Оглавление

[Введение 6](#_Toc440241985)

[1 Обзор существующих ГИС 8](#_Toc440241986)

[1.1 Технологическая схема построения ГИС 8](#_Toc440241987)

[1.2 Коммерческие ГИС 11](#_Toc440241988)

[1.3 Построение ГИС на основе свободно распространяемого ПО 12](#_Toc440241989)

[2 Способы представления геоданных 16](#_Toc440241990)

[3 Виды и структуры геоданных 16](#_Toc440241991)

[4 Система технического зрения 18](#_Toc440241992)

[4.1 Описание аппаратной части системы технического зрения 18](#_Toc440241993)

[4.2 Состыковка ГИС и системы технического зрения 19](#_Toc440241994)

[5 Средство разработки базы данных 21](#_Toc440241995)

[6 Алгоритм работы программы для управляющего компьютера 22](#_Toc440241996)

[6.1 Реализация сервера 22](#_Toc440241997)

[6.2 Модуль Python 25](#_Toc440241998)

[7 Разработка web-интерфейса ГИС 28](#_Toc440241999)

[7.1 Установка компонентов 28](#_Toc440242000)

[7.2 Создание базы данных 28](#_Toc440242001)

[7.3 Импортирование данных 29](#_Toc440242002)

[7.4 Отображение карты 31](#_Toc440242003)

[Заключение 36](#_Toc440242004)

[Список литературы 37](#_Toc440242005)

# Введение

В настоящее время все больший интерес исследователей и разработчиков робототехнических комплексов привлекают автономные наземные транспортные средства (наземные беспилотники), с помощью которых возможно решение задач автоматизации множества видов человеческой деятельности (например, транспортировки грузов, выполнения разведывательных операций и тому подобных).

Основные успехи данной области достигнуты в разработке мобильных роботов индустриального и бытового применения. Работа таких роботов, как правило, происходит внутри помещений, характеризуемых однородной плоской горизонтальной поверхностью и наличием непреодолимых препятствий. Однако важный класс наземных беспилотников, предназначенных для работы на открытых пространствах, в том числе пересеченной местности, остается в настоящее время недостаточно развитым [1].

Одной из задач, решение которой необходимо для создания производительных автономных наземных транспортных средств, способных совершать исследование открытой пересеченной местности, является разработка геоинформационной системы [2].

Актуальность данной работы заключается в том, что существующие геоинформационные системы, применяемые в робототехнике, являются частными разработками, к которым нет открытого доступа. Следовательно, для реализации функции предоставления информации об окружающем пространстве и текущем местоположении многофункциональной роботизированной платформы необходима разработать собственную геоинформационную систему.

Целью данной работы является предоставить проект геоинформационной системы для многофункциональной роботизированной платформы.

В процессе написания данной работы была описана модель базы данных геоинформационной системы; описан принцы взаимодействия с системой технического зрения; разработан web-интерфейс геоинформационной системы.

# 1 Обзор существующих ГИС

Передо мной была поставлена задача предоставления данных об окружающем пространстве многофункциональной роботизированной платформы (МРП) для систем МРП и пользователя. Также было необходимо разработать хранилище данных для системы технического зрения. Для решения данных проблем было решено спроектировать геоинформационную систему с web-интерфейсом. Перед тем как начать проектирование и разработку были рассмотрены существующие ГИС и выделены их достоинства и недостатки.

## 1.1 Технологическая схема построения ГИС

Наиболее прогрессивная технологическая схема построения ГИС на сегодняшний день, основана на использовании в качестве хранилища пространственных данных специализированных расширений для наиболее распространённых SQL серверов, которые на сегодня имеются у всех основных поставщиков подобных решений, в том числе Oracle Locator/Spatial для Oracle SQL Server, Microsoft Spatial для Microsoft SQL Server, PostGIS для PostgreSQL, MySQL Saptial для одноименного сервера (права на последнюю редакцию принадлежат фирме Oracle), SpatialLite для SQLLite и т. п. Данные расширения добавляют необходимый функционал для хранения пространственных данных в соответствующих SQL серверах, облегчающих, ускоряющих, а также стандартизирующих работу с пространственными данными на данном сервере БД.

Ещё одной тенденцией, характерной для решений данного поколения ГИС, является переход к использованию в качестве рабочего места конечного пользователя ГИС приложения на основе WEB-браузера, а также встраивания необходимого набора скриптов для работы с системой в геоинформационные интернет-порталы. В некоторых случаях данные решения являются вспомогательными и выполняют в основном функции просмотра пространственных данных, а в качестве редактора используется обычная программа, но также имеются решения, когда весь функционал по работе с ГИС, включая её администрирование и ввод пространственных данных, реализован в виде WEB-приложения работающего через  WEB-браузер.

**Достоинства:** структура хранения пространственных данных не зависит от разработчика конкретной ГИС, что резко расширяет возможности по работе с пространственными данными и обмену ими, интеграции с другими системами, использованию программного обеспечения сторонних разработчиков, в том числе класса FreeWare (свободно распространяемое) и OpenSource (с открытым исходным кодом). У всех разработчиков хранилищ пространственных данных имеется обширная техническая документация (в основном на английском языке). При использовании данных решений пользователь ГИС в гораздо меньшей степени зависит от конкретного поставщика, может сменить используемую ГИС или расширить имеющийся функционал за счёт использования других ГИС, работающих с тем же хранилищем пространственных данных. При этом затраты по переносу данных существенно меньше, чем для остальных вариантов, либо отсутствуют вообще.

Данные решения позволяют также реализовывать распределённые ГИС, когда с одним общим хранилищем пространственных данных работают несколько различных ГИС разных организаций, в том числе территориально находящихся в разным местах и объединённых каналами передачи данных (либо интернет, либо защищённые каналы передачи данных). Кроме этого для Oracle, Microsoft и PostgreSQL имеются штатные средства создания распределённых БД и поддержания их целостности на уровне базового SQL сервера. Это позволяет создать систему с несколькими независимыми хранилищам пространственных данных, которые периодически производят синхронизацию изменений для поддержания логической целостности единой БД, а также создавать системы повышенной надёжности, устойчивые к сбоям за счёт дублирования и независимости общей работы системы от функционирования одного из узлов.

**Недостатки:** данные решения существенно сложнее в установке, настройке и администрировании, чем все остальные варианты, особенно при использовании решений на основе WEB-технологий, поскольку помимо самой ГИС, SQL сервера с хранилищем пространственных данных добавляются ещё и работы по интернет-серверу и системе безопасности. Решения на базе WEB-технологий обладают заметно меньшим быстродействием, а также весьма ограниченным функционалом по сравнению с традиционными приложениями. Часто они реализуют необходимый минимум функций, без которых использование ГИС будет невозможно, а для решения специализированных или аналитических задач необходимо использовать другие программы (что для данного варианта построения системы не вызывает проблем).

Необходимо отметить, что многие разработчики развивают свои решения добавляя новые функции и возможности, постепенно переходя от одной технологической схемы построения системы к другой. При этом для обеспечения совместимости с предыдущими версиями своих систем они оставляют поддержку старых технологий работы и форматов данных, поскольку у пользователей уже накоплены большие объёмы данных и имеется множество специалистов, привыкших к старым методикам работы. В результате у наиболее старых и крупных разработчиков ГИС систем, таких как ESRI (ArcGIS) или Intergraph, на сегодня имеются продукты, которые при необходимости позволяют построить ГИС с использованием нескольких технологических схем работы с пространственными данными в разных модулях системы.

Также следует сказать о том, что появление новых технологических схем работы с пространственными данными не означает, что они будут эффективны и целесообразны для всех случаев использования. Имеется множество прикладных задач с использованием ГИС, где иная технологическая схема работы с файлами собственного формата будет наиболее предпочтительной и эффективной. Выбор той или иной схемы во многом определяется решаемой задачей, но когда речь заходит о создании ГИС, которые работают с крупными банками разнородных пространственных данных, особенно распределённых, основным трендом является переход к использованию четвёртой технологической схемы, как наиболее удобной для интеграции различных систем и модулей именно на уровне данных.

## 1.2 Коммерческие ГИС

В данном разделе я опишу самую популярную коммерческую платформу построения ГИС – ArcGIS, характеристики остальных ГИС приведены в таблице 1.

С помощью программ ArcGIS можно построить ГИС первой, третьей и четвёртой технологических схем работы с пространственными данными. Имеется возможность работы как с собственным форматом фалов данных, так и с хранилищами пространственных данных под управлением SQL серверов Oracle, Microsoft, IBM DB2, PostgreSQL, Informix и др. При этом возможно как создание базы пространственных данных во внутреннем формате с использованием средства ArcSDE (третье поколение), так и работа с собственными расширениями для хранения пространственных данных (четвёртое поколение).

Одна из немногих ГИС платформ, которая поддерживает полноценную работу с топологической моделью представления данных (узловая и цепочно-узловая модели), а также хранение, обработку и визуализацию трехмерного представления пространственных данных. Позволят создавать системы для работы с большими и очень большими объёмами данных (при использовании соответствующего аппаратного и системного обеспечения), рассчитанными на большое количество обращений пользователей.

Высокая сложность установки и обслуживания системы. Требуется наличие хорошо подготовленного персонала, желательно прошедшего специальное обучение и сертификацию фирмы ESRI по работе и обслуживанию ArcGIS.

## 1.3 Построение ГИС на основе свободно распространяемого ПО

В настоящее время состояние продуктов, разрабатываемых сообществом OSGeo, достиг такого состояния, что их можно уже рассматривать как самостоятельную платформу для создания ГИС систем. При этом мы имеем возможность создавать как простые системы первого поколения, работающие с файлами данных на одном компьютере, так и распределённые сетевые системы четвёртого поколения. Типовой набор программ для формирования подобной ГИС:

PostGIS – хранилище пространственных данных на базе свободно распространяемого сервера PostgreSQL;

Quantum GIS (QGIS) – инструментальная ГИС для профессиональной обработки пространственных данных, с помощью библиотеки GDAL поддерживается более 50 растровых и более 20 векторных форматов, включая ESRI – Shape, MapInfo – mif/mid и tab/dat; Autodesk – DXF и другие. Имеется собственный модуль обработки растровых изображений, позволяющий выполнять геопривязку и имеющий несколько алгоритмов трансформации растров.

MapServer — серверная часть для построения интернет-геопорталов и интернет-приложений, обеспечивающий формирование и выдачу изображения по стандарту WMS.

OpenLayers — набор скриптов на языке JavaScript встраиваемых в интернет-сайты для организации клиентской части работы с пространственными данными по протоколу WMS и WFS.

GRASS GIS – пакет более чем из 400 программ обработки и анализа пространственных данных, который может рассматриваться как мощный инструментально-аналитический модуль открытой платформы. Реализует множество алгоритмов и методик работы с пространственными данными и данным ДЗЗ. Может использоваться как отдельно, так и с помощьюQuantum GIS, к которой подключается в виде модулей расширения.

***Достоинства платформы:***

* Позволяет создавать современные распределённые ГИС четвёртого поколения, полностью поддерживающие стандарты обмена пространственными данными консорциума OGC.
* Нет затрат на приобретение лицензий на программные компоненты платформы.
* Поддерживается большое количество программно-аппаратных платформ. Для серверных модулей и хранилища пространственных данных имеются 64-битные версии.
* Поддержка большинства популярных векторных и растровых форматов пространственных данных.
* Поддержка работы с трехмерным представлением пространственных данных.
* Большое количество различных прикладных и расчётных программ, позволяющих обрабатывать или использовать пространственные данные данной платформы.

***Недостатки платформы:***

* На данный момент платформа больше представляет собой конструктор «собери сам», чем законченное решение, которое можно установить и работать в виде готового решения, как в случае с платными платформами. При отсутствии затрат на покупку лицензий, затраты на внедрение системы могут быть более значительными, чем для платных систем.
* Практически отсутствуют законченные типовые решения для прикладных или специализированных задач, которые также распространялись бы как свободное программное обеспечение. Известные автору прикладные разработки на базе данной платформы предлагаются как коммерческие продукты.
* Имеются проблемы с русским языком. Не все модули системы переведены, в QGIS, даже в редакции NextGIS, имеются проблемы при работе с данными, содержащими русский текст.
* Имеются проблемы с отображением сложных условных знаков, принятых в Российской Федерации для оформления топографических карт.
* Возможности для формирования документов и отчётов для печати весьма ограничены и включают только базовый набор функций. Нет возможности с помощью базовых модулей распечатать каталог координат или оформить карточку выбранного пространственного объекта, которая бы содержала фрагмент карты и информацию из семантической таблицы для данного объекта. Для решения подобных задач требуется разработка дополнительных модулей, либо приобретение готовых платных решений на базе данной платформы.

Данная платформа очень популярна за пределами Российской Федерации, особенно в Европе. В частности, она является одной из базовых для создания национальной инфраструктуры пространственных данных Испании.

В таблице 1 приведены характеристики, существующих ГИС.

Таблица 1 – Характеристики ГИС

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Программа, разработчик | файлы собств. Формат | клиент-сервер | внешняя СУБД | трехмерное представление данных | WEB приложения |
| ***Зарубежные ГИС*** |  |  |  |  |  |
| семейство ArcGIS, ESRI | \* |  | \* | \* | \* |
| семейство GeoMedia, Intergraph | **\*** |  | **\*** | \* | \* |
| MapINFO — Pitney Bowes Software | **\*** |  | **\*** | **\*** | \* |
| Продукты Autodesk | **\*** |  | **\*** | **\*** | \* |
| ***Российские ГИС*** |  |  |  |  |  |
| ГИС Карта 2011 — ЗАО КБ Панорама | **\*** | \* |  | **\*** | \* |
| ГИС ИнГео - ЗАО «ЦСИ Интегро» |  |  | \* |  | \* |
| Продолжение таблицы 1 |  |  |  |  |  |
| ГИС платформа CSoft |  |  | \* | \* | \* |
| ГИС платформа ОАО «Рекод» | \* | \* | \* | \* | \* |
| Zulu 7.0 — ООО «Политерм» | \* | \* |  | \* |  |
| ГЕОКАД плюс — Geocad Systems | \* | \* |  | \* | \* |
| платформа ИндорСОФТ | \* |  | \* | \* |  |
| семейство CREDO | \* |  |  | \* |  |
| ГИС ЦентрПрограммСистем |  | \* |  |  | \* |
| «Геомиксер» - НТЦ «СКАНЭКС» |  |  |  |  | \* |
| ***ГИС с открытым кодом*** |  |  |  |  |  |
| PostGIS + QGIS + MapServer + OpenLayers | \* |  | \* | \* | \* |

Проанализировав все достоинства и недостатки существующих геоинформационных систем, было решено спроектировать собственную систему, проект которой представлен в данной работе.

# 2 Способы представления геоданных

Геоинформационная система (ГИС) работает с географическими данными (они же «геоданные», «пространственные данные» или «геопространственные данные») [3,4]. Сами ГИС появились как гибрид баз данных (БД) и инженерной графики (САПР), поэтому объекты в ГИС также объединяют в себе две сущности: геометрию и атрибутивную информацию (атрибутику). Так, объект «дорога» в понимании ГИС это:

- геометрический примитив: совокупность поворотных точек каждая из которых имеет координаты и соединенных друг с другом в определенном порядке

- атрибутивное описание: совокупность пар ключ-значение.

Эта особенность ГИС поначалу обособила ГИС от параллельно развивавшихся каждая в своём «русле» баз данных и САПР. Сейчас границы технологий существенно размылись. У нас есть базы данных способные хранить геоданные (например PostgreSQL/PostGIS) и САПР имеющие представление о том, где в географическом пространстве находятся объекты, с которыми мы в них работаем.

Для того, чтобы создать геоданные необходимо выполнить процесс географической привязки. Источником координат являются: системы геопозицинирования (GPS, ГЛОНАСС) и другие уже привязанные карты. [5]

# 3 Виды и структуры геоданных

По содержанию геоданные подразделяются на: базовые, или общегеографические и тематические, пространственные и атрибутивные, количественные и качественные [6]. Пространственные данные в свою очередь подразделяются на позиционные и геометрические. По назначению геоданные делятся на основные, или объектные (с помощью которых описываются реальные территории или проектируемые территориальные объекты), идентификации, или идентификационные (для связывания данных), кодовые (для обозначения и машинного представления информации) и метаданные (данные, описывающие объектные данные: тему, источник, название объектов и т.д.) По форме они бывают цифровые, буквенно-числовые, или текстовые и графические [7].

В ГИС используются три вида БД: атрибутивные, пространственные и совместные, или интегральные. Атрибутивные БД (АБД) — это классические базы данных, применяемые во всех типах информационных систем. Различают три основные модели организации АБД: иерархические, сетевые и реляционные. В ГИС используются главным образом реляционные АБД, состоящие из таблиц, столбцы которых соответствуют характеристикам предметов и называются полями, а строки соответствуют значениям характеристик и называются записями. Отдельная таблица содержит значения характеристик предметов одного класса или вида. СУБД состоит из программных и языковых средств и обеспечивает создание БД, ее поддержку (поиск, сортировку, добавление и исправление таблиц, создание новых таблиц, привязку их к пространственной БД), а также выполнение запросов пользователей с помощью структурного языка SQL.

Пространственные БД (ПБД) подразделяются на слоевые и объектно-ориентированные. Последние применяются редко, поэтому рассматриваться не будут. В первых БД геоданные сгруппированы в геометрически и тематически однородные плоские структуры (покрытия, или слои), и представляют собой упорядоченные слоевые наборы. Важно отметить, что такая организация данных хорошо согласуется с организацией традиционной карты. Отдельному слою ПБД соответствует отдельный тематический срез, или компонент содержания карты, а ПБД в целом - карта или серия карт. Кроме того, слоевая организация ПБД удобна для хранения, отображения, обработки и анализа геоданных как послойно, так и в различных сочетаниях слоев. Каждый слой логически (и часто физически) представляет собой отдельную единицу хранения и имеет собственную систему идентификаторов.

# 4 Система технического зрения

Система технического зрения (СТЗ) представляет из себя программно-аппаратный комплекс, предназначенный для поиска и детектирования объектов [8]. База данных СТЗ представляет из себя объектно-реляционную СУБД PosrgreSQL, с расширением PostGIS. В базе хранятся объекты их свойства и координаты.

## 4.1 Описание аппаратной части системы технического зрения

В СТЗ используются три видеокамеры. Две камеры закреплены на поворотном устройстве, имеющем 2 степени свободы и обеспечивают стереозрение.

Для получения качественного видеоизображения, используются видеокамеры NetCam4 от компании Vocord. Эти видеокамеры разработаны специально для использования в системах распознавания лиц, распознавания номеров машин и для других задач видеоаналитики, которые требуют изображения высокого качества. Для корректной работы СТЗ камера формирует качественное изображение при разных условиях освещенности: днем и ночью, в солнечную и пасмурную погоду. Для работы при «неблагоприятных» условиях в VOCORD NetCam4 реализованы алгоритмы автоматической адаптации изображения к задачам видеоаналитики. Частота кадров, получаемая с этих камер составляет 30 кадров в секунду. Улучшение качества изображения достигается за счет:

* Автоматической регулировки экспозиции с точно задаваемым минимальным и максимальным временем экспонирования
* Программно-управляемого многопараметрического эквалайзера гистограммы
* Преобразования разрядности изображения от 14 бит к 8 битам с автоматической адаптацией к изменению условий освещенности и сохранением эффективного динамического диапазона
* Автоматической регулировки диафрагмы
* Подстройки уровня черного
* Автоматического баланса белого (для цветных камер)

Также для удобства масштабирования изображения в камерах предусмотрено встроенное управление моторизованными объективами (автоматически и вручную через web-интерфейс).

Для автоматического и ручного слежения за объектом осуществлена установка наклонно-поворотного устройства для фронтальной камеры. В классической реализации наклонно-поворотных устройств используется только две оси координат, это поворот видеокамеры и наклон вперед – назад. В предлагаемом случае вводится дополнительная ось наклона влево – вправо, которою можно использовать для стабилизации изображения, так как СТЗ установлена на мобильной роботизированной платформе. Платформа передвигается в том числе и по пересеченной местности (кочки, ямы и пр.), и для снижения нагрузки на вычислительный модуль и снятия с него функция первичной стабилизации изображения, камера откланяется на нужный угол относительно поверхности.

## 4.2 Состыковка ГИС и системы технического зрения

Для сопряжения геоинформационной системы и системы технического зрения (СТЗ), необходимо рассмотреть алгоритм функционирования СТЗ.

На вход системы СТЗ подается обученная модель с объектами. Система обеспечивать поиск объектов согласно заранее заданной базе. Далее осуществляется идентификации найденного объекта, а также вычисление координат объекта и передача их основной системе управления. В случае если объект удаляется из обзора, поворотное устройство корректирует свое положение, чтобы объект всегда находился по центру объектива камеры. Так как роботизированную платформу планируется использовать и в ночное время суток, в СТЗ предусмотрено получение изображения в инфракрасном диапазоне. Для этого используются ИК прожектора. СТЗ передает системе управления облако точек полученной 3D сцены, а также ID распознанного объекта.

Исходя из алгоритма функционирования системы СТЗ, можно сделать вывод, что для сопряжения двух систем (СТЗ и ГИС), в базе данных ГИС необходимо хранить образное описание объекта в форме jpg файла, параметры объекта, его идентификатор и координаты. СТЗ используется для уточнения местоположения существующих объектов, хранящихся в БД.

Специально для системы СТЗ в базе данных ГИС были созданы новые таблицы stz и coordinates\_stz (См. Рисунок 1).

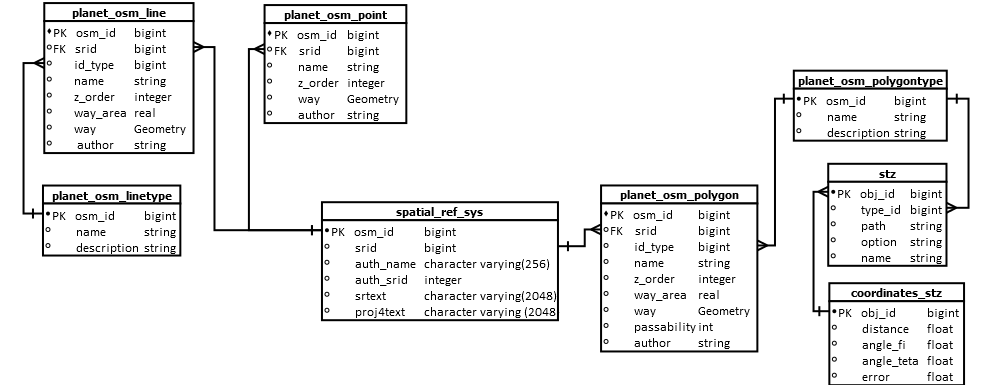


Рисунок 1 – Часть физической модели БД

СТЗ хранит координаты объекта в сферической системе координат с определенной погрешностью, данные координаты хранятся в таблице coordinates\_stz.

Для сопряжения ГИС и СТЗ был создан модуль STZ, написанный на языке Python. Функция GetPath(), в модуле STZ, возвращает образное описание объектов (расположение jpg файлов, каждого из объектов). В функции GetObject() обрабатываются ответы STZ, данная функция возвращает описание объекта и его координаты. Функция SetObjectToDatabase(Object) предназначена для записи нового объекта Object в базу данных ГИС, на вход данной функции подается строка в формате JSON.

# 5 Средство разработки базы данных

В качестве СУБД была выбрана – PostgreSQL [9] с расширением PostGIS [10]. Основным достоинством PostGIS является возможность использования языка SQL совместно с пространственными операторами и функциями. Кроме простого хранения данных, PostGIS позволяет осуществлять любые виды операций над ними, малая часть этих операций[11] описана ниже:

* **ST\_Distance(geometry, geometry) –** возвращает декартово расстояние между двумя геометриями в заданных единицах.
* **ST\_DWithin(geometry, geometry, float) –** возвращает истину, если геометрии находятся в пределах указанного расстояния одна от другой.
* **ST\_Intersects(geometry, geometry) –** возвращает TRUE, если геометрии пространственно пересекаются.
* **ST\_Touches(geometry, geometry) – в**озвращает TRUE, если геометрии "пространственно соприкасаются".
* **ST\_Within(geometry A, geometry B) –** возвращает 1 (TRUE), если геометрия A находится пространственно внутри геометрии B.
* **ST\_Centroid(geometry) – в**озвращает точку - центроид геометрии.
* **ST\_Length(geometry) –** возвращает длину кривой в соответствующей ей системе координат.
* **ST\_Buffer(geometry, double, [integer]) –** возвращает геометрию, все точки которой находятся на меньшем или равном расстоянии, чем заданное, от заданной геометрии. Расчеты производятся в пространственной системе координат заданной геометрии. Опциональный третий параметр задает число сегментов, используемых для аппроксимации четверти окружности (по умолчанию - 8).

# 6 Алгоритм работы программы для управляющего компьютера

## 6.1 Реализация сервера

Для работы web-интерфейса ГИС, необходимо обрабатывать запросы, отправляемые пользователем. Для этого на главном сервере МРП, необходимо добавить обработчик запросов на получение текущего местоположения МРП и построение нового пути. Для реализации сервера будет достаточно использовать «класс-заготовку» - BaseHTTPRequestHandler. Этот класс может использоваться как базовый для реализации собственного обработчика запросов, действующего в составе сервера HTTP. После того как клиент установит соединение и HTTP-заголовки его запроса будут проанализированы, выполняется попытка вызвать метод вида do\_REQUEST, имя которого определяется исходя из типа запроса: для обработки GET-запроса типа будет вызван метод do\_GET(), для обработки POST-запроса – метод do\_POST() и так далее. По умолчанию этот класс ничего не делает, и предполагается, что эти методы должны быть переопределены в подклассах. Для этого были созданы соответствующие методы, включенные в класс GISRequestHandler() изображенный на рисунке 2.



Рисунок 2 – Класс GISRequestHandler()

Ниже приведено описание функций:

* *load\_str(self, data) –* отправка строки *data* клиенту в кодировке 'utf-8'
* *to\_json(self, data)* – отправка клиенту JSON строки *data*
* *do\_GET(self) –* обработчик GET запросов
* *do\_POST(self)* – обработчик POST запросов
* *process\_gis\_URLs(self, path, data = None)* – обработчик URL-адреса *path*

На рисунке 3 изображены алгоритмы отправки клиенту обычной строки и JSON строки. Эти два алгоритма схожи, при успешном соединении клиенту отправляется код ответа – информация о том, что запрашиваемая страница была обнаружена (код 200: HTTP/ok). Затем необходимо отправить эскейп-посдедовательность, которая свидетельствует о завершении заголовка ответа, однако при отправке JSON строки, необходимо дополнительно отправить http-заголовок, указывающий, на тип содержимого ответа ( 'application/json'), отправляемого клиенту.

Также следует обратить внимание на атрибут wfile класса BaseHTTPRequestHandler. Он представляет собой объект, который реализует поток сервера, транслируемый на клиент. Любой вывод в браузер осуществляется методом wfile.write('something')[12].



Рисунок 3 – Алгоритмы отправки клиенту обычной строки и JSON строки

На рисунке 4 показан алгоритм работы функции *process\_gis\_URLs()*, предназначенной для обработки URL-адресов, поступающих от клиента. Принцип работы этого алгоритма достаточно прост, в зависимости от запроса клиента, на сервере выполняется соответствующая операция, и клиенту отправляется определенный ответ. Если клиент отправит некорректный запрос, то ему в ответ придет сообщение об ошибке.



Рисунок 4 – Алгоритм обработчика URL-адресов

И в заключении, необходимо создать объект класса HTTPServer, передав ему в качестве параметров IP-адрес и порт, на котором будет работать http-сервер, а также наш класс-потомок GISRequestHandler, который собственно и будет отвечать за обработку запросов к серверу: *server = HTTPServer(('0.0.0.0', 8001), GISRequestHandler)*.

## 6.2 Модуль Python

Psycopg [13] является самым популярным адаптером базы данных PostgreSQL для языка программирования Python. Его основные особенности полная реализация Python API 2.0 DB спецификации, и безопасность потоков (несколько потоков могут совместно использовать одно соединение). Она была разработана для тяжелых многопоточных приложений, которые создают и разрушают много курсоров и делают большое количество одновременных операций INSERT или UPDATE[14].

Mapnik - свободный инструментарий отрисовки карты [15]. Он написан на C++ и Python. Использует библиотеку AGG и дает возможность сглаживать объекты на карте с большой точностью. Может читать данные в формате компании ESRI, PostGIS, точечные рисунки TIFF, файлы.osm, а также поддерживает любые GDAL или OGR форматы. Пакеты доступны для большинства выпусков Linux, двоичные файлы доступны для Mac OS X и Windows.

PostGIS – наиболее используемый источник для отрисовки данных OSM посредством Mapnik. Данные OSM преобразуются в формат доступный Mapnik с помощью osm2pgsql, который загружает полученные данные в базу PostGIS. Встроенные в базу данных функции GIS и языка запросов SQL дают более широкие возможности отрисовки, недоступные для файлов OSM XML.

Данные о местоположение роботизированной платформы и маршрут находится в базе данных, для отображения их на карте необходимо представить их в формате GeoJSON. Были написаны функции на Python, для выполнения данной операции, рассмотрим функции GetCurrentLoc() и GetWay(), в качестве адаптера базы данных PostgreSQL был использован Psycopg [13].

Принципы работы функций GetCurrentLoc() и GetWay() схожи (См. Рис.5), поэтому рассмотрим одну из них, функция SetCurrentLocation() служит для записи текущего местоположения в БД. Алгоритм работы функции GetCurrentLoc():

1. Выполнение запроса к базе о текущем местоположении роботизированной платформы. Функция ST\_AsGeoJSON() [16], в PostGIS [17], осуществляет выводит данные в формате GeoJSON. Ниже представлен запрос к БД.

*SELECT ST\_AsGeoJSON(ST\_AsText(ST\_Transform(way,4326))) FROM planet\_osm\_point where name='Robot\_location';*

1. В LeafLeat структура строки GeoJson отличается от той, которую возвращает PostGIS, для того что бы устранить эти отличия следует поменять местами широту и долготу в данных от PostGIS.

*for value in cursor:*

*GeoJsonString = value ['st\_asgeojson']*

*json\_parse = json.loads(GeoJsonString)*

*json\_parse['coordinates'].reverse()*

*GeoJsonString = json.dumps(json\_parse)*

1. Запись данных в файл JsonString.js

*f.write(GeoJsonString)*

 Рисунок 5 – Принципы работы функций GetCurrentLoc(), GetWay() и SetCurrentLocation

# 7 Разработка web-интерфейса ГИС

На данный момент ведется множество работ по исследованию и разработке робототехнических платформ, с помощью которых можно автоматизировать ряд задач, таких как автономный сбор данных об окружающем пространстве. Собранные данные необходимо хранить для последующей обработки и уметь их отображать для оценки проделанной работы. Для этой цели необходимо реализовать web-интерфейс геоинформационной системы [1, 2].

## 7.1 Установка компонентов

Для начала необходимо установить компоненты, которые потребуются для работы с базой данных: postgresql-9.3-postgis-2.1 [9, 10] и osm2pgsql. Для их установки необходимо ввести в терминале следующие команды:

1) sudo apt-get install postgresql-9.3-postgis-2.1.

2) sudo apt-get install osm2pgsql.

Программа osm2pgsql [18] предназначена для импортирования данных из формата osm (Open Street Maps) в СУБД Postgresql/PistGIS [17, 19]. OSM — это открытая геопространственная база данных с мировым весьма подробным покрытием. Для пользователя — это карта, которую можно посмотреть on-line, загрузить на навигатора, телефон, отредактировать, распечатать и т.д. С точки зрения разработчика — эта весьма своеобразная база данных. В OSM нет четкой схемы данных. Тип объекта, его свойства, а в некотором смысле и геометрия, задаются через теги — текстовые пары ключ-значение. Геометрия хранится лишь для точек. Линия — это массив идентификаторов точек. Полигон — набор ссылок на линии с указанием, в какую из границ входит линия — внутреннюю или внешнюю.

## 7.2 Создание базы данных

Далее необходимо создать саму базу данных для этого в терминал вводим команды, представленные ниже:

1. *sudo -u postgres psql* – вход в БД от имени администратора;
2. *CREATE DATABASE test\_database;* – создание БД *test\_database*;
3. *CREATE USER test\_user WITH password '123';* – создание пользователя *test\_user* с паролем ‘123’;
4. *GRANT ALL privileges ON DATABASE test\_database TO test\_user;* – присвоение прав на администрирование БД пользователю *test\_user*;
5. *alter role test\_user SUPERUSER*; – присвоение прав “суперпользователя”
6. *psql -h localhost test\_database test\_user*– вход в БД *test\_database* от имени пользователя *test\_user*
7. *create extension postgis*; – добавляем расширение PostGIS

## 7.3 Импортирование данных

Для импортирования данных, потребуется дамп карты в формате osm, дамп можно скачать на официальном сайте OpenStreetMaps. Импортирование данных происходит после ввода команды: *osm2pgsql -d test\_database -U test\_user -H localhost --cache-strategy sparse StudCity.osm*. Разберем структуру данной команды:

* *test\_database* – название базы данных в которую происходит импорт данных
* *test\_user* – имя пользователя, обладающего правами редактирования базы данных
* *localhost* – *IP*-адресс хоста базы данных
* *sparse* – параметр для корректной работы с кэш памятью
* *StudCity.osm* – дамп карты скачанный с OpenStreetMaps

После импортирования данных было решено добавить авто инкремент для первичных ключей таблиц: planet\_osm\_point, planet\_osm\_line и planet\_osm\_polygon. Ниже представлен пример требуемого запроса для таблицы planet\_osm\_polygon:

CREATE SEQUENCE user\_id\_seq;

ALTER TABLE planet\_osm\_polygon ALTER osm\_id SET DEFAULT NEXTVAL('user\_id\_seq');

На рисунке 6 представлена логическая модель базы данных.

Рисунок 6 – Логическая модель базы данных

На рисунке 7 представлена физическая модель базы данных.

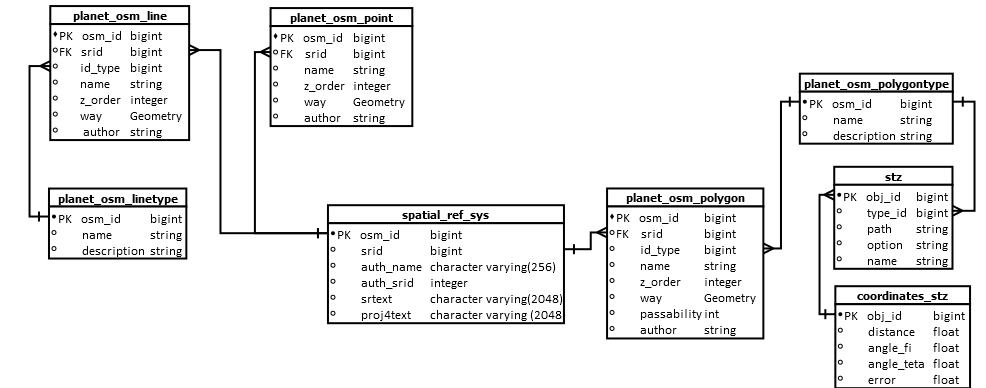
Рисунок 7 – Физическая модель базы данных

Таблица SPATIAL\_REF\_SYS имеет следующие столбцы:

* SRID - уникальный идентификатор системы координат (Spatial Referencing System, SRS) в пределах базы данных.
* AUTH\_NAME - название стандарта или стандартизирующего организации, являющейся источником информации о данной системе координат. Например, правильным значением AUTH\_NAME будет "EPSG".
* AUTH\_SRID - идентификатор системы координат, так как он определяется организацией указанной в AUTH\_NAME. В случае EPSG, это должен быть код проекции EPSG.
* SRTEXT - WKT представление системы координат.
* PROJ4TEXT - PostGIS использует библиотеку Proj4 для преобразований систем координат. Столбец PROJ4TEXT содержит строку определение координат Proj4 для данного SRID.

В PostGIS есть несколько типов пространственных объектов – точка, линия и полигон. Каждый объект, имеющий замкнутый контур, заносится в таблицу planet\_osm\_polygon, в которой хранятся данные о пространственных объектах типа полигон, таких как здания, горы, леса и других объектах, обладающих замкнутым контуром. Координаты, точки и местоположение объединены в таблице planet\_osm\_point, в этой таблице хранятся объекты типа точка. Таблица planet\_osm\_line содержит сущности, подходящие под тип линия – дорога, улица, река, маршрут и т.д. Таблица spatial\_ref\_sys содержит числовые идентификаторы и текстовые описания систем координат, используемых в пространственной базе данных. В таблицах planet\_osm\_linetype и planet\_osm\_polygontype, содержатся типы линий и полигонов соответственно.

## 7.4 Отображение карты

Для отображения карты была использована библиотека Mapnik[20], для её установки необходимо выполнить в терминале следующую команду: sudo aptget install python-mapnik [21]. На рисунке 8 показан принцип отображения карты.

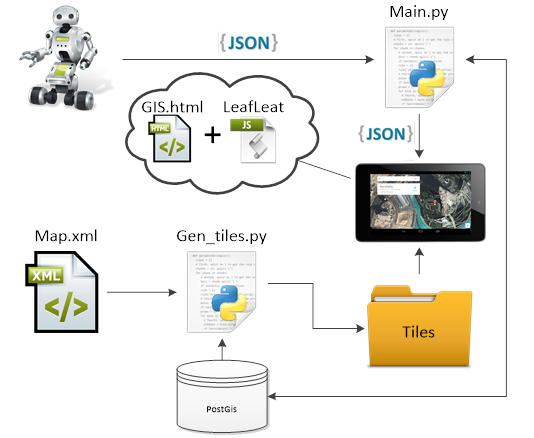


Рисунок 8 – Принцип отображения карты

Прежде всего, для отображения карты необходимо сгенерировать тайлы [22], для генерации тайлов за основу был использован скрипт Gen\_tiles.py, взятый с SVN репозитария Open Street Map. Репозитарий содержит множество скриптов и утилит по картографической тематике. Файл map.xml (мапфайл) – xml файл стилей для корректного отображения карты, данный файл также содержит данные о подключении к базе данных. При запуске скрипта Gen\_tiles.py начинается процесс генерации тайлов, которые записываются в папку Tiles. Исходные данные берутся из базы данных PostgreSQL.

После генерации тайлов необходимо сформировать из них целостную карту, для этого был использован Leaflet [23] – библиотека с открытым исходным кодом, написанная на JavaScript [24], которая служит для отображения карт на веб-сайтах. Данная библиотека используется для отображения карт на таких крупных сайтах как: OpenStreetmap и Wkimedia. При весе всего около 33 Кб, имеет широкий функционал, например, слои карты, могут быть интерактивными, т.е. при клике на какой-либо объект на карте есть возможность отобразить информацию о данном объекте. Разберем ключевые фрагменты кода в файле GIS.html.

Для создания карты Перми необходимо выполнить ряд действий:

1. Инициализация карты при помощи метода setView();

*var map = L.map('map').setView([58.05418,56.22439], 17);*

1. Добавление слоя тайлов к созданной карте;

*L.tileLayer('TileGenPerm/{z}/{x}/{y}.png', {*

*maxZoom: 17*

*}).addTo(map);*

1. Добавление слоя с местоположением роботизированной платформы и маршрута;

*L.geoJson(CurrLoc).addTo(map);*

*L.geoJson(MrkWay, {style: myStyle}).addTo(map);*

На рисунке 10 изображен интерфейс ГИС. При помощи элементов 1 – 4(См. Рис. 10), пользователь может произвести масштабирование карты и построение нового маршрута движения МРП. Пользователем является человек, управляющий МРП. Для выполнения данных операций, пользователю необходимо следовать алгоритму, представленному на рисунке 9.



Рисунок 9 – Масштабирование карты и построение нового маршрута

Пользователь может изменить масштаб карты, нажимая на 1 (См. Рис. 10) для приближения и 2 («-») для отдаления карты. Для построения нового маршрута, по которому будет следовать МРП, пользователю необходимо нажатиями на карту задать предполагаемый маршрут движения МРП, после нажатия на кнопку 3 начинается построение оптимального маршрута. В строке 5, отображается статус построения маршрута. Для сброса маршрута пользователю необходимо нажать на кнопку 4.



Рисунок 10 – Интерфейс ГИС

# Заключение

В ходе данной научно-исследовательской работы был проведен анализ существующих ГИС, выявлены их достоинства и недостатки. Определен формат представления данных, выбраны программно-аппаратные средства для реализации, произведено сопряжение с системой технического зрения и разработан web-интерфейс. Для корректного функционирования интерфейса был разработан сервер обработки запросов на языке Python.

Также, в данной работе был описан процесс создания базы данных PostgreSQL с расширением PostGIS; рассмотрен процесс импортирования данных с помощью программы osm2pgsql; описан принцип и реализация отображения карты, используя mapnik и LeafLeat, в геоинформационной системе.

# Список литературы

1. Файзрахманов Р. А., Курушин Д. С., Долгова Е. В. Принципы организации работ с применением мобильного робота // Научное обозрение. - 2014. - №7.

2. GIS and Robotics: Robotic Platforms as Tools for Spatial Data Collection and Consumption // Directions Magazine URL: http://www.directionsmag.com/ entry/ gis-and-robotics-robotic-platforms-as-tools-for-spatial-data-collection-and/123024 (дата обращения: 02.09.2015).

3. The National Science Foundation Center for Geographic Information and Analisis. - Int. J. Of Geographical Information Systems, 1987, v. I, N 4, pp. 302-306.

4. Fundamental operation in computer-assisted map analisis. - International Journal of Geographical Information Systems, 1987, v. I, pp. 119-136.

5. Введение в геоинформационные системы для начинающих — GIS-Lab [Электронный ресурс] // URL: http://wiki.gis-lab.info /w / Введение\_в\_геоинформационные\_системы\_для\_начинающих (дата обращения: 01.09.2015)

6. Типы данных ГИС // Isoline GIS Геоинформационная система для нефтяных компаний URL: http://www.isoline-gis.ru/gis\_types.htm (дата обращения: 05.09.2014).

7. Заставной Д.А. Представление атрибутивной информации в ГИС WinMap и язык WMSL // Электронный научный журнал «Инженерный вестник Дона». - 2011. - №1.

8. Jonker P., Caarls J., Bokhove W. Fast and Accurate Robot Vision // 4th International Workshop on RoboCup. - 2000. - С. 72-82.

9. About // PostgreSQL URL: http://www.postgresql.org/about/ (дата обращения: 01.09.2015).

10. About PostGIS // PostGIS — Spatial and Geographic Objects for PostgreSQL URL: http://postgis.net/ (дата обращения: 01.09.2015).

11. GIS-Lab: Руководство по PostGIS: Глава 6. Справочник PostGIS URL:http://gis-lab.info/docs/postgis/manual/ch06.html#id2701103(дата обращения: 29.12.2015)

12. Python: реализуем веб-сервер своими руками / Песочница / Хабрахабр URL:http://habrahabr.ru/sandbox/28540/ (дата обращения: 29.12.2015)

13. PostgreSQL + Python | Psycopg URL: http://initd.org/psycopg/ (дата обращения: 01.09.2015).

14. Using psycopg2 with PostgreSQL - PostgreSQL wiki [Электронный ресурс] // URL:https://wiki.postgresql.org/wiki/Using\_psycopg2\_with\_Postgre SQL (дата обращения: 01.09.2015)

15. The core of geospatial visualization & processing // Mapnik.org URL: http://mapnik.org/ (дата обращения: 08.09.2015).

16. ST\_AsGeoJSON. URL: http://postgis.net/docs/ST\_AsGeoJSON.html (дата обращения: 01.09.2015).

17. PostGIS 2.1 Raster Cheatsheet. URL: http://www.postgis.us/ downloads/ postgis21\_ raster\_ cheatsheet.pdf (дата обращения: 01.09.2015).

18. Openstreetmap/osm2pgsql // GitHub. URL: https://github.com/ openstreetmap/osm2pgsql (дата обращения: 08.09.2015).

19. PostGIS 2.2.1dev Manual. URL: http://postgis.net/docs/manual-2.2/ (дата обращения: 01.09.2015).

20. Mapnik • GitHub. URL: https://github.com/mapnik (дата обращения: 08.09.2015).

21. UbuntuInstallation mapnik/mapnik Wiki // GitHub. URL: https:// github.com/mapnik/mapnik/wiki/UbuntuInstallation (дата обращения: 08.09.2015).

22. Установка и настройка генератора тайлов на основе OSM данных в Ubuntu или Debian // Хабрахабр. URL: http://habrahabr.ru/post/144675/ (дата обращения: 12.08.2015).

23. Leaflet/Leaflet // GitHub. URL: https://github.com/Leaflet/Leaflet (дата обращения: 01.09.2015).

24. Lebresne S., Richards G., Wrigstad T., Vitek J. Understanding the dynamics of JavaScript // In Workshop on Script to Program Evolution. - 2009.