Лабораторно-практическое задание № 4

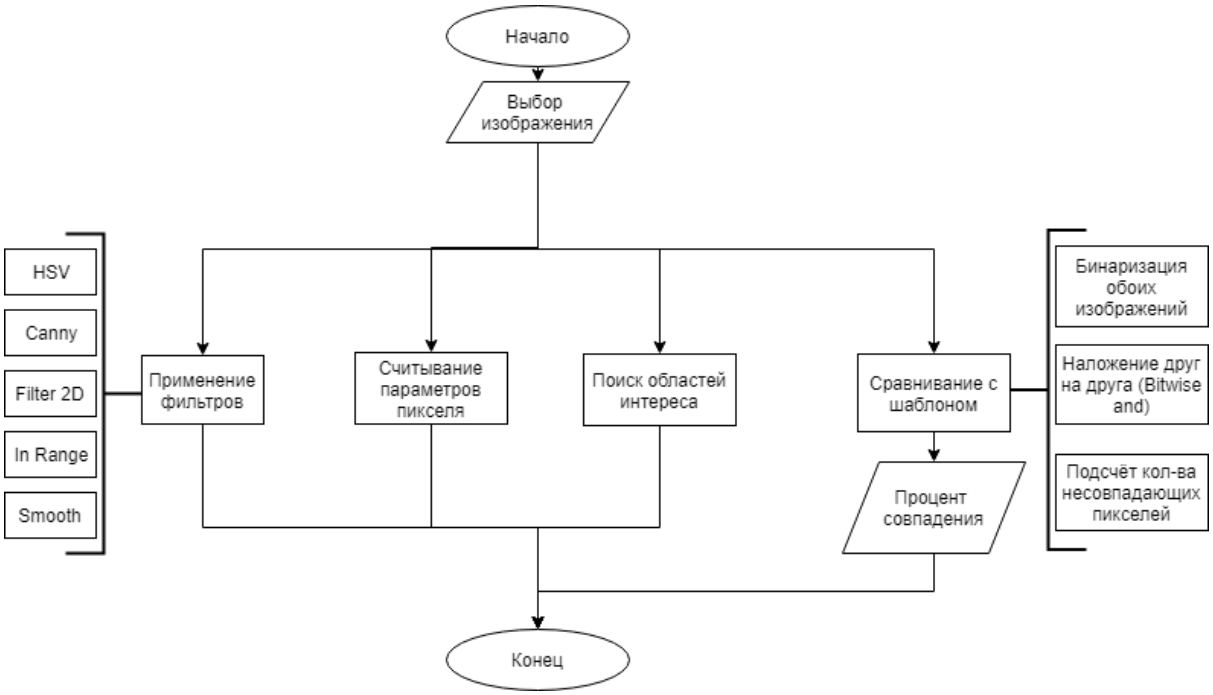
«Системы технического зрения в автоматизированных системах»

Сайфудинов Р.Р. 171-261

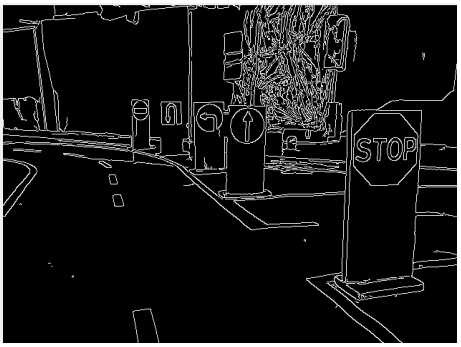
Тема:Техническое зрение в задачах управления.
Цель работы: Разработать программное обеспечение для поиска объектов интереса в видеопотоке и визуализации информации по детектированию
Задачи:

- Разработать приложение получения изображения с видеокамеры, видеофайла или файла-изображения с использованием библиотеки OpenCV (OpenCVSharp);
- Реализовать функции преобразования цветовых пространств, бинаризации, фильтрации (свёрточной), оконтуривания (Canny);
- Реализовать функцию считывания параметров пикселя под курсором;
- Разработать функцию пользовательской попиксельной фильтрации изображения по цветовому ключу и с использованием логических операций над изображением (маскирование через Cv.And);
- Реализовать функцию поиска контуров, детектирования и подсветки (выделения нарисованным прямоугольником) объектов по замкнутым контурам (Cv.MinEnclosingCircle и Cv. boundingRect) с выводом параметров объекта (координаты и размер);
- Реализовать функцию вырезки и масштабирования объекта, а также попиксельного сравнения с заданными шаблонами.

Блок схема:

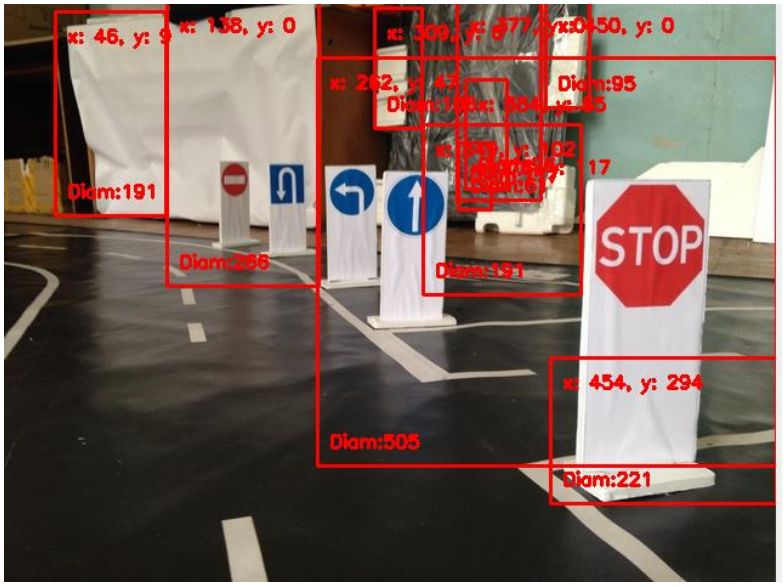

Начало
Выбор изображения
HSV
Canny
Filter 2D
In Range
Smooth
Применение фильтров
Считывание параметров пикселя
Поиск областей интереса
Сравнивание с шаблоном
Процент совпадения
Бинаризация обоих изображений
Наложение друг на друга (Bitwise and)
Подсчёт кол-ва несовпадающих пикселей
Конец

Скриншоты

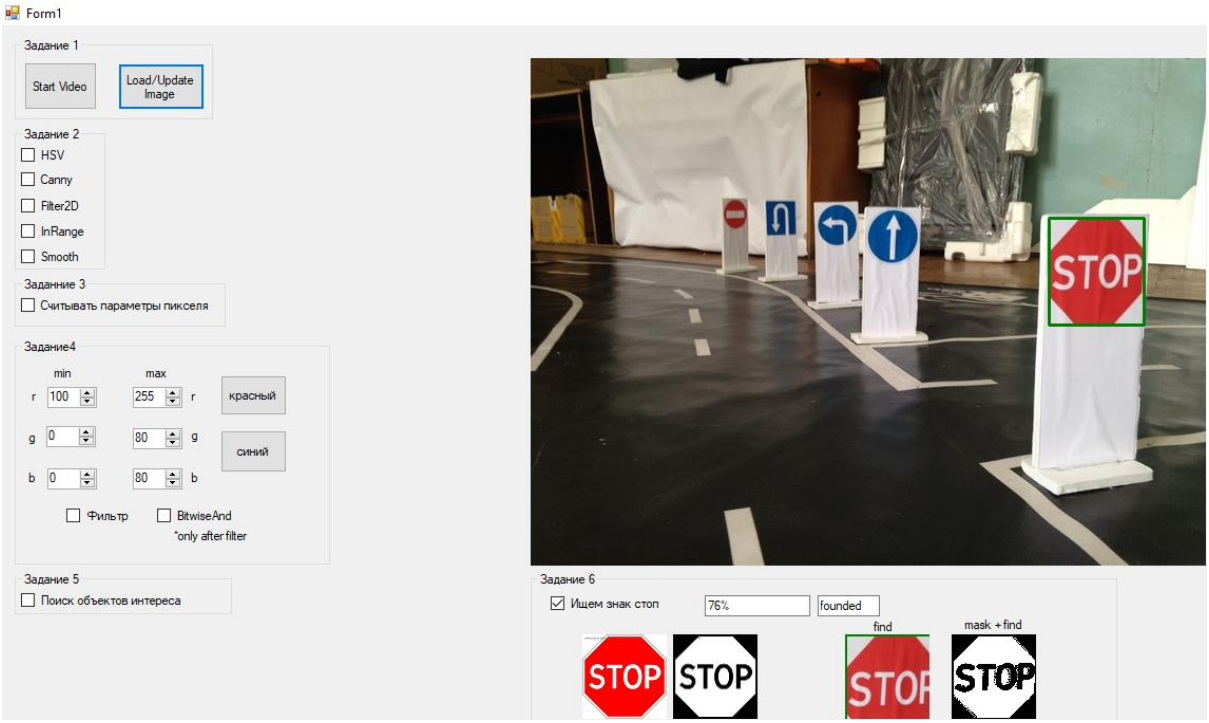



Бинаризация

Поиск объектов интереса



Сравнивание изображений



Код программы:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Imaging;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;

using OpenCvSharp;
using OpenCvSharp.Extensions;

namespace lab4
{
    public partial class Form1 : Form
    {
        Bitmap Sign_mask = Properties.Resources.STOP_binar1 as Bitmap;//засовываем в битмап
бинаризованный знак стоп
        Bitmap sign_find;     //найденный знак для сравнение
        Bitmap sign_find_bmp;
        Bitmap Sign_mask_bmp;
        Bitmap bitwise;
        Mat bitwise_mat = new Mat();
        double Sign_mask_pixels = 6320; //79x80 пикселей
        double Sign_find_pixels = 0;
        VideoCapture capture;
        Mat frame_in;
        Mat frame_out;
        Mat filter_frame = new Mat();
        Bitmap image;
        private Thread camera;
        bool isCameraRunning = false;



        // ЗАПУСК КАМЕРЫ
        public void CaptureCamera()
        {
            camera = new Thread(new ThreadStart(CaptureCameraCallback));
            camera.Start();
        }

        public void CaptureCameraCallback()
        {
            frame_in = new Mat();
            frame_out = new Mat();

            capture = new VideoCapture(0); // (0) -встроенная, (1) - первая подключенная
            capture.Open(0);

            if (capture.IsOpened())
            {
                while (isCameraRunning == true) //если камера запущена
                {
                    capture.Read(frame_in);
```

```
            if (frame_in.Cols != 0)
            {

                Processing();  //крутим в цикле обработку каждого кадра
            }
        }
    }

    //Сброс изображения  графического потока (при отсутствии данных с камеры)
    pictureBox1.Image = null;
}
public Form1()
{
    InitializeComponent();
}

private void button1_Click(object sender, EventArgs e)
{
    if (button1.Text.Equals("Start Video"))
    {
        CaptureCamera();
        button1.Text = "Stop";
        isCameraRunning = true;
    }
    else
    {
        capture.Release();
        button1.Text = "Start Video";
        isCameraRunning = false;
    }
}

private void button2_Click(object sender, EventArgs e)
{

    //image = new Bitmap(Convert.ToString(@"C:\Users\Rauf\source\repos\lab4\Sign.jpg")
    frame_in = new Mat(@"C:\Users\desgr\source\repos\lab4\Sign.jpg");
    frame_out = new Mat();
    image = BitmapConverter.ToBitmap(frame_in);
    pictureBox1.Image = image;
    Processing();

}

void Processing()
{
        //ОБРАБОТКА(HSV)
        if (checkBox1.Checked == true)
        {
            Cv2.CvtColor(frame_in, frame_out, ColorConversionCodes.RGB2HSV);
        }

        //ОБРАБОТКА(ФИЛЬТРАЦИЯ ПО KERNEL(ТУТ УЛУЧШ. ЧЁТКОСТИ))
        else if (checkBox3.Checked == true)
        {
            Mat kernel = new Mat(3, 3, MatType.CV_32F, new Scalar(0));
            kernel.Set<float>(1, 1, 5.0f);
            kernel.Set<float>(0, 1, -1.0f);
            kernel.Set<float>(2, 1, -1.0f);
            kernel.Set<float>(1, 0, -1.0f);
            kernel.Set<float>(1, 2, -1.0f);
            Cv2.Filter2D(frame_in, frame_out, MatType.CV_8U, kernel, anchor: new
```

```
OpenCvSharp.Point(0, 0));
                }
                //ОБРАБОТКА(БИНАРИЗАЦИЯ)
                else if (checkBox4.Checked == true)
                {
                    Scalar lower_limit = new Scalar(0, 0, 0);
                    Scalar upper_limit = new Scalar(50, 50, 50);
                    Cv2.InRange(frame_in, lower_limit, upper_limit, frame_out);
                }
                //ОБРАБОТКА(РАЗМЫТИЕ)
                else if (checkBox5.Checked == true)
                {
                    Cv2.Blur(frame_in, frame_out, new OpenCvSharp.Size(10, 10));
                }

                //ОБРАБОТКА(КОНТУРИРОВАНИЕ)
                else if (checkBox7.Checked == true)
                {
                     Cv2.CvtColor(frame_in, frame_in, ColorConversionCodes.RGB2GRAY);
                     Cv2.Canny(frame_in, frame_out, 32, 192);

                }
                //СЧИТЫВАНИЕ ПИКСЕЛЯ ИЗ ЦЕНТРА
                else if (checkBox8.Checked == true)
                {
                    get_pixel();
                }
                //БИНАРИЗАЦИЯ ПО ЦВЕТОВОМУ КЛЮЧУ
                else if ((checkBox9.Checked == true)&& (checkBox10.Checked == true))
                {
                    Mat frame_in1 = frame_in.Clone();
                    RGB_Filter();
                    Cv2.BitwiseOr(frame_in,frame_in1,frame_out,filter_frame );
                }
                else if(checkBox9.Checked==true)
                {
                    RGB_Filter_only();
                }

                //ИЩЕМ ОБЛАСТИ ИНТЕРЕСА(+ПРЯМОУГОЛЬНИКИ)
                else if (checkBox11.Checked == true)
                {

                    OpenCvSharp.Point[][] contours;
                    HierarchyIndex[] hierarchy;
                    Mat canny_out = new Mat();
                    Mat contour_out = new Mat();
                    Mat approx_out = new Mat();
                    Mat gray_out = new Mat();
                    Mat blur_out = new Mat();
                    //фильтруем изображение и ищем контуры
                    Cv2.CvtColor(frame_in, gray_out, ColorConversionCodes.BGRA2GRAY);
                    Cv2.GaussianBlur(gray_out, blur_out, new OpenCvSharp.Size(3, 3), 0);
                    Cv2.Canny(blur_out, canny_out, 10, 250);
                    Cv2.FindContours(canny_out, out contours, out hierarchy, mode:
RetrievalModes.External, method: ContourApproximationModes.ApproxSimple);
                    // double peri = Cv2.ArcLength(canny_out, true);
                    //  Cv2.ApproxPolyDP(canny_out, approx_out, 17, true);
                    //  frame_out = approx_out;


                    var pixels_list = new List<OpenCvSharp.Rect>();
                    foreach (var find_countours in contours)
```

```
                    {
                        if (find_countours.Length > 200) //если кол-во точек больше 200
                        {
                            pixels_list.Add(Cv2.BoundingRect(find_countours));
                        }
                    }

                    //РИСОВАНИЕ ОПИСЫВАЮЩИЕ ПРЯМОУГОЛЬНИКИ
                    frame_out = frame_in;
                    foreach (var selected_contour in pixels_list)
                    {
                        int Diam = (int)(Math.Sqrt(Math.Pow(selected_contour.Width, 2) +
Math.Pow(selected_contour.Height, 2)));
                        Cv2.Rectangle(frame_out, new OpenCvSharp.Point(selected_contour.X,
selected_contour.Y), new OpenCvSharp.Point(selected_contour.X + selected_contour.Width,
selected_contour.Y + selected_contour.Height), Scalar.Red, 2);
                        Cv2.PutText(frame_out, string.Format("x: {0}, y: {1}", selected_contour.X,
selected_contour.Y), new OpenCvSharp.Point(selected_contour.X + 10, selected_contour.Y + 25),
HersheyFonts.HersheySimplex, 0.5, Scalar.Red, 2);
                        Cv2.PutText(frame_out, string.Format("Diam:" + Diam), new
OpenCvSharp.Point(selected_contour.X + 10, selected_contour.Y + selected_contour.Height - 15),
HersheyFonts.HersheySimplex, 0.5, Scalar.Red, 2);

                    }
                }
                //ПОИСК ЗНАКА СТОП
                else if (checkBox6.Checked == true)
                {
                    bool founded = true;
                    Mat HSV_out = new Mat();
                    Mat binar_out = new Mat();
                    Mat blur_out = new Mat();
                    Mat canny_out = new Mat();
                    //фильтруем изображение и ищем контуры
                    Cv2.CvtColor(frame_in, HSV_out, ColorConversionCodes.RGB2HSV);
                    Cv2.InRange(HSV_out, new Scalar(114, 130, 160), new Scalar(160, 222, 250),
binar_out);
                    Cv2.GaussianBlur(binar_out, blur_out, new OpenCvSharp.Size(3, 3), 0);
                    Cv2.Canny(blur_out, canny_out, 20, 135);

                    OpenCvSharp.Point[][] contours;
                    HierarchyIndex[] hierarchy;
                    Cv2.FindContours(canny_out, out contours, out hierarchy, mode:
RetrievalModes.External, method: ContourApproximationModes.ApproxSimple);

                    var contours_list = new List<OpenCvSharp.Rect>();
                    foreach (var viewed_contour in contours)
                    {
                        //Пропуск слишком маленьких контуров
                        float height = Cv2.BoundingRect(viewed_contour).Height;
                        float width = Cv2.BoundingRect(viewed_contour).Width;
                        if ((height > 60) && (width >= 60) && (width / height <= 1.2) && (width /
height >= 0.8))
                        {
                            contours_list.Add(Cv2.BoundingRect(viewed_contour));
                        }
                    }
                    frame_out = frame_in;
                    Rect selected_contour = new Rect(); //Описывающий контур найденного знака
                    for (int i = 0; i < contours_list.Count; i++)
                    {
                        selected_contour = contours_list[i];
                        Cv2.Rectangle(frame_out, new OpenCvSharp.Point(selected_contour.X,
```

```
selected_contour.Y), new OpenCvSharp.Point(selected_contour.X + selected_contour.Width,
selected_contour.Y + selected_contour.Height), Scalar.Green, 2);
                    }
                    /*Если нашли контур то
                     * 1.берём 4 угла прямоугольника, сужаем изображение до 80x80 и засовываем в
picturebox
                     * 2. Обрабатываем изображение с бинаризацией
                     * 3. Сравниваем каждый пиксель найденного знака с пикселями масочного знака
                     * 4. Если совпало 30% значит нашли
                     */

                    Mat Sign_find_Mat = new Mat(frame_in, new Rect(selected_contour.X,
selected_contour.Y, selected_contour.Width, selected_contour.Height));
                    if (Sign_find_Mat.Rows > 0)
                    {
                        founded = true;
                        Mat resize_out = new Mat();
                        Mat hsv_out2 = new Mat();
                        Mat binar_put2 = new Mat();
                        Cv2.Resize(Sign_find_Mat, resize_out, new OpenCvSharp.Size(80, 80));
                       // Cv2.CvtColor(Sign_find_Mat, hsv_out2, ColorConversionCodes.RGB2HSV);
                        Cv2.InRange(resize_out, new Scalar(0, 0, 100), new Scalar(80, 80, 255),
binar_put2);
                    // Mat result = new Mat(masked_sign.Width- Sign_find_Mat.Width + 1,
masked_sign.Height - Sign_find_Mat.Height + 1,32,1);
                    //  Sign_mask = BitmapConverter.ToBitmap(Sign_mask_Mat);
                        Mat sign_mask_binar = BitmapConverter.ToMat(Sign_mask);
                        sign_find = BitmapConverter.ToBitmap(binar_put2);
                        sign_find_bmp = BitmapConverter.ToBitmap(Sign_find_Mat);
                        pictureBox2.Image = sign_find_bmp;


                    //Cv2.MatchTemplate( masked_sign, Sign_find_Mat, result,
TemplateMatchModes.CCoeffNormed);
                    // Bitmap result_bmp = BitmapConverter.ToBitmap(result);
                    //pictureBox2.Image = result_bmp;

                    //СРАВНИВАЕМ НАЙДЕННЫЙ ЗНАК С ШАБЛОНОКМ
                      sign_mask_binar
=Cv2.ImRead(@"C:\Users\desgr\source\repos\lab4\STOP_binar.jpg",0);
                      Cv2.BitwiseAnd(binar_put2, binar_put2, bitwise_mat,sign_mask_binar);
                      bitwise = BitmapConverter.ToBitmap(bitwise_mat);
                      pictureBox4.Image = bitwise;
                      int Template = Cv2.CountNonZero(sign_mask_binar);
                      int sign_find_pixels = Cv2.CountNonZero(bitwise_mat);
                     //  compare();
                      int similar_pixels = sign_find_pixels * 100 / Template;
                      this.Invoke((MethodInvoker)delegate ()
                      {
                           textBox1.Text = string.Format(similar_pixels + "%");
                      });
                      //ЕСЛИ СОВПАДЕНИЕ >70% ЗНАЧИТ ЭТО НАШ ЗНАК
                      if (similar_pixels >= 70)
                      {
                              this.Invoke((MethodInvoker)delegate ()
                              {
                                 textBox2.Text = string.Format("founded");
                              });
                      }
                      else
                      {
                            this.Invoke((MethodInvoker)delegate ()
                            {
```

```
                            textBox2.Text =null;
                        });
                    }
                }


            }

            else{ frame_out = frame_in; }
            image = BitmapConverter.ToBitmap(frame_out);
            pictureBox1.Image = image;
        }

        void get_pixel()
        {
            try
            {
                Vec3b color = frame_out.Get<Vec3b>(320, 240);
                int b = color.Item0;
                int g = color.Item1;
                int r = color.Item2;


                Cv2.Circle(frame_out, new OpenCvSharp.Point(320, 240), 3, new Scalar(0, 0, 0));
                Cv2.PutText(frame_out, string.Format("r: {0}, g: {1}, b: {2}", r, g, b), new
OpenCvSharp.Point(240, 255), HersheyFonts.HersheyPlain, 1, new Scalar(0, 0, 0), 1);
                image = BitmapConverter.ToBitmap(frame_out);
            }
            catch { MessageBox.Show("No image", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Information); }
        }

        void RGB_Filter()
        {
            int r_min = Convert.ToInt32(Rmin.Value);
            int g_min = Convert.ToInt32(Gmin.Value);
            int b_min = Convert.ToInt32(Bmin.Value);
            int r_max = Convert.ToInt32(Rmax.Value);
            int g_max = Convert.ToInt32(Gmax.Value);
            int b_max = Convert.ToInt32(Bmax.Value);

            Scalar lower_limit = new Scalar(b_min, g_min, r_min);
            Scalar upper_limit = new Scalar(b_max, g_max, r_max);
            Cv2.InRange(frame_in, lower_limit, upper_limit, filter_frame);
        }
        void RGB_Filter_only()
        {
            int r_min = Convert.ToInt32(Rmin.Value);
            int g_min = Convert.ToInt32(Gmin.Value);
            int b_min = Convert.ToInt32(Bmin.Value);
            int r_max = Convert.ToInt32(Rmax.Value);
            int g_max = Convert.ToInt32(Gmax.Value);
            int b_max = Convert.ToInt32(Bmax.Value);

            Scalar lower_limit = new Scalar(b_min, g_min, r_min);
            Scalar upper_limit = new Scalar(b_max, g_max, r_max);
            Cv2.InRange(frame_in, lower_limit, upper_limit, frame_out);
        }
        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 = 100; Rmax.Value = 255;
Gmin.Value = 0; Gmax.Value = 80; Bmin.Value = 0; Bmax.Value = 80; ; }
```

```
        void compare()
        {



            Sign_find_pixels = 0;
            for (int x = 0; x < 80; x++)
            {
                for (int y = 0; y < 79; y++)
                {
                    int mask_color_r = Sign_mask.GetPixel(x, y).R;
                    int mask_color_g = Sign_mask.GetPixel(x, y).G;
                    int mask_color_b = Sign_mask.GetPixel(x, y).B;

                    int find_color_r = sign_find.GetPixel(x, y).R;
                    int find_color_g = sign_find.GetPixel(x, y).G;
                    int find_color_b = sign_find.GetPixel(x, y).B;


                    //ЕСЛИ ПИКСЕЛЬ ШАБЛОНА = ПИКСЕЛЬ НАЙДЕННОГО ЗНАКА
                    if ((mask_color_r == 255) && (mask_color_g == 255) && (mask_color_b == 255))
                    {
                        if ((find_color_r == 255) && (find_color_g == 255) && (find_color_b == 255))
                        {

                            Sign_find_pixels++;
                        }
                    }
                    else if ((mask_color_r == 0) && (mask_color_g == 0) && (mask_color_b == 0))
                    {
                        if ((find_color_r == 0) && (find_color_g == 0) && (find_color_b == 0))
                        {
                            Sign_find_pixels++;
                        }

                    }

                }
            }

        }
    }
}
```