następnie poddaje je analizie za pomocą różnorodnych algorytmów wykrywających i mierzących ruch powstającego obrazu. Jeśli algorytm sklasyfikuje przesunięcie jako wstrząs, uruchamiane są odpowiednie przekształcenia, zazwyczaj polegające na dostosowaniu obrazu w celu zniwelowania efektów zmiany. Aby system EIS skutecznie przeciwdziałał wstrząsom i przemieszczał obraz w określonych granicach, konieczne jest utworzenie pewnych rezerwowych obszarów na krawędziach obrazu.

2.5 Bokeh

Efekt bokeh to estetyczny efekt w fotografii, charakteryzujący się miękkim, rozmytym tłem wokół ostro sfotografowanego przedmiotu.
Efekt ten powstaje głównie dzięki zastosowaniu obiektywów z szerokim otworem przysłony. Gdy przysłona obiektywu jest otwarta na maksimum (niskie liczby f-stop, na przykład f/1.4 lub f/2.8), powstaje płytka głębia ostrości. To oznacza, że tylko niewielki obszar przedmiotu będzie ostro sfotografowany, podczas gdy tło oraz przedmioty znajdujące się przed lub za głównym motywem staną się rozmyte.

3. Rozwiązanie zadań wraz z krótkim opracowaniem

3.1 Kod

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Imaging;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using AForge;
using AForge.Video.DirectShow;
using AForge.Video;
using AForge.Imaging.Filters;
using Accord.Video.VFW;
using Accord.Video.FFMPEG;

namespace camera
{
   public partial class Form1 : Form
   {
      FilterInfoCollection videoDevicesList;
      VideoCaptureDevice cameraInput;
      int brightess1 = 0;
      int contrast1 = 0;
      float saturation1 = 0;
      bool isRecording1 = false;
      VideoFileWriter writer;

      private Bitmap oldBitmap;
      bool wykrywajruch = false;
      float pixeldiff = 0.1f;
      int pixelcount = 100;
      int framedelay = 10;
      int framecount = 0;


 public static bool CompareBitmapsFast(Bitmap bmp1, Bitmap bmp2)
      {
         if (bmp1 == null || bmp2 == null)
            return false;
         if (object.Equals(bmp1, bmp2))
            return true;
         if (!bmp1.Size.Equals(bmp2.Size) || !bmp1.PixelFormat.Equals(bmp2.PixelFormat))
            return false;

         int bytes = bmp1.Width * bmp1.Height * (Image.GetPixelFormatSize(bmp1.PixelFormat) / 8);

         bool result = true;
         byte[] b1bytes = new byte[bytes];
         byte[] b2bytes = new byte[bytes];

         BitmapData bitmapData1 = bmp1.LockBits(new Rectangle(0, 0, bmp1.Width, bmp1.Height),
ImageLockMode.ReadOnly, bmp1.PixelFormat);
         BitmapData bitmapData2 = bmp2.LockBits(new Rectangle(0, 0, bmp2.Width, bmp2.Height),
ImageLockMode.ReadOnly, bmp2.PixelFormat);

         Marshal.Copy(bitmapData1.Scan0, b1bytes, 0, bytes);
         Marshal.Copy(bitmapData2.Scan0, b2bytes, 0, bytes);

         for (int n = 0; n <= bytes - 1; n++)
```

```
    {
        if (b1bytes[n] != b2bytes[n])
        {
            result = false;
            break;
        }
    }

    bmp1.UnlockBits(bitmapData1);
    bmp2.UnlockBits(bitmapData2);

    return result;
}

private void video_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());

    if (isRecording1)
    {
        writer.WriteVideoFrame(bitmap1);
    }
    else
    {
        if (oldBitmap != null)
        {
            if (wykrywajruch == true)
            {
                framecount++;
                if (framecount > framedelay)
                {
                    framecount = 0;
                    if (CompareBitmapsFast(bitmap1, oldBitmap))
                    {

                    }
                    else
                    {

                    }
                }
            }
        }
        oldBitmap = bitmap1;
        camoutput.Image = bitmap1;
    }
}
public Form1()
{
    InitializeComponent();
    startcamera.Enabled = false;
    stopcamera.Enabled = false;
    takephoto.Enabled = false;
    startfilm.Enabled = false;
    stopfilm.Enabled = false;
    detectstart.Enabled = false;
    detectstop.Enabled = false;
}

private void button2_Click(object sender, EventArgs e)
{
```

```
    stopcamera_Click(sender, e);
    Bitmap picture = (Bitmap)camoutput.Image;
    saveFileDialog1.Filter = "Bitmap Image|*.bmp";
    saveFileDialog1.Title = "Save an Image File";
    saveFileDialog1.ShowDialog();
    System.IO.FileStream fs = (System.IO.FileStream)saveFileDialog1.OpenFile();
    picture.Save(fs, System.Drawing.Imaging.ImageFormat.Bmp);
    fs.Close();
    startcamera_Click(sender, e);
}


    startcamera.Enabled = true;
    camlist.Items.Clear();
    camlist.ResetText();
    videoDevicesList = new FilterInfoCollection(FilterCategory.VideoInputDevice
    foreach (FilterInfo videoDevice in videoDevicesList)
    {
        camlist.Items.Add(videoDevice.Name + videoDevice.MonikerString.Substring(62));
    }
}

private void startcamera_Click(object sender, EventArgs e)
{
    startcamera.Enabled = false;
    stopcamera.Enabled = true;
    takephoto.Enabled = true;
    startfilm.Enabled = true;
    detectstart.Enabled = true;
    if (camlist.SelectedIndex >= camlist.Items.Count) return;
    if (camlist.SelectedIndex < 0) return;

    cameraInput = new VideoCaptureDevice(videoDevicesList[camlist.SelectedIndex].MonikerString);
    cameraInput.NewFrame += new NewFrameEventHandler(video_NewFrame);
    cameraInput.Start();
}

private void stopcamera_Click(object sender, EventArgs e)
{
    startcamera.Enabled = true;
    stopcamera.Enabled = false;
    takephoto.Enabled = false;
    startfilm.Enabled = false;
    stopfilm.Enabled = false;
    detectstart.Enabled = false;
    detectstop.Enabled = false;
    if(cameraInput!=null)
        cameraInput.Stop();
}

private void startfilm_Click(object sender, EventArgs e)
{
    startcamera.Enabled = false;
    stopcamera.Enabled = false;
    startfilm.Enabled = false;
    stopfilm.Enabled = true;
    detectstart.Enabled = false;
    detectstop.Enabled = false;
    if (cameraInput != null)
        if (cameraInput.IsRunning)
    {
        try
        {
            saveFileDialog1 = new SaveFileDialog();
            saveFileDialog1.Filter = "Avi Files (*.avi)|*.avi";
            saveFileDialog1.Title = "Save a Video File";
            saveFileDialog1.ShowDialog();
```

```
                writer = new VideoFileWriter();
                writer.Open(saveFileDialog1.FileName, camoutput.Image.Width, camoutput.Image.Height, 30,
VideoCodec.MPEG4);
                isRecording1 = true;
            }
            catch
            {

            }
        }
    }

    private void brightness_Scroll(object sender, EventArgs e)
    {
        brightess1 = brightness.Value;
    }

    private void contrast_Scroll(object sender, EventArgs e)
    {
        contrast1 = contrast.Value;
    }

    private void saturation_Scroll(object sender, EventArgs e)
    {
        saturation1 = saturation.Value/100.0f;
    }

    private void stopfilm_Click(object sender, EventArgs e)
    {
       startcamera.Enabled = false;
       stopcamera.Enabled = true;
       startfilm.Enabled = true;
       stopfilm.Enabled = false;
       detectstart.Enabled = true;
       detectstop.Enabled = false;
       if (cameraInput != null)
       if (cameraInput.IsRunning && isRecording1)
       {
          isRecording1 = false;
          writer.Close();
       }
    }


    private void pixdiff_Scroll(object sender, EventArgs e)
    {
       pixeldiff = pixdiff.Value;
    }

    private void pixcount_Scroll(object sender, EventArgs e)
    {
       pixelcount = pixcount.Value;
    }

    private void delayframe_Scroll(object sender, EventArgs e)
    {
       framedelay = delayframe.Value;
    }

    private void detectstart_Click(object sender, EventArgs e)
    {
       startcamera.Enabled = false;
       stopcamera.Enabled = false;
       startfilm.Enabled = false;
       stopfilm.Enabled = false;
       detectstart.Enabled = false;
       detectstop.Enabled = true;
```

```
            wykrywajruch = true;
        }

        private void detectstop_Click(object sender, EventArgs e)
        {
            startcamera.Enabled = false;
            stopcamera.Enabled = true;
            startfilm.Enabled = true;
            stopfilm.Enabled = false;
            detectstart.Enabled = true;
            detectstop.Enabled = false;
            wykrywajruch = false;
        }

    }
}
```

3.2 Omówienie kodu

Kod przedstawia implementację programu do obsługi kamery w aplikacji okienkowej przy użyciu bibliotek AForge i Accord.

- Zdefiniowano interfejs użytkownika przy użyciu klasy Form1, na formularzu umieszczono różne kontrolki, takie jak przyciski, suwaki, pola wyboru, etykiety, itp., do obsługi różnych funkcji kamery
- Lista dostępnych kamer jest wczytywana do camlist (ComboBox) po naciśnięciu przycisku "Refresh" (button1).
- Po wybraniu kamery i naciśnięciu przycisku "Start Camera" (startcamera), program tworzy obiekt VideoCaptureDevice i zaczyna przechwytywać obrazy z kamery. Wyświetlane są one w kontrolce camoutput (PictureBox).
- Po wybraniu kamery i naciśnięciu przycisku "Start Camera" (startcamera), program tworzy obiekt VideoCaptureDevice i zaczyna przechwytywać obrazy z kamery. Wyświetlane są one w kontrolce camoutput (PictureBox).
- Po naciśnięciu przycisku "Start Recording" (startfilm), program inicjuje nagrywanie wideo. Wykorzystywany jest obiekt VideoFileWriter z Accord.NET.
- Wideo jest nagrywane do pliku AVI o wybranej przez użytkownika ścieżce i nazwie.
- Po naciśnięciu przycisku "Stop Recording" (stopfilm), nagrywanie zostaje zakończone.
- istnieje metoda CompareBitmapsFast, która porównuje dwa obrazy piksel po pikselu.

4. Wnioski

Realizacja ćwiczenia w znaczący sposób wpłynęła na nasz poziom rozumienia tematu w obszarze kamer cyfrowych. Podczas trwania laboratorium udało się utworzyć spełniający założenia zadania kod, który został przedstawiony i krótko opisany powyżej.

5. Literatura

https://vesta.astro.amu.edu.pl/Education/ccd2/node7.html
https://avicon.pl/en/dzialalnoscbr/technologie-wizyjne/cmos-ccd/
https://www.izabezpieczenia.pl/kom pendium-wiedzy/faq-pytania/czym-rozni-sie-kamera-cmos-od-ccd-jaki-rodzaj-kamery-jest-lepszy/
https://www.neonet.pl/slowniczek-pojec/rozdzielczosc-matrycy.html

http://www.fotoporadnik.pl/elementy-aparatu-kolory-cyfrowe.html
https://www.optyczne.pl/115-słownik-Matryca_filtrów.html
http://www.fotoporadnik.pl/elementy-aparatu-kolory-cyfrowe.html