МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

(ФГБОУ ВПО МПУ)



Кафедра СМАРТ технологий

Лабораторно-практическое задание № 2

“ Разработка и применение систем технического зрения “

По дисциплине «Системы технического зрения в автоматизированных системах управления»

Группа 201-325

Студент Холодилов И.В.

Дата 11.05.2022

Преподаватель Идиатулов Т.Т.

2022

**Тема:**

Базовые принципы обработки изображений и поиска зон интереса.

**Цель работы:**

Разработать алгоритм обработки и коррекции изображения для поиска зон интереса с использованием цветового и структурного ключей.

**Ход работы:**

1. Подготовить приложение для показа изображений из рабочей папки с выбором изображения из списка:

Подготовить приложения для отображения списка графических файлов из рабочей папки (может быть задана строкой на форме приложения). Щелчок на строке в списке файлов должен приводить к загрузке изображения для просмотра в компонент picturebox. Изображение также должно загружаться в буферный объект bitmap для дальнейшего использования.

1. Разработать функцию задания области интереса указателем мыши с использованием буфера для хранения исходного изображения:

Подготовить обработчики событий mousedown и mouseup указателя мыши для отрисовки прямоугольника выделения области интереса. Изображения для отрисовки прямоугольника должно браться из буферного объекта. Разработать функцию вычисления среднего значения каждой компоненты цвета RGB по всем пикселям области интереса и вычислить коэффициент коррекции для метода «серого мира» (как множитель к яркости компонента цвета).

1. Разработать функцию вычисления параметров цветокоррекции по заданной области интереса и выполнения цветокоррекции:

Разместить на форме переключатель для выбора режима выделения указателем мыши между «выделение области интереса» и «считывание параметров пикселя». Реализовать функцию вывода значения RGB пикселя в точке клика указателем мыши. Выводить результат поверх изображения со смещением от точки клика

1. Разработать функцию считывания параметров пикселя и вывода значений на изображении:

Разместить на форме поля для ввода значений цветового ключа (Rmin, Rmax, Gmin, Gmax, Bmin, Bmax). Создать функцию фильтрации изображения, заменяющего цвет пикселей, не попадающих в диапазон ключевых значений на чёрный цвет (0, 0, 0). Реализовать возможность отдельного показа изображениямаски как чёрно-белого изображения. Разработать функцию поиска центра масс кластера с использованием гистрограмм частоты размещения точек в сетке 10 х 10 ячеек (относительно исходного изображения). Реализовать функцию поиска размера объекта исходя из критерия плотности заполнения 0,3. Выполнить обводку прямоугольником найденного объекта. Определить все возможные объекты на изображении.

1. Разработать функцию фильтрации изображения с использованием цветового ключа по всем компонентам RGB:

Разработать функцию детектирования цвета (сигнала) светофора для серии фотографий объекта, в предположении об известной удаленности объекта от точки съемки (то есть примерно известен визуальный размер объекта). При выборе фотографии из списка должен обводиться найденный сигнал и печататься выдаваться сообщение о текущем режиме светофора.

1. Разработать функцию расчёта координат центра кластера выделенных фильтром пикселей в области модальных значений и отсечения выпадающих по удалённости точек, учитывать возможное наличие нескольких объектов (кластеров):
2. Провести анализ объектов по структурному ключу (плотное круглое пятно с заданным диапазоном размеров):
3. Разработать программу детектирования работы светофора:

**Программный код:**

|  |
| --- |
| 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;  namespace Lab\_2  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  pictureBox1.Image = new Bitmap(400, 400);  //pictureBox2.Image = new Bitmap(400, 400);  pictureBox2.Image = new Bitmap(640, 480);  pictureBox3.Image = new Bitmap(400, 400);  }  Image work\_image = new Bitmap(400, 400);  Image output\_image = new Bitmap(400, 400);  bool Open\_flag = false;  public void filtred\_image()  {  if (filter.Checked)  {  pictureBox3.Image = (Bitmap)work\_image.Clone();  double R = 0; double G = 0; double B = 0;  for (int i = 0; i < work\_image.Width; i++)  {  for (int j = 0; j < work\_image.Height; j++)  {  Color pixel = ((Bitmap)work\_image).GetPixel(i, j);  R += pixel.R;  G += pixel.G;  B += pixel.B;  }  }  R /= work\_image.Width \* work\_image.Height;  B /= work\_image.Width \* work\_image.Height;  G /= work\_image.Width \* work\_image.Height;  double k = (R + B + G) / 3;  R = k / R;  G = k / G;  B = k / B;  for (int i = 0; i < work\_image.Width; i++)  {  for (int j = 0; j < work\_image.Height; j++)  {  Color pixel = ((Bitmap)work\_image).GetPixel(i, j);  ((Bitmap)work\_image).SetPixel(i, j, Color.FromArgb(data\_rgb(pixel.R \* R), data\_rgb(pixel.G \* G), data\_rgb(pixel.B \* B)));  }  }  }  else  {  Graphics graphics = Graphics.FromImage(pictureBox3.Image);  graphics.FillRectangle(Brushes.White, new Rectangle(0, 0, pictureBox3.Width, pictureBox3.Height));  }  output\_image = (Bitmap)work\_image.Clone();  pictureBox1.Image = output\_image;  pictureBox1.Refresh();  pictureBox3.Refresh();  }  private void Open\_but\_Click(object sender, EventArgs e)  {  try  {  DialogResult res = openFileDialog1.ShowDialog();  if (res == DialogResult.OK)  {  work\_image = Image.FromFile(openFileDialog1.FileName);  filtred\_image();  }  else MessageBox.Show("Error, you don't take any file.");  }  catch (Exception ex)  {  MessageBox.Show("Error, your file have incorrect type. You must take .png, .jpg or .bmp.");  }  Start\_trafic\_but.BackColor = Color.White;  Output\_string.Text = "";  Graphics graphics2 = Graphics.FromImage(pictureBox2.Image);  graphics2.FillRectangle(Brushes.Black, new Rectangle(0, 0, pictureBox2.Width, pictureBox2.Height));  pictureBox1.Refresh();  pictureBox2.Refresh();  Open\_flag = true;  }  int[] rectangle = new int[4];  int[] pix = new int[2];  Rectangle rect;  int R = 0; int G = 0; int B = 0;  public int data\_rgb (double data)  {  if (data > 255) return 255;  else if (data < 0 ) return 0;  return (int)data;  }  private void pictureBox1\_MouseDown(object sender, MouseEventArgs e)  {  Graphics graphics = Graphics.FromImage(pictureBox1.Image);  if (comboBox1.Text == "One pixel")  {  pix[0] = e.Location.X;  pix[1] = e.Location.Y;  Font drawFont = new Font("Arial", 20);  StringFormat drawFormat = new StringFormat();  SolidBrush drawBrush = new SolidBrush(Color.Blue);  Color pixel = ((Bitmap)work\_image).GetPixel(pix[0], pix[1]);  drawFormat.FormatFlags = StringFormatFlags.DirectionRightToLeft;  graphics.DrawString(pixel.B.ToString()+", "+ pixel.G.ToString() + ", "+ pixel.R.ToString(), drawFont, drawBrush, e.Location.X, e.Location.Y, drawFormat);  pictureBox1.Refresh();  }  else  {  rectangle[0] = e.Location.X;  rectangle[1] = e.Location.Y;  }  }  private void pictureBox1\_MouseUp(object sender, MouseEventArgs e)  {  Graphics graphics = Graphics.FromImage(pictureBox1.Image);  if (comboBox1.Text != "One pixel")  {  rectangle[2] = e.Location.X;  rectangle[3] = e.Location.Y;  Pen pen = new Pen(Color.Blue);  graphics.DrawRectangle(pen, GetRect());  Font drawFont = new Font("Arial", 20);  StringFormat drawFormat = new StringFormat();  SolidBrush drawBrush = new SolidBrush(Color.Blue);  drawFormat.FormatFlags = StringFormatFlags.DirectionRightToLeft;  graphics.DrawString(B.ToString() + ", " + G.ToString() + ", " + R.ToString(), drawFont, drawBrush, e.Location.X, e.Location.Y, drawFormat);    pictureBox1.Refresh();  }  }  public Rectangle GetRect(){  rect = new Rectangle();  rect.X = Math.Min(rectangle[0], rectangle[2]);  rect.Y = Math.Min(rectangle[1], rectangle[3]);  rect.Width = Math.Abs(rectangle[0] - rectangle[2]);  rect.Height = Math.Abs(rectangle[1] - rectangle[3]);  sr\_RGB(rect.X, rect.Y, rect.Height, rect.Width);  return rect;  }  public void sr\_RGB (int x, int y, int w, int h)  {  Color pixel = ((Bitmap)work\_image).GetPixel(x, y);  R = pixel.R; G = pixel.G; B = pixel.B;  int R\_minn = 255; int G\_minn = 255; int B\_minn = 255;  int R\_maxx = 0; int G\_maxx = 0; int B\_maxx = 0;  for (int i = x; i < x+h; i++)  {  for (int j = y; j < y+w; j++)  {  pixel = ((Bitmap)work\_image).GetPixel(i, j);  R = (R + pixel.R)/2;  G = (G + pixel.G)/2;  B = (B + pixel.B)/2;  if (R\_minn > R) R\_minn = R;  else if (G\_minn > G) G\_minn = G;  else if (B\_minn > B) B\_minn = B;  else if (R\_maxx < R) R\_maxx = R;  else if (G\_maxx < G) G\_maxx = G;  else if (B\_maxx < B) B\_maxx = B;  }  }  B\_min.Text = R\_minn.ToString();  G\_min.Text = G\_minn.ToString();  R\_min.Text = B\_minn.ToString();  B\_max.Text = R\_maxx.ToString();  G\_max.Text = G\_maxx.ToString();  R\_max.Text = B\_maxx.ToString();  pictureBox1.Refresh();  }  public void clear()  {  //Graphics graphics = Graphics.FromImage(pictureBox1.Image);  //graphics.FillRectangle(Brushes.White, new Rectangle(0, 0, pictureBox1.Width, pictureBox1.Height));  Graphics graphics2 = Graphics.FromImage(pictureBox2.Image);  graphics2.FillRectangle(Brushes.Black, new Rectangle(0, 0, pictureBox2.Width, pictureBox2.Height));  output\_image = (Bitmap)work\_image.Clone();  pictureBox1.Image = output\_image;  Output\_string.Text = "";  pictureBox1.Refresh();  pictureBox2.Refresh();  }  private void Clear\_but\_Click(object sender, EventArgs e)  {  if (Open\_flag)  {  clear();  }  else MessageBox.Show("First of all open some image");  }  private void Start\_but\_Click(object sender, EventArgs e)  {  if (Open\_flag)  {  Graphics graphics = Graphics.FromImage(pictureBox2.Image);  if (Start\_but.Text != "Refresh")  {  detect\_claster(pictureBox2, Int32.Parse(B\_min.Text), Int32.Parse(B\_max.Text), Int32.Parse(G\_min.Text), Int32.Parse(G\_max.Text), Int32.Parse(R\_min.Text), Int32.Parse(R\_max.Text));  pictureBox2.Refresh();  Start\_but.Text = "Refresh";  }  else  {  Start\_but.Text = "Start find color";  graphics.FillRectangle(Brushes.Black, new Rectangle(0, 0, pictureBox2.Width, pictureBox2.Height));  pictureBox2.Refresh();  }  }  else MessageBox.Show("First of all open some image");  }  List<Rectangle> list\_claster = new List<Rectangle>();  public int detect\_claster(PictureBox pictureBox, int R\_minn, int R\_maxx, int G\_minn, int G\_maxx, int B\_minn, int B\_maxx, bool is\_detect\_claster = false)  {  Graphics graphics = Graphics.FromImage(pictureBox1.Image);  list\_claster.Clear();  int[,] frame = new int[work\_image.Width, work\_image.Height];  Rectangle rectangle = new Rectangle();  Pen pen = new Pen(Color.Black);  for (int i = 0; i < work\_image.Width; i++)  {  for (int j = 0; j < work\_image.Height; j++)  {  Color pixel = ((Bitmap)work\_image).GetPixel(i, j);  if (pixel.R >= R\_minn && pixel.R <= R\_maxx && pixel.B >= B\_minn && pixel.B <= B\_maxx && pixel.G >= G\_minn && pixel.G <= G\_maxx)  {  frame[i, j] = 1;  ((Bitmap)pictureBox2.Image).SetPixel(i, j, Color.White);  }  }  }  pictureBox2.Refresh();  progressBar2.Maximum = work\_image.Width;  for (int i = 10; i < work\_image.Width; i += 10)  {  for (int j = 10; j < work\_image.Height; j += 10)  {  if (frame[i, j] == 1)  {  int k = 1;  bool last = false;  double f;  while (true )  {  if (last) f = counttt(frame, i, j, k, true);  else f = counttt(frame, i, j, k);  if (f < 0.2 && !last) break;  else if (f > 0.4)  {  k++;  last = true;  }  else if (f <= 0.4 && f >= 0.2)  {  rectangle.X = i - 5 \* (k - 1);  rectangle.Y = j - 5 \* (k - 1);  rectangle.Height = 10 \* (k - 1);  rectangle.Width = 10 \* (k - 1);  pictureBox1.Refresh();  list\_claster.Add(rectangle);  break;  }  }  }  }  progressBar2.Value = i;  }  progressBar2.Value = 0;  int r = 0;  pen = new Pen(Color.Blue);  while (list\_claster.Count > 1)  {  int g = list\_claster.Count;  while (r < list\_claster.Count)  {  while (r < list\_claster.Count && (list\_claster[r].Width == 0 || list\_claster[r].Width == 0)) list\_claster.RemoveAt(r);  int j = r+1;  while (j < list\_claster.Count)  {  while (j < list\_claster.Count && (list\_claster[j].Width == 0 || list\_claster[j].Width == 0)) list\_claster.RemoveAt(j);  if (j < list\_claster.Count && check\_claster(list\_claster[r], list\_claster[j])) add\_claster(list\_claster[r], list\_claster[j]);  else j++;  }  r++;  }  r = 0;  if (g - list\_claster.Count <= 0) break;  g = list\_claster.Count;  }  if (list\_claster.Count == 1 && list\_claster[0].Height + list\_claster[0].Width < 2 \* Int32.Parse(min\_claster\_size.Text)) list\_claster.Clear();  for (int k = 0; k < list\_claster.Count; k++)  {  graphics.DrawRectangle(pen, list\_claster[k]);  }  pictureBox2.Refresh();  pictureBox1.Refresh();  return list\_claster.Count;  }  public void add\_claster(Rectangle last, Rectangle neww)  {  int x\_min = Math.Min(last.X, neww.X);  int x\_max = Math.Max(last.X + last.Width, neww.X + neww.Width);  int y\_min = Math.Min(last.Y, neww.Y);  int y\_max = Math.Max(last.Y + last.Height, neww.Y + neww.Height);  if ((x\_max - x\_min >= Int32.Parse(min\_claster\_size.Text) && y\_max - y\_min >= Int32.Parse(min\_claster\_size.Text)) && (x\_max - x\_min <= Int32.Parse(max\_claster\_size.Text) && y\_max - y\_min >= Int32.Parse(max\_claster\_size.Text)))  {  list\_claster.Remove(last);  list\_claster.Remove(neww);  list\_claster.Add(new Rectangle(x\_min, y\_min, x\_max - x\_min, y\_max - y\_min));  }  else  {  if (last.Height \* last.Width > neww.Height \* neww.Width)  {  list\_claster.Remove(neww);  if (last.Height + last.Width <= 2 \* Int32.Parse(min\_claster\_size.Text)) list\_claster.Remove(last);  }  else  {  list\_claster.Remove(last);  if (neww.Height + neww.Width <= 2 \* Int32.Parse(min\_claster\_size.Text)) list\_claster.Remove(neww);  }  }  }  public bool check\_claster(Rectangle first, Rectangle last)  {  int first\_x = first.X+ first.Width / 2;  int first\_y = first.Y + first.Height / 2;  int last\_x = last.X + last.Width / 2;  int last\_y = last.Y + last.Height / 2;  if ((last.X <= first\_x && first\_x <= last.X + last.Width) && (last.Y <= first\_y && first\_y <= last.Y + last.Height)) return true;  else if ((first.X <= last\_x && last\_x <= first.X + first.Width) && (first.Y <= last\_y && last\_y <= first.Y + first.Height)) return true;  return false;  }  public double counttt(int[,] frame, int i, int j, int h, bool tr = false)  {  int[] sum = new int[2] { 0, 0 };  for (int k\_x = i - 5\*h; k\_x < i + 5\*h && i + 5 \* h < 400 && k\_x >= 0; k\_x++)  {  for (int k\_y = j - 5\*h; k\_y < j + 5\*h && j + 5 \* h < 400 && k\_y >= 0; k\_y++)  {  sum[0] += frame[k\_x, k\_y];  sum[1]++;  }  }  return (double)sum[0] / sum[1];  }  private void Start\_trafic\_but\_Click(object sender, EventArgs e)  {  if (Open\_flag) {  clear();  int rect = detect\_claster(pictureBox1, 240, 255, 240, 255, 10, 80);  if (rect != 0) Output\_string.Text += " Yellow: "+ rect.ToString();  rect = detect\_claster(pictureBox1, 122, 202, 230, 255, 180, 255);  if (rect != 0) Output\_string.Text += " Green: " + rect.ToString();  rect = detect\_claster(pictureBox1, 185, 229, 13, 140, 26, 115);  if (rect != 0) Output\_string.Text += " Red: " + rect.ToString();  if (Output\_string.Text == "") Output\_string.Text = "No traffic lights detected";  }  else MessageBox.Show("First of all open some image");  }  private void filter\_CheckedChanged(object sender, EventArgs e)  {  if (Open\_flag) filtred\_image();  else MessageBox.Show("First of all open some image");  }  private void min\_claster\_size\_TextChanged(object sender, EventArgs e)  {  if (Int32.Parse(max\_claster\_size.Text) < Int32.Parse(min\_claster\_size.Text))  {  MessageBox.Show("Min");  }  }  private void max\_claster\_size\_TextChanged(object sender, EventArgs e)  {  if (Int32.Parse(max\_claster\_size.Text) < Int32.Parse(min\_claster\_size.Text))  {  MessageBox.Show("Min");  }  }  }  } |

**Приложения:**

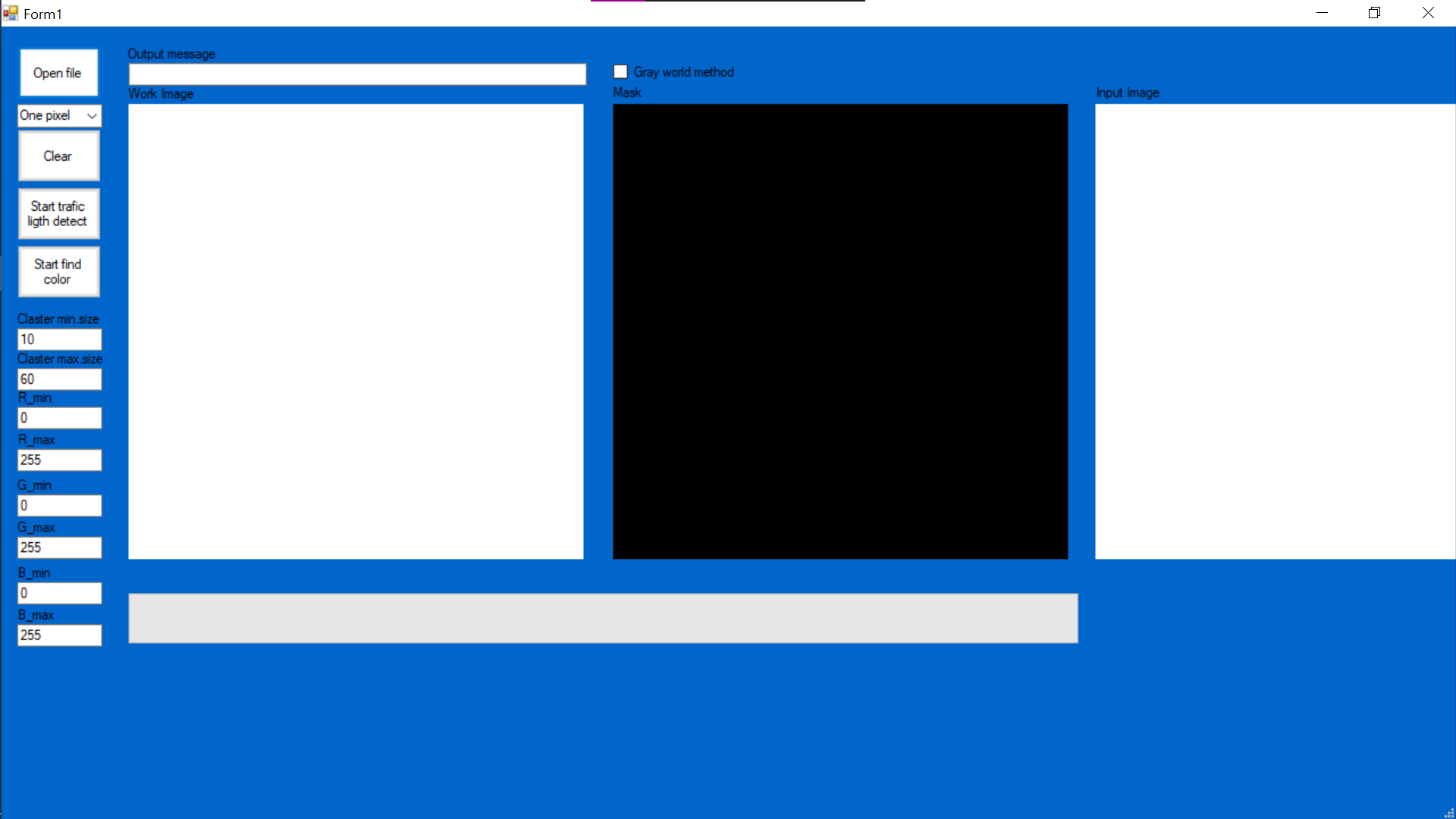


Рисунок - Изначальный вид формы

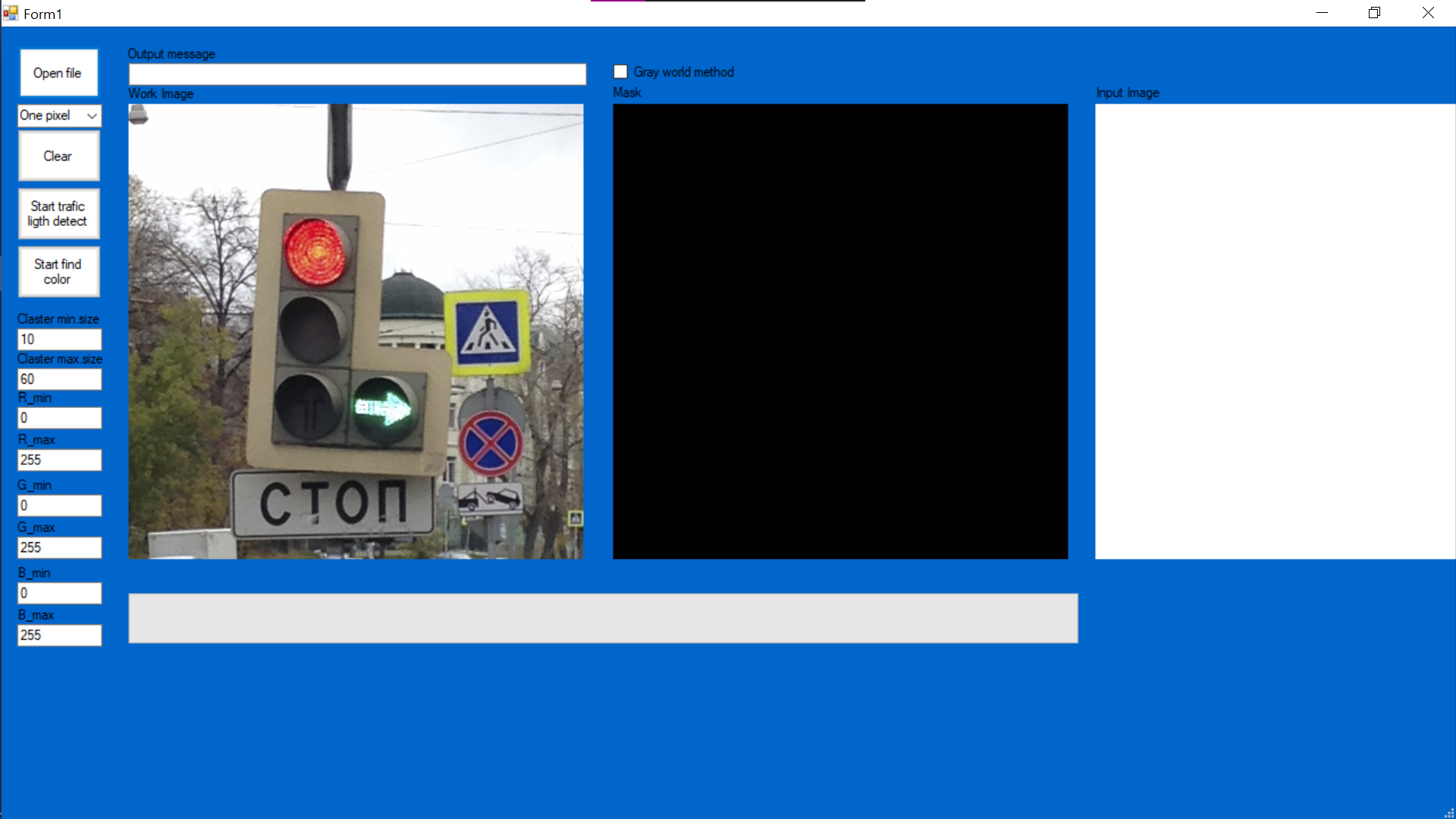


Рисунок - Генерация и отображение данных для 10000 точек

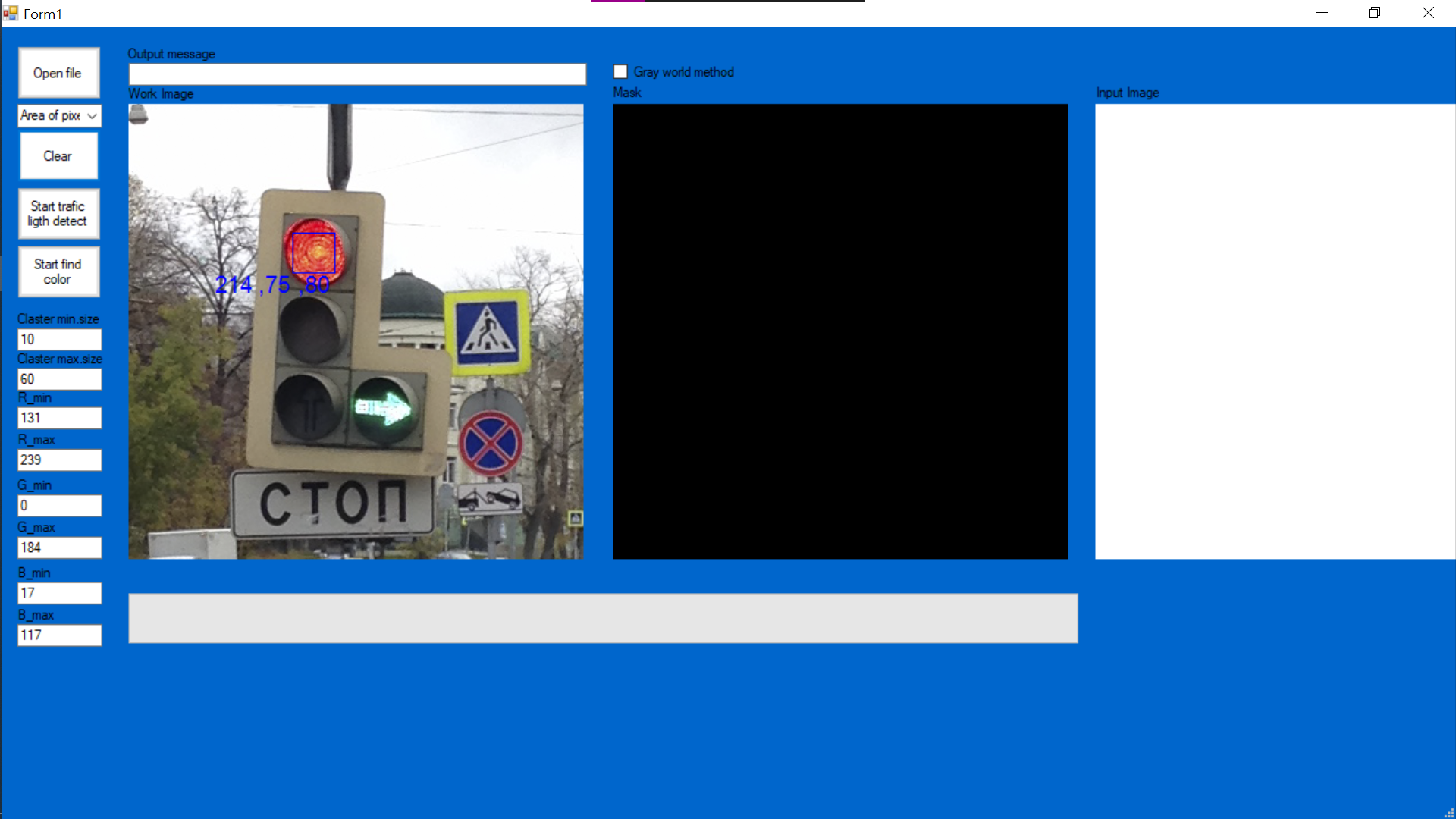
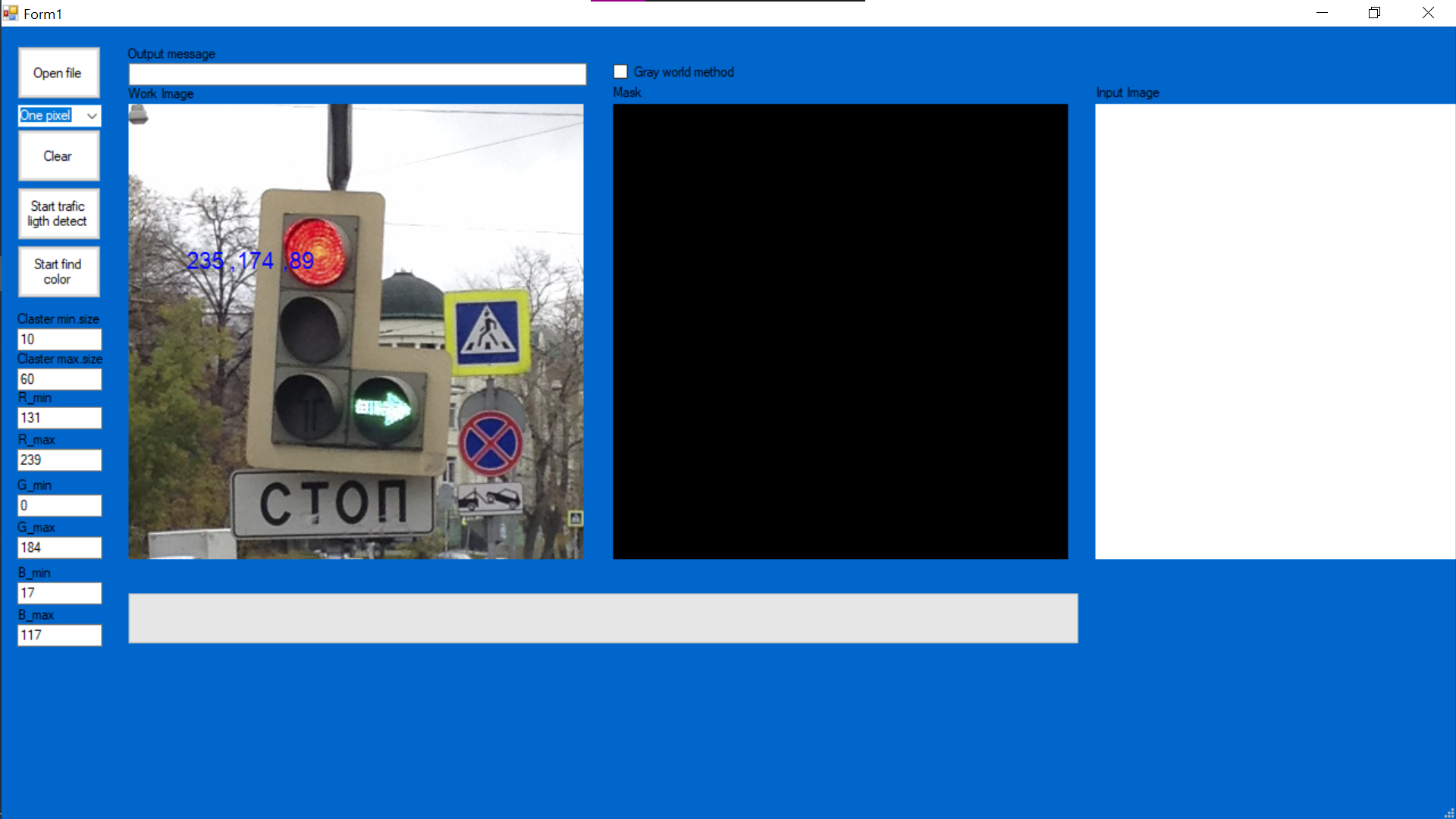
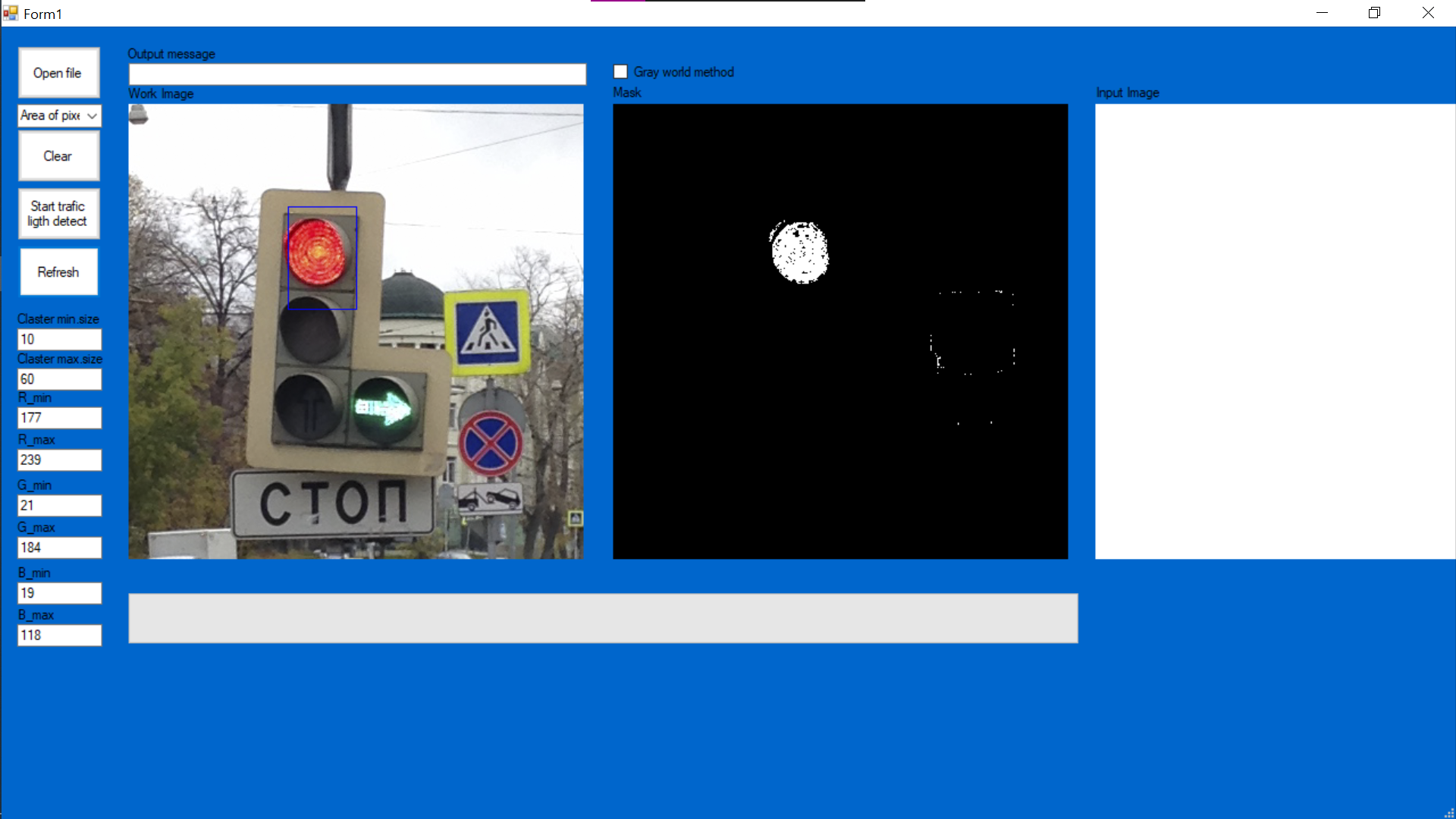
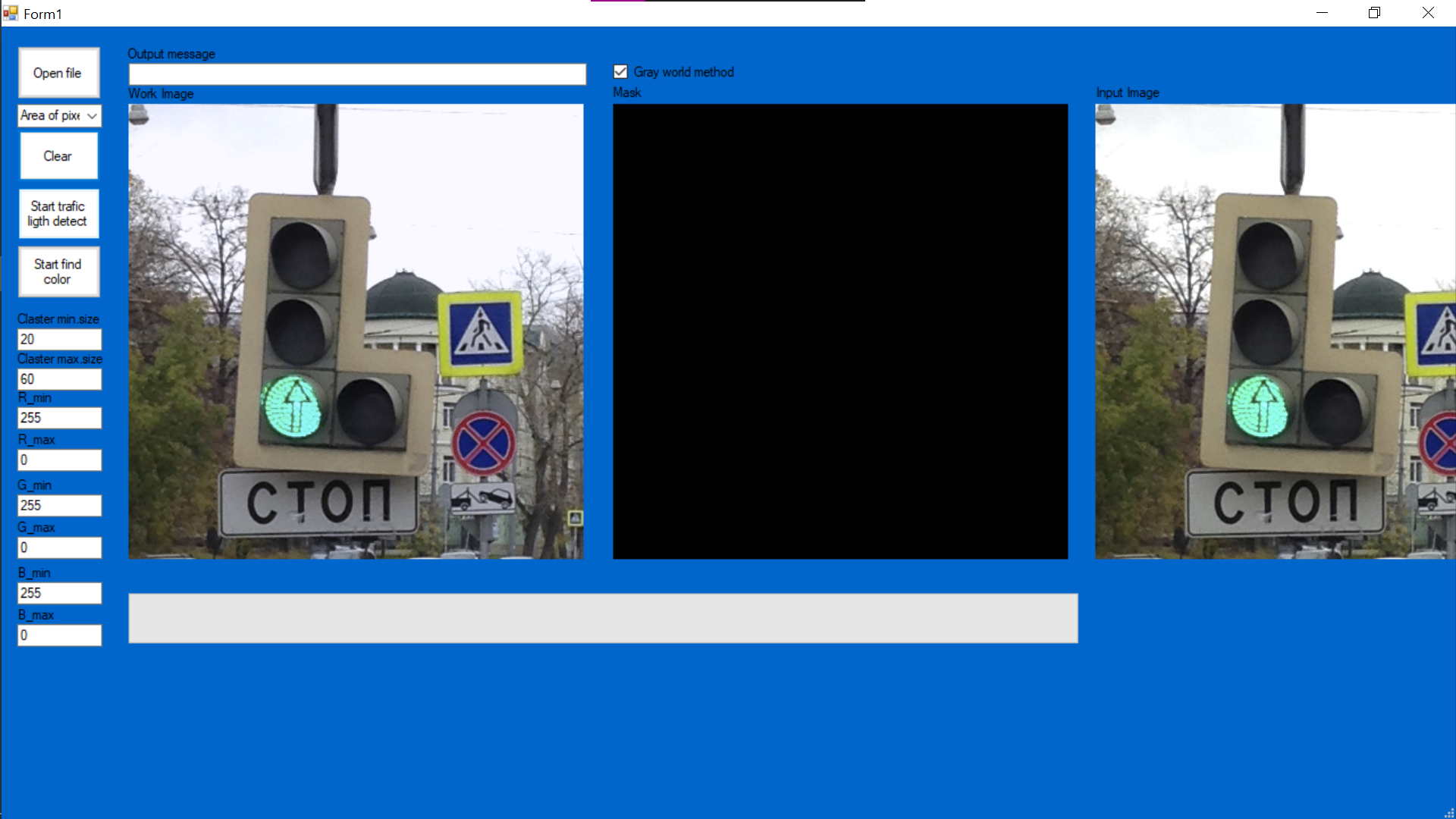
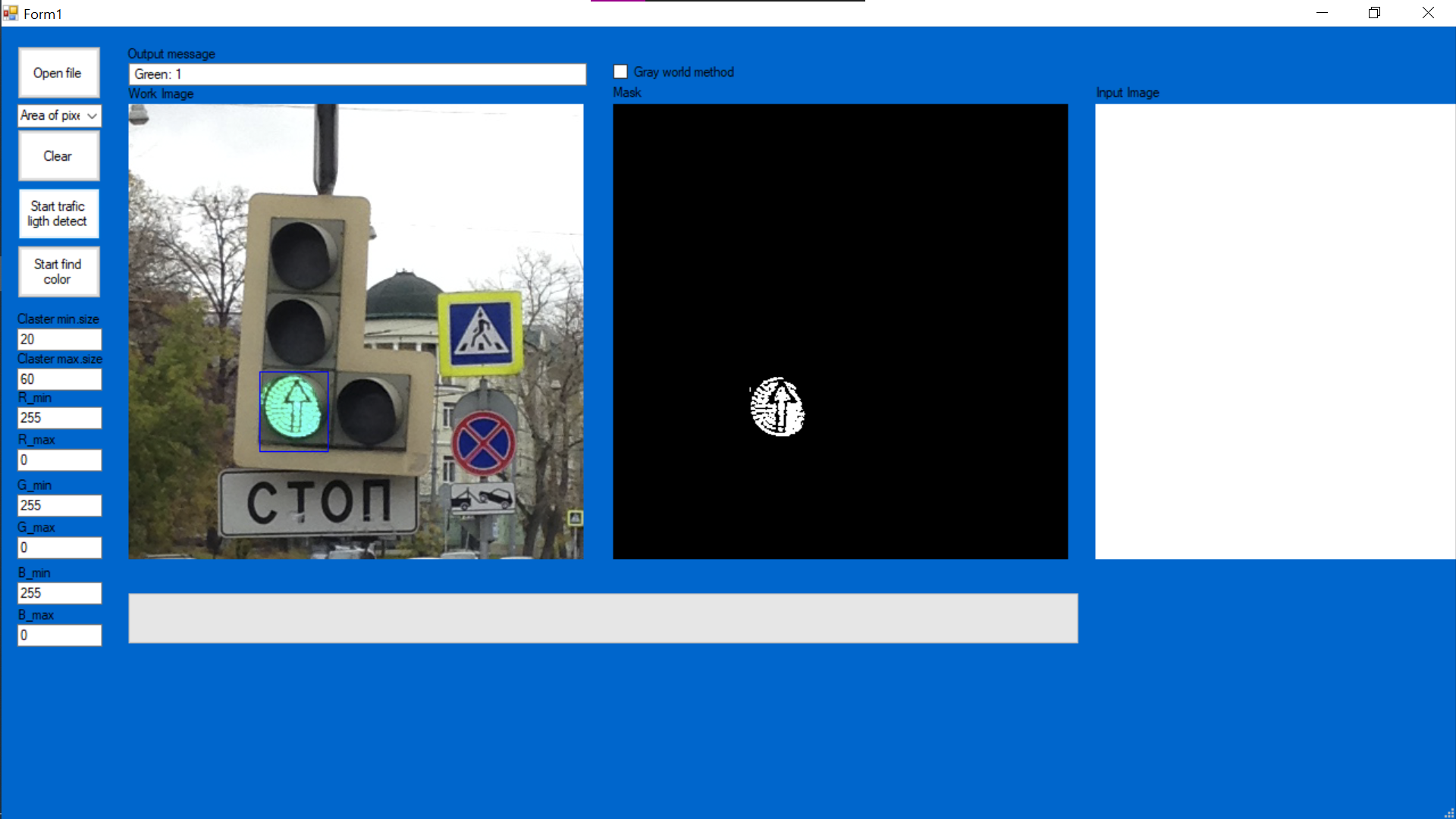
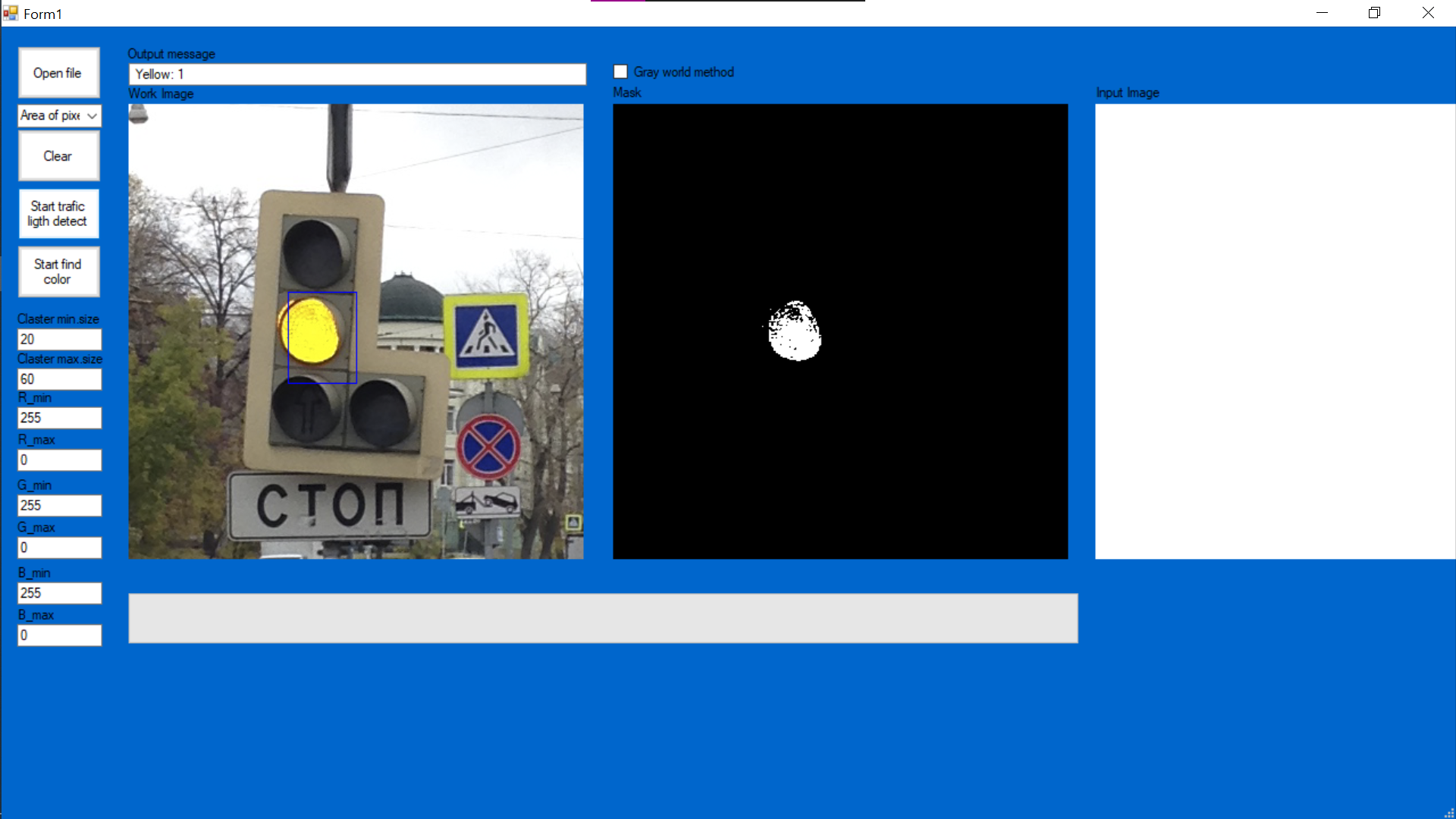
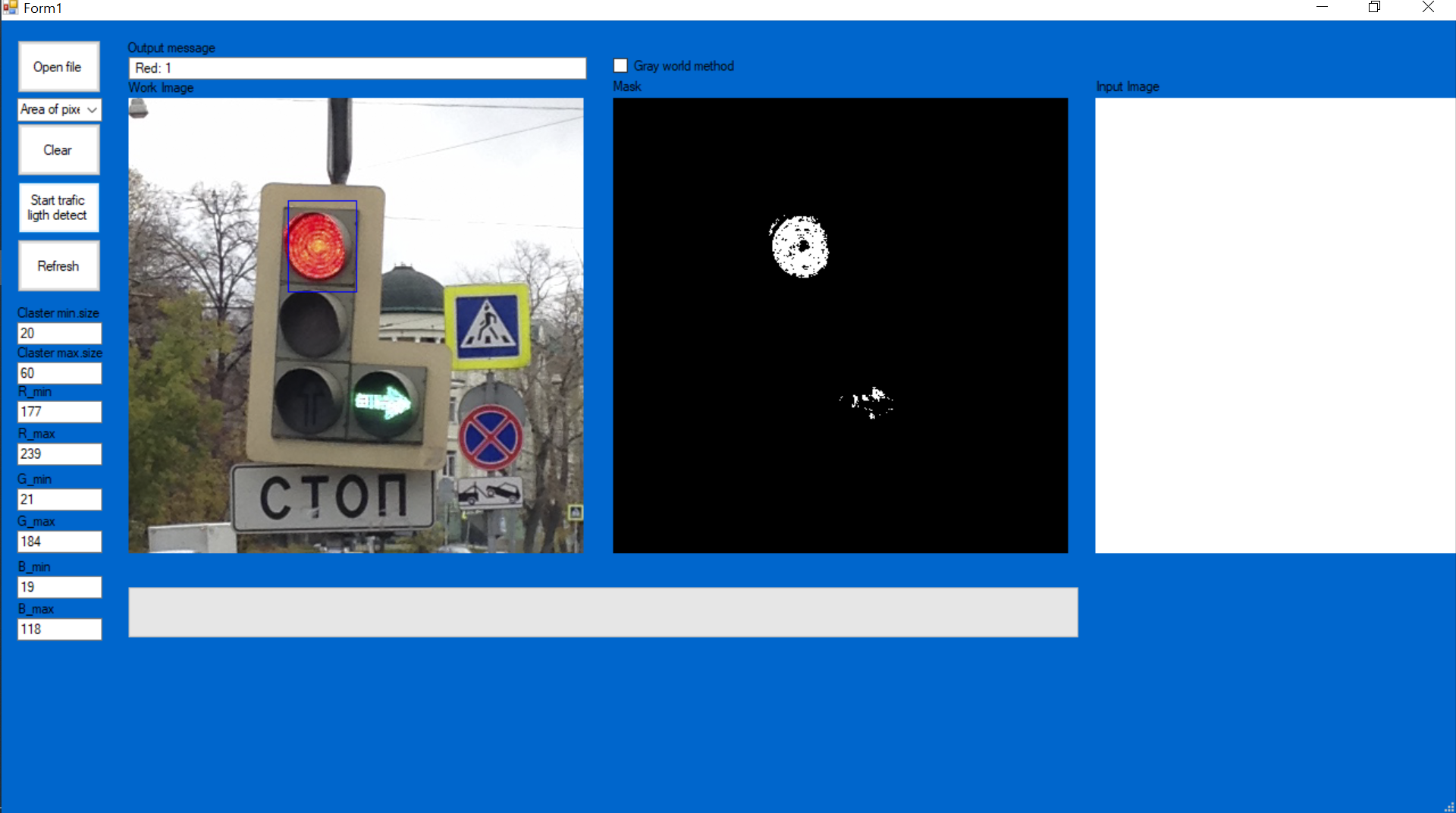
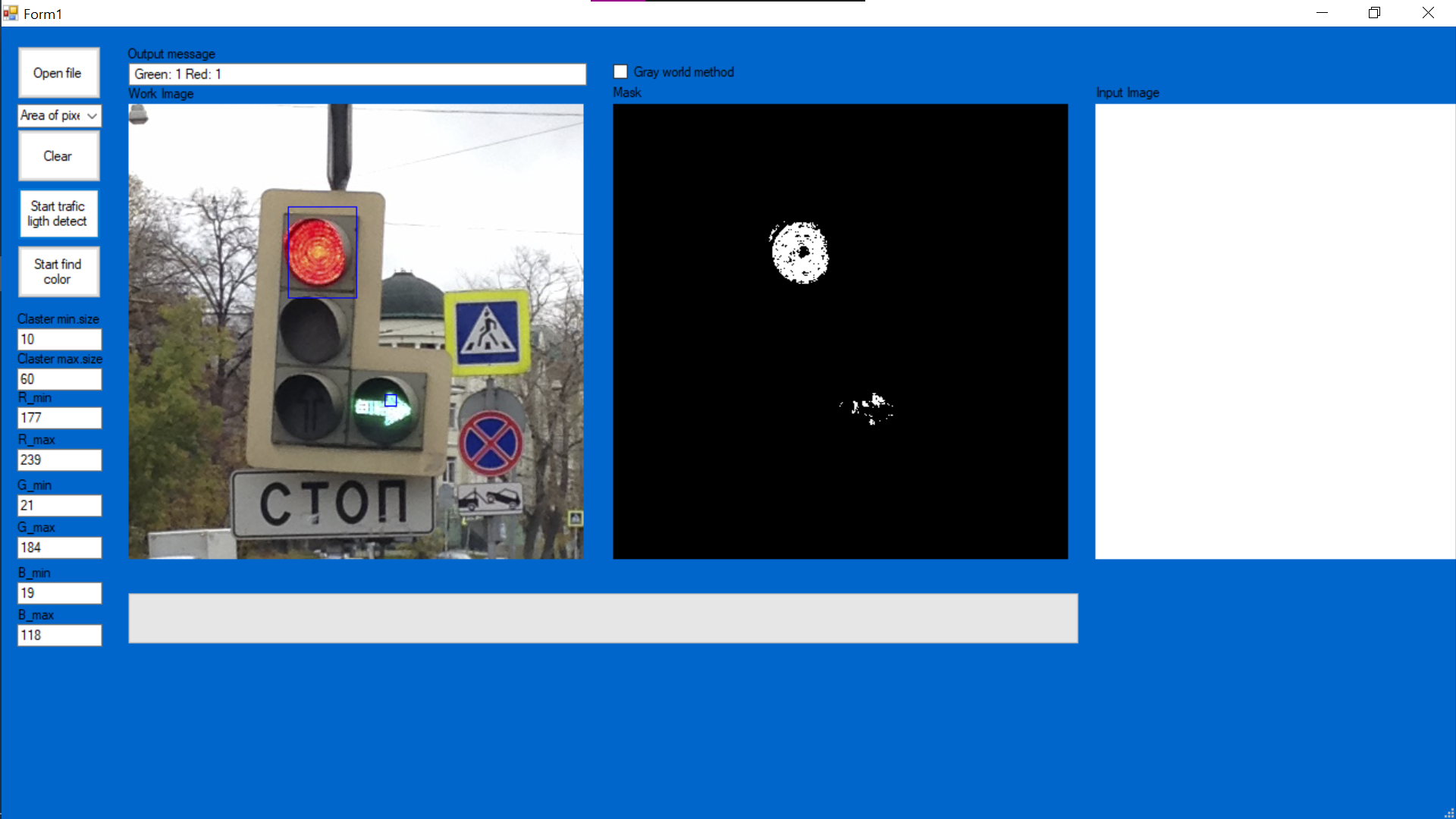


Рисунок - Кластеризация данных для 10000 точек



****

****

**Вывод:**

Было разработаны базовые алгоритмы обмена данными с устройствами хранения, статистической обработки данных и визуализации результатов анализа. В ходе решения данных задач возникли трудности с применением алгоритма поиска кластеров, так как по своей не внимательности был взят более грузоемкий алгоритм (k-means), но позднее был взят алготитм из задания.