МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДНР

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ»

Факультет Физико-технический
Кафедра Компьютерных технологий (КТ)

Зав. кафедрой КТ
_____ Т. В. Ермоленко
(подпись)
«___» _____ 2017 г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе бакалавра 2 курса
на тему:

РАЗРАБОТКА ПРОГРАММЫ ВЫДЕЛЕНИЯ КОНТУРОВ ОБЪЕКТОВ С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ КОМПЬЮТЕРНОГО ЗРЕНИЯ

Автор работы _____ И.И. Иванов
подпись

Направление 09.03.01 Информатика и вычислительная техника

Руководитель работы _____ ст. преп. В.Н. Котенко
подпись

Консультанты по разделам:

Сетевые технологии _____ ст. преподаватель А.Е. Гукай
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 Иванова И.И.

Тема курсовой работы: Разработка программы выделения контуров объектов с использованием библиотеки компьютерного зрения

Краткая постановка задачи: 1. Изучить и проанализировать предметную область программной обработки изображения. 2. Ознакомиться с программными продуктами для создания приложений, выполнить их установку. 3. Разработать техническое задание на создание приложения. 4. Разработать алгоритмы приложения. 5. Разработать средствами C# и Windows Forms приложение, которое позволяет выполнять выделение контуров обьектов на изображении. 8. Протестировать и описать программное обеспечение; 9. Оформить отчёт.

Исходные данные: 1. Документация об программной обработке изображения. 2. Документация по Visual Studio. 3. Документация по языку программирования C#. 4. Документация по Windows Forms.

Ожидаемые результаты: Десктоп приложение для выделения контуров обьектов на изображениях.

Календарный план работы:

Даты консультаций	Этапы выполнения работы	Отметки о выполнении
30.02.2017	Постановка задачи и обсуждение литературы	выполнено
05.03.2017	Предварительное утверждение содержания отчёта	выполнено
13.03.2017	Утверждение проекта, алгоритмов, методов, технологий	выполнено
26.03.2017	Ход реализации проекта	выполнено
28.03.2017	Обсуждения организации тестирования программы	выполнено
03.04.2017	Демонстрация программного продукта руководителю	выполнено
11.04.2017	Оформление отчёта	выполнено
17.04.2017	Предоставление отчёта руководителю	выполнено

Дата выдачи задания 30.01.2017 года

Студент _____ инициалы, фамилия

Руководитель _____ В.Н. Котенко

АННОТАЦИЯ

Отчёт о курсовой работе: 46 с., 15 рис., 3 табл., 2 приложения, 6 источников.

Объект исследования – программная обработка изображений.

Предмет исследования – выделение контуров изображения, работа с фильтрами.

Цель работы – разработать приложение для выделения контуров объектов с использованием библиотеки компьютерного зрения.

Метод исследования – анализ возможностей языка C# для создания приложений и работы с изображениями.

В курсовой работе было разработано приложение для выделения контуров объектов с использованием библиотеки компьютерного зрения. Данное приложение может быть использовано в сфере образования для демонстрации разнообразных алгоритмов для выделения контуров обьектов, в сфере распознавания текста и обьектов, а также в областях, связанных с обработкой изображений.

Дальнейшее развитие системы связанно с расширением функционала, добавлением новых фильтров и персонализацией.

ОБРАБОТКА ИЗОБРАЖЕНИЯ, ФИЛЬТРЫ, КОМПЬЮТЕРНОЕ ЗРЕНИЕ, C#, WINDOWS FORM, VISUAL STUDIO

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Автоматическая обработка визуальной информации является одним из важнейших направлений в области искусственного интеллекта. Интерес к проблемам компьютерной обработки определяется расширением возможностей как самих компьютерных систем, так и разработкой новых технологий обработки, анализа и идентификации различных видов изображений.

Одной из наиболее сложных проблем обработки визуальной информации является выделение контуры объектов, так как контура — это самые информативные структурные элементы объектов. Тем не менее контуры, которые выделяются на слабоконтрастных размытых изображениях известными методами, имеют такие изъяны, как разрывы, отсутствие контурных линий или наличие ошибочных, которые не отвечают исследуемому объекту. Большинство получаемых изображений являются слабоконтрастными, имеют неравномерный фон, а также содержат различного рода шумы. Поэтому для анализа такой информации необходимо обеспечить высокое визуальное качество и эффективность предварительной обработки исследуемого изображения, которое может быть получено с помощью современных методов выделения контуров и границ. Это позволит улучшить решения большого количества.

В рамках курсовой работы будет разработано приложение для определения контуров обьектов на изображении, из чего можно выделить частные задачи для достижения данной цели:

1. Проанализировать основные методы выделения контуров обьектов на изображении
2. Изучить технологии: .NET, Windows Forms, C#, Visual Studio 2015.
3. Разработать структуру программы, разработать функционал программы, соответствующий техническому заданию.
4. Разработать интерфейс и логическую составляющую приложения.

1 Анализ предметной области

1.1 Состояние вопроса

Автоматическая обработка визуальной информации является одним из важнейших направлений в области искусственного интеллекта. Интерес к проблемам компьютерной обработки определяется расширением возможностей как самих компьютерных систем, так и разработкой новых технологий обработки, анализа и идентификации различных видов изображений. При этом для создания эффективных технологий разрабатываемые методы и алгоритмы должны удовлетворять ряду требований по быстродействия и точности. Каждый алгоритм специализируется на своем типе изображения. Поэтому в системах технического зрения необходимо сочетание нескольких методов, которые решают одну и ту же задачу различными способами, обеспечивая при этом необходимые показатели по быстродействию и достоверности.

Одной из наиболее сложных проблем обработки визуальной информации является выделение контуры объектов, так как контура — это самые информативные структурные элементы объектов. Тем не менее контуры, которые выделяются на слабоконтрастных размытых изображениях известными методами, имеют такие изъяны, как разрывы, отсутствие контурных линий или наличие ошибочных, которые не отвечают исследуемому объекту.

Большинство получаемых изображений являются слабоконтрастными, имеют неравномерный фон, а также содержат различного рода шумы. Поэтому для анализа такой информации необходимо обеспечить высокое визуальное качество и эффективность предварительной обработки исследуемого изображения, которое может быть получено с помощью современных методов выделения контуров и границ. Это позволит улучшить решения большого количества задач.

1.2 Описание предметной области

Обнаружение границ — фундаментальный инструмент для сегментации изображения. Такие алгоритмы преобразуют входное изображение в изображение с контурами объектов, преимущественно в серых тонах. В обработке изображений, особенно в системах компьютерного зрения, с помощью выделения контура рассматривают важные изменения уровня яркости на изображении, физические и геометрические параметры объекта на сцене. Это фундаментальный процесс, который обрисовывает в общих чертах объекты, получая тем самым некоторые знания об изображении. Обнаружение границ является самым популярным подходом для обнаружения значительных неоднородностей.

Границы объектов на изображении в значительной степени уменьшают количество данных, которые необходимо обработать, и в то же время сохраняет важную информацию об объектах на изображении, их форму, размер, количество. Главной особенностью техники обнаружения границ является возможность извлечь точную линию с хорошей ориентацией.

Граница является местным изменением яркости на изображении. Они, как правило, проходят по краю между двумя областями. Функции их получения используются передовыми алгоритмами компьютерного зрения и таких областях, как медицинская обработка изображений, биометрия и тому подобные. Обнаружение границ — активная область исследований, так как он облегчает высокоуровневый анализ изображений. На полутоновых изображениях существует три вида разрывов: точка, линия и граница. Для обнаружения всех трех видов неоднородностей могут быть использованы пространственные маски.

В технической литературе приведено и описано большое количество алгоритмов выделения контуров и границ. В данной работе рассмотрены наиболее популярные методы. К ним относятся: оператор Робертса, Собеля, Превитта, Кирша, Лапласса.

1.3 Актуальность и цель работы

Выделение контура объекта на изображении является одной из актуальных задач в цифровой обработке сигнала. Психологические исследования выявили, что с точки зрения распознавания и анализа объектов на изображении наиболее информативным является не значение яркости объектов, а характеристики их границ — контуров.

При анализе изображений и распознавании объектов, присутствующих на нем, весомую долю принимают на себя методы и алгоритмы выделения контуров, так как они значительно упрощают работу с изображением и/или цифровым рядом. Но большинство существующих на сегодняшний день алгоритмов не может предоставить достаточно хорошую точность выделения контура объектов, так как всегда присутствуют разрывы и ложные границы. Это в значительной степени усложняет дальнейшую работу над изображением. Данная работа нацелена на разработку алгоритма, позволяющего максимально улучшить выделение границ изображения путем комбинации и модификации существующих методов обработки изображений.

На сегодняшний день существует огромное множество методов выделения контуров. Эти методы эффективны для обработки изображений с низким уровнем шума. Таким образом, на сегодняшний день остается актуальной разработка методов выделения объектов на слабоконтрастных изображениях.

2 Техническое задание

2.1 Описание области применения и исходных данных приложения

Данное приложение предназначено для обработки изображений с целью выделения контуров обьектов на нем. Его можно использовать в сферах, оно может использоваться для:

- демонстрации существующих алгоритмов выделения контуров обьектов на изображении
- распознавания текста и обьектов
- изучения характеристик существующих алгоритмов выделения контуров обьектов на изображении

Исходными данными для приложения являются цифровые изображения форматов: jpg, bmp, png, jpeg.

2.2 Требования к пользовательским интерфейсам

Интерфейс программы должен быть не особо сложным, приятным на вид и быть интуитивно понятным, чего можно достигнуть, использовав стандартные элементы управления Windows. Среди элементов интерфейса обязательно должны быть стандартные кнопки закрытия, свертывания и развертывания приложения, из чего следует, что окно приложения должно быть масштабируемое, при этом необходим

Приложение должно содержать всего одно окно, содержащее весь функционал. Для облегчения взаимодействия программы с пользователем следует визуально разбить интерфейс на части, каждая из которых будет иметь свой функционал и предназначение.

На рисунке 1 изображено позиционирование логических блоков интерфейса, которое выполняет все вышеперечисленные особенности и требования.

Рисунок 1 – Схема пользовательского интерфейса

Каждый из логических блоков интерфейса имеет свои элементы интерфейса. На рисунке 1 отображены следующие блоки интерфейса:

- Блок выбора изображения
- Блок настроек обработки
- Блок выбора метода обработки
- Блок с выводом результата

Блок выбора изображения должен иметь кнопку для вызова файлового диалога и элемент для отображения выбранной картинки, при этом, после выбора картинки по мимо ее изображения должен выводится и ее полный путь.

В блоке для выбора настроек обработки должно быть два элемента checkbox, с помощью которых можно установить будет ли результирующее изображение размыто или переведено в черно-белые цвета.

Блок выбора метода обработки должен содержать кнопки для запуска обработки. Каждой кнопке должен идти в соответствие определенный метод выведения контуров.

Последний блок – блок вывода результатов имеет единственную цель и для него будет достаточно всего одного элемента интерфейса для вывода результирующего изображения. Особенностью данного блока должно стать то, что при масштабировании окна блок с картинкой должен соответственно изменять размеры без деформации (сохранять пропорции высоты и ширины), а при выводе элемент, выводящий результат обработки, может содержать пустые места, но ни в коем случае не обрезать результирующее изображение.

При возникновении ошибок при использовании программы необходимо предоставлять пользователю информацию об ошибке и ее причине с помощью модального окна.

2.3 Требования к аппаратным и программным интерфейсам

Для установки и работы программы необходимо иметь вычислительную систему следующей базовой конфигурации:

- процессор: 1.8 ГГц
- оперативная память: от 512 мб
- видеокарта от 256 мб памяти
- более 50 мб памяти на диске для установки

Необходимо обеспечить программное взаимодействие системы с операционными системами Windows XP/7/8/8.1/10 и необходимо наличие на компьютере установленного Microsoft .NET Framework 4.6.

2.4 Требования к пользователям продукта

Пользователи программы должны:

- иметь базовые навыки работы с компьютером
- понимать подсказки программы
- уметь изучать справочную документацию

2.5 Функции продукта

Пользователи внутри приложения не разделены на классы и всех есть полный доступ к функциям приложения, из которых можно выделить:

- загрузка изображения в программу
- установка настроек обработки изображения (возможность сгладить или сделать изображение черно-белым)
- выбор метода обработки (Кирш, Собель, Превитт, Лапласс)
- масштабирование результирующего изображения

2.6 Ограничения

- продукт будет поддерживать только русский язык пользовательского интерфейса
- продукт не предусматривает автоматического перехода на платформы, не перечисленные в данном документе
- скорость работы приложения будет зависеть только от производительности машины и степени нагрузки ЦП
- не будет возможности сохранить полученный результат

2.7 Сценарии использования

Пользователь запускает приложение.
Сценарий 1: пользователь запускает обработку изображения. Появляется модальное окно с предупреждением, что исходное изображение не выбрано.
Сценарий 2: пользователь нажимает кнопку выбора изображения, открывается файловый диалог, в котором пользователь выбирает изображение.
Сценарий 2.1: пользователь выбирает настройки и запускает один из методов обработки. В элементе с результатом появляется отредактированное изображение.

3 Обоснование выбора инструментальных средств для реализации поставленной задачи

3.1 Язык программирования C#

C# — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework и впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников — языков C++, Pascal, Модула, Smalltalk и, в особенности, Java — C#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественное наследование интерфейсов).

3.2 .NET Framework

.NET Framework — программная платформа, выпущенная компанией Microsoft в 2002 году. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), которая подходит для разных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду.

Программа для .NET Framework, написанная на любом поддерживаемом языке программирования, сначала переводится компилятором в единый для .NET промежуточный байт-код Common Intermediate Language (CIL). В терминах .NET получается сборка, англ. assembly. Затем код либо исполняется виртуальной машиной Common Language Runtime (CLR), либо транслируется утилитой NGen.exe в исполняемый код для конкретного целевого процессора. Данные процессы видны на рисунке 2.

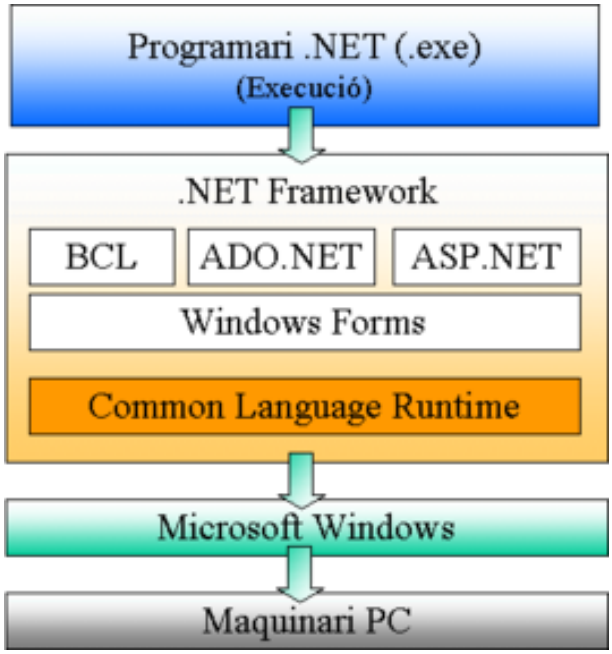

Рисунок 2 – Выполнение программы в .NET

Использование виртуальной машины предпочтительно, так как избавляет разработчиков от необходимости заботиться об особенностях аппаратной части. В случае использования виртуальной машины CLR встроенный в неё JIT-компилятор «на лету» (just in time) преобразует промежуточный байт-код в машинные коды нужного процессора. Современная технология динамической компиляции позволяет достигнуть высокого уровня быстродействия. Виртуальная машина CLR также сама заботится о базовой безопасности, управлении памятью и системе исключений, избавляя разработчика от части работы.

3.3 Windows Forms

Windows Forms — интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и 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. Причём управляемый код — классы, реализующие API для Windows Forms, не зависят от языка разработки. То есть программист одинаково может использовать Windows Forms как при написании ПО на C#, C++, так и на VB.Net, J# и др.

3.4 Visual Studio

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения (рисунок 3), веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности

практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

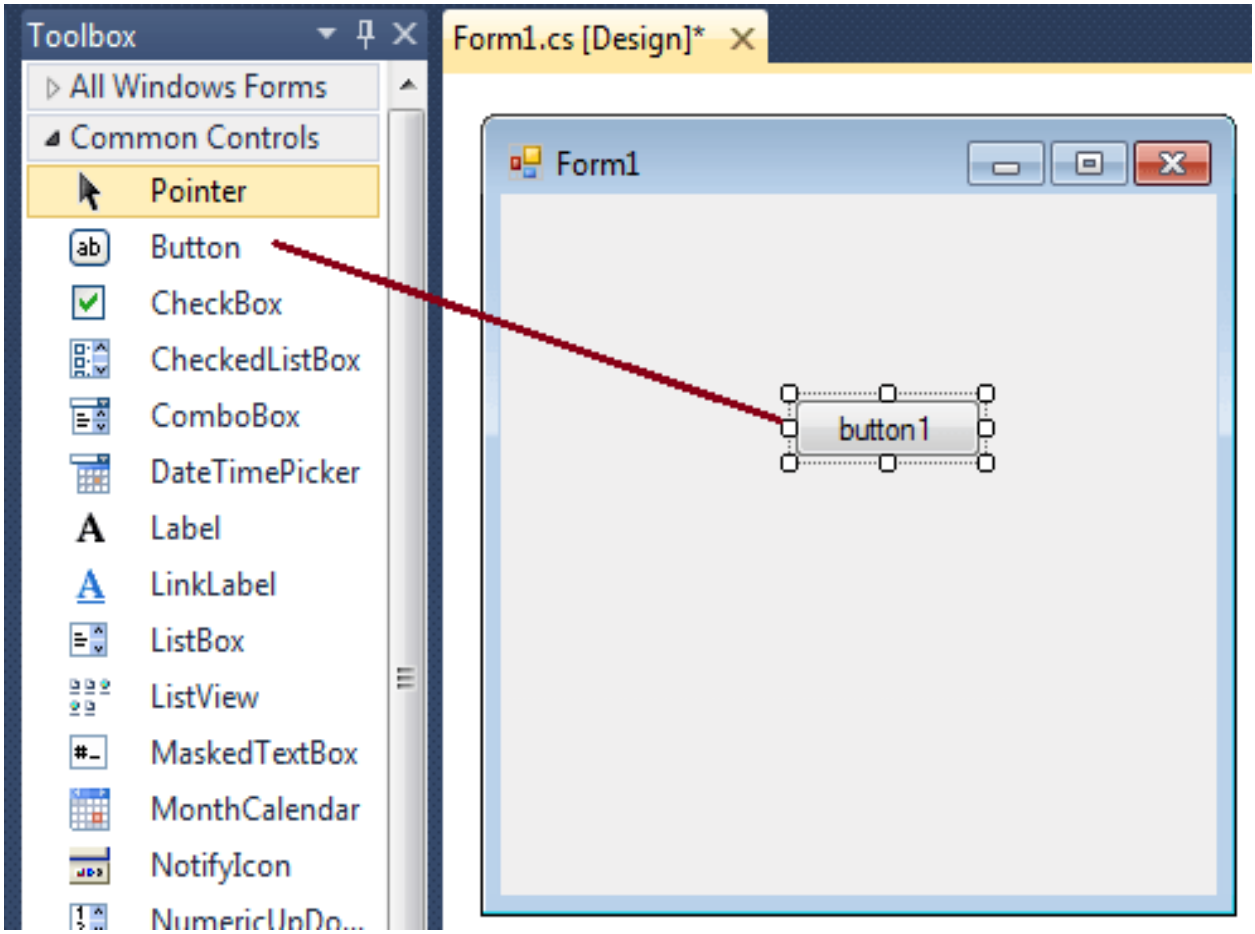

Рисунок 3 – Графический редактор форм

Данный набор библиотек поможет создать понятный для пользователя интерфейс, так как в Windows Form элементы, использованные по умолчанию уже знакомы всем пользователям Windows. Также Windows Form ускорит разработку, не отвлекаясь на разработку графических элементов интерфейса.

3.5 Visual Studio Installer Projects

Для создания инсталляционного файла в данной КР используется инструмент Visual Studio Installer Projects, который позволяет создавать инсталляционные пакеты .msi для Windows Installer.

Windows Installer (установщик Windows) — подсистема Microsoft Windows, обеспечивающая установку программ. Является компонентом Windows, начиная с Windows 2000; может устанавливаться и на более ранние версии Windows. Вся необходимая для установки информация (иногда и вместе с устанавливаемыми файлами) содержится в установочных пакетах (installation packages), имеющих расширение .msi. Файл .msi представляет собой составной документ OLE (OLE compound document — в том же формате-контейнере хранятся документы Microsoft Word, Excel и т. д.), в котором содержится небольшая реляционная база данных — набор из нескольких десятков взаимосвязанных таблиц, содержащих различную информацию о продукте и процессе установки. При этом все строковые данные в базе хранятся вместе в отдельном потоке документа, а в таблицах базы на них имеются ссылки; таким образом избегают дублирования строк, что значительно уменьшает размер базы. Кроме базы, структура файла .msi предусматривает помещение туда пользовательских сценариев и вспомогательных DLL, если таковые требуются для установки, а также самих устанавливаемых файлов, запакованных в формате .cab. Файлы можно размещать и отдельно от пакета, в запакованном или распакованном виде (с сохранением структуры каталогов).

4 Разработка ПО

4.1 Входные и выходные данные

Входными данными для приложения являются изображения разнообразных форматов:

- PNG
- JPEGs
- Bitmaps
- GIFs
- TIFF
- ICO

Однако внутри приложения все изображения преобразовываются к формату BMP. Это формат хранения растровых изображений, разработанный компанией Microsoft. С форматом BMP работает огромное количество программ, так как его поддержка интегрирована в операционные системы Windows и OS/2. Кроме того, данные этого формата включаются в двоичные файлы ресурсов RES и в PE-файлы.

В большинстве случаев пиксели изображения формата Bitmaps хранятся в виде относительно простого двумерного массива. Для битностей 4 и 8 доступно RLE-кодирование, которое может уменьшить их размер. Формат BMP также поддерживает встраивание данных в форматах JPEG и PNG. Но последнее скорее больше предназначено не для компактного хранения, а для обхода ограничений архитектуры GDI, которая не предусматривает прямую работу с изображениями отличных от BMP форматов.

Результатом работы приложения являются изображения с контурами обьектов исходного изображения. При сохранении изображения можно выбрать форматы: bmp, jpeg, png.

4.2 Структура приложения

4.2.1 Файловая структура приложения

Файловая структура проекта приложения изображена на рисунке 4.

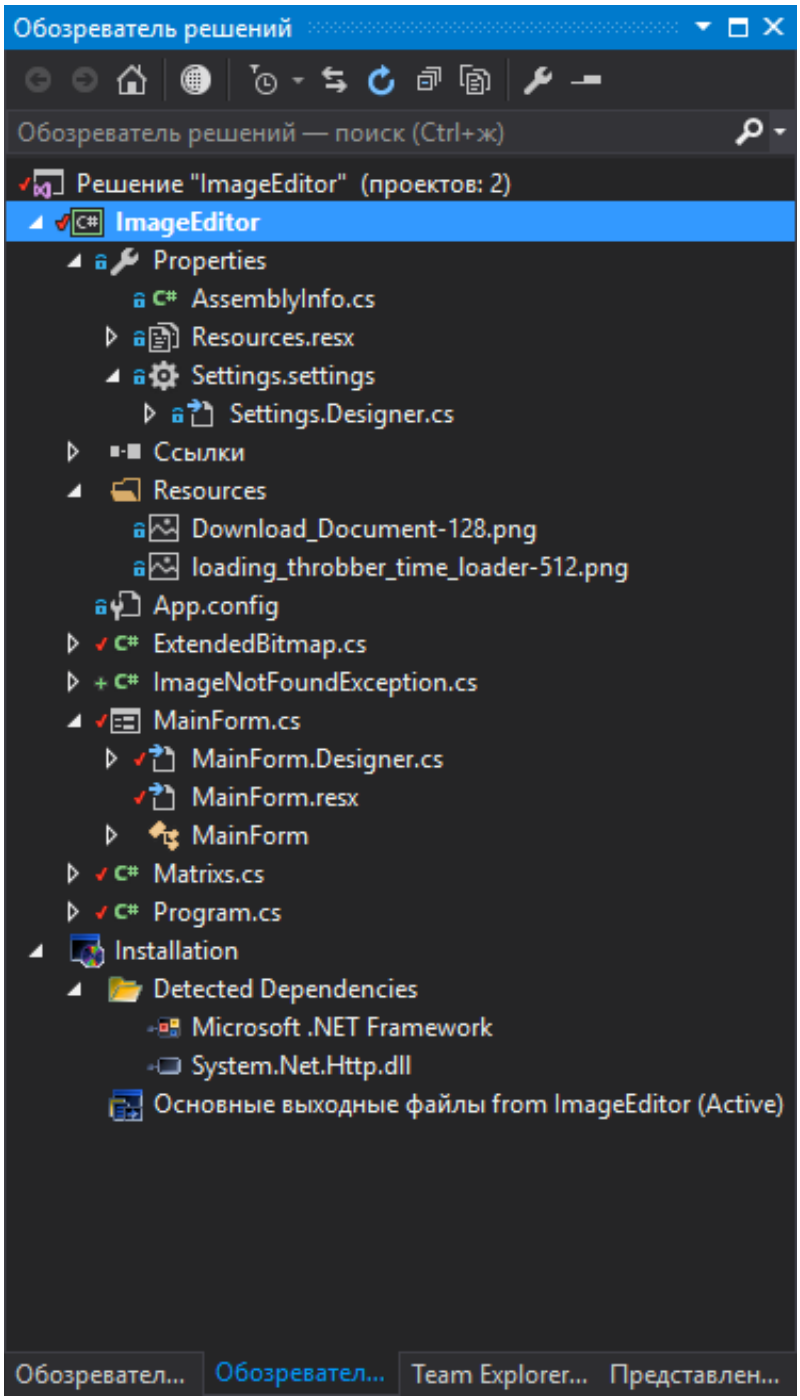

Рисунок 4 – Файловая структура проекта

Рассмотрим детальнее компоненты проекта с приложением (ImageEditor):

1. Properties содержит основные настройки приложения (целевая версия платформы .NET Framework, тип сборки и т.д)
   1.1. AssemblyInfo.cs – файл содержит в себе информацию о сборке
   1.2. Resourses.resx – файл, описывающий ресурсы
   1.3. Settings.settings – файл, в котором можно хранить некоторые настройки проекта для динамического изменения
2. References - содержит все ссылки на внешние компоненты в проекте C#
3. Resourses – папка с ресурсами проекта, которая содержит 2 изображения
4. App.config – конфигурационный файл XML, который может содержать, например строку подключения к БД и т.д.
5. MainForm.cs – файл, содержащий класс главного окна приложения, отличается от других файлов классов тем, что к данному файлу прикреплен файл
6. ExtendedBitmap.cs, ImageNotFoundException.cs, Matrixs.cs, Program.cs – файлы с классами C#

Также, в расширении есть дополнительный проект для создания инсталлятора приложения. Данный проект содержит всего два элемента: файл описания зависимостей, в котором указываются необходимые библиотеки и xml-файл настроек сборки инсталлятора. В Visual Studio последний отображается, как на рисунке 5, и позволяет задавать файлы, которые скопируются на компьютер пользователя после установки.

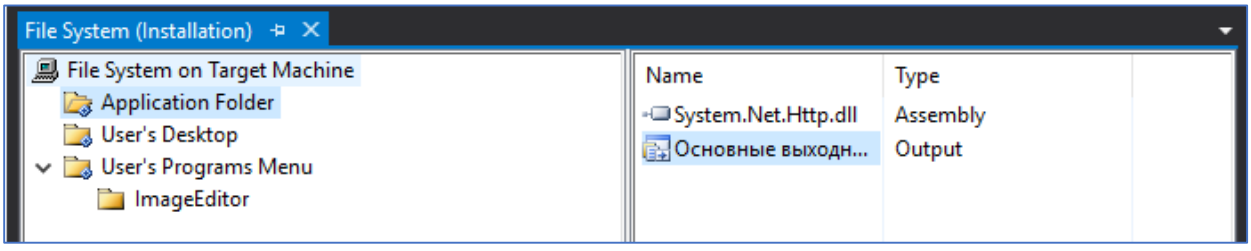

Рисунок 5 – Представление xml-файла настроек инсталлятора

4.2.2 Описание обьектов и их взаимодействия

Структура взаимодействия классов приложения приведена на рисунке 6.

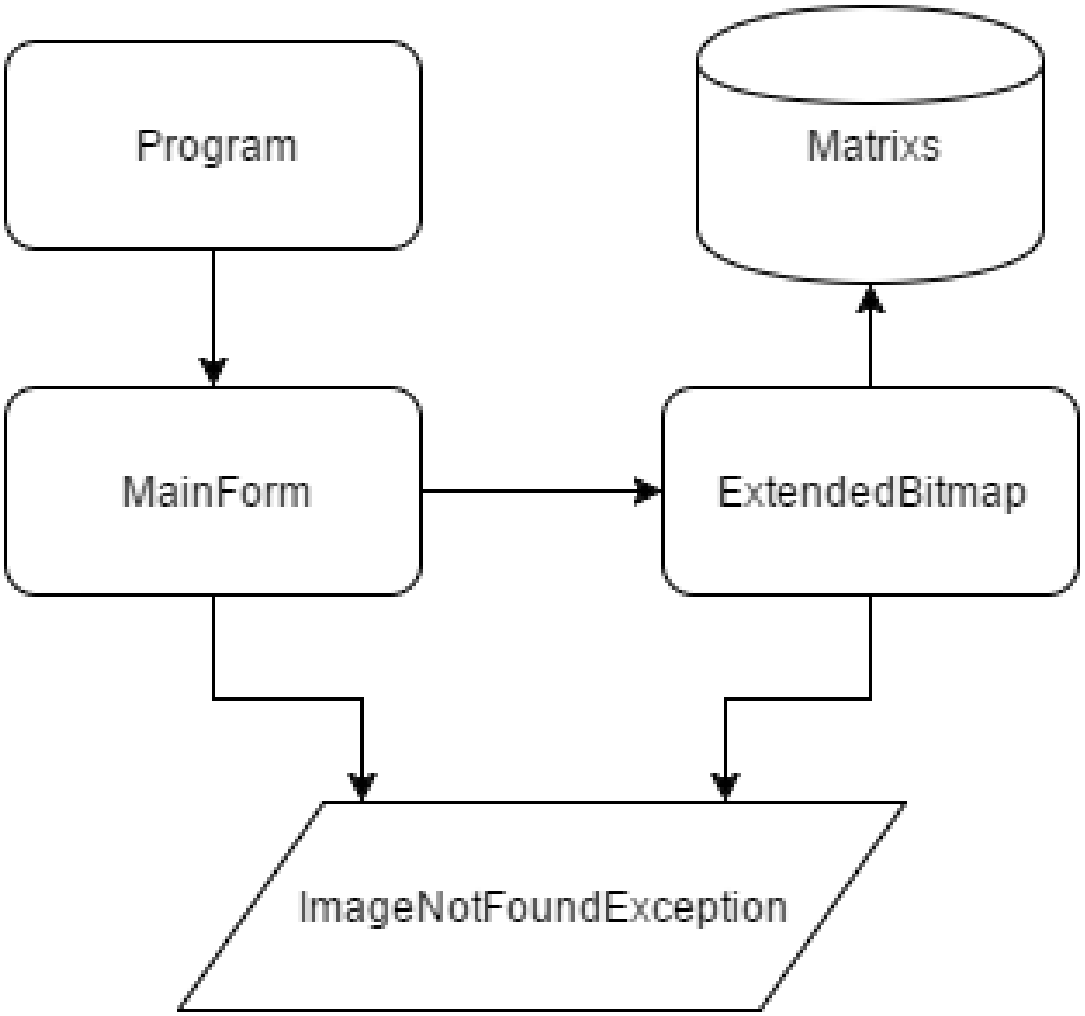

Рисунок 6 - Структура взаимодействия классов приложения

Рассмотрим подробнее схему взаимодействия классов на рисунке 6. Стрелки символизируют то, что класс, из которого исходит стрелка содержит экземпляр класса или обращается к статическим методам класса, к которому указывает стрелка. Форма элемента схемы просто разделяет классы по их функционалу. Подробнее про классы, содержащиеся на рисунке 1, рассмотрены в таблице 2.

Таблица 1 – Описание классов разрабатываемого приложения

Название класса	Описание
Program	Класс, сгенерированный при создании проекта Image Editor. Данный класс запускает приложение, создав новый экземпляр класса MainForm и передав его в статический метод Run класса Application (библиотеки Windows Form).
MainForm	Класс, наследуемый от класса Form, библиотеки Windows Form. Данный класс описан сразу в нескольких файлах (MainForm.cs, MainForm.Designer.cs), что позволяет разбить логику, управляемую дизайнером Windows Form (содержащиеся элементы интерфейса и их начальные значения, параметры окна, прописанные в свойствах графического редактирования окна) и логику поведения элементов при разных событиях, которую указывает пользователь.
Matrixs	Класс, содержащий массивы, описывающие операторы Собеля, Кирша, Лапласса и т.д. размерностей 3x3 и 5x5. Для операторов Собеля и по два массива (горизонтальный и вертикальный операторы).
ExtendedBitmap	Класс, который содержит методы расширяющие стандартный класс System.Drawing.Bitmap, добавляя методы, которые позволяют выполнять редактировать изображения, выделяя контуры обьектов с помощью операторов Собеля, Кирша, Лапласса и Превитта (Собель и Лапласс доступны в размерах 3x3 и 5x5).

Окончание таблицы 1

ImageNotFoundException	Класс пользовательского исключения. Вызывается в методах класса ExtendedBitmap, когда среди параметров одного из методов обработки изображения передается пустое изображение. Перехватывается исключение методах класса MainForm, после чего создается модальное окно для отображения ошибки пользователю.

По сути, основными классами приложения являются MainForm, который отвечает за интерфейсную часть приложения и логику работы графических элементов и класс ExtendedBitmap, методы которого выполняют обработку изображения. В таблицах 2 и 3 подробно описаны методы классов MainForm и ExtendedBitmap соответственно.

Таблица 2 – Методы класса MainForm

Метод	Описание
public MainForm()	Конструктор класса, вызывает инициализацию компонентов.
private void ChooseFileBtn (object sender, EventArgs e)	Вызывается при клике на кнопку загрузки изображения. Использует класс OpenFileDialog для выбора файла, после чего загружает файл и преобразовывает файл в формат bmp, инициализирует переменную bitmap, которая содержит текущее изображение.

Окончание таблицы 2

private void ResultPicture_Click (object sender, EventArgs e)	Метод, вызываемый при клике на результирующее изображение. Использует класс SaveFileDialog для диалога выбора пути и формата сохранения, в котором устанавливается фильтр на формат видимых изображений (jpeg, png, bmp). Если пользователь не выбрал формат, либо выбрал формат не и списка, то автоматически будет установлен формат png.
private void <method>Btn_Click (object sender, EventArgs e)	Ряд однотипных методов, вызываемых при нажатиях кнопок, каждая из которых отвечает за собственный метод обработки изображения. Все эти методы вызывают определенный метод (из класса ExtendedBitmap) у обьекта bitmap и возвращенное обработанное изображение возвращается в свойство Image обьекта, отвечающего за графический элемент отображающий результат обработки. Данное действие заключено в конструкцию try…catch, в которой происходит процесс отлова пользовательской ошибки ImageNotFoundException.

Таблица 3 – Методы класса ExtendedBitmap

Метод	Описание
private static Bitmap ConvolutionFilter (Bitmap sourceBitmap, double[,] filterMatrix, double factor = 1, int bias = 0, bool grayscale = false)	Метод единичной матричной свертки. Схож с обычными алгоритмами свертки, однако, работая с изображениями, свертывает отдельно значения цветов RGB в каждом пикселе. Метод имеет следующие параметры: • sourceBitmap – изображение для обработки • filterMatrix – матрица фильтра • factor – множитель степени обработки изображения • bias – смещение цветов к белому • grayscale – будет ли картинка обрабатываться как черно-белая
public static Bitmap ConvolutionFilter(this Bitmap sourceBitmap, double[,] xFilterMatrix, double[,] yFilterMatrix, double factor = 1, int bias = 0, bool grayscale = false)	Горизонтальная и вертикальная свертка матрицы изображения. Параметры данного метода похожи на параметры перегруженного им метода, но принимает две матрицы для свертки.

Окончание таблицы 3

public static Bitmap <method>Filter (this Bitmap sourceBitmap, bool gaus, bool grayscale = true)	Ряд однотипных методов для применения фильтров, обнаруживающих границы объектов на изображениях. Методы имеют следующие параметры: • sourceBitmap – изображение для обработки • gaus – параметр, определяющий нужно ли сгладить изображение • grayscale – параметр, определяющий нужно ли сгладить изображение Данные методы могут генерировать пользовательское исключение ImageNotFoundException.

За поведение элементов (изменение галочек в элементах checkbox при нажатии и т.д.) отвечают классы данных элементов из библиотеки Windows Form.

4.3 Описание алгоритмов работы приложения

В целом, большинство windows-приложений имеют одинаковый алгоритм работы приложения (рисунок 7), поскольку основаны на событийной системе. Т.е. внутри программы есть цикл, который отлавливает разнообразные события и обрабатывает их, запуская прикрепленные к ним методы.

Рисунок 7 – Алгоритм работы windows-приложения

Процесс обработки сообщений в Windows Form схематически показан на рисунке 8.

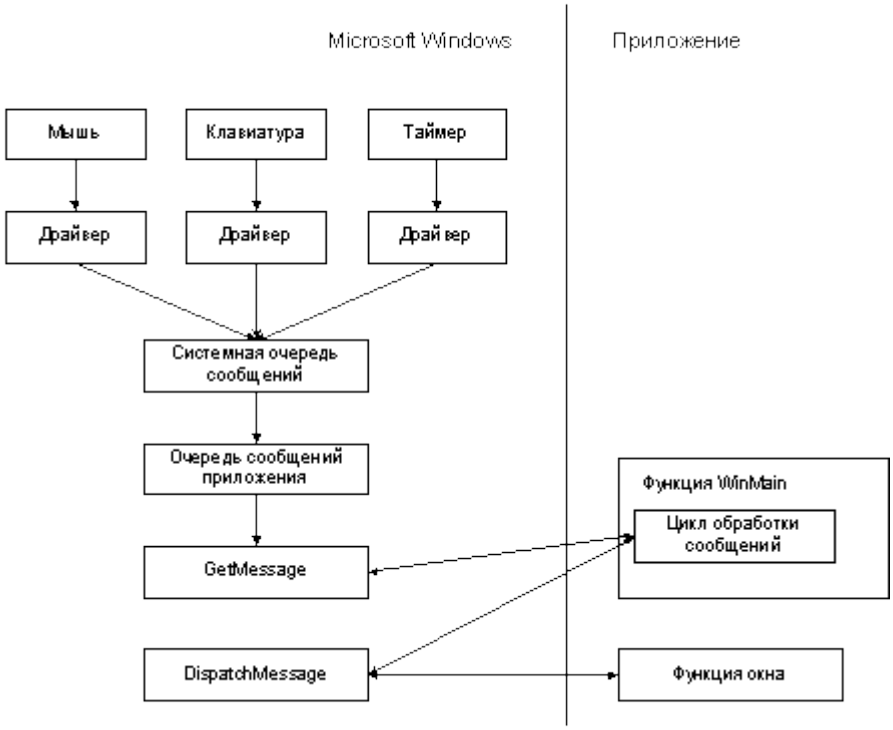

Рисунок 8 – Процесс обработки событий

Методы, привязанные к некоторым событиям, и методы, которые они вызывают также имеют свои алгоритмы работы. Алгоритм метода Sobel3x3Fillter и подобных ему фильтров класса ExtendedBitmap, которые производят обработку изображения изображен на рисунке 9.

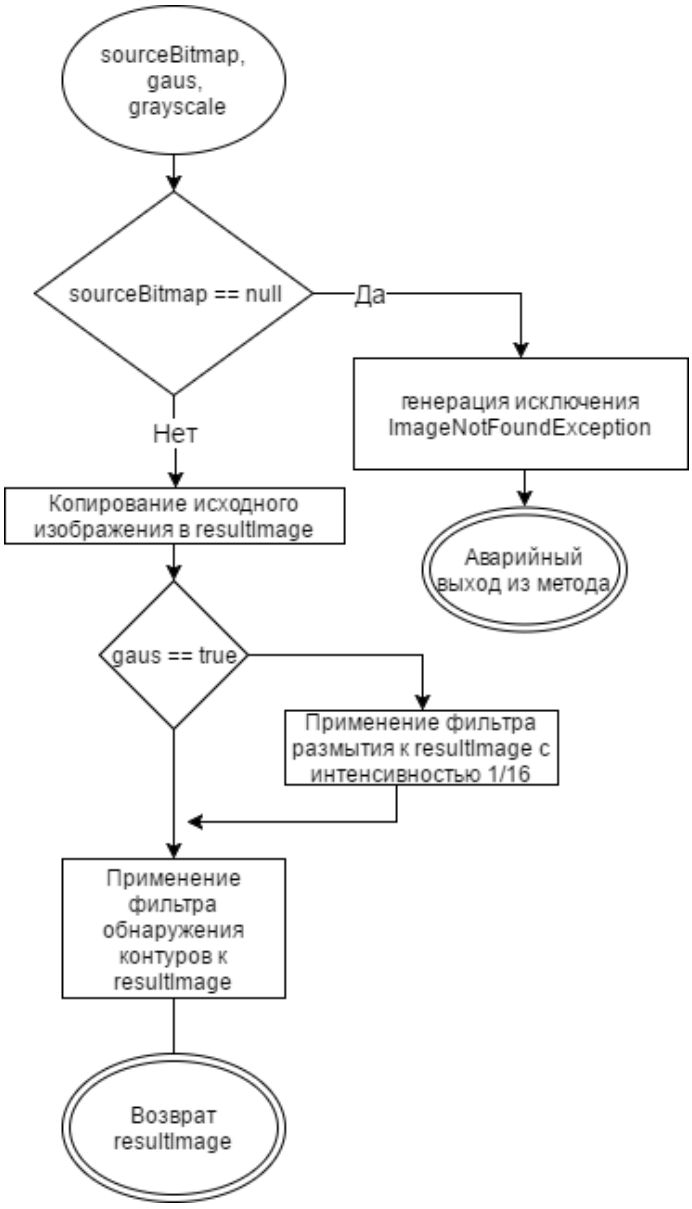

Рисунок 9 – Алгоритм метода Sobel3x3Fillter

Фильтры применяются с помощью одного из методов свертки, в зависимости от оператора. Копирование необходимо, чтобы исходное изображение не изменилось (входной параметр принимается по указателю).

5 Тестирование программного продукта

5.1 Аппаратные, системные и программные требования

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.

Для функционирования приложения следует, чтобы на компьютере пользователя были установлены:

- Операционная система —Windows XP или Windows 7/8/8.1/10;
- Microsoft .NET Framework 4.6

5.2 Руководства установки и использования

5.2.1 Руководство пользователю

Для установки приложения на компьютер пользователя необходимо выполнить следующие действия:

- запустить установщик
- ознакомиться с необходимой информацией, нажать «далее»
- выбрать папку для установки и пользователей для установки (для всех или только для текущего), нажать «далее»
- нажать «далее» и подождать до конца установки, нажать «готово»

После выполненных шагов exe-файл программы вместе с необходимыми ресурсами скопируется в выбранную при установке папку, а также ярлык для запуска программы появится в меню «Пуск» и на рабочем столе.

5.2.2 Руководство разработчику

Для разработки приложения и его компонентов необходимо:

- иметь установленную Visual Studio 2015 или боле позднюю версию программы с установленными компонентами для работы с языком C#
- наличие на компьютере установленного Microsoft .NET Framework 4.6
- установленное расширение Visual Studio Installer Projects в Visual Studio

5.3 Описание контрольных примеров

После запуска приложения появляется главное окно приложения. На нем находятся два элемента для отображения картинок с уже установленными изображениями.

При попытке обработать изображение без предварительного его выбора выскакивает диалоговое окно, предупреждающее об ошибке (рисунок 10). После закрытия этого окна или нажатии в нем на кнопку ОК пользователь возвращается к основному окну приложения.

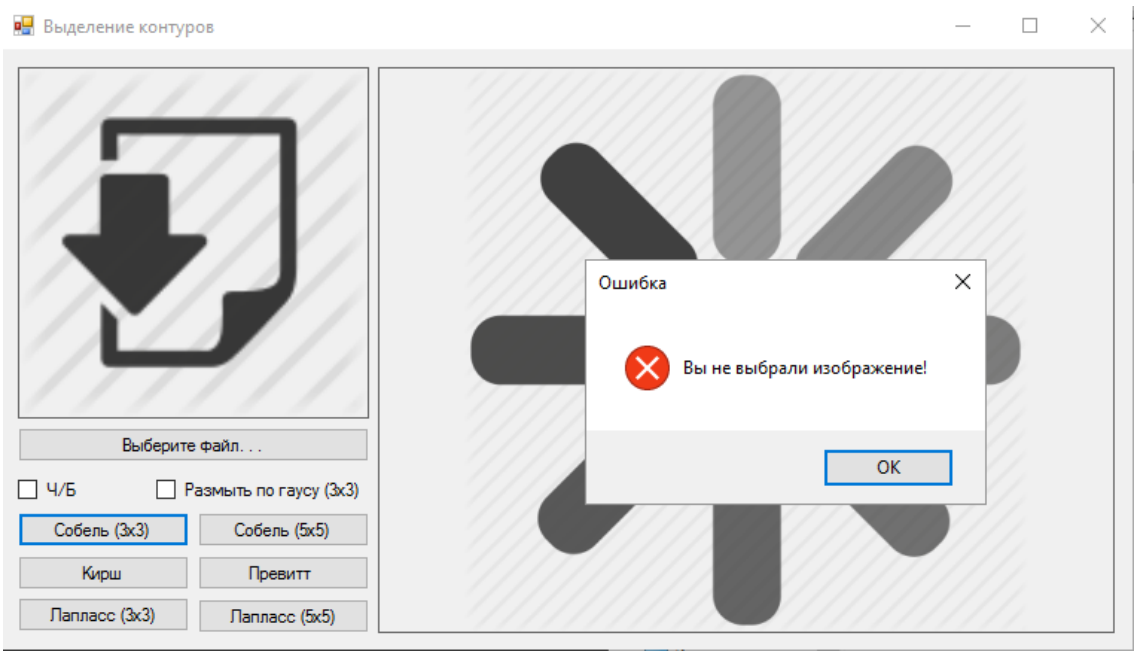

Рисунок 10 – Пример информирования пользователя об ошибке

При нажатии на кнопку с текстом «Выберите файл. . .» открывается файловый диалог для выбора файла. При выборе неподходящего файла пользователю будет показана данная ошибка в форме модального окна, а после выбора корректного изображения в левом элементе picture box появится выбранное изображение (приложение А.2), а текст в кнопке для выбора файла изменится на полный путь к файлу.

При нажатии на одну из кнопок обработки изображения через некоторое время (в зависимости от системных характеристик компьютера, в среднем около секунды) в правой части экрана появится обработанное исходное изображение с выделенными контурами. Если включить одну из настроек и повторить операцию, то результат будет отличаться в зависимости от выбранных настроек. Например, на рисунке 11 изображены два результата обработки одной и той же картинки с разными параметрами.

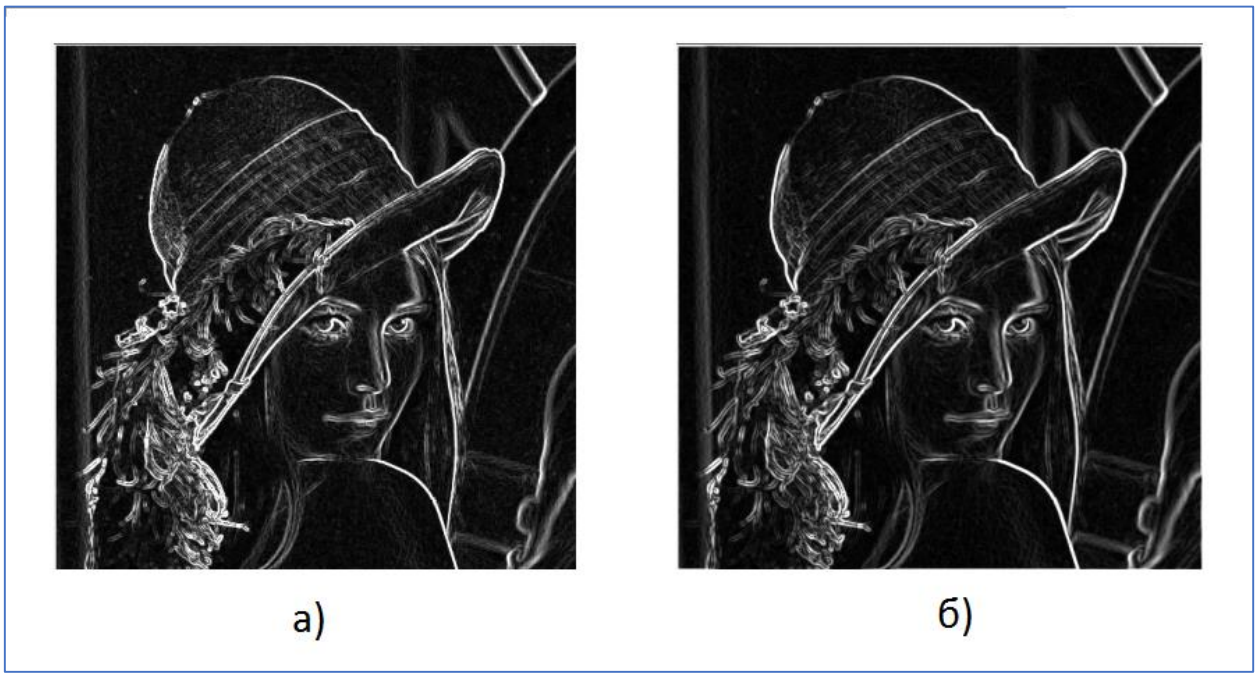

Рисунок 11 - результаты обработки одной и той же картинки с разными параметрами: а – ч/б, без сглаживания; б – ч/б, сглаживание

При масштабировании окна приложения результирующее изображение изменяется в размерах вместе с окном, а при нажатии на этот элемент интерфейса открывается файловый диалог для сохранения изображения.

ЗАКЛЮЧЕНИЕ

В результате работы над курсовой работой выполнены все поставленные задачи. Разработано рабочее приложение для выделения контуров обьектов на изображении. Был изучен ряд технологий для создания проекта: .NET, Windows Forms, C#, Visual Studio 2015. Также были изучены операторы Робертса, Собеля, Лапласса, Кирша и Превитта. Написано обоснование выбора данных технологий в курсовом проекте. Функционал приложения был утвержден согласно разработанной структуре программы. На основе разработанной функциональной схемы проекта составлен программный продукт, обладающий интерфейсной частью и работой с файловой системой.

На данный момент при работе с приложением доступны следующие возможности: загрузка изображения в приложение из файловой системы, настройка обработки изображения, выбор одного из шести операторов для обработки изображения, масштабирования главного окна приложения вместе с результирующим изображением.

Дальнейшее развитие системы связанно с расширением функционала, добавлением новых фильтров и персонализацией. Тестирование программы показало ее работоспособность.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Draw.io – онлайн конструктор диаграмм:
URL: https://www.draw.io/ (дата обращения: 22.03.2017)

2. ProfessionalC# 5.0 and .NET 4.5 / Автор: НейгелК.,ИвьенБ. / Издательство - М.: Вильямс, Год: 2014.

3. Хейлсберг А. Язык программирования C#. Классика Computers Science: 4-е изд. / А.Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд.

4. Стиллмен Э. Изучаем C#: 3-е изд. / Э. Стиллмен, Дж. Грин/

5. DependencyInjectionin .NET Букинистическое издание / Автор: Марк Симан, А. Барышнев, Евгений Зазноба, издательство: Питер.

6. Wikipedia – информационный ресурс:
URL: https://ru.wikipedia.org/wiki/.NET_Framework
(дата обращения: 26.03.2017).
URL: https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio
(дата обращения: 26.03.2017).
URL: https://ru.wikipedia.org/wiki/Windows_Forms
(дата обращения: 26.03.2017).

ПРИЛОЖЕНИЕ А

Экранные формы

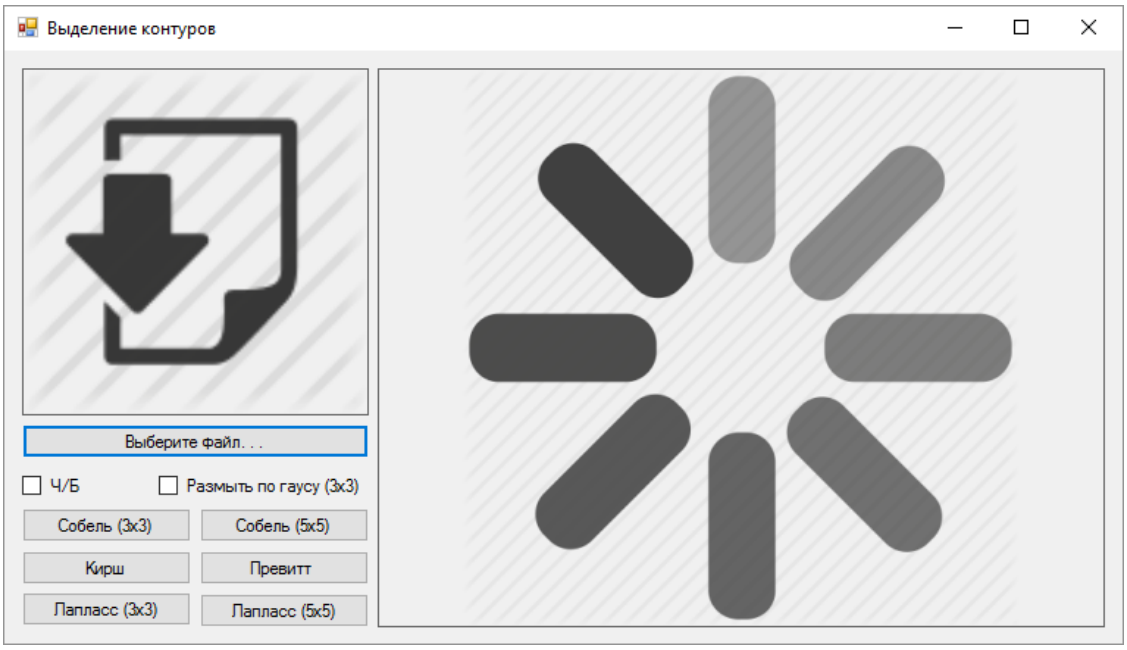

Рисунок А.1 – Начальный вид главного окна

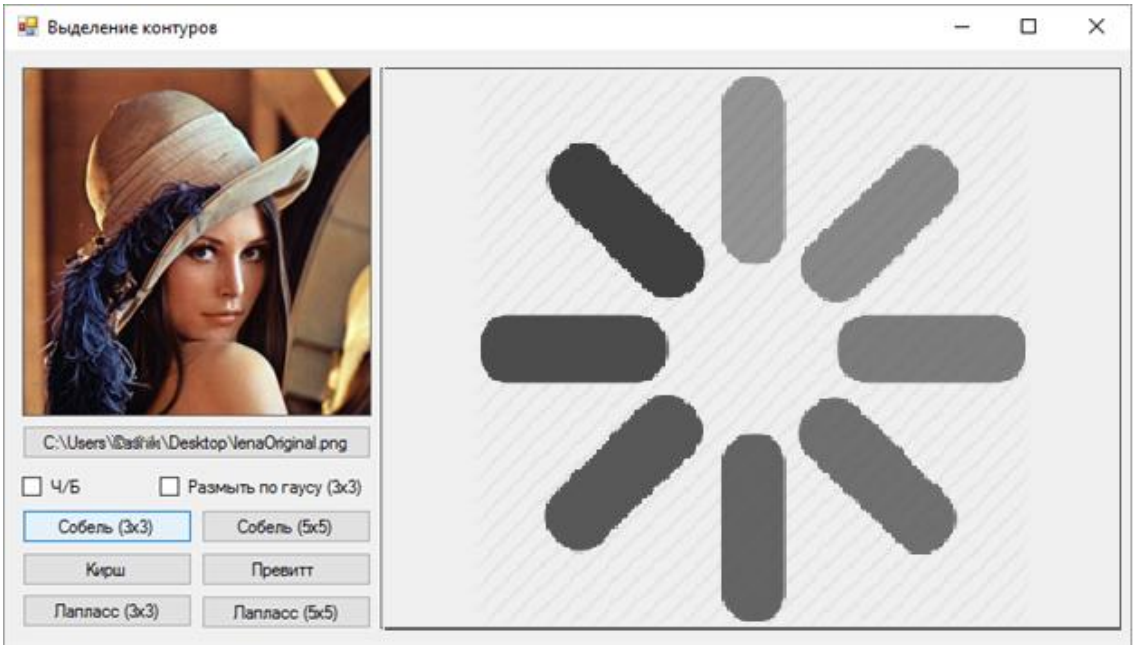

Рисунок А.2 - Вид окна после выбора изображения

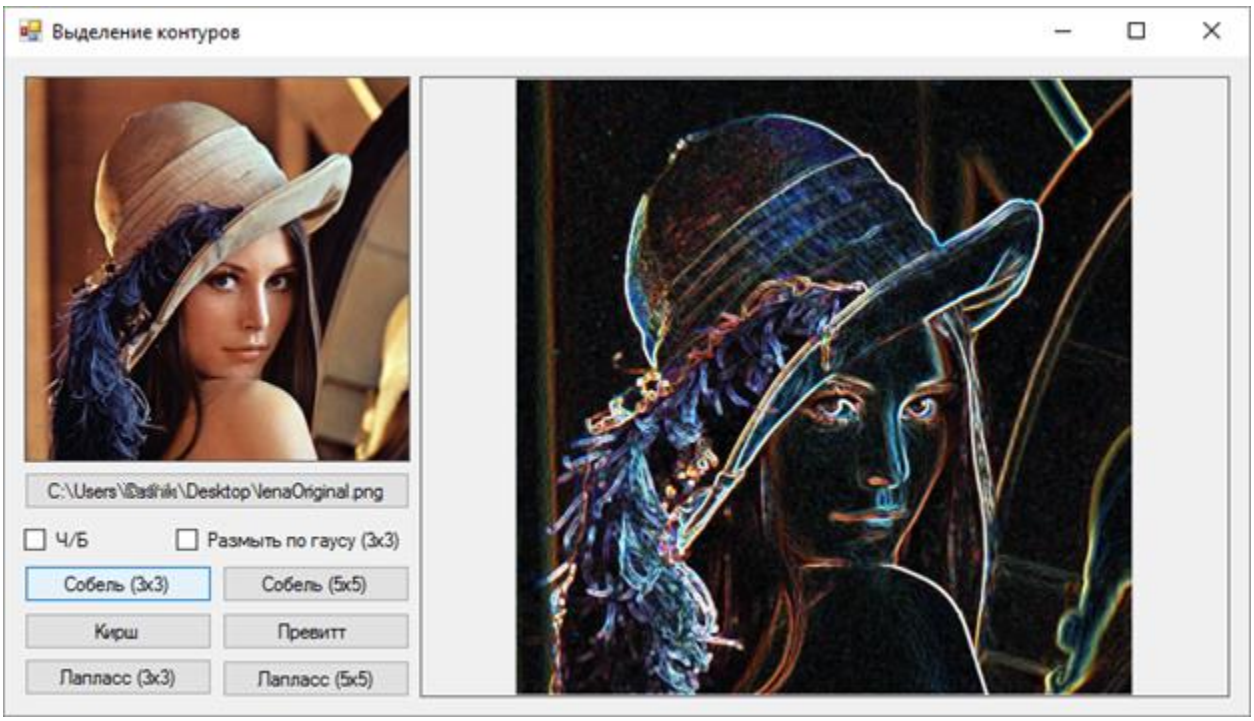

Рисунок А.3 – Вид окна после выбора и обработки изображения

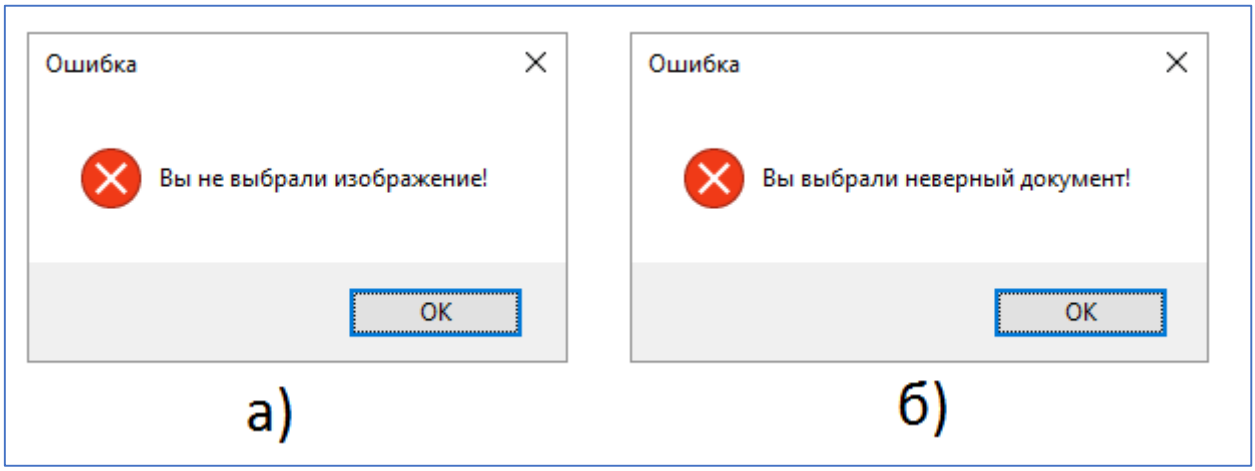

Рисунок А.4 Информирование пользователя об ошибках:
а – при попытке запуска обработки без выбранного изображения;
б – при попытке выбрать файл неподлежащего формата

ПРИЛОЖЕНИЕ Б

Фрагменты листинга

Листинг Б.1 – Файл MineForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using ImageEditor;
using System.IO;
using System.Drawing.Imaging;

namespace ImageEditor
{
    public partial class MainForm : Form
    {
        private Bitmap bitmap;
        public MainForm(){InitializeComponent();}
        private void button1_Click(object sender, EventArgs e)
        {
            OpenFileDialog file_dialog = new OpenFileDialog();
            file_dialog.Filter                =                "All
files|*.*|PNG|*.png|JPEGs|*.jpg|Bitmaps|*.bmp|GIFs|*.gif";
            file_dialog.FilterIndex = 1;
            if (file_dialog.ShowDialog() == DialogResult.OK)
            {
```

```
                StreamReader streamReader = new
StreamReader(file_dialog.FileName);
                try
                {
                    bitmap =
(Bitmap)Bitmap.FromStream(streamReader.BaseStream);
                    streamReader.Close();
                    StartPicture.Image = bitmap;
                    ChooseFileBtn.Text = file_dialog.FileName;
                }
                catch (ArgumentException)
                {
                    MessageBox.Show("Вы выбрали неверный
документ!",
                    "Ошибка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
                }

            }
        }
        private void SobelBtn_Click(object sender, EventArgs e)
        {
            try
            {
                ResultPicture.Image =
bitmap.Sobel3x3Filter(GaussianCB.Checked, BlackWhiteCB.Checked);
            }
            catch (ImageNotFoundException ex)
            {
                MessageBox.Show("Вы не выбрали изображение!",
                    "Ошибка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
            }
```

```
        }
        private void Sobel5x5Btn_Click(object sender, EventArgs e)
        {
            try
            {
            ResultPicture.Image                                    =
bitmap.Sobel5x5Filter(GaussianCB.Checked, BlackWhiteCB.Checked);
            }
            catch (ImageNotFoundException ex)
            {
                MessageBox.Show("Вы не выбрали изображение!",
                    "Ошибка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
            }
        }
        private void Kirsch_Click(object sender, EventArgs e)
        {
            try
            {
                ResultPicture.Image                                =
bitmap.KirschFilter(GaussianCB.Checked, BlackWhiteCB.Checked);
            }
            catch (ImageNotFoundException ex)
            {
                MessageBox.Show("Вы не выбрали изображение!",
                    "Ошибка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
            }
        }
        private void Prewitt_Click(object sender, EventArgs e)
        {
            try
            {
```

```
                ResultPicture.Image                                    =
bitmap.PrewittFilter(GaussianCB.Checked, BlackWhiteCB.Checked);
            }
            catch (ImageNotFoundException ex)
            {
                MessageBox.Show("Вы не выбрали изображение!",
                    "Ошибка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
            }
        }
        private    void    Laplacian3x3Btn_Click(object    sender,
EventArgs e)
        {
            try
            {
                ResultPicture.Image                                    =
bitmap.Laplacian3x3Filter(GaussianCB.Checked,
BlackWhiteCB.Checked);
            }
            catch (ImageNotFoundException ex){
                MessageBox.Show("Вы не выбрали изображение!",
                    "Ошибка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
            }
        }
        private    void    Laplacian5x5Btn_Click(object    sender,
EventArgs e)
        {
            try
            {
                ResultPicture.Image                                    =
bitmap.Laplacian5x5Filter(GaussianCB.Checked,
BlackWhiteCB.Checked);
```

```
            }
            catch (ImageNotFoundException ex)
            {
                MessageBox.Show("Вы не выбрали изображение!",
                    "Ошибка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
            }
        }
      private void ResultPicture_Click(object sender, EventArgs e)
        {
            SaveFileDialog sfd = new SaveFileDialog();
            sfd.Filter = "Images|*.png;*.bmp;*.jpg";
            ImageFormat format;
            if                (sfd.ShowDialog()                ==
System.Windows.Forms.DialogResult.OK)
            {
                string                    ext                    =
System.IO.Path.GetExtension(sfd.FileName);
                switch (ext)
                {
                    case ".jpg":
                        format = ImageFormat.Jpeg;
                        break;
                    case ".bmp":
                        format = ImageFormat.Bmp;
                        break;
                    default:
                        format = ImageFormat.Png;
                        break;
                }
                ResultPicture.Image.Save(sfd.FileName, format);
            }
        }
    }
```

Листинг Б.2 – Фрагмент файла ExtendedBitmap.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Drawing;
using System.Drawing.Imaging;
using System.Runtime.InteropServices;
using System.Drawing.Drawing2D;

namespace ImageEditor
{
    public static class ExtendedBitmap
    {
        private static Bitmap ConvolutionFilter(Bitmap sourceBitmap, double[,] filterMatrix, double factor = 1, int bias = 0, bool grayscale = false)
        {
            BitmapData sourceData = sourceBitmap.LockBits(new Rectangle(0, 0, sourceBitmap.Width, sourceBitmap.Height), ImageLockMode.ReadOnly, PixelFormat.Format32bppArgb);
            byte[] pixelBuffer = new byte[sourceData.Stride * sourceData.Height];
            byte[] resultBuffer = new byte[sourceData.Stride * sourceData.Height];
            Marshal.Copy(sourceData.Scan0, pixelBuffer, 0, pixelBuffer.Length);
            sourceBitmap.UnlockBits(sourceData);
            if (grayscale == true)
            {
                float rgb = 0;
                for (int k = 0; k < pixelBuffer.Length; k += 4)
                {
                    rgb = pixelBuffer[k] * 0.11f;
```

```
            rgb += pixelBuffer[k + 1] * 0.59f;
            rgb += pixelBuffer[k + 2] * 0.3f;
            pixelBuffer[k] = (byte)rgb;
            pixelBuffer[k + 1] = pixelBuffer[k];
            pixelBuffer[k + 2] = pixelBuffer[k];
            pixelBuffer[k + 3] = 255;
        }
    }
    double blue = 0.0;
    double green = 0.0;
    double red = 0.0;

    int filterWidth = filterMatrix.GetLength(1);
    int filterHeight = filterMatrix.GetLength(0);

    int filterOffset = (filterWidth - 1) / 2;
    int calcOffset = 0;

    int byteOffset = 0;

    for (int offsetY = filterOffset; offsetY <
        sourceBitmap.Height - filterOffset; offsetY++)
    {
        for (int offsetX = filterOffset; offsetX <
            sourceBitmap.Width - filterOffset; offsetX++)
        {
            blue = 0;
            green = 0;
            red = 0;

            byteOffset = offsetY *
                         sourceData.Stride +
                         offsetX * 4;

            for (int filterY = -filterOffset;
```

```
                    filterY <= filterOffset; filterY++)
                {
                    for (int filterX = -filterOffset;
                        filterX <= filterOffset; filterX++)
                    {

                        calcOffset = byteOffset +
                                     (filterX * 4) +
                                     (filterY                *
sourceData.Stride);

                            blue                                 +=
(double)(pixelBuffer[calcOffset]) *
                                       filterMatrix[filterY       +
filterOffset,
                                                                  filterX +
filterOffset];

                            green                                +=
(double)(pixelBuffer[calcOffset + 1]) *
                                        filterMatrix[filterY      +
filterOffset,
                                                                  filterX +
filterOffset];

                            red                                  +=
(double)(pixelBuffer[calcOffset + 2]) *
                                      filterMatrix[filterY        +
filterOffset,
                                                                filterX   +
filterOffset];
                    }
                }

                blue = factor * blue + bias;
```

```
                    green = factor * green + bias;
                    red = factor * red + bias;

                    if (blue > 255)
                    { blue = 255; }
                    else if (blue < 0)
                    { blue = 0; }

                    if (green > 255)
                    { green = 255; }
                    else if (green < 0)
                    { green = 0; }

                    if (red > 255)
                    { red = 255; }
                    else if (red < 0)
                    { red = 0; }

                    resultBuffer[byteOffset] = (byte)(blue);
                    resultBuffer[byteOffset + 1] = (byte)(green);
                    resultBuffer[byteOffset + 2] = (byte)(red);
                    resultBuffer[byteOffset + 3] = 255;
                }
            }
            Bitmap resultBitmap = new Bitmap(sourceBitmap.Width,
sourceBitmap.Height);

            BitmapData resultData = resultBitmap.LockBits(new
Rectangle(0,  0,  resultBitmap.Width,  resultBitmap.Height),
ImageLockMode.WriteOnly, PixelFormat.Format32bppArgb);
            Marshal.Copy(resultBuffer,  0,  resultData.Scan0,
resultBuffer.Length);
            resultBitmap.UnlockBits(resultData);
            return resultBitmap;
        }
```

Листинг Б.3 – Фрагмент файла Matrixs.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ImageEditor
{
    public static class Matrix
    {
        public static double[,] Laplacian3x3
        {
            get
            {
                return new double[,]
                { { -1, -1, -1,  },
                  { -1,  8, -1,  },
                  { -1, -1, -1,  }, };
            }
        }

        public static double[,] Laplacian5x5
        {
            get
            {
                return new double[,]
                { { -1, -1, -1, -1, -1, },
                  { -1, -1, -1, -1, -1, },
                  { -1, -1, 24, -1, -1, },
                  { -1, -1, -1, -1, -1, },
                  { -1, -1, -1, -1, -1  }, };
            }
        }
```

```
public static double[,] Gaussian3x3
{
    get
    {
        return new double[,]
        { { 1, 2, 1, },
          { 2, 4, 2, },
          { 1, 2, 1, }, };
    }
}


public static double[,] Sobel3x3Horizontal
{
    get
    {
        return new double[,]
        { { -1,  0,  1, },
          { -2,  0,  2, },
          { -1,  0,  1, }, };
    }
}

public static double[,] Sobel3x3Vertical
{
    get
    {
        return new double[,]
        { {  1,  2,  1, },
          {  0,  0,  0, },
          { -1, -2, -1, }, };
    }
}
```