# 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Предметная область проекта

Предметная область проекта связана с разработкой десктопного приложения, предназначенного для расчета маршрута полета сельскохозяйственного дрона. Основной задачей приложения является определение оптимального маршрута полета, учитывающего имеющиеся характеристики дрона и заданные параметры полета.

**Анализ аналогичных проектов и существующих решений для   
реализации проекта**

В рамках разработки десктопного приложения для расчета маршрута сельскохозяйственного дрона по имеющимся характеристикам, необходимо реализовать возможность планировки маршрута для фотографирования посевов полей. Эти данные, после планировки маршрута, уже передаются в другое приложение для склейки фотографий поверхности и, при необходимости, анализа. Важным аспектом при планировке маршрута будет учет размеров поля и наличие препятствий на его территории. Кроме того, приложение должно иметь возможность визуализировать маршрут на карте и предоставлять информацию о параметрах полета, таких как высота, длительность полета и т.д. Все эти функции должны быть реализованы в удобном и интуитивно понятном интерфейсе, который позволит пользователям быстро и эффективно планировать маршруты для фотографирования посевов полей.

Существует множество инструментов для планирования полетов дронов и обработки полученных данных, которые могут быть полезны при разработке десктопного приложения для расчета маршрута сельскохозяйственного дрона по имеющимся характеристикам. Рассмотрим некоторые из них.

DroneDeploy [2] – это один из наиболее распространенных инструментов для планирования полетов дронов и обработки данных. С помощью этого приложения пользователь может задавать параметры полета, такие как высоту, скорость, угол наклона камеры и другие, а также задавать целевые точки на карте, которые дрон должен посетить. Приложение автоматически рассчитывает оптимальный маршрут полета и предоставляет возможность обработки полученных данных, например, создание карт и 3D-моделей местности.

Litchi [3] – еще один инструмент для планирования полетов дронов. Он позволяет пользователю задавать различные параметры полета, а также создавать специальные миссии, включающие несколько точек на карте, которые дрон должен посетить в определенном порядке. Litchi также предоставляет возможность записи видео и фото во время полета, а также позволяет управлять камерой дрона в режиме реального времени.

Pix4D Capture [4] – специализированный инструмент для создания карт и 3D-моделей с помощью дронов. Приложение позволяет задавать различные параметры полета, такие как высоту, скорость и угол наклона камеры, а также задавать целевые точки на карте. После полета Pix4D Capture обрабатывает полученные данные и создает точные 3D-модели местности.

UgCS [5] – это инструмент, который позволяет пользователю управлять не только дронами, но и другими беспилотными системами. Пользователь может задавать различные параметры полета, такие как скорость, высота полета и угол наклона камеры, а также задавать целевые точки на карте. Приложение автоматически рассчитывает оптимальный маршрут полета и позволяет обрабатывать полученные данные.

Для сравнительного анализа возможностей описанных продуктов можно привести таблицу, в которой будут указаны такие критерии сравнения как, планирование маршрута полета, управление полетом дрона, обработка полученных данных, визуализация карты, наличие ограниченного ряда поддерживаемых дронов, добавление собственных дронов, поддерживаемые OC, лицензия, стоимость (таблица 1).

Таблица 1 – Сравнительный анализ приложений

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Возможность** | **DroneDeploy** | **Litchi** | **Pix4D Capture** | **UgCS** |
| Планирование маршрута полета | Да | Да | Да | Да |
| Управление полетом дрона | Да | Да | Да | Да |
| Обработка полученных данных | Да | Да | Да | Да |
| Визуализация карты | Да | Да | Да | Да |
| Ограниченный ряд поддерживаемых дронов | Да | Да | Да | Да |
| Добавление собственных дронов | Ограничение функционала | Ограничение функционала | Нет | Ограничение функционала |
| Поддержка OC | iOS, Android + Windows, macOS, Linux | iOS, Android + Windows, macOS, Linux | iOS, Android | Windows, macOS, Linux, Android |
| Лицензия | Проприетарная | Проприетарная | Проприетарная | Проприетарная |
| Стоимость | $149+/месяц | $25 | Бесплатно | €790+ или €149+/месяц |

Рассмотрев аналогичные проекты и существующие решения для реализации проекта, можно заметить, что многие из них имеют проприетарную лицензию и не предоставляют возможность добавления собственных дронов. Кроме того, некоторые из них могут быть слишком дорогими для малого бизнеса, к тому же, стоит отметить, что все анализируемые аналоги приложения разработаны зарубежными компаниями. В свою очередь, наше приложение будет предоставлять открытый исходный код, что позволит пользователям настраивать и дополнять его функциональность. В приложении будет возможность добавления собственных дронов, что поможет людям с моделями дронов от малоизвестных компаний. Кроме того, распространение приложения будет бесплатным, что обеспечит его доступность и удобство использования для малых предприятий в сельском хозяйстве.

В целом, все эти решения могут быть полезны при разработке десктопного приложения для расчета маршрута сельскохозяйственного дрона по имеющимся характеристикам. Также необходимо обеспечить возможность интеграции с другими системами, например, с системами отправки маршрута на дрон или экспорта в уже существующие форматы представления координат.

1.2. Особенности реализации

Для реализации десктопного приложения для расчета маршрута сельскохозяйственного дрона по имеющимся характеристикам, было принято решение использовать веб технологии. Такой подход позволяет обеспечить кроссплатформенность приложения и увеличить его доступность для пользователей.

Для реализации приложения будет использоваться Tauri [6]. Он выбран из-за своей высокой производительности и возможности создания кроссплатформенных десктопных приложений на основе веб-технологий. Tauri, обеспечивает быстрое и эффективное взаимодействие между ядром приложения и веб-интерфейсом.

Для обработки данных будет использоваться язык программирования Rust [7]. Rust позволяет обеспечить высокую производительность и безопасность при обработке данных, что критически важно для такого типа приложений. Более того, Rust имеет большую и быстрорастущую экосистему, что обеспечивает доступность и готовность библиотек и инструментов, необходимых для реализации конкретных функциональных требований приложения.

В качестве фреймворка для разработки пользовательского интерфейса будет использоваться SvelteKit [8]. Этот инструмент позволяет создавать эффективные и быстрые веб-приложения с помощью компиляции кода во время сборки и минимизации размера бандла приложения.

Для визуализации карты будет использована OpenLayers [9] – библиотека для работы с картами. Она предоставляет широкий спектр функциональности для работы с картами, таких как отображение маршрутов, меток, векторных слоев и т.д. Это позволит создать интерактивную карту, на которой можно будет планировать маршруты для фотографирования посевов полей. Для ограничения поля можно использовать полигон, а для мест съемки точки.

Таким образом, использование Tauri, Rust, SvelteKit и OpenLayers обеспечивает высокую производительность, безопасность, готовность библиотек и инструментов, а также удобную и эффективную визуализацию карты. Все эти особенности сделают приложение более доступным для пользователей, а также обеспечат надежность в работе и обеспечивает простоту разработки.

**Вывод по первой главе**

Анализ предметной области и существующих работ по тематике выпускной квалификационной работы показал, что было бы полезно разработать приложение, предоставляющее возможность расчета маршрута сельскохозяйственного дрона по имеющимся характеристикам для мониторинга полей. Также был выявлен набор инструментальных средств для реализации поставленной задачи, наиболее полно удовлетворяющий требованиям к подобного рода системам. Было принято решение реализовать систему на основе на базе технологий Tauri, Rust, SvelteKit и OpenLayers.

# 2. АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОЙ СИСТЕМЕ

2.1. Основные требования

В результате анализа предметной области и обзора существующих аналогов были сформированы следующие два основных типа требований:

* функциональные требования – перечень сервисов, которые должна выполнять система;
* нефункциональные требования – описание характеристики системы и ее окружения, также содержит перечень ограничений.

**Функциональные требования к проектируемой системе**

1. Пользователь должен иметь возможность добавлять, просматривать, изменять и удалять записи о дронах и камерах в базе данных. Через пользовательский интерфейс системы.
2. Система должна иметь возможность строить маршрут на основе заданных характеристик дрона, камеры, координат старта съемки, координат границ, указываемых на карте и процента перекрытия полученных снимков.
3. Система должна иметь возможность отображать маршрут на карте.

**Нефункциональные требования к проектируемой системе**

1. Система должна использовать язык программирования Rust.
2. Система должна проверять корректность вводимых данных.
   1. Величина полезной нагрузки дрона лежит в пределах: от 100 грамм до 10 кг.
   2. Скорость полета дрона находится в пределах от 0.1 м/с до 50 м/с.
   3. Продолжительность полета дрона находится в пределах от 1 минуты до 8 часов.
   4. Минимальная и максимальная высота полета находятся в пределах от 2 метров до 5 км. При этом, максимальная высота полета больше минимальной.
   5. Камера, установленная на дроне либо отсутствует, либо принадлежит списку доступных камер.
   6. Масса камеры должна быть меньше 10 килограмм.
   7. Углы обзора камеры по х и у должны быть меньше 180 градусов.
   8. Разрешение съемки камеры по х и у – целочисленные.
3. Система должна использовать веб технологии такие как Tauri, Svelte, TypeScript для визуализации интерфейса и библиотеку OpenLayers для отображения карты и маршрута.

2.2. Диаграмма вариантов использования

На основе требований, предъявляемых к разрабатываемому приложению, были разработаны варианты его использования, которые представлены на рисунке 1.

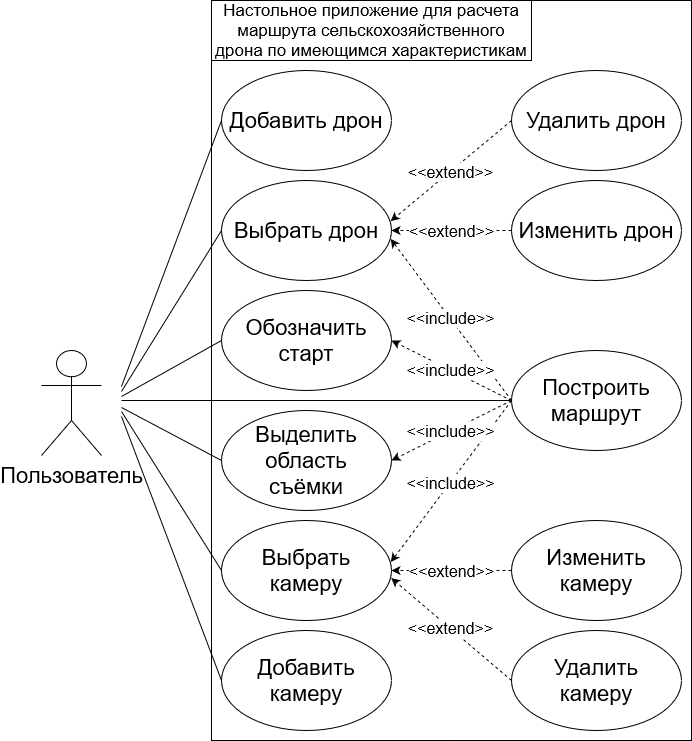


Рисунок 1 – Диаграмма вариантов использования

Актером является пользователь приложения, которому доступна возможность использовать весь функционал приложения.

Пользователь может совершать следующие действия:

1. добавить дрон – пользователь может добавить новый дрон с его характеристиками;
2. выбрать дрон – пользователь может выбрать один из добавленных дронов для расчета маршрута или редактирования;
3. удалить дрон – пользователь может удалить выбранный дрон;
4. изменить дрон – пользователь может изменить характеристики выбранного дрона;
5. обозначить старт – пользователь может задать стартовую точку маршрута;
6. выделить область съемки – пользователь может указать область, которую требуется снять с помощью дрона;
7. выбрать камеру – пользователь может выбрать камеру, которая будет установлена на дроне для расчетов или редактирования;
8. изменить камеру – пользователь может изменить характеристики выбранной камеры;
9. удалить камеру – пользователь может удалить выбранную камеру;
10. добавить камеру – пользователь может добавить новую камеру с ее характеристиками;
11. построить маршрут – пользователь может рассчитать маршрут дрона на основе введенных ранее данных.

Спецификации ВИ представлены в таблицах 1–11 приложения А.

**Вывод по второй главе**

В результате анализа предметной области и обзора существующих аналогов были сформированы функциональные и нефункциональные требования. Основываясь на требованиях, предъявляемых к разрабатываемому приложению, была разработана диаграмма вариантов использования приложения и их спецификации.

# 3. АРХИТЕКТУРА СИСТЕМЫ

3.1. Общее описание архитектуры системы

На рисунке 2 изображена диаграмма компонентов системы для планирования маршрутов сельскохозяйственных дронов. Система состоит из следующих сервисов и компонентов:

* сервис расчета маршрутов (Algorithms);
* сервис управления данными (Data Managment);
* компонент пользовательского интерфейса карты (Map);
* компонент пользовательского интерфейса левого меню (LMenu);
* компонент пользовательского интерфейса правого меню (RMenu), в состав которого входит сервис вызова функций расчета (Calculate);
* компонент базы данных (SQLite Database).

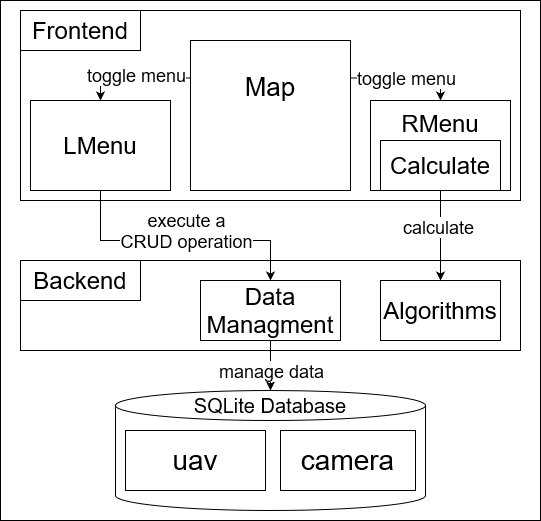


Рисунок 2 – Компоненты системы

Система включает в себя компонент базы данных, который содержит данные о дронах и камерах. Сервис расчета маршрутов использует данные, предоставленные пользователем через компоненты интерфейса. Сервис управления данными осуществляет взаимодействие с базой данных для создания, чтения, обновления и удаления данных о дронах и камерах. Компоненты пользовательского интерфейса взаимодействуют с сервисами для отправки данных, получения результатов расчетов и отображения информации пользователю.

3.2. Описание компонентов и сервисов, составляющих систему

Map – центральное меню с картой, компонент пользовательского интерфейса в его функционал входят:

* отображение карты и маршрута;
* переключение отображения правого и левого меню.

LMenu – этот компонент позволяет пользователю работать с дронами и камерами, он предлагает следующий набор функций:

* выбор дрона для построения маршрута;
* CRUD операции над дроном;
* выбор камеры для построения маршрута;
* CRUD операции над камерой.

RMenu – этот компонент позволяет пользователю управлять параметрами миссии, он предоставляет пользователю данный функционал:

* ввод высоты полета;
* выбор алгоритма расчета;
* панель инструментов для рисования маршрута;
* кнопка вызова Calculate;
* отображение текущих параметров миссии;
* экспорт маршрута в формат GeoJSON.

Calculate – часть сервиса Map он включает в себя:

* валидацию данных, он проверяет корректность ввода всех необходимых данных;
* отправку данных на бэкенд для расчетов;
* обновление отображаемого маршрута на карте.

Data Managment – этот сервис взаимодействует с базой данных SQLite, управляя данными о дронах и камерах. Он выполняет операции CRUD на основе команд с фронтенда.

Algorithms – это сервис расчета маршрута, он отвечает за расчет маршрута дрона. Детальное описание алгоритмов представлено в главе 4. Этот сервис обладает следующим набором возможностей:

* выполнение дискретизация области;
* выполнение алгоритма ближайшего соседа;
* выполнение алгоритма полного перебора;
* выполнение алгоритма прямоугольных областей;
* отправка результатов на фронтенд.

SQLite Database – база данных, является частью инфраструктуры системы, в которой находятся 2 таблицы uav и camera для хранения данных о дронах и камерах, хранится локально в файле mydatabase.db.

3.3. Модель базы данных

На рисунке 3 представлена модель базы данных приложения для планирования маршрутов сельскохозяйственных дронов. Она состоит из двух таблиц – uav, отвечающей за дроны и camera отвечающей за камеры. Эти таблицы содержат информацию о различных дронах и камерах, которые могут использоваться в миссиях. Связь между таблицами один к одному. Связь указывает, какая камера установлена на дроне. При этом, камера может быть не установлена на дрон.

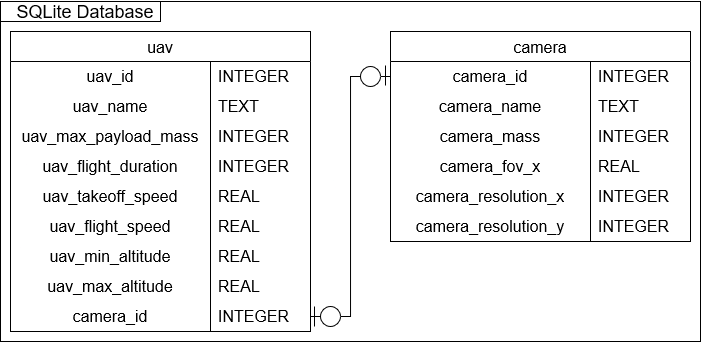


Рисунок 3 – Модель базы данных

Таблица uav – представляет собой экземпляр дрона и включает в себя следующие атрибуты:

* uav\_id – идентификатор дрона;
* uav\_name – название дрона;
* uav\_max\_payload\_mass – максимальная полезная нагрузка дрона в граммах;
* uav\_flight\_duration – средняя продолжительность полета в секундах;
* uav\_takeoff\_speed – средняя скорость взлета в метрах в секунду;
* uav\_flight\_speed – средняя скорость полета в метрах в секунду;
* uav\_min\_altitude – минимальная безопасная высота полета в метрах;
* uav\_max\_altitude – максимальная безопасная высота полета в метрах;
* camera\_id – идентификатор камеры, установленной на дрона, может быть не указан.

Таблица camera – представляет собой экземпляр камеры и включает в себя следующие атрибуты:

* camera\_id – идентификатор камеры;
* camera\_name – название камеры;
* camera\_mass – масса камеры в граммах;
* camera\_fov\_x – угол обзора камеры по оси в градусах;
* camera\_resolution\_x – разрешение камеры по оси в пикселях;
* camera\_resolution\_y – разрешение камеры по оси в пикселях.

3.4. Процесс работы с системой

На рисунке 4 приведена диаграмма деятельности, которая подробно описывает процесс работы с системой планирования маршрутов для сельскохозяйственных дронов.



Рисунок 4 – Диаграмма деятельности

Процесс делится на 3 основных потока. Первый начинается одного из шести действий. Пользователь может установить начальную точку маршрута, обозначить область съемки, которую необходимо заснять, может выбрать алгоритм, модель дрона и камеры из списка доступных в системе (алгоритм ближайшего соседа, алгоритм полного перебора, алгоритм, предназначенный для прямоугольных областей [13]), установить процент перекрытия и определить высоту полета дрона. Здесь у пользователя есть два варианта: ввести высоту вручную или рассчитать ее. Во втором случае пользователю необходимо ввести разрешение съемки, после чего система сама рассчитывает высоту полета. После того как все параметры заданы, система производит расчет маршрута. Если все данные корректны, и валидация прошла успешно, система отображает маршрут на карте и предлагает пользователю экспортировать его в формате GeoJSON для дальнейшего использования. GeoJSON [10] является форматом данных, основанным на JSON, и используется для представления простых географических объектов. GeoJSON файл, который экспортирует данное приложение включает в себя последовательность точек с координатами в формате WGS84, которые должен посетить дрон. Если в процессе валидации обнаруживаются ошибки, система отображает сообщение об ошибке, давая пользователю возможность исправить введенные данные.

В то же время, независимо от основного процесса планирования маршрута, пользователь может выбрать и изменить характеристики модели дрона или камеры.

**Вывод по третьей главе**

В третьей главе была описана архитектура системы. Система включает в себя различные компоненты и сервисы, которые взаимодействуют для обеспечения планирования маршрутов. Была представлена модель базы данных, состоящая из двух таблиц, содержащих информацию о дронах и камерах. Также был описан процесс работы с системой, включающий в себя различные этапы от выбора параметров миссии до экспорта маршрута.

# 4. ТЕОРЕТИЧЕСКОЕ ОПИСАНИЕ АЛГОРИТМОВ

4.1. Описание задачи и алгоритмов

Не существует эффективных алгоритмов для решения задачи коммивояжера. Задача коммивояжера формулируется так, бродячий торговец (коммивояжер) должен посетить n пунктов. Известна стоимость проезда между любыми двумя пунктами. Требуется выбрать наиболее «дешевый» замкнутый путь, проходящий через все пункты. Эта задача является классической задачей дискретной оптимизации и относится к классу NP-сложных задач. [11]

Если рассматривать эту задачу в контексте данной работы, можно провести соответствие, пунктами являются точки съемки дрона, а стоимость выражена в расстоянии между пунктами. Данное приложение предназначено для применения некоторых алгоритмов, представляющих приближенные решения или оптимальное решение для построения маршрута сельскохозяйственного дрона.

Задачу построения маршрута дрона можно рассматривать, как задачу на плоскости, потому как дрон летает на высоте достаточной, чтобы ему не препятствовали деревья и потому как основной целью съемки являются поля, отсутствуют горы. Поэтому процесс расчета маршрута поделить на несколько основных этапов:

* дискретизация области съемки для получения точек, на карте, в которых будет произведена съемка;
* построение маршрута на точках, полученных в результате дискретизации.

В качестве алгоритмов построения маршрутов были выбраны алгоритмы ближайшего соседа (Nearest neighbour) [12], полного перебора (Brute force) и алгоритм, включающий обработку прямоугольных областей (Rectangular areas). Сравнение алгоритмов представлено в таблице 2.

Таблица 2 – Сравнительный анализ алгоритмов построения маршрута

|  |  |  |  |
| --- | --- | --- | --- |
| **Алгоритм** | **Nearest neighbour** | **Brute force** | **Rectangular areas** |
| Входные данные | Одномерный массив точек | Одномерный массив точек | Трехмерный массив точек, стартовая точка, направление дискретизации |
| Выходные данные | Одномерный массив точек | Одномерный массив точек | Одномерный массив точек |
| Класс | Аппроксимационный алгоритм | Алгоритм оптимального решения | Аппроксимационный алгоритм |
| Ограничения | Нет | Количество точек съемки <13 | Количество полей <10, прямоуголькики, ограничивающие поля не пересекаются |
| Результат | Неоптимальный | Оптимальный | Неоптимальный, входят точки, не принадлежащие полю |
| Время работы алгоритма | O(N2) | O(N!) | Объединение полей O(ElogV), построение пути внутри поля O(N) |

Из таблицы 2 видно, что при различных входных данных лучше использовать тот или иной алгоритм, наиболее подходящий под нужды. Алгоритм ближайшего соседа может быть применен для большого количества точек, потому как его скорость выполнения очень высока, но его результат не является оптимальным. Алгоритм полного перебора применим лишь для случаев с минимальным количеством точек, потому как на его расчеты требуется несоизмеримо большое количество времени. Алгоритм для расчета прямоугольных областей может применен для большого количества точек и является достаточно точным для нужд построения маршрута для съемки сельскохозяйственным дроном, но его особенностью является то, что он применим лишь для непересекающихся прямоугольных областей. Для реализации данных алгоритмов, необходимо привести их высокоуровневое описание. Далее будут рассмотрены шаги каждого из алгоритмов подробнее.

4.2. Описание алгоритмов

**Алгоритм ближайшего соседа**

Входные данные:

* points: вектор пар чисел с плавающей точкой, представляющих координаты точек;
* start\_point: пара чисел с плавающей точкой, представляющая координаты начальной точки.

Выходные данные: вектор пар чисел с плавающей точкой, представляющих координаты точек в порядке обхода ближайшего соседа.

Шаги алгоритма.

1. Проверить, не пуст ли вектор points. Если пуст, вернуть ошибку.
2. Инициализировать remaining\_points как копию points, result как пустой вектор, и current\_point как start\_point.
3. Пока remaining\_points не пуст, выполнить следующие шаги.
   1. Найти индекс и значение ближайшей точки к current\_point в remaining\_points.
   2. Удалить ближайшую точку из remaining\_points.
   3. Если result пуст, добавить start\_point в result.
   4. Добавить ближайшую точку в result.
   5. Обновить current\_point как ближайшую точку.
4. Добавить start\_point в result.
5. Вернуть result как результат.

**Алгоритм полного перебора**

Входные данные:

* points: вектор пар чисел с плавающей точкой, представляющих координаты точек;
* start\_point: пара чисел с плавающей точкой, представляющая координаты начальной точки.

Выходные данные: вектор пар чисел с плавающей точкой, представляющих координаты точек в порядке обхода ближайшего соседа.

Шаги алгоритма.

1. Создать пустой вектор best\_path для хранения лучшего пути и переменную best\_distance для хранения минимального расстояния. Инициализировать best\_distance максимальным значением.
2. Вызвать рекурсивную функцию brute\_force, передав в нее вектор точек, начальную точку, текущий путь (сначала только с начальной точкой), текущее расстояние (сначала 0) и ссылки на best\_path и best\_distance.
   1. Если вектор точек пуст, вычислить общее расстояние, добавив расстояние от последней точки в текущем пути до начальной точки. Если общее расстояние меньше best\_distance, обновить best\_path и best\_distance.
   2. Если вектор точек не пуст, перебрать все точки. Для каждой точки создать новый путь, добавив эту точку к текущему пути, и вычислить новое расстояние. Если новое расстояние меньше best\_distance, продолжить поиск с новым путем и новым расстоянием.
3. Вернуть best\_path как результат функции.

**Алгоритм дискретизации**

Входные данные:

* photo\_width: ширина фотографии;
* photo\_height: высота фотографии;
* direction\_degrees: направление в градусах;
* check\_inside: проверка, находятся ли точки внутри полигона.

Выходные данные: трехмерный вектор кортежей, представляющих несколько дискретизированных областей.

Шаги алгоритма.

1. Преобразовать направление из градусов в радианы.
2. Инициализировать вектор для хранения результатов для каждого полигона.
3. Для каждого полигона выполнить.
   1. Инициализировать минимальные и максимальные значения и до крайних противоположностей.
   2. Применить преобразование к каждой точке.
   3. Пройти через координаты полигона, чтобы найти минимальные и максимальные значения и .
   4. Проверить, находится ли точка внутри полигона.
   5. Инициализировать вектор результата.
   6. Рассчитать половину ширины и высоты камеры.
   7. Рассчитать количество фотографий по ширине и высоте.
   8. Для каждой фотографии по ширине и высоте выполнить.
      1. Если check\_inside установлен на true, то рассчитать углы прямоугольника, проверить, находится ли какой-либо угол прямоугольника внутри полигона и, если какой-либо угол находится внутри, рассчитать центр прямоугольника и добавить его в результат.
      2. В противном случае, рассчитать центр прямоугольника и добавить его в результат.
   9. Добавить результат в результаты.
4. Вернуть результаты.

**Алгоритм прямоугольных областей**

Алгоритм прямоугольных областей состоит из трех основных этапов. В него входит алгоритм построения маршрута на прямоугольной области, алгоритм нахождение минимального остовного дерева и алгоритм связывания прямоугольных областей [13].

Алгоритм построения маршрута на прямоугольной области предназначен для построения эффективного маршрута внутри прямоугольника, представляющего поле. Он описывает нахождение маршрута с использованием прямоугольных областей и состоит из несколько этапов.

Нахождение минимального остовного дерева, используемый в алгоритме связывания областей, основан на использовании алгоритм Борувки.

Алгоритм связывания прямоугольных областей предназначен для объединения маршрутов, полученных на этапе расчета маршрутов в прямоугольных областях.

**Алгоритм построения маршрута на прямоугольной области**

Входные данные: region\_points: двумерный вектор, представляющий прямоугольную, дискретизированную область.

Выходные данные: одномерный вектор, представляющий последовательность точек, описывающих оптимальный замкнутый маршрут на прямоугольной области.

Шаги алгоритма.

1. Проверить, является ли входной вектор прямоугольным (имеет ли он одинаковую длину всех своих строк). Если это не так, вернуть сообщение об ошибке.
2. Если ширина вектора меньше 2, то все точки вектора собираются в один регион, который добавляется в общий результат.
3. Для векторов с шириной больше или равной 2 происходит следующее.
   1. Собрать точки с нижней строки вектора.
   2. Собрать точки с правого столбца вектора.
   3. Собрать точки, образующие рампы внутри региона, это делается путем прохода по внутренним столбцам с определенным шагом.
   4. Если ширина вектора нечетная, обработать сегменты coils, собирая точки по сегментам.
   5. В случае, если высота четная, добавить точки region\_points[1][1] и region\_points[0][1], чтобы завершить цепочку.
4. Сформировать результирующий регион, содержащий точки, собранные в предыдущих шагах.
5. Полученные регионы добавляются в общий результат.

Результат работы алгоритма в базовом случае представлена на рисунке 5, частный случай маршрута на прямоугольной области с четной высотой представлен на рисунке 6.

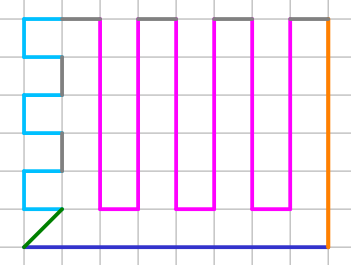


Рисунок 5 – Базовый случай маршрута на прямоугольной области

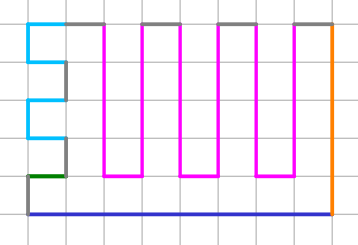


Рисунок 6 – Частный с случай маршрута на прямоугольной области   
с четной высотой

**Алгоритм** **Борувки для построения минимального остовного   
дерева**

Входные данные: weights: взвешенный связный граф G с вершинами V и ребрами E.

Выходные данные: одномерный вектор, представляющий последовательность точек, описывающих оптимальный замкнутый маршрут на прямоугольной области.

Шаги алгоритма.

1. Инициализировать T как пустое множество ребер.
2. Пока T не образует дерево (что эквивалентно условию: пока число ребер в T меньше, чем V − 1).
   1. Для каждой компоненты связности (то есть, дерева в остовном лесе) в подграфе с ребрами T, найти самое дешевое ребро, связывающее эту компоненту с некоторой другой компонентой связности.
   2. Добавить все найденные ребра в множество T.
3. Вернуть T как минимальное остовное дерево входного графа.

**Алгоритм связывания прямоугольных областей**

Входные данные:

* points: трехмерный вектор кортежей представляющий координаты нескольких дискретизированных областей;
* start\_point: координата стартовой точки;

Выходные данные: вектор кортежей, представляющих последовательность координат для посещения, проходящих через несколько прямоугольных областей.

Шаги алгоритма.

1. Инициализировать weights, представляющей матрицу кортежей, которая указывает расстояния и направления между прямоугольными областями.
2. Между каждой областью A и областью B рассчитать расстояние и направление следующим образом.
   1. Если правая сторона B левее левой стороны A.
      1. Если верхняя сторона A ниже нижней стороны B, направление вверх-влево.
      2. Иначе, если верхняя сторона B ниже нижней стороны A, направление вниз-влево.
      3. В противном случае, направление влево.
   2. Если правая сторона A левее левой стороны B.
      1. Если верхняя сторона A ниже нижней стороны B, направление вверх-вправо.
      2. Иначе, если верхняя сторона B ниже нижней стороны A, направление вниз-враво.
      3. В противном случае, направление вправо.
   3. В остальных случаях:
      1. Если верхняя сторона A ниже нижней стороны B, направление вверх.
      2. В противном случае, направление вниз.
   4. В соответствии с направлением, высчитать расстояние между прямоугольными областями.
      1. В случае, если направление соответствует значению вверх, вниз, влево, вправо, взять расстояние между ближайшими ребрами прямоугольников A и B.
      2. В случае, если направление соответствует значению вверх-влево, вверх-влево, вниз-влево, вниз-вправо, взять расстояние между ближайшими вершинами.
3. Для каждой области рассчитать путь внутри, используя алгоритм построения маршрута на прямоугольной области.
4. Рассчитать минимальное остовное дерево используя алгоритм Борувки.
5. Для минимального остовного дерева выполнить связывание областей следующим образом.
   1. Если направление между областями соответствует вверх, вниз, влево, вправо, то связать ближайшие точки ближайших ребер, вставив последовательность посещения внутри области, начиная с ближайших точек.
   2. В ином случае связать области по диагоналям, вставив последовательность посещения внутри области, начиная с ближайшей вершины.
6. Добавить к результату стартовую точку, вставив ее перед ближайшей точке в последовательности посещения.

Визуализация шага 2 алгоритма связывания прямоугольных областей представлен на рисунке 7. Этот рисунок показывает, что в алгоритме является предпочтительным связывание областей по вертикали и горизонтали. Также стоит уточнить, что алгоритм не предназначен для связывания пересекающихся областей, для таких случаев необходим выбор других способов исследования.

На практике эти алгоритмы можно использовать даже если необходимо изменить направление дискретизации используя преобразование координат при повороте для всех полей разом, это будет учтено при разработке.

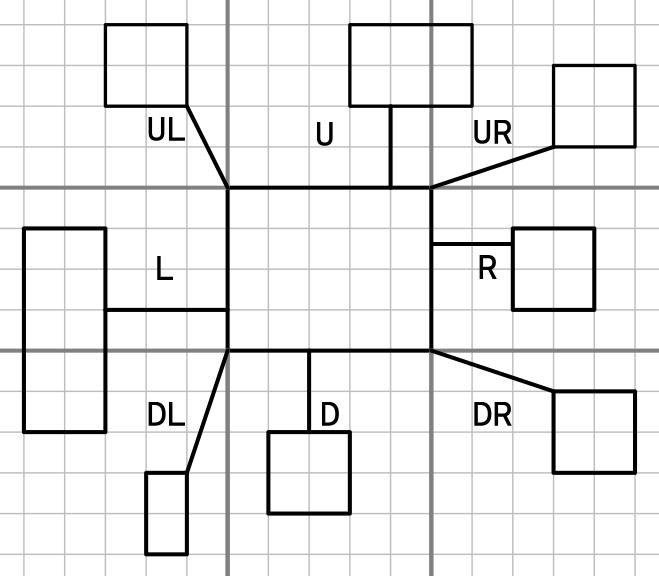


Рисунок 7 – Кратчайшие пути между парой регионов

**Вывод по** **четвертой главе**

В четвертой главе было приведено теоретическое описание алгоритмов, описана задача, которую необходимо решить, указаны подходящие алгоритмы. Описан алгоритм ближайшего соседа, алгоритм полного перебора, алгоритм дискретизации, алгоритм построения маршрута на прямоугольной области, алгоритм Борувки для построения минимального остовного дерева и алгоритм связывания прямоугольных областей.

# 5. РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ СИСТЕМЫ

5.1. Реализация компонентов системы

В реализации компонентов системы имеется сервис Data Management, который предоставляет функциональность для работы с информацией о дронах и камерах. Этот сервис состоит из следующих файлов.

1. Основной файл «uav/mod.rs» определяет структуру uav и методы для создания экземпляров дронов с заданными параметрами. Пример структуры дрона представлен в листинге 1.

Листинг 1 – Структура дрона

#[derive(Debug, Deserialize, Serialize)]

pub struct Uav {

id: u64, // uav id

pub name: String, // uav name

pub max\_payload\_mass: u64, // maximum payload in grams

pub flight\_duration: u64, // average flight duration in seconds

pub takeoff\_speed: f64, // average takeoff speed in meters per second

pub flight\_speed: f64, // average flight speed in meters per second

pub min\_altitude: f64, // minimum safe flight altitude in meters

pub max\_altitude: f64, // maximum safe flight altitude in meters

}

1. Файл «uav/uav\_handle.rs» содержит функции для работы с дронами, такие как создание, обновление и удаление дронов, а также получение списка всех дронов. Пример функции для создания нового дрона представлен в листинге 2.

Листинг 2 – Функции для создания нового дрона

#[tauri::command]

pub fn new\_uav(uav: Uav) -> String {

    let conn = Connection::open("mydatabase.db").expect("Cant open base");

    println!("Received new UAV: {:?}", uav);

    match uav\_sql::insert(&uav, &conn) {

        Ok(\_) => "Ok».to\_string(),

        Err(e) => e.to\_string(),

    }

}

1. Файл «uav/uav\_sql.rs» содержит функции для работы с базой данных, включая создание таблицы для хранения информации о дронах, а также вставка, обновление, удаление и получение списка всех дронов. Пример функции для создания таблицы в базе данных представлен в листинге 3.

Листинг 3 – Функции для создания таблицы uav

pub fn create\_table(conn: &Connection) -> Result<usize> {

let db\_create = conn.execute(

"CREATE TABLE IF NOT EXISTS uav (

uav\_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

uav\_name TEXT NOT NULL,

uav\_max\_payload\_mass INTEGER NOT NULL CHECK (uav\_max\_payload\_mass >= 0),

uav\_flight\_duration INTEGER NOT NULL CHECK (uav\_flight\_duration >= 0),

uav\_takeoff\_speed REAL NOT NULL CHECK (uav\_takeoff\_speed >= 0),

uav\_flight\_speed REAL NOT NULL CHECK (uav\_flight\_speed >= 0),

uav\_min\_altitude REAL DEFAULT 0 NOT NULL CHECK (uav\_min\_altitude >= 0),

uav\_max\_altitude REAL DEFAULT 0 NOT NULL CHECK (uav\_max\_altitude >= 0),

camera\_id INTEGER,

FOREIGN KEY (camera\_id)

REFERENCES camera (camera\_id)

ON DELETE SET NULL

)",

(),

);

let index\_create = conn.execute(

"CREATE UNIQUE INDEX IF NOT EXISTS camera\_id\_index

ON uav(camera\_id)",

(),

);

match (db\_create, index\_create) {

(Ok(val1), Ok(val2)) => Ok(val1 + val2),

(Err(err), \_) => Err(err),

(\_, Err(err)) => Err(err),

}

}

1. Основной файл «camera/mod.rs» содержит структуру и методы для работы с камерой. Структура саmera содержит различные свойства камеры, включая ее идентификатор, имя, массу, угол обзора по оси и разрешение. Здесь также определены метод для создания новых камер. Пример структуры камеры представлен в листинге 4.

Листинг 4 – Структура камеры

#[derive(Debug, Deserialize, Serialize)]

pub struct Camera {

    id: u64,               // id

    pub name: String,      // name

    pub mass: u64,         // mass in grams

    pub fov\_x: f64,        // x-axis viewing angle in degrees

    pub resolution\_x: u16, // camera resolution x

    pub resolution\_y: u16, // camera resolution y

}

1. Файл «camera/camera\_handle.rs» предоставляет функции для обработки запросов к базе данных. Здесь определены функции для создания, обновления, удаления камеры и получения всех камер из базы данных. Каждая из этих функций открывает соединение с базой данных и вызывает соответствующую функцию из файла «camera/camera\_sql.rs». Пример функции для обновления данных о камере представлен в листинге 5.

Листинг 5 – Функция для обновления данных о камере

#[tauri::command]

pub fn update\_camera(camera: Camera) -> String {

let conn = Connection::open("mydatabase.db").expect("Cant open base");

println!("Received updated camera: {:?}", camera);

match camera\_sql::update(&camera, &conn) {

Ok(\_) =>"Ok».to\_string(),

Err(e) => e.to\_string(),

}

}

1. Файл «camera/camera\_sql.rs» содержит функции для взаимодействия с базой данных. Здесь определены функции для создания таблицы в базе данных, вставки новой камеры в базу данных, обновления существующей камеры, удаления камеры из базы данных и получения всех камер из базы данных. Все эти функции работают с базой данных, выполняя соответствующие SQL-запросы. Пример функции обновления данных о камере представлен в листинге 6.

Листинг 6 – Функция для обновления данных о камере в базе

pub fn update(camera: &Camera, conn: &Connection) -> Result<usize> {

    conn.execute(

        "

                UPDATE camera SET

                    camera\_name = ?1,

                    camera\_mass = ?2,

                    camera\_fov\_x = ?3,

                    camera\_resolution\_x = ?4,

                    camera\_resolution\_y = ?5

                WHERE camera\_id = ?6",

        (

            &camera.name,

            &camera.mass,

            &camera.fov\_x,

            &camera.resolution\_x,

            &camera.resolution\_y,

            &camera.id,

        ),

    )

}

Сервис Algorithms, описанный файлом «algorithms.rs», представляет собой ключевую составляющую системы и выполняет алгоритмические функции. Он включает в себя основные функции для вызова из пользовательского интерфейса.

1. Функция discretize\_area реализует алгоритм дискретизации области, описанный ранее, эта функция вызывается при нажатии кнопки «Discretize» пользовательского интерфейса, ее реализация представлена в листинге 1 приложения Б.
2. Функция nearest\_neighbor реализует алгоритм ближайшего соседа для определения пути через все точки. Выполнить эту функции можно, если в блоке пользовательского интерфейса «Algorithm selection» выбрать пункт «Nearest Neighbor» и нажать на кнопку «Calculate». Реализация этой функции представлена в листинге 2 в приложения Б.
3. Вспомогательная функция euclidean\_distance принимает две точки и возвращает расстояние между ними, используется евклидова метрика потому как в контексте данной работы задача построения маршрута рассматривается как задача на плоскости, поэтому расстояние вычисляется по теореме Пифагора. Реализация представлена в листинге 7.

Листинг 7 – Функция расчета расстояния между двумя точками

pub fn euclidean\_distance(a: &(f64, f64), b: &(f64, f64)) -> f64 {

let (x1, y1) = \*a;

let (x2, y2) = \*b;

((x2 - x1).powi(2) + (y2 - y1).powi(2)).sqrt()

}

1. Функция brute\_force использует алгоритм полного перебора для поиска оптимального пути через набор точек. Выполнить эту функции можно, если в блоке пользовательского интерфейса «Algorithm selection» выбрать пункт «Brute Force» и нажать на кнопку «Calculate». Использование данной функции очень затратно по времени, алгоритм представлен лишь в демонстративных целях. Функция начинает с создания отдельного потока для каждой точки, каждый из которых запускает функцию brute\_force\_helper. Эта вспомогательная функция рекурсивно исследует все возможные пути через оставшиеся точки, для обнаружения более короткого пути. Реализация представлена в листинге 3 в приложения Б.
2. Функция rectangular\_areas использует алгоритм, предназначенный для нахождения маршрута на нескольких прямоугольных полях. Выполнить эту функции можно, если в блоке пользовательского интерфейса «Algorithm selection» выбрать пункт «Rectangular Areas» и нажать на кнопку «Calculate». Реализация представлена в листинге 4 в приложения Б. Эта функция также использует другие служебные функции для совей работы. Функция rectangles\_shortest\_path используется для нахождения направления и расстояния между прямоугольниками, ее реализация представлена в листинге 5 приложения Б, она использует find\_direction для нахождения направления, реализация представлена в листинге 6 приложения Б. Функция coordinate\_transformation используется для замены системы координат на плоскости, в частности, ее поворот, реализация представлена в листинге 7 приложения Б. Функция boruvka\_mst используется для нахождения минимального остовного дерева, ее реализация представлена в листинге 8 приложения Б.

Функция calculate\_distance вычисляет общую длину пути, проходящего через заданный набор точек. Используется, для отображения длины пути на интерфейсе. Реализация представлена в листинге 8. Исходные коды остальных функций расположены в репозитории GitHub. [14]

Листинг 8 – Функция расчета общей длины пути

#[tauri::command]

pub fn calculate\_distance(points: Vec<(f64, f64)>) -> f64 {

points

.iter()

.zip(points.iter().cycle().skip(1))

.map(|(a, b)| euclidean\_distance(a, b))

.sum()

}

5.2. Реализация интерфейса системы

Пользовательский интерфейс системы разделен на три основных блока: центральное меню, левое меню и правое меню.

Центральное меню является основной частью интерфейса и играет важную роль, так как оно содержит функции для работы с картой, показывает текущий маршрут и обеспечивает доступ к левому и правому меню, обеспечивает основную навигации по системе. Оно представлено на рисунке 8.



Рисунок 8 – Центральное меню

Центральное меню состоит из нескольких элементов.

1. Карта и маршрут: в центре экрана отображается карта (загружаемая с OpenStreetMap [15]), на которой отображается маршрут дрона. Маршрут представлен в виде линии, соединяющей точки съемки, по которым дрон должен пролететь. Это визуальное представление планируемого маршрута.
2. Кнопки переключения меню: в верхней части интерфейса расположены кнопки, которые позволяют переключать отображение левого и правого меню. Пользователь может скрыть или показать эти меню по своему усмотрению для освобождения места на экране.

Левое меню включает два блока: блок UAV (Беспилотный Летательный Аппарат) и блок Camera (Камера). Левое меню представлено на рисунке 9.

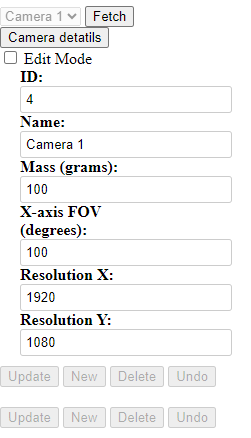
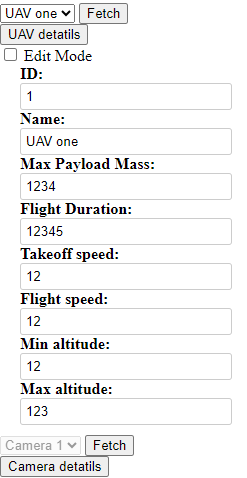


Рисунок 9 – Левое меню

Блок UAV включает в себя инструменты для выбора и редактирования UAV и состоит из нескольких блоков.

1. Выбор и отображение списка UAV: в верхней части блока расположен выпадающий список (select), который позволяет пользователю выбрать один из доступных UAV. Каждый элемент списка содержит имя UAV. При выборе UAV из списка его параметры отображаются в полях ниже.
2. Кнопка «Fetch»: рядом с выпадающим списком находится кнопка «Fetch», которая позволяет пользователю обновить список UAV, загрузив его с базы.
3. Кнопка «UAV details»: эта кнопка позволяет пользователю переключать отображение блока с параметрами UAV. При нажатии на кнопку блок с параметрами UAV может быть скрыт или показан на экране.
4. Внутри блока UAV параметры UAV отображаются в виде полей для ввода которые позволяют пользователю просматривать и редактировать параметры UAV.
   1. ID: идентификатор UAV (только для чтения).
   2. Name: имя UAV.
   3. Max Payload Mass: максимальная масса груза, которую может нести UAV.
   4. Flight Duration: длительность полета UAV.
   5. Takeoff speed: скорость взлета UAV.
   6. Flight speed: скорость полета UAV.
   7. Min altitude: минимальная высота полета UAV.
   8. Max altitude: максимальная высота полета UAV.
5. Под блоком с параметрами UAV располагается панель инструментов, которая содержит следующие кнопки.
   1. «Update»: кнопка, которая позволяет пользователю обновить параметры выбранного UAV.
   2. «New»: кнопка, которая позволяет пользователю создать новый UAV с указанными в полях параметрами.
   3. «Delete»: кнопка, которая позволяет пользователю удалить выбранный UAV.
   4. «Undo»: кнопка, которая позволяет пользователю отменить последнее изменение параметров UAV.

Блок Camera аналогичен по функционалу блоку UAV и включает в себя элементы, представленные ниже.

1. Выпадающий список (select), который позволяет пользователю выбрать одну из доступных камер. Каждый элемент списка содержит имя камеры. При выборе камеры из списка ее параметры отображаются в полях ниже.
2. Кнопка «Fetch»: эта кнопка позволяет пользователю обновить список камер, загрузив его с базы.
3. Кнопка «Camera details»: эта кнопка позволяет пользователю переключать отображение блока с параметрами камеры. При нажатии на кнопку блок с параметрами камеры может быть скрыт или показан на экране.
4. Внутри блока Camera параметры камеры отображаются в виде полей для ввода (input) и меток (label), которые позволяют пользователю просматривать и редактировать параметры камеры.
   1. ID: идентификатор камеры (только для чтения).
   2. Name: имя камеры.
   3. Mass (grams): масса камеры в граммах.
   4. X-axis FOV (degrees): угол обзора камеры по оси в градусах.
   5. Resolution X: разрешение камеры по оси .
   6. Resolution Y: разрешение камеры по оси .

Правое меню состоит из трех блоков: «Altitude Menu» (меню высоты), «Algorithm Menu» (меню алгоритма) и «Mission Parameters» (параметры миссии). Правое меню представлено на рисунке 10.

Рисунок 10 – Правое меню

Далее рассмотрен каждый из блоков подробнее.

1. Блок «Altitude selection», здесь можно выбрать режим ввода высоты. Есть два варианта: «Manual altitude input» (ручной ввод высоты) и «Calculate using sm/px» (расчет высоты на основе sm/px).
   1. Если выбран режим ручного ввода, можно указать процент перекрытия и ввести высоту вручную.
   2. Если выбран режим расчета на основе sm/px, нужно указать процент перекрытия и ввести значение sm/px. После ввода всех параметров можно нажать кнопку «Calculate Altitude» (рассчитать высоту) для получения результата.
2. Блок «Algorithm selection», здесь можно выбрать алгоритм для расчета маршрута. Есть три варианта: «Nearest Neighbor» (ближайший сосед), «Brute Force» (полный перебор), «Rectangular Areas» (прямоугольные области).
3. Блок «Area selection», предназначен для управления выбором поля. Он состоит из нескольких элементов.
   1. Кнопка «Start Drawing», предназначена для перехода в режим выделения поля. Позволяет построить полигоны, ограничивающие поля, которые необходимо заснять.
   2. Кнопка «Undo Polygon», предназначена для удаления последнего установленного полигона.
   3. Кнопка «Undo Point», предназначена для удаления последней установленной точки полигона.
   4. Кнопка «Check», предназначена для проверки установки полигона и выхода из режима выделения поля.
   5. Установка стартовой точки (Set Starting Point): кнопка, позволяющая пользователю установить стартовую точку для маршрута дрона. После выбора этой опции пользователь может щелкнуть на карте, чтобы установить стартовую точку.
4. Блок «Area discretization», здесь устанавливаются параметры дискретизации области съемки. Он состоит из нескольких элементов.
   1. Поле ввода «Discretization Direction», здесь устанавливается направление дискретизации области съемки в градусах.
   2. Кнопка «Discretize», предназначена для выполнения алгоритма дискретизации области.
5. Кнопка «Calculate», выполняет валидацию введенных данных и отправляет их на обработку. Результаты расчета маршрута отображаются на карте.
6. Блок «Mission Parameters», здесь отображаются параметры миссии, такие как длина маршрута, продолжительность миссии и количество фотографий. Также есть кнопка «Export to GeoJSON» (экспорт в формате GeoJSON), которая позволяет экспортировать координаты миссии в этот формат для дальнейшего использования.

Дополнительные иллюстрации, демонстрирующие пользовательский интерфейс, представлены в приложении В.

5.3. Тестирование системы

В ходе разработки и последующего программного обеспечения были разработаны и проведены комплексные тесты для проверки функциональности и стабильности работы системы. Проведенные тесты относятся к виду функционального тестирования, а также тестирования пользовательского интерфейса.

Тестирование проводилось в соответствии с предложенной методологией, при которой каждый из тестовых сценариев включал в себя конкретные шаги для воспроизведения действий пользователя, а также определенный ожидаемый результат. В процессе тестирования внимание было уделено как отдельным функциям, так и взаимодействию между различными компонентами системы.

Процесс тестирования осуществлялся с использованием различных входных данных, включая граничные значения и невалидные данные, чтобы оценить устойчивость и надежность системы в различных условиях, а также чтобы выявить и устранить возможные недостатки. Протоколы тестирования приведены в таблице 3.

Таблица 3 – Протоколы тестирования системы

| **№** | **Название теста** | **Шаги** | **Ожидаемый  результат** | **Тест пройден?** |
| --- | --- | --- | --- | --- |
| 1 | Загрузка дрона | 1. В левом меню выбирать идентификатор дрона.  2. Нажать кнопку «Fetch». | В полях деталей дрона отображается информация о выбранном дроне. | Да |
| 2 | Редактирование дрона | 1. Выбирать дрон.  2. Включить режим редактирования.  3. Изменить параметры дрона.  4. Нажать кнопку «Update». | Информация о дроне в базе данных обновляется. | Да |
| 3 | Расчет маршрута | 1. Ввети необходимые параметры, высоту полета, алгоритм, обозначить точку старта, обозначить зону съемки, выбрать дрон, выбрать камеру.  2. Нажать кнопку. «Calculate» | На карте отображается рассчитанный маршрут. | Да |
| 4 | Некорректная попытка расчета маршрута | 1. Не ввести необходимые параметры, высоту полета, алгоритм, точку старта, зону съемки, дрон, камеру.  2. Нажать кнопку «Calculate». | Отображается ошибка о том, что не все параметры выбраны | Да |
| 5 | Проверка параметров миссии | Просмотрите отображаемые параметры миссии в правом меню. | Параметры миссии (длина маршрута, продолжительность миссии, количество фотографий) отображаются корректно. | Да |
| 6 | Отображение маршрута | После расчета маршрута проверьте визуализацию маршрута на карте. | Маршрут корректно отображается на карте. | Да |
| 7 | Экспорт в GeoJSON | 1. После расчета маршрута нажать кнопку «Export to GeoJSON». 2. Выбрать путь сохранения | Файл GeoJSON с информацией о маршруте сохраняется. | Да |

Окончание таблицы 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Название теста** | **Шаги** | **Ожидаемый  результат** | **Тест пройден?** |
| 8 | Загрузка камеры | 1. В левом меню выберать идентификатор камеры.  2. Нажать кнопку «Fetch». | В полях деталей камеры отображается информация о выбранной камере. | Да |
| 9 | Редактирование камеры | 1. Выбрать камеру.  2. Включить режим редактирования.  3. Изменить параметры камеры.  4. Нажать кнопку «Update». | Информация о камере в базе данных обновляется. | Да |
| 10 | Выбор алгоритма | В правом меню выберите желаемый алгоритм. | Выбранный алгоритм устанавливается как текущий. | Да |
| 11 | Ручной ввод высоты | 1. В правом меню выберите режим ручного ввода высоты.  2. Введите желаемую высоту. | Введенная высота устанавливается как текущая. | Да |
| 12 | Расчет высоты | 1. В правом меню выберите режим расчета высоты.  2. Введите значение sm/px.  3. Нажмите кнопку «Calculate Altitude». | Высота автоматически рассчитывается и устанавливается как текущая. | Да |
| 13 | Изменение параметра перекрытия | В правом меню ввести значение в поле «Overlap (%)». | Значение перекрытия изменяется в соответствии с введенным значением. | Да |
| 14 | Проверка работы с отсутствующими данными | Нажать кнопку «Calculate», когда база данных пустая. | Система сообщает о том, что дрон не выбран. | Да |

Результат выполнения теста 5 «Проверка параметров миссии» представлен на рисунке 10. На рисунке отображены параметры миссии, длина маршрута, продолжительность миссии, количество фотографий.

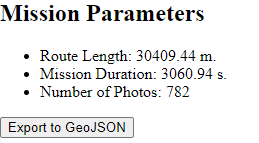


Рисунок 10 – Результат выполнения теста 5   
«Проверка параметров миссии»

Результат выполнения теста 6 «Отображение маршрута» представлен на рисунке 11. Маршрут представлен в виде синей линии, соединяющей зеленые точки, представляющие точки съемки дрона.



Рисунок 11 – Результат выполнения теста 6 «Отображение маршрута»

**Вывод по пятой главе**

В пятой главе была приведена реализация компонентов системы, включающая реализацию функций расчета и реализацию пользовательского интерфейса системы. Также проведено тестирование системы.