|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | «Информатика и системы управления» |

|  |  |
| --- | --- |
| КАФЕДРА | «Теоретическая информатика и компьютерные технологии» |

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К КУРСОВОЙ РАБОТЕ***

***НА ТЕМУ:***

***Обработка и фильтрация цифровых изображений***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ9-52Б |  |  |  | Караник А.А. |
|  | (Группа) |  | (Подпись, дата) |  | (И.О. Фамилия) |

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель курсовой работы |  |  | Каганов Ю.Т. |
|  | (Подпись, дата) |  | (И.О. Фамилия) |

*2023*

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 1](#_Toc159985375)

[ВВЕДЕНИЕ 3](#_Toc159985376)

[1. Цель и основные задачи 4](#_Toc159985377)

[2. Описание алгоритмов 5](#_Toc159985378)

[2.1. Фильтр размытия (box blur) 5](#_Toc159985379)

[2.2. Разделяемый фильтр размытия (separable box blur) 5](#_Toc159985380)

[2.3. Размытие по Гауссу 6](#_Toc159985381)

[2.4. Разделяемое размытие по Гауссу 6](#_Toc159985382)

[2.5. Медианный фильтр 7](#_Toc159985383)

[2.6. Фильтр Лапласа 7](#_Toc159985384)

[2.7. Повышение резкости по Лапласу 8](#_Toc159985385)

[2.8. Оператор Собеля 8](#_Toc159985386)

[2.9. Оператор Прюитт 9](#_Toc159985387)

[2.10. Перекрёстный оператор Робертса 9](#_Toc159985388)

[2.11. Изменение яркости 10](#_Toc159985389)

[2.12. Изменение оттенка и насыщенности в HSV 10](#_Toc159985390)

[2.13. Изменение светлости в HSL 12](#_Toc159985391)

[2.14. Гамма-коррекция 12](#_Toc159985392)

[2.15. Формирование порога (Thresholding) 13](#_Toc159985393)

[2.16. Добавление гауссовского шума 14](#_Toc159985394)

[3. Реализация 14](#_Toc159985395)

[3.1. Выбор стека технологий 14](#_Toc159985396)

[3.2. Реализация алгоритмов 15](#_Toc159985397)

[3.3. Интерфейс программы 19](#_Toc159985398)

[4. Тестирование 21](#_Toc159985399)

[ЗАКЛЮЧЕНИЕ 32](#_Toc159985400)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 33](#_Toc159985401)

# ВВЕДЕНИЕ

Обработка и фильтрация цифровых изображений является неотъемлемой частью современной технологической среды. Эта область науки и инженерии занимается анализом и изменением цифровых изображений с целью улучшения их качества или извлечения полезной информации. Актуальность темы обработки и фильтрации цифровых изображений подчеркивается ее важностью в различных областях, начиная от медицины и заканчивая развлекательной индустрией.

В медицине обработка и фильтрация цифровых изображений играет решающую роль в диагностике и лечении различных заболеваний. Она используется для анализа медицинских изображений, таких как рентгеновские снимки, МРТ и КТ, что помогает в раннем обнаружении патологий и улучшении точности диагноза.

В области безопасности обработка и фильтрация цифровых изображений используется для видеонаблюдения, распознавания лиц, автомобильных номеров, анализа поведения людей и обнаружения потенциально опасных ситуаций. Это помогает в обеспечении общественной безопасности и предотвращении преступлений.

В промышленности обработка и фильтрация цифровых изображений применяется для контроля качества продукции, обнаружения дефектов на поверхности и автоматизации процессов сортировки и упаковки. Это позволяет улучшить эффективность производства и сократить потери из-за брака.

В развлекательной индустрии цифровая обработка изображений используется для создания спецэффектов в кино, разработки компьютерных игр, улучшения качества видеороликов и фотографий, что делает развлечения более захватывающими и реалистичными для зрителей и пользователей.

В научных исследованиях цифровая обработка изображений играет важную роль в анализе и интерпретации данных в различных научных областях, от астрономии до биологии и геологии. Это позволяет исследователям получать более точные и полезные результаты и делает возможным новые открытия и инновации.

## Цель и основные задачи

Целью данной курсовой работы является освоение и практическое применение методов обработки и фильтрации цифровых изображений. В рамках исследования предполагается осуществить углубленное изучение основных принципов обработки изображений, включая такие методы, как фильтрация, подавление шума, выделение краев и другие техники. Для достижения этой цели будет проанализирован и реализован широкий спектр методов и алгоритмов, применяемых в обработке цифровых изображений. Такой подход позволит не только овладеть теоретическими знаниями в области обработки изображений, но и разработать практические навыки работы с различными инструментами и методами обработки цифровых изображений.

Основные задачи:

1. Изучение алгоритмов обработки и фильтрации цифровых изображений:
   1. Провести обзор основных алгоритмов обработки цифровых изображений, включая методы подавления шума, размытия, увеличения резкости, выделения краев и другие.
   2. Изучить принципы работы каждого алгоритма, его математическую основу и область применения.
2. Реализация алгоритмов обработки и фильтрации цифровых изображений:
   1. Написать код для реализации изученных алгоритмов обработки и фильтрации цифровых изображений с использованием языка программирования C++.
   2. Обеспечить эффективную и оптимизированную работу алгоритмов для обработки изображений большого размера.
3. Написание интерфейса для применения данных алгоритмов:
   1. Разработать пользовательский интерфейс программы, который позволит пользователям легко выбирать и применять различные алгоритмы обработки и фильтрации цифровых изображений.
   2. Обеспечить удобство использования интерфейса и интуитивно понятный способ настройки параметров алгоритмов.
4. Тестирование и оценка результатов:
   1. Провести тестирование разработанных алгоритмов и интерфейса на разнообразных тестовых изображениях с различными характеристиками.
   2. Оценить качество и эффективность работы алгоритмов.

## Описание алгоритмов

## Фильтр размытия (box blur)

Box blur (также известный как box linear filter) – это линейный фильтр пространственной области, в котором каждый пиксель результирующего изображения имеет значение, равное среднему значению соседних пикселей во входном изображении. Размытие 3 на 3 радиуса 1 можно записать как матрицу [1]:

Для применения Box blur к краям исходного изображения часто используется техника отражения, при которой краевые пиксели изображения дублируются и добавляются к нему, создавая эффект плавного перехода. Это позволяет избежать появления артефактов и сохранить естественный внешний вид изображения после применения размытия.

## Разделяемый фильтр размытия (separable box blur)

Separable box blur – это box blur представляющий собой разделяемый фильтр, поэтому для каждого пикселя потребуется только два одномерных прохода размером пикселей: один по горизонтали и один по вертикали. Это снижает сложность с до , где – радиус размытия, – количество пикселей изображения [1].

## Размытие по Гауссу

Размытие по Гауссу – это тип фильтра размытия изображения, который использует функцию Гаусса для расчета преобразования, применяемого к каждому пикселю изображения.

Это широко используемый метод в графическом программном обеспечении, обычно для уменьшения шума изображения и уменьшения детализации. Визуальный эффект этого метода размытия представляет собой плавное размытие, напоминающее просмотр изображения через полупрозрачный экран. Сглаживание по Гауссу также используется в качестве этапа предварительной обработки в алгоритмах компьютерного зрения для улучшения структуры изображений в разных масштабах.

Формула функции Гаусса в одном измерении выглядит следующим образом:

(1)

В двух измерениях это произведение двух таких функций Гаусса, по одной в каждом измерении:

(2)

где – расстояние от начала координат по горизонтальной оси, – расстояние от начала координат по вертикальной оси, а – стандартное отклонение распределения Гаусса. Значения из этого распределения используются для построения матрицы свертки, которая применяется к исходному изображению [2].

Как и в случае Box blur для применения размытия по Гауссу к краям исходного изображения часто используется техника отражения.

## Разделяемое размытие по Гауссу

Аналогично Separable box blur, чтобы уменьшить вычислительную сложность, применяется разделяемый подход, который разделяет двумерный фильтр Гаусса на два одномерных.

Сначала изображение размывается по горизонтали с помощью одномерного фильтра Гаусса, а затем результат размытия обрабатывается по вертикали с использованием такого же фильтра. Этот метод значительно упрощает вычисления и ускоряет процесс размытия.

## Медианный фильтр

Медианный фильтр — это метод нелинейной цифровой фильтрации, часто используемый для удаления шума из изображения или сигнала. Такое снижение шума является типичным этапом предварительной обработки для улучшения результатов последующей обработки (например, обнаружения краев на изображении).

Основная идея медианного фильтра состоит в том, чтобы заменить каждое значение пикселя изображения на медианное значение его соседних пикселей [3].

Медианная фильтрация – эффективная процедура обработки сигналов, подверженных воздействию импульсных помех [4].

## Фильтр Лапласа

Фильтр Лапласа или дискретный оператор Лапласа — это фильтр для обнаружения границ, который аппроксимирует оператор Лапласа:

, где – исходное изображение.

Существует несколько способов определения дискретного лапласиана с использованием окрестностей. Одно из наиболее часто используемых определений имеет следующий вид:

Это уравнение можно реализовать с помощью матрицы свертки:

Еще один распространенный вариант определения дискретного оператора Лапласа задается следующей матрицей свертки [5]:

При применении фильтра Лапласа к изображению может возникнуть ситуация, когда значения пикселей выходят за пределы допустимого диапазона, что может привести к искажению или потере качества изображения. В таких случаях чаще всего применяются два основных метода для решения проблемы: нормализация и обрезка. Нормализация позволяет масштабировать значения пикселей таким образом, чтобы они оставались в пределах допустимого диапазона, сохраняя при этом относительные различия между значениями. Обрезка, с другой стороны, просто обрезает значения пикселей до максимального или минимального допустимого значения в случае выхода за пределы. Оба подхода имеют свои преимущества и применяются в зависимости от конкретных требований и условий обработки изображений.

## Повышение резкости по Лапласу

Повышение резкости по Лапласу (Laplacian sharpening или Laplacian enhancement) – это способ использования фильтра Лапласа для улучшения краев изображения, чтобы они выглядели более резкими.

Пусть – исходное изображение, – преобразованное изображение, – дискретный лапласиан, – центральный коэффициент матрицы свертки. Тогда повышение резкости по Лапласу задается следующим образом [5]:

(3)

## Оператор Собеля

Оператор Собеля – дискретный дифференциальный оператор, вычисляющий приближённое значение градиента яркости изображения. Результатом применения оператора Собеля в каждой точке изображения является либо вектор градиента яркости в этой точке, либо его норма. Используется в области обработки изображений, в частности, часто применяется в алгоритмах выделения границ.

Оператор использует две матрицы свертки 3 на 3, с которыми сворачивают исходное изображение для расчета приближённых значений производных как по горизонтали, так и по вертикали.

Пусть A – исходное изображение, а и – два изображения, на которых каждая точка содержит приближённые производные. Они вычисляются следующим образом:

(4)

Здесь означает двумерную операцию свертки.

В каждой точке изображения приближённое значение величины градиента можно вычислить путём использования полученных приближенных значений производных [6]:

(5)

## Оператор Прюитт

Оператор Прюитт практически аналогичен оператору Собеля, отличается лишь вычислением и соответственно [7]:

(6)

## Перекрёстный оператор Робертса

Перекрёстный оператор Робертса – один из ранних алгоритмов выделения границ, который вычисляет на плоском дискретном изображении сумму квадратов разниц между диагонально смежными пикселями. Это может быть выполнено сверткой изображения с двумя следующими матрицами [8]:

## Изменение яркости

Изменение яркости изображения играет важную роль в обработке цифровых изображений, позволяя корректировать экспозицию и улучшать качество изображения. Эта техника полезна для исправления недостатков съемки, что позволяет достичь более точного воспроизведения яркости и контрастности. Благодаря изменению яркости, можно также подчеркнуть важные детали на изображении или создать определенное настроение, делая изображение более привлекательным и информативным для зрителя.

Изменение яркости изображения можно достичь несколькими способами. Один из них - это изменение яркости путём сложения или вычитания одной константы из каждого канала цвета в модели RGB. Например, для увеличения яркости изображения можно добавить к каждому каналу одно и то же положительное значение, а для уменьшения – вычесть.

Другой способ изменения яркости – умножение каждого канала на определенное число. При умножении на число больше 1 изображение становится ярче, а при умножении на число меньше 1 – темнее. Этот метод более гибкий и сохраняет контрастность изображения, поскольку он изменяет относительные яркости пикселей.

## Изменение оттенка и насыщенности в HSV

Изменение оттенка изображения имеет немаловажное значение в области обработки и фильтрации цифровых изображений, поскольку позволяет создавать эффекты, корректировать цветовые тона и улучшать визуальное восприятие. Эта техника дает возможность исправлять баланс цветов. Например, изменение оттенка может быть использовано для коррекции нежелательных оттенков в изображении или для придания изображению определенной атмосферы, что делает его более привлекательным и выразительным.

Для изменения оттенка и насыщенности цветов изображения можно воспользоваться цветовой моделью HSV. Эта модель представляет цвет в виде трех компонентов: оттенок (hue), насыщенность (saturation), value.

Если изображение представлено в цветовой модели RGB, то тогда необходимо преобразовать это изображение к модели HSV:

Пусть . Тогда

(7)

Здесь V – это value, – это оттенок, – это насыщенность.

После изменения оттенка и насыщенности путём прибавления или вычитания определенных значений можно выполнить обратное преобразование:

Пусть . Тогда [9]

(8)

## Изменение светлости в HSL

Для изменения светлости изображения нужно воспользоваться цветовой моделью HSL. Эта модель в свою очередь представляет цвет в виде трех компонентов: оттенок (hue), насыщенность (saturation), светлость (lightness).

Преобразование изображения с цветовой моделью RGB к HSL выглядит следующим образом:

(9)

Здесь – это светлость, – это оттенок, – это насыщенность.

Обратное преобразование из цветовой модели HSL к модели RGB совпадает с преобразованием из HSV к RGB, за исключением того, что в качестве берется значение [9].

## Гамма-коррекция

Гамма-коррекция или коррекция гаммы – предыскажения яркости чёрно-белого или цветоделённых составляющих цветного изображения при его записи в телевидении и цифровой фотографии. В качестве передаточной функции при гамма-коррекции чаще всего используется степенная в виде:

(10)

где служит коэффициентом, а входные и выходные значения – неотрицательные вещественные числа. В общем случае, если , то входные и выходные значения находятся в пределах от 0 до 1. При равенстве единице характеристика передачи полутонов линейна и перепады освещённости объекта в светах и тенях отображаются одинаково [10].

Осветление изображения с помощью гамма-коррекции может привести к появлению новых деталей в темных областях изображения, делая их более различимыми.

## Формирование порога (Thresholding)

Thresholding – это фундаментальный метод в обработке изображений, который используется для преобразования изображения в двоичное, разделяя его на области пикселей с высокой и низкой яркостью. Суть его заключается в установлении порогового значения, по которому определяется, какой пиксель будет принадлежать к одной из двух категорий: белый или черный. Обычно пиксели с яркостью выше порога принимают белый цвет, а пиксели с яркостью ниже порога – черный цвет. Этот метод широко используется для выделения объектов, сегментации изображений, удаления шума и других задач обработки изображений. Однако эффективное применение требует тщательного выбора порогового значения, что может зависеть от конкретного изображения и требуемых результатов.

В дополнение к простому одномерному пороговому значению, существует возможность использовать два пороговых значения: одно для верхней границы и другое для нижней. Это позволяет более гибко контролировать процесс преобразования изображения, позволяя выделить определенный диапазон яркости или цвета.

Кроме того, thresholding может быть применен к отдельным каналам цвета в многоканальных изображениях, таких как RGB. Это позволяет проводить сегментацию и выделение объектов на основе их цвета или цветовых характеристик.

## Добавление гауссовского шума

Гауссовский шум – статистический шум, имеющий плотность вероятности, равную плотности вероятности нормального распределения, также известного как гауссовское распределение.

Плотность вероятности гауссовской случайной величины равна:

(11)

где отображает серый уровень, – среднее значение и – стандартное отклонение [11].

Затем полученное значение добавляется к пикселю исходного изображения.

## Реализация

## Выбор стека технологий

При выборе стека технологий для реализации программы по обработке и фильтрации цифровых изображений было принято во внимание несколько ключевых критериев. Язык программирования C++ был выбран в качестве основного языка разработки благодаря его высокой производительности, широкому спектру возможностей и обширной поддержке в индустрии разработки программного обеспечения. Использование C++ обеспечит эффективную работу с изображениями и реализацию сложных алгоритмов обработки.

Для создания пользовательского интерфейса была выбрана библиотека OpenGL [12]. OpenGL является открытым стандартом для разработки графических приложений и обеспечивает высокую производительность визуализации изображений на различных платформах.

Вместе с библиотекой OpenGL была выбрана GLEW (OpenGL Extension Wrangler Library) [13], библиотека, предназначенная для управления расширениями OpenGL. Она обеспечивает доступ к новым функциям и возможностям OpenGL, которые не являются частью основной спецификации. GLEW упрощает загрузку и использование этих расширений, обеспечивая кросс-платформенную совместимость и обработку зависимостей между ними. Это делает GLEW важным компонентом при разработке приложений, использующих новейшие возможности OpenGL для создания качественной графики и визуализации.

Для создания и управления окнами, контекстами OpenGL и обработки ввода в графических приложениях была выбрана легковесная кросс-платформенная библиотека GLFW (Graphics Library Framework) [14]. Она обеспечивает удобный интерфейс для создания окон приложения, управления их параметрами и обработки событий ввода с клавиатуры, мыши и джойстика. GLFW также поддерживает множество расширений OpenGL и обеспечивает кросс-платформенную совместимость, что делает ее идеальным выбором для разработки графических приложений на различных операционных системах.

Для загрузки изображений из файлов различных форматов была выбрана библиотека stb\_image [15]. Это простая и легковесная библиотека, которая обеспечивает быструю и эффективную загрузку изображений без необходимости использования дополнительных зависимостей.

Для параллельной обработки изображений на многоядерных процессорах, чтобы повысить эффективность работы алгоритмов, была выбрана библиотека OpenMP [16]. OpenMP является стандартом параллельного программирования для языка C++ и обеспечивает простой и эффективный способ распараллеливания выполнения задач на несколько потоков.

Наконец, для обеспечения интеграции с операционной системой Windows был использован Windows API [17]. Он предоставляет доступ к функциям операционной системы, таким как работа с файлами, окнами и т.д., что позволяет создать полностью функциональное приложение для пользователей Windows.

Таким образом, выбранный стек технологий обеспечивает оптимальное сочетание производительности, функциональности и удобства разработки для реализации программы по обработке и фильтрации цифровых изображений.

## Реализация алгоритмов

Все алгоритмы обработки и фильтрации цифровых изображений наследуются от базового класса ImageOperation. Этот класс содержит виртуальный метод apply, который принимает входные и выходные массивы байт изображения, а также его ширину, высоту и формат.

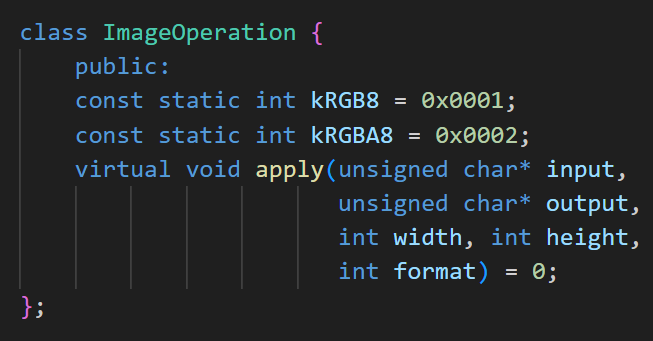


Рисунок 1 – базовый класс алгоритмов обработки изображений

Этот подход позволяет обобщить интерфейс для всех алгоритмов обработки изображений и упростить добавление новых алгоритмов в систему, сохраняя при этом совместимость с существующим кодом.

Формат, определенный в классе ImageOperation, представляет только цветовые модели RGB и RGBA с 8-битной глубиной. Это означает, что каждый канал цвета (красный, зеленый, синий и, в случае RGBA, альфа-канал) представлен 8-битным значением, которое находится в диапазоне от 0 до 255.

Такое ограничение формата упрощает реализацию и использование алгоритмов обработки изображений, предполагая, что входные и выходные изображения всегда имеют один и тот же формат данных. Это также уменьшает сложность кода и повышает его производительность за счет отсутствия необходимости обрабатывать различные форматы изображений.

В реализованных алгоритмах обработки и фильтрации цифровых изображений часто используются оптимизации для повышения производительности. Одним из способов оптимизации является параллельное выполнение циклов с помощью библиотеки OpenMP.

Использованные директивы для распараллеливания циклов позволяют одновременно выполнять итерации цикла на нескольких процессорных ядрах. Это особенно полезно для операций обработки изображений, так как многие алгоритмы могут быть легко распараллелены, например, обход пикселей изображения или применение фильтров.

Еще одним способом оптимизации является применение сортировки подсчетом, которая в контексте медианного фильтра, где требуется найти медианное значение из набора значений окрестности пикселя, обеспечивает быструю и эффективную сортировку этого набора значений.

Список реализованных классов, представляющие алгоритмы, с набором дополнительных параметров:

1. Класс: BoxBlurFilter

Алгоритм: Box blur

Дополнительные параметры: радиус размытия

1. Класс: SeparableBoxBlurFilter

Алгоритм: Separable box blur

Дополнительные параметры: радиус размытия по оси X, Y, выбор режима размытия (только по горизонтали, только по вертикали, в обоих вариантах)

1. Класс: GaussianBlurFilter

Алгоритм: Размытие по Гауссу

Дополнительные параметры: радиус размытия, стандартное отклонение

1. Класс: SeparableGaussianBlurFilter

Алгоритм: Сепарабельное размытие по Гауссу

Дополнительные параметры: радиус размытия по оси X, Y, стандартное отклонение, выбор режима размытия (только по горизонтали, только по вертикали, в обоих вариантах)

1. Класс: MedianFilter

Алгоритм: Медианный фильтр

Дополнительные параметры: радиус, процентиль

1. Класс: LaplaceFilter

Алгоритм: Фильтр Лапласа

Дополнительные параметры: вариант матрицы свертки, состояние режима обрезки.

1. Класс: LaplaceSharpeningFilter

Алгоритм: Повышение резкости по Лапласу

Дополнительные параметры: множитель резкости

1. Класс: SobelFilter

Алгоритм: Оператор Собеля

Дополнительные параметры: не имеет

1. Класс: PrewittFilter

Алгоритм: Оператор Прюитт

Дополнительные параметры: не имеет

1. Класс: RobertsFilter

Алгоритм: Перекрестный оператор Робертса

Дополнительные параметры: не имеет

1. Класс: BrightnessOperation

Алгоритм: Изменение яркости через сложение

Дополнительные параметры: значение смещения по яркости

1. Класс: ExposureOperation

Алгоритм: Изменение яркости через умножение

Дополнительные параметры: множитель яркости

1. Класс: HueOperation

Алгоритм: Изменение оттенка в HSV

Дополнительные параметры: значение смещения по оттенку

1. Класс: SaturationOperation

Алгоритм: Изменение насыщенности в HSV

Дополнительные параметры: значение смещения по насыщенности

1. Класс: LightnessOperation

Алгоритм: Изменение светлости в HSL

Дополнительные параметры: значение смещения по светлости

1. Класс: GammaCorrectionOperation

Алгоритм: Гамма-коррекция

Дополнительные параметры: значение гаммы

1. Класс: ThresholdOperation

Алгоритм: Thresholding

Дополнительные параметры: левая граница, правая граница, режим (максимальное значение на всех каналах, только на красном канале, только на зеленом канале, только на синем канале)

1. Класс: GaussianNoiseOperation

Алгоритм: Добавление Гауссовского шума

Дополнительные параметры: среднее значение, стандартное отклонение

## Интерфейс программы

Программа представляет собой систему обработки и фильтрации цифровых изображений, организованную в виде узлов и соединительных линий, что позволяет пользователям создавать сложные обработочные цепочки. В рамках этой узловой системы, каждый узел представляет собой совокупность данных, операций или алгоритм обработки и фильтрации цифровых изображений.

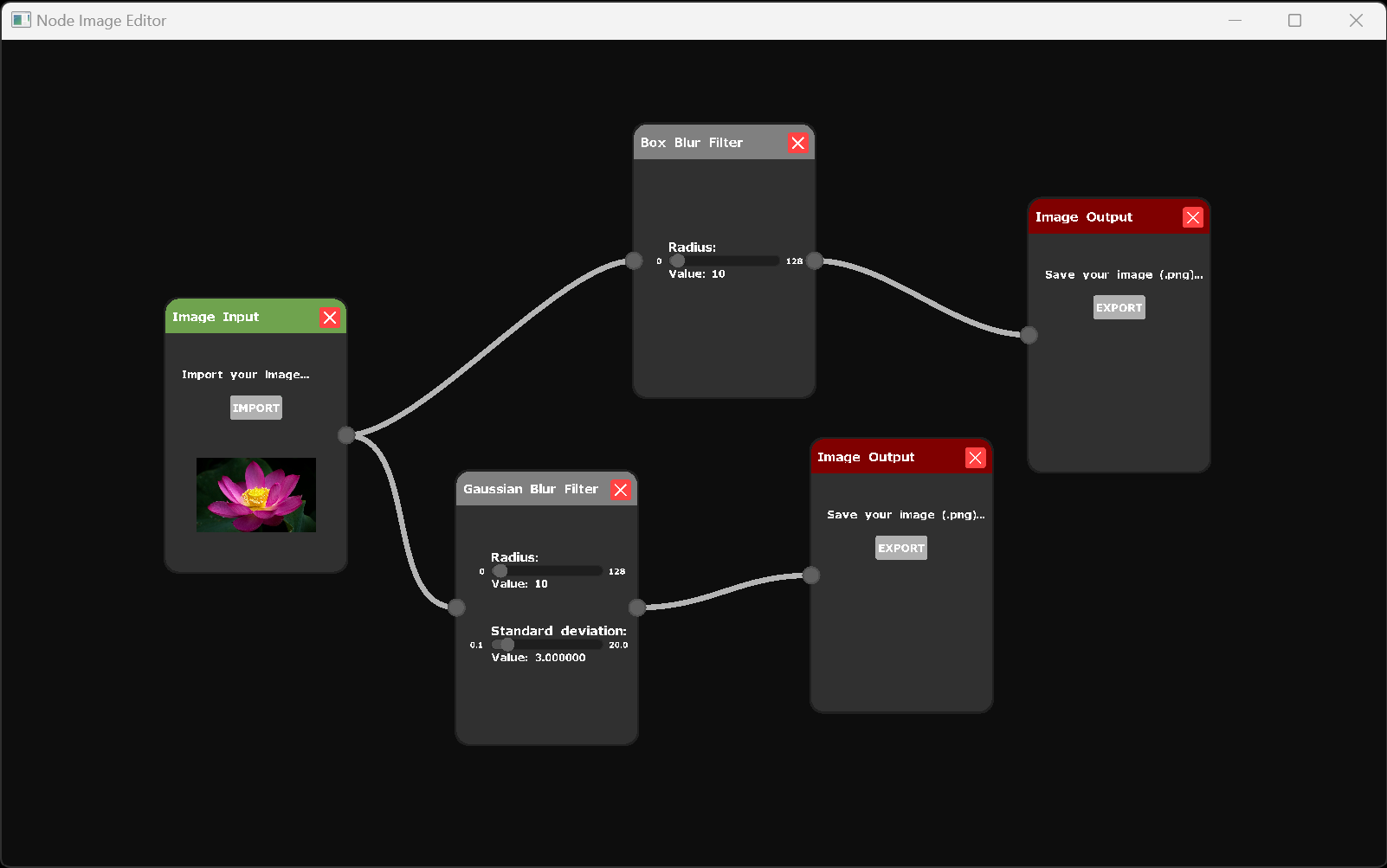


Рисунок 2 – внешний вид программы

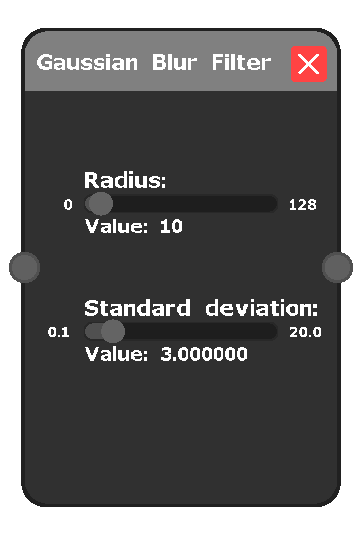


Рисунок 3 – графическое представление узла, соответствующий алгоритму размытия по Гауссу

Узлы могут быть соединены между собой с помощью соединительных линий, что позволяет передавать данные от одного узла к другому. Подобно потоку данных, эти линии обеспечивают передачу промежуточных результатов обработки изображений между различными алгоритмами. Такая узловая структура позволяет гибко комбинировать различные алгоритмы обработки для создания сложных цепочек обработки изображений.

Каждый узел в системе обладает своим уникальным названием и имеет возможность быть перетаскиваемым по пользовательскому интерфейсу. Кроме того, каждый узел обладает входными и выходными портами, через которые осуществляется передача данных. Входные порты принимают входные данные для алгоритма обработки, в то время как выходные порты предоставляют результаты его работы для последующих узлов. Это обеспечивает легкость в создании и модификации обработочных цепочек, поскольку пользователь может легко добавлять, удалять и изменять порядок узлов в сети.

Программа обладает функциональностью изменения масштаба и перемещения по узловой сети, что обеспечивает пользователю удобное и эффективное взаимодействие с алгоритмами обработки и фильтрации цифровых изображений.

Базовым классом всех пользовательских элементов является класс UIComponent. Одной из ключевых особенностей этого класса является наличие слушателей событий, которые позволяют элементам реагировать на действия пользователя.

Основные классы пользовательских элементов, которые являются составляющими для более сложных пользовательских элементов:

1. ShapeView: Этот класс представляет пользовательский элемент, отображающий графическую форму, такую как обычный прямоугольник, прямоугольник с закругленными углами или круг.
2. TextureView: Данный класс представляет элемент интерфейса, отображающий текстуры или изображения. Он позволяет интегрировать графические ресурсы в пользовательский интерфейс.
3. TextView: Этот класс представляет элемент интерфейса, отображающий текстовую информацию. Он позволяет выводить текст различного формата и размера на экране, что делает его одним из основных компонентов пользовательского интерфейса для отображения информации пользователю.

За осуществление управления всеми пользовательскими элементами отвечает класс UIManager. Данный класс хранит все элементы пользовательского интерфейса, отвечает за их отрисовку на экране и обрабатывает их в правильном порядке. UIManager обеспечивает централизованное управление всеми аспектами пользовательского интерфейса, включая расположение элементов, их взаимодействие и обновление в ответ на действия пользователя или изменения в системе.

Для запуска программы необходимо выполнить файл NodeImageEditor.exe. После запуска пользователь имеет возможность перемещать и масштабировать элементы по пользовательскому интерфейсу, изменять размеры окна, а также перетаскивать узлы и взаимодействовать с другими элементами с помощью мыши или сенсорной панели. Вызов меню с алгоритмами обработки и фильтрации цифровых изображений осуществляется при нажатии на правую кнопку мыши в любой месте пользовательского интерфейса.

## Тестирование

Рассмотрим фильтр размытия (box blur) и разделяемый фильтр размытия (separable box blur) на примере входного изображения:



Рисунок 4 – изображение лотоса с разрешением 708x442

Применим фильтр размытия (box blur) с радиусом, равным 10, к исходному изображению, представленному на рисунке 4, получим следующий результат:



Рисунок 5 – изображение лотоса после применения фильтра размытия (box blur)

Далее применим разделяемый фильтр размытия (separable box blur) с радиусом, равным 10 по оси X и Y, и различными режимами к изображению, представленном на рисунке 4, получим следующие результаты:



Рисунок 6 – изображение лотоса после применения фильтра размытия (separable box blur) по горизонтали



Рисунок 7 – изображение лотоса после применения фильтра размытия (separable box blur) по вертикали



Рисунок 8 – изображение лотоса после применения фильтра размытия (separable box blur) по горизонтали и вертикали одновременно

Заметим, что полученные результаты, изображенные на рисунках 5 и 8, визуально ничем не отличаются, однако примененные к ним алгоритмы имеют разную временную сложность.

Представление программы для тестирования данных алгоритмов имеет следующий вид:

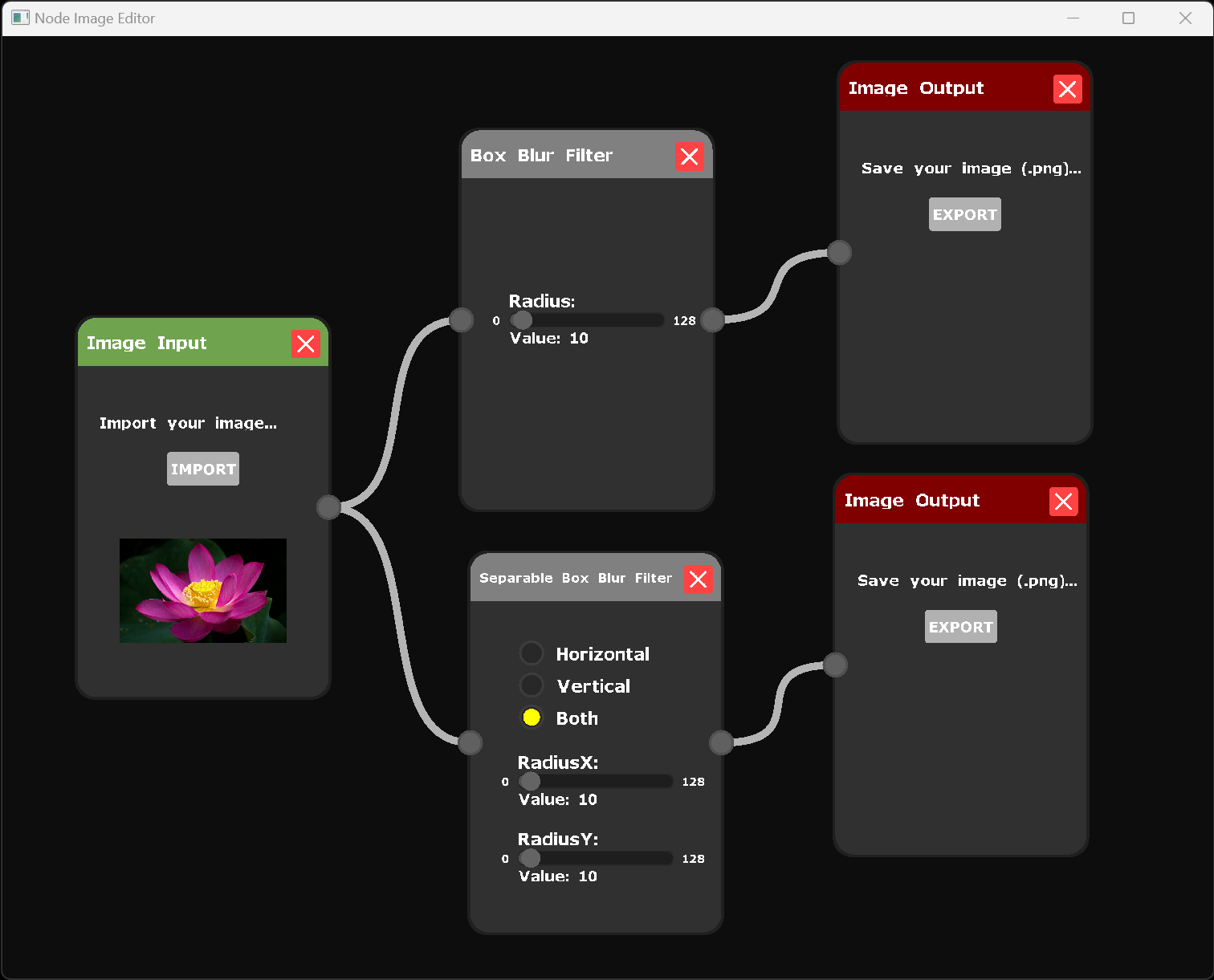


Рисунок 9 – вид программы при применении фильтра размытия (box blur) и разделяемого фильтра размытия (separable box blur)

Рассмотрим размытие по Гауссу и разделяемое размытие по Гауссу на примере следующего входного изображения:



Рисунок 10 – изображение незабудки с разрешением 604x402

Применим размытие по Гауссу с радиусом, равным 10, к исходному изображению, представленному на рисунке 10, получим следующий результат:



Рисунок 11 – изображение незабудки после применения размытия по Гауссу

Далее применим разделяемое размытие по Гауссу с радиусом, равным 10 по оси X и Y, и различными режимами к изображению, представленном на рисунке 10, получим следующие результаты:



Рисунок 12 – изображение незабудки после применения разделяемого размытия по Гауссу по горизонтали



Рисунок 13 – изображение незабудки после применения разделяемого размытия по Гауссу по вертикали



Рисунок 14 – изображение незабудки после применения разделяемого размытия по Гауссу по горизонтали и вертикали одновременно

Полученные результаты, изображенные на рисунках 11 и 14, визуально совпадают, но примененные к ним алгоритмы имеют разную временную сложность.

При применении данных алгоритмов программа выглядит следующим образом:

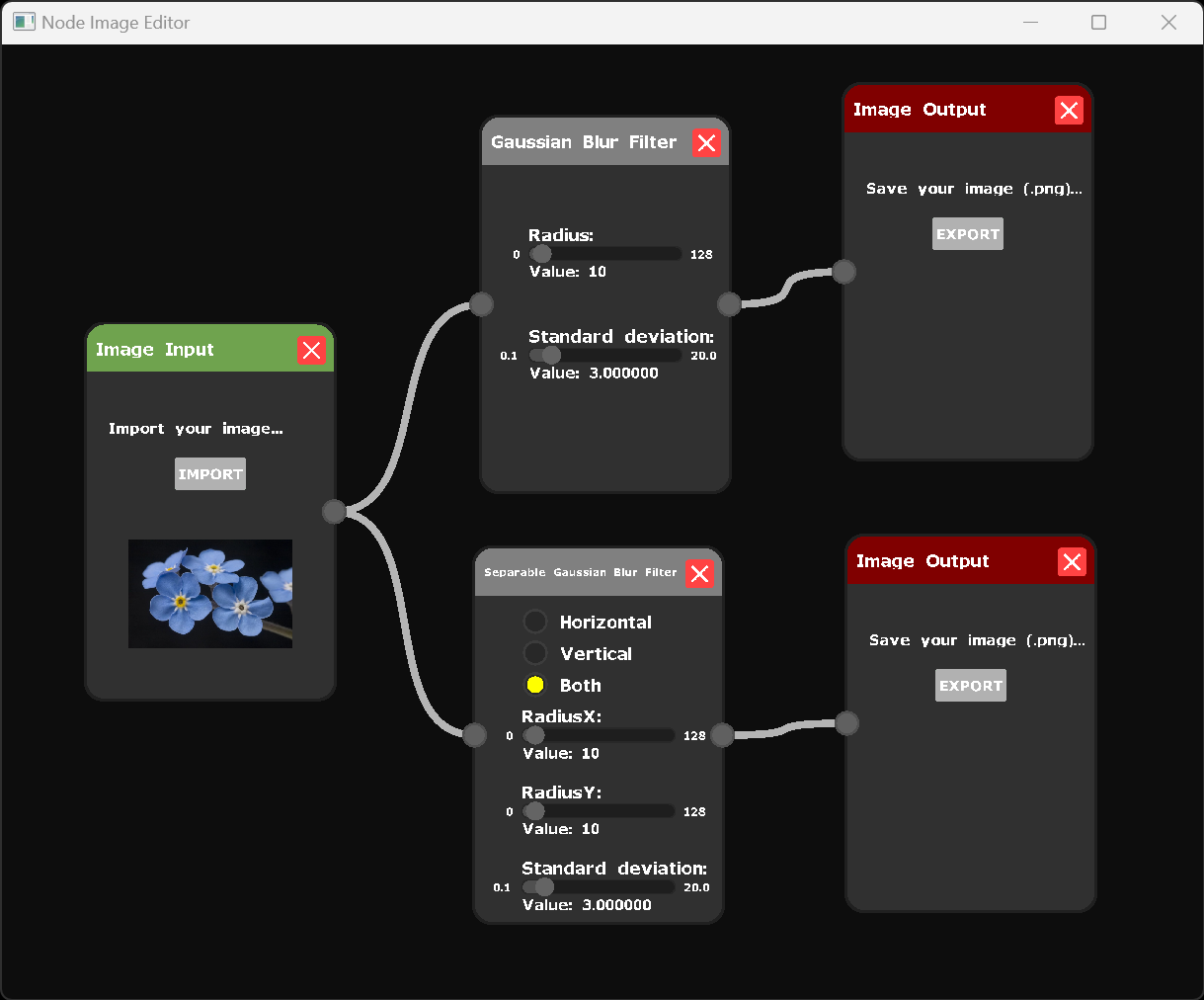


Рисунок 15 – вид программы при применении размытия по Гауссу и разделяемого размытия по Гауссу

Рассмотрим медианный фильтр со следующим входным изображением:



Рисунок 16 – подверженное шумом изображение с разрешением 206x214

Применим данный фильтр к исходному изображению, представленному на рисунке 16, с радиусом и процентилем, равным 1, получим следующий результат:



Рисунок 17 – подверженное шумом изображение после применения медианного фильтра

Программа с использованием медианного фильтра выглядит следующим образом:

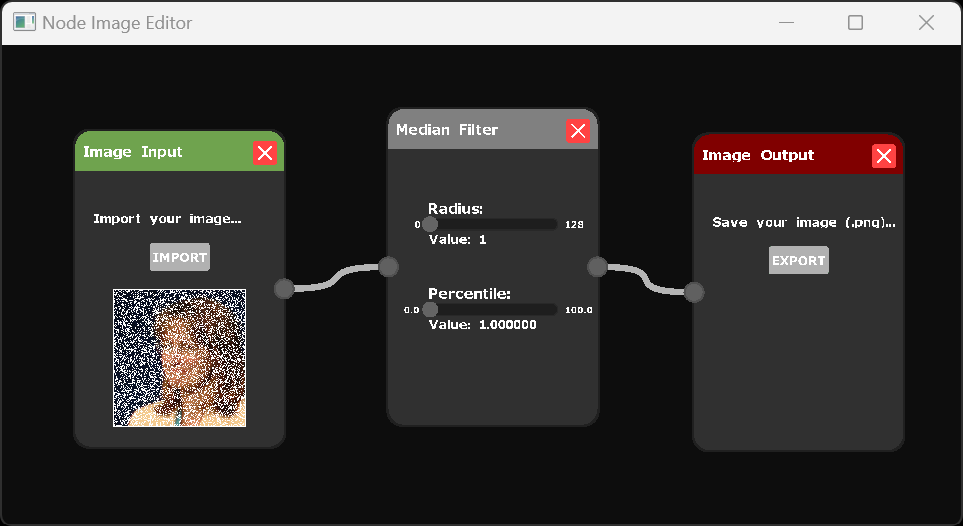


Рисунок 18 – вид программы при применении медианного фильтра

Рассмотрим фильтр Лапласа на примере следующего изображения:



Рисунок 19 – изображение строения с разрешением 370x300

Применим данный фильтр к исходному изображению, представленному на рисунке 19, с двумя вариантами матрицами свертки и с включенным режимом обрезки для первого варианта:



Рисунок 20 – изображение строения после применения фильтра Лапласа с первым вариантом матрицы свертки

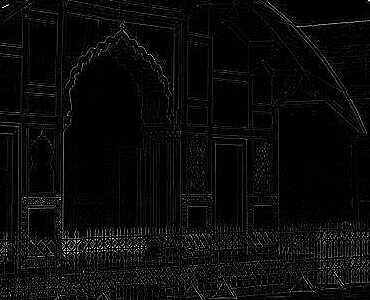


Рисунок 21 – изображение строения после применения фильтра Лапласа с первым вариантом матрицы свертки с режимом обрезки



Рисунок 22 – изображение строения после применения фильтра Лапласа со вторым вариантом матрицы свертки

Представление программы с применением фильтра Лапласа имеет следующий вид:

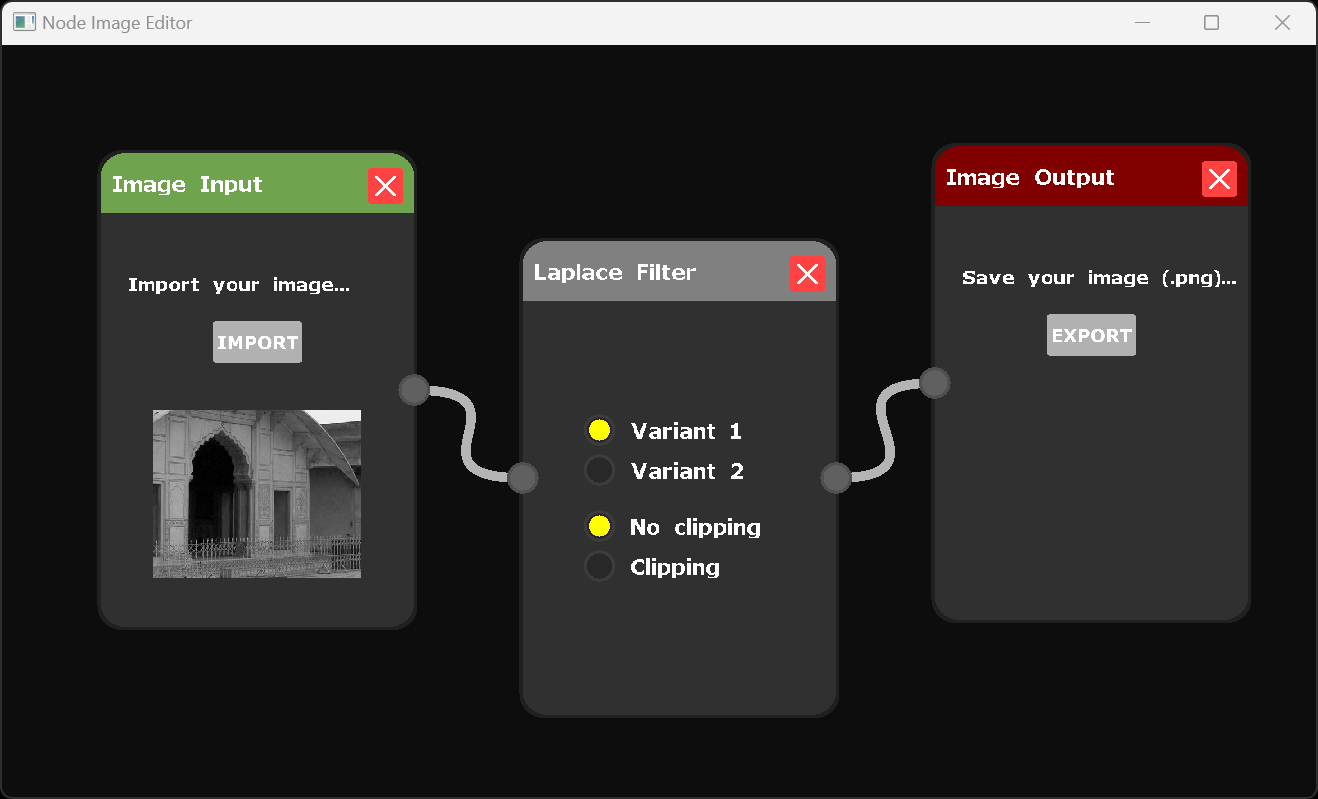


Рисунок 23 – вид программы при применении фильтра Лапласа

Рассмотрим алгоритм повышения резкости по Лапласу, применив его к следующему изображению:



Рисунок 24 – изображение луны с разрешением 900x900

Получим следующий результат при множителе резкости, равному 5:



Рисунок 25 – изображение луны после применения повышения резкости по Лапласу

Программа с использование алгоритма повышения резкости по Лапласу выглядит следующим образом:

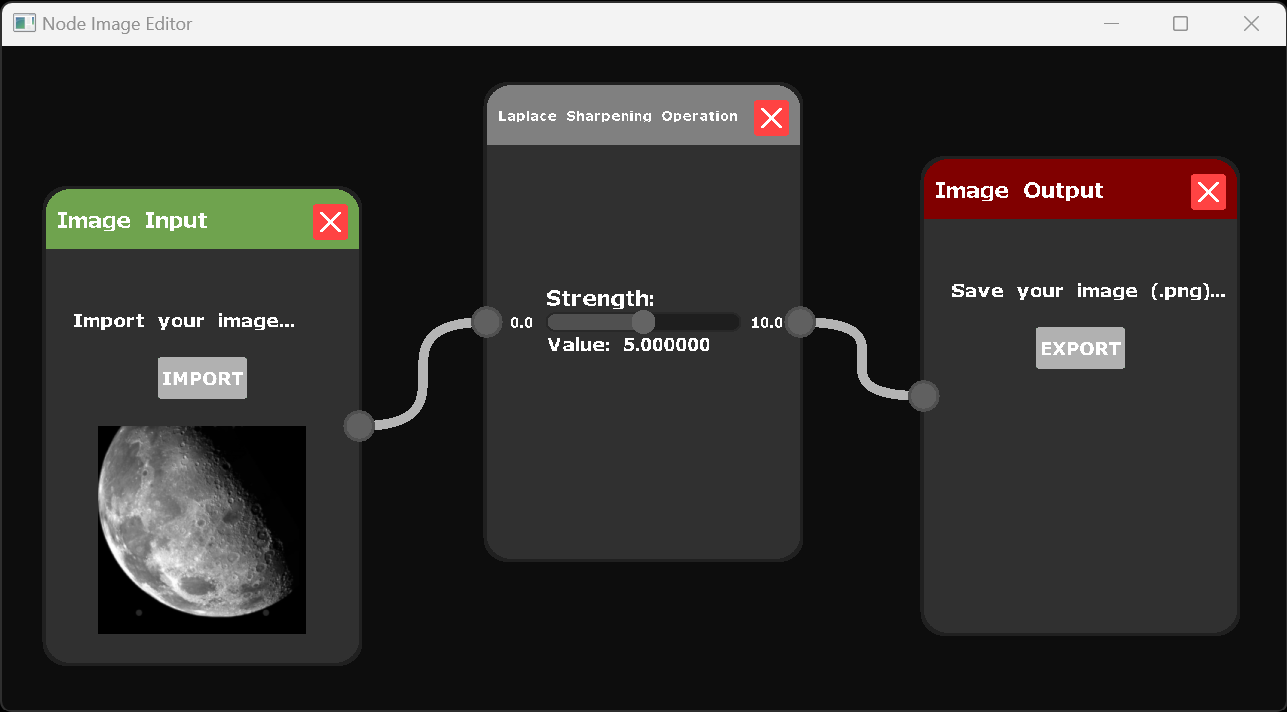


Рисунок 26 – вид программы при применении алгоритма повышения резкости по Лапласу

Рассмотрим оператор Собеля, оператор Прюитт и перекрёстный оператор Робертса на примере входного изображения, представленного на рисунке 19, применив их, получим следующие результаты:



Рисунок 27 – изображение строения после применения оператора Собеля



Рисунок 28 – изображение строения после применения оператора Прюитт



Рисунок 29 – изображение строения после применения перекрёстного оператора Робертса

Программа выглядит при данных алгоритмах следующим образом:

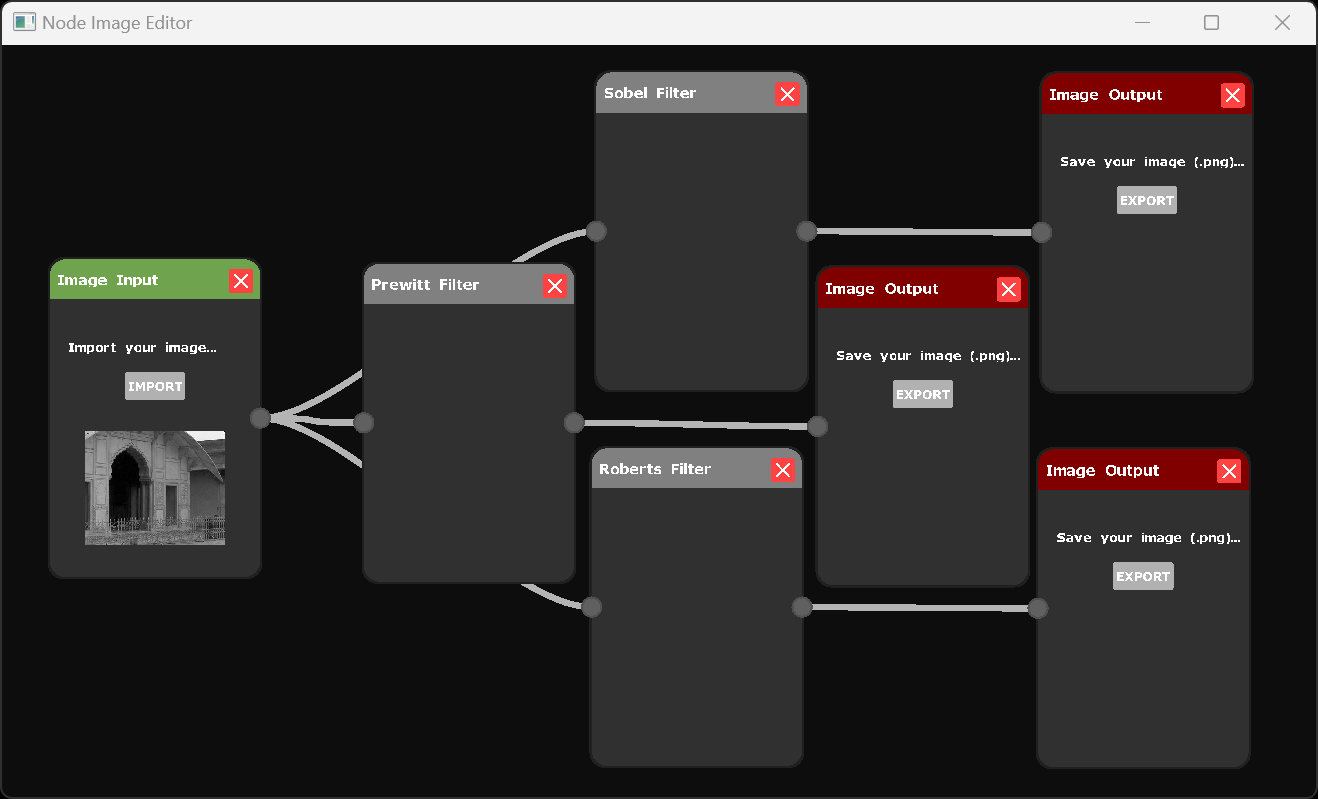


Рисунок 30 – вид программы при применении оператора Собеля, оператора Прюитта, перекрёстного оператора Робертса

Рассмотрим алгоритмы изменения яркости, оттенка в HSV, насыщенности в HSV и светлости в HSL, применив их к изображению, представленному на рисунке 10, получим следующие результаты:



Рисунок 31 – изображение незабудки после изменения яркости через сложение со смещением, равным 65



Рисунок 32 – изображение незабудки после изменения яркости через умножение с множителем, равным 2



Рисунок 33 – изображение незабудки после изменения оттенка со смещением, равным 140



Рисунок 34 – изображение незабудки после изменения насыщенности со смещением, равным -1



Рисунок 35 – изображение незабудки после изменения светлости со смещением, равным -0.2

Программа в этом случае имеет вид:

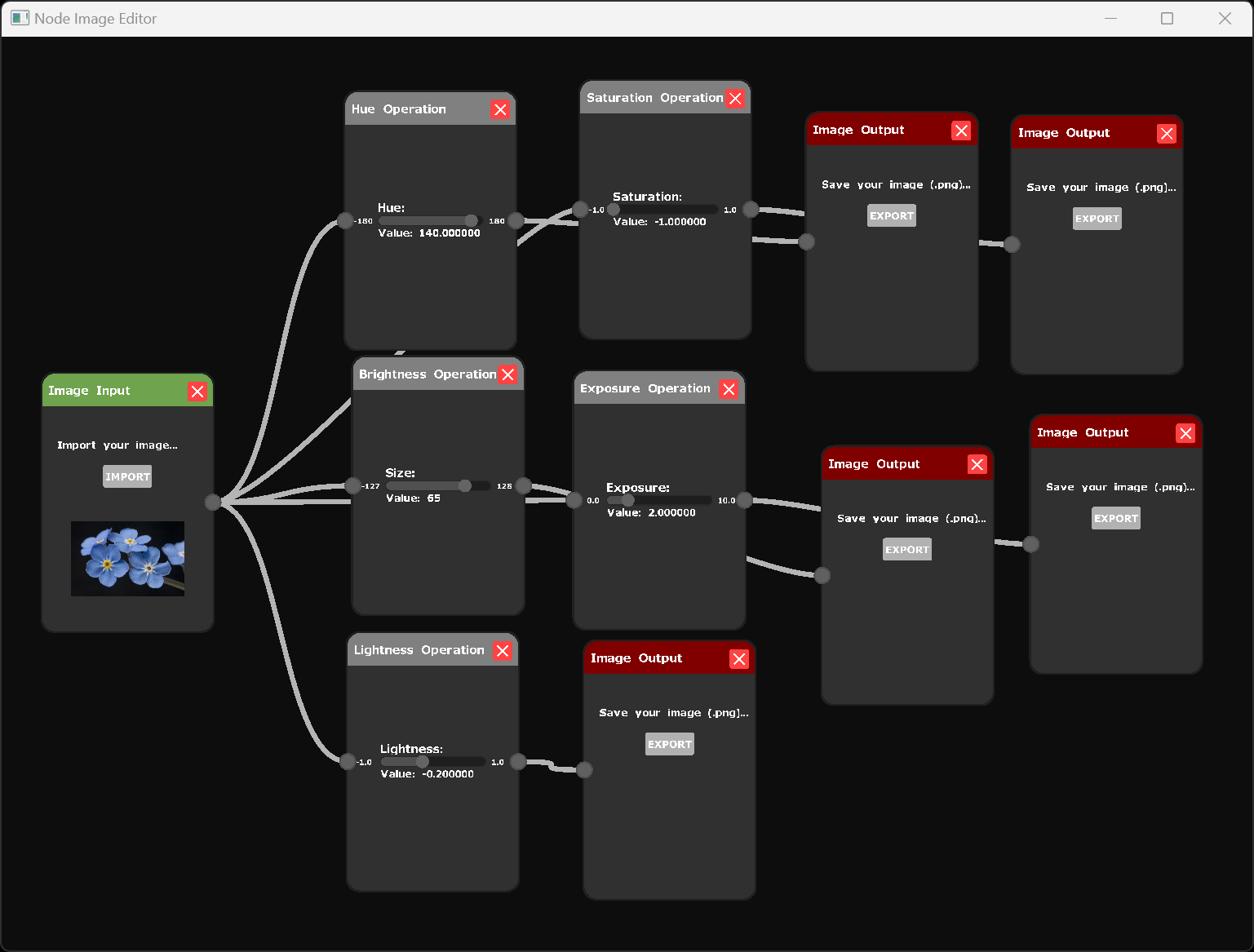


Рисунок 36 – вид программы при применении изменения яркости, оттенка в HSV, насыщенности в HSV и светлости в HSL

Проведенное также тестирование программы с применением алгоритмов гамма коррекции, формирования порога (thresholding) и добавления гауссовского шума показало успешные результаты, соответствующие ожиданиям. В ходе испытаний были проверены различные параметры, и все алгоритмы продемонстрировали стабильную работу.

# ЗАКЛЮЧЕНИЕ

В ходе данной курсовой работы были проведены исследования и разработаны алгоритмы обработки и фильтрации цифровых изображений. Это включало в себя изучение основных методов обработки изображений, таких как фильтрация, подавление шума, выделение краев и другие техники. На основе полученных знаний были разработаны соответствующие алгоритмы, которые были реализованы в виде программы с удобным интерфейсом, что позволяет пользователям легко применять их к своим изображениям.

Важным этапом работы являлось тестирование программы, в ходе которого было проверено её функциональность, надёжность и эффективность. Результаты тестирования подтвердили работоспособность программы и соответствие алгоритмов заданным требованиям.

Разработанная программа для обработки и фильтрации цифровых изображений представляет собой универсальный инструмент, который может быть использован во многих сферах. Она может быть полезна для специалистов в области компьютерного зрения, обработки изображений, медицинской диагностики, графического дизайна, а также в научных и исследовательских работах.

Однако проект не ограничивается созданием программы. В дальнейших перспективах представляется возможность расширения функционала программы путём добавления новых алгоритмов обработки и фильтрации, а также улучшения существующих. Кроме того, возможна оптимизация алгоритмов для повышения скорости работы программы и качества обработки изображений. Не менее важным является улучшение интерфейса программы, чтобы сделать его ещё более интуитивно понятным и удобным для пользователей.

Таким образом, проект представляет собой важный шаг в области обработки цифровых изображений и имеет потенциал для дальнейшего развития и совершенствования.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Интернет-ресурс: https://en.wikipedia.org/wiki/Box\_blur
2. Интернет-ресурс: https://en.wikipedia.org/wiki/Gaussian\_blur
3. Интернет-ресурс: https://en.wikipedia.org/wiki/Median\_filter
4. Интернет-ресурс: https://ru.wikipedia.org/wiki/Медианный\_фильтр
5. Digital Image Processing Third Edition, Rafael C. Gonzalez, Richard E. Woods, 2008, Pearson Education
6. Интернет-ресурс: https://ru.wikipedia.org/wiki/Оператор\_Собеля
7. Интернет-ресурс: https://en.wikipedia.org/wiki/Prewitt\_operator
8. Интернет-ресурс: https://ru.wikipedia.org/wiki/Перекрёстный\_оператор\_Робертса
9. Интернет-ресурс: https://en.wikipedia.org/wiki/HSL\_and\_HSV
10. Интернет-ресурс: https://ru.wikipedia.org/wiki/Гамма-коррекция
11. Интернет-ресурс: https://ru.wikipedia.org/wiki/Гауссовский\_шум
12. OpenGL Programming Guide Seventh Edition: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1, Dave Shreiner, 2009, Addison-Wesley
13. Интернет-ресурс: http://glew.sourceforge.net
14. Интернет-ресурс: https://www.glfw.org
15. Интернет-ресурс: https://github.com/planetack/stb\_image
16. Using OpenMP Portable Shared Memory Parallel Programming, Barbara Chapman, Gabriele Jost, Ruud van der Pas, 2007, MIT Press
17. Интернет-ресурс: https://learn.microsoft.com/en-us/windows/win32