Название работы: Обнаружение движения

Цель работы: Целью данной работы является изучение методик обнаружения движения в видеопотоке

Постановка задачи:

Необходимо разработать приложение Windows Forms, способное осуществлять:

1. обнаружение подвижных объектов в видеопотоке

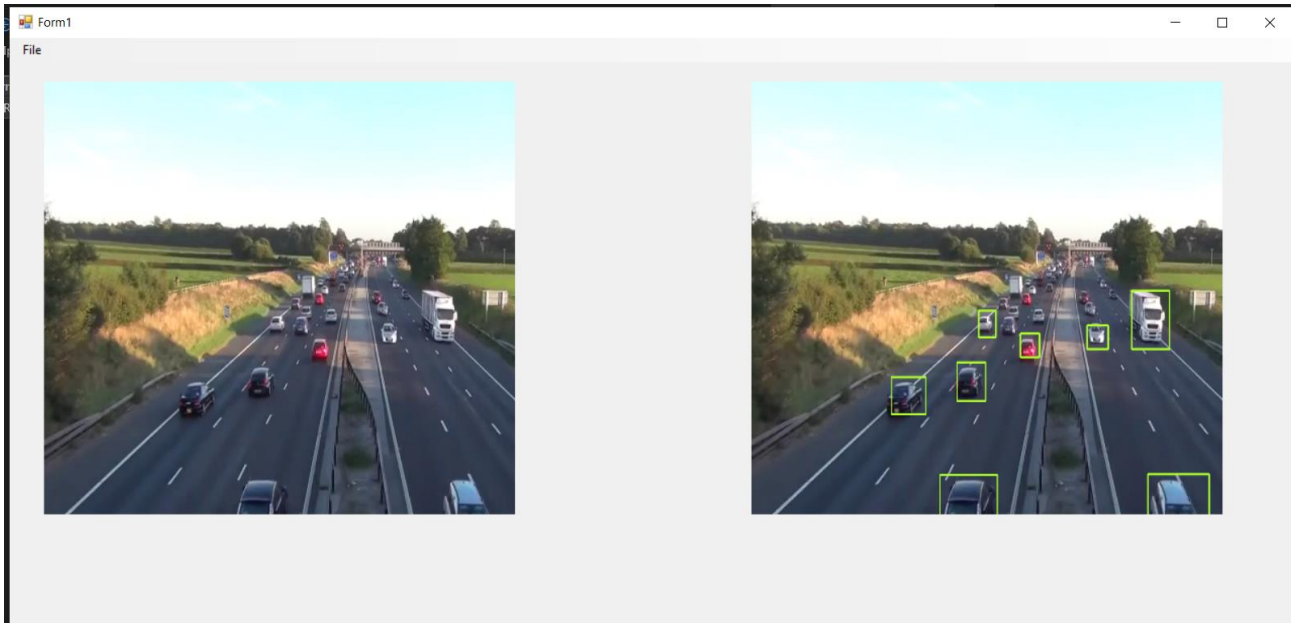
Задание:

Реализовать программное средство, позволяющее отображать в одном окне два изображения: «оригинальное» слева и «результат обработки» справа.

Реализовать интерфейс, позволяющий по нажатию на соответствующие кнопки выполнять следующие операции:

1. обнаружение и обозначение перемещающихся объектов в видеопотоке с веб-камеры;

2. обнаружение и обозначение перемещающихся объектов в видеопотоке из видеофайла.



Листинг кода:

```
using Emgu.CV.Structure;
using Emgu.CV;
using Emgu.CV.CvEnum;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;
using Emgu.CV.OCR;
using Emgu.CV.Util;
using static System.Windows.Forms.VisualStyles.VisualStyleElement.TrayNotify;
using Emgu.CV.Dnn;
using Emgu.CV.Ocl;
using Emgu.CV.BgSegm;

namespace Roviac_6
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private VideoCapture capture;

        BackgroundSubtractorMOG2 subtractor = new BackgroundSubtractorMOG2(1000, 32,
true);
        public void ProcessFrame(Image<Bgr, byte> sourceImage)
        {

            Image<Gray, Byte> grayFrame = sourceImage.Convert<Gray, Byte>();

            var foregroundMask = grayFrame.CopyBlank();
            subtractor.Apply(grayFrame, foregroundMask);

          //  imageBox2.Image = FilterMask(foregroundMask).Resize(621, 527,
Inter.Linear);

            VectorOfVectorOfPoint contours = new VectorOfVectorOfPoint();

            CvInvoke.FindContours(
            FilterMask(foregroundMask),
            contours,
            null,
            RetrType.External, // получение только внешних контуров
            ChainApproxMethod.ChainApproxTc89L1); // применение одной из
разновидностей
                                                  // алгоритма аппроксимации цепочки
Teh-Chin
            var resultImage = sourceImage.Clone();
            for (int i = 0; i < contours.Size; i++)
            {
                if (CvInvoke.ContourArea(contours[i]) > 700) //игнорирование маленьких
контуров
                {
```

```
                    Rectangle boundingRect = CvInvoke.BoundingRectangle(contours[i]);
                    resultImage.Draw(boundingRect, new Bgr(Color.GreenYellow), 2);
                }
            }
            imageBox2.Image = resultImage.Resize(621, 527, Inter.Linear);
        }

        private Image<Gray, byte> FilterMask(Image<Gray, byte> mask)
        {
            var anchor = new Point(-1, -1);
            var borderValue = new MCvScalar(1);
            // создание структурного элемента заданного размера и формы для
морфологических операций
            var kernel = CvInvoke.GetStructuringElement(ElementShape.Ellipse, new
Size(3, 3), anchor);
            // заполнение небольших тёмных областей
            var closing = mask.MorphologyEx(MorphOp.Close, kernel, anchor, 1,
BorderType.Default,
            borderValue);
            // удаление шумов
            var opening = closing.MorphologyEx(MorphOp.Open, kernel, anchor, 1,
BorderType.Default,
            borderValue);
            // расширение для слияния небольших смежных областей
            var dilation = opening.Dilate(7);
            // пороговое преобразование для удаления теней
            var threshold = dilation.ThresholdBinary(new Gray(240), new Gray(255));
            return threshold;
        }

        private void ProcessFrameVideo(object sender, EventArgs e)
        {
            var frame = new Mat();
            capture.Retrieve(frame); // получение текущего кадра
            Image<Gray, Byte> grayFrame = frame.ToImage<Bgr,byte>().Convert<Gray,
Byte>();

            var picture = frame.ToImage<Bgr, byte>();
            ProcessFrame(picture);
            imageBox1.Image = picture.Resize(621, 527, Inter.Linear);


            //ProcessFrame(grayFrame);
            imageBox1.Image = picture.Resize(621, 527,Inter.Linear);
            //imageBox2.Image = picture.Resize(621, 527, Inter.Linear); ;
            Thread.Sleep((int)capture.GetCaptureProperty(Emgu.CV.CvEnum.CapProp.Fps));
        }
        private void openVideoToolStripMenuItem_Click(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.Filter = "Video Files (*mp4, *.flv)| *.mp4;*.flv";
            var result = openFileDialog.ShowDialog(); // открытие диалога выбора файла
            if (result == DialogResult.OK) // открытие выбранного файла
            {
                string fileName = openFileDialog.FileName;
                capture = new VideoCapture(fileName);
                capture.ImageGrabbed += ProcessFrameVideo;
                capture.Start(); // начало обработки видеопотока
            }
        }


        private void ProcessFrameWebcam(object sender, EventArgs e)
        {
            var frame = new Mat();
```

```
            capture.Retrieve(frame); // получение текущего кадра
            imageBox1.Image = frame;

            ProcessFrame(frame.ToImage<Bgr,byte>());

        }

        private void openWebcamToolStripMenuItem_Click(object sender, EventArgs e)
        {
            capture = new VideoCapture();
            capture.ImageGrabbed += ProcessFrameWebcam;
            capture.Start(); // начало обработки видеопотока
        }
    }
}
```