

Лабораторно-практическое задание

№ 3

«Системы технического зрения в автоматизированных системах»

Сайфудинов Р.Р. 171-261

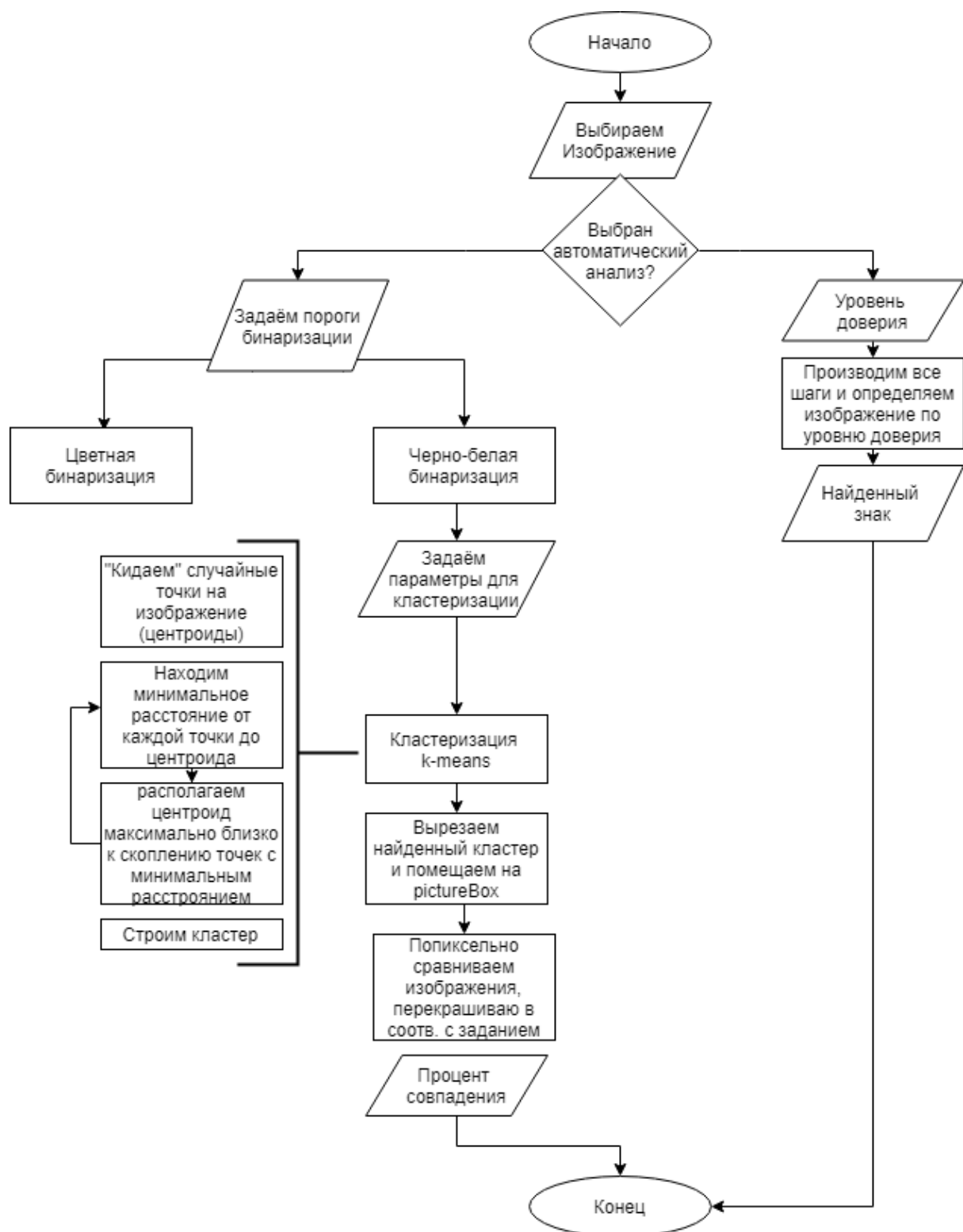
Тема: Базовые принципы поиска шаблонов на изображении.

Цель работы: Разработать алгоритм обработки изображения для поиска стандартизированных шаблонов

Задачи:

- Подготовить приложение для показа изображений из рабочей папки с выбором изображения из списка;
- Разработать функцию установки и применения цветового ключа для бинаризации;
- Разработать функцию локализации объектов интереса с помощью кластеризации и применения структурного (размерного) фильтра для отбора целевых объектов;
- Реализовать функциональность формирования списка целевых объектов и возможности просмотра (подсветки) их расположения на исходном изображении;
- Разработать функцию копирования целевого объекта из исходного изображения и его масштабирования до размеров шаблона;
- Разработать функцию сопоставления целевого объекта с шаблоном, в том числе с применением цветовых и структурных ключей (задания зон обработки). Сформировать карту визуализации областей совпадения целевого объекта с шаблоном.
- Реализовать функцию автоматического распознавания шаблонов на исходном изображении при их открытии.

Блок схема:



Скриншоты:

Form1

Выбор

Выбрать файл

Signs.jpg

Бинаризация

min max

r 0 0 r красный

g 0 0 g синий

b 0 0 b черно-цветное

черно-белое

Кластеризация K-means

Rmin 10 Rmax 50 Density 20 K-count 2

Кластеризовать

Распознавание

Sign

Template

STOP

Совпало

Сопоставить изображения

Уровень доверия 73

☐ Анализ

Выбираем файл

Form1

Выбор

Выбрать файл

Signs.jpg

Бинаризация

min max

r 116 255 r красный

g 21 60 g синий

b 16 255 b черно-цветное

черно-белое

Кластеризация K-means

Rmin 10 Rmax 50 Density 20 K-count 2

Кластеризовать

Распознавание

Sign

Template

STOP

Совпало

Сопоставить изображения

Уровень доверия 73

☐ Анализ

Бинаризация по красному

Form1

Выбор

Выбрать файл

Signs.jpg

Бинаризация

min max

r 116 255 r

g 21 60 g

b 16 255 b

красный

черно-белое

синий

черно-цветное

Кластеризация K-means

Rmin 10 Rmax 50 Density 20 K-count 2

Кластеризовать

Распознавание

Sign Template

Совпало

Сопоставить изображения

Уровень доверия 73

☐ Анализ

Бинаризация цветная

Form1

Выбор

Выбрать файл

Signs.jpg

Бинаризация

min max

r 116 255 r

g 21 60 g

b 16 255 b

красный

черно-белое

синий

черно-цветное

Кластеризация K-means

Rmin 10 Rmax 50 Density 20 K-count 2

Кластеризовать

Распознавание

Sign Template

Совпало

Сопоставить изображения

Уровень доверия 73

☐ Анализ

Кластеризация(k-means)

Form1

Выбор

Выбрать файл

Signs.jpg

Бинаризация

min max

r 116 255 r красный черно-белое

g 21 60 g синий черно-цветное

b 16 255 b e

Кластеризация K-means

Rmin 10 Rmax 50 Density 20 K-count 2

Кластеризовать

0 1

Распознавание

Sign Template

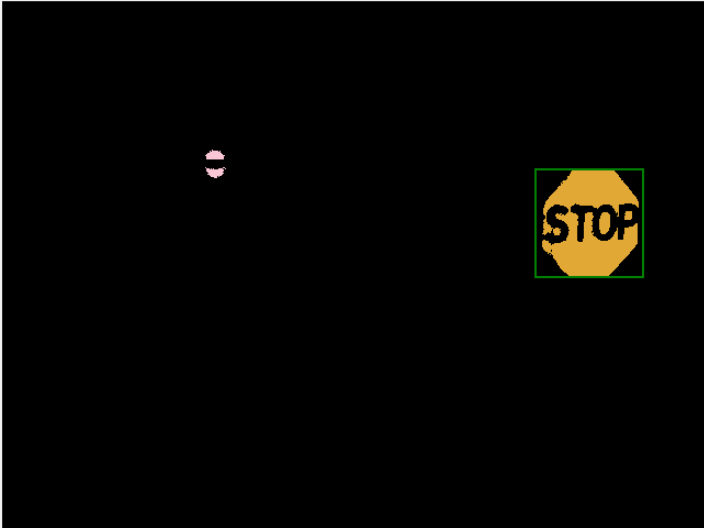
STOP STOP

Совпало 74%

Сопоставить изображения

Уровень доверия 73

☐ Анализ



Сравнение с шаблоном

Form1

Выбор

Выбрать файл

Signs.jpg

Бинаризация

min max

r 116 255 r красный черно-белое

g 21 60 g синий черно-цветное

b 16 255 b e

Кластеризация K-means

Rmin 10 Rmax 50 Density 20 K-count 2

Кластеризовать

0 1

Распознавание

Sign Template


STOP STOP

Совпало 74%

Сопоставить изображения

Уровень доверия 73

☒ Анализ



Анализ

Код программы:

```
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 System.IO;
using System.Media;

namespace lab3
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            pictureBoxTemplate.Load(Convert.ToString(@"C:\Users\desgr\source\repos\lab3\Signs\STOP
SIGNS\STOP.bmp"));
            Bitmap buffer;
            Bitmap buffer_mask;
            int k, R_min, R_max, Density;
            //ВЫБОР ПАПКИ
            private void button1_Click(object sender, EventArgs e)
            {
                FolderBrowserDialog FDB = new FolderBrowserDialog();
                FDB.SelectedPath = @"C:\Users\desgr\source\repos\lab3\Signs\";
                if (FDB.ShowDialog() == DialogResult.OK) //если нажали ок то выбрали папку
                {
                    listBox1.Items.Clear();
                    string[] files = Directory.GetFiles(FDB.SelectedPath); //массив из файлов
                    foreach (string file in files)
                    {
                        listBox1.Items.Add(Path.GetFileName(file));
                    }
                }
            }

            //ВЫБОР СИГНАЛА
            private void listBox1_DoubleClick(object sender, EventArgs e)
            {
                try
                {
                    if (listBox1.SelectedItem.ToString() == "Signs.jpg")
                    {
                        pictureBox1.Invalidate();

                        pictureBox1.Load(Convert.ToString(@"C:\Users\desgr\source\repos\lab3\Signs\Signs.jpg"));
                        // display picture
                        buffer = new
```

```

Bitmap(Convert.ToString(@"C:\Users\desgr\source\repos\lab3\Signs\Signs.jpg")); // buffer
object
    }
    if (listBox1.SelectedItem.ToString() == "Signs2.jpg")
    {
        pictureBox1.Invalidate();

pictureBox1.Load(Convert.ToString(@"C:\Users\desgr\source\repos\lab3\Signs\Signs2.jpg"));
// display picture
        buffer = new
Bitmap(Convert.ToString(@"C:\Users\desgr\source\repos\lab3\Signs\Signs2.jpg")); //
buffer object
    }
    if (listBox1.SelectedItem.ToString() == "Signs3.jpeg")
    {
        pictureBox1.Invalidate();

pictureBox1.Load(Convert.ToString(@"C:\Users\desgr\source\repos\lab3\Signs\Signs3.jpeg"));
; // display picture
        buffer = new
Bitmap(Convert.ToString(@"C:\Users\desgr\source\repos\lab3\Signs\Signs3.jpeg")); //
buffer object
    }
    analysis();

}
catch
{
    MessageBox.Show("ERROR", "ERROR", MessageBoxButtons.OK,
    MessageBoxIcon.Information);
}
}

//Предустановки
private void button4_Click(object sender, EventArgs e) { Rmin.Value = 0;
Rmax.Value = 61; Gmin.Value = 0; Gmax.Value = 122; Bmin.Value = 69; Bmax.Value = 255; }
private void button3_Click(object sender, EventArgs e) { Rmin.Value = 116;
Rmax.Value = 255; Gmin.Value = 21; Gmax.Value = 60; Bmin.Value = 16; Bmax.Value = 255; }

//БИНАР. ЦВЕТНАЯ
List<List<int>> white = new List<List<int>>(); // [num][x,y]
private void button2_Click(object sender, EventArgs e)
{
    try
    {
        buffer_mask = new Bitmap(buffer);
        for (int x = 0; x < 640; x++)
        {
            for (int y = 0; y < 480; y++)
            {
                int b = buffer_mask.GetPixel(x, y).B;
                int g = buffer_mask.GetPixel(x, y).G;
                int r = buffer_mask.GetPixel(x, y).R;

                if (b >= Bmin.Value && b <= Bmax.Value && g >= Gmin.Value && g <=
Gmax.Value && r >= Rmin.Value && r <= Rmax.Value)

```

```

        {
        }
        else
        {
            buffer_mask.SetPixel(x, y, Color.Black);
        }
    }
    pictureBox1.Image = buffer_mask;
}
catch { MessageBox.Show("Error", "ERROR", MessageBoxButtons.OK,
    MessageBoxIcon.Information); }
}

//Бинар. ЧЕРНО-БЕЛАЯ
private void button5_Click(object sender, EventArgs e)
{
    white.Clear();
    try
    {
        buffer_mask = new Bitmap(buffer);
        int i = 0;
        for (int x = 0; x < 640; x++)
        {
            for (int y = 0; y < 480; y++)
            {
                int b = buffer_mask.GetPixel(x, y).B;
                int g = buffer_mask.GetPixel(x, y).G;
                int r = buffer_mask.GetPixel(x, y).R;

                if (b >= Bmin.Value && b <= Bmax.Value && g >= Gmin.Value && g <=
Gmax.Value && r >= Rmin.Value && r <= Rmax.Value)
                {
                    buffer_mask.SetPixel(x, y, Color.White);
                    white.Add(new List<int>());
                    white[i].Add(x);
                    white[i].Add(y);
                    i++;
                }
                else
                {
                    buffer_mask.SetPixel(x, y, Color.Black);
                }
            }
        }
        pictureBox1.Image = buffer_mask;
    }
    catch { MessageBox.Show("Error", "ERROR", MessageBoxButtons.OK,
        MessageBoxIcon.Information); }
}

//Задание3
Bitmap bmp_clusters;

```



```

List<Cluster> clusters = new List<Cluster>(); // лист с кластерами от класса
private void button6_Click(object sender, EventArgs e)
{
    k = (int)numericUpDown4.Value;
    R_min = (int)numericUpDown1.Value;
    R_max = (int)numericUpDown2.Value;
    Density = (int)numericUpDown3.Value;
    clusters.Clear();

    // кидаем случайные центроиды
    Random rand = new Random();
    for (int n = 0; n < k; n++)
    {
        clusters.Add(new Cluster());
        clusters[n].centroidX = white[rand.Next(640)][0]; // x
        clusters[n].centroidY = white[rand.Next(480)][1]; // y
    }

    double len, lenOld;
    //кол-во повторений поиска центроидов
    for (int n = 0; n < 10; n++)
    {
        // Очищаем лист
        for (int c = 0; c < k; c++) { clusters[c].points.Clear(); }

        //Распределение точек на кластеры
        for (int j = 0; j < white.Count; j++)
        {
            int x2 = white[j][0];
            int y2 = white[j][1];

            // Ищем расстояние от точки до центроида
            int x1 = clusters[0].centroidX;
            int y1 = clusters[0].centroidY;
            len = Math.Sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1));
            lenOld = len;
            int clNum = 0;

            // далее проходим по всем центроидам
            for (int c = 1; c < k; c++)
            {
                x1 = clusters[c].centroidX;
                y1 = clusters[c].centroidY;
                // calc length to centroid
                len = Math.Sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1));
                if (len < lenOld) { lenOld = len; clNum = c; }
            }

            // если точка вошла то добавляем её в список
            if (clNum != -1)
            {
                clusters[clNum].points.Add(j);
            }
        }
    }
}

```

```

// считаем среднее ариф. для центроида
for (int c = 0; c < k; c++)
{
    if (clusters[c].points.Count != 0)
    {
        clusters[c].centroidX = 0;
        clusters[c].centroidY = 0;
    }
    // count all points
    foreach (int point in clusters[c].points)
    {
        clusters[c].centroidX += white[point][0];
        clusters[c].centroidY += white[point][1];
    }
    // calculate average (new) centroid
    if (clusters[c].points.Count != 0)
    {
        clusters[c].centroidX = clusters[c].centroidX /
clusters[c].points.Count;
        clusters[c].centroidY = clusters[c].centroidY /
clusters[c].points.Count;
    }
}

// Закрашиваем случайным цветом найденный кластер
bmp_clusters = new Bitmap(buffer_mask);
for (int i = 0; i < k; i++)
{
    Color color = Color.FromArgb(rand.Next(255), rand.Next(255),
rand.Next(255));
    for (int c = 0; c < clusters[i].points.Count; c++)
    {
        int x = white[clusters[i].points[c]][0];
        int y = white[clusters[i].points[c]][1];
        bmp_clusters.SetPixel(x, y, color);
    }
}
pictureBox1.Image = bmp_clusters;

// Вычисляем радиус и заполненность кластера
for (int i = 0; i < k; i++)
{
    for (int c = 0; c < clusters[i].points.Count; c++)
    {
        if (Math.Abs(white[clusters[i].points[c]][0] -
clusters[i].centroidX) > clusters[i].R ||
            Math.Abs(white[clusters[i].points[c]][1] -
clusters[i].centroidY) > clusters[i].R)
        {
            clusters[i].R =
Math.Max(Math.Abs(white[clusters[i].points[c]][0] - clusters[i].centroidX),
Math.Abs(white[clusters[i].points[c]][1] - clusters[i].centroidY));
        }
    }
}

```

```

        }
        clusters[i].Density = (double)clusters[i].points.Count * 100 /
(double)(4 * clusters[i].R * clusters[i].R);
    }

    // Проверка на условия кластера
    List<int> cluster_paint = new List<int>();
    for (int i = 0; i < k; i++)
    {
        if (clusters[i].R < R_max && clusters[i].R > R_min &&
clusters[i].Density > Density) { cluster_paint.Add(i); }
    }
    // если прошел, то заносим в листбокс
    listBox2.DataSource = cluster_paint;
}

//ЗАДАНИЕ4
private void listBox2_SelectedIndexChanged(object sender, EventArgs e)
{
    if (listBox2.SelectedIndex != -1)
    {
        Bitmap rect_bmp = new Bitmap(bmp_clusters);
        Graphics rect_graph = Graphics.FromImage(rect_bmp);
        try
        {
            int num = listBox2.SelectedIndex;
            Rectangle rect = new Rectangle(clusters[num].centroidX -
clusters[num].R, clusters[num].centroidY - clusters[num].R, 2*clusters[num].R, 2*
clusters[num].R);
            rect_graph.DrawRectangle(new Pen(Color.Green, 2), rect);
            pictureBox1.Image = rect_bmp;
            Bitmap bmp_crop = new Bitmap(buffer.Clone(rect, buffer.PixelFormat),
new Size(80,80));
            pictureBoxCrop.Image = bmp_crop;
        }
        catch (Exception exc) { Console.WriteLine(exc.Message); }
    }
}

private Bitmap Binarization(Bitmap bmp_bin, int bmin, int bmax, int gmin, int
gmax, int rmin, int rmax) // бинаризация шаблона
{
    if (bmp_bin != null)
    {
        for (int x = 0; x < 80; x++)
        {
            for (int y = 0; y < 80; y++)
            {
                int b = bmp_bin.GetPixel(x, y).B;
                int g = bmp_bin.GetPixel(x, y).G;
                int r = bmp_bin.GetPixel(x, y).R;

                if (b >= bmin && b <= bmax &&

```

```

        g >= gmin && g <= gmax &&
        r >= rmin && r <= rmax)
    {
        bmp_bin.SetPixel(x, y, Color.White);
    }
    else
    {
        bmp_bin.SetPixel(x, y, Color.Black);
    }
    }
}
}
return bmp_bin;
}
//ЗАДАНИЕ 5
int Result;
private void Button7_Click(object sender, EventArgs e) { if (buffer != null) {
Compare(); } }
private int Compare()
{
    Bitmap bmp_check_temp= new
Bitmap(@"C:\Users\desgr\source\repos\lab3\Signs\STOP SIGNS\STOP_binar.jpg");
    Bitmap bmp_check_crop = Binarization(new Bitmap(pictureBoxCrop.Image),
0,80,0,80,100,255);
    Bitmap bmp_check_temp_square = Binarization(new
Bitmap(Convert.ToString(@"C:\Users\desgr\source\repos\lab3\Signs\STOP
SIGNS\circle.bmp")), 0,80,0,80,100,255);
    Bitmap bmp_check_res = new Bitmap(80, 80);
    int equal = 0; // count equaling pixels
    int outside = 0;

    for (int x = 0; x < 80; x++)
    {
        for (int y = 0; y < 80; y++)
        {
            if (bmp_check_crop.GetPixel(x, y) == Color.FromArgb(255, 255, 255) &&
                bmp_check_temp_square.GetPixel(x, y) == Color.FromArgb(255, 255,
255)) { bmp_check_res.SetPixel(x, y, Color.LightBlue); } // светло-синий - пиксель белый
            if (bmp_check_crop.GetPixel(x, y) == Color.FromArgb(255, 255, 255) &&
                bmp_check_temp.GetPixel(x, y) == Color.FromArgb(255, 255, 255)) {
bmp_check_res.SetPixel(x, y, Color.LightGreen); equal++; } // светло-зеленый - белый и
белый
            if (bmp_check_crop.GetPixel(x, y) == Color.FromArgb(0, 0, 0) &&
                bmp_check_temp.GetPixel(x, y) == Color.FromArgb(0, 0, 0)) {
bmp_check_res.SetPixel(x, y, Color.DarkGreen); equal++; } // темно-зеленый - черный и
черный
            if (bmp_check_crop.GetPixel(x, y) == Color.FromArgb(0, 0, 0) &&
                bmp_check_temp.GetPixel(x, y) == Color.FromArgb(255, 255, 255)) {
bmp_check_res.SetPixel(x, y, Color.Red); } // светло-красный
            if (bmp_check_crop.GetPixel(x, y) == Color.FromArgb(255, 255, 255) &&
                bmp_check_temp.GetPixel(x, y) == Color.FromArgb(0, 0, 0)) {
bmp_check_res.SetPixel(x, y, Color.DarkRed); } // темно-красный
            if (bmp_check_crop.GetPixel(x, y) == Color.FromArgb(0, 0, 0) &&
                bmp_check_temp_square.GetPixel(x, y) == Color.FromArgb(0, 0, 0))

```

```

{ bmp_check_res.SetPixel(x, y, Color.DarkBlue); outside++; } // темно-синий
    }
}
pictureBinarizationResult.Image = bmp_check_res;
Result = equal * 100 / (80 * 80 - outside);
textBox1.Text = Result.ToString()+"%";
return Result;
}

//ЗАДАНИЕ 6
public void analysis()
{
    if ((checkBox6.Checked == true)&&(pictureBox1.Image !=null))
    {
        button3_Click(null, null); //предустановка на красный цвет
        button5_Click(null,null); //вызов бинаризации
        button6_Click(null, null); //вызов кластеризации
        int trustLevel = (int)numericUpDown5.Value;
        List<int> signs = new List<int>();
        for (int i = 0; i < listBox2.Items.Count; i++)
        {
            listBox2.SelectedIndex = i;
            if (Compare() > trustLevel) { signs.Add(i); }
        }
        listBox2.SelectedIndex = -1;
        pictureBox1.Image = buffer;
        foreach (int sign in signs)
        {
            listBox2.SelectedIndex = sign;
            if (listBox2.SelectedIndex != -1)
            {
                Bitmap bmp_detect = new Bitmap(buffer);
                Graphics rect_graph = Graphics.FromImage(bmp_detect);
                try
                {
                    int num = listBox2.SelectedIndex;
                    Rectangle rect = new Rectangle(clusters[num].centroidX -
clusters[num].R, clusters[num].centroidY - clusters[num].R, 2 * clusters[num].R, 2 *
clusters[num].R);

                    rect_graph.DrawRectangle(new Pen(Color.Green, 2), rect);
                    rect_graph.DrawString("СТОП",
new Font("Roboto", 15),
new SolidBrush(Color.Green),
new PointF(clusters[num].centroidX - clusters[num].R - 5,
clusters[num].centroidY - clusters[num].R - 20));
                    pictureBox1.Image = bmp_detect;
                }
                catch (Exception exc) { Console.WriteLine(exc.Message); }
            }
        }
    }
}

public class Cluster
{
    public int centroidX = 0;

```

```
    public int centroidY = 0;
    public List<int> points = new List<int>();

    public int R = 0;
    public double Density = 0;
}
}
```