ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ ГОРОДА МОСКВЫ
Юго-Западный административный округ
Государственное бюджетное общеобразовательное учреждение города Москвы
«Школа № 1533 «ЛИТ»

ВЫПУСКНОЙ ПРОЕКТ

Учащегося группы 10.5
Камардина Богдана Михайловича
(Специальность: программирование)

МОДЕЛЬ ВЫДЕЛЕНИЯ НАНОФРАКЦИЙ ВЕЩЕСТВА ПО ЕГО ЦИФРОВОМУ СНИМКУ

Заказчик и руководитель — **Селиверстов А.В. (**старший научный сотрудник физического факультета МГУ имени М.В. Ломоносова**)**

Москва
2021

Оглавление

Введение

Стоит начать начать с того, что такое нанофракции и зачем их вообще выделять. Нанофракции – группы мельчайших мельчайших частиц размером от 1 до 100 нм. В большинстве своём нанофракции представлены в виде нанопорошков различного содержания. У большого количества веществ свойства и функции могут зависеть от того, как сгруппированы его составляющие. Простейший пример – кровь. Она содержит в себе и лейкоциты, и тромбоциты и чего только не. Кроме того, выделение нанофракций упрощают их дальнейшее изучение. Это используется для:

- анализа медицинских изображений

  Например, учёными был разработан нанопорошок, способный регулировать температуру области, в которой проводится операция по удалению раковой опухоли. Также выделение нанофракций используется для экспресс-анализа крови на структурные единицы (те же эритроциты, лейкоциты и т.п)

- выделения объектов на спутниковых снимках

  Анализ астрономических фотоархивов. Поиск скоплений, небесных тел и различных аномалий.

- распознавания объектов с помощью машинного зрения

  Разработке различных приложений для контроля качества.

Как можно заметить, выделение объектов используется в большом количестве отраслей и является востребованной задачей

Постановка задачи, целевая аудитория, аналоги

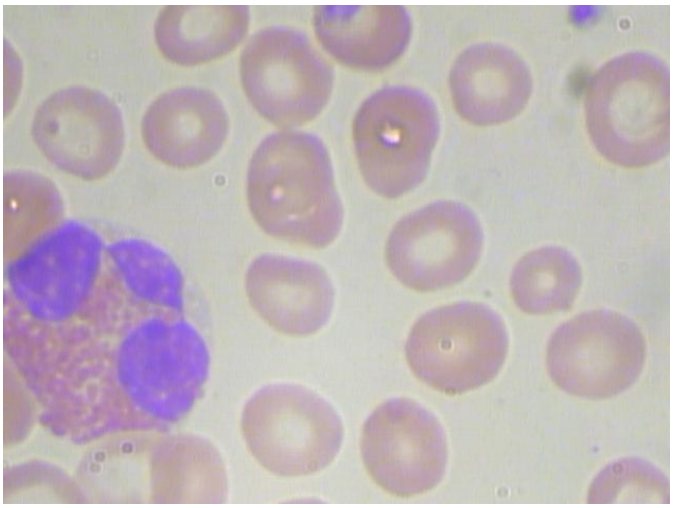
Целевой аудиторией нашего проекта являются ученые в области медицины, а также микробиологи. Наш проект может быть полезен и для физиков-любителей, которые проводят эксперименты или изучают различные вещества в домашних условиях.

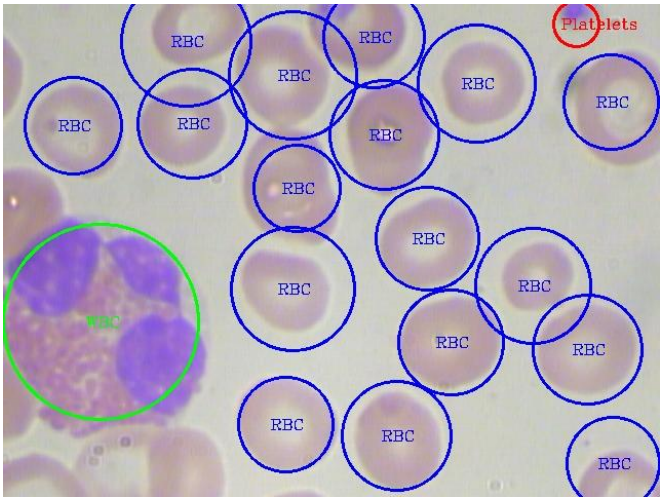
В мои задачи входило:

- Создание программы, способной выделять фракции из чёрно-белых изображений
- Сравнение алгоритмов, работающих с использованием нейросетей и без них
- Изучение методов работы с изображениями в целом

На просторах интернета были найдены две программы, выполняющие похожую задачу. Первой из них является программа Павла Кутлунина, программиста из Омска. Однако результатом его работы было огромное количество отдельных мелких изображений, с которыми невозможно работать. Мы попытались получить рабочую версию его программы, написав ему письмо, так как в открытом доступе есть только результат, но ответа не получил. Подробнее ознакомиться с тем, что получилось у Павла можно здесь https://fap.sbras.ru/node/4116 .

В качестве второго аналога была выбрана программа пользователя github'a MahmudulAlam'a. Он с помощью нейросетей выделял составляющие крови (см. изображение)




RBC
RBC
RBC
RBC
Platelets
RBC
RBC
RBC
RBC
RBC
RBC
RBC
RBC
RBC
RBC
RBC
RBC
RBC
RBC
RBC
RBC

Его разработка работает наглядно и стабильно, однако применима только к снимкам крови. Также нет никаких инструкций или справок о том, как пользоваться его программой. На основании всего вышесказанного можно составить следующую сравнительную таблицу:

	Доступность	**Удобство использования**	**Разнообразие условий входных данных**
Программа Павла Кутлунина	-	-	?
Программа MahmudulAlam	+	-	+-
Наш проект	+	+	+

Решение

Среда и язык

Программа была написана на языке C# с использованием Microsoft Visual Studio 2017. Так как работать надо было с изображениями, мы решили воспользоваться Windows Forms для лучшей наглядности и удобства использования.

Также я изучил программу ImageMagick, а именно её модуль Magick.NET, разработанный специально для C#. Эта библиотека очень полезна при работе с изображениями, так имеет огромный функционал. Был использован метод конвертирования изображения из цветного (ну или оттенков серого) в чёрно-белое для лучшей работы алгоритма.

Алгоритм

Главный алгоритм по выделению самих нанофракций работают следующим образом:

1. Переносим изображение в двоичный массив A: если яркость пиксела (i, j) меньше некоторого порога (в моём варианте этот порог равен 85), то заносим в этот элемент 1, иначе – 0.
2. Дальше происходит сглаживание изображения: берём из массива A подмассив 3x3 вокруг текущего пиксела. Если в нём заполнено не менее 5 пикселов, то в двоичном массиве B ставим на месте центрального пиксела 1, иначе – 0.
3. Из массива B, содержащего много объектов, копируем один объект в список C. В массиве B его стираем.

Проще говоря, после выполнения первого пункта объекты представляются в виде единичек, а пустые области между ними в виде нулей. Яркость пикселей изображения я рассчитывал по следующей формуле: $Y=0.3 * R + 0.59 * G + 0.11 * B$, где Y – яркость пикселя, а R, G и B – составляющие красного, зеленого и синего цветов соответственно. Коэффициенты именно такие, потому что глаз человека более чувствителен к волнам зеленого цвета и менее чувствителен к волнам синего. Исходя из этого уравнения, можно получить, что полностью чёрный цвет равен 0, а чистый белый – 255.

Затем выполняется сглаживание нашего двоичного изображения. Это необходимо для приведения всех объектов к общему стандарту. На свойства самих объектов это практически не повлияет, так как расхождения с оригиналом минимальные. Это можно считать погрешностью в измерениях. Однако данное действие поможет упростить выделение фракций и их дальнейшее изучение, так как границы становятся более явными и чёткими.

Далее начинается процесс самого выделения нанофракций вещества. Состоит он из трёх шагов:

1. Поиск
2. Перенос
3. Удаление в исходнике

Поиск объекта происходит следующим образом: мы идём по нашему пиксельному изображению, до тех пор пока не найдем единицу. Далее идём по единицам вперед, пока не дойдём до нуля, это и будет концом объекта. Аналогично ищем нижнюю границу объекта. Теперь, когда мы знаем начало и конец нашего объекта, нам не составит никакого труда. Переносим объект в список C. Именно объект, а не массив, потому что изначально мы не знаем, сколько объектов на изображении у нас будет. Затем заменяем все единицы нулями на месте нашего объекта в исходном массиве B, чтобы при новом проходе цикла не выделять один и тот же элемент дважды.

Эти шаги повторяются до тех пор, пока в массиве B есть хотя бы один объект. Таким образом мой алгоритм не упускает ни единого элемента.

Ход работы

Ещё летом 2020-го года я понял, что хотел бы взять проект по теме, связанной с физикой, поэтому в октябре-ноябре с выбором темы не было никаких проблем. В декабре я впервые связался с заказчиком. Моим заказчиком выступил Алексей Валентинович Селивёрстов – старший научный сотрудник физического факультета МГУ имени М.В. Ломоносова. До Нового года я понял, зачем нужен этот проект, и где он может быть применим. В тот же момент я начал изучение предметной области. Однако в данной задаче она в основном играла роль ограничителя, то есть задаёт рамки для моего алгоритма, но в самом коде особой роли не играет.

После Нового года начался поиск алгоритма для решения поставленной задачи. Однако я не сразу понял, делаю ли я кластеризацию или же выделение частиц по краям, из-за чего впоследствии возникли проблемы. В любом случае было два глобальных подхода: использовать алгоритмы машинного обучения, или же писать алгоритм самому. Я выбрал второй вариант, так как, по моему мнению, не успел бы обучить нейросеть за такой небольшой отрезок времени, к тому же я никогда с ними не работал ранее.

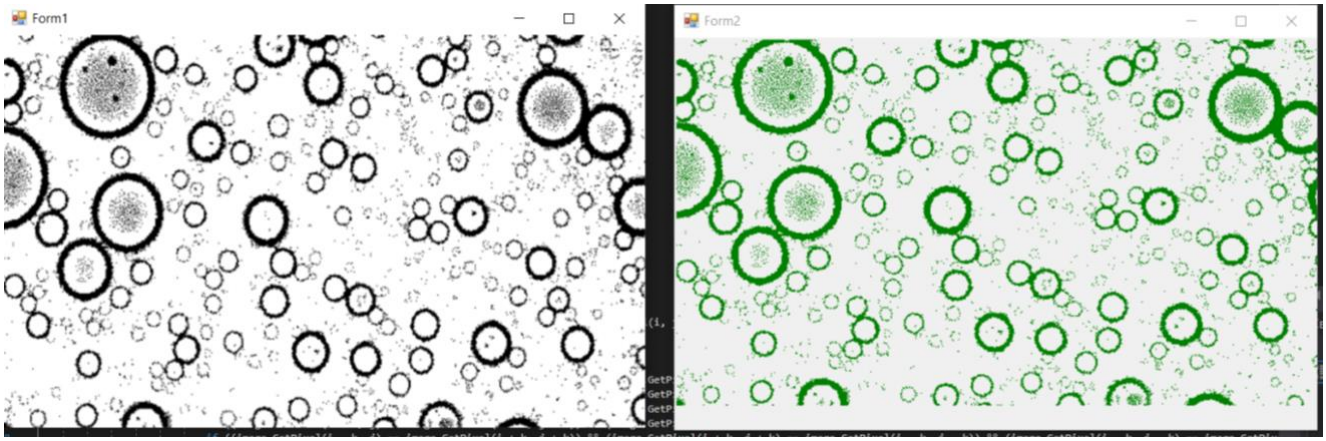
Из-за неполной ясности в задаче я делал что-то между выделением по краям и кластеризацией объектов. То есть я выделял фракции, но по какому-то общему признаку. Изначально было три варианта этого общего признака:

- По цвету

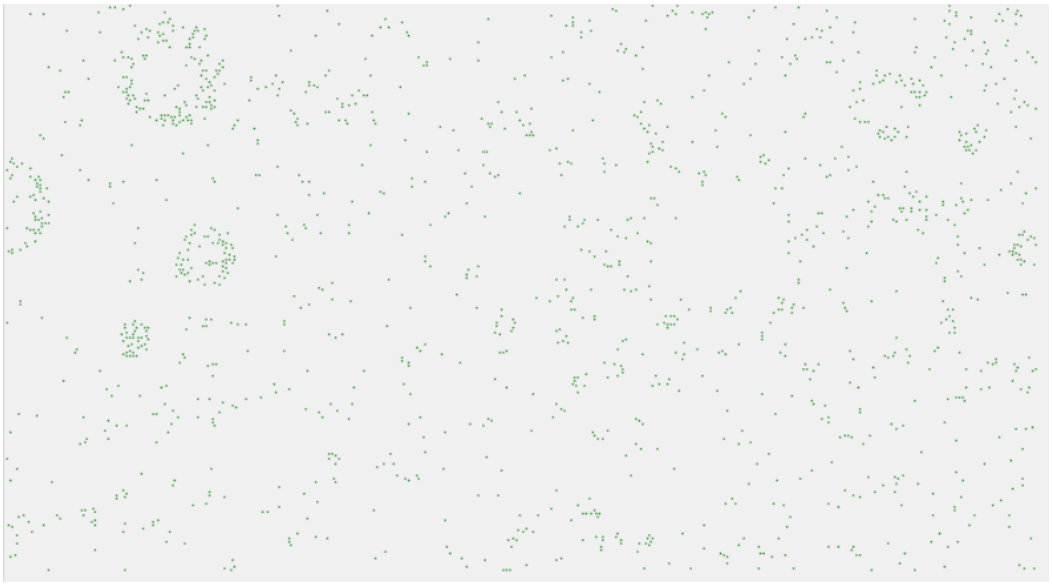
- По форме
- По размеру

Так как мы работаем с чёрно-белыми изображениями, первый вариант отпал сразу же. Из второго и третьего я вывел итоговый признак – правильная фигура. Я начал писать алгоритм, который выделял мне из изображения только те объекты, которые являлись правильными фигурами, например, круги или квадраты. Для каждого пикселя в изображении я смотрел его схожесть с соседними. То есть я смотрел изменения по четырём направлениям, четырём прямым: вертикальной, горизонтальной, и наклонным y=x и y=-x. Однако в природе, как оказалось, практически не существует правильных фигур. Поэтому я решил ввести погрешность. Условно, если у меня фигура со всех сторон кроме одной одинаковая, всё равно считать её правильной. Но какие бы я условия для этой погрешности ни вводил, получалось два варианта: либо у меня все фигуры проходят под это условие, либо никакие. Приведу пример.

Вот результат, когда ограничения не стоят или же стоят минимальные (слева исходное изображение, справа результат работы):



А это – работа алгоритма с максимальными или близкими к ним ограничениями:



Как можно заметить, правильными фигурами являются только мельчайшие частицы, которые в связи с плохим качеством исходного изображения превратились в единичные пиксели, то есть в маленькие квадратики.

Вывод: искать правильные фигуры абсолютно не требуется, так как и они практически отсутствуют. В связи с этим мне пришлось найти новый способ решения поставленной задачи.

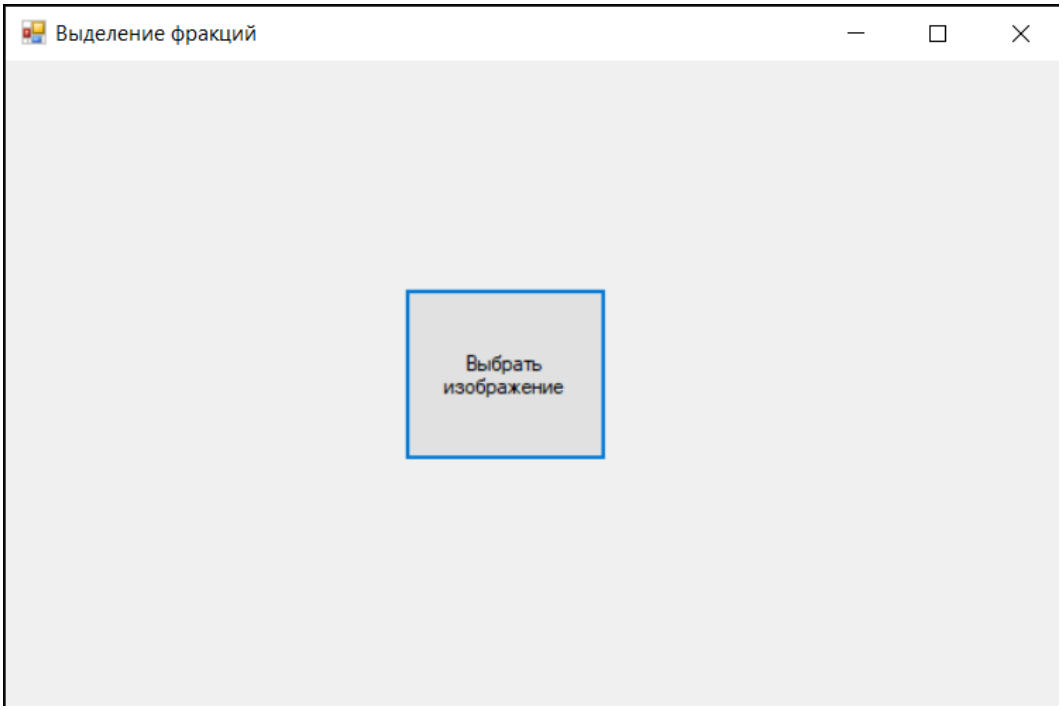
На выполнение этого алгоритма я потратил март и апрель. В мае после консультации с заказчиком и преподавателями в лицее было найдено новое решение. Май-июнь я как раз посвятил работе над алгоритмом, описанном в начале этого пункта.

Пример использования

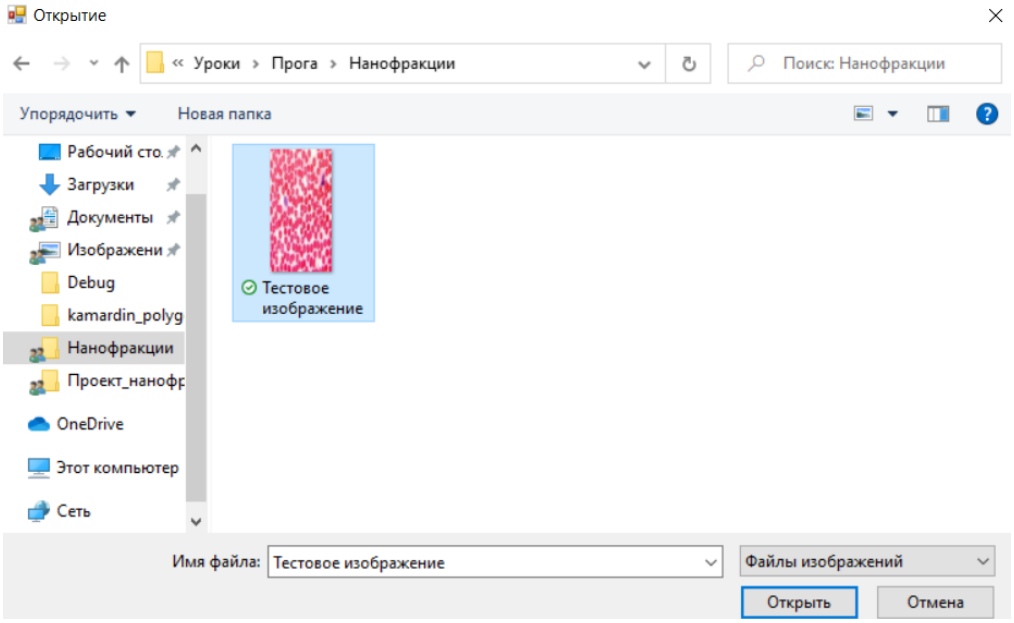
А сейчас я бы хотел продемонстрировать работу нашей программы и показать, как ею пользоваться. В качестве тестового изображения возьмём следующий снимок случайного вещества:

Вертикальное изображение я беру для лучшей наглядности в работе алгоритмов.

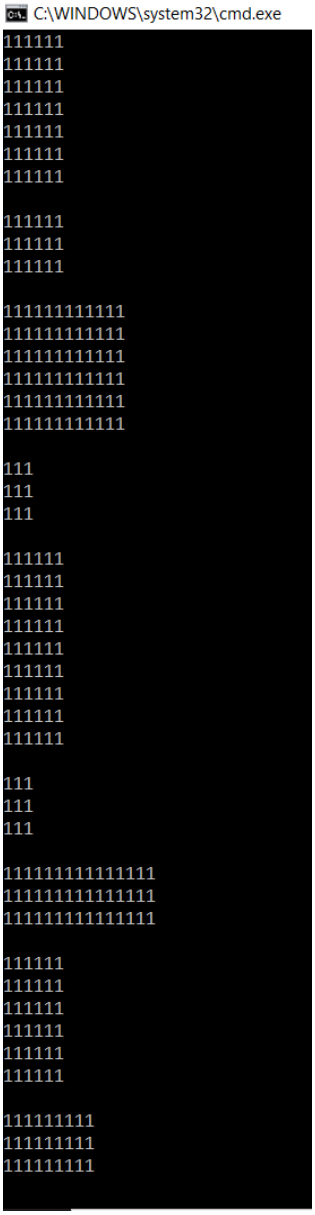
При запуске программы пользователь видит большую кнопку Выбрать Изображение.



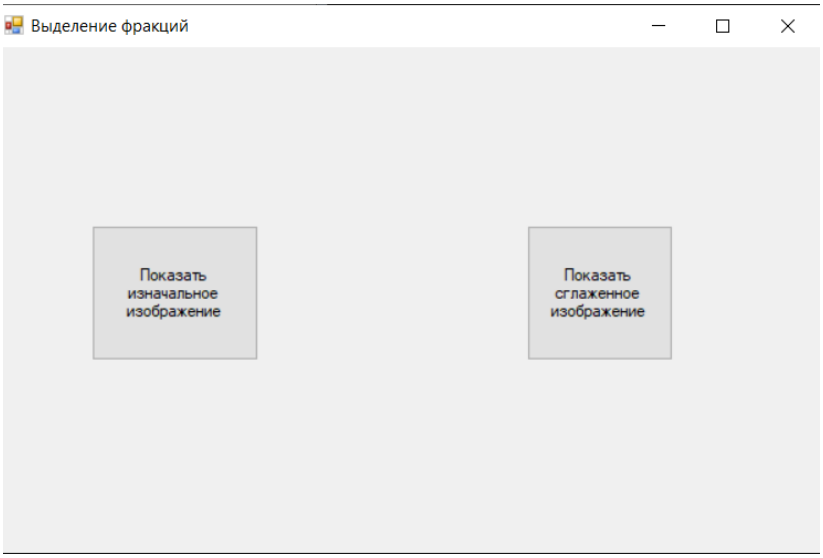
После нажатия этой кнопки появляется диалоговое окно для выбора нашего изображения.



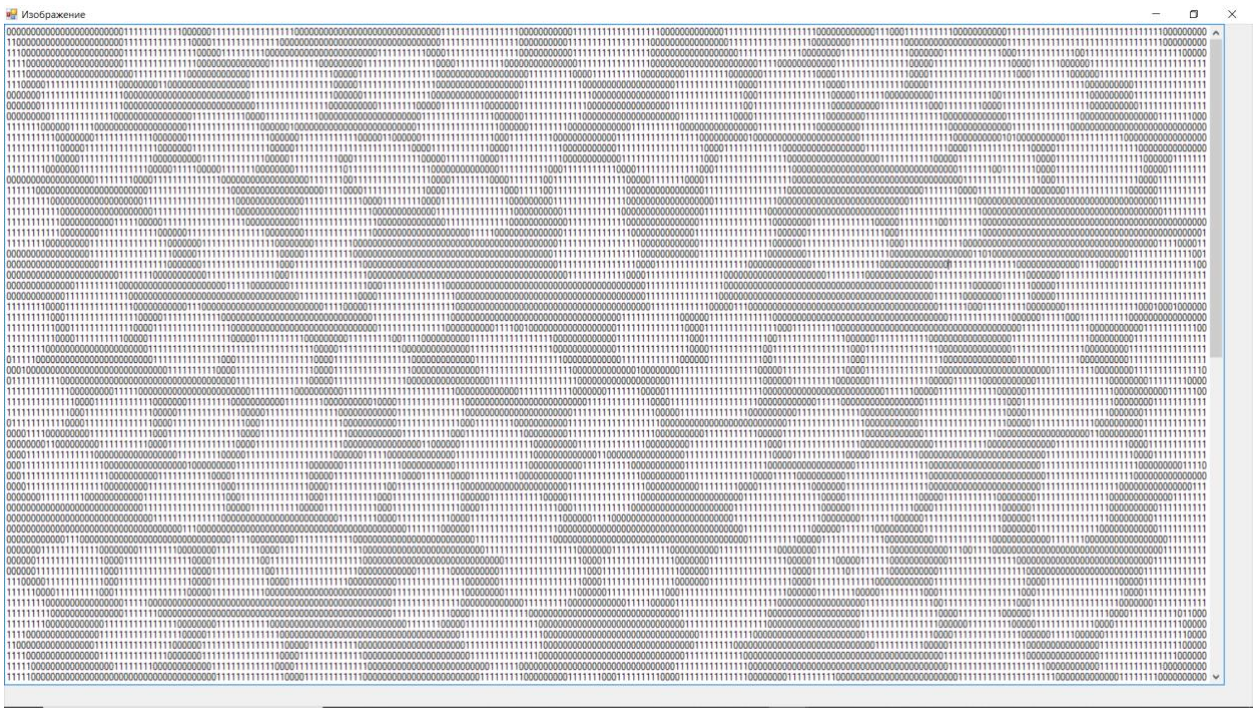
После выбора изображения в консоли появится готовая коллекция элементов, выделенных их этого изображения.



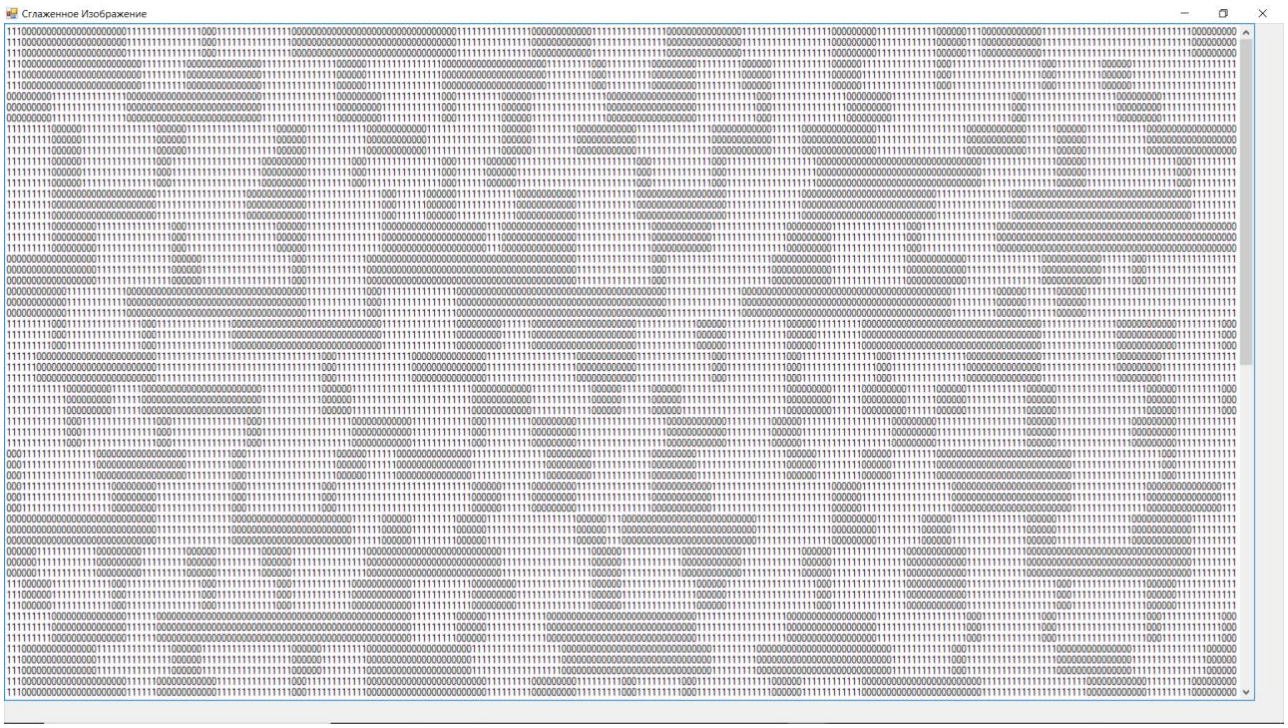
Также на форме появятся две кнопки: Показать изначальное изображение и Показать сглаженное изображение.



Нажав на левую кнопку, пользователь сможет увидеть, как изображение было переведено в двоичный код (работу первого пункта).



Выбрав правую кнопку, мы сможем увидеть работу второго пункта, то есть то, как мой алгоритм сгладил изображение.


Сглаженное Изображение

Выводы

- Была разработана программа для выделения нанофракций вещества по его цифровому снимку.
- Была изучена библиотека Magick.NET и улучшены навыки работы с изображениями в целом.
- Проведено сравнение работы алгоритмов машинного обучения и алгоритма, написанного без его помощи.

Алгоритмы с использованием нейросетей работают лучше и нагляднее, однако применимы к гораздо более узкому спектру входных данных. Кроме того, на их разработку и обучение уходит большее количество времени.

Перспективы дальнейшей разработки

Главная перспектива – работа с выделенными фракциями. Есть три основных варианта дальнейшей разработки:

- Анализ объектов
- Кластеризация объектов
- Работа с цветными изображениями

Анализ объектов подразумевает под собой изучение определённых свойств, таких как размер, форма, центр масс и другие.

Список литературы

1. Threshold heating temperature for magnetic hyperthermia: Controlling the heat exchange with the blocking temperature of magnetic nanoparticlesя / B. Pimentel [и др.] // Journal of Solid State Chemistry. – 2018. – С. 2-5
2. Zennolab [Электронный ресурс]: сообщество профессионалов автоматизации – Режим доступа: https://zennolab.com/discussion/threads/biblioteka-magick-net-c-image-magick-rabota-s-izobrazhenijami-i-drugie-poleznjashki.70162/ (Дата обращения – 20.03.2021)
3. Delphi Sources [Электронный ресурс]: международный интернет-форум для разработчиков – Режим доступа: https://delphisources.ru/forum/showthread.php?t=21949 (Дата обращения – 26.05.2021)
4. MSDN [Электронный ресурс]: официальные инструкции по разработке на C# от Microsoft – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/ (Дата обращения – 5.06.2021)

Приложения

Form1.cs (главная форма)

```
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 ImageMagick;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;

namespace Модель_выделения_нанофракций
{
    public partial class Form1 : Form
    {
        Bitmap image;
        Form2 PictureF;
        Form3 ResultF;
        public SaveFileDialog saveFileDialog;
        byte[,] A;
        byte[,] B;
        List<byte[,]> C = new List<byte[,]>();
        public Form1()
        {
            InitializeComponent();
        }

        private void ВыбратьИзображение_Click(object sender, EventArgs e)
        {
            OpenFileDialog openDialog = new OpenFileDialog();
            openDialog.Filter = "Файлы изображений|*.bmp;*.png;*.jpg";
            if (openDialog.ShowDialog() != DialogResult.OK)
                return;

            try
            {
                using (MagickImage image = new MagickImage(openDialog.FileName))
                {
                    image.Format = MagickFormat.Png;
                    image.ColorType = ColorType.Bilevel;
                    image.Write(openDialog.FileName);
                    pictureBox1.Image = Bitmap.FromFile(openDialog.FileName);

                }
            }
            catch (OutOfMemoryException except)
            {
                MessageBox.Show("Ошибка чтения картинки");
                return;
            }
            ВыбратьИзображение.Visible = false;
            pictureBox1.Visible = false;
            button1.Visible = true;
            button2.Visible = true;
            image = (Bitmap)pictureBox1.Image;
            A = BrightnessCheck(image);
            B = PictureSmooth(A);
            int ik = 0;
            int jk = 0;
            int i = 0;
            int j = 0;
            byte[,] transfer = new byte[1,1];
            TransferForAnalysis(B, i, j, ik, jk, transfer, C);
```

```
    foreach (byte[,] objects in C) ProverkaDeystvii(objects);
}

private byte[,] BrightnessCheck(Bitmap image)
{
    byte[,] A = new byte[image.Width, image.Height];
    byte R;
    byte G;
    byte B;
    for(int i=0; i<image.Width;i++)
    {
        for(int j=0; j<image.Height; j++)
        {
            R = image.GetPixel(i, j).R;
            G = image.GetPixel(i, j).G;
            B = image.GetPixel(i, j).B;
            if (Y(R, G, B) <= 85) A[i, j] = 1;
            else A[i, j] = 0;
        }
    }
    return A;
}

private byte Y(byte R, byte G, byte B)
{
    return (byte) (0.3 * R + 0.59 * G + 0.11 * B);
}

private byte[,] PictureSmooth (byte[,] A)
{
    byte[,] B = new byte[image.Width, image.Height];
    int s = 0;
    for (int i = 1; i < A.GetLength(0)-1; i+=3)
    {
        for (int j = 1; j < A.GetLength(1) - 1; j+=3)
        {
            s = 0;
            for(int n=i-1; n<=i+1;n++)
            {
                for (int m = j - 1; m <= j + 1; m++) if (A[n, m] == 1) s++;
            }
            if (s>=5)
            {
                for (int n = i - 1; n <= i + 1; n++)
                {
                    for (int m = j - 1; m <= j + 1; m++) B[n,m]=1;
                }
            }
            else
            {
                for (int n = i - 1; n <= i + 1; n++)
                {
                    for (int m = j - 1; m <= j + 1; m++) B[n, m] = 0;
                }
            }
        }
    }
    return B;
}

private void ProverkaDeystvii(byte[,]A)
{
    for (int i = 0; i < A.GetLength(0); i++, Console.WriteLine())
    {
        for (int j = 0; j < A.GetLength(1); j++)
        {
            Console.Write("{0}", A[i, j]);
        }
```

```
            }
            Console.WriteLine();
        }

        private void TransferForAnalysis(byte[,] B, int i, int j, int ik, int jk, byte[,]
transfer, List<byte[,]> C)
        {
            while (FullCheck(B))
            {
                C.Add(ObjectSearch(B, i, j, ik, jk, transfer));
            }

        }

        private byte[,] ObjectSearch(byte[,] B, int i, int j, int ik, int jk, byte[,]
transfer)
        {
            bool q = false;
            ik = 0;
            jk = 0;
            if(B[i,j]==0)
            {
                for (i=0; i < B.GetLength(0); i++)
                {
                    for (j=0; j < B.GetLength(1); j++)
                    {
                        if (B[i, j] == 1)
                        {
                            q = true;
                            break;
                        }
                    }
                    if (q) break;
                }
            }
            while(i+ik<B.GetLength(0))
            {
                if (B[i + ik, j] == 1) ik++;
                else break;
            }
            while(j + jk < B.GetLength(1))
            {
                if (B[i, j+jk] == 1) jk++;
                else break;
            }
            transfer = new byte[ik, jk];
            for (int n = 0; n < transfer.GetLength(0); n++)
            {
                for (int m = 0; m < transfer.GetLength(1); m++) transfer[n, m] = 1;
            }
            DeleteObject(B, i, j, ik, jk);
            return transfer;
        }

        private void DeleteObject(byte[,] B, int i, int j, int ik, int jk)
        {
            for(int n = i; n<i+ik; n++)
            {
                for (int m = j; m < j + jk; m++) B[n, m] = 0;
            }
        }

        private bool FullCheck(byte[,] B)
        {
            for (int i = 0; i < B.GetLength(0); i++)
            {
                for (int j = 0; j < B.GetLength(1); j++)
                {
```

```
                    if (B[i, j] == 1) return true;
                }
            }
            return false;
        }

        private void PictureRotate(byte[,] B)
        {
            byte[,] temp = new byte[B.GetLength(1), B.GetLength(0)];
            for (int i = 0; i < B.GetLength(0); i++)
            {
                for(int j=0; j<B.GetLength(1); j++)
                {
                    temp[j, i] = B[i, j];
                }
            }
            B = temp;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            var binaryFormatter = new BinaryFormatter();
            string str = "Фракции" + ".txt";
            using (FileStream fs = new FileStream(str, FileMode.OpenOrCreate))
            {
                binaryFormatter.Serialize(fs, A);

            }
            PictureF = new Form2();
            PictureF.Show();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            var binaryFormatter = new BinaryFormatter();
            string str = "Фракции" + ".txt";
            using (FileStream fs = new FileStream(str, FileMode.OpenOrCreate))
            {
                binaryFormatter.Serialize(fs, PictureSmooth(A));
            }
            ResultF = new Form3();
            ResultF.Show();
        }
    }
}
```

Form2.cs (двоичное изображение)

```
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 ImageMagick;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;

namespace Модель_выделения_нанофракций
{
    public partial class Form2 : Form
    {
        byte[,] A;
        public OpenFileDialog openFileDialog;
        string s;
```

```
        public Form2()
        {
            InitializeComponent();
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            var binaryFormatter = new BinaryFormatter();
            string Open = "";
            Open = "Фракции" + ".txt";
            using (FileStream fs = new FileStream(Open, FileMode.OpenOrCreate))
            {
                A = (byte[,])binaryFormatter.Deserialize(fs);
            }
            for (int i = 0; i < A.GetLength(0); i++)
            {
                for (int j = 0; j < A.GetLength(1); j++)
                {
                    s += A[i, j];
                }
                s += "\n";
            }
            this.textBox1.Text = s;
        }
    }
}
```

Form3.cs (сглаженное изображение)

```
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 ImageMagick;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;

namespace Модель_выделения_нанофракций
{
    public partial class Form3 : Form
    {
        public OpenFileDialog openFileDialog;
        string s;
        public Form3()
        {
            InitializeComponent();
        }

        private void Form3_Load(object sender, EventArgs e)
        {
            int count = 0;
            var binaryFormatter = new BinaryFormatter();
            string Open = "";
            Open = "Фракции" + ".txt";
            byte[,] A;
            using (FileStream fs = new FileStream(Open, FileMode.OpenOrCreate))
            {
                A = (byte[,])binaryFormatter.Deserialize(fs);
            }
            for (int i = 0; i < A.GetLength(0); i++)
            {
                for (int j = 0; j < A.GetLength(1); j++)
                {
```

```
                    s += A[i, j];
                }
                s += "\n";
            }
            this.textBox1.Text = s;
        }
    }
}
```