SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR
GÉPÉSZMÉRNÖKI, INFORMATIKAI ÉS VILLAMOSMÉRNÖKI KAR

GÉPI LÁTÁS

GKNB_INTM038

BIZTONSÁGI KAMERA:

KAMERA VIDEÓFELVÉTELEN EGYIDEJŰLEG EGY MOZGÁS FELISMERÉSE ÉS NYOMON KÖVETÉSE.

Szerző

Név: **Litter Ádám**
Neptun kód: **FFX181**
E-mail: **adamlitter99@gmail.com**

Tartalom

Felhasználói dokumentáció

Feladat

Kamera videofelvételen egyidejűleg egy mozgás felismerése és nyomon követése. A felvétel lehet kamera élő képe, vagy pedig előre elkészített felvétel. Az forrás kiválasztását lehetővé kell tenni a felhasználó számára.

Környezet

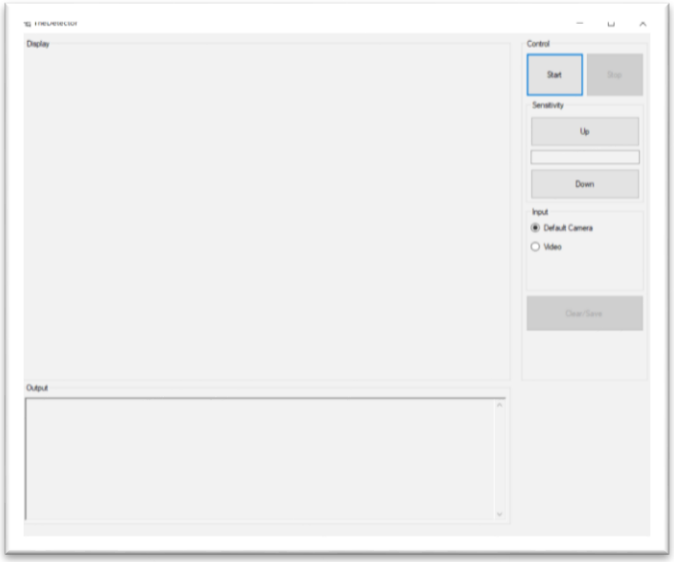
- A fejlesztés nyelve: C#,
- target framework: .Net 4.7.2,
- target platform: x86
- target OS: Windows

Felhasználás

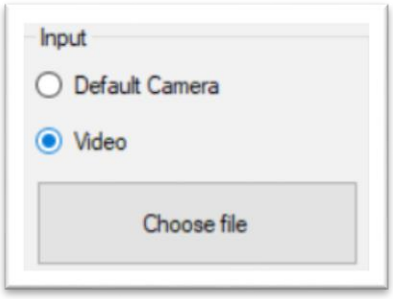
Futtatás

Az indítás lehetséges a megfelelő (.EXE kiterjesztésű) fájl futtatásával, illetve tetszőleges IDE-ban, vagy Command Prompt-ban való futtatással.

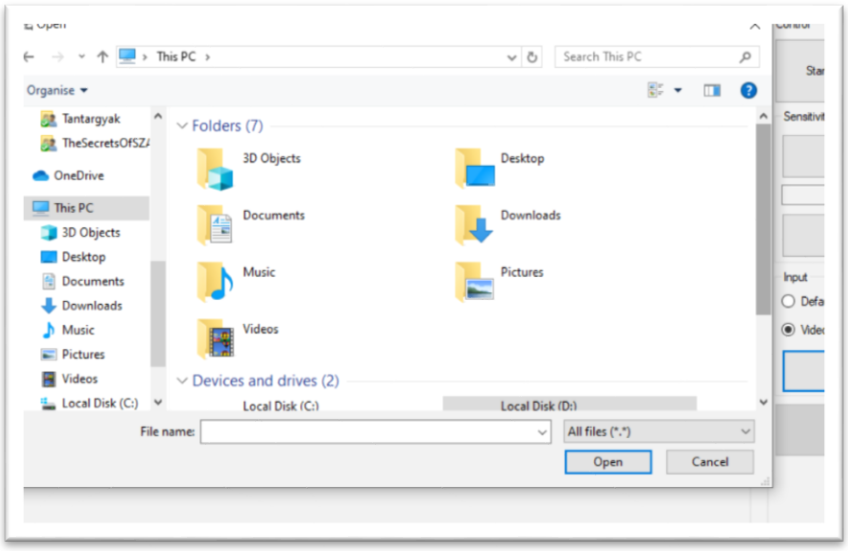
Kezelőfelület



A Start gombra kattintva a felhasználó automatikusan az alapértelmezett kamera élő képét jelenítheti meg a képernyőn.



A „Video” rádiógomb kiválasztása után lehetősége nyílik egy tetszőleges fájl importálására.



A kiválasztásban egy megjelenő dialógus ablak segít.
(A projekthez mellékelve megtalálható egy .avi kiterjesztésű fájl a teszteléshez)

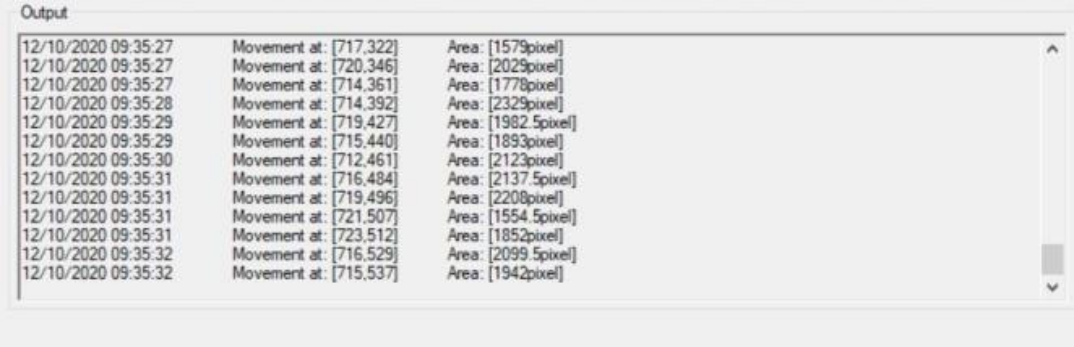
A fájl kiválasztása után szintén a „Start” gomb lenyomásával indítható el a lejátszás.

A „Stop” gomb segítségével leállíthatjuk a folyamatot, majd pedig az ezután elérhetővé váló „Clear/Save” gomb használatával törölhetjük az output adatokat, vagy szöveges (.txt) fájl formájában menthetjük egy dialógus ablak segítségével.

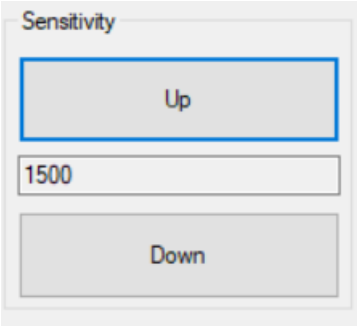
A program outputja

A program futása során az inputjára érkező felvételeken, mozgás detektálása esetén a mozgó objektumot kék négyzettel határolja, valamint a középpontjának haladási útvonalát zöld összefüggő vonallal jelzi.

Továbbá a középpontok koordinátáit, a detektálás idejét és az érzékelt objektum kiterjedésének (pixel) mértékét az output ablakon felsorolja.



Érzékenység beállítsa



Az „Up” és „Down” gombokkal a felhasználó állíthatja azt a kiterjedési méretet (pixel) amelynél a program csak nagyobb mozgásokat fog kiszűrni.

Fejlesztői dokumentáció

Feladat

Kamera videofelvételen egyidejűleg egy mozgás felismerése és nyomon követése. A felvétel lehet kamera élő képe, vagy pedig előre elkészített felvétel. A forrás kiválasztását lehetővé kell tenni a felhasználó számára.

Környezet

- A fejlesztés nyelve: C#,
- target framework: .Net 4.7.2,
- target platform: x86
- target OS: Windows

Forráskód

Összefoglalva

A program egy egyszerű Windows Form App feléítését követi.
Futáskor a megfelelő gombok lenyomása esetén beolvas egy inputot kameráról, vagy kiválasztott videó fájlból.
A felvételt egy képernyőn jeleníti meg és közben különböző transzformációkat hajt végre a képkockákon.
Mindehhez EmguCV cross platformot használ, amely lehetővé teszi számunkra a különböző OpenCV metódusok hívását.

Elméleti háttér

A feladat két lényegi részfeladatra bontható.

Az első részben sorra megkeressük a különbséget minden képkocka és rákövetkező képkocka között. Mivel a pixelek intenzitása beolvasásuk során tömbökben tárolódnak, ezért egyszerűen csak kiszámíthatjuk a képkockák pixelértékeinek páronkénti abszolút különbségét.

A második részben az alapvető feladat az éldetektálás.
Az éleket mindig két adott képkocka abszolút különbségén keressük, mivel ezen a képkockán csakis a két felvétel közötti elmozdulás jelenik meg.
Az éldetektálás során azokat a pontokat keressük, ahol az intenzitás hirtelen változik, azonban ez nem csak az éleknél lehetséges, ezért különböző transzformációkat hajtunk végre, hogy a felesleges elemeket eltüntessük, illetve a megfelelő részeket nyomatékosítsuk

A feladat pontos kivitelezése a program kódjának kommentelésénél nyomonkövethető.

Hibaforrás

A transzformációk sorrendje illetve paraméterei megfelelő beállításokat igényelnek.
A túlzott zavar miatt az éldetektálás nem képes nagyobb objektumokat behatárolni, mivel több kisebb objektumként érzékeli.

Forráskód

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/// Description:Biztonsági kamera: Kamera videófelvételen egyidejűleg egy mozgás felismerése és nyomon követése.
///
///                    A feladat megvalósításához felhasználtam:                                    ///
///                    EmguCV (cross platform .Net wrapper)                                         ///
///                    ->OpenCV (imagine processing library) funkcióinak hívásához.                 ///
///                    A feladat részeként felhasználói valamint fejlesztői dokumentáció            ///
///                    is készült.                                                                  ///
/// Author : Litter Ádám                                                                            ///
/// Neptun : FFX181                                                                                 ///
/// GitHub : https://github.com/LitterAdamDev/                                                      ///
/// E-mail : adamlitter99 @gmail.com                                                                ///
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

using System;
using System.Drawing;
using System.Windows.Forms;
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.CvEnum;
using Emgu.CV.Util;
using System.IO;

namespace CVision
{
   public partial class Form1 : Form
   {
      /*OpenCV adatfolyam deklarálása*/
      VideoCapture capture = null;

      /*2db képkocaka deklarálása*/
      Image<Bgr, Byte> frame1 = null;
      Image<Bgr, Byte> frame2 = null;

      /*Pontok és tároló a contúrvonalak szélsőértékeinek kereséséhez és tárolásához*/
      VectorOfPoint vp = new VectorOfPoint();
      Point last = Point.Empty;
      Point next = Point.Empty;

      /*Érzékenység és vezérlési mód, egyébb metódusok által használt globális változók */
      double m_focus = 1500;
      string startMode = string.Empty;
      bool noRecordYet = true;
      bool stillgoing = true;
      bool drawlineenabled = false;

      /*Form kezdete*/
      public Form1()
      {
         InitializeComponent();
      }
      private void Form1_Load(object sender, EventArgs e)
      {

      }

      //ProcessUpdate metódus:
      //  -Képkockák cseréje
      //  -Transzformációk elvégzése
      //  -Éldetektálás
      //  -Output kezelése
      void ProcessUpdate(object sender, EventArgs e)
      {
         try
         {
            /*Képkockák inicializálása UI-en definiált input alapján*/
```

```
            /*capture.QueryFrame() megragadja, dekódolja és returnolja a következő képkockát*/
            frame1 = capture.QueryFrame().ToImage<Bgr, Byte>();
            frame2 = capture.QueryFrame().ToImage<Bgr, Byte>();
        }
        catch (Exception)
        {
          /*A lényeg hogy nem áll le*/
        }

        if (frame1 == null || frame2 == null)
        {
            btnStart.Enabled = true;
            btnStop.Enabled = false;
            stillgoing = false;
        }
        if (stillgoing)
        {
            /*Pontokat tartalmazó vektorokat tartalmazó vektor a kontúrvonalak pontjainak tárolására*/
            VectorOfVectorOfPoint contours = null;

            /*Szürkeárnyalati konverzió mindkettő képre*/
            Image<Gray, Byte> gray_frame1 = new Image<Gray, Byte>(frame1.Width, frame1.Height);
            Image<Gray, Byte> gray_frame2 = new Image<Gray, Byte>(frame1.Width, frame1.Height);
            CvInvoke.CvtColor(frame1, gray_frame1, ColorConversion.Bgr2Gray);
            CvInvoke.CvtColor(frame2, gray_frame2, ColorConversion.Bgr2Gray);

            /*Két kép közötti differencia számítás*/
            Image<Bgr, Byte> diff = new Image<Bgr, Byte>(frame1.Width, frame1.Height);
            CvInvoke.AbsDiff(gray_frame1, gray_frame2, diff);

            /*Laplace és Gauss simítás alkalmazása a differenciált képen*/
            Image<Gray, Byte> blur_diff = new Image<Gray, Byte>(frame1.Width, frame1.Height);
            CvInvoke.GaussianBlur(diff, blur_diff, new Size(3, 3), 5);
            Image<Gray, Byte> laplacian = new Image<Gray, Byte>(frame1.Width, frame1.Height);
            CvInvoke.Laplacian(blur_diff, laplacian, DepthType.Default,3,2,0,BorderType.Replicate);

            /*CvInvoke.Dilate() kitágítja a képet hogy kitöltse a lyukakat.*/
            Image<Gray, Byte> dilate = new Image<Gray, Byte>(frame1.Width, frame1.Height);
            CvInvoke.Dilate(laplacian, dilate, null, new Point(-1, -1), 2, BorderType.Default, new MCvScalar(1, 1, 1));

            /* Küszöbölés*/
            Image<Gray, Byte> threshold = new Image<Gray, Byte>(frame1.Width, frame1.Height);
            CvInvoke.AdaptiveThreshold(dilate, threshold, 60, AdaptiveThresholdType.GaussianC, ThresholdType.BinaryInv,
3, 2);

            /*Kontúrvonalak megkeresése*/
            contours = new VectorOfVectorOfPoint();
            Mat hier = new Mat();
            CvInvoke.FindContours(threshold, contours, hier, RetrType.Tree, ChainApproxMethod.ChainApproxSimple);

            //Kontúrvonalak végigiterálása
            //Szélsőértékek megkeresése
            //Határoló négyzetek rajzolása
            //Középpontok keresése
            //Középpontok összekötése és kiiratása outputra
            for (int i = 0; i < contours.Size; i++)
            {

                try
                {
                    Rectangle rectangle = CvInvoke.BoundingRectangle(contours[i]);

                    double focus = CvInvoke.ContourArea(contours[i], true);
                    if (focus > m_focus)
                    {
                        int x = rectangle.X;
                        int y = rectangle.Y;
```

```
                    int w = rectangle.Width;
                    int h = rectangle.Height;

                    if (last == Point.Empty)
                    {
                        Point[] temp = new Point[1] { new Point { X = x, Y = y } };
                        vp.Push(temp);
                        last = temp[0];
                    }
                    else
                    {
                        Point[] temp = new Point[1] { new Point { X = x+w/2, Y = y+h/2 } };
                        vp.Push(temp);
                        next = temp[0];
                        drawlineenabled = true;
                    }
                    if (drawlineenabled)
                    {
                        for (var v = 1; v < vp.Size; v++)
                        {
                            CvInvoke.Line(frame1, vp[v - 1], vp[v], new MCvScalar(0, 255, 0),5);
                        }

                        last = next;
                    }
                    CvInvoke.Rectangle(frame1, new Rectangle(new Point(x, y), new Size(w, h)), new MCvScalar(255, 0, 0),
3, LineType.Filled);
                    if (txtOut.Text != "")
                    {
                        txtOut.AppendText(Environment.NewLine);
                    }
                    DateTime t = DateTime.UtcNow;

                    txtOut.AppendText(t +"\tMovement at: [" + x + "," + y + "]\t Area: [" + focus + "pixel]");
                    txtOut.ScrollToCaret();
                }
            }
            catch (Exception exp)
            {
                MessageBox.Show("Error at draw:" + exp.Message, "ERROR", MessageBoxButtons.OK);
            }
        }
        if (startMode != string.Empty)
        {
            imgBox.Image = frame1.Rotate(90, new Bgr(0, 0, 0));
        }
        else
        {
            /*Szerkeztett képkocka megjelenítése*/
            imgBox.Image = frame1;
        }
        /*Képkockák cseréje*/
        frame1 = frame2;
        try
        {
            frame1 = capture.QueryFrame().ToImage<Bgr, Byte>();

        }
        catch (Exception)
        {
        }
        txtFocus.Text = m_focus.ToString();
    }

}
/*START gomb*/
private void btnStart_Click(object sender, EventArgs e)
```

```
{
    btnClear.Enabled = false;
    btnStop.Enabled = true;
    btnStart.Enabled = false;
    if (startMode == string.Empty)
    {
        capture = new VideoCapture(0);
    }
    else
    {
        capture = new VideoCapture(startMode);
    }
    if (noRecordYet)
    {
        Application.Idle += ProcessUpdate;
        noRecordYet = !noRecordYet;
    }
    else
    {
        if (frame1 != null && frame2 != null)
        {
            Application.Idle += ProcessUpdate;
        }
    }

}
/*UP gomb*/
private void btnUp_Click(object sender, EventArgs e)
{
    m_focus += 25;
}
/*DOWN gomb*/
private void btnDown_Click(object sender, EventArgs e)
{
    if (m_focus > 0)
    {
        m_focus -= 25;
    }
}
/*CAMERA  rádiógomb*/
private void rbtnCamera_CheckedChanged(object sender, EventArgs e)
{
    if(rbtnCamera.Checked)
    {
        startMode = string.Empty;
        button1.Visible = false;
    }
}
/*File  rádiógomb*/
private void rbtnFile_CheckedChanged(object sender, EventArgs e)
{
    button1.Visible = true;
}
/*Choose File gomb*/
private void button1_Click(object sender, EventArgs e)
{
    using (OpenFileDialog openFileDialog = new OpenFileDialog())
    {
        openFileDialog.InitialDirectory = "c:\\";
        openFileDialog.Filter = "All files (*.*)|*.*";
        openFileDialog.FilterIndex = 1;
        openFileDialog.RestoreDirectory = true;

        if (openFileDialog.ShowDialog() == DialogResult.OK)
        {
            startMode = openFileDialog.FileName;
        }
```

```
            }
        }
        /*STOP gomb*/
        private void btnStop_Click(object sender, EventArgs e)
        {
            btnStop.Enabled = false;
            btnStart.Enabled = true;
            btnClear.Enabled = true;
            Application.Idle -= ProcessUpdate;
            vp.Clear();
        }
        /*CLEAR gomb*/
        private void btnClear_Click(object sender, EventArgs e)
        {

          var answer = MessageBox.Show("Do you want to save the recorded data?", "Clear",
MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question);
          if(answer == DialogResult.Yes)
          {
              if(saveFileDialog.ShowDialog() == DialogResult.OK)
              {
                  StreamWriter sr = new StreamWriter(File.Create(saveFileDialog.FileName));
                  sr.Write(txtOut.Text);
                  sr.Flush();
                  sr.Dispose();
                  sr.Close();
                  saveFileDialog.Dispose();
                  //txtOut.SaveFile(saveFileDialog.FileName);
              }
              txtOut.Clear();
              btnClear.Enabled = false;
          }
          else if(answer == DialogResult.No)
          {
              txtOut.Clear();
              btnClear.Enabled = false;
          }
          else if(answer == DialogResult.Cancel)
          {

          }
        }
    }
}
```