МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**
Кафедра **«Прикладная математика и информатика»**

02.03.03 Математическое обеспечение и администрирование информационных систем

ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ

БАКАЛАВРСКАЯ РАБОТА

на тему Разработка алгоритмов кластеризации флюорографических данных

Студент А.Р. Гилимханов _____

Руководитель В.С. Климов _____

Допустить к защите
Заведующий кафедрой, к.тех.н, доцент, А.В. Очеповский _____

«_____»_____2016 г.

Тольятти 2016

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

УТВЕРЖДАЮ
Зав.кафедрой «Прикладная
математика и информатика»
_____А.В.Очеповский

«____»_____2016 г.

ЗАДАНИЕ
на выполнение бакалаврской работы

Студент Гилимханов Артур Рустамович

1. Тема Разработка алгоритмов кластеризации флюорографических данных
2. Срок сдачи студентом законченной выпускной квалификационной работы 25 мая 2016 г.
3. Исходные данные к выпускной квалификационной работе: учебная литература, периодические издания, интернет-ресурсы
4. Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов, разделов): анализ состояния вопроса; разработка алгоритма кластеризации изображений; программная реализация предложенных решений; апробация предложенных решений на реальных данных; выводы по работе.
5. Ориентировочный перечень графического и иллюстративного материала Набор формул, объясняющих математический аппарат; демонстрация работы алгоритма на реальном наборе данных; графики и диаграммы, поясняющие результат работы системы.
6. Дата выдачи задания « 11 » января 2016 г.

Руководитель выпускной
квалификационной работы _____ В.С. Климов

Задание принял к исполнению _____ А.Р. Гилимханов

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

УТВЕРЖДАЮ
Зав.кафедрой «Прикладная
математика и информатика»
_____А.В.Очеповский

«____»_____2016 г.

КАЛЕНДАРНЫЙ ПЛАН
выполнения бакалаврской работы

Студента Гилимханова Артура Рустамовича

по теме Разработка алгоритмов кластеризации флюорографических данных

Наименование раздела работы	Плановый срок выполнения раздела	Фактический срок выполнения раздела	Отметка о выполнении	Подпись руководителя
1. Анализ состояния вопроса	1.04.2016	1.04.2016	Выполнено	
2. Разработка алгоритма подбора товаров	11.04.2016	11.04.2016	Выполнено	
3. Программная реализация предложенных решений	25.04.2016	25.04.2016	Выполнено	
4. Апробация предложенных решений на реальных данных	9.05.2016	9.05.2016	Выполнено	
Оформление пояснительной записки	07.04.2016	07.04.2016	Выполнено	
Подготовка доклада и графического материала к защите	14.04.2016	14.04.2016	Выполнено	
Предварительная защита бакалаврской	20.05.2016-20.06.2016	20.05.2016-20.06.2016	Выполнено	

работы				
Проверка ВКР в системе «Антиплагиат.ВУЗ»	27.05.2016	27.05.2016	Выполнено	
Сдача пояснительной записки ВКР	20.06.2016	20.06.2016	Выполнено	

Руководитель выпускной квалификационной работы _____ В.С. Климов

Задание принял к исполнению _____ А.Р. Гилимханов

Аннотация

Тема данной выпускной квалификационной работы: Разработка алгоритмов кластеризации флюорографических данных.

Работа направлена на решение актуальной проблемы – извлечение выводов на основе анализа графической информации, содержащейся в флюорографических изображениях.

Целью данной выпускной квалификационной работы является снижение нагрузки на медицинский персонал посредством программного обеспечения и автоматизации обработки данных флюорографического исследования.

Для достижения цели данной выпускной квалификационной работы были поставлены и решены следующие задачи:

– был произведен анализ флюорографических изображений (первая глава);

– была спроектирован и разработан алгоритм кластеризации флюорографических данных (вторая глава);

– была разработана программа с настройками выбора изображений, затем было произведено внедрение в него разработанного алгоритма кластеризации изображений (третья глава).

При разработке алгоритма кластеризации флюорографических изображений были использованы такие инструменты разработки как объектно-ориентированный язык программирования C#, интерфейс программирования приложений Windows Forms, одна из оптимальных сред разработки для программистов Microsoft Visual Studio, кластеризационная метрика Евклида.

Работа содержит в себе введение, в котором дается краткая характеристика вопроса, анализ вопроса и предложенное решение. 2 глава, в которой был спроектирован и разработан данный алгоритм, и в 3 главе было разработано программное обеспечение, в котором реализован алгоритм

флюорографических изображений, группировка снимков и сохранение итогов в «txt» файл и отдельные папки. Выпускная квалификационная работа содержит пояснительную записку объемом 41 страниц, включая 40 рисунков, список используемой литературы, состоящий из 20 источников.

Оглавление

Введение

Флюорография - это рентгенологическое исследование, при котором рентгеновское изображение объекта фотографируется с флюоресцирующего экрана на фотоплёнку или на электронный носитель.

Стандартной процедурой является флюорография грудной клетки. Результаты исследования используются для диагностики заболеваний легких, сердца, грудных желез. С помощью флюорографии могут быть обнаружены опухоли, участки воспаления полости, представляющие собой патологические образования, каверны, клероз, фиброз и инородные предметы. Флюорография легких позволяет выявить злокачественную опухоль или туберкулёз ещё на ранней стадии, когда симптомы заболевания не проявляются и диагностика болезни затруднительна.

В нашей стране флюорографию проходят все занятое население РФ, поэтому для снижения нагрузки на медицинский персонал необходимо обеспечение автоматизации обработки данных флюорографического исследования.

Программа будет обрабатывать большое количество фотографий с помощью искусственного интеллекта, и затем группировать эти фото по группам заболеваний. Это значительно ускорит процесс проверки изображений на предмет выявления болезней у людей.

1 Анализ состояния вопроса

1.1 Выбор программного обеспечения для реализации

Microsoft Visual Studio – это одна из оптимальных сред разработки для программистов различных уровней подготовки. Microsoft Visual Studio – это передовое решение для разработки, позволяющее командам любого размера проектировать и создавать привлекательные ——приложения, которые удовлетворят самым взыскательным требованиям. В ней можно использовать расширенные средства моделирования, обнаружения и проектирования архитектуры, чтобы описать свою систему и обеспечить полную реализацию концепции архитектуры.

Microsoft Visual Studio построена на архитектуре, поддерживающей возможность использования встраиваемых дополнений — плагинов от сторонних разработчиков, что позволяет расширять возможности среды разработки (рисунок 1.1).

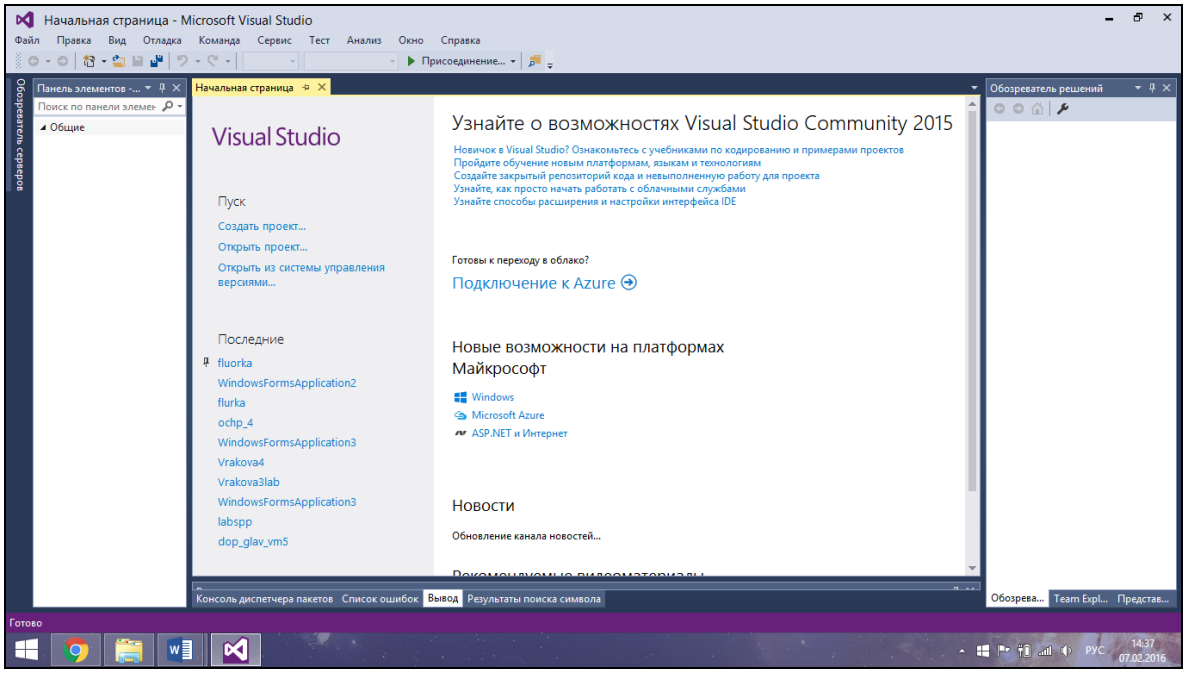


Рисунок 1.1 – Интерфейс Microsoft Visual Studio

Для реализации программного обеспечения обработки флюорографических изображений я выбрал объектно-ориентированный язык программирования C#. Он относится к языкам C-подобным синтаксисом. Данный язык программирования в системе Windows широко используется

для написания программ. C# очень прост в использовании для написания программ в Windows Forms.

Windows Forms — интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя [1] (рисунок 1.2).

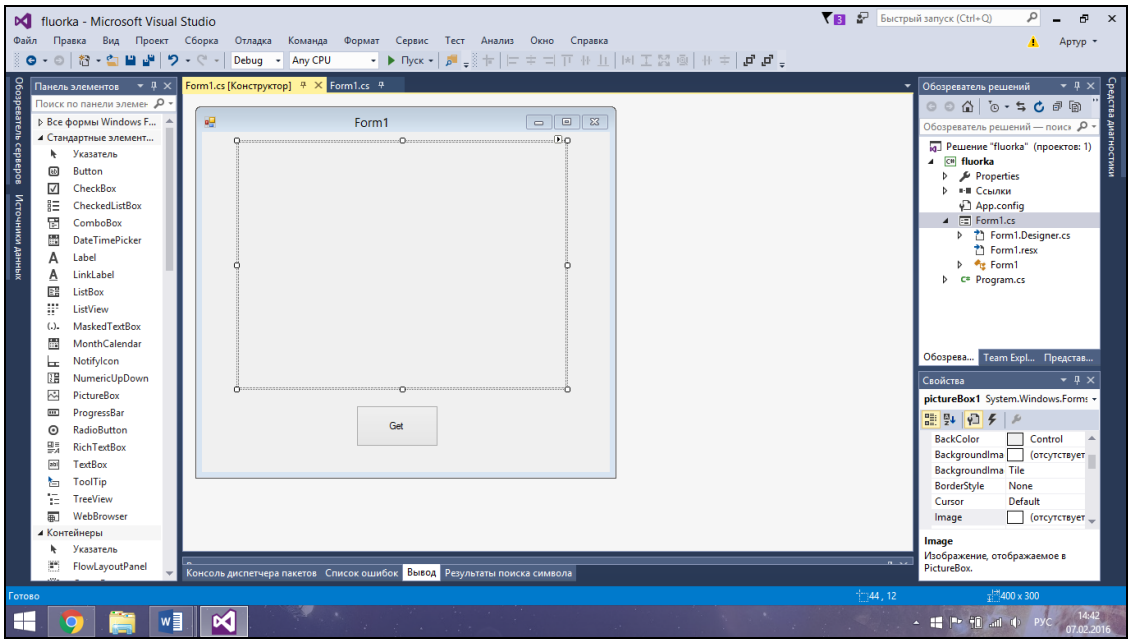


Рисунок 1.2 – интерфейс Windows Forms.

Так же в Windows Forms есть панель элементов, с которой с легкостью можно сделать приложение, потратив минимальное время на выполнение (рисунок 1.3).

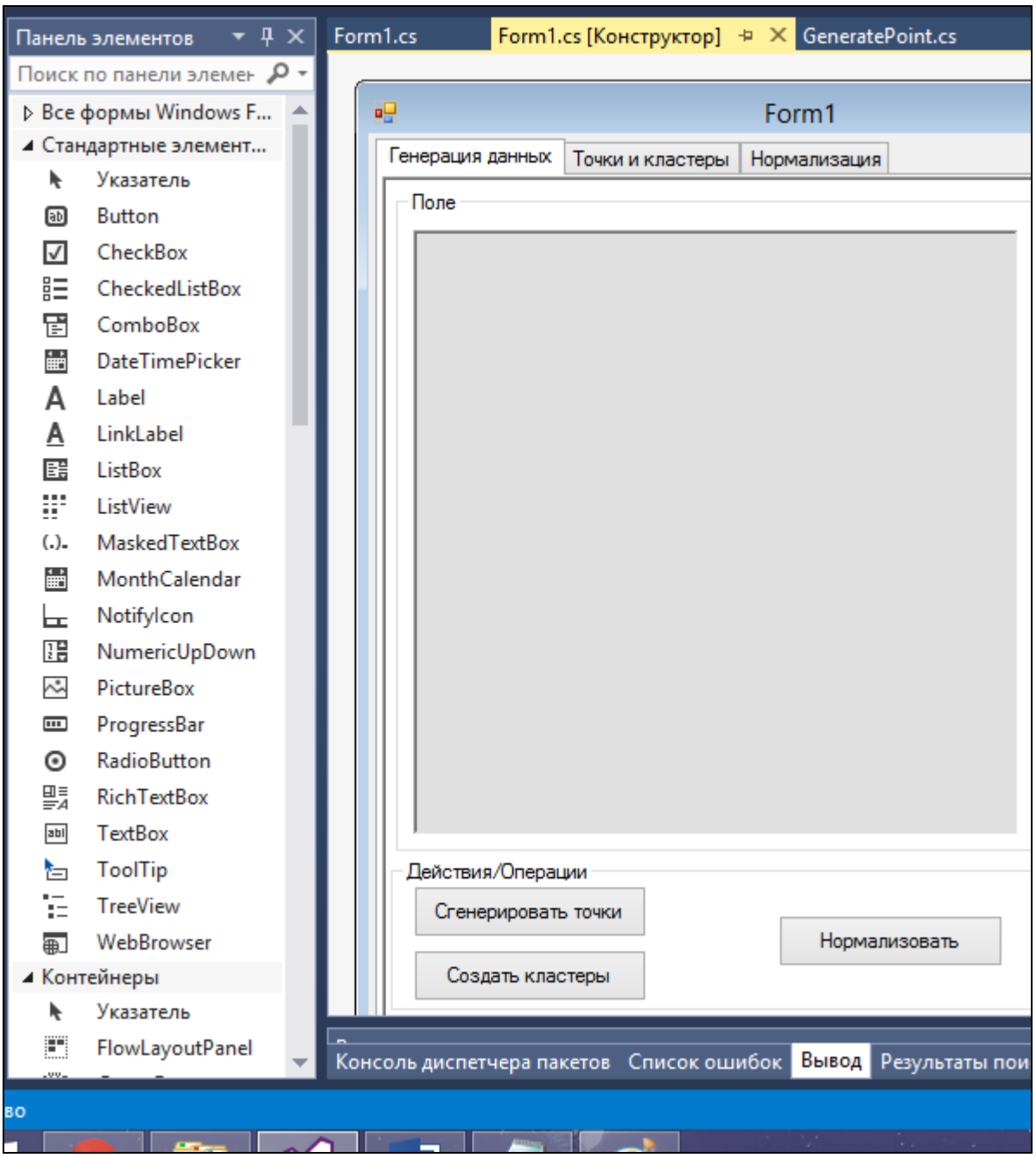


Рисунок 1.3 – Панель элементов в Windows Forms

1.2 Теоретическое исследование сегментации изображения

Сегментация – это процесс разбиения на группы. Наиболее часто используемыми являются два вида сегментации - сегментация по цветовым координатам для цветных изображений и сегментация по яркости для бинарных изображений. Алгоритмы сегментации характеризуются некоторыми параметрами надежности и достоверности обработки. Они зависят от того, насколько полно учитываются дополнительные

характеристики распределения яркости в областях объектов или фона, количество перепадов яркости, форма объектов и др.

Существует много изображений, которые содержат исследуемый объект достаточно однородной яркости на фоне другой яркости. В качестве примера можно привести рукописный текст, ряд медицинских изображений и т.д. Если яркости точек объекта резко отличаются от яркостей точек фона, то решение задачи установления порога является несложной задачей. На практике это не так просто, поскольку исследуемое изображение подвергается воздействию шума и на нем допускается некоторый разброс значений яркости. Известно несколько аналитических подходов к пороговому ограничению по яркости. Один из методов состоит в установлении порога на таком уровне, при котором общая сумма элементов с подпороговой яркостью согласована с априорными вероятностями этих значений яркости.

Аналогичные подходы можно применить для обработки цветных и спектрозональных изображений. Существует также такой вид сегментации как контурная сегментация. Довольно часто анализ изображений включает такие операции, как получение внешнего контура изображений объектов и запись координат точек этого контура. Известно три основных подхода к представлению границ объекта: аппроксимация кривых, прослеживание контуров и связывание точек перепадов. Для полноты анализа следует отметит, что есть также текстурная сегментация и сегментация формы [2].

Рассмотрим один из подходов решения задачи сегментации изображений - метод водораздела. В данном методе берется изображение как карта местности, где яркость представляется как значение высот относительно уровней. Заполняя эту местность водой, мы получим бассейны, где места объединения бассейнов на данной карте выделяются как линии водораздела.

Метод маркерного водораздела – это разделение предметов на изображении, которые соприкасаются друг с другом. Он является

неотъемлемой частью обработки изображений, так же является одним из наиболее эффективных методов сегментации изображений. Для реализации данного метода обработки изображений выполняются определенные шаги, такие как:

1) Шаг 1: Считывание цветного изображения и преобразование его в полутоновое (рисунок 1.4);

Рисунок 1.4 – Полученное изображение после 1 шага

2) Шаг 2: Использование значения градиента в качестве функции сегментации (рисунок 1.5);

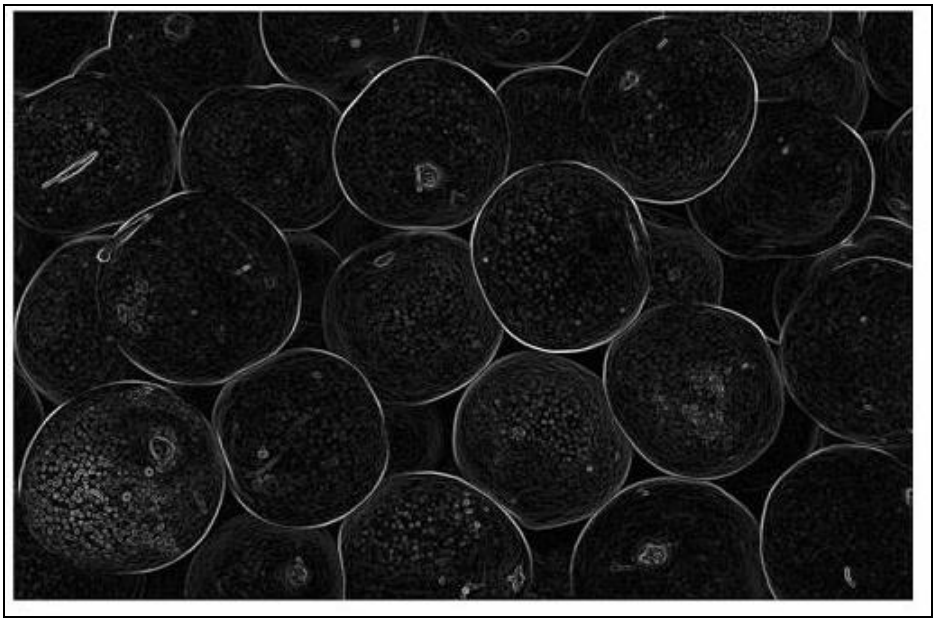


Рисунок 1.5 – Полученное изображение после 2 шага

3) Шаг 3: Маркировка объектов переднего плана (рисунок 1.6);

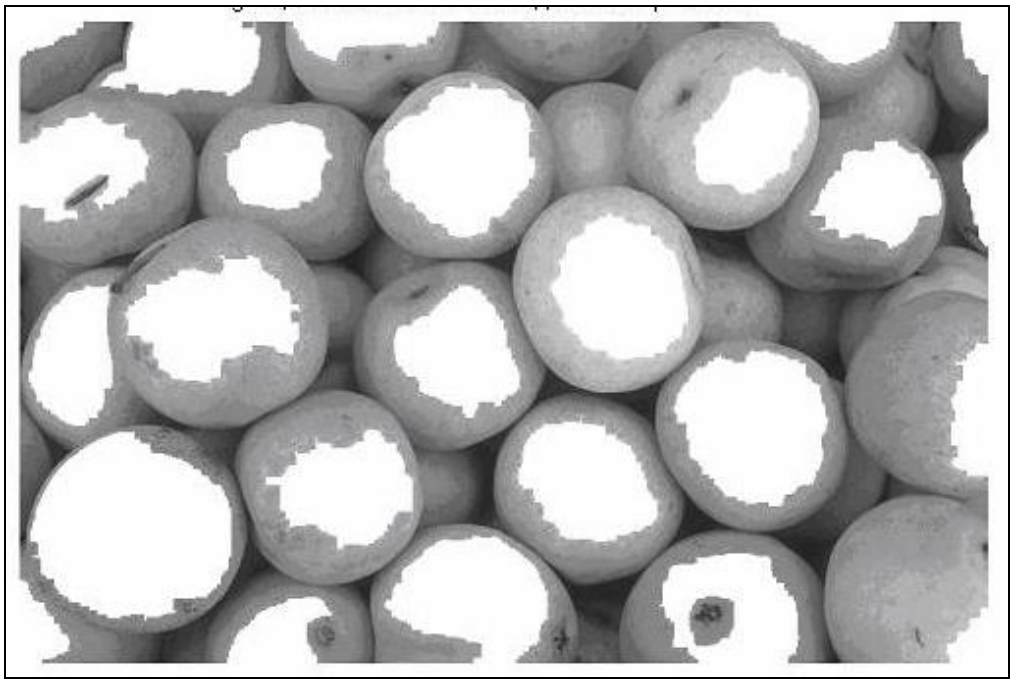


Рисунок 1.6 – Полученное изображение после 3 шага

4) Шаг 4: Вычисление маркеров фона (рисунок 1.7);

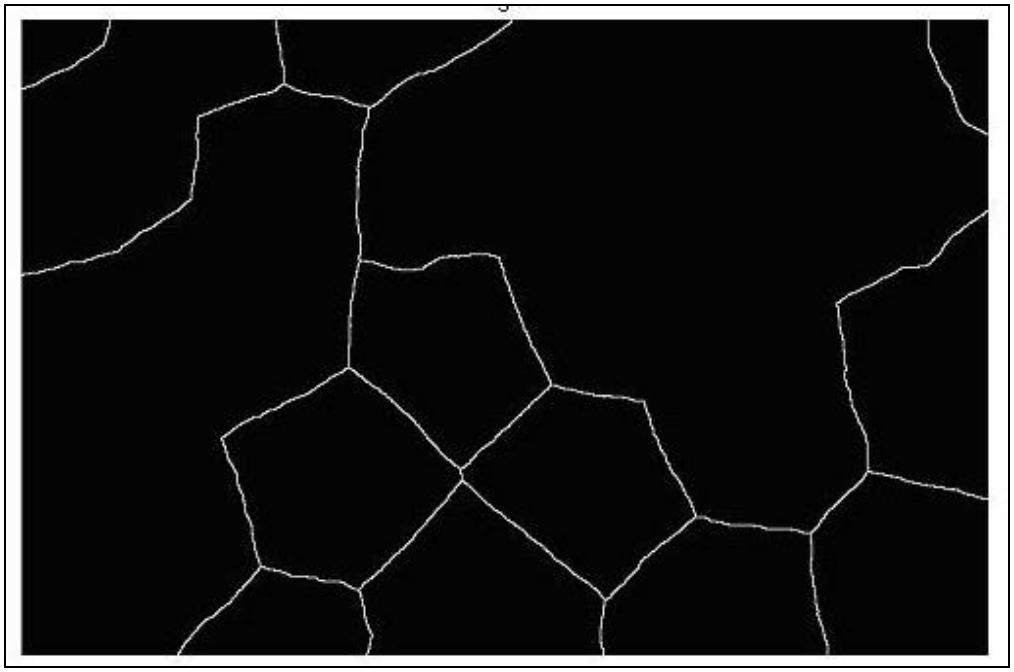


Рисунок 1.7 – Полученное изображение после 4 шага

5) Шаг 5: Вычисление по методу маркерного водораздела на основании модифицированной функции сегментации (рисунок 1.8);

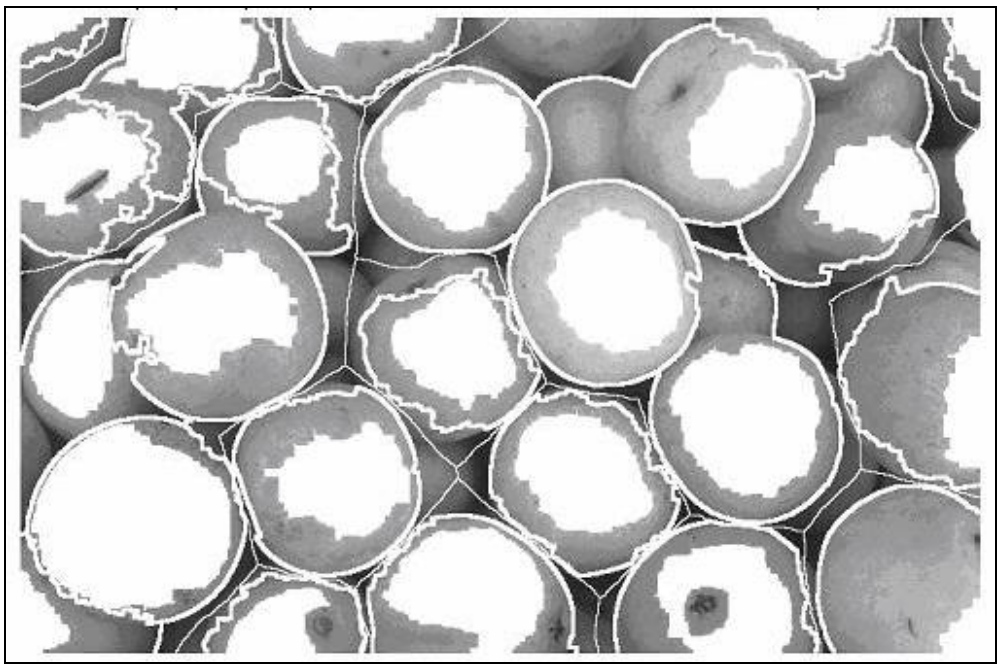


Рисунок 1.8 – Полученное изображение после 5 шага

6) Шаг 6: Визуализация результата обработки (рисунок 1.9).

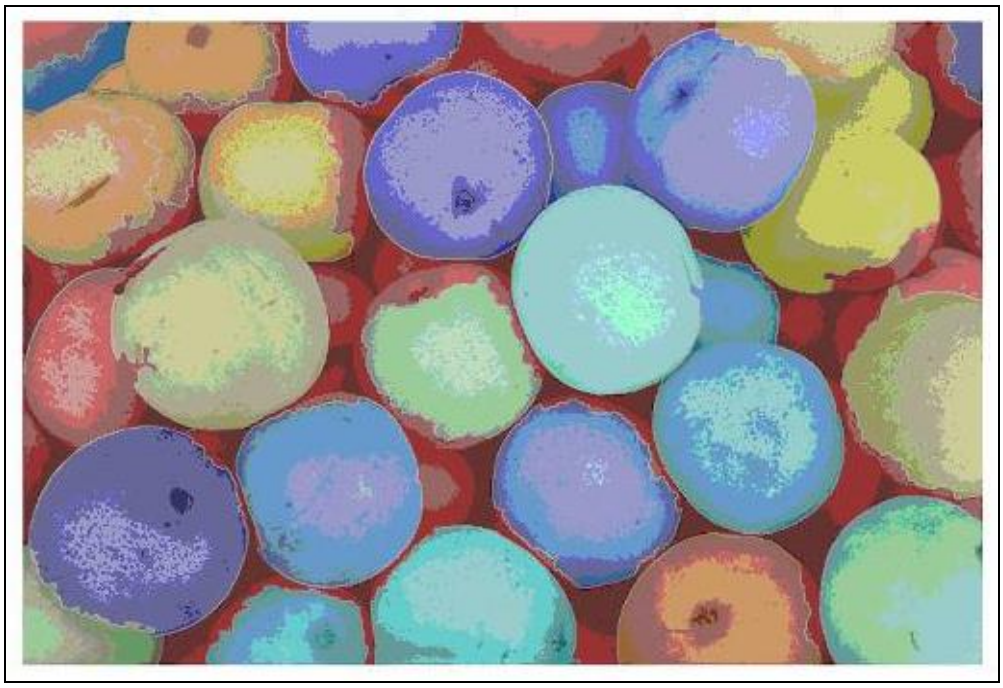


Рисунок 1.9 – Полученное изображение после 6 шага

Таким образом мы получили изображение, в котором мы провели сегментацию, и разделили объекты друг от друга.

1.3 Теоретическое исследование алгоритма K-means

Кластеризация – объединение в группы схожих объектов – является одной из фундаментальных задач в области анализа данных. Список прикладных областей, где она применяется, широк: сегментация

изображений, маркетинг, борьба с мошенничеством, прогнозирование, анализ текстов и многие другие. На современном этапе кластеризация часто выступает первым шагом при анализе данных. После выделения схожих групп применяются другие методы, для каждой группы строится отдельная модель.

Задачу кластеризации в том или ином виде формулировали в таких научных направлениях, как статистика, распознавание образов, оптимизация, машинное обучение. Отсюда многообразие синонимов понятию кластер – класс, таксон, сгущение.

На сегодняшний момент число методов разбиения групп объектов на кластеры довольно велико – несколько десятков алгоритмов и еще больше их модификаций.

Основная идея заключается в том, что на каждой итерации пере-вычисляется центр масс для каждого кластера, полученного на предыдущем шаге, затем векторы разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике.

Алгоритм завершается, когда на какой-то итерации не происходит изменения центра масс кластеров. Это происходит за конечное число итераций, так как количество возможных разбиений конечного множества конечно, а на каждом шаге суммарное квадратичное отклонение V не увеличивается, поэтому зацикливание невозможно.

1.4 Способы реализации алгоритмов K-means

Действие алгоритма заключается в том, что он стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров (рисунок 1.10):

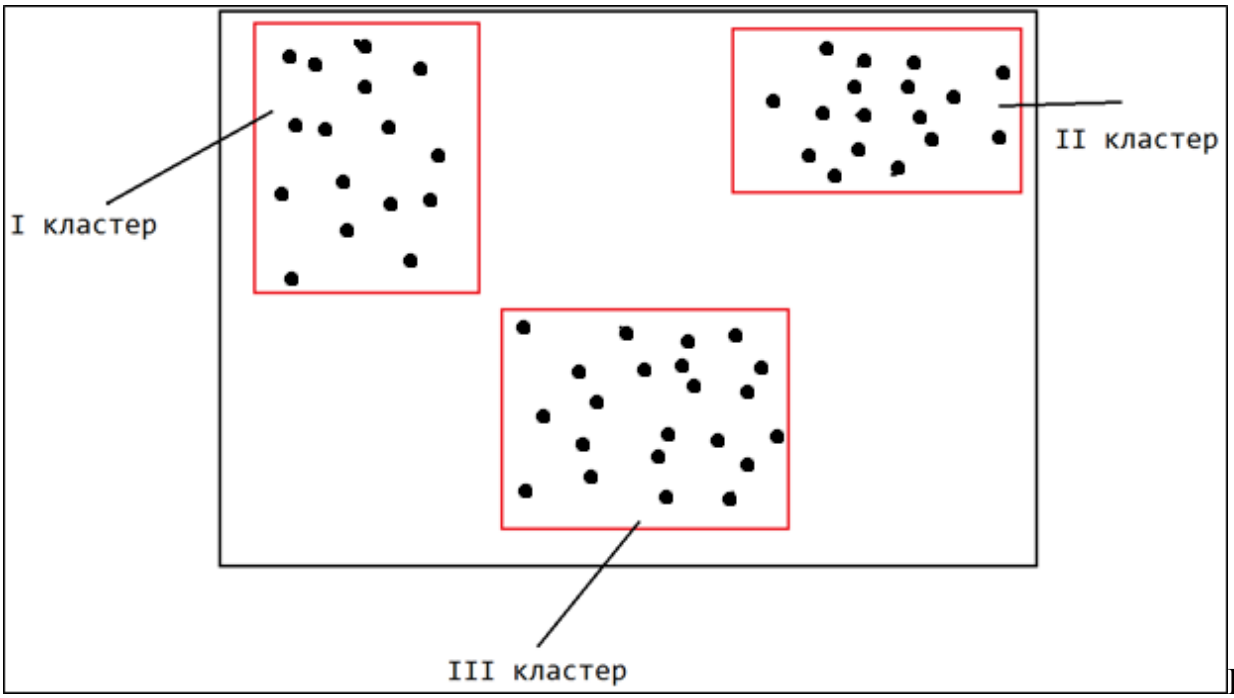


Рисунок 1.10 – Алгоритм кластеризации K-Means

Технология метода кластеризации K-means может быть реализована множеством путей. Основа у них одна – на вход подаются данные и количество кластеров (групп) [3].

Основное отличие у технологий кластеризации - метрика, на основе которой будет происходить «группировка» входных данных. Рассмотрим некоторые из них:

Метрика Евклида - эта метрика является наиболее используемой и отражает среднее различие между объектами (1) [4].

$$d_{ij} = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2} \quad (1)$$

Метрика нормированного Евклида. Нормализованные евклидовы расстояния более подходят для переменных, измеряемых в различных единицах или значительно различающихся по величине [5].

Если дисперсии по характеристикам отличаются друг от друга, то (2):

$$d_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(x_{ik} - x_{jk})^2}{S_k^2}}$$

(2)

Если масштаб данных различен, например, одна переменная измерена в стэнах, а другая в баллах, то для обеспечения одинакового влияния всех характеристик на близость объектов используется следующая формула подсчета расстояния (3) [6]:

$$d_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(x_{ik} - x_{jk})^2}{S_{k\,\max}^2}}$$

(3)

City-block - Метрика city-block (манхэттенская метрика, получившая свое название в честь района Манхэттен, который образуют улицы, расположенные в виде пересечения параллельных прямых под прямым углом; как правило, применяется для номинальных или качественных переменных) (4) [7]:

$$d_{ij} = \sqrt{\sum_{k=1}^{n} |x_{ik} - x_{jk}|}$$

(4)

1.5 Выбор метода кластеризации

Для разработки моей программы и выполнения кластеризации я выбрал метрику Евклида. Чаще всего применяют расстояние Евклида, которое позволяет формировать центры кластеров [8].

2 Описание требований программы

2.1 Требования к составу выполняемых функций

Программа должна обеспечивать возможность выполнения следующих функций:

- Обработка флюорографических изображений;
- Группировка снимков по типу заболеваний;
- Сохранение результатов работы программы в формате txt.

2.2 Эксплуатационное назначение программы

Программа может эксплуатироваться в государственных учреждениях здравоохранения. Программа может использоваться медицинским персоналом в качестве инструмента поддержки принятия решения.

2.3 Требования к обеспечению надежного функционирования программы

Надежное (устойчивое) функционирование программы должно быть обеспечено выполнением организационно-технических мероприятий, перечень которых приведен ниже:

- организацией бесперебойного питания технических средств;
- выполнением требований ГОСТ Р 51583-2000. Защита информации. Порядок создания автоматизированных систем в защищенном исполнении.

2.4 Отказы из-за некорректных действий оператора

Надежность системы не должна зависеть от данных, вводимым пользователем.

2.5 Климатические условия эксплуатации

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации – ГОСТ 15150-69.

2.6 Требования к защите информации и программ

В Системе должен быть обеспечен надлежащий уровень защиты информации в соответствии с законом о защите персональной информации и программного комплекса в целом от несанкционированного доступа - “Об информации, информатизации и о защите информации” РФ N 149-ФЗ от 27.07.2006.

3 Реализация программного обеспечения

Разработка алгоритма флюорографических данных построена на 4 основных этапах:

1) Нормализация изображений.

2) Кластеризация изображений.

3) Изучение свойств каждого кластера.

4) Подача ранее неизвестного изображения.

Далее подробно расписаны каждые этапы.

3.1 Нормализация

Под нормализацией изображения понимается процесс преобразования изображений к единообразному виду. Первое что нужно сделать – выровнять изображение относительно позвоночника человека, для того что бы все изображения были схожи (рисунок 3.1).

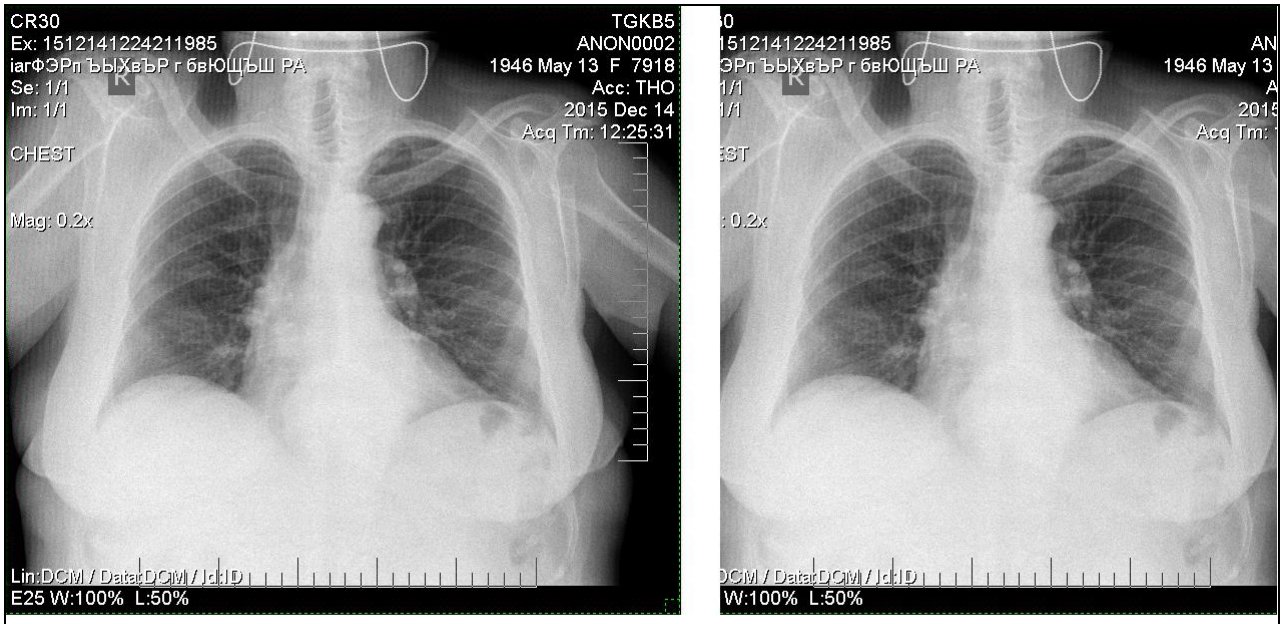


Рисунок 3.1 – Вырывание изображения по оси позвоночника человека

Следующим шагом нормализации является приведение изображений к одной цветовой гамме. В моем программном обеспечении реализуется это при помощи класса Bitmap, метода GetPixel и математической формулы.

Программа будет извлекать фотографии из выбранной папки, и при помощи класса Bitmap обрабатывать изображение. Bitmap инкапсулирует точечный рисунок, состоящий из данных пикселей графического изображения и атрибутов рисунка [2]. Объект Bitmap используется для работы с изображениями, определяемыми данными пикселей (рисунок 3.2).

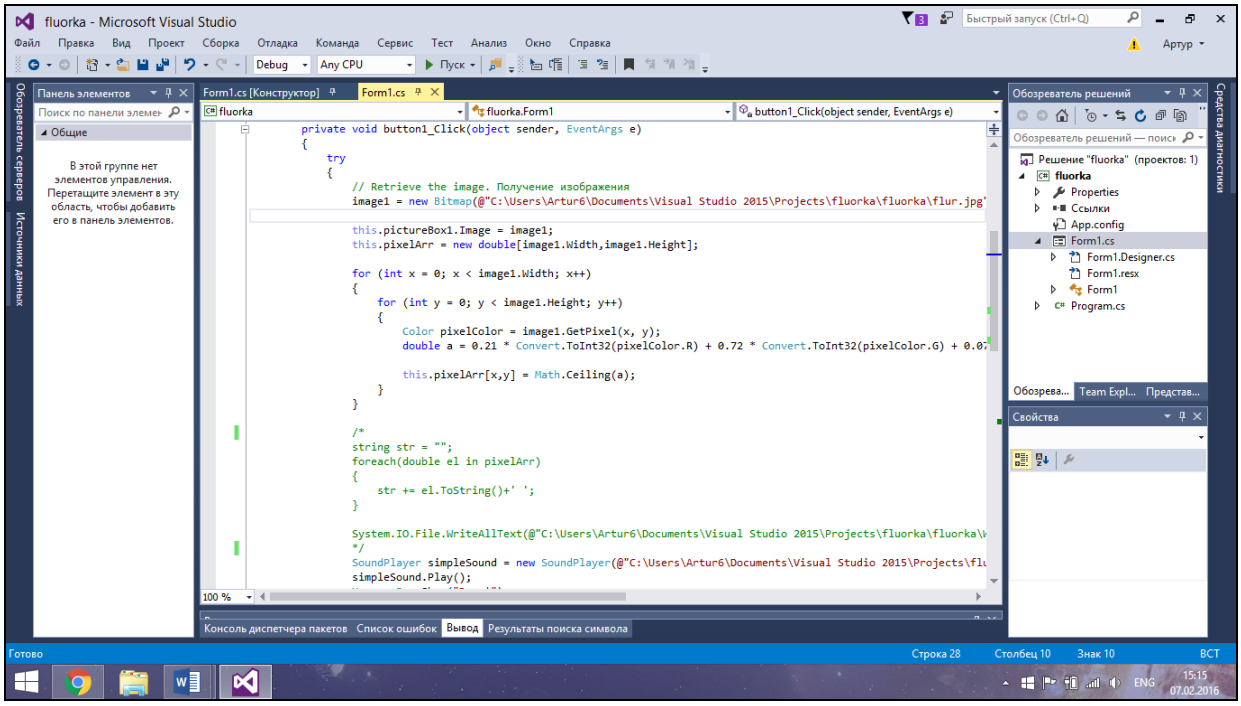


Рисунок 3.2 – Класс Bitmap для обработки изображения

В своем программном обеспечении я использовал метод GetPixel. Суть этого метода заключается в том, что этот метод получает цвет указанного пикселя в этом изображении и его координаты x и y. Работу данного метода я привел на рисунках ниже (рисунок 3.3).

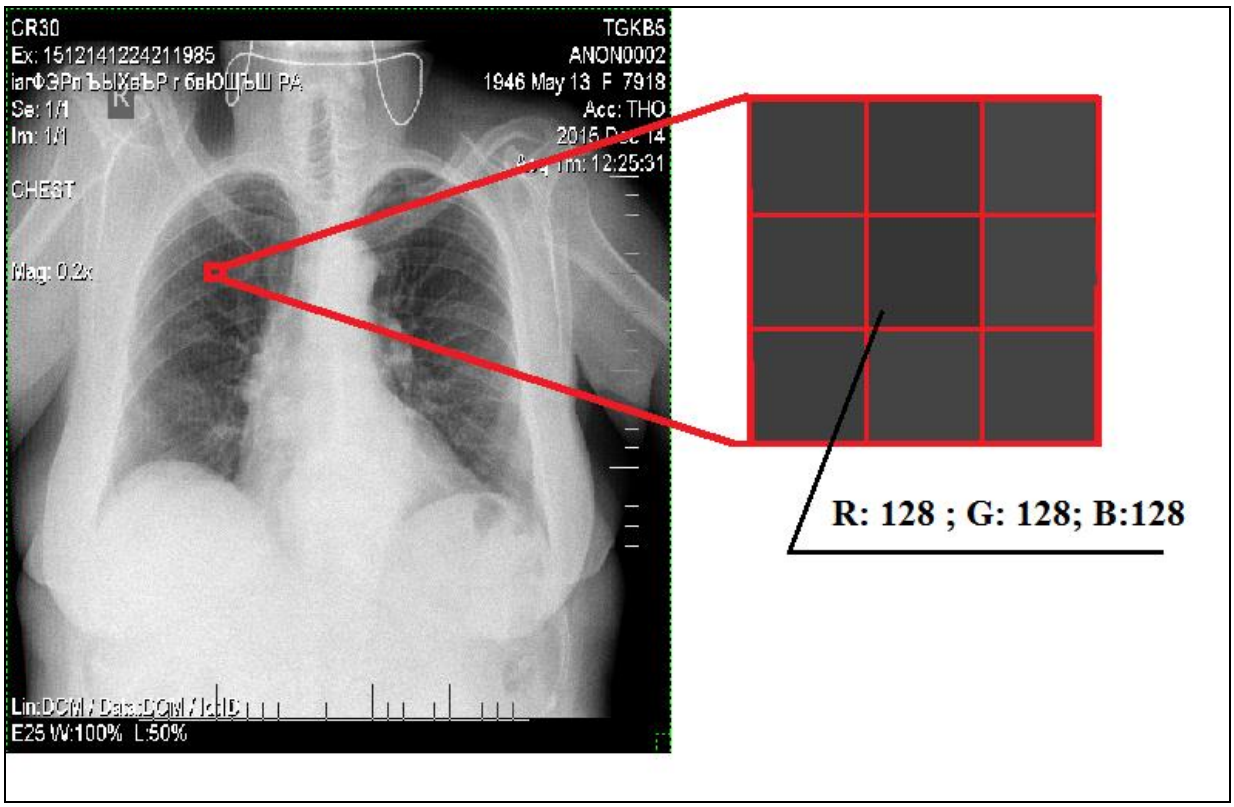


Рисунок 3.3 – Получение информации о пикселе при помощи класса GetPixel

Следующим шагом идет преобразование цвета изображения к оттенкам серого цвета. Это требуется по причине того, что некоторые изображения флюорографии имеют различные цветовые оттенки. Делается это при помощи общей формулы преобразования изображения. Тем самым мы получаем изображение, приведенное к шкале оттенкам серого при этом пользуясь следующим программным кодом (рисунок 3.4, 3.5).

```
double a = 0.21 * Convert.ToInt32(pixelColor.R) + 0.72 *
Convert.ToInt32(pixelColor.G) + 0.07 * Convert.ToInt32(pixelColor.B);
```

То есть,

```
ЗНАЧЕНИЯ ЦВЕТА = 0,21*R + 0.72*G + 0.07*B
```

Рисунок 3.4 – Преобразование изображения и получение значения цвета

```
for(int i = 0; i < 250; i++)
                {
                    for(int j = 0; j < 250; j++)
                    {
                        Color pixel = cl1.GetPixel(i, j);
                        cluster1[i, j] = Math.Ceiling(ToGray(pixel));
                    }
                }

                for (int i = 0; i < 250; i++)
                {
                    for (int j = 0; j < 250; j++)
                    {
                        Color pixel = cl2.GetPixel(i, j);
                        cluster2[i, j] = Math.Ceiling(ToGray(pixel));
                    }
                }

                for (int i = 0; i < 250; i++)
                {
                    for (int j = 0; j < 250; j++)
                    {
                        Color pixel = cl3.GetPixel(i, j);
                        cluster3[i, j] = Math.Ceiling(ToGray(pixel));
                    }
                }

                GC.Collect();
            }
        }

        private double ToGray(Color pixel)
        {
            return 0.21 * Convert.ToInt32(pixel.R) + 0.72 * Convert.ToInt32(pixel.G) + 0.07 * Convert.ToInt32(pixel.B);
        }
```

Рисунок 3.5 – Код преобразования изображения и получение значения цвета

Пройдя все эти шаги, мы получаем каждое изображение в виде массива упорядоченных чисел, каждое из которых является значением цвета каждого пикселя. В моем программе использовались изображения 250x250 пикселей, а значит размер полученного массива - 62500 значений (рисунок 3.6).

```
Bitmap cl1 = ResizeImage(new Bitmap(@items[0].Path, true), 250, 250);
Bitmap cl2 = ResizeImage(new Bitmap(@items[1].Path, true), 250, 250);
Bitmap cl3 = ResizeImage(new Bitmap(@items[2].Path, true), 250, 250);
private Bitmap ResizeImage(Image image, int width, int height)
    {
        var destRect = new Rectangle(0, 0, width, height);
        var destImage = new Bitmap(width, height);
```

```
            destImage.SetResolution(image.HorizontalResolution,
image.VerticalResolution);

            using (var graphics = Graphics.FromImage(destImage))
            {
               graphics.CompositingMode = CompositingMode.SourceCopy;
               graphics.CompositingQuality = CompositingQuality.HighQuality;
               graphics.InterpolationMode                                    =
InterpolationMode.HighQualityBicubic;
               graphics.SmoothingMode = SmoothingMode.HighQuality;
               graphics.PixelOffsetMode = PixelOffsetMode.HighQuality;

               using (var wrapMode = new ImageAttributes())
               {
                  wrapMode.SetWrapMode(WrapMode.TileFlipXY);
                  graphics.DrawImage(image, destRect, 0, 0, image.Width,
image.Height, GraphicsUnit.Pixel, wrapMode);
               } }
            return destImage;       }
```

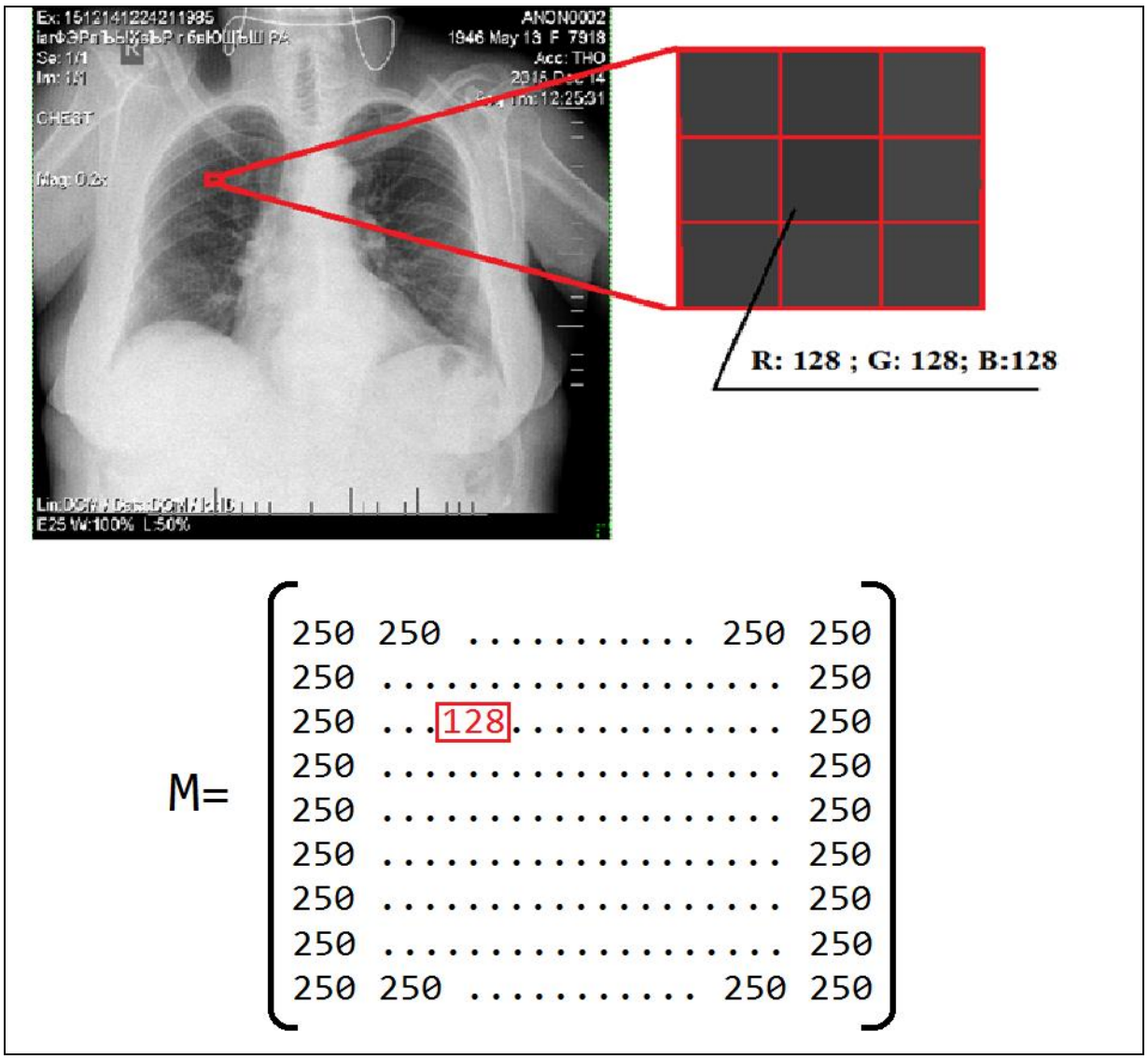


Рисунок 3.6 – Полученный массив чисел

Далее, после нормализации, программа выполняет следующий шаг – кластеризацию изображений.

3.2 Кластеризация

Кластеризация – это процесс разбиения объектов (в моем случае – массивов с числами) на кластеры, то есть группы. В каждом кластере находятся похожие объекты, отличные от объектов в другом кластере (рисунок 3.7).

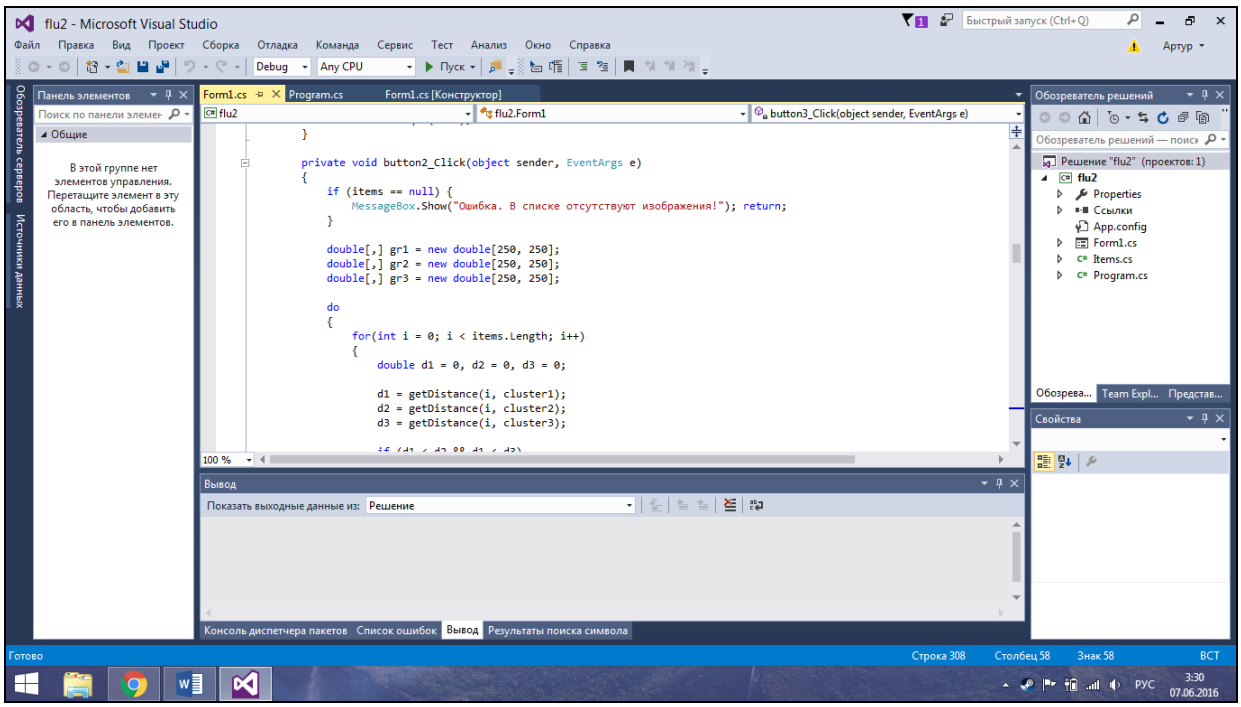


Рисунок 3.7 – Кластеризация

В своей программе я использовал алгоритм кластеризации k-means. Он состоит из четырех шагов.

1. Задается число кластеров k, которое должно быть сформировано из объектов исходной выборки.

2. Случайным образом выбирается k записей исходной выборки, которые будут служить начальными центрами кластеров.

3. Для каждой записи исходной выборки определяется ближайший к ней центр кластера, т.е. вычисляется расстояние между записями и центрами кластеров. Правило, по которому производится вычисление расстояния в многомерном пространстве признаков, называется метрикой. В моей программе я использовал метрику Евклида.

4. Производится вычисление центроидов – центров тяжести кластеров. Это делается путем простого определения средних значений каждого признака для всех записей в кластере.

Шаги 3 и 4 повторяются до тех пор, пока выполнение алгоритма не будет прервано. Остановка алгоритма производится:

- Когда границы кластеров и расположения центроидов не перестанут изменяться от итерации к итерации, т.е. на каждой итерации в каждом кластере будет оставаться один и тот же набор записей. На практике алгоритм k-meansобычно находит набор стабильных кластеров за несколько десятков итераций.

- Когда достигнут критерий сходимости.

На рисунке ниже я отобразил пример кластеризации точек (рисунок 3.8).

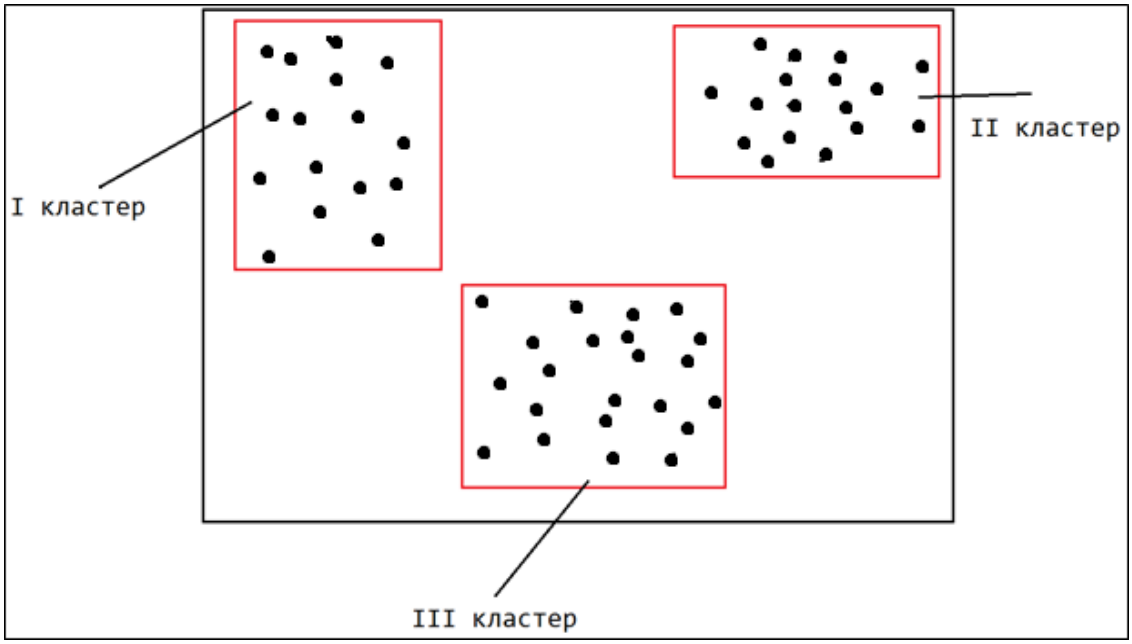


Рисунок 3.8 – Пример кластеризации точек

Выше был приведен пример кластеризации точек. Отличие кластеризации, используемой здесь и у меня в том, что у каждой точки используется 2 значения – x и y, когда в моей же программе значение каждой точки (изображения) используется 62500 значений, так как в каждом массиве содержится 62500 чисел.

Следующим шагом в моем программном обеспечении является изучение свойств каждого кластера.

3.3 Изучение свойств кластера

Под изучением свойств кластера понимается изучение каждого объекта, который попал в кластер. То есть, мы открываем каждый кластер, и

после изучения добавляем ему свойство (описание). Таким образом, мы будем понимать, какие объекты попали в данный кластер (рисунок 3.9).

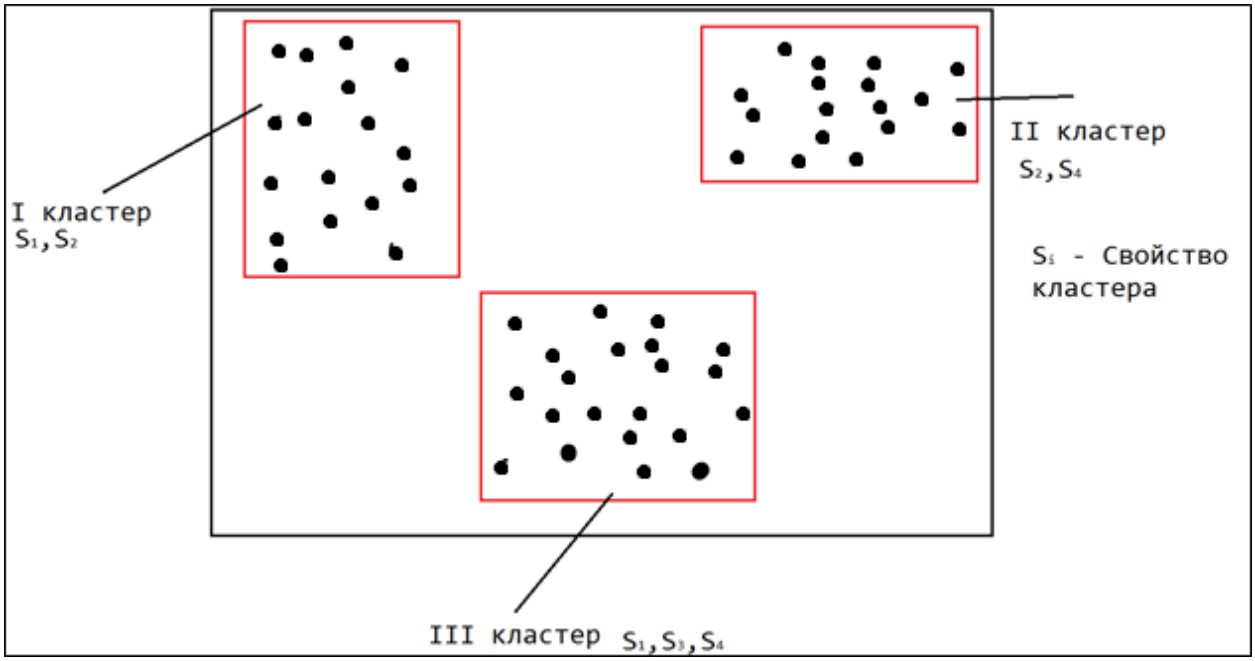


Рисунок 3.9 – Пример изучения свойств кластера

В примере выше возьмем за основу флюорографические снимки, то есть каждая точка — это изображение. На этом примере видно, что в кластеры попали изображения, и изучив их мы дали каждому кластеру описание. Например, у первого кластера свойства S_1, S_2, это значит, что в первый кластер попали здоровые люди (S_1) с вероятностью 80% и немного больные (S_2) с вероятностью в 20%. Таким образом, мы поймем с каким заболеванием человек, увидев к какому кластеру относится его флюорографический снимок. Следующим шагом моего программного обеспечения является подача неизвестного ранее изображения.

3.4 Подача неизвестного ранее изображения

После того как кластеризация закончилась, мы можем двумя шагами определить отношение нового изображения к кластеру (рисунок 3.10).

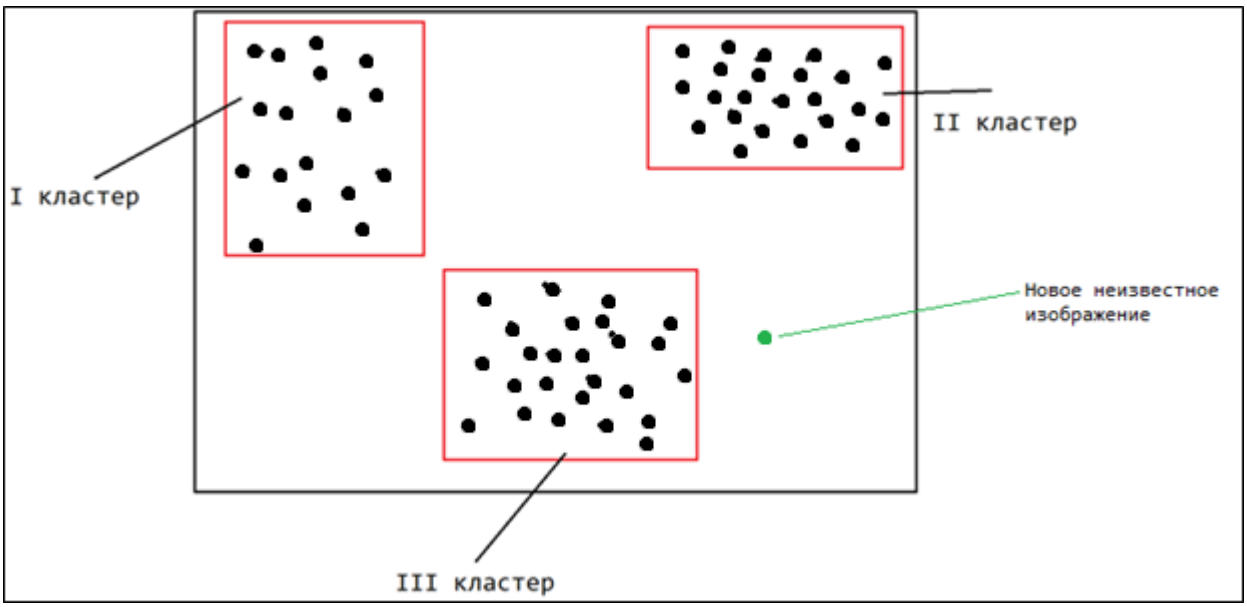


Рисунок 3.10 – Подача неизвестного ранее изображения

После того как мы добавили изображение, мы рассчитываем расстояние от этого изображения к каждому кластеру (рисунок 3.11).

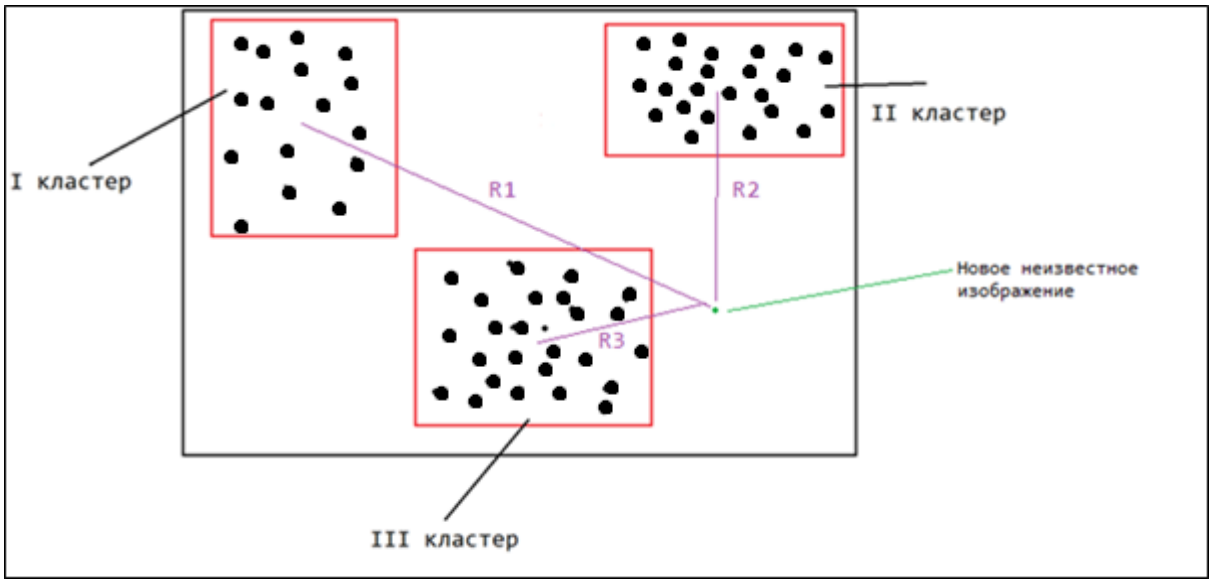


Рисунок 3.11 – Наглядность расчёта расстояния

После расчета расстояния, программа присваивает это изображение к кластеру, расстояние до которого будет наименьшим (рисунок 3.12).

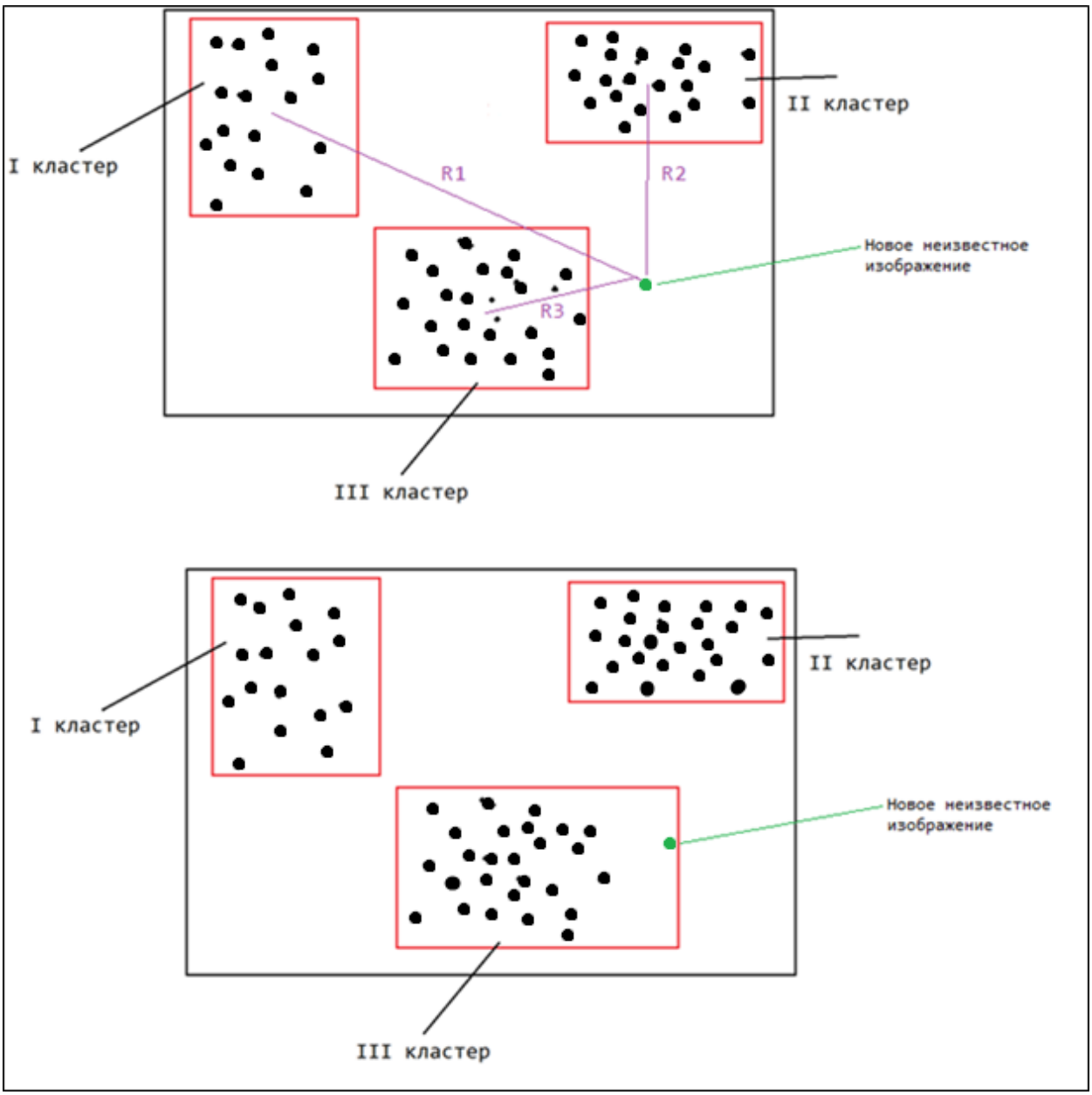


Рисунок 3.12 – Соотношение изображения к кластеру

То есть как видно на Рисунке 15, что расстояние от нового изображения до III-кластера меньше, а значит он принадлежит этому кластеру.

3.5 Интерфейс программы

Для более упрощенного использования данной программы, я реализовал интерфейс, который будет понятен любому пользователю моего программного обеспечения, включая медицинский персонал, для которого было реализована данная программа (рисунок 3.13).

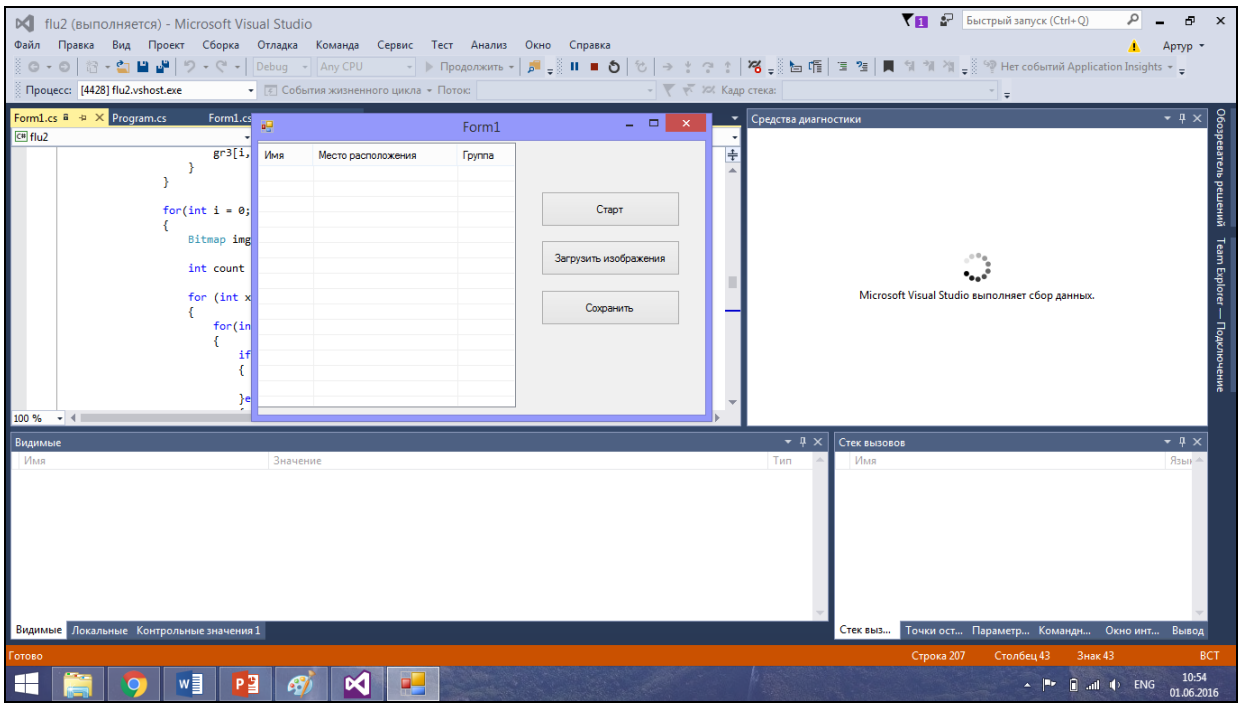


Рисунок 3.13 – интерфейс программного обеспечения

При нажатии кнопки «Загрузить изображения» программа открывает окно, в котором нужно выбрать фотографии, которые будут кластеризованы (Рисунок 3.14).

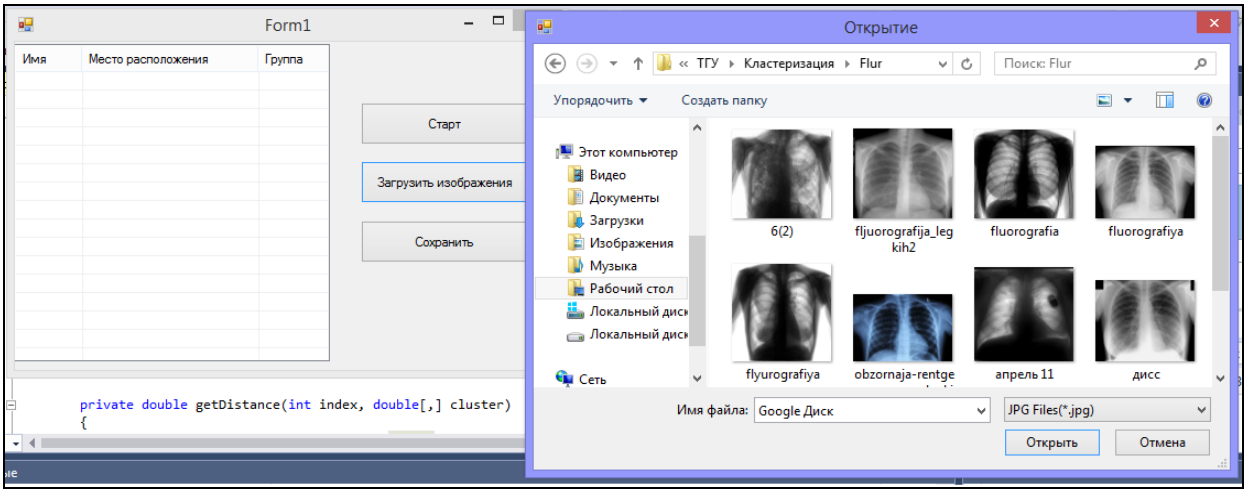


Рисунок 3.14 – Окно выбора фотографий

Если вдруг пользователь не выбрал фотографии, либо выбрал менее 3 фотографий, программа проинформирует об ошибке и даст совет по правильной работе с ней (рисунок 3.15).

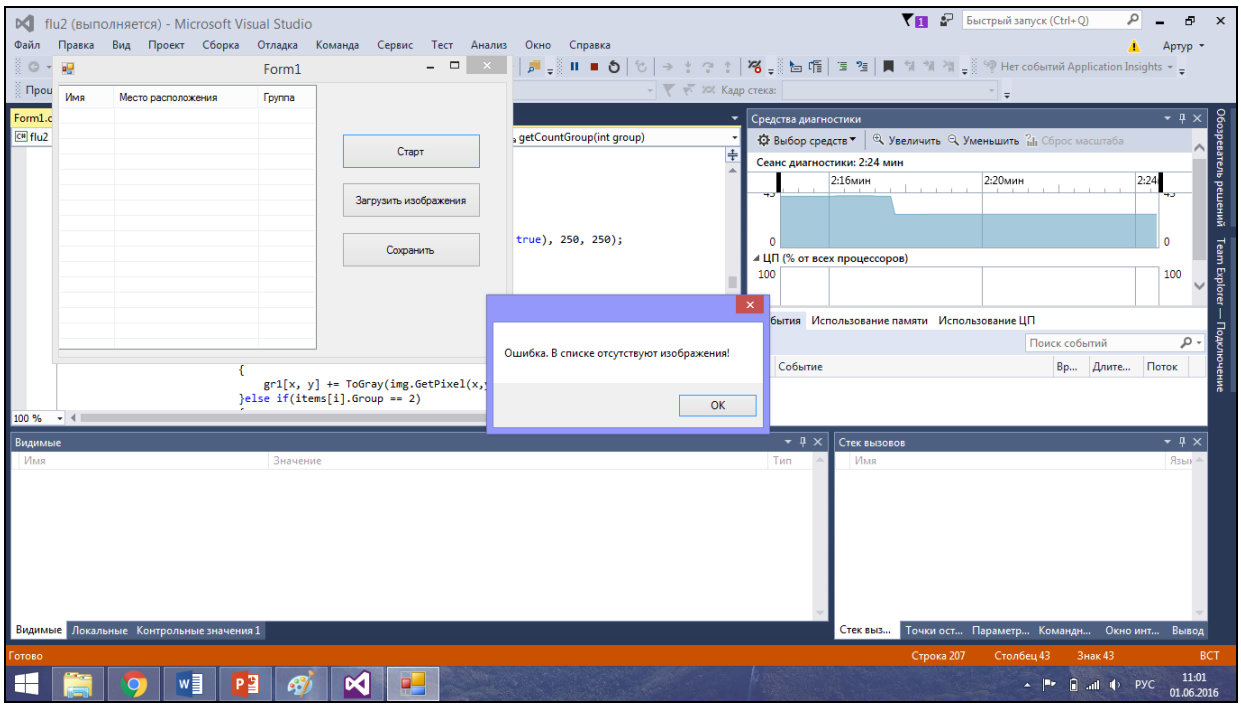


Рисунок 3.15 – Окно сообщения неудачного выполнения задачи

Сделано это для более удобного пользования программой.

После выбора фотографий нажимая кнопку «Старт» программа начинает кластеризацию, и выводит информацию о фотографии: имя файла, расположение, и к какому кластеру она принадлежит (рисунок 3.16).

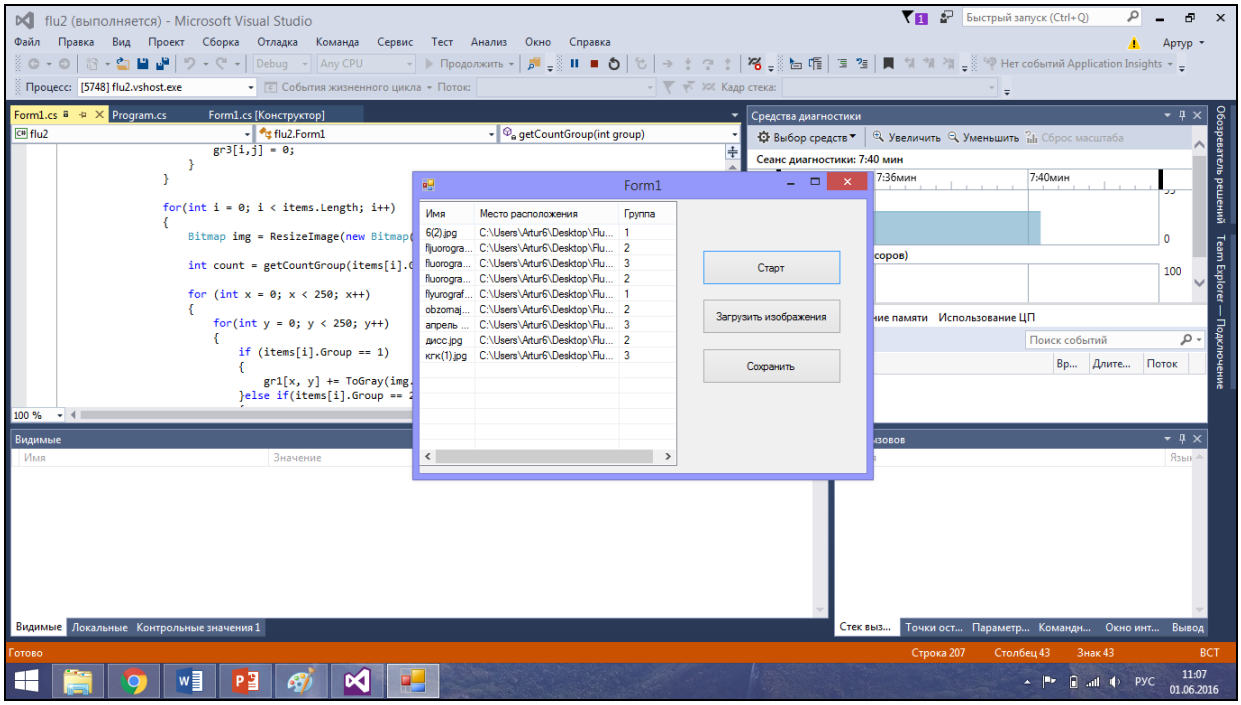


Рисунок 3.16 – Окно информации кластеризации

Также я сделал специальную кнопку сохранения результатов. При нажатии на кнопку «Сохранить» программа выведет окно выбора пути (рисунок 3.17).

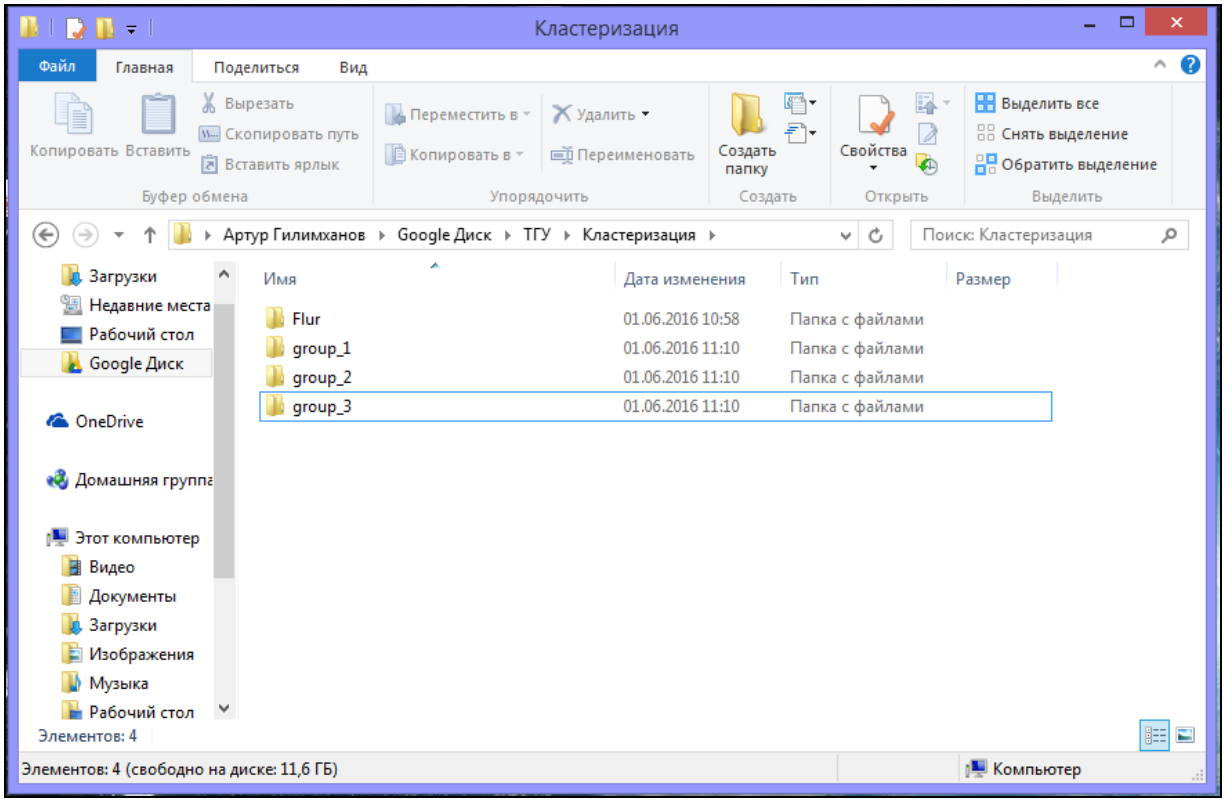


Рисунок 3.17 – Папки с сохраненными итогами

После чего сохранит 3 папки, в которых будут откластеризованные фотографии (рисунок 3.18).

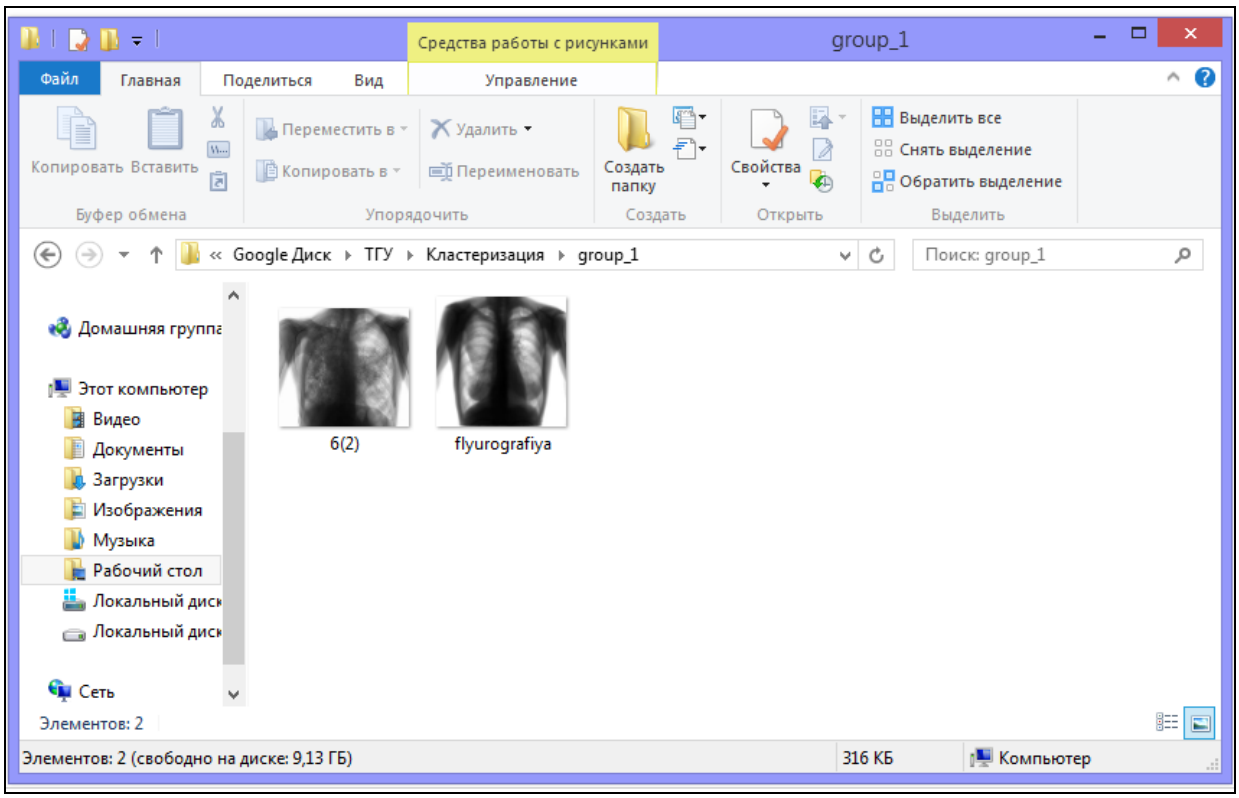


Рисунок 3.18 – Папки с откластеризованными изображениями

3.6 Bitmap

Программа будет извлекать фотографии из базы данных, и при помощи класса Bitmap обрабатывать изображение. Для обработки наших снимков мы используем класс Bitmap. Bitmap он инкапсулирует точечный рисунок, состоящий из данных пикселей графического изображения и атрибутов рисунка [2]. Объект Bitmap используется для работы с изображениями, определяемыми данными пикселей (Рисунок 3.19).

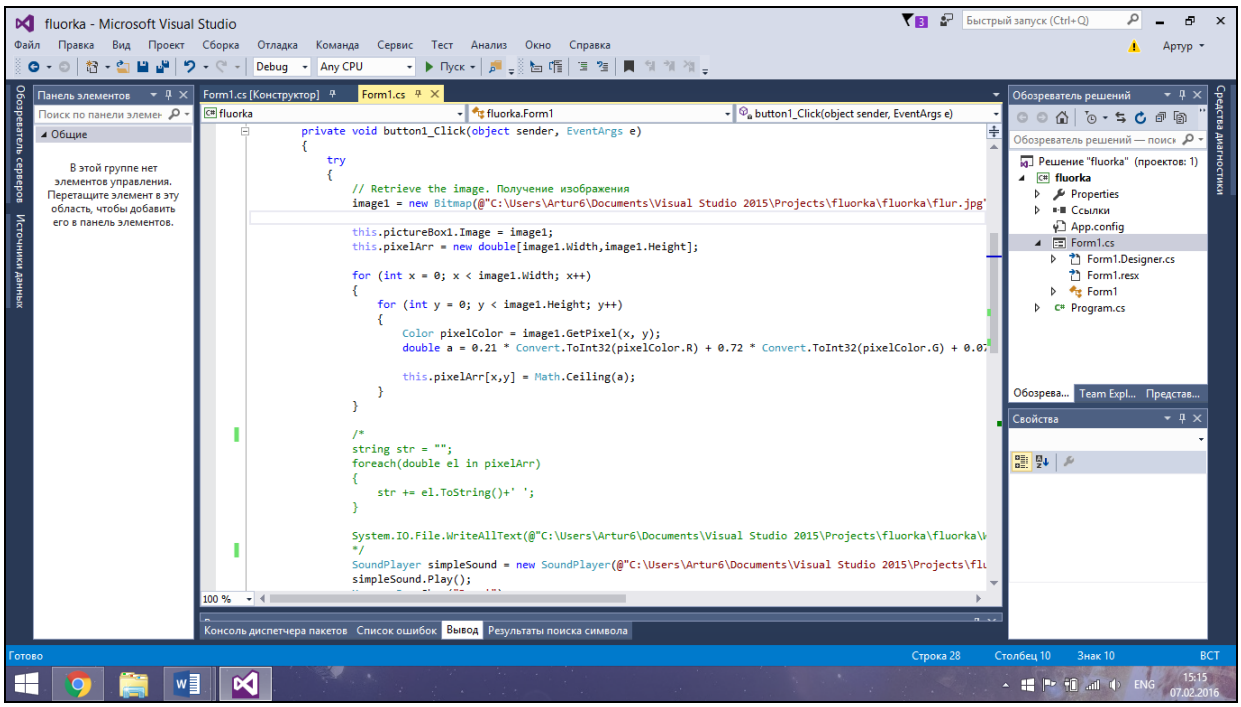


Рисунок 3.19 – Класс Bitmap для обработки изображения

Также мы обрабатываем фотографию методом ResizeImage, чтобы выбранные изображения сделать одного размера.

Мы обрабатываем наше изображение, при помощи метода getpixel получаем цвет каждого пикселя изображения [9]. В конечном итоге получаем наше изображение в виде матрицы чисел, каждое из которых является цветовым индексом каждого пикселя нашего изображения (рисунок 3.20) [10].

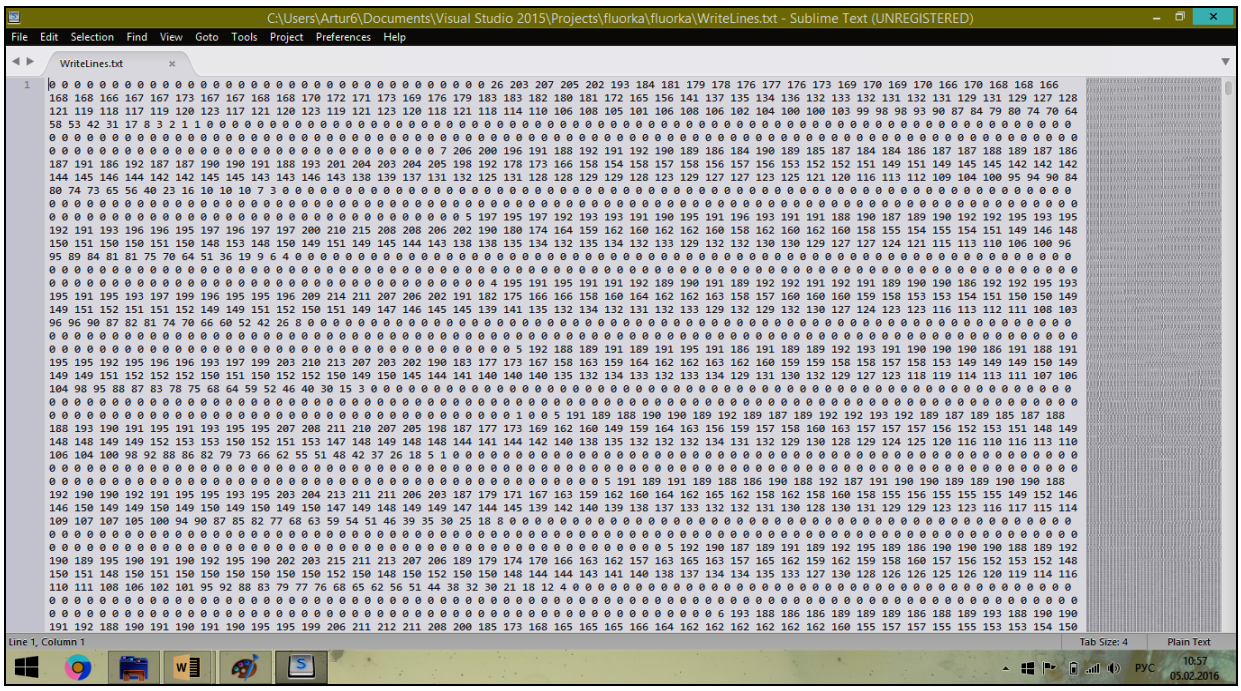


Рисунок 3.20 – Обработанное флюорографическое изображение

Сохраняется этот массив чисел в файл «.txt» формата для более упрощенной работы с этим массивом в дальнейшем.

3.7 Кластеризация флюорографических снимков

При помощи кластеризации методом K-means и метрики Евклида мы кластеризуем наши обработанные флюорографические снимки (рисунок 3.21).

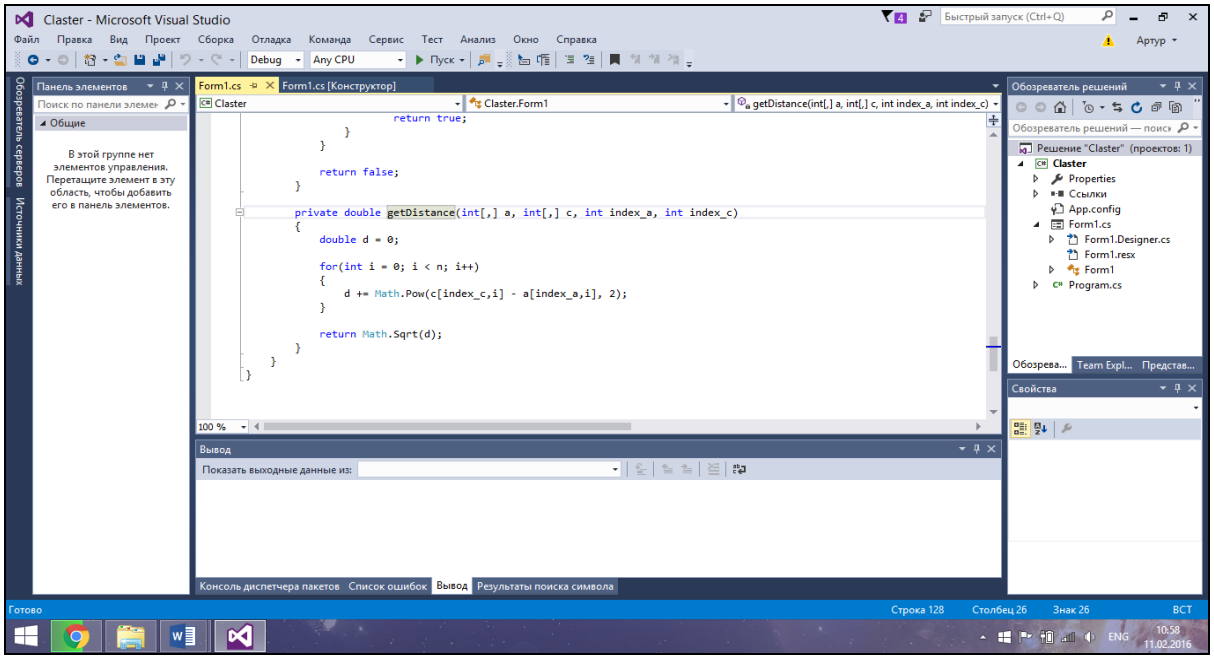


Рисунок 3.21 – Кластеризация

Первый наш шаг – итерация, суть которой заключается в том, что для каждой записи исходной выборки определяется ближайший к ней центр кластера, т.е. вычисляется расстояние между записями и центрами кластеров. (рисунок 3.22).

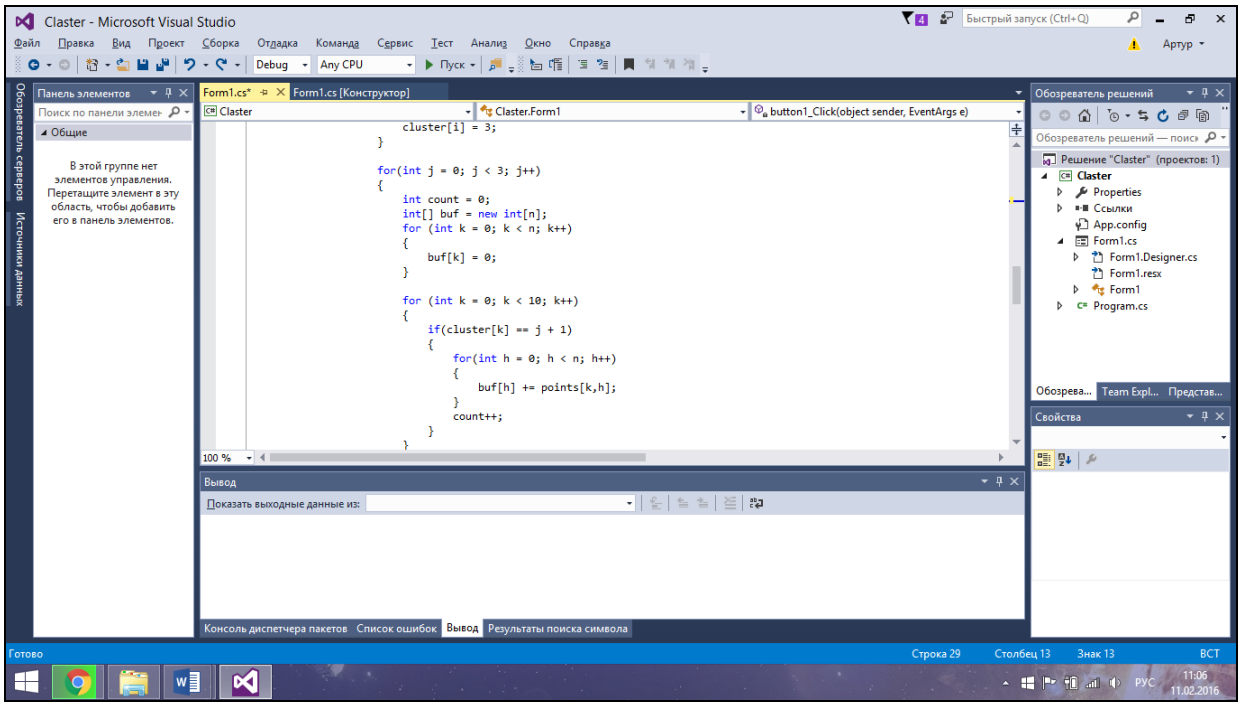


Рисунок 3.22 – Кластеризация изображений

Далее производится вычисление центроидов – центров тяжести кластеров. Это делается путем простого определения средних значений каждого признака для всех записей в кластере. Затем старый центр кластера смещается в его центроид. Таким образом, центроиды становятся новыми центрами кластеров для следующей итерации алгоритма. (рисунок 3.23).

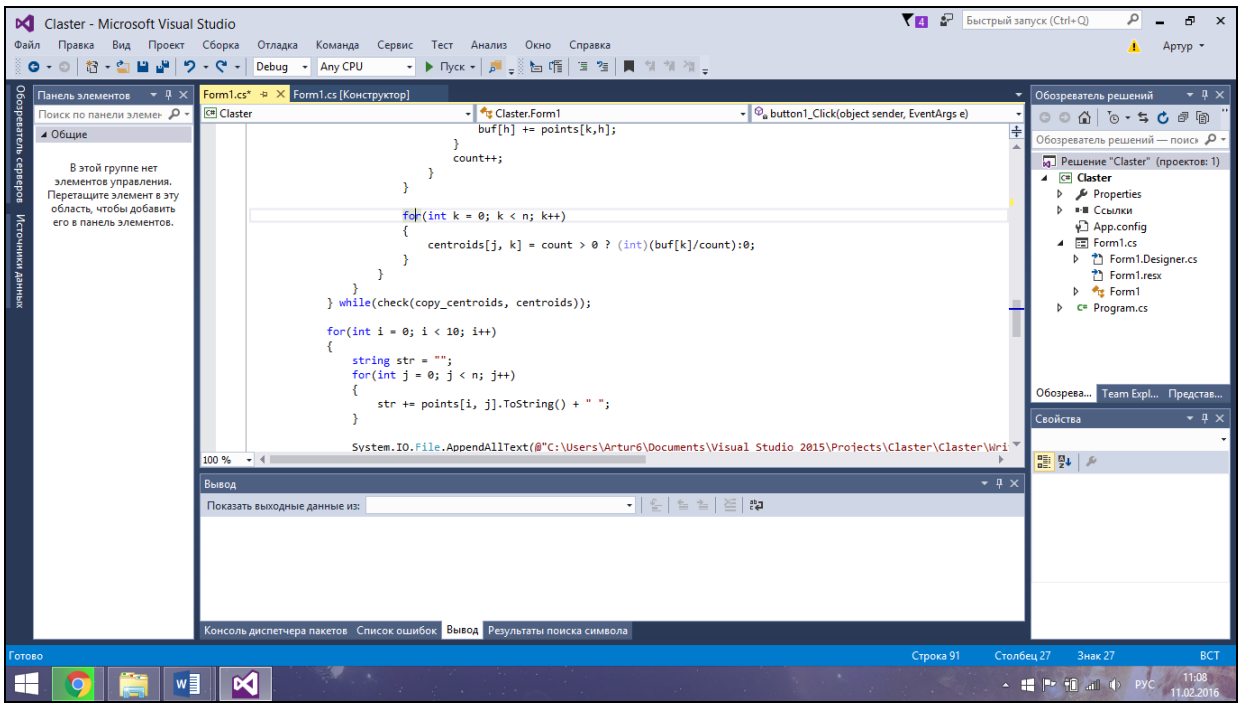


Рисунок 3.23 – Сдвиг (центрирование) кластеров

По завершению центрирования кластеров мы проводим итерацию. Если результат текущей итерации, то есть положение кластеров и критерии сходимости, отличается от результата предыдущей итерации – мы снова проводим центрирование кластеров. Если результаты сходятся – значит кластеризация завершена до конца.

Для более понятной работы моего программного обеспечения, я сделал окно сообщений, в котором говорится о завершении работы программы (рисунок 3.24).

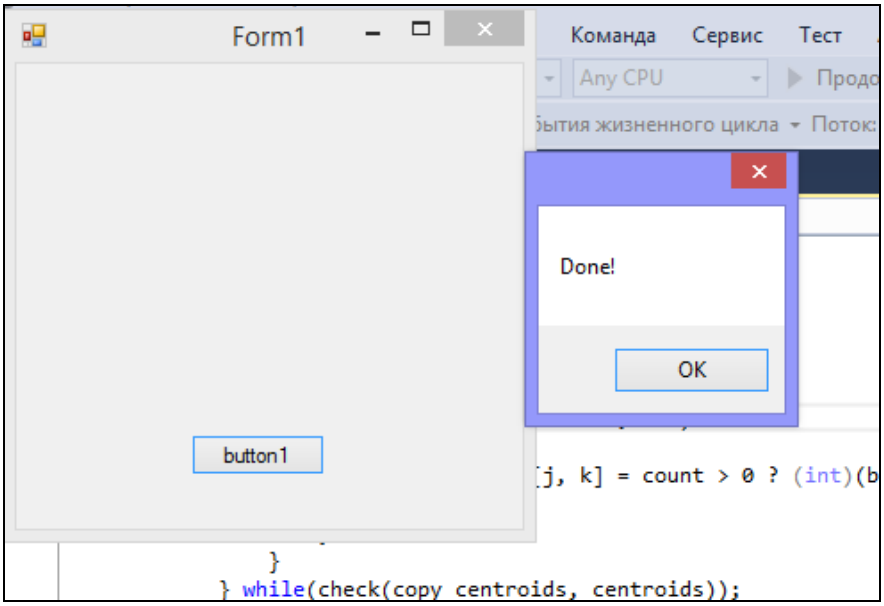


Рисунок 3.24 – Окно сообщений о завершении кластеризации

Так же внутри программы я сделал сохранение массива числа фотографий в отдельный .txt файл (рисунок 3.25).

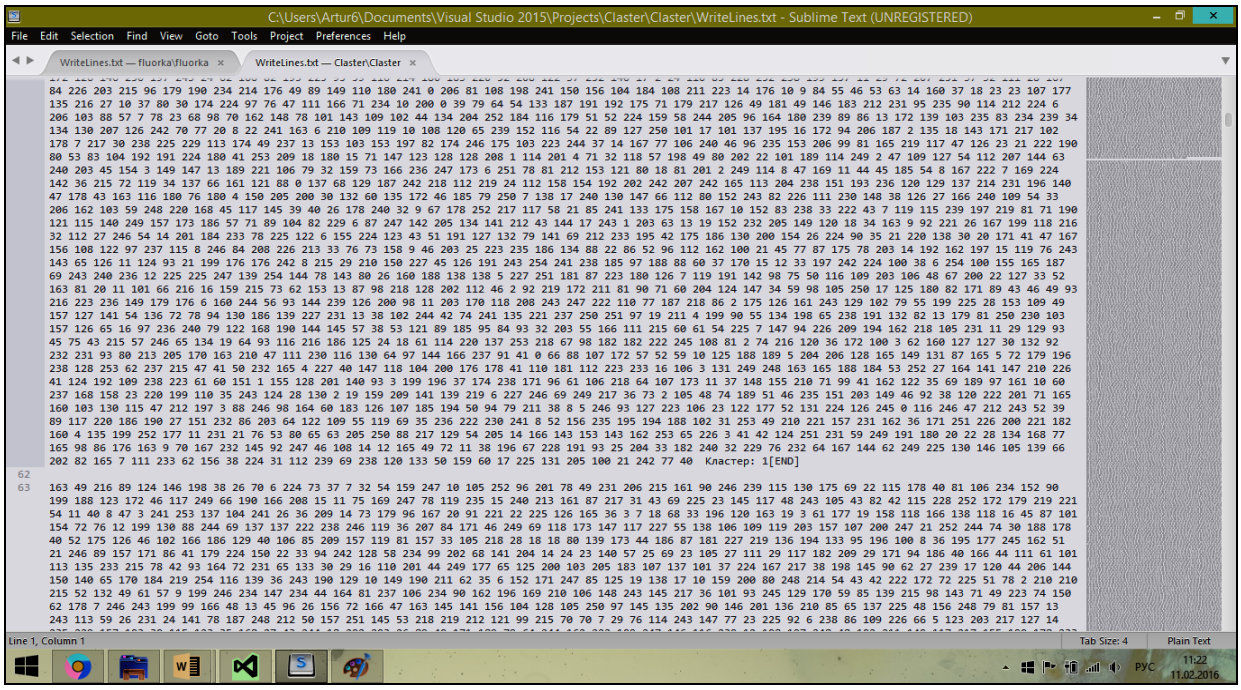


Рисунок 3.25 – Файл .txt с описанием изображений и их отношения к кластерам

В конечном итоге мы получаем сгруппированные изображения, которые можно сохранить в папку, или же просто посмотреть информацию в окне программы (рисунок 3.26, 3.27).

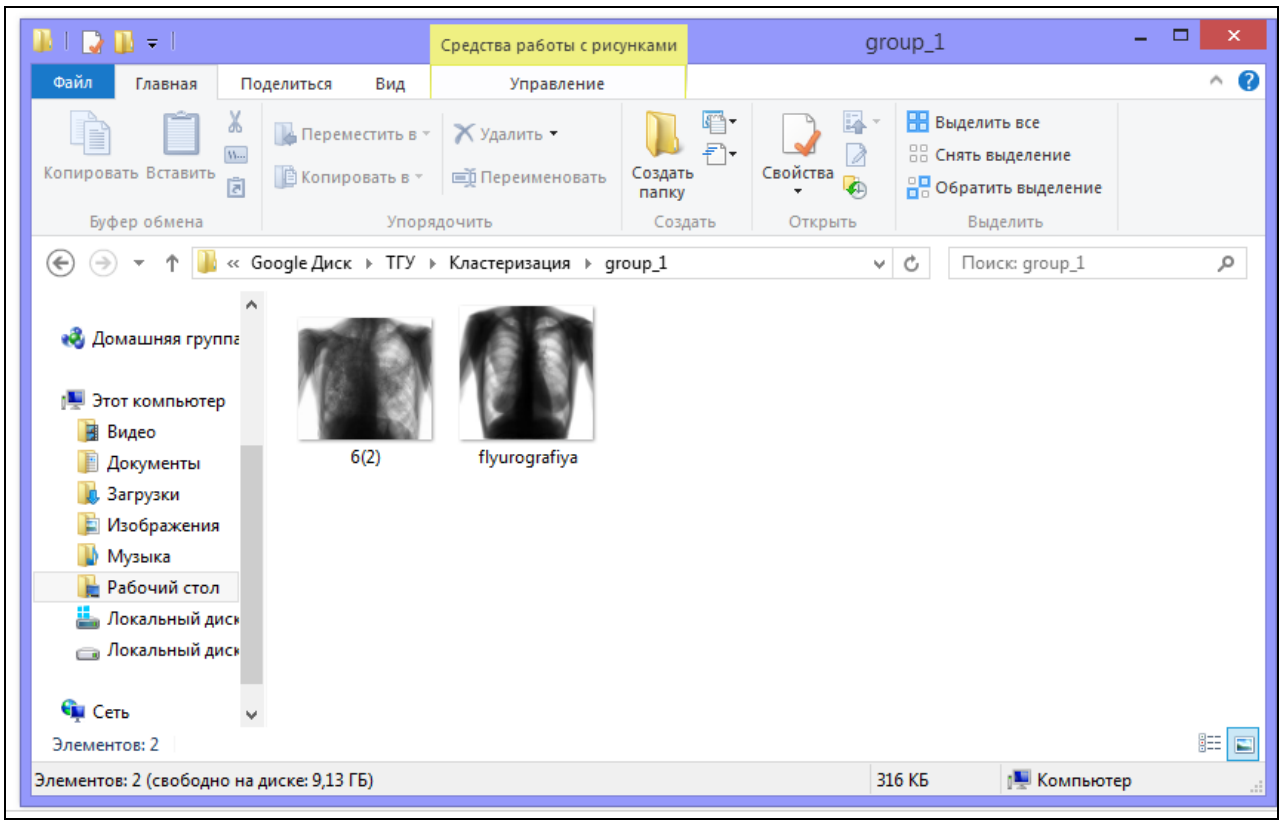


Рисунок 3.26 – Итоговое сохранение всей работы

Здесь уже все изображения соотнесены, и можно назвать с какой вероятностью какое-либо из изображений относится к группе свойств либо здорового человека, либо с заболеванием.

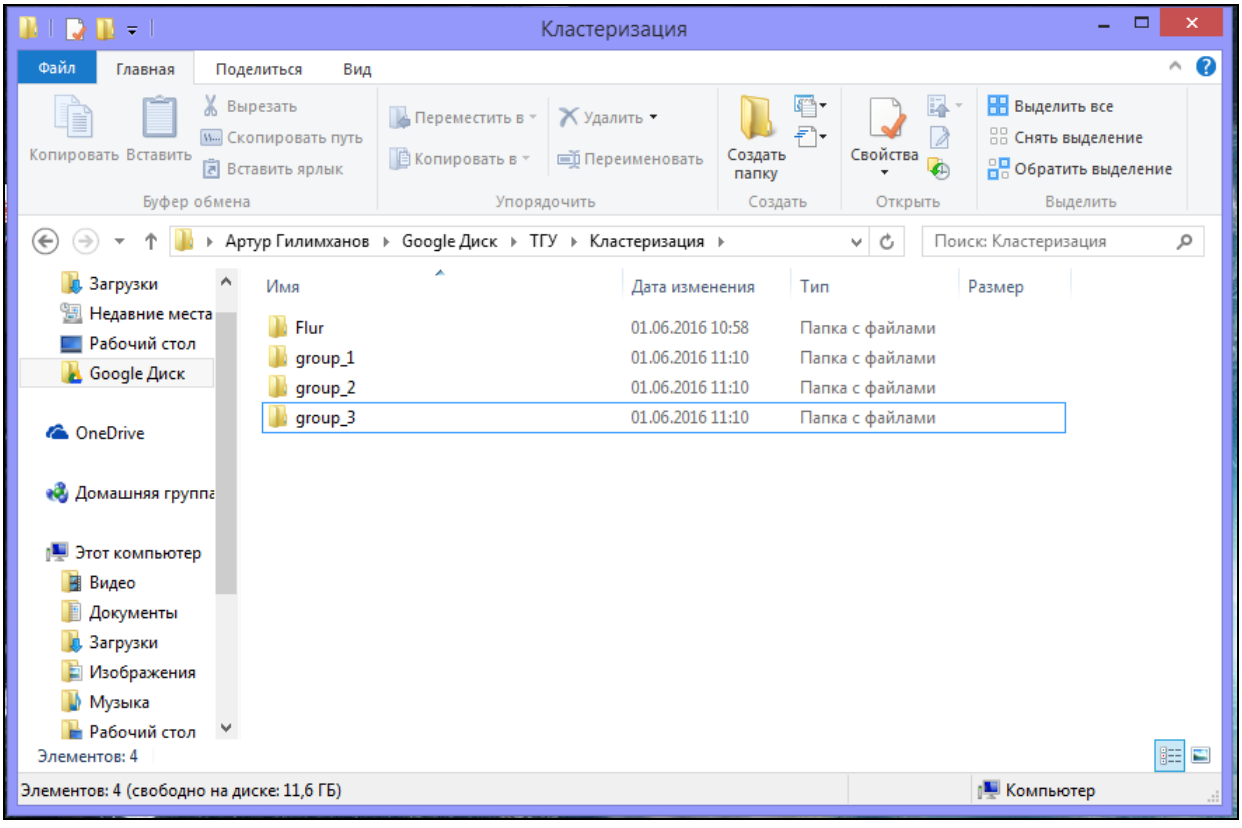


Рисунок 3.27 – Итоговое сохранение всей работы

Также мы получаем папку со всеми изображениями, которые использовались в данном цикле кластеризации (рисунок 3.28).

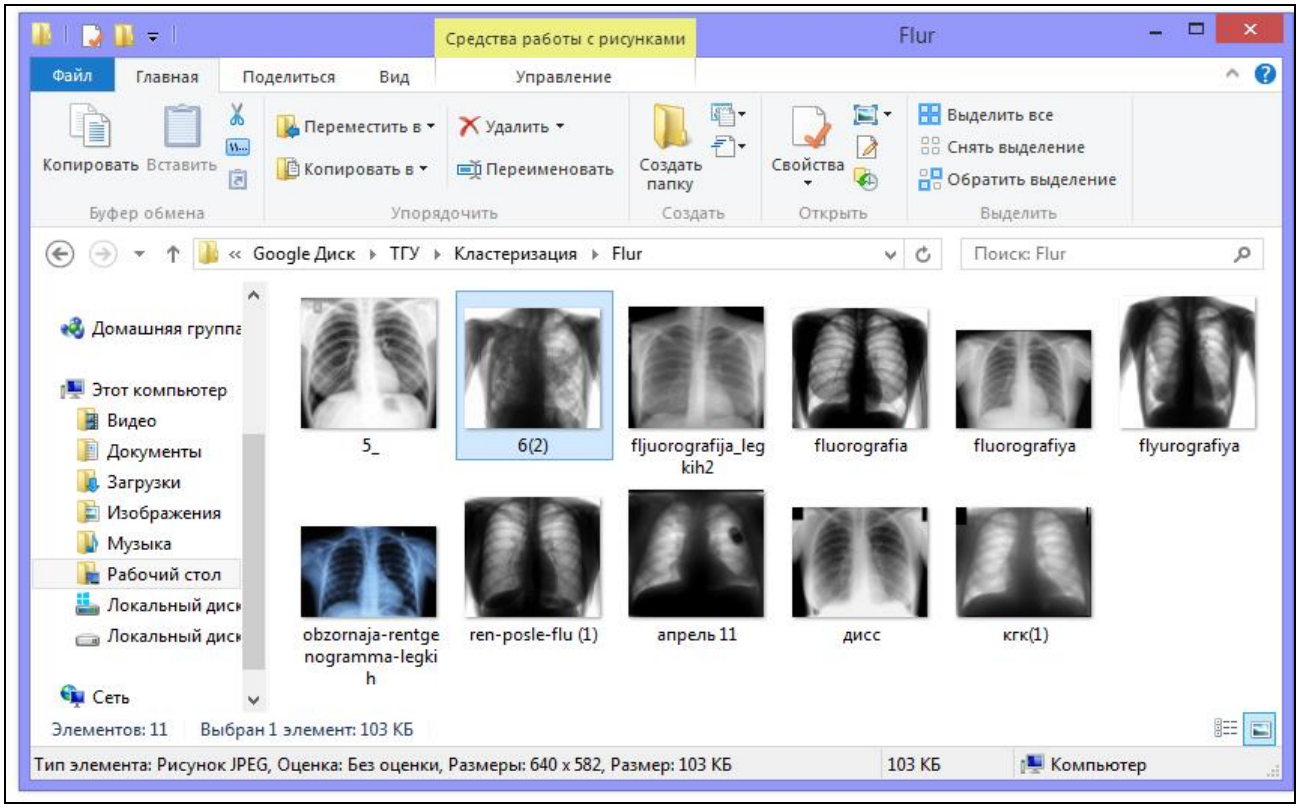


Рисунок 3.28 – Папка со всеми полученными изображениями

3.8 Полученные итоги

Использовав мой алгоритм кластеризации, я получил группу изображений. Для сравнения результатов работы, я взял 3 научные иностранные статьи: «CLUE: Cluster-Based Retrieval of Images. by Unsupervised Learning, Yixin Chen, Member, IEEE, James Z. Wang, Member, IEEE, and Robert Krovetz», «A New Hierarchical Approach for Image Clustering. By Lei Wang and Latifur Khan» и « Clustering visually similar images to improve image search engines. By Thomas Deselaers, Daniel Keysers, and Hermann Ney». В данных статьях были получены схожие результат по кластеризации изображений общего содержания (природа, животные, люди, интерфейсы программы). Однако в данных работах используется более сложный математический аппарат, основанный на алгоритмах Linde-Buzo-Gray, dynamically growing self-organizing tree, Cluster Expander of Compound Graphs.

В нашей работе применяется алгоритм k-means с метрикой Евклида, при этом обеспечивается такие же результаты кластеризации, как и в работах, приведенных выше (Рисунок 3.29).

Кластеры	Отгруппированные изображения			
1 Кластер				
2 Кластер				
3 Кластер				

Рисунок 3.29 – Полученные кластеры

При попытке использования такого же подхода для кластеризации флюорографических изображений были получены неудовлетворительные результаты, с точки зрения автоматизации диагностики заболеваний (Рисунок 3.30). Это связано с тем, что большую часть имеющихся изображений занимают – позвоночник пациент, его ребра, размерные шкалы и другие элементы, не относящиеся к диагностике заболеваний легких и сердца.

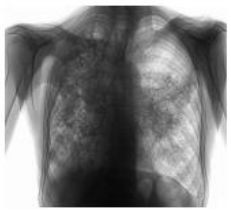
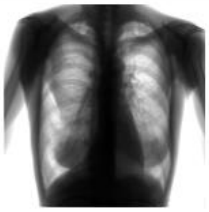
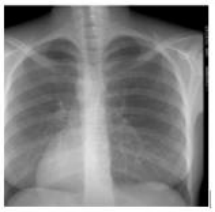
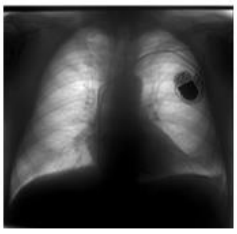
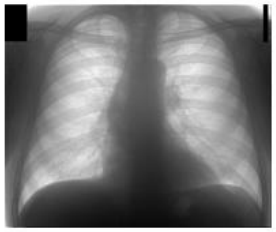
Кластеры	Отгруппированные изображения		
1 Кластер			
2 Кластер			
3 Кластер			

Рисунок 3.30 – Полученные кластеры

Это подтверждает необходимость применения технологий сегментации изображений, т.е. выделений из изображений отельных ее частей, подвергающихся анализа. Ненужные части при сегментации изображений отбрасываются и поэтому они не оказывают вредного влияния на кластеризацию.

Существуют алгоритмы анализа изображений, включающие в себя этапы сегментации и дальнейшего анализа частей изображения. Данные алгоритмы основаны на технологиях искусственного интеллекта, но их применение для решения поставленной задачи затруднительна, по причине малого количества имеющихся изображений (12 штук) и их низкой детализации, предоставленных из городской клинической больницы №5. То есть, нельзя реализовать обучающую выборку удовлетворительного размера для использования более сложных алгоритмов. Алгоритм k-means не требует обучающей выборки больших размеров.

Заключение

При помощи IDE Microsoft Visual Studio 2015 было создано приложение Windows Forms. Программное обеспечение, которое я реализовал, будет проходить 4 шага кластеризации изображений. Первым шагом она проводит нормализацию изображений к единому виду. После нормализации, мое программное обеспечение проводит кластеризацию. В следствие кластеризации мы сможем дать свойство каждому кластеру, полученному после предыдущего шага. Так же, после всех проделанных шагов работы программы, мы сможем соотносить новое изображение к уже полученной с помощью программы кластерной структуры. Таким образом, мы получили новое решение по применению кластеризации графических изображений, который будет нормализовать и кластеризовать изображения точной метрикой Евклида. После реализации программного обеспечения, я провел кластеризацию изображений общего назначения (природа, люди и автомобили), результат которой был схож с результатами статей «CLUE: Cluster-Based Retrieval of Images. by Unsupervised Learning, Yixin Chen, Member, IEEE, James Z. Wang, Member, IEEE, and Robert Krovetz», «A New Hierarchical Approach for Image Clustering. By Lei Wang and Latifur Khan» и « Clustering visually similar images to improve image search engines. By Thomas Deselaers, Daniel Keysers, and Hermann Ney». То есть результат кластеризации изображений с использованием предложенного подхода оказались удовлетворительными. Однако при использовании такого предложенного подхода к кластеризации флюорографических изображений результаты кластеризации неудовлетворительны, так как требуется сегментация изображений для отсечения неинформативных, с точки зрения заболеваний, признаков.

Список используемой литературы

1. MSDN – сеть разработчиков Microsoft [Электронный ресурс]: // Электрон. дан. – Windows Forms – Microsoft Corp. - https://msdn.microsoft.com/ru-ru/library/dd30h2yb(v=vs.110).aspx (дата обращения 01.02.2015)

2. Image Processing Toolbox– Matlab and Toolboxes [Электронный ресурс]: // Электрон. дан. – Обработка сигналов и изображений – MATLAB.Exponenta - http://matlab.exponenta.ru/imageprocess/book2/48.php (дата обращения 01.02.2015)

3. Visual Studio 2015 – Microsoft Corp. [Электронный ресурс]: // Электрон. дан. – Обзор продукта Visual Studio 2015 – Microsoft Corp. - https://www.visualstudio.com/ru-ru/products/vs-2015-product-editions.aspx (дата обращения 01.02.2015)

4. BaseGroup Labs – технология анализа данных [Электронный ресурс]: // Электрон. дан. – Алгоритмы кластеризации на службе Data Mining – BaseGroup Labs - https://basegroup.ru/community/articles/datamining (дата обращения 01.02.2015)

5. Технологии анализа данных. Data Mining, Visual Mining, Text Mining, OLAP // А. А. Барсегян, М. С. Куприянов, В. В. Степаненко, И. И. Холод , БХВ-Петербург. 2007. С.35.

6. Методы и модели анализа данных: OLAP и Data Mining // А.А. Барсегян, М.С. Куприянов, В.В. Степаненко, И.И. Холод, БХВ-Петербург. 2007. С.12.

7. Бизнес-аналитика: от данных к знаниям // Паклин Н.Б., Орешков В.И. , СПб.: Питер. 2013. С.2

8. Анализ данных и процессов // Барсегян А., Куприянов М., Холод И., Тесс М., Елизаров С. , БХВ-Петербург. 2009. С.12.

9. Информационные технологии анализа данных. Data Analysis // Юрий Петрунин , КДУ 2010. С.87

10. MSDN – сеть разработчиков Microsoft [Электронный ресурс]: // Электрон. дан. - Bitmap - класс (System.Drawing) – Microsoft Corp. - https://msdn.microsoft.com/ru-ru/library/system.drawing.bitmap(v=vs.110).aspx (дата обращения 01.02.2015)

11. Цифровая обработка изображений // Рафаэль Гонсалес, Ричард Вудс , Техносфера 2006. С.59

12. Richard P. Lippmann – Information [Electronic resource]: [Информация о докторе Ричарде Липпмане] . – Electronic data. – Cambridge: Massachusetts Institute of Technology. – Mode of access: https://www.ll.mit.edu/mission/cybersec/cybersec-bios/lippmann-bio.html

13. Хайкин С. Нейронные сети. Полный курс. 2-е издание / С. Хайкин; пер. с анг. Н.Н. Куссуль и А.Ю. Шелестова, под ред. Н.Н. Куссуль. – 2-е изд. – М.: «Вильямс», 2006. – 1104 с.

14. Koprinkova-Hristova, P. Artificial Neural Networks / Petia Koprinkova-Hristova, Valeri Mladenov, Nikola K. Kasabov. – Springer International Publishing, 2015. – 488 p.

15. Li, S.Z. Hamming Distance / S.Z. Li, A. Jain. – Encyclopedia of Biometrics, 2009. – 668 p.

16. Rigatos, G. G. Advanced Models of Neural Networks / Gerasimos G. Rigatos. – Springer Berlin Heidelberg, 2015. – 396 p.

17. F. Santoyo, E. Chávez, E.S. Téllez, «A Compressed Index for Hamming Distances», Similarity Search and Applications, Oct. 2014, pp. 113-126.

18. Ульман, Л. PHP и MySQL. Создание интернет-магазинов, 2-е изд. / Л. Ульман; пер. с анг. А. Сергеев. – 2-е изд. – М.: «Вильямс», 2015. – 544 с.

19. Бейли, Л. Изучаем SQL / Л. Бейли. – СПб.: Изд-во «Питер», 2012. – 573 с.

20. Прохоренок, Н.А. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / Н.А. Прохоренок, В.А. Дронов. – 4-е изд. – СПб.: Изд-во «БХВ-Петербург», 2015. – 766 с.

Приложение А

Код программы

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.IO;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;

namespace flu2
{
    public partial class Form1 : Form
    {
        Items[] items;
        double[,] cluster1 = new double[250, 250];
        double[,] cluster2 = new double[250, 250];
        double[,] cluster3 = new double[250, 250];
        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog1 = new OpenFileDialog();
            openFileDialog1.Filter = "JPG Files(*.jpg)|*.jpg|BMP Files
(*.bmp)|*.bmp|JPEG Files (*.jpeg)|*.jpeg";
            openFileDialog1.FilterIndex = 1;
            openFileDialog1.RestoreDirectory = true;
            openFileDialog1.Multiselect = true;
```

```
            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
               if (openFileDialog1.SafeFileNames.Length < 3)
               {
                  MessageBox.Show("Ошибка. Необходимо загрузить не менее трёх изображений."); return;
               }

               listView1.Items.Clear();
               items = new Items[openFileDialog1.SafeFileNames.Length];

               for (int index = 0; index < openFileDialog1.SafeFileNames.Length; index++) {
                  FileInfo info = new FileInfo(@openFileDialog1.FileNames[index]);

                  items[index] = new Items();
                  items[index].FileName = info.Name;
                  items[index].Path = info.FullName;

                  ListViewItem lvi = new ListViewItem(info.Name);
                  lvi.SubItems.Add(info.FullName);
                  lvi.SubItems.Add("0");
                  listView1.Items.Add(lvi);
               }

               Bitmap cl1 = ResizeImage(new Bitmap(@items[0].Path, true), 250, 250);
               Bitmap cl2 = ResizeImage(new Bitmap(@items[1].Path, true), 250, 250);
```

```
            Bitmap cl3 = ResizeImage(new Bitmap(@items[2].Path, true), 250,
250);
            for(int i = 0; i < 250; i++)
            {
               for(int j = 0; j < 250; j++)
               {
                  Color pixel = cl1.GetPixel(i, j);
                  cluster1[i, j] = Math.Ceiling(ToGray(pixel));
               }
            }
            for (int i = 0; i < 250; i++)
            {
               for (int j = 0; j < 250; j++)
               {
                  Color pixel = cl2.GetPixel(i, j);
                  cluster2[i, j] = Math.Ceiling(ToGray(pixel));
               }
            }
            for (int i = 0; i < 250; i++)
            {
               for (int j = 0; j < 250; j++)
               {
                  Color pixel = cl3.GetPixel(i, j);
                  cluster3[i, j] = Math.Ceiling(ToGray(pixel));
               }
            }
            GC.Collect();
         }
      }
      private double ToGray(Color pixel)
```

```
        {
            return 0.21 * Convert.ToInt32(pixel.R) + 0.72 *
Convert.ToInt32(pixel.G) + 0.07 * Convert.ToInt32(pixel.B);
        }

        private double getDistance(int index, double[,] cluster)
        {
            Bitmap img = ResizeImage(new Bitmap(@items[index].Path, true), 250,
250);
            double dist = 0;
            for(int i = 0; i < 250; i++)
            {
                for(int j = 0; j < 250; j++)
                {
                    dist += Math.Pow(cluster[i, j] - ToGray(img.GetPixel(i,j)), 2);
                }
            }
            return Math.Sqrt(dist);
        }
        private void button2_Click(object sender, EventArgs e)
        {
            if (items == null) {
                MessageBox.Show("Ошибка. В списке отсутствуют
изображения!"); return;
            }
            double[,] gr1 = new double[250, 250];
            double[,] gr2 = new double[250, 250];
            double[,] gr3 = new double[250, 250];
            do
            {
```

```
for(int i = 0; i < items.Length; i++)
{
    double d1 = 0, d2 = 0, d3 = 0;

    d1 = getDistance(i, cluster1);
    d2 = getDistance(i, cluster2);
    d3 = getDistance(i, cluster3);

    if (d1 < d2 && d1 < d3)
    {
        items[i].Group = 1;
    }
    else if (d2 < d1 && d2 < d3)
    {
        items[i].Group = 2;
    }
    else
    {
        items[i].Group = 3;
    }

    GC.Collect();
}

for(int i = 0; i < 250; i++)
{
    for(int j = 0; j < 250; j++)
    {
        gr1[i,j] = 0;
        gr2[i,j] = 0;
```

```
                gr3[i,j] = 0;
            }
        }
        for(int i = 0; i < items.Length; i++)
        {
            Bitmap img = ResizeImage(new Bitmap(@items[i].Path, true), 250,
250);
            int count = getCountGroup(items[i].Group);
            for (int x = 0; x < 250; x++)
            {
                for(int y = 0; y < 250; y++)
                {
                    if (items[i].Group == 1)
                    {
                        gr1[x, y] += ToGray(img.GetPixel(x,y)) / count;
                    }else if(items[i].Group == 2)
                    {
                        gr2[x, y] += ToGray(img.GetPixel(x, y)) / count;
                    }else
                    {
                        gr3[x, y] += ToGray(img.GetPixel(x, y)) / count;
                    }
                }
            }
            GC.Collect();
        }
        GC.Collect();
        //MessageBox.Show("Итерация закончена");
    } while (difference(gr1, gr2, gr3));
    listView1.Items.Clear();
```

```
    for (int index = 0; index < items.Length; index++)
    {
        ListViewItem lvi = new ListViewItem(items[index].FileName);
        lvi.SubItems.Add(items[index].Path);
        lvi.SubItems.Add(items[index].Group.ToString());
        listView1.Items.Add(lvi);
    }
}
private int getCountGroup(int group)
{
    int count = 0;
    for(int i = 0; i < items.Length; i++)
    {
        if(items[i].Group == group)
        {
            count++;
        }
    }
    return count;
}
private bool difference(double[,] cl1, double[,] cl2, double[,] cl3)
{
    bool flag = false;
    for(int i = 0; i < 250; i++)
    {
        for(int j = 0; j < 250; j++)
        {
            if (cluster1[i, j] != Math.Ceiling(cl1[i, j])) flag = true;
            cluster1[i, j] = Math.Ceiling(cl1[i,j]);
```

```
            }
        }
        for (int i = 0; i < 250; i++)
        {
            for (int j = 0; j < 250; j++)
            {
                if (cluster2[i, j] != Math.Ceiling(cl2[i, j])) flag = true;
                cluster2[i, j] = Math.Ceiling(cl2[i, j]);
            }
        }
        for (int i = 0; i < 250; i++)
        {
            for (int j = 0; j < 250; j++)
            {
                if (cluster3[i, j] != Math.Ceiling(cl3[i, j])) flag = true;
                cluster3[i, j] = Math.Ceiling(cl3[i, j]);
            }
        }
        return flag;
    }
    private Bitmap ResizeImage(Image image, int width, int height)
    {
        var destRect = new Rectangle(0, 0, width, height);
        var destImage = new Bitmap(width, height);
        destImage.SetResolution(image.HorizontalResolution,
image.VerticalResolution);
        using (var graphics = Graphics.FromImage(destImage))
        {
            graphics.CompositingMode = CompositingMode.SourceCopy;
            graphics.CompositingQuality = CompositingQuality.HighQuality;
```

```
            graphics.InterpolationMode = InterpolationMode.HighQualityBicubic;
            graphics.SmoothingMode = SmoothingMode.HighQuality;
            graphics.PixelOffsetMode = PixelOffsetMode.HighQuality;
            using (var wrapMode = new ImageAttributes())
            {
               wrapMode.SetWrapMode(WrapMode.TileFlipXY);
               graphics.DrawImage(image, destRect, 0, 0, image.Width,
image.Height, GraphicsUnit.Pixel, wrapMode);
            }
         }
         return destImage;
      }
      private void button3_Click(object sender, EventArgs e)
      {
         FolderBrowserDialog openFolderDialog = new FolderBrowserDialog();
         String path = "";
         if (openFolderDialog.ShowDialog() == DialogResult.OK)
         {
            path = openFolderDialog.SelectedPath;
         }
         try
         {
            Directory.Delete(Path.Combine(path, "group_1"), true);
         }
         catch { }
         try
         {
            Directory.Delete(Path.Combine(path, "group_2"), true);
         }
         catch { }
```

```
            try
            {
               Directory.Delete(Path.Combine(path, "group_2"), true);
            }
            catch { }
            for (int i = 0; i < items.Length; i++)
            {
               String subfolder = Path.Combine(path,
"group_"+items[i].Group.ToString());
               if (!File.Exists(subfolder))
               {
                  Directory.CreateDirectory(subfolder);
               }
               Bitmap img = new Bitmap(@items[i].Path, true);
               img.Save(@subfolder+"\\"+items[i].FileName);
            }
            MessageBox.Show("Файлы успешно сохранены.");
         }
      }
   }
```