Название работы: Поиск контуров и примитивов на изображении

Цель работы: Целью данной работы является изучение алгоритмов поиска контуров и примитивов на изображениях.

Постановка задачи:

Необходимо разработать приложение Windows Forms, способное осуществлять:

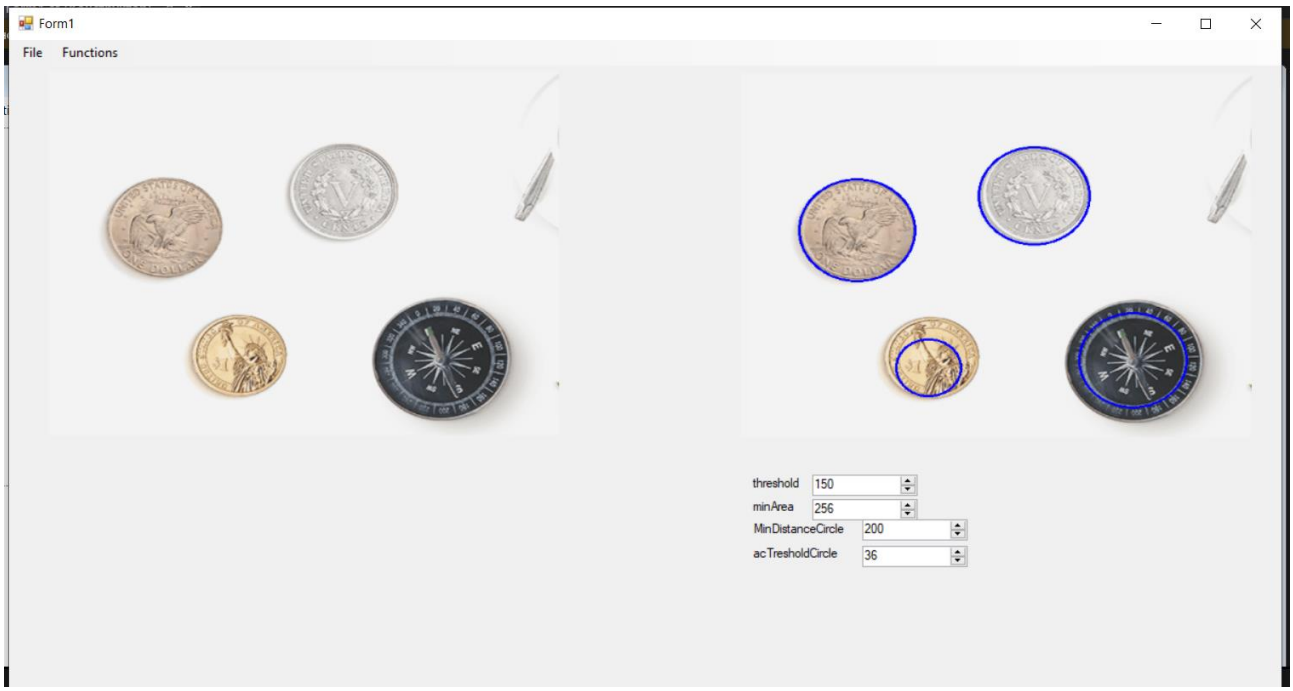
1. Обнаружение контуров.
2. Обнаружение примитивов (окружность, треугольник, четырёхугольник).

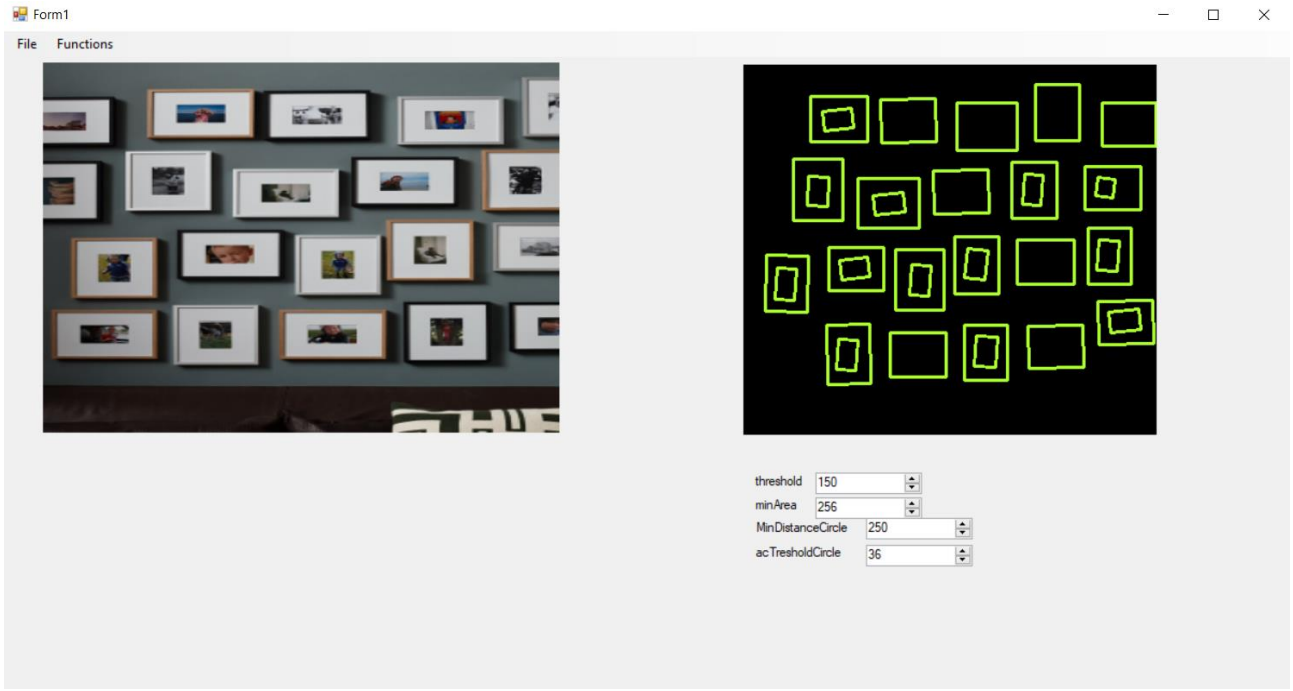
Задание:

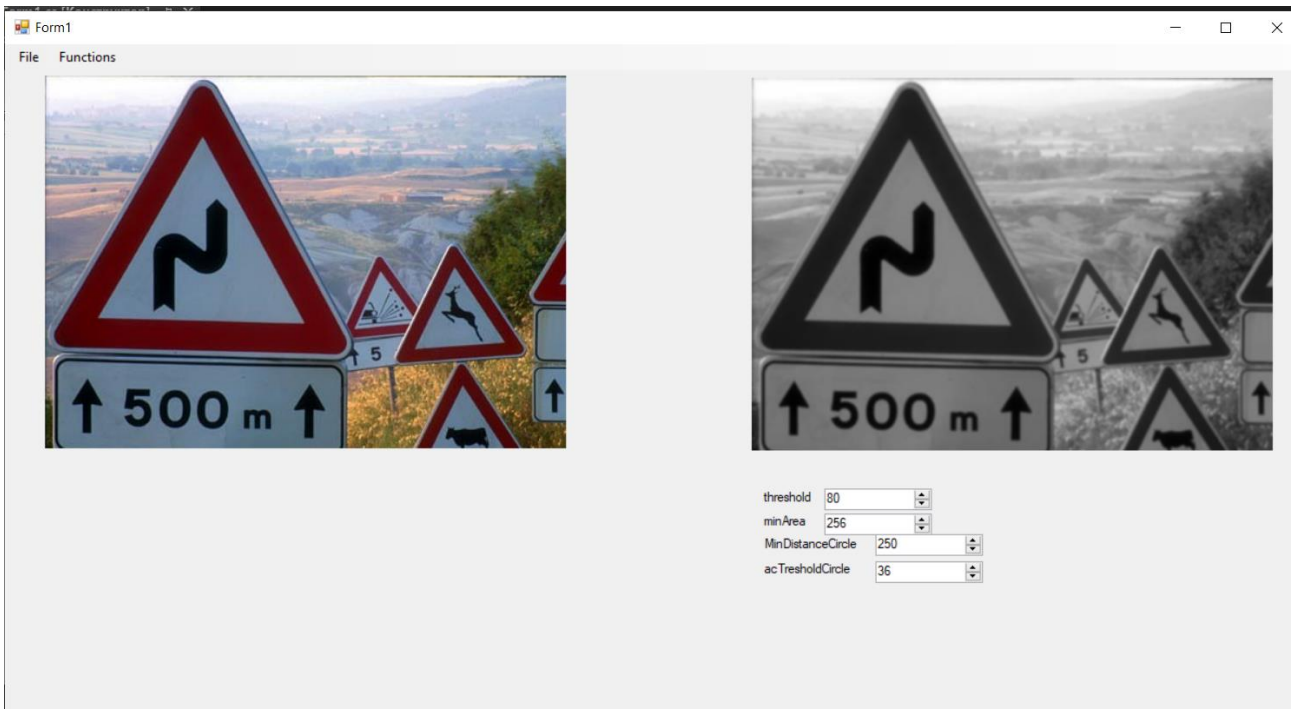
Реализовать программное средство, позволяющее отображать в одном окне два изображения,
«оригинальное» слева и «результат обработки» справа. Реализовать интерфейс, позволяющий по
нажатию на соответствующие кнопки выполнять следующие операции:

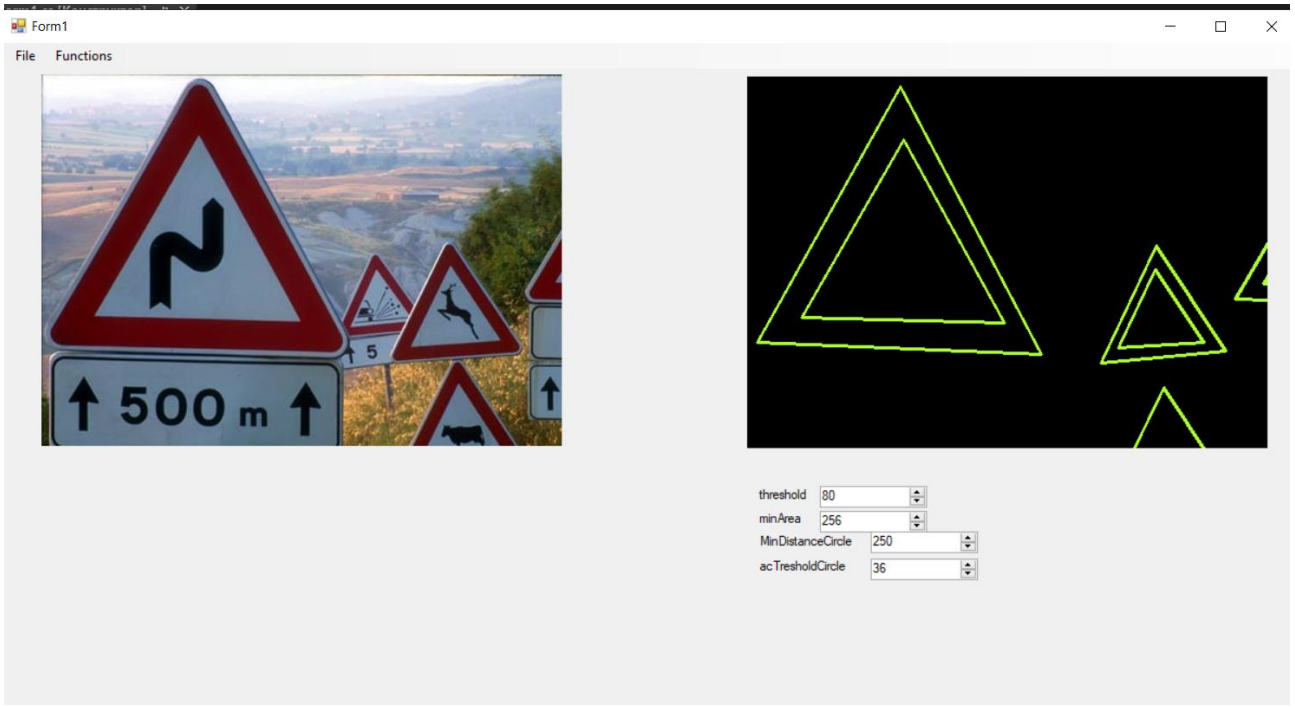
1. Предварительная обработка изображения для устранения шумов и перепадов яркости.
2. Поиск и отображение аппроксимированных контуров на изображении.
3. Поиск и отображение примитивов на изображении.
4. Возможность регулирования пользователем параметров порогового преобразования и
минимальной площади контуров.
5. Отображение числа примитивов на изображении выбранного вида

Листинг программы с комментариями:


Form1
File
Functions
threshold
150
minArea
256
MinDistanceCircle
200
acTresholdCircle
36


Form1
File
Functions
threshold
150
minArea
256
MinDistanceCircle
250
acTresholdCircle
36






```
using Emgu.CV;
using Emgu.CV.Structure;
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.Threading.Tasks;
using System.Windows.Forms;
using Emgu.CV.Util;
using static System.Windows.Forms.MonthCalendar;


//1.Предварительная обработка изображения для устранения шумов и
перепадов яркости. done
//2. Поиск и отображение аппроксимированных контуров на изображении. done
//3. Поиск и отображение примитивов на изображении. done
//4. Возможность регулирования пользователем параметров порогового
преобразования и
//минимальной площади контуров. done
//5. Отображение числа примитивов на изображении выбранного вида.

namespace Roviac_4
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        public Image<Bgr, byte> sourceImage { get; set; }
```

```
        private void openToolStripMenuItem_Click(object sender, EventArgs e)
        {
            OpenFileDialog ofd = new OpenFileDialog();
            var result = ofd.ShowDialog();
            if (result == DialogResult.OK)
            {
                string filename = ofd.FileName;
                sourceImage = new Image<Bgr, byte>(filename);
                imageBox1.Image = sourceImage.Resize(640, 422, Inter.Linear);
            }
        }

        private void removeToolStripMenuItem_Click(object sender, EventArgs e)
        {

        }

        public static Image<Gray, byte> GaussianBlur(Image<Bgr, byte> sourceImage)
        {
            var grayImage = sourceImage.Convert<Gray, byte>();
            int kernelSize = 5; // Радиус размытия
            var BluredImage = grayImage.SmoothGaussian(kernelSize);

            return BluredImage;
        }
        private void removeNoiseToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            imageBox2.Image =
GaussianBlur(sourceImage).Resize(640,422,Inter.Linear);
```

```
    }

    public static Image<Gray, byte> FindingAreasOfinterest(Image<Bgr, byte>
sourceImage, int threshold_)
    {
        var bluredImage = GaussianBlur(sourceImage);
        var threshold = new Gray(threshold_); // пороговое значение
        var color = new Gray(255); // этим цветом будут закрашены пиксели,
имеющие значение > threshold
        var binarizedImage = bluredImage.ThresholdBinary(threshold, color);
        return binarizedImage;
    }

    public static Image<Bgr, byte> FindCounters(Image<Bgr, byte> sourceImage,
int threshold)
    {
        var contours = new VectorOfVectorOfPoint(); // контейнер для хранения
контуров
        CvInvoke.FindContours(FindingAreasOfinterest(sourceImage, threshold), //
исходное чёрно-белое изображение
         contours, // найденные контуры
         null, // объект для хранения иерархии контуров (в данном случае не
используется)
         RetrType.List, // структура возвращаемых данных (в данном случае
список)
         ChainApproxMethod.ChainApproxSimple) ; // метод аппроксимации
(сжимает горизонтальные,
                                        //вертикальные и диагональные сегменты
                                        //и оставляет только их конечные точки)
```

```
        var contoursImage = sourceImage.CopyBlank(); //создание "пустой" копии
исходного изображения
        for (int i = 0; i < contours.Size; i++)
        {
            var points = contours[i].ToArray();
            contoursImage.Draw(points, new Bgr(Color.GreenYellow), 2); //
отрисовка точек
        }



        return contoursImage;
    }

    private void findContoursToolStripMenuItem_Click(object sender, EventArgs
e)
    {
        imageBox2.Image = FindCounters(sourceImage,
(int)thresholdValue.Value).Resize(640, 422, Inter.Linear);
    }



    public static bool isRectangle(ref Point[] points)
    {
        int delta = 10; // максимальное отклонение от прямого угла
        LineSegment2D[] edges = PointCollection.PolyLine(points, true);
        for (int i = 0; i < edges.Length; i++) // обход всех ребер контура
        {
            double angle = Math.Abs(edges[(i + 1) %
            edges.Length].GetExteriorAngleDegree(edges[i]));
            if (angle < 90 - delta || angle > 90 + delta) // если угол непрямой
```

```
            {
                return false;
            }
        }
        return true;
    }

    //public static Image<Bgr, byte> Contour_approximations(Image<Bgr, byte> sourceImage,int threshold,int minArea_)
    //{
    //    var grayImage = sourceImage.Convert<Gray, byte>(); // Для окружностей
    //    var bluredImage = grayImage.SmoothGaussian(9);


    //    var image = FindingAreasOfinterest(sourceImage,threshold); // Получили чб пикчу

    //    /////// Нашли контуры //////////

    //    var contours = new VectorOfVectorOfPoint(); // контейнер для хранения контуров
    //    CvInvoke.FindContours(
    //    image, // исходное чёрно-белое изображение
    //     contours, // найденные контуры
    //     null, // объект для хранения иерархии контуров (в данном случае не используется)
    //     RetrType.List, // структура возвращаемых данных (в данном случае список)
    //     ChainApproxMethod.ChainApproxSimple);
    //    ////////////////////////////////////////
```

```
//    var contoursImage = sourceImage.CopyBlank(); //создание "пустой" копии исходного изображения
//    var minArea = minArea_;
//    for (int i = 0; i < contours.Size; i++)
//    {
//        var approxContour = new VectorOfPoint();
//        CvInvoke.ApproxPolyDP(
//        contours[i], // исходный контур
//        approxContour, // контур после аппроксимации
//        CvInvoke.ArcLength(contours[i], true) * 0.05, // точность аппроксимации, прямо
//                                                      //пропорциональная площади контура
//        true); // контур становится закрытым (первая и последняя точки соединяются)

//        var points = approxContour.ToArray();
//        if ((CvInvoke.ContourArea(approxContour, false) > minArea) && (approxContour.Size == 3)) // Треугольники
//        {
//            contoursImage.Draw(new Triangle2DF(points[0], points[1], points[2]), new Bgr(Color.GreenYellow), 2);
//        }

//        if (isRectangle(ref points) && (CvInvoke.ContourArea(approxContour, false) > minArea)) // Прямоугольники
//        {
//            contoursImage.Draw(CvInvoke.MinAreaRect(approxContour), new Bgr(Color.GreenYellow), 2);
//        }
```

```
//        //if (approxContour.Size == 3) // если контур содержит 3 точки, то рисуется треугольник
//        //{
//        //    var points = approxContour.ToArray();
//        //    contoursImage.Draw(new Triangle2DF(points[0], points[1], points[2]),new Bgr(Color.GreenYellow), 2);
//        //}
//        //if (CvInvoke.ContourArea(approxContour, false) > minArea)
//        //{

//        //}

//    }

//    return contoursImage;
//}



        private void findPrimitivesToolStripMenuItem_Click(object sender, EventArgs e)
        {
            //imageBox2.Image = Contour_approximations(sourceImage,(int)thresholdValue.Value, (int)minAreaValue.Value).Resize(640, 422, Inter.Linear);
        }

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

```
        }

        public static Image<Bgr, byte> Contour_approximations_circle(Image<Bgr,
byte> sourceImage, int minDistance_, int acTreshold_)
        {
            var grayImage = sourceImage.Convert<Gray, byte>();
            var bluredImage = grayImage.SmoothGaussian(9);

            var minDistance = minDistance_;
            var acTreshold = acTreshold_;
            List<CircleF> circles = new
List<CircleF>(CvInvoke.HoughCircles(bluredImage,
            HoughModes.Gradient,
            1.0,
            minDistance, // Мин дистанция между центрами
            100,
            acTreshold, // Пороговое значение при определнии центра
            10, // Мин радиус
            100)); // Макс радиус

            var resultImage = sourceImage.Copy();
            foreach (CircleF circle in circles) resultImage.Draw(circle, new
Bgr(Color.Blue), 2);

            return resultImage;
        }

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

```
        imageBox2.Image = Contour_approximations_circle(sourceImage,
(int)minDistanceCircleValue.Value, (int)acTresholdValue.Value).Resize(640, 422,
Inter.Linear);
    }

    public static Image<Bgr, byte> Contour_approximations_triangle(Image<Bgr,
byte> sourceImage, int threshold, int minArea_)
    {
        var image = FindingAreasOfinterest(sourceImage, threshold); // Получили чб
пикчу

        /////// Нашли контуры //////////

        var contours = new VectorOfVectorOfPoint(); // контейнер для хранения
контуров
        CvInvoke.FindContours(
        image, // исходное чёрно-белое изображение
         contours, // найденные контуры
         null, // объект для хранения иерархии контуров (в данном случае не
используется)
         RetrType.List, // структура возвращаемых данных (в данном случае
список)
         ChainApproxMethod.ChainApproxSimple);
        ////////////////////////////////////////////

        var contoursImage = sourceImage.CopyBlank(); //создание "пустой" копии
исходного изображения
        var minArea = minArea_;
        for (int i = 0; i < contours.Size; i++)
        {
```

```
            var approxContour = new VectorOfPoint();
            CvInvoke.ApproxPolyDP(
            contours[i], // исходный контур
            approxContour, // контур после аппроксимации
            CvInvoke.ArcLength(contours[i], true) * 0.05, // точность аппроксимации, прямо
                                                          //пропорциональная площади контура
            true); // контур становится закрытым (первая и последняя точки соединяются)

            var points = approxContour.ToArray();
            if ((CvInvoke.ContourArea(approxContour, false) > minArea) && (approxContour.Size == 3)) // Треугольники
            {
                contoursImage.Draw(new Triangle2DF(points[0], points[1], points[2]), new Bgr(Color.GreenYellow), 2);
            }
        }
        return contoursImage;
    }

    private void triangleToolStripMenuItem_Click(object sender, EventArgs e)
    {
        imageBox2.Image = Contour_approximations_triangle(sourceImage, (int)thresholdValue.Value, (int)minAreaValue.Value);
    }


    public static Image<Bgr, byte> Contour_approximations_rectangle(Image<Bgr, byte> sourceImage, int threshold, int minArea_)
```

```
    {
        var image = FindingAreasOfinterest(sourceImage, threshold); // Получили чб пикчу

        /////// Нашли контуры //////////

        var contours = new VectorOfVectorOfPoint(); // контейнер для хранения контуров
        CvInvoke.FindContours(
        image, // исходное чёрно-белое изображение
         contours, // найденные контуры
         null, // объект для хранения иерархии контуров (в данном случае не используется)
         RetrType.List, // структура возвращаемых данных (в данном случае список)
         ChainApproxMethod.ChainApproxSimple);
        ////////////////////////////////////////

        var contoursImage = sourceImage.CopyBlank(); //создание "пустой" копии исходного изображения
        var minArea = minArea_;
        for (int i = 0; i < contours.Size; i++)
        {
           var approxContour = new VectorOfPoint();
           CvInvoke.ApproxPolyDP(
           contours[i], // исходный контур
           approxContour, // контур после аппроксимации
           CvInvoke.ArcLength(contours[i], true) * 0.05, // точность аппроксимации, прямо
                                                         //пропорциональная площади контура
```

```
            true); // контур становится закрытым (первая и последняя точки соединяются)

            var points = approxContour.ToArray();
            if (isRectangle(ref points) && (CvInvoke.ContourArea(approxContour, false) > minArea)) // Прямоугольники
            {
                contoursImage.Draw(CvInvoke.MinAreaRect(approxContour), new Bgr(Color.GreenYellow), 2);
            }
        }
        return contoursImage;
    }

    private void rectangleToolStripMenuItem_Click(object sender, EventArgs e)
    {
        imageBox2.Image = Contour_approximations_rectangle(sourceImage, (int)thresholdValue.Value, (int)minAreaValue.Value);
    }
  }
}
```