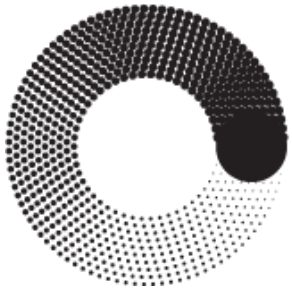
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
(ФГБОУ ВПО МПУ)



МОСКОВСКИЙ ПОЛИТЕХ

Кафедра СМАРТ технологий

Лабораторно-практическое задание № 1
“ 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:

Освоение базовых инструментов обработки и хранения массивов данных на языке программирования.

Цель работы:

Разработать базовые алгоритмы обмена данными с устройствами хранения, статистической обработки данных и визуализации результатов анализа.

Алгоритм поиска кластеров:

Поиск кластеров рекомендуется выполнять путем последовательного перебора списка записей, считая каждую точку, удаленную от текущего центра больше чем на радиус кластера, кандидатом на центр нового кластера. Первая точка списка принимается за центр первого кластера. Затем просматриваются все точки на предмет вхождения в первый кластер и подходящие помечаются номером кластера. После окончания просмотра ищется первая точка, не входящая в первый кластер и она принимается за центр второго кластера. Процедура повторяется до исчерпания набора точек.

Ход работы:

1. Подготовил описание проектов задач для обработки данных:

Было описанны проекты задач с темой исследования и задачами, которые могут быть решены с использованием технологий компьютерного зрения. Ниже отображено курсивом:

*«*Отслеживание изъянов на товарах на складах.*

**Построение 3д пространст, с помощью стерео-камер.*

**.....»*

2. Подготовил модуль загрузки, сохранения и отображения массивов данных, хранимых в текстовом представлении (формат CSV):

Сформировать текстовый файл, содержащий тройки {X, Y, Z} целых положительных чисел, разделённых пробельными символами. Файл должен содержать не менее 50 точек. Разработать приложение WindowsForms на языке C# для загрузки данных из 2 файла в формате CSV (разделитель проблем или точка с запятой) и их отображения в виде текстового списка без разделения на колонки. Также необходимо добавить возможность выгрузки данных в файл формата CSV. Разработать алгоритм случайной генерации точек. Диапазоны значений [0 ÷ 1023] полученных значений. Количество данных должно настраиваться, но не быть меньшим одной записи.

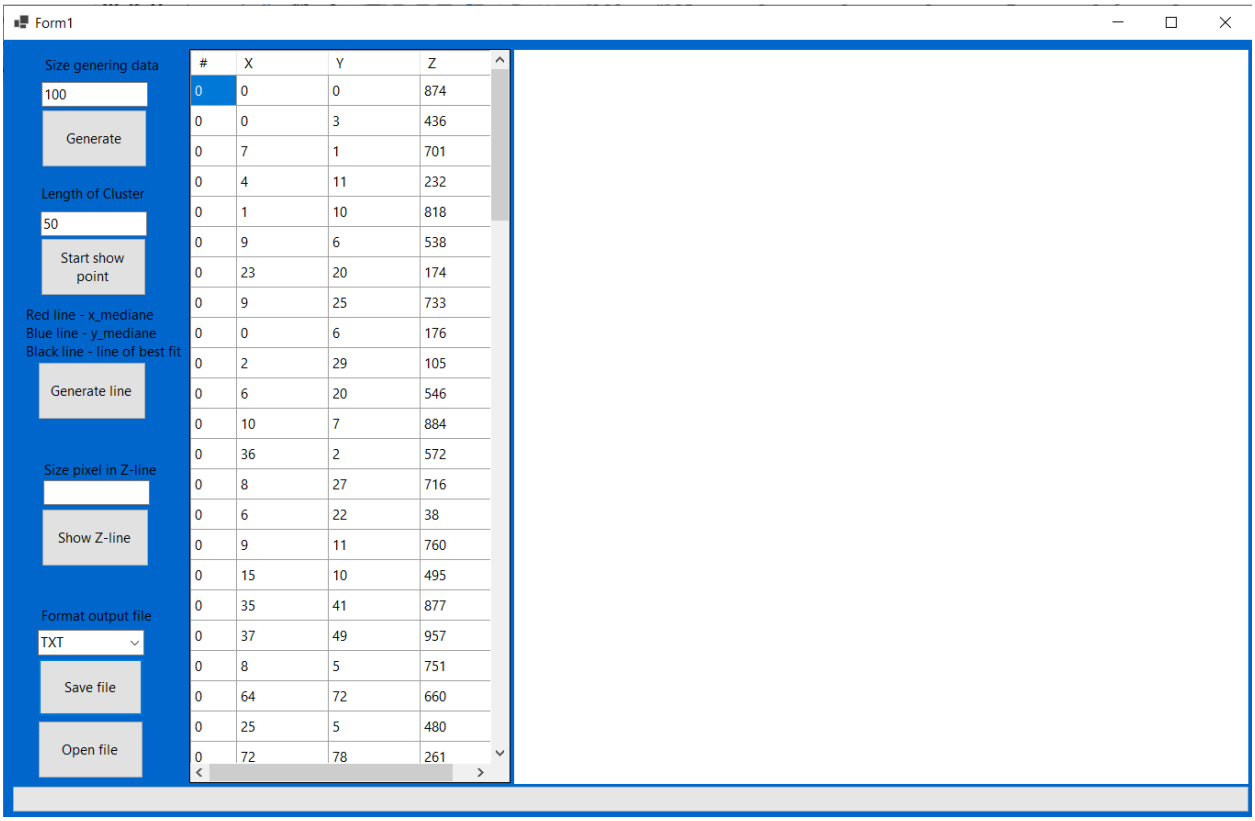


Рисунок 1 - Генерация данных, сто точек.

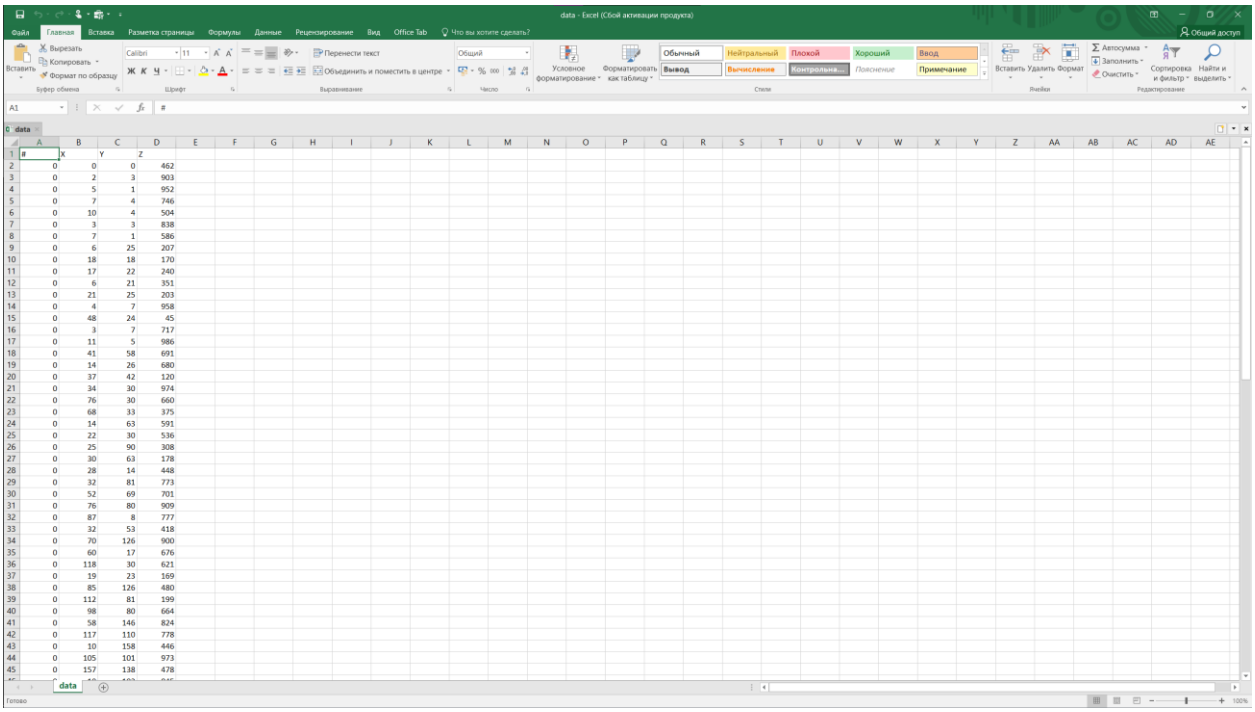


Рисунок 2 - Сохраненный файл

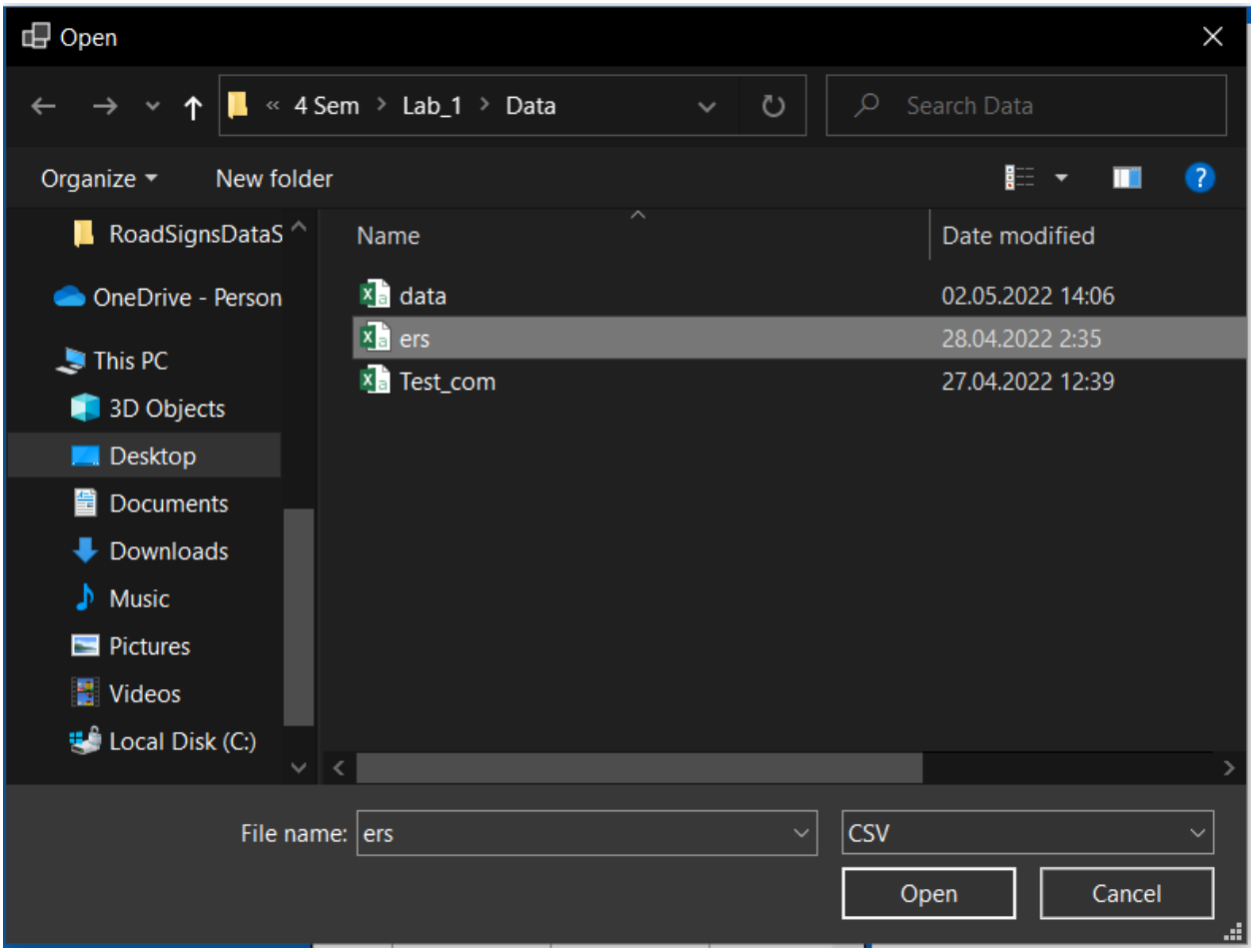


Рисунок 3 - Открытие файла

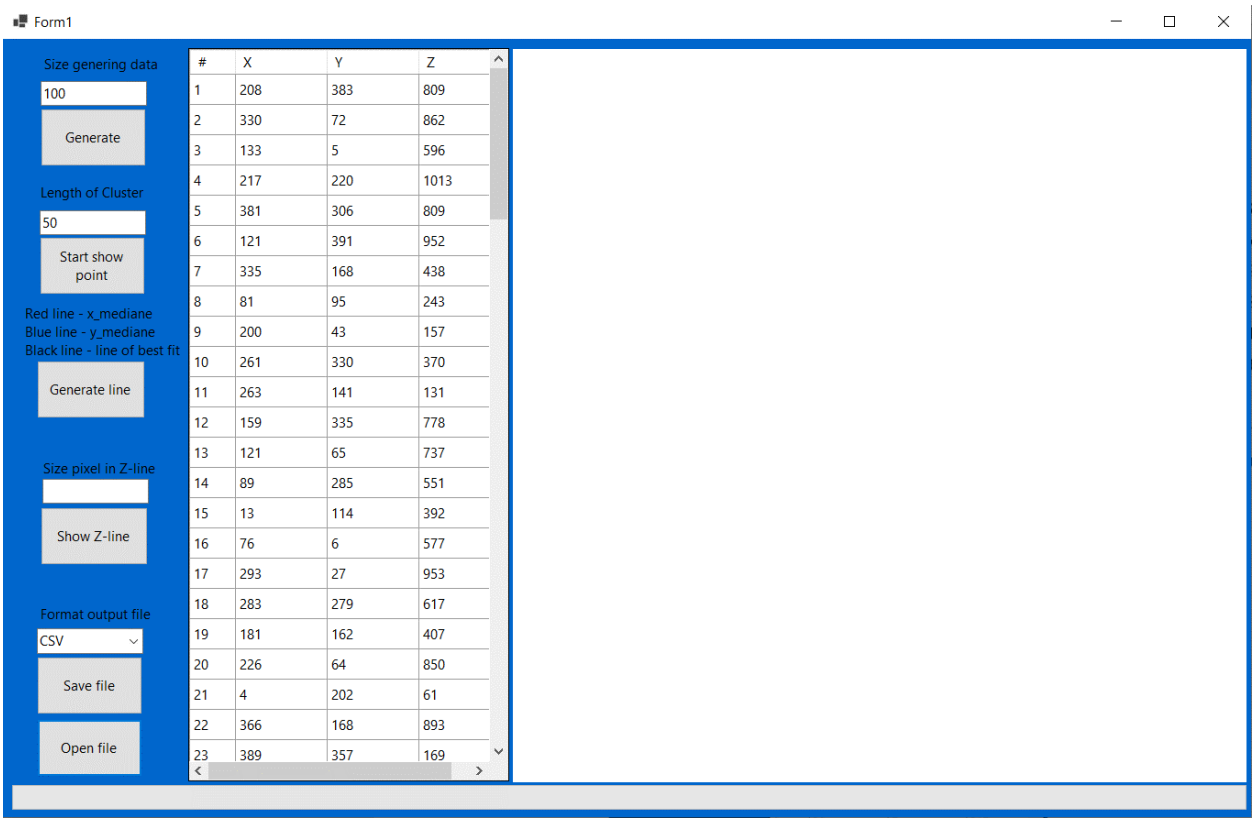


Рисунок 4 - Отображения данных из файла

3. Подготовил модуль визуализации исходных (сырых) данных на двумерной плоскости в формате битовой карты (bitmap):

Разработать (доработать) приложение WinForm на языке C# для загрузки данных в формате CSV (таблица DataGrid с трёмя столбцами: X, Y, Z) и их отображении в виде двумерного изображения (битовой карты), где каждая запись отображалась как точка (символ). Данные третьего столбца должны представляться цветом (яркостью) точек. Подобрать коэффициенты преобразования значений так, чтобы все точки отображались на битовой карте, а контраст был достаточным.

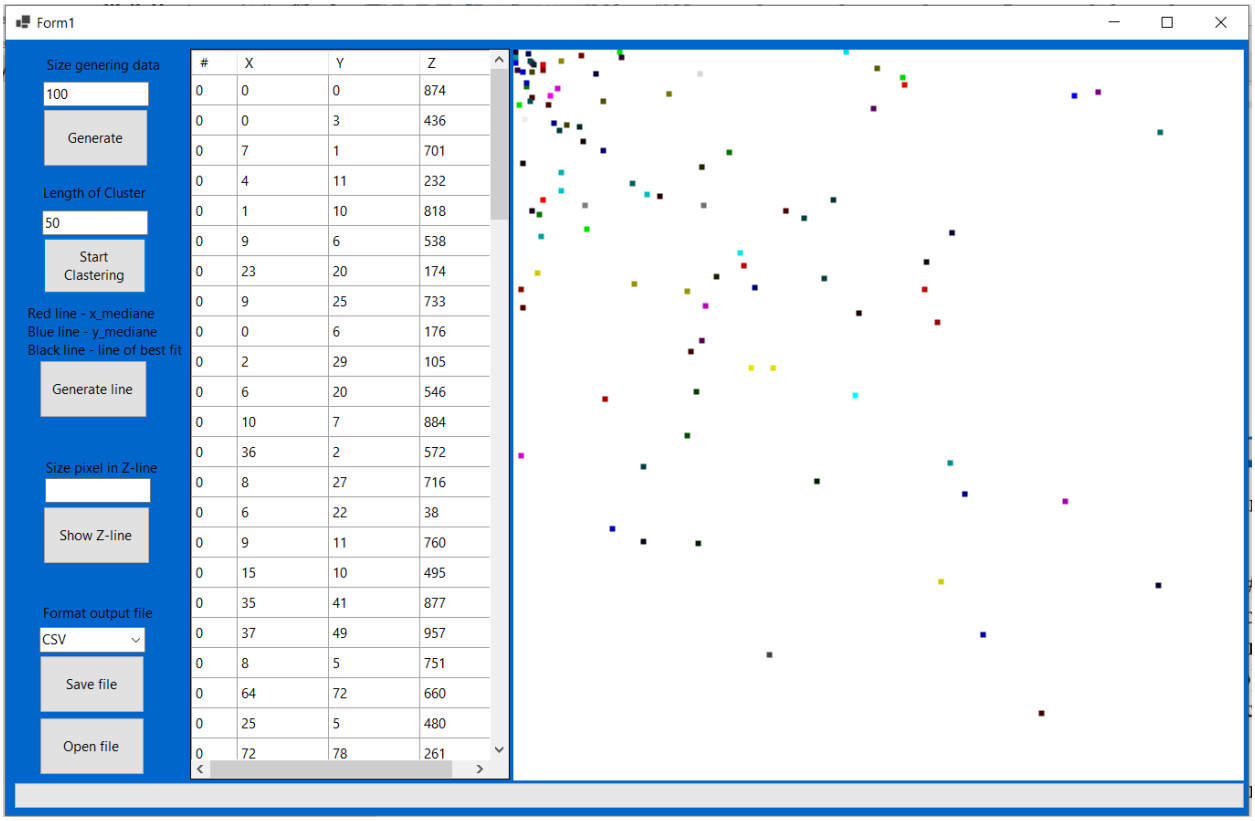


Рисунок 5 - Визуализация данных

4. Подготовил модуль визуализации исходных данных в виде линейного представления с заданным периодом:

Разместить на форме текстовое поле для указания периода повторения данных и вывести данные из набора (первый столбец) в виде линейной последовательности с указанным периодом. Каждый период начинается с новой строки битовой карты. Увеличить размер точек (элементов, символов), чтобы изображение было наглядным. Значения из записей ставить пропорциональным яркости точек.

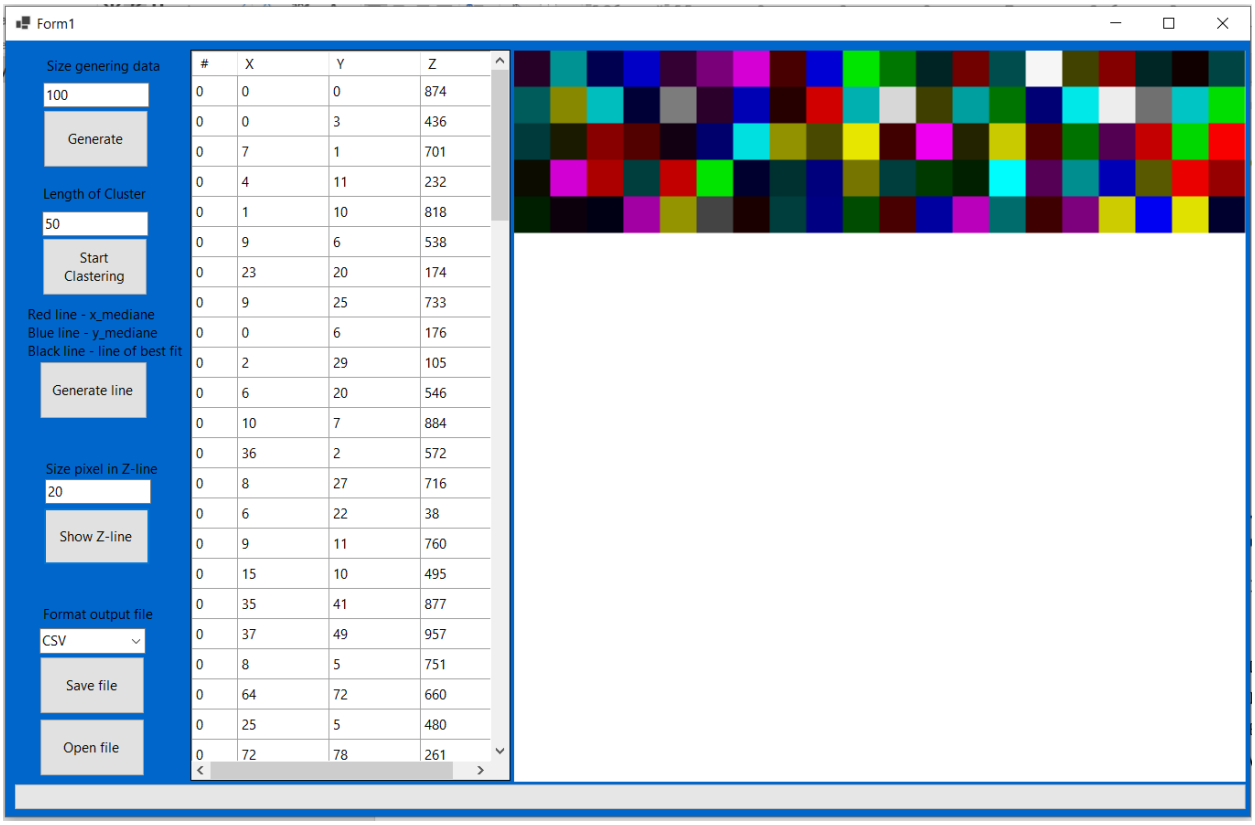


Рисунок 6 - Линейная визуализация Z(цвет) для каждой из точек

5. Рассчитал и отобразил на двумерной плоскости линию наилучшего приближения к данным, значение среднего:

Провести вычисления параметров линии наилучшего приближения для имеющегося массива данных. Отобразить линию наилучшего приближения на форме приложения из задачи 3. Рассчитать значение среднего по X и по Y и отобразить соответствующие линии.

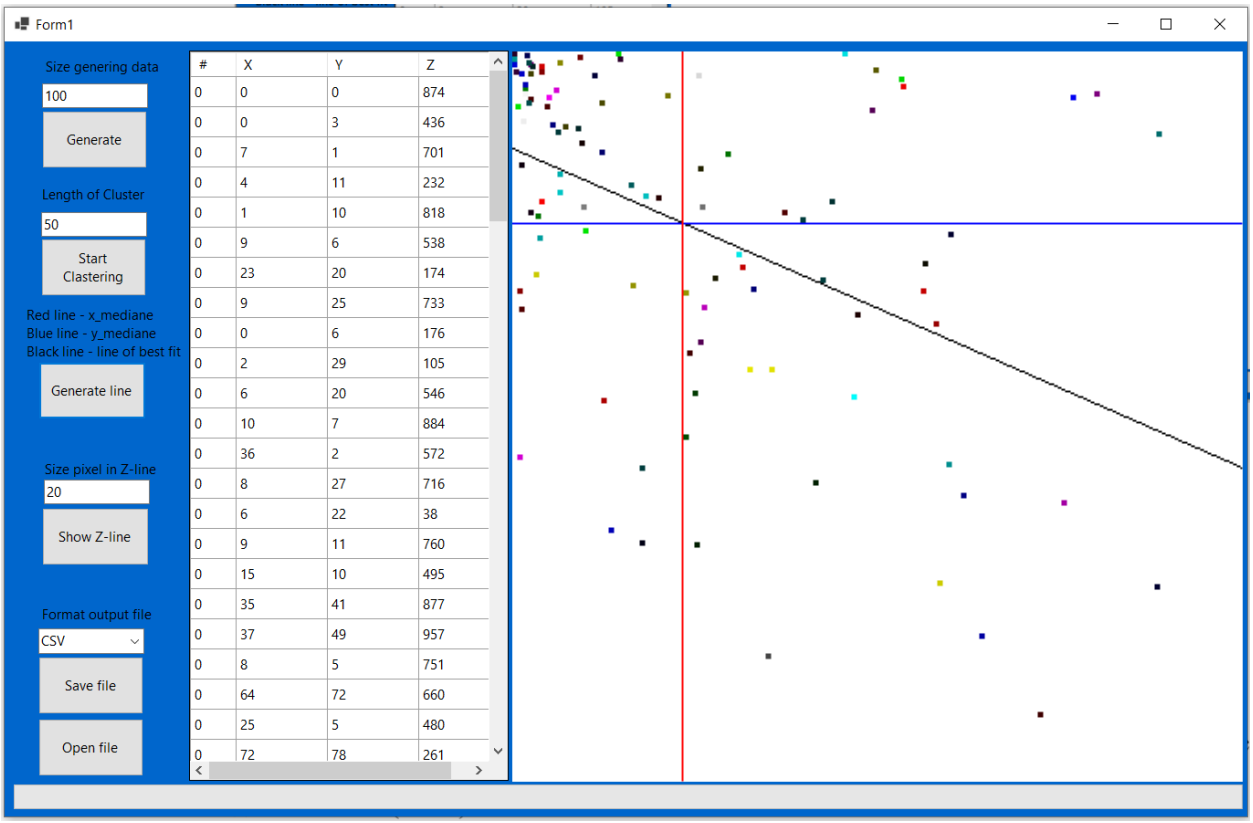


Рисунок 7 – Отображение наилучшего приближения к данным, значение среднего по x и y

6. Выполнил кластеризацию данных при фиксированном размере (радиусе) кластера:

Выполнить первичный поиск кластеров с заданным радиусом и отобразить их на битовой плоскости в виде окружностей с номерами в центре каждого кластера. Перестроить изображения с выделением цветом точек, входящих в каждый кластер. Перестроить таблицу с данными (DataGrid), добавив столбец с указанием номера кластера, к которому относится данная запись (точка).

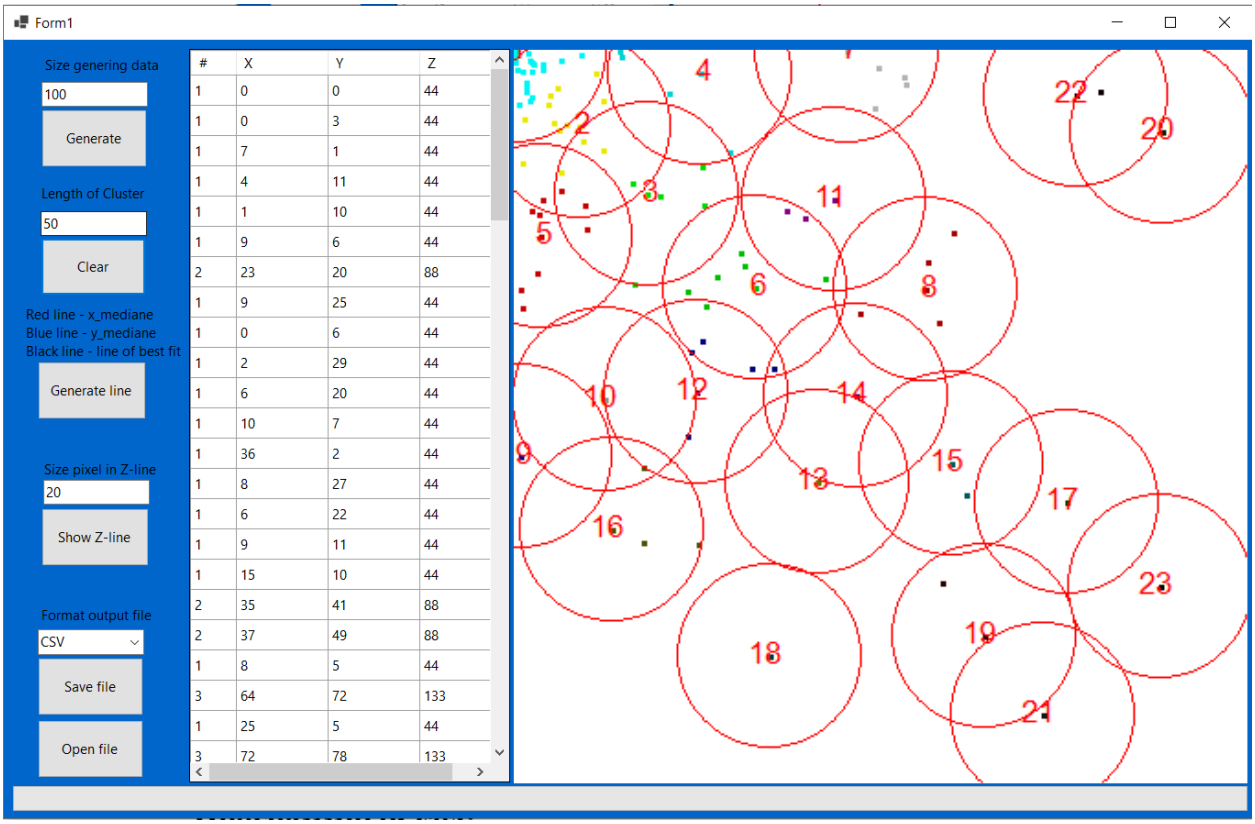


Рисунок 8 - Кластеризация данных

Программный код:

```
namespace Lab_1_CompVision
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            pictureBox1.Image = new Bitmap(400, 400);
            dataGridView1.ColumnCount = 4;
            dataGridView1.RowCount = 1;
            dataGridView1.RowHeadersVisible = false;
            dataGridView1.Columns[0].HeaderText = "#";
            dataGridView1.Columns[0].Width = 50;
            dataGridView1.Columns[1].HeaderText = "X";
            dataGridView1.Columns[1].Width = 100;
            dataGridView1.Columns[2].HeaderText = "Y";
            dataGridView1.Columns[2].Width = 100;
            dataGridView1.Columns[3].HeaderText = "Z";
            dataGridView1.Columns[3].Width = 100;
            dataGridView1.Width = 350;

        }
        public void OpenCSV(string csvPath)
        {
            string csvContentStr = File.ReadAllText(csvPath);
            dataGridView1.RowCount = 1;
            string[] vs = csvContentStr.Split('\n');
```

```
        string[] vs2;
        for (int i = 1; i < vs.Length-1; i++)
        {
            vs2 = vs[i].Split(';');
            dataGridView1.RowCount += 1;
            dataGridView1.Rows[i-1].Cells[0].Value = Int32.Parse(vs2[0]);
            dataGridView1.Rows[i-1].Cells[1].Value = Int32.Parse(vs2[1]);
            dataGridView1.Rows[i-1].Cells[2].Value = Int32.Parse(vs2[2]);
            dataGridView1.Rows[i-1].Cells[3].Value = Int32.Parse(vs2[3]);
        }
    }

    public void OpenTXT(string txtPath)
    {
        string txtContentStr = File.ReadAllText(txtPath);
        dataGridView1.RowCount = 1;
        string[] vs = txtContentStr.Split('\n');
        string[] vs2;
        for (int i = 1; i < vs.Length-1; i++)
        {
            vs2 = vs[i].Split(' ');
            dataGridView1.RowCount += 1;
            dataGridView1.Rows[i - 1].Cells[0].Value = Int32.Parse(vs2[0]);
            dataGridView1.Rows[i - 1].Cells[1].Value = Int32.Parse(vs2[1]);
            dataGridView1.Rows[i - 1].Cells[2].Value = Int32.Parse(vs2[2]);
            dataGridView1.Rows[i - 1].Cells[3].Value = Int32.Parse(vs2[3]);
        }
    }
    public void SaveTXT(string txtPath)
    {
        string data = "# X Y Z\n";
        for (int i = 0; i < dataGridView1.RowCount-1; i++)
        {
            data += dataGridView1.Rows[i].Cells[0].Value.ToString() + " ";
            data += dataGridView1.Rows[i].Cells[1].Value.ToString() + " ";
            data += dataGridView1.Rows[i].Cells[2].Value.ToString() + " ";
            data += dataGridView1.Rows[i].Cells[3].Value.ToString();
            data += "\n";
        }
        File.WriteAllText(txtPath, data);
    }
    public void SaveCSV(string csvPath)
    {
        string data = "#;X;Y;Z\n";
        for (int i = 0; i < dataGridView1.RowCount - 1; i++)
        {
            data += dataGridView1.Rows[i].Cells[0].Value.ToString() + ";";
            data += dataGridView1.Rows[i].Cells[1].Value.ToString() + ";";
            data += dataGridView1.Rows[i].Cells[2].Value.ToString() + ";";
            data += dataGridView1.Rows[i].Cells[3].Value.ToString();
            data += "\n";
        }
        File.WriteAllText(csvPath, data);
    }

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

```
            try
            {
                DialogResult res = openFileDialog1.ShowDialog();
                if (res == DialogResult.OK)
                {
                    if (openFileDialog1.FileName.Contains(".csv")) OpenCSV(openFileDialog1.FileName);
                    else OpenTXT(openFileDialog1.FileName);
                }
                else MessageBox.Show("Error, you don't take any file.");
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error, your file have incorrect type. You must take .txt or .csv.");
                MessageBox.Show(ex.Message);

            }
            Start_Claster.Text = "Start show point";
        }

        private void Save_but_Click(object sender, EventArgs e)
        {
            if (dataGridView1.RowCount >= 2)
            {
                try
                {
                    if (fileFormat.Text == "TXT") SaveTXT(@"C:\Users\ilyah\Desktop\Долги\Прога\4
Sem\Lab_1\Data\data.txt");
                    else SaveCSV(@"C:\Users\ilyah\Desktop\Долги\Прога\4 Sem\Lab_1\Data\data.csv");
                }
                catch
                {
                    MessageBox.Show("First carry out the division into clusters");
                }
            }
        }
        public void drawClaster(Graphics graphics)
        {
            int radius = 10;
            Pen pen = new Pen(Color.Red);
            Font drawFont = new Font("Arial", 10);
            StringFormat drawFormat = new StringFormat();
            SolidBrush drawBrush = new SolidBrush(Color.Red);
            drawFormat.FormatFlags = StringFormatFlags.DirectionRightToLeft;
            for (int j = 0; j < claster_index + 1; j++)
            {
                graphics.DrawString((j + 1).ToString(), drawFont, drawBrush, Clasters_point_x[j] + radius,
Clasters_point_y[j] - radius, drawFormat);
                graphics.DrawEllipse(pen, Clasters_point_x[j]- Int32.Parse(Len_Claster.Text),
Clasters_point_y[j]- Int32.Parse(Len_Claster.Text), Int32.Parse(Len_Claster.Text)*2,
Int32.Parse(Len_Claster.Text)*2);
            }
            pictureBox1.Refresh();
        }
        public void show_point()
        {
            Graphics graphics = Graphics.FromImage(pictureBox1.Image);
```

```
        graphics.FillRectangle(Brushes.White, new Rectangle(0, 0, pictureBox1.Width,
pictureBox1.Height));
        progressBar1.Maximum = dataGridView1.RowCount - 1;
        for (int i = 0; i < dataGridView1.RowCount-1; i++)
        {
           if ((int)dataGridView1.Rows[i].Cells[0].Value != 0) dataGridView1.Rows[i].Cells[3].Value
= (int)(1023 * ((int)dataGridView1.Rows[i].Cells[0].Value) / (claster_index+1));
           for (int j = (int)dataGridView1.Rows[i].Cells[1].Value; j <
(int)dataGridView1.Rows[i].Cells[1].Value + 3; j++)
           {
              for (int d = (int)dataGridView1.Rows[i].Cells[2].Value; d <
(int)dataGridView1.Rows[i].Cells[2].Value + 3; d++)
                 try
                 {
                    int a = (255 * (int)dataGridView1.Rows[i].Cells[3].Value) / 1023;
                    switch ((int)dataGridView1.Rows[i].Cells[3].Value % 7)
                    {
                       case 0: ((Bitmap)pictureBox1.Image).SetPixel(j, d, Color.FromArgb(0, 255 - a, 0));
break;
                       case 1: ((Bitmap)pictureBox1.Image).SetPixel(j, d, Color.FromArgb(0, 0, 255 - a));
break;
                       case 2: ((Bitmap)pictureBox1.Image).SetPixel(j, d, Color.FromArgb(0, 255 - a,
255 - a)); break;
                       case 3: ((Bitmap)pictureBox1.Image).SetPixel(j, d, Color.FromArgb(255 - a, 255 -
a, 255 - a)); break;
                       case 4: ((Bitmap)pictureBox1.Image).SetPixel(j, d, Color.FromArgb(255 - a, 255 -
a, 0)); break;
                       case 5: ((Bitmap)pictureBox1.Image).SetPixel(j, d, Color.FromArgb(255 - a, 0, 0));
break;
                       case 6: ((Bitmap)pictureBox1.Image).SetPixel(j, d, Color.FromArgb(255 - a, 0,
255 - a)); break;
                    }
                    //((Bitmap)pictureBox1.Image).SetPixel(j, d, Color.FromArgb(0,0,0));
                 }
                 catch { continue; }
           }
           progressBar1.Value = i;
        }
        progressBar1.Value = 0;
     }
     private void Generate_but_Click(object sender, EventArgs e)
     {
        Random rand = new Random();
        dataGridView1.RowCount = 1;
        Graphics graphics = Graphics.FromImage(pictureBox1.Image);
        graphics.FillRectangle(Brushes.White, new Rectangle(0, 0, pictureBox1.Width,
pictureBox1.Height));
        progressBar1.Maximum = Int32.Parse(Size_data.Text)-1;
        for (int i = 0; i < Int32.Parse(Size_data.Text); i++)
        {
           dataGridView1.RowCount += 1;
           dataGridView1.Rows[i].Cells[0].Value = 0;
           dataGridView1.Rows[i].Cells[1].Value = rand.Next(0, (400 * i /
Int32.Parse(Size_data.Text)));
           dataGridView1.Rows[i].Cells[2].Value = rand.Next(0, (400 * i /
Int32.Parse(Size_data.Text)));
           dataGridView1.Rows[i].Cells[3].Value = rand.Next(0, 1023);
```

```
            progressBar1.Value = i;
        }
        Start_Claster.Text = "Start show point";
        pictureBox1.Refresh();
        progressBar1.Value = 0;
    }
    int claster_index = 0;
    List<int> Clasters_point_x = new List<int>();
    List<int> Clasters_point_y = new List<int>();
    private void Start_Claster_Click(object sender, EventArgs e)
    {
        if (Start_Claster.Text == "Start show point")
        {
            claster_index = 0;
            Clasters_point_x.Clear();
            Clasters_point_y.Clear();
            show_point();
            Start_Claster.Text = "Start Clastering";
            pictureBox1.Refresh();
        }
        else if (Start_Claster.Text == "Start Clastering")
        {
            Graphics graphics = Graphics.FromImage(pictureBox1.Image);
            Clasters_point_x.Add((int)dataGridView1.Rows[0].Cells[1].Value);
            Clasters_point_y.Add((int)dataGridView1.Rows[0].Cells[2].Value);
            dataGridView1.Rows[0].Cells[0].Value = claster_index+1;
            progressBar1.Maximum = dataGridView1.RowCount - 1;
            for (int i = 1; i < dataGridView1.RowCount - 1; i++)
            {
                bool f = false;
                int x = (int)dataGridView1.Rows[i].Cells[1].Value;
                int y = (int)dataGridView1.Rows[i].Cells[2].Value;
                int g = 0;
                double min = 400;
                for (int j = 0; j < claster_index+1; j++)
                {
                    if (Math.Sqrt(Math.Pow(x - Clasters_point_x[j], 2) + Math.Pow(y - Clasters_point_y[j],
2)) <= Int32.Parse(Len_Claster.Text))
                    {
                        if (min > Math.Sqrt(Math.Pow(x - Clasters_point_x[j], 2) + Math.Pow(y -
Clasters_point_y[j], 2)))
                        {
                            min = Math.Sqrt(Math.Pow(x - Clasters_point_x[j], 2) + Math.Pow(y -
Clasters_point_y[j], 2));
                            g = j;
                        }
                        f = true;
                    }
                }
                if (!f)
                {
                    claster_index++;
                    Clasters_point_x.Add(x);
                    Clasters_point_y.Add(y);
                    dataGridView1.Rows[i].Cells[0].Value = claster_index + 1;
                }
                else
```

```
                {
                    dataGridView1.Rows[i].Cells[0].Value = g+1;
                    //Pen pen = new Pen(Color.Blue);
                    //graphics.DrawLine(pen, x, y, Clasters_point_x[g], Clasters_point_y[g]);
                }
                progressBar1.Value = i;
            }
            progressBar1.Value = 0;
            for (int i = 0; i < dataGridView1.RowCount - 1; i++)
            {
                int x = (int)dataGridView1.Rows[i].Cells[1].Value;
                int y = (int)dataGridView1.Rows[i].Cells[2].Value;
                int g = 0;
                double min = 400;
                for (int j = 0; j < claster_index + 1; j++)
                {
                    if (Math.Sqrt(Math.Pow(x - Clasters_point_x[j], 2) + Math.Pow(y - Clasters_point_y[j],
2)) <= Int32.Parse(Len_Claster.Text))
                    {
                        if (min > Math.Sqrt(Math.Pow(x - Clasters_point_x[j], 2) + Math.Pow(y -
Clasters_point_y[j], 2)))
                        {
                            min = Math.Sqrt(Math.Pow(x - Clasters_point_x[j], 2) + Math.Pow(y -
Clasters_point_y[j], 2));
                            g = j;
                        }
                    }
                }
                dataGridView1.Rows[i].Cells[0].Value = g + 1;
                //Pen pen = new Pen(Color.Blue);
                //graphics.DrawLine(pen, x, y, Clasters_point_x[g], Clasters_point_y[g]);
                progressBar1.Value = i;
            }
            progressBar1.Value = 0;
            show_point();
            drawClaster(graphics);
            pictureBox1.Refresh();
            Start_Claster.Text = "Clear";
        }
        else if (Start_Claster.Text == "Clear")
        {
            Graphics graphics = Graphics.FromImage(pictureBox1.Image);
            graphics.FillRectangle(Brushes.White, new Rectangle(0, 0, pictureBox1.Width,
pictureBox1.Height));
            pictureBox1.Refresh();
            Start_Claster.Text = "Start Clastering";
        }
    }
    bool is_sort = false;
    private void button1_Click(object sender, EventArgs e)
    {
        object[] mas = new object[4];
        int a11 = 0;
        int a2 = 0;
        int mx = 0;
        int my = 0;
        for (int i = 0; i < dataGridView1.RowCount - 1; i++)
```

```
        {
            int x = (int)dataGridView1.Rows[i].Cells[1].Value;
            int y = (int)dataGridView1.Rows[i].Cells[2].Value;
            a11 += x * y;
            a2 += x * x;
            my += y;
            mx += x;
            if (is_sort)
            {
                for (int j = i + 1; j < dataGridView1.RowCount - 1; j++)
                {
                    if ((int)dataGridView1.Rows[i].Cells[1].Value >
(int)dataGridView1.Rows[j].Cells[1].Value)
                    {
                        mas[0] = dataGridView1.Rows[i].Cells[0].Value;
                        mas[1] = dataGridView1.Rows[i].Cells[1].Value;
                        mas[2] = dataGridView1.Rows[i].Cells[2].Value;
                        mas[3] = dataGridView1.Rows[i].Cells[3].Value;

                        dataGridView1.Rows[i].Cells[0].Value = dataGridView1.Rows[j].Cells[0].Value;
                        dataGridView1.Rows[i].Cells[1].Value = dataGridView1.Rows[j].Cells[1].Value;
                        dataGridView1.Rows[i].Cells[2].Value = dataGridView1.Rows[j].Cells[2].Value;
                        dataGridView1.Rows[i].Cells[3].Value = dataGridView1.Rows[j].Cells[3].Value;

                        dataGridView1.Rows[j].Cells[0].Value = mas[0];
                        dataGridView1.Rows[j].Cells[1].Value = mas[1];
                        dataGridView1.Rows[j].Cells[2].Value = mas[2];
                        dataGridView1.Rows[j].Cells[3].Value = mas[3];
                    }
                }
            }
        }
        a11 *= (dataGridView1.RowCount);
        a2 *= (dataGridView1.RowCount);
        int x_med = mx / dataGridView1.RowCount;
        int y_med = my / dataGridView1.RowCount;
        double k = (double)((a11 - mx * my) / (double)(a2 - (mx * mx)));
        int b = (int)((double)(my - (double)(k * mx))) / (dataGridView1.RowCount);
        Graphics graphics = Graphics.FromImage(pictureBox1.Image);
        Pen pen = new Pen(Color.Black);
        graphics.DrawLine(pen, 0, (int)b, 400, (int)((double)(400 * k) + b));
        pen = new Pen(Color.Blue);
        graphics.DrawLine(pen, 0, y_med, 400, y_med);
        pen = new Pen(Color.Red);
        graphics.DrawLine(pen, x_med, 0, x_med, 400);
        pictureBox1.Refresh();
    }

    private void button1_Click_1(object sender, EventArgs e)
    {
        Graphics graphics = Graphics.FromImage(pictureBox1.Image);
        graphics.FillRectangle(Brushes.White, new Rectangle(0, 0, pictureBox1.Width,
pictureBox1.Height));
        pictureBox1.Refresh();
        int size = 1;
        if (Z_size.Text.Trim() == "")
        {
```

```
            if (Int32.Parse(Size_data.Text) <= 6400) size = ((int)Math.Sqrt((double)(400 * 400 /
Int32.Parse(Size_data.Text))) / 5) * 5;
         }
         else
         {
            size = Int32.Parse(Z_size.Text);
            if (size * size * Int32.Parse(Size_data.Text) > 160000)
            {
               MessageBox.Show("Your size of pixel is to big for image. Program use 1x1 pixels? as
default");
               size = 1;
            }
         }
         if (Int32.Parse(Size_data.Text) > 160000 || size*size* Int32.Parse(Size_data.Text) > 160000)
MessageBox.Show("Your number of points exceeds the allowed format, is 160000 points");
         else
         {
            progressBar1.Maximum = dataGridView1.RowCount - 1;
            for (int x = 0; x < dataGridView1.RowCount - 1; x++)
            {
               for (int j = (x * size) % 400; j <= (x * size + size - 1) % 400; j++)
               {
                  for (int d = ((int)(x * size) / 400) * size; d <= ((int)(x * size) / 400) * size + size - 1;
d++)
                     try
                     {
                        int a = (255 * (int)dataGridView1.Rows[x].Cells[3].Value) / 1023;
                        switch ((int)dataGridView1.Rows[x].Cells[3].Value % 7)
                        {
                           case 0: ((Bitmap)pictureBox1.Image).SetPixel(j, d, Color.FromArgb(0, 255 - a,
0)); break;
                           case 1: ((Bitmap)pictureBox1.Image).SetPixel(j, d, Color.FromArgb(0, 0, 255 -
a)); break;
                           case 2: ((Bitmap)pictureBox1.Image).SetPixel(j, d, Color.FromArgb(0, 255 - a,
255 - a)); break;
                           case 3: ((Bitmap)pictureBox1.Image).SetPixel(j, d, Color.FromArgb(255 - a,
255 - a, 255 - a)); break;
                           case 4: ((Bitmap)pictureBox1.Image).SetPixel(j, d, Color.FromArgb(255 - a,
255 - a, 0)); break;
                           case 5: ((Bitmap)pictureBox1.Image).SetPixel(j, d, Color.FromArgb(255 - a, 0,
0)); break;
                           case 6: ((Bitmap)pictureBox1.Image).SetPixel(j, d, Color.FromArgb(255 - a, 0,
255 - a)); break;
                        }
                     }
                     catch (Exception ex) { MessageBox.Show(Text, ex.Message,
MessageBoxButtons.OK); }
               }
               progressBar1.Value = x;
            }
            progressBar1.Value = 0;
            pictureBox1.Refresh();
         }
      }
   }
}
```

Приложения:

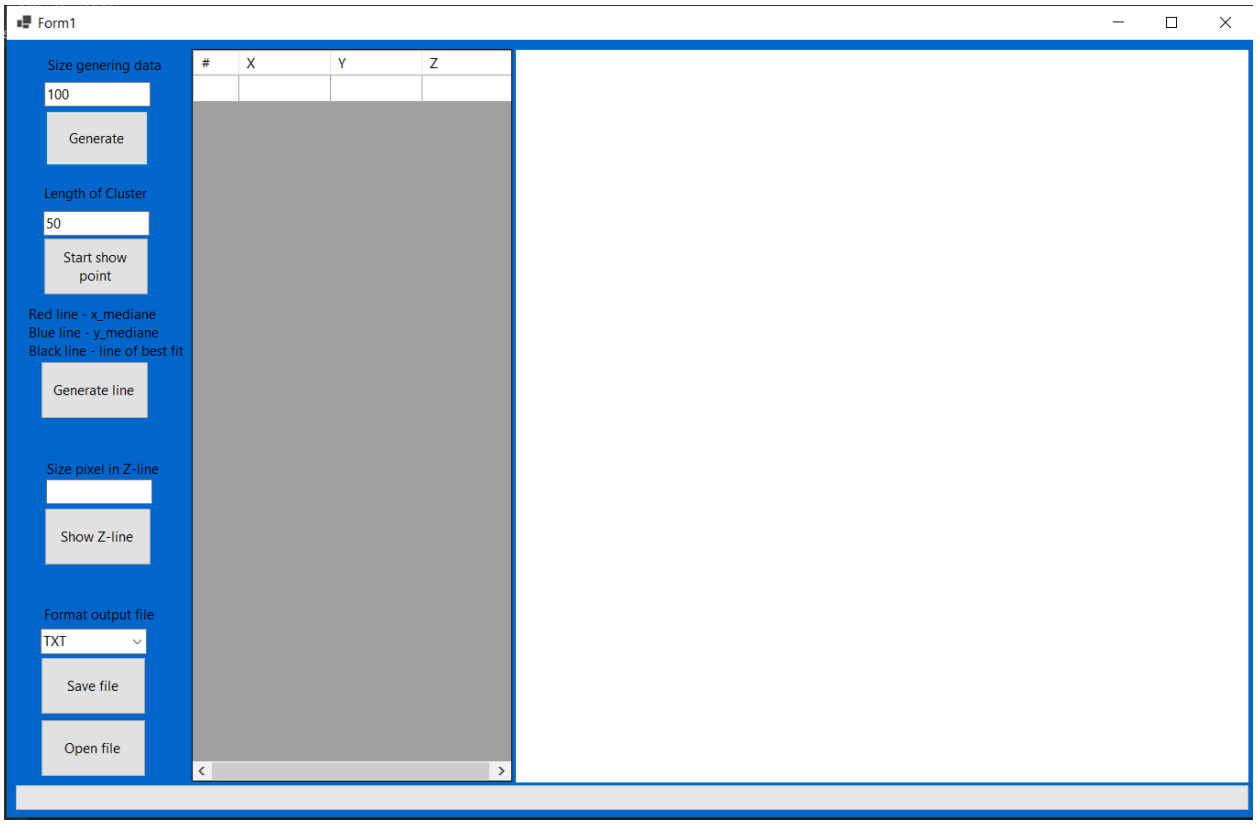


Рисунок 9 - Изначальный вид формы

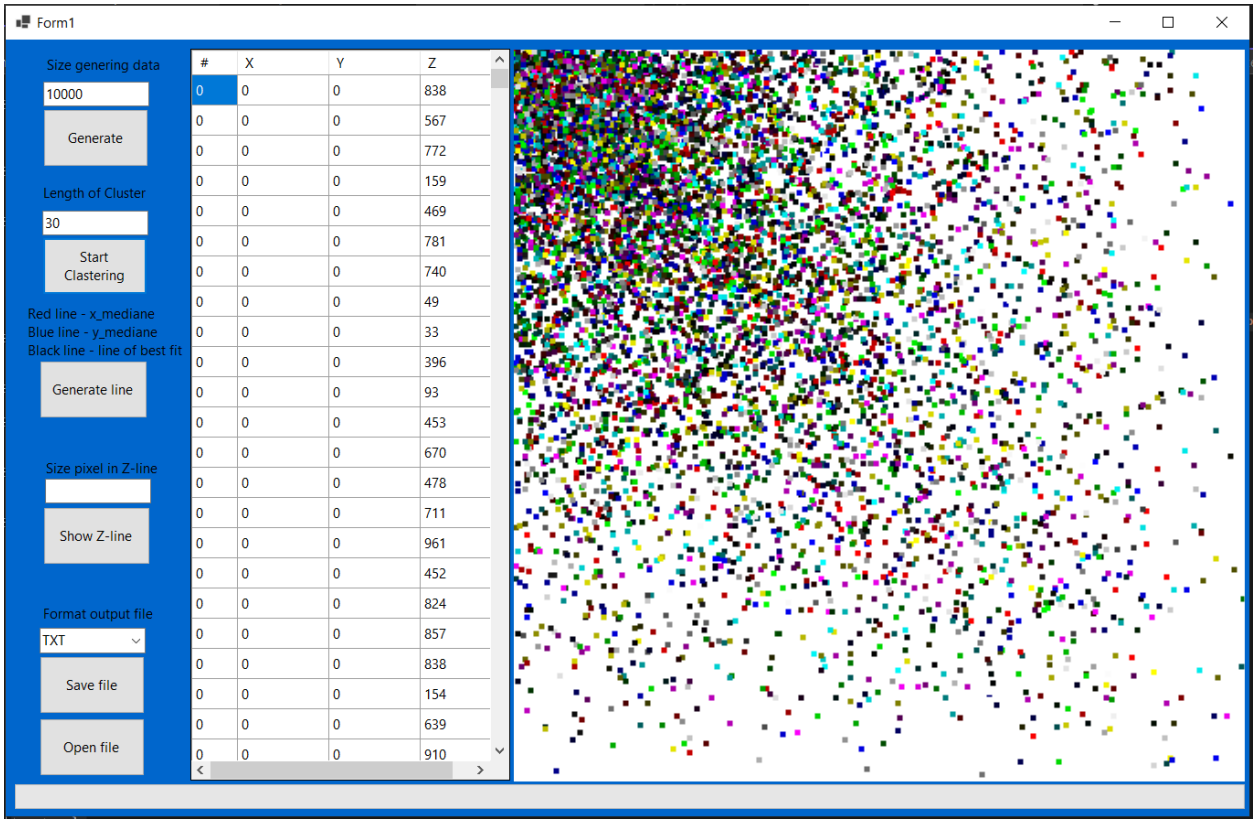


Рисунок 10 - Генерация и отображение данных для 10000 точек

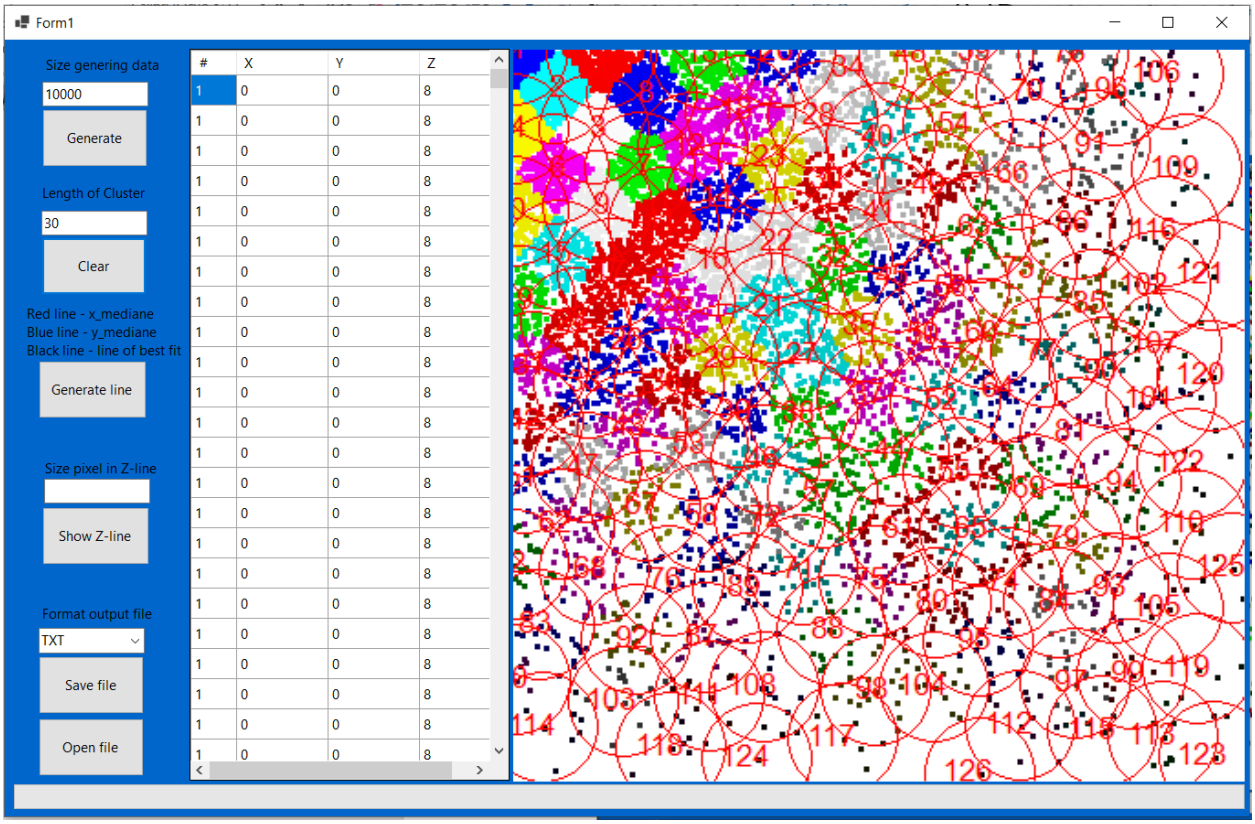


Рисунок 11 - Кластеризация данных для 10000 точек

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.