МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДНР

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ»

Факультет Физико-технический

Кафедра Компьютерных технологий (КТ)

И.о.зав. кафедрой КТ

_____ Т.В. Ермоленко
(подпись)

«___» _____ 2017 г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе бакалавра 2 курса
на тему:

Разработка приложения «Восстановление поврежденных изображений»

Автор работы _____ А.А. Плейсхолдер
подпись

Направление 09.03.01 Информатика и вычислительная техника

Руководитель работы _____ ст.преп. А.А. Тичер
подпись

Консультанты по разделам:

Техническое задание _____ доц. Т.В. Шарий
подпись

Нормоконтроль _____ ст. лаборант В.Г. Медведева
подпись

Курсовая работа защищена _____ _____
дата итоговая оценка комиссия

Подписи членов комиссии: _____
_____
_____
_____

Донецк
2017

ГОУ ВПО «Донецкий национальный университет»
кафедра Компьютерных технологий

Утверждаю
зав. кафедрой КТ

_____
подпись

_____
дата

ЗАДАНИЕ

на курсовую работу студента 2 курса Плейсхолдера А.А.

Тема курсовой работы: Разработка приложения «Восстановление поврежденных изображений».

Краткая постановка задачи: 1. Изучить и проанализировать предметную область цифровой обработки изображений. 2. Ознакомиться с существующими программными продуктами для решения задачи. 3. Разработать техническое задание на создание приложения. 4. Разработать проект программного обеспечения: спроектировать классы и интерфейсы (не менее семи) предметной области; спроектировать графический интерфейс пользователя; оформить техническую документацию проекта. 5. Разработать средствами языка программирования C#, приложение, которое позволяет загружать изображения, искусственно наносить им повреждения и в соответствии с настройками пользователя восстанавливать их. 6. Оформить отчёт.

Исходные данные: 1. Литературные источники по цифровой обработке изображений. 2. Документация по WinForms. 3. Набор тестовых изображений в форматах bmp и jpg.

Ожидаемые результаты: настольное приложение для цифровой обработки изображений и исследования методов восстановления поврежденных изображений

Календарный план работы:

Даты консультаций	Этапы выполнения работы	Отметки о выполнении
30.01.2017	Постановка задачи и обсуждение литературы	выполнено
08.02.2017	Предварительное утверждение содержания отчёта	выполнено
22.02.2017	Утверждение проекта, алгоритмов, методов, технологий	выполнено
01.03.2017	Ход реализации проекта	выполнено
21.03.2017	Обсуждения организации тестирования программы	выполнено
12.04.2017	Демонстрация программного продукта руководителю	выполнено
19.04.2017	Оформление отчёта	выполнено
26.04.2017	Предоставление отчёта руководителю	выполнено

Дата выдачи задания 30.01.2017 года

Студент _____ Плейсхолдер А.А.

Руководитель _____ Тичер А.А.

АННОТАЦИЯ

Отчёт о курсовой работе: 34 с., 7 рис., 7 табл., 2 приложения, 8 источников.

Объект исследования – цифровое изображение.

Предмет исследования – модели и методы цифровой обработки поврежденных изображений с целью их восстановления.

Цель работы – разработка программы, которая позволяет загружать изображения, искусственно наносить им повреждения и в соответствии с настрйоками пользователя восстанавливать их.

Метод исследования – методы цифровой обработки изображений, методы объектно-ориентированного проектирования и программирования.

В работе были использованы технологии C#, Windows Forms, .NET Framework 4.6.

В результате решения задачи было разработано приложение по обработке изображений. Приложение позволяет загружать картинки, искусственно зашумлять их и затем восстанавливать.

Дальнейшее развитие программы связано с расширением ее функционала, увеличением опций, а также подключением базы данных для организации персистентности информации системы.

ИЗОБРАЖЕНИЕ, ФИЛЬТР, ЦВЕТОВАЯ СХЕМА, BMP, JPEG, C#, .NET, WINFORMS.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Создание искусственных систем обработки изображений остаётся сложной теоретической и технической проблемой. Необходимость в таких системах возникает в самых разных областях. Они могут использоваться в системах видеонаблюдения, в медицине, в технической диагностике и в других областях. Важность и актуальность данного вопроса обусловлена, в первую очередь, ростом объемов получаемой информации, необходимостью в качественной, максимально быстрой ее обработке.

Обработка изображений представляет собой любую форму обработки информации, для которой входные данные представлены изображением, например, фотографиями или видеокадрами. Обработка изображений может осуществляться как для получения изображения на выходе (например, подготовка к полиграфическому тиражированию, к телетрансляции и т. д.), так и для получения другой информации (например, распознавание текста, подсчёт числа и типа клеток в поле микроскопа и т. д.).

При компьютерной обработке изображений решается широкий круг задач, таких как улучшение качества изображений, измерение параметров, спектральный анализ многомерных сигналов, распознавание изображений, сжатие изображений и т.д.

Сегодня трудно представить область деятельности, в которой можно обойтись без компьютерной обработки изображений. Интернет, сотовый телефон, видеокамера, фотоаппарат, сканер, принтер, вошедшие в наш быт – немыслимы без компьютерной обработки изображений.

Целью курсовой работы является разработка программы обработки изображений "Восстановление поврежденных изображений". Данная программа должна предоставлять удобный интерфейс для выбора интересующего пользователя изображения в формате .bmp или .jpg, выбора параметров преобразования данного изображения и просмотра результатов его преобразования.

Для достижения поставленной цели необходимо решить задачи:

1. Изучить литературу и освоить алгоритмы, методы и особенности цифровой обработки изображений.
2. На основе полученных знаний разработать структуры данных для хранения в памяти входа, выхода, изображений и внутреннего состояния сущностей приложения.
3. Используя алгоритмы обработки изображений разработать программу «Восстановление поврежденных изображений».
4. Оформить проектную и техническую документацию по программе.

1 Анализ предметной области

Восстановление изображений. Задача улучшения изображения заключается в изменении его качества, которое зависит от особенностей человеческого зрения [1-4]. Целью восстановления является реконструкция изображения, которое ранее было искажено в результате внешнего воздействия, о котором имеется некоторая информация. Поэтому методы восстановления изображения основаны на моделировании процессов искажения и применении обратных процессов для воссоздания исходного изображения. Процесс ухудшения изображения моделируется в виде функции искажения, которая вместе с аддитивным шумом действует на исходное изображение *f(x,y)* и порождает искаженное изображение *g(x,y)*:

$$g(x,y) = H[f(x,y)] + \eta(x,y) \tag{1.1}$$

Имея функцию *g(x,y)* и обладая некоторой информацией об искажающем операторе *H* и зная основные характеристики аддитивного шума *η(x,y)*, можно построить некоторое приближение:

$$\tilde{f}(x,y) = H^{-1}[g(x,y) - \eta(x,y)] \tag{1.2}$$

Поскольку функция шума *η(x,y)* случайная и ее нельзя смоделировать абсолютно точно, то возможно только приблизительное восстановление изображения. Поэтому целью восстановления изображения является построение приближения $\tilde{f}(x,y)$, которое было бы максимально близко к исходному изображению. При этом чем больше информации мы имеем об операторе *H* и шуме *η(x,y)*, тем точнее можно приблизить изображение *f(x,y)* функцией $\tilde{f}(x,y)$. Предполагается, что *H* – тождественный оператор, т. е. искажения вносятся исключительно шумом.

Импульсный шум. Для импульсного (биполярного) шума процесс добавления шума состоит в том, что значение яркости каждой точки изображения с вероятностью $P_s = P_a + P_b \le 1$ заменяется на значение шума. При этом яркость пикселей определяется двумя положительными числами P_a и P_b . Яркость любого пикселя с вероятностью P_a заменяется на значение a, с вероятностью P_b – на значение b и с вероятностью $1–P_b–P_a$ остается неизменной. Плотность распределения вероятности может быть записана с использованием дельта функции Дирака в виде:

$$p(z) = P_a\delta(z-a) + P_b\delta(z-b) \qquad (1.3)$$

Если $b>a$, то пиксель с яркостью b выглядит как светлая точка на изображении. Пиксель с яркостью a выглядит, наоборот, как темная точка. Если одно из значений вероятности P_a или P_b равно нулю, то импульсный шум называется униполярным. Если ни одна из вероятностей не равна нулю, импульсный шум походит на крупицы соли и перца, рассыпанные по изображению. Поэтому, такой шум часто называют шумом типа «соль и перец». При оцифровке изображения обычно происходит ограничение значений яркости. Поэтому обычно полагают, что значения a и b равны минимальному и максимальному значениям, которые в принципе могут присутствовать в оцифрованном изображении. Для 8 – битовых изображений это означает, что a=0 (черное, перец), b=255 (белое, соль).

Оценивание параметров шума. В случае пространственного шума параметры плотности распределения вероятности часто определяют с помощью анализа простых тестовых изображений. Задача заключается в оценивании среднего значения и дисперсии по тестовым образцам, которые можно затем использовать для определения параметров уравнений. Пусть z_i – дискретная случайная величина, значениями которой являются уровни яркости изображения. Обозначим через $p(z_i)$, $i=1,2,...,L–1$ соответствующую

нормированную гистограмму, где L – число возможных значений яркости. Тогда число $p(z_i)$ приближает вероятность появления величины яркости z_i на изображении.

Иначе говоря, нормированную гистограмму можно рассматривать как приближение плотности распределения для яркости. Тогда среднее значение яркости определится по формуле:

$$m = \sum_{i=0}^{L-1} z_i p(z_i) \tag{1.4}$$

Дисперсия случайной величины z (яркости) определится по формуле:

$$D = \sum_{i=0}^{L-1} (z_i - m)^2 p(z_i) \tag{1.5}$$

Фильтр, основанный на вычислении среднего арифметического. Пусть S_{xy} обозначает прямоугольную окрестность размера M x N точки (x,y). Значение восстановленного изображения f в произвольной точке (x,y) представляет собой среднее арифметическое значений в точках искаженного изображения $g(x,y)$, принадлежащих окрестности S_{xy}:

$$\tilde{f}(x,y) = \frac{1}{MN} \sum_{(s,t)\in S_{xy}} g(s,t) \tag{1.6}$$

Среднегеометрический фильтр. Изображение, восстановленное с использованием среднегеометрического фильтра, задается выражением

$$\tilde{f}(x,y) = \left[\prod_{(s,t)\in S_{xy}} g(s,t) \right]^{1/MN} \tag{1.7}$$

Для применения среднегеометрического фильтра нужно к логарифму зашумленного изображения применить усредняющий фильтр, а затем применить функцию *exp(...)*. Таким образом нелинейная задача умножения сводится к линейной задаче сложения.

Медианные фильтры. Самым известным фильтром порядковых статистик в цифровой обработке изображений является медианный фильтр, который соответствует среднему элементу маски. Действие медианного фильтра состоит в замене значения в точке изображения на медиану значений яркости в окрестности этой точки

$$\tilde{f}(x,y) = \underset{(s,t)\in S_{xy}}{med}\{g(s,t)\} \tag{1.8}$$

Медианные фильтры особенно эффективны при наличии биполярного или униполярного импульсного шума.

Узкополосные фильтры пропускают, а не подавляют частоты в окрестностях некоторых своих центральных частот. Их передаточные функции получаются по формулам:

$$H_{np}(u,v) = 1 - H_{nr}(u,v) \tag{1.5}$$

где $H_{np}(u,v)$ – передаточная функция узкополосного (пропускающего) фильтра,) соответствующая режекторному фильтру $H_{nr}(u,v)$. Применяя к изображению с периодическим шумом передаточную функцию $H_{np}(u,v)$, будут отфильтрованы все детали изображения, оставив только составляющие шума.

Актуальность и цель работы. На сегодняшний день в республике, как и по всему миру, значительная часть графического материала оцифрована, однако, к сожалению, зачастую качество оцифровки, в силу разных факторов и причин, весьма низкое. В таких случаях становятся крайне востребованы приложения по автоматической цифровой обработке изображений с целью улучшения их качества.

Целью данного проекта является разработка программы обработки изображений "Восстановление поврежденных изображений". Данная программа должна быть бесплатной и предоставлять удобный интерфейс для выбора интересующего нас изображения в формате .bmp или .jpg, выбора параметров преобразования данного изображения и просмотра результатов его преобразования.

Для достижения поставленной цели необходимо решить задачи:

1) Изучить литературу и освоить алгоритмы, методы и особенности цифровой обработки изображений.
2) На основе полученных знаний разработать структуры данных для хранения в памяти входа, выхода, изображений и внутреннего состояния сущностей приложения.
3) Используя алгоритмы обработки изображений разработать программу «Восстановление поврежденных изображений».

2 Техническое задание

2.1 Описание исходных данных приложения

Данное приложение может быть использовано в научно-исследовательских целях; компаниями, предоставляющими фотоуслуги; медицинскими организациями; правоохранительными органами и т.д.

В системе должна обрабатываться и отображаться следующая информация:

1. Информация по текущим и ранее загруженным изображениям.
2. Наличие некоторого постоянного хранилища данных, в которую пользователь сможет добавлять или удалять изображения.
3. Просмотр характеристик изображения.
4. Искусственное зашумление изображения.
5. Просмотр исходного изображения и обработанного с помощью разных выбираемых методов.
6. Настройки приложения в виде выбора учитываемых шумов, алгоритмов и методов обработки изображения.

Исходными данными для приложения являются:

1. Изображения в формате BMP.
2. Изображения в формате JPEG.
3. Изображения в формате PNG.

2.2 Требования к пользовательским интерфейсам

Интерфейс должен предполагать стандартную системную цветовую палитру и разрабатываться под разрешение экрана 800x600 и более. Окна должны обладать системным меню с кнопкой закрытия.

Требования к окнам:

Окно входа в систему:

- строка для ввода логина, для ввода пароля;
- кнопка для скрытия пароля;
- кнопка «Войти», кнопка «Выйти».

Основное окно с главной панелью:

- PictureBox слева, в котором отображается загружаемое изображение;
- PictureBox справа, в котором отображается обработанное изображение;
- Кнопки:
    1) Обработать;
    2) Вывести характеристики;
    3) Изменить настройки;
    4) Искусственное зашумление;
    5) Выход.

Окно выбора файла (должно иметь стандартный Windows-вид).

Окно просмотра лога действий пользователя.

Окно просмотра ранее загруженных файлов.

Окно справки «Информация»

- логотип приложения;
- дата выпуска;
- информация о приложении.

Окно настроек:

- Выпадающие списки для выбора методов обработки изображения; при выборе метода отобразить краткую информацию по нему.
- Выпадающие списки для выбора типа шума изображения; при выборе шума отобразить краткую информацию по нему.

2.3 Требования к аппаратным и программным интерфейсам

Необходимо обеспечить программное взаимодействие системы с операционными системами Windows XP / Vista / 7 / 8 / 8.1 / 10.

Программа должна занимать не более 128 Мб оперативной памяти.

Модули программы должны занимать также не более 128 Мб памяти на жестком диске.

2.4 Требования к пользователям продукта

- Владение компьютером на уровне пользователя ОС Windows;
- Навыки работы с файловой системой ОС Windows.

2.5 Функции продукта

Основной функционал продукта:

1. Вход в систему.
2. Выход из системы.
3. Открыть изображение.
4. Восстановить изображение.
5. Сохранить восстановленное изображение.
6. Искусственно автоматически зашумить изображение.
7. Сохранить зашумленное изображение.
8. Просмотреть информацию по изображению.
9. Высчитать и отобразить характеристики изображения.
10. Изменить настройки приложения.
11. Выбрать метод обработки.
12. Выбрать тип шума.
13. Просмотреть характеристики шума.
14. Сохранить лог действий.

Сценарий «Вход в систему»:

1) Пользователь вводит логин и пароль.

2) Нажимает кнопку «Войти». Если пароль или логин не правильный, то система отображает окно с ошибкой «Неверный логин или пароль».

3) В случае успеха открывается главное окно с основной панелью.

4) Система логирует событие авторизации пользователя.

Сценарий «Выход из системы»:

1) По нажатию на кнопку «Выйти» в основном окне, система запоминает, какое изображение было открыто в левой панели главного окна. При повторном запуске программы система предлагает открыть послежнее изображение.

2) Система логирует время выхода пользователя из приложения.

Сценарий «Загрузить изображение»:

1) Запускается в основном окне с главными панелями по нажатию на кнопку «Загрузить изображение»;

2) Система отображает окно выбора изображения;

3) Если расширение не из списка BMP, JPG, PNG, система выдает ошибку;

4) Если расширение корректное, система подгружает изображение в левую панель и запоминает путь к нему.

5) Система логирует событие выбора и загрузки изображения пользователем.

Сценарий «Обработать изображение (восстановить)»:

1) Пользователь нажимает кнопку «Восстановить изображение».

2) Система проверяет, что загружено какое-либо изображение.

3) Если изображение не было подгружено, система выдает ошибку.

4) Система обрабатывает изображение и выводит результат в правую панель.

5) Система логирует событие обработки изображения.

Сценарий «Зашумить изображение»:

1) Пользователь нажимает кнопку «Зашумить изображение».

2) Система проверяет, что загружено какое-либо изображение.

3) Если изображение не было подгружено, система выдает ошибку.

4) Система зашумляет изображение и выводит результат в правую панель.

5) Система логирует событие зашумления изображения.

2.6 Ограничения

1) Должна использоваться кодировка UTF-8.

2) Продукт будет поддерживать исключительно английский язык пользовательского интерфейса.

3) Продукт не предусматривает автоматического перехода на платформы, не перечисленные в данном документе.

4) Скорость и качество обработки фото будет зависеть только от качества фото и производительности машины и степени нагрузки CPU (центрального процессора).

3 Обоснование выбора инструментальных средств для решения поставленной задачи

Для разработки приложения была выбрана технология **Microsoft .NET Framework 4.6**, обладающая обширным функционалом для графического представления данных и построения окон.

Для создания графического интерфейса пользователя был выбран интерфейс программирования приложений **Windows Forms**, являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде.

В качестве инструментальной среды разработки (IDE) была выбрана **Microsoft Visual Studio 2015 Community** и **язык C#**, который является языком разработки приложений для платформы Microsoft .NET Framework.

Код написан в рамках объектно-ориентированной парадигмы, т.к. C# является объектно-ориентированным языком, а каждое окно формы является экземпляром класса.

Для ведения истории разработки и контроля версий была выбрана распределённая система управления версиями **Git**. Программа является свободной и выпущена под лицензией GNU GPL версии 2.

Для автодокументирования кода выбран иструмент **SandCastle**.

Установщик приложения setup.msi написан с помощью популярного инструмента **InnoSetup**, обладающего развитым функционалом и широкими возможностями.

4 Разработка приложения «Восстановление поврежденных изображений»

4.1 Описание алгоритма обработки данных

Для организации разработки программы можно выделить следующие подзадачи:

1) перевод изображения в оттенки серого;
2) свёртка;
3) упрощение значений цвета.

Градации серого – цветовой режим изображений, который трансформирует 24-битное трехканальное цветное изображение в одноканальное 8-битное изображение путем вычисления средневзвешенного количества красного, зеленого и синего цветов. Коэффициенты, используемые для расчета свечения, выбраны из соображений коррекции люминисцентов цветного монитора.

Понятие свертки означает операцию вычисления нового значения выбранного пикселя, учитывая значения окружающих его пикселей. Для вычисления значения используется матрица, называемая ядром свертки. Ядро свертки является квадратной матрицей n×n, где n — нечетное, однако ничто не мешает сделать матрицу прямоугольной. Во время вычисления нового значения выбранного пикселя ядро свертки как бы «прикладывается» своим центром (именно тут важна нечетность размера матрицы) к данному пикселю. Окружающие пиксели также накрываются ядром. Далее высчитывается сумма, где слагаемыми являются произведения значений пикселей на значения ячейки ядра, накрывшей данный пиксель. Сумма делится на сумму всех элементов ядра свертки. Полученное значение как раз и является новым значением выбранного пикселя. Если применить свертку к каждому пикселю изображения, то в результате получится некий эффект, зависящий от выбранного ядра свертки [5].

4.2 Описание классов и интерфейсов проекта

Логика приложения (классы и соответствующие файлы) приведена в таблице 4.1.

Таблица 4.1 – Классы и файлы проекта

IAuthorizeService.cs	Интерфейс, который отвечает за вход в систему (авторизацию)
AuthorizeService.cs	Класс с конкретной реализацией функционала авторизации
ImageHelper.cs	Класс, который отвечает за изображение и его характеристики
INoise.cs	Интерфейс шума
GaussianNoise.cs	Класс с конкретной реализацией интерфейса шума в виде гауссового шума
ImpulseNoise.cs	Класс с конкретной реализацией интерфейса шума в виде импульсного шума
IRestorator.cs	Интерфейс, который отвечает за цифровую обработку изображения с целью его восстановления
MedianRestorator.cs	Класс с конкретной реализацией функционала восстановления поврежденного изображения в виде медианого фильтра
MovingAverageRestorator.cs	Класс с конкретной реализацией функционала восстановления поврежденного изображения в виде фильтра скользящего среднего
DeconvolutionRestorator.cs	Класс с конкретной реализацией функционала восстановления поврежденного изображения в виде деконволюции
Settings.cs	Статический класс с настройками приложения
ILogService	Интерфейс, который отвечает за логгирование действий пользователя
LogService	Класс с конкретной реализацией функционала логгирования действий пользователя

Так как в интерфейсе Windows Forms все окна и графические элементы формы являются отдельными классами, в таблице 4.2 и на рис.4.1–4.2 представлены файлы оконных форм и пользовательских элементов управления, созданных для восстановления поврежденных изображений.

Таблица 4.2 – Классы проекта, отвечающие за графический интерфейс

AuthorizationForm.cs	Окно входа в систему, необходим логин и пароль
MainForm.cs	Главное окно, в котором происходит вся основная работа с изображениями: загрузка, обработка, сохранение результатов
SettingsForm.cs	Окно, позволяющее задать настройки обрабокти изображения: выбрать тип шума и метод восстановления
HelpForm.cs	Окно справки по приложению: дата выпуска, право владения, автор, логотип
OpenFileDialog	Стандартное окно WinForms .NET для выбора и открытия файла
SaveFileDialog	Стандартное окно WinForms .NET для выбора и сохранения файла

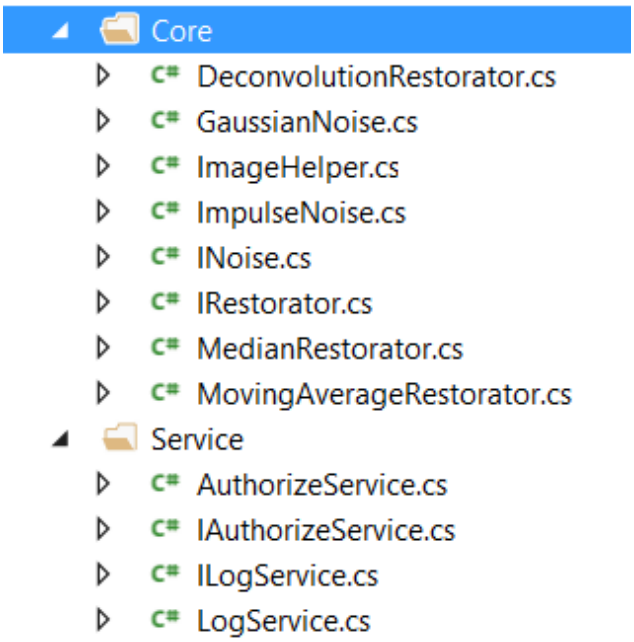

Рисунок 4.1 – Структура классов бизнес-логики приложения

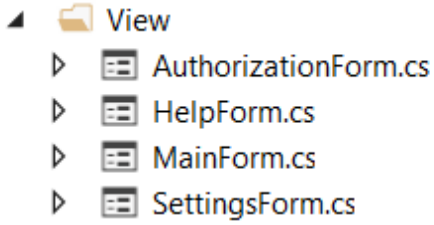

Рисунок 4.2 – Структура классов представления приложения

На рисунке 4.3 представлена общая структура проекта.

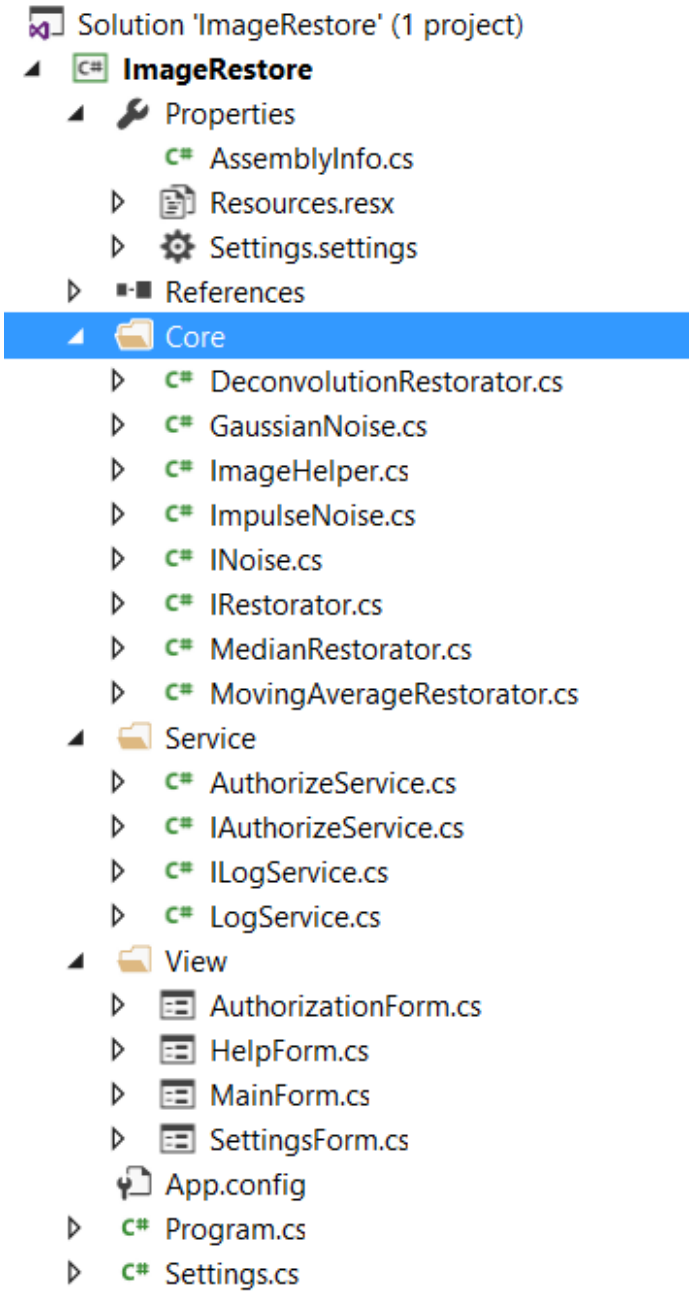

Рисунок 4.3 – Общая структура проекта

На рисунках 4.4–4.5 приведены фрагменты UML-диаграммы классов проекта ImageRestore.

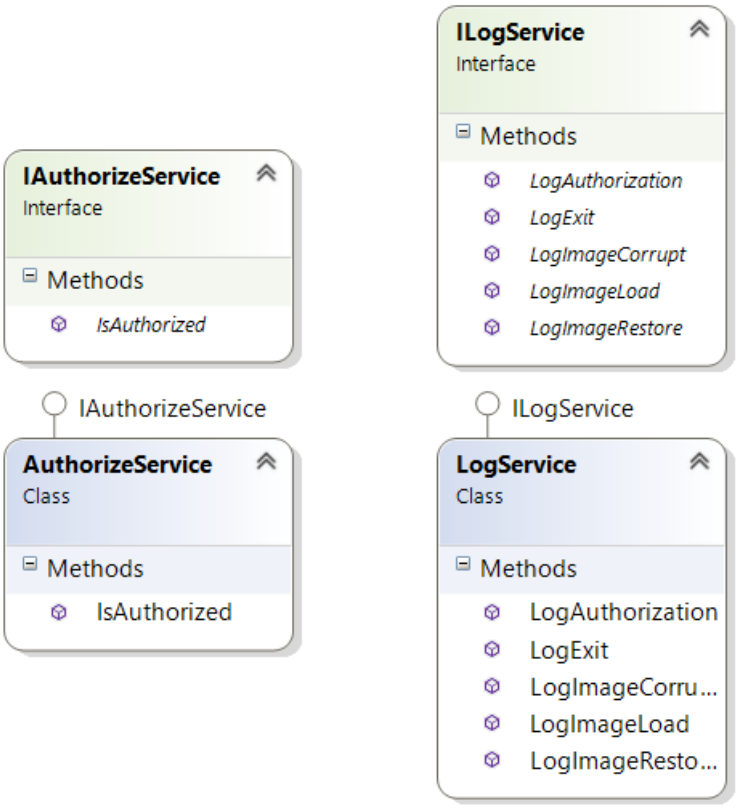

Рисунок 4.4 – UML-диаграмма классов приложения (Services)

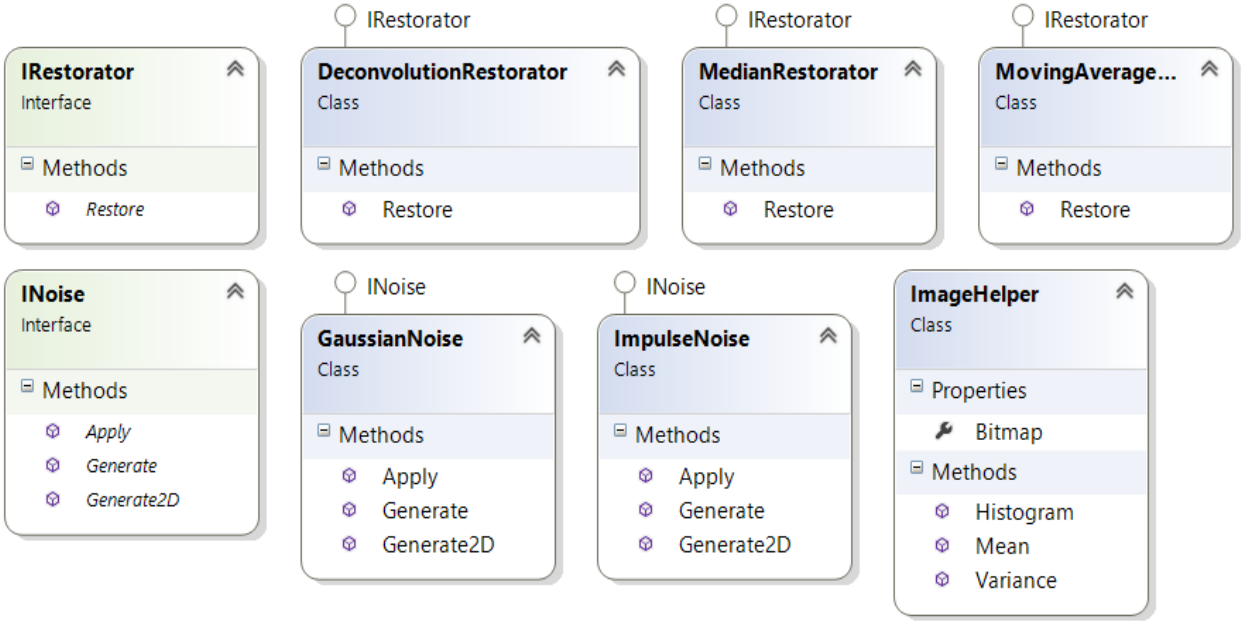

Рисунок 4.4 – UML-диаграмма классов приложения (Core)

Реализации основных методов классов описаны в приложении Б.

Входные данные хранятся в графических изображениях с форматом BMP, JPG, PNG, а также файлах лога с расширением *.log.

Выходными данными являются графические изображения в формате BMP, JPG, PNG, а также файлы лога с расширением *.log.

Ниже в таблицах рассмотрен каждый класс более детально.

Таблица 4.3 – Методы, используемые в классе ImageHelper

Название метода	Реализация
public Bitmap Bitmap { get; set; }	Геттер и сеттер для самой картинки в виде объекта .NET класса Bitmap
public int[] Histogram (int binCount = 256)	Посчитать и вернуть гистограмму пикселей
public double Mean()	Посчитать выборочное среднее
public double Variance()	Посчитать дисперсию
public byte GrayScale()	Получает изображение в оттенках серого

Таблица 4.4 – Методы, используемые в интерфейсе INoise и классах-наследниках

Название метода	Реализация
double[] Generate(int size);	Сгенерировать одномерный шум размера size элементов
double[][] Generate2D (int height, int width);	Сгенерировать двумерный шум размера height * width элементов
ImageHelper Apply (ImageHelper image);	Применить шум к изображению image и получить на выходе зашумленное изображение

Таблица 4.5 – Методы, используемые в классе IRestorator и классах-наследниках

Название метода	Реализация
ImageHelper Restore (ImageHelper image);	Интерфейс и данный метод реализует паттерн «Стратегия»: метод служит для реализации алгоритма цифровой обработки изображения: восстановления после повреждения и зашумления

Таблица 4.6– Методы, используемые в интерфейсе ILogService и классе LogService

Название метода	Реализация
void LogAuthorization(DateTime timeStamp, string login);	Метод для логирования события входа пользователя в систему. Для метода нужна дата/время события и логин пользователя
void LogImageLoad (DateTime timeStamp, string filename);	Метод для логирования события загрузки изображения в систему. Для метода нужна дата/время события и полное имя файла изображения
void LogImageRestore (DateTime timeStamp, string filename);	Метод для логирования события восстановления изображения. Для метода нужна дата/время события и полное имя файла изображения
void LogImageCorrupt (DateTime timeStamp, string filename);	Метод для логирования события искусственного зашумления изображения. Для метода нужна дата/время события и полное имя файла изображения
void LogExit (DateTime timeStamp, string login);	Метод для логирования события входа пользователя в систему. Для метода нужна дата/время события и логин пользователя

Таблица 4.7 – Методы, используемые в классе IAuthorizeService

Название метода	Реализация
bool IsAuthorized (string login, string password);	Метод для произведения авторизации: Возвращает true, если логин и пароль корректны, false – иначе

5 Тестирование программного продукта

5.1 Требования к продукту и установка

Аппаратные требования для работы приложения:

- Процессор с тактовой частотой 1.0 ГГц.
- Оперативная память 512 Мб и более.
- Видеокарта с объёмом памяти 64 Мб и выше.
- Монитор 800x600 или с более высоким разрешением.

Программные требования к приложению:

- Операционная система —Windows XP / Vista / 7 / 8 / 10;
- Microsoft .NET Framework 4.6

Приложение на данный момент является переносным отдельным исполняемым exe-файлом. Для развертывания на пользовательской машине должен быть установлен .NET Framework 4.6.

5.2 Описание контрольных примеров

После запуска приложения появляется окно авторизации:

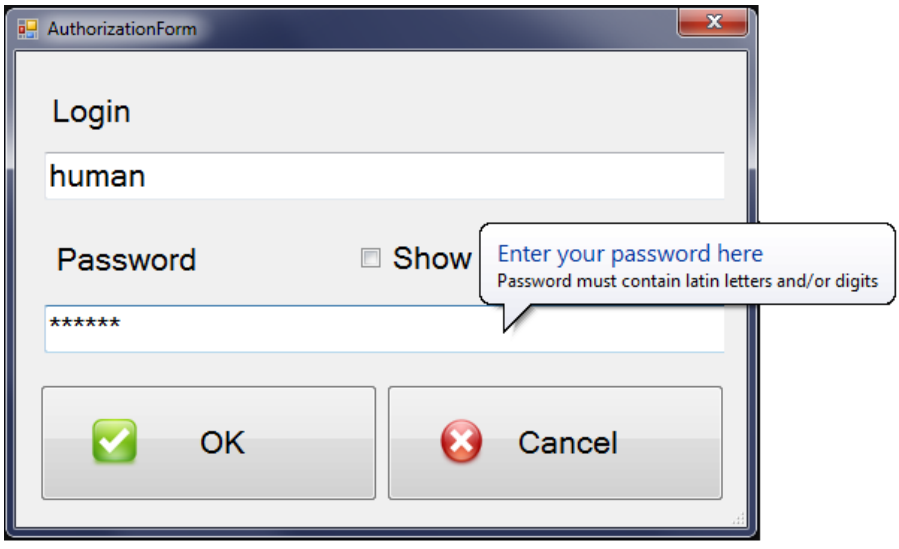

Рисунок 5.1 – Окно входа в систему

Введем в него логин «human» и пароль «12345». Система в тестовом режиме реагирует только на данный логин и пароль. Если введем неверный логин и пароль, программа выдаст сообщение об ошибке.

После ввода корректных данных появляется главное окно приложения:

Зайдем в пункт меню Settings и посмотрим, какие настройки выставлены по умолчанию. Тип фильтра – медианный; тип шума - импульсный. Можем поменять настройки, но пока оставим по умолчанию.

Откроем какой-нибудь графический файл. Для этого выберем пункт меню «File» и подпункт «Open...». В появившемся окне выбора файла укажем интересующий нас файл и нажмем ОК. Файл загрузится в левую панель окна:

Рисунок 5.2 – Вид главного окна приложения после загрузки файла

Выберем пункт меню «Process» и затем подпункт «Restore». Через некоторое время программа произведет необходимые операции и обновит правую панель окна:

Рисунок 5.3 – Вид главного окна приложения после обработки файла

ЗАКЛЮЧЕНИЕ

В результате проделанной работы была создана программа цифровой обработки изображений "Восстановление поврежденных изображений", предоставляющая удобный интерфейс для выбора изображения в формате bmp или jpg, выбора параметров преобразования данного изображения и просмотра результатов его преобразования.

В ходе работы удалось решить следующие задачи:

- изучена литература и освоены алгоритмы, методы и особенности цифровой обработки изображений;
- на основе полученных знаний разработаны структуры данных для хранения в памяти входа, выхода, изображений и внутреннего состояния сущностей приложения;
- на основе моделей и методов обработки изображений разработана программа «Восстановление поврежденных изображений».

Приложение может использоваться в научно-исследовательских целях; компаниями, предоставляющими фотоуслуги; медицинскими организациями; правоохранительными органами, а также всеми пользователями, интересующимися программированием. Дальнейшее развитие приложения связано с расширением его функционала, увеличением опций, а также подключением базы данных для организации персистентности информации системы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Гонсалес Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. – М.: Техносфера, 2005. – 1121 с.
2. Гонсалес Р. Цифровая обработка изображений в среде Matlab / Р. Гонсалес, Р. Вудс С. Эддинс. – М.: Техносфера, 2006. – 712 с.
3. П. Доля. Восстановление изображений с шумом [Электронный ресурс]. – URL: http://geometry.karazin.ua/resources/documents/20141222211648_186ab376c3701.pdf (дата обращения 5.05.2017).
4. Фисенко В.Т. Компьютерная обработка и распознавание изображений: учеб. пособие / В.Т. Фисенко, Т.Ю. Фисенко. – СПб.: СПбГУ ИТМО, 2008. – 192 с.
5. Яне Б. Цифровая обработка изображений / Б. Яне. – М.: Техносфера, 2007. – 584с.
6. Фильтрация изображений методом свертки: [Электронный ресурс]. – URL: http://habrahabr.ru/post/62738 (дата обращения 5.05.2017).
7. Nicolas Devillard. Fast Median Search. Implementation [Электронный ресурс]. – URL: http://ndevilla.free.fr/median/median.pdf (дата обращения 5.05.2017).
8. Нейгел К. C#5.0 и платформа .NET 4.5 для профессионалов / К. Нейгел, Б. Ивьен., Дж. Глини. – М.: Издательский дом «Вильямс», 2014. – 1440 с.

ПРИЛОЖЕНИЕ А

Экранные формы

Рисунок А.1 – Главное окно приложения

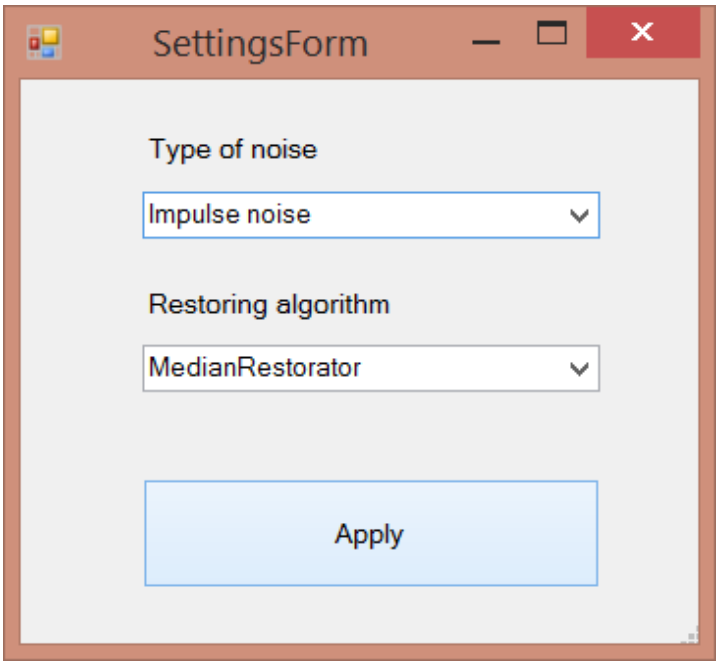

Рисунок А.2 – Окно настроек приложения

ПРИЛОЖЕНИЕ Б

Фрагменты листинга

Листинг Б.1 – Коды интерфейсов

```
namespace ImageRestore.Core
{
    interface INoise
    {
        /// <summary>
        /// Generate 1D noise of size
        /// </summary>
        /// <returns>1D noise</returns>
        double[] Generate(int size);

        /// <summary>
        /// Generate 2D noise of size (height, width)
        /// </summary>
        /// <returns>2D noise</returns>
        double[][] Generate2D(int height, int width);

        /// <summary>
        /// Corrupt an image with the noise
        /// </summary>
        /// <param name="image">Image to be changed</param>
        /// <returns>Corrupted image</returns>
        ImageHelper Apply(ImageHelper image);
    }
}
```

```
namespace ImageRestore.Core
{
    /// <summary>
    /// Simple strategy interface for image restoration
    /// </summary>
    interface IRestorator
    {
        /// <summary>
        /// Restoring routine
        /// </summary>
        /// <param name="image">Input image</param>
        /// <returns>Restored image</returns>
        ImageHelper Restore(ImageHelper image);
    }
}
```

```
namespace ImageRestore.Service
{
    interface IAuthorizeService
    {
        bool IsAuthorized(string login, string password);
    }
}
```

Листинг Б.2 – Код класса MedianRestorator

```
public class MedianRestorator : IRestorator
    {
        private int size = 3;

        public int Size
        {
            get { return size; }
            set { size = Math.Max( 3, Math.Min( 25, value | 1 ) ); }
        }

        public MedianRestorator()
        {
        }

        public MedianRestorator(int size) : this( )
        {
            Size = size;
        }

        public unsafe void Restore(   UnmanagedImage source,
                                UnmanagedImage destination,
                                Rectangle rect)
        {
            int pixelSize =
                      Image.GetPixelFormatSize(source.PixelFormat) / 8;

            int startX  = rect.Left;
            int startY  = rect.Top;
            int stopX   = startX + rect.Width;
            int stopY   = startY + rect.Height;

            int srcStride = source.Stride;
            int dstStride = destination.Stride;
            int srcOffset = srcStride - rect.Width * pixelSize;
            int dstOffset = dstStride - rect.Width * pixelSize;

            int i, j, t;
            int radius = size >> 1;
            int c;

            byte[] r = new byte[size * size];
            byte[] g = new byte[size * size];
            byte[] b = new byte[size * size];

            byte* src = (byte*) source.ImageData.ToPointer( );
            byte* dst = (byte*) destination.ImageData.ToPointer( );
            byte* p;

            src += ( startY * srcStride + startX * pixelSize );
            dst += ( startY * dstStride + startX * pixelSize );
```

Листинг Б.2, продолжение

```
if ( destination.PixelFormat == PixelFormat.Format8bppIndexed )
       {
           for ( int y = startY; y < stopY; y++ )
           {
               for (int x = startX; x < stopX; x++, src++, dst++)
               {
                   c = 0;

                   for ( i = -radius; i <= radius; i++ )
                   {
                       t = y + i;

                       if ( t < startY )
                           continue;
                       if ( t >= stopY )
                           break;

                       for ( j = -radius; j <= radius; j++ )
                       {
                           t = x + j;

                           // skip column
                           if ( t < startX )
                               continue;

                           if ( t < stopX )
                           {
                               g[c++] = src[i * srcStride + j];
                           }
                       }
                   }

                   Array.Sort( g, 0, c );

                   *dst = g[c >> 1];
               }
               src += srcOffset;
               dst += dstOffset;
           }
       }
       else
       {
           for ( int y = startY; y < stopY; y++ )
           {
               for ( int x = startX; x < stopX; x++,
                           src += pixelSize, dst += pixelSize )
               {
                   c = 0;


                    for ( i = -radius; i <= radius; i++ )
                   {
                       t = y + i;
```

Листинг Б.2, продолжение

```
                            if ( t < startY )
                                continue;
                            // break
                            if ( t >= stopY )
                                break;

                            // for each kernel column
                            for ( j = -radius; j <= radius; j++ )
                            {
                                t = x + j;

                                // skip column
                                if ( t < startX )
                                    continue;

                                if ( t < stopX )
                                {
                                    p = &src[i * srcStride + j *
                                                        pixelSize];
                                    r[c] = p[RGB.R];
                                    g[c] = p[RGB.G];
                                    b[c] = p[RGB.B];
                                    c++;
                                }
                            }
                        }

                        Array.Sort( r, 0, c );
                        Array.Sort( g, 0, c );
                        Array.Sort( b, 0, c );

                        t = c >> 1;
                        dst[RGB.R] = r[t];
                        dst[RGB.G] = g[t];
                        dst[RGB.B] = b[t];
                    }
                    src += srcOffset;
                    dst += dstOffset;
                }
            }
        }
    }
```