МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет

«Московский институт электронной техники»

Факультет микроприборов и технической кибернетики

Кафедра информатики и программного обеспечения вычислительных систем

Еленский Иван Владимирович

Бакалаврская работа

по направлению 09.03.04 «Программная инженерия»

Разработка программного модуля визуализации изображений местности на основе данных беспилотного летательного аппарата

Студент _____ Еленский И.В.

Научный руководитель,

к.т.н., доцент кафедры ИПОВС _____ Касимов Р.А.

Москва 2016

Содержание

Перечень сокращений

АНО – автономная некоммерческая организация

АО – акционерное общество

БПЛА – беспилотный летательный аппарат

ВИЗ – визуализация изображений местности

ВКР – выпускная квалификационная работа

ВПК – военная пожарная команда

ГИС – геоинформационная система

НИР – научно-исследовательская работа

НКО – некоммерческая организация

ПЗУ – постоянное запоминающее устройство

ПМ – программный модуль

ЦМР – цифровая модель рельефа

BCL – Base Class Library (стандартная библиотека классов платформы «.NET Framework»)

CLR – Common Language Runtime (общеязыковая исполняющая среда)

CTS – Common Type System (стандартная система типов платформы «.NET Framework»)

ER-модель – модель «сущность-связь»

GNSS – Global Navigation Satellite Systems (глобальная спутниковая система навигации) [2]

GPL – General Public License (открытое лицензионное соглашение)

GUI – graphical user interface (графический пользовательский интерфейс)

JVM – Java virtual machine (виртуальная машина Java)

RCP – rich client platform (платформа расширенного клиента) [3]

Введение

На сегодняшний день все чаще возникают задачи о получении информации с целью изучения местности для использования в самых разнообразных сферах, в частности, разведки местности и ее рельефа в военных операциях, прогнозирования метеорологической ситуации, проведения операций по спасению терпящих бедствие, и т.п. При этом критическим параметром является скорость обработки поступающей информации.

Применение непосредственного наблюдения порой сильно затруднено, так как наблюдателю требуется быстро достичь удобной точки обзора для сбора информации, что нередко сопряжено со значительными затратами времени и средств, а также с опасностью для здоровья и жизни наблюдателя. С течением времени применялись следующие методы:

- применение созданных заранее карт местности для прогнозирования событий, которые могут произойти в будущем;
- сооружение точек обзора для осуществления удаленного наблюдения вкупе с использованием систем получения и отправления сообщений (сигнальные жесты, огни, голубиная почта);
- применение воздушных шаров для проведения разведки с небольшой высоты, а в дальнейшем и дирижаблей;
- использование разведывательных самолетов, управляемых пилотом;
- съемка местности с орбитальных спутников, оборудованных модулями получения и передачи изображений.

Сегодня более актуальным решением поставленной задачи является применение аппаратных решений, которые позволяют наблюдателю-оператору оставаться на удалении от места съемки и в то же время получать данные высокого качества. Для проведения оперативного мониторинга в жестких рамках времени и при необходимости покрытия больших площадей способность беспилотных летательных аппаратов развивать внушительную скорость и подняться на значительную высоту нередко является важным параметром для решения поставленных задач.

Целью данной ВКР является разработка ПМ, обеспечивающего поддержку автоматизации полного цикла операций над обработкой и визуализацией получаемых данных с учетом требований оператора.

Пояснительная записка состоит из списка сокращений, содержания, введения, трех разделов и выводов после каждого из них, заключения, списка использованной литературы и двух приложений.

В исследовательском разделе проведен анализ предметной области, описаны ключевые параметры, которые необходимо учитывать при решении поставленной задачи, осуществлено сравнение существующих технологий-аналогов и изучено возможное их применение для решения поставленной задачи. Для ускорения процесса разработки ПМ выявлены основные функциональные возможности аналогов. Также описаны взаимодействия сущностей разрабатываемого ПМ на примере ER-диаграммы, составлены функциональные требования к ПМ и описан желаемый процесс работы ПМ ВИЗ.

В конструкторском разделе проведено сравнение языков и технологий программирования, выбраны Python3 и QML 2.0 при использовании библиотеки PyQt5, метода параллельного программирования и механизма слотов и сигналов. Проведено сравнение сред разработки, выбраны среда Qt Creator IDE и утилита qmlscene. Также описаны структуры входных и выходных данных, уточнены работа и взаимодействие сущностей, разработан алгоритм параллельной обработки блоков информации и уточнены детали реализации используемых алгоритмов обработки и вывода данных.

В технологическом разделе описаны применявшихся средства отладки программы, помимо средств используемых сред и утилит использована утилита интерактивной отладки модулей на языке Python pdb. Проведен анализ методов и средств тестирования, выбраны методы «белого ящика», модульного и интеграционного тестирований, составлены кейс-тесты, обеспечивающие оптимальное покрытие программного кода и основного функционала ПМ ВИЗ, на их основе проведены модульное и интеграционное тестирования, результаты которых проанализированы и учтены в ходе разработки.

В заключении описаны основные свойства разработанного программного модуля, также приведены отличительные особенности и преимущества использования ПМ ВИЗ, благодаря которым достигается высокая актуальность и скорость обработки данных.

Приложение 1 содержит полный текст программного кода ПМ ВИЗ.

Приложение 2 содержит документ «Руководство оператора».

1. Исследовательский раздел

1.1. Предварительные исследования

В настоящее время российские БПЛА применяются по пяти гражданским направлениям помимо ВПК. А именно: чрезвычайные ситуации (поиск людей, предупреждение ЧС, спасательные операции и т. д.); безопасность (охрана объектов и людей, а также их обнаружение); мониторинг (АЭС, ЛЭП, земельные, лесные, нефтегазовые, водные ресурсы, сельское хозяйство и пр.); аэрофотосъемка (геодезия, картография, авиаучет); наука (исследования Арктики, исследование оборудования, НИОКР).

Беспилотные самолеты используются, прежде всего, для мониторинга линейных и площадных участков местности. БПЛА способны преодолевать большие расстояния, выполняя сложнейшую аэросъемку онлайн при любых метеорологических условиях и в любое время суток. Максимальные эффективность выполняемых задач и качество работы возможны на удалении не более 70 километров от наземной станции управления. Развиваемая скорость достигает значения до 400 км/час. Время нахождения в полете составляет от 30 минут до 8 часов. [4]

Сегодня повсеместно используются два способа получения снимков местности: съемка с БПЛА и орбитальная съемка. Современный беспилотный самолет может успешно конкурировать со спутником на своей высоте полета. Отслеживая все, что происходит на территории площадью около миллиона квадратных километров, он сам становится своего рода «аэродинамическим спутником». Беспилотные самолеты могут конкурировать со спутниками и в сфере создания навигационных систем и телекоммуникационных сетей. На БПЛА можно возложить непрерывное круглосуточное наблюдение за поверхностью Земли в широком диапазоне частот.

Чтобы наблюдать за каким-нибудь объектом или вести фото- и видеосъемку из космоса, нужны 24 спутника, но при данных условиях информация от них будет поступать один раз в час. Данное ограничение возникает вследствие того, что спутник находится над объектом наблюдения всего 15-20 минут, а затем уходит из зоны его видимости и возвращается на то же место, совершив оборот вокруг Земли. Объект же за это время уходит из заданной точки, поскольку Земля вращается, и снова оказывается в ней только через 24 часа. В отличие от спутника, беспилотный самолет сопровождает точку

наблюдения постоянно. Проработав на высоте около 20 км более 24 часов, он возвращается на базу, а его сменяет другой. При этом стоимость одного спутника составляет порядка 100 миллионов долларов, 24 спутников - 2,4 миллиарда соответственно. В противовес этому стоимость трех современных беспилотных летательных аппаратов с наземной инфраструктурой составит немногим более 30 миллионов долларов.

Беспилотные летательные аппараты помогают решать целый спектр прикладных и научных задач, связанных с экологией, геологией, зоологией, сельским хозяйством, с изучением климата, метеорологией и поиском полезных ископаемых. Примерами могут служить наблюдения за миграцией птиц, косяков рыбы, млекопитающих, изменением ледовой обстановки на реках и других метеоусловий, перемещением транспорта и людей, движением судов, проведение аэро-, фото- и видеосъемки, радиационной и радиолокационной разведки, многоспектрального мониторинга поверхности, проникающего вглубь поверхности Земли до 100 метров. [5]

Учитывая свойства БПЛА, оптимальным решением задачи поиска объекта или изучения местности является применение камеры высокого разрешения на борту беспилотного летательного аппарата в совокупности с использованием сервисов географических карт для получения первичной информации о местности, которая сохраняется в виде отдельных изображений. При съемке с высоким разрешением с большой высоты и использовании широкого объектива камеры нерационально сохранять полученные данные в формате видео, поскольку при высоком разрешении снимков и высокой скорости беспилотного летательного аппарата можно считать, что движение снимаемых объектов незначительно.

Необходимо учитывать возможность поступления больших объемов информации, что в сочетании с необходимостью проведения ее обработки наиболее быстрым способом предполагает создание параллельного алгоритма работы ПМ, в ходе выполнения которого будет происходить блочная обработка данных.

Для выполнения всего комплекса требуемых функций с удовлетворительными показателями скорости и времени обработки данных на борту БПЛА требуется установить камеру высокого качества, обеспечивающую ведение съемки с разрешением один квадратный метр на один пиксель или лучше. Также необходимо оборудовать БПЛА процессором Intel Pentium IV с тактовой частотой не менее 2.0 ГГц, оперативной памятью не меньше 1 Гбайт и жестким диском объемом не менее 20 ГБ. Для увеличения скорости

обработки данных следует применять более совершенные компоненты, также рекомендуется использовать видеокарту, в которой реализована поддержка технологии CUDA NVidia версии не ранее 7.0, обладающая не менее 500 ядер CUDA, базовой тактовой частотой не менее 1 ГГц и объемом памяти не менее 2 Гб. БПЛА должен обладать модулями навигации, управления и передачи данных на землю. Для достижения высокой актуальности данных нужно учитывать тот факт, что пропускная способность канала передачи данных должна обеспечивать возможность передачи обработанных изображений оператору в кратчайшие сроки. Для удовлетворения этого требования пропускная способность канала должна составлять не менее 20-25 Мбит/сек, при этом желательно использовать модуль передачи данных технологии 4G или совершеннее. Для работы разрабатываемого ПМ требуется установить UNIX-совместимую операционную систему.

Для применения в гражданских целях БПЛА Supercam 350f является универсальным решением, особенно в случае проведения картографических и кадастровых работ. Аппарат обладает отличными летными характеристиками и может находиться в воздухе на протяжении длительного времени: максимальная дальность полёта Supercam 350-f составляет 360 км. Корпус самолета разработан таким образом, что на борту БПЛА может свободно размещаться аппаратура для аэрофотосъемки, при этом никак не влияя на летные качества летательного аппарата.

Сфера применения Supercam 350f чрезвычайно широка. Отличное соотношение цена – характеристики и универсальная конструкция данного БПЛА дают возможность эксплуатации в целях для мониторинга и аэрофотосъемки в строительстве, лесном хозяйстве, сельском хозяйстве, геодезии, энергетике и других областях.

Главным достоинством аппаратно-программного комплекса является то, что он автоматически выполняет сбор всех необходимых данных: для задания маршрута достаточно загрузить координаты рабочих точек, после этого оптимальную траекторию полета над ними БПЛА выберет самостоятельно.

Supercam 350-f оснащается профессиональной фотоаппаратурой, которая дает возможность получения снимков с высочайшей детализацией: с высоты в несколько сотен метров различимы объекты размером меньше метра. Такое качество фотографий существенно упрощает их последующую обработку и составление картографических планов. Каждый снимок, выполненный этим БПЛА, несет в себе полный набор данных,

включая высоту, крен, тангаж, угол наклона летательного аппарата, координаты местности и прочие полезные для автоматической обработки параметры.

Комплекс Supercam 350f запускается благодаря стартовому механизму, которым служит эластичная катапульта, позволяющая запускать аппарат не только в полевых, но и в городских условиях. Посадка может осуществлять в двух режимах: ручном и автоматическом. В первом случае оператор направляет устройство к точке посадки вручную, во втором же беспилотный комплекс приземляется согласно показателям, заданным на старте. [6]

При установке на борт БПЛА специального высокоточного двухчастотного GNSS-приёмника (опциональная установка) БПЛА S350-F становится геодезическим БПЛА. В течение полета БПЛА записывает высокодетальные фотографии, привязанные к GNSS, которые можно применять в программном обеспечении для создания фотопланов, ортофотопланов, ЦМР. Для каждого снимка имеется полный набор телеметрической информации: географические координаты центральной точки, угол и высота съемки, крен, скорость и тангаж БПЛА.

Планирование полета геодезического БПЛА S350-F осуществляется с помощью программного обеспечения, входящего в состав комплекса. При планировании маршрута учитывается:

- заданная путевая скорость БПЛА;
- высота полета над поверхностью земли;
- требуемое перекрытие снимков - частота съемки;
- тип объекта - протяженный или площадной объект;
- угол раствора объектива фотоаппарата.

При планировании программа автоматически рассчитывает полетное время, длину траектории полета, проводит проверку полетного задания на предмет соответствия эксплуатационным ограничениям данного БПЛА.

Достоинства использования комплекса S350-F:

- лучшие технические характеристики;
- высокоточный результат;
- безопасность;
- мобильность;
- оперативность;

- простота в использовании;
- современный автопилот (взлёт, выполнение полётного задания, возвращение, посадка – все этапы маршрута выполняет автопилот);
- выгодная стоимость комплекса;
- контроль полета в реальном времени. [7]

1.2. Программные продукты визуализации изображений местности

В настоящее время существует несколько программных продуктов, с помощью которых можно решить поставленную задачу отображения изображений местности в требуемый временной период.

Для ускорения разработки ПМ можно использовать популярные общедоступные продукты-ГИС. [8]

gvSIG — свободная геоинформационная система, исходный код которой является открытым и доступным всем желающим. Данная ГИС работает под многими операционными системами — Microsoft Windows, Apple Mac OS X, POSIX-совместимыми вообще и Linux в частности. Программа поддерживает все необходимые функции ГИС:

- функции масштабирования карты;
- поддержка сохранения необходимых ракурсов карты;
- работа со слоями, благодаря которой можно отображать лишь необходимые в данный момент объекты;
- размещение активных объектов на карте;
- автоматические расчёты расстояния между объектами и площадей областей;
- создание профессиональных географических карт с необходимыми элементами, которые можно впоследствии печатать.

Поддержка работы со многими базами данных и веб-сервисами и популярными растровыми и векторными форматами делают gvSIG очень привлекательным продуктом для применения. Данная ГИС распространяется под лицензией GNU GPL.

GRASS — еще один аналог ГИС с открытым исходным кодом. Данная ГИС построена по принципу модульности и интегрирует в себя множество различных модулей, которые решают задачи от визуализации до импорта/экспорта в различные форматы данных. Изначально система ориентирована на работу с командной строкой, но на данный

момент существуют два графических интерфейса к данной системе. Продукт представляет удобные механизмы для обработки изображений, управления данными, пространственного моделирования и производства графики. Распространяется, как и gvSIG, под лицензией GNU GPL. [9]

Продукты gvSIG и GRASS предоставляют идентичный функционал с точки зрения решения поставленной задачи. Внедрение данных продуктов не представляется возможным ввиду отсутствия возможности быстрого встраивания разрабатываемого кода в код продукта. Также потребуется убрать лишний функционал и разработать алгоритмы последовательной обработки поступающей информации, что уменьшит итоговую скорость работы ПМ, что неблагоприятно скажется на времени работы и актуальности получаемых данных. Невозможность применения сторонних модулей для обработки данных и открытость исходного кода продуктов не позволяют применять их для решения задачи.

Лучшим аналогом выступает продукт QGIS. Он позволяет просматривать и совмещать растровые и векторные данные в различных форматах и проекциях без преобразования в общий или какой-либо внутренний формат, при этом сохраняя соответствие различных форматов географической привязки. [10]

С помощью удобного графического интерфейса можно исследовать пространственные данные и создавать пользовательские карты. Графический интерфейс включает в себя множество полезных инструментов, которые обладают дружественным к пользователю интерфейсом, нежели интерфейс GRASS (из которого QGIS применила множество гибких и удобных решений), в некоторых аспектах и деталях он явно превосходит широко распространённые проприетарные ГИС. Приведем некоторые инструменты:

- перепроецирование «на лету»;
- панель обзора;
- определение/выборка объектов;
- подписывание объектов;
- редактирование/просмотр/поиск атрибутов;
- пространственные закладки;
- компоновщик карт;
- изменение символики векторных и растровых слоёв.

В QGIS можно создавать и редактировать векторные данные, а также экспортировать их в различные форматы. Чтобы иметь возможность редактировать и экспортировать в другие форматы растровые данные, необходимо сначала импортировать их в ГИС GRASS. Приведем самые значительные возможности QGIS по работе с данными:

- геокодирование изображений с помощью модуля пространственной привязки;
- визуализация и редактирование данных OpenStreetMap;
- инструменты GPS для импорта и экспорта данных в формате GPX, преобразования прочих форматов GPS в формат GPX или загрузка и скачивание непосредственно в прибор GPS (в Linux usb: был добавлен в список устройств GPS);
- сохранение снимков экрана как изображений с пространственной привязкой.

В настоящее время QGIS предоставляет возможность использовать инструменты анализа, выборки, геопроцессинга, управления геометрией и базами данных. Также можно использовать интегрированные инструменты GRASS, которые включают в себя функциональность более чем 300 модулей GRASS.

QGIS может быть адаптирован к особым потребностям с помощью расширяемой архитектуры модулей. QGIS предоставляет библиотеки, которые могут использоваться для создания модулей. Можно создавать отдельные приложения, используя языки программирования C++ или Python. [11]

Единственным недостатком продукта QGIS для решения проблемы является то, что QGIS распространяется под лицензией GNU GPL, что делает невозможным коммерческое применение разрабатываемого на его базе ПМ. Также нужно учитывать необходимость удаления лишнего функционала для повышения скорости работы.

Сравнительная характеристика аналогов и разрабатываемого ПМ приведена в таблице 1.1.

Таблица 1.1 – Результаты проведения предварительных НИР

Параметры	gvSIG Association. gvSIG	GRASS Development Team. GRASS	QGIS Development Team. QGIS	ПМ ВИЗ
Возможность внедрения модулей в существующую систему	Нет	Нет	Да	Да
Закрытый исходный код	Нет	Нет	Нет	Да

Продолжение таблицы 1.1

Необходимость адаптации к требованиям заказчика	Да	Да	Да	Нет
Возможность последующей модификации	Нет	Нет	Да	Да
Возможность использования сторонних модулей обработки изображений	Нет	Нет	Да	Да
Возможность параллельной обработки информации	Нет	Нет	Да	Да
Скорость параллельной обработки информации	-	-	10-15 Мб/с за 1 поток выполнения	60-80 Мб/с за 1 поток выполнения

[8] [9] [10] [11]

Как показывают результаты сравнения продуктов, задача, поставленная АО «ЭЛиПС», требует создание принципиально нового ПМ, который будет обеспечивать не только гибкость применения и скорость обработки поступающих данных, но и обладать свойством закрытости программного кода.

1.3. Цель и задачи исследования

Целью проводимого исследования является увеличение скорости проведения поисковых операций с помощью беспилотного летательного аппарата.

Для достижения поставленной цели необходимо решить ряд задач:

- провести исследование предметной области;
- составить подробное описание входных и выходных данных;
- провести сравнительный анализ существующих аналогов;
- провести разработку схемы данных ПМ ВИЗ;
- провести разработку схем алгоритмов работы ПМ ВИЗ;
- выбор инструментальных средств и среды разработки;
- разработать пользовательский интерфейс;

- провести программную реализацию;
- успешно завершить этапы отладки и тестирования;
- составить руководство применения ПМ для оператора.

1.4. Инфологическая модель данных

Для отображения взаимосвязей сущностей входных и выходных данных воспользуемся ER-диаграммой по методологии Питера Чена, изображенной на рисунке 1.1. [12]

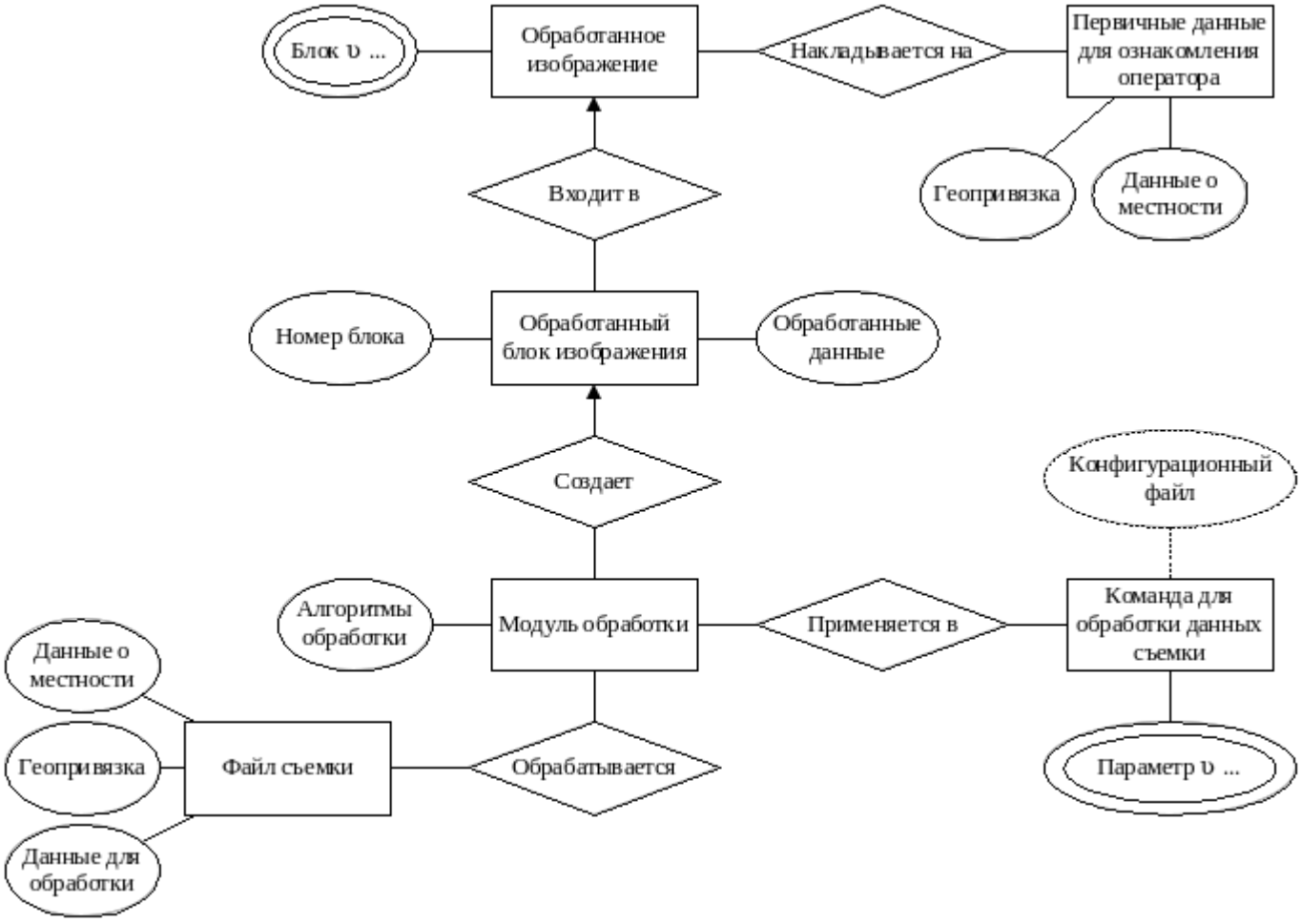

Рисунок *1*.1 – ER-диаграмма ПМ ВИЗ по методологии Питера Чена

Как видим из диаграммы, для работы ПМ требуется подать в качестве входных данных файл съемки и параметры обработки, которые могут быть записаны в файле-конфигураторе. В ходе работы ПМ в файл съемки может производиться запись, потому необходимо разработать алгоритмы по неблокирующему запись чтению файла. Первичные данные о местности загружаются на интерфейс пользователя после извлечения координат географической привязки. В дальнейшем эти данные будут частично или полностью

замещены обработанными данными-высококачественными изображениями с актуальной информацией. Для выполнения требуемых задач модуль обработки данных должен предоставлять возможность параллельной обработки поступающих данных.

Опишем сущности и их атрибуты более подробно. Файл съемки представляет собой файл специального формата, хранящего всю необходимую информацию о проводимой съемке местности с борта БПЛА (географические координаты центральной точки, высота и угол съемки, скорость, крен и тангаж борта БПЛА и другие параметры). После сохранены данные о проводимой съемке (запланированные размеры, сдвиги блоков и другие необходимые для проведения планирования обработки данные). Далее в файле съемки данных записываются результаты съемки, размещаемые в информационные блоки-единицы последующей обработки вспомогательным модулем. Данная структура позволяет эффективно и быстро получать результаты обработки информации. При использовании камеры высокого качества и применении алгоритмов экономичного хранения данных (черно-белый диапазон цветов) на пиксель необработанного изображения приходится 2 байта ПЗУ.

Команда для обработки данных съемки представляет собой набор параметров, используемых при работе модуля обработки. Набор состоит из пар «аргумент-значение», что облегчает работу с конфигурацией и делает ее прозрачной для пользователя. Команда применяется ко всем обрабатываемым блокам.

Модуль обработки представляет собой ПМ, содержащий различные алгоритмы (применение цифрового автофокуса, проведение оценки Доплера для полученных отличными от камеры способами данных, методы работы с геодезическими данными и другое), в ходе работы которых из «сырых» данных съемки получаются высококачественные изображения большого разрешения. Работа ПМ ВИЗ абстрагирована от работы модуля обработки, потому его можно легко заменить на любой другой, не изменяя программный интерфейс ПМ ВИЗ.

Перед началом обработки поступающих данных для пользователя выводится первичная информация о местности, в которой проводится съемка. Представляет собой данные из нескольких общедоступных сервисов электронных карт и ГИС на выбор пользователя. При дальнейшей обработке данных съемки частично замещаются на соответствующих географических позициях уже готовыми блоками итогового изображения. Перед началом обработки файла съемки ПМ ВИЗ формирует список

требуемых к обработке блоков данных и по мере их поступления и сформирования нового информационного блока приступает к обработке в параллельном режиме с другими заданиями.

Каждый поступающий в модуль обработки блок данных преобразуется в изображение, содержащее соответствующий участок местности. Каждая такая часть входит в итоговое изображение под определенным номером и географическим положением, чтобы получить соответствующую действительности картину местности.

1.5. Требования к алгоритмам работы программы, основанные на информационных потребностях пользователей

Так как процесс обработки должен быть завершен в кратчайшие сроки, то должна быть предусмотрена параллельная блочная обработка данных. После обработки полученное изображение должно сразу выводиться на экран, а в процессе обработки требуется получать данные о ходе обработки в виде сообщений, выводимых на пользовательский интерфейс. Таким образом, ход обработки, вывод сообщений о нем и изображения как результата работы составляют единую цепочку обработки информационного блока.

1.6. Общая постановка задачи

Требуется разработать ПМ для визуализации изображений местности, полученных из произвольного источника данных, с возможностью изменения параметров обработки поступающих данных для улучшения качества выходных изображений.

Для решения данной задачи разрабатываемый ПМ должен обеспечивать выполнение следующих функций:

- обработку данных с учетом введенных оператором параметров (размер и количество блоков, отрезы и отступы, циклический сдвиг блоков и др.);
- при поступлении большого объема данных (порядка нескольких гигабайт) должна проводиться блочная параллельная обработка;
- редактирование масштаба и угла поворота обработанных снимков;

- получение первичной информации о местности проведения съемки в виде карты из общедоступных сервисов;
- вывод результата обработки данных на экран оператора в соответствии с географической привязкой к местности.

При реализации описанных выше функций поставленная задача будет разрешена.

Выводы к исследовательскому разделу

1. Было проведено исследование предметной области, описаны ключевые параметры, которые необходимо учитывать при решении поставленной задачи;
2. Было осуществлено сравнение существующих технологий-аналогов и проанализировано возможное их применение для решения поставленной задачи;
3. Для ускорения процесса разработки ПМ были выявлены основные функциональные возможности аналогов;
4. Поставлены цели и задачи ВКР;
5. Описаны взаимодействия сущностей разрабатываемого ПМ на примере ER-диаграммы;
6. Составлены функциональные требования к ПМ;
7. Описан желаемый процесс работы ПМ ВИЗ.

2. Конструкторский раздел

2.1. Обоснование выбора средств и методов разработки

2.1.1. Выбор языка программирования

Прогресс компьютерных технологий определил процесс появления новых разнообразных знаковых систем для записи алгоритмов, которые сегодня известны как языки программирования. Значение появления такого языка – это образование оснащённого набора вычислительных формул дополнительной информации, что превращает данный набор в алгоритм. Язык программирования служит двум связанным между собой целям: он формирует концепции, которыми пользуется программист для достижения поставленной задачи, и он даёт программисту аппарат для задания действий, которые должны быть выполнены.

Сформируем список наиболее популярных языков программирования и проведем обзор на установление их соответствия для решения поставленной задачи:

- C;
- C++;
- Python;
- QML;
- Java;
- Ruby;
- C#;
- Perl;
- PHP.

C — статически типизированный компилируемый язык программирования общего назначения. Был разработан как развитие языка Би. Будучи первоначально разработанным для реализации операционной системы UNIX, в дальнейшем он был перенесён на множество других платформ. Согласно дизайну языка C, его конструкции близко сопоставляются типичным машинным инструкциям, благодаря чему он нашёл широкое применение и распространение в проектах, для которых был свойственен язык ассемблера, в том числе как в операционных системах, так и в отрасли различного прикладного ПО для множества устройств — от встраиваемых систем до суперкомпьютеров. [13]

Одно из самых примечательных свойств языка программирования C – это отличающий его минимализм. Код на C можно легко писать на низком уровне абстракции, почти как на ассемблере. Иногда C называют «ассемблером высокого уровня» или «универсальным ассемблером», что отражает единство стандарта C, код которого может быть скомпилирован без изменений практически на любой модели компьютера, и различие языков ассемблера для разных платформ. C часто называют языком среднего или даже низкого уровня, учитывая то, насколько близко он работает к реальным устройствам. Но в строгой классификации он является языком высокого уровня.

C создавался с одной важной целью: сделать написание больших программ с минимумом ошибок более простым, используя правила процедурного программирования, не добавляя на итоговый код программ дополнительных и часто лишних накладных расходов для компилятора, как это делают многие языки высокого уровня. Потому с данной точки зрения C предлагает следующие отличительные особенности:

- ориентацию на процедурное программирование, обеспечивающую удобство применения структурного стиля программирования;
- использование препроцессора для, например, определения макросов и включения файлов с исходным кодом;
- непосредственный доступ к памяти компьютера через использование указателей;
- простую языковую базу, из которой вынесены в библиотеки многие существенные возможности, вроде математических функций или функций управления файлами;
- систему типов, предохраняющую от бессмысленных операций;
- структуры и объединения — определяемые пользователем собирательные типы данных, которыми можно манипулировать как одним целым;
- средства объектно-ориентированного программирования и другие особенности.

После своего появления язык C был хорошо принят, так как он позволял создавать компиляторы для новых платформ быстро и в кратчайшие сроки, а также позволял программистам довольно точно представлять, как выполняются их программы. Благодаря этому программы, написанные на C, работают эффективнее написанных на многих других языках. Как правило, лишь вручную оптимизированный код на ассемблере может работать ещё быстрее, так как он даёт полный контроль над машиной и позволяет раскрыть весь ее вычислительный потенциал, однако усложнение современных процессоров вместе с развитием современных компиляторов сократило этот разрыв. [14]

Одним из последствий переносимости и высокой эффективности C стало то, что многие компиляторы, интерпретаторы и библиотеки других языков высокого уровня часто написаны на языке C.

Однако данный язык не лишён недостатков: у него достаточно высокий порог вхождения, что затрудняет его использование в обучении в качестве первого языка программирования. Он стимулирует часто небезопасный и довольно специфичный стиль программирования, поощряющий написание запутанного кода. Вместе с тем многие элементы C потенциально опасны, при этом зачастую последствия неправильного использования этих элементов непредсказуемы. В связи со сравнительно низким уровнем языка многие случаи неправильного использования опасных элементов не могут быть обнаружены ни при компиляции, ни во время исполнения, что часто приводит к непредсказуемому поведению программы. Иногда в результате неграмотного использования элементов языка появляются уязвимости в системе безопасности, при этом необходимо заметить, что использования многих таких элементов можно избежать. Более того, за более чем 40 лет существования, язык успел несколько устареть, и в нём достаточно проблематично использовать многие современные приёмы и парадигмы программирования.

Язык программирования C оказал значительное влияние на развитие индустрии программного обеспечения, при этом его синтаксис стал основой для таких языков программирования, как C++, C#, Objective-C и Java. Язык C уникален, так как именно он стал первым языком высокого уровня, существенно потеснившим ассемблер в разработке системного программного обеспечения. Он остаётся одним из самых популярных языков программирования, реализованным на максимальном количестве аппаратных платформ, при этом наибольшим распространением он обладает в мире свободного программного обеспечения.

C++ - это универсальный язык программирования. За исключением второстепенных деталей C++ является надмножеством языка программирования C. Помимо возможностей, которые дает C, C++ предоставляет эффективные и гибкие средства определения новых типов. Используя определения новых типов, точно отвечающих концепциям приложения, программист может разделять разрабатываемую программу на легко поддающиеся контролю части. Такой метод построения программ часто называют абстракцией данных. Информация о типах содержится в некоторых объектах типов, определённых

пользователем. Такие объекты просты и надёжны в использовании в тех ситуациях, когда их тип нельзя установить на стадии компиляции. Программирование с применением таких объектов часто называют объектно-ориентированным. При правильном использовании этот метод даёт легче контролируемые программы, более короткие и проще понимаемые. [15]

C++ предлагает программисту полный набор операторов структурного программирования. Он также обладает очень большим набором операций. Многие операции C++ соответствуют машинным командам, и поэтому допускают прямую трансляцию в код ассемблера. Разнообразие операций позволяет выбирать их различные наборы для минимизации результирующего поля. C++ поддерживает указатели на переменные и функции. Указатель на объект программы соответствует машинному адресу данного объекта. Посредством разумного использования указателей можно создавать эффективные программы, которые выполняются быстро, так как указатели позволяют ссылаться на объекты тем же самым путём, как это делает машина. C++ поддерживает алгебру указателей, и тем самым позволяет осуществлять прямой доступ и непосредственные манипуляции с адресами памяти. В своём составе C++ содержит препроцессор, который обрабатывает текстовые файлы перед компиляцией. Среди его полезных приложений при написании программ на C++ являются: определение программных констант, условная компиляция и замена вызова функций аналогичными, но более быстрыми макросами. Препроцессор не ограничен процессированием только исходных текстовых файлов C++, он может быть использован для любого текстового файла.

C++ - гибкий язык, позволяющий принимать в конкретных ситуациях самые разнообразные решения. В C++ полностью поддерживаются принципы объектно-ориентированного программирования, включая три кита, на которых оно стоит: наследование, инкапсуляцию, и полиморфизм. Язык C++ поддерживает наследование. Это значит, что можно объявить новый тип данных (класс), который является расширением существующего. Инкапсуляция в C++ поддерживается посредством создания пользовательских типов данных, называемых классами. [16]

Хотя язык C++ справедливо называют продолжением C, и любая работоспособная программа на языке C будет поддерживаться компилятором C++, при переходе от C к C++ был сделан весьма существенный скачок. Язык C++ выигрывал от своего родства с языком C в течение многих лет, поскольку сообщество программистов обнаружило, что для того,

чтобы в полной мере воспользоваться особенностями и преимуществами языка C++, им нужно приобрести новые и отказаться от некоторых своих прежних знаний, а именно: изучить новый способ решения проблем программирования и концептуальности. [17]

Python – высокоуровневый язык программирования общего назначения, который ориентирован на повышение читаемости кода и производительности разработчика. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных в ходе разработки функций.

Python поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное, императивное, аспектно-ориентированное и функциональное. Основными архитектурными чертами являются такие особенности как автоматическое управление памятью, динамическая типизация, механизм обработки исключений, полная интроспекция, гибкие высокоуровневые структуры данных и поддержка многопоточных вычислений. Код в Python организовывается в классы и функции, которые могут объединяться в модули, которые могут быть объединены в пакеты. [18]

Эталонной реализацией Python является интерпретатор CPython, поддерживающий большинство активно используемых платформ. Он распространяется под свободной лицензией Python Software Foundation License, которая позволяет использовать его без ограничений в любых приложениях, включая проприетарные.

Наиболее часто Python сравнивают с Ruby и Perl. Эти языки обладают примерно одинаковой скоростью выполнения программ и также являются интерпретируемыми. Как и Ruby, Python является хорошо продуманной системой для ООП. Сравнимо с Perl, Python также может успешно применяться для написания различных скриптов. В среде коммерческих приложений скорость выполнения программ на Python часто сравнивают с приложениями на языке программирования Java. [19]

Богатая стандартная библиотека является одной из привлекательных сторон Python. Здесь имеются средства для работы со многими форматами и сетевыми протоколами Интернета, например, модули для написания HTTP- клиентов и серверов, для работы с XML, для создания и разбора почтовых сообщений и т.п. Существуют модули для работы с текстовыми кодировками, регулярными выражениями, сериализацией данных, архивами, криптографическими протоколами, мультимедийными форматами, также присутствует поддержка юнит-тестирования и др. Набор модулей для работы с операционной системой

позволяет писать кроссплатформенные приложения. Наряду со стандартной библиотекой существует большое количество сторонних библиотек, значительно уменьшающих время разработки и увеличивающих функциональность. Python и подавляющее большинство библиотек к нему поставляются в исходных кодах и являются бесплатными. Более того, в отличие от многих открытых систем, лицензия не налагает никаких обязательств кроме указания авторских прав и никак не ограничивает использование Python в коммерческих разработках.

В число недостатков входят низкое быстродействие (хотя реализованное на Python множество программ и библиотек для интеграции с другими языками программирования предоставляют возможность использовать другой язык для написания критических участков, потому для сохранения совместимости со многими языками программирования модуль обработки снимков реализован на C++), глобальная блокировка интерпретатора и невозможность модификации встроенных классов. Данные недостатки не являются серьезными для разработки и работы ПМ ВИЗ. [20]

QML (Qt Meta-Object Language) —язык программирования, являющийся декларативным. Он основан на языке JavaScript и предназначен для создания дизайна приложений, делающих основной упор на графический интерфейс и работу с пользователем. QML является частью Qt Quick, среды разработки пользовательского интерфейса, которая распространяемой вместе с Qt. В основном QML используется для создания приложений, ориентированных на мобильные устройства с поддержкой сенсорного управления, при этом является (поддерживает):

- декларативным языком (т.е. описывает интерфейс);
- мета-объектную систему Qt;
- быструю разработку функциональных и удобных интерфейсов;
- легкое расширение функционала QML с помощью плагинов (Qt/C++/Python);
- описание логики работы с помощью JavaScript. [21]

Java – объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems, которая позже была приобретена компанией Oracle. Программы на Java транслируются в байтовый код, который выполняется виртуальной машиной Java (JVM) — программой, обрабатывающей байт-код и в качестве интерпретатора передающей инструкции оборудованию. [22]

Преимуществом данного способа выполнения программ является полная независимость байт-кода от аппаратно-программного комплекса, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важным достоинством технологии Java является гибкая система безопасности, в границах которой выполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного соединения с другим компьютером или доступа к данным), вызывают немедленное прерывание. [23]

Часто к недостаткам технологии виртуальной машины относят снижение производительности. Следующий ряд улучшений несколько увеличил скорость исполнения программ на Java:

- широкое использование платформенно-ориентированного кода (native-код) в стандартных библиотеках;
- применение технологии трансляции байт-кода в машинный код непосредственно во время работы программы (JIT-технология) с возможностью сохранения версий класса в машинном коде;
- аппаратные средства, обеспечивающие ускоренную обработку байтового кода (например, технология Jazelle, которая поддерживается некоторыми процессорами фирмы ARM). [24]

Основные возможности Java:

- расширенные возможности обработки исключительных ситуаций;
- автоматическое управление памятью;
- набор стандартных коллекций: массив, список, стек и т. п.;
- богатый набор средств фильтрации ввода-вывода;
- наличие классов, позволяющих выполнять HTTP-запросы и обрабатывать ответы;
- наличие простых средств создания сетевых приложений (в том числе с использованием протокола RMI);
- унифицированный доступ к базам данных:
    - на уровне отдельных SQL-запросов — на основе JDBC, SQLJ;
    - на уровне концепции объектов, обладающих способностью к хранению в базе данных — на основе Java Data Objects и Java Persistence API;

- встроенные в язык средства создания многопоточных приложений;
- параллельное выполнение программ;
- поддержка лямбд, обобщений, замыканий, встроенные возможности функционального программирования. [25]

Ruby – рефлективный, интерпретируемый и динамический высокоуровневый язык программирования, который замечательно подходит для удобного и быстрого объектно-ориентированного программирования. Язык обладает следующими возможностями:

- полностью объектно-ориентированный язык программирования, т.е. все данные в Ruby являются объектами в понимании Smalltalk. Также поддерживается добавление методов в класс и даже в конкретный экземпляр класса во время выполнения программы [26];
- не поддерживает множественное наследование, но вместо него может использоваться концепция «примесей», основанная в данном языке на механизме модулей;
- позволяет обрабатывать исключения в стиле Python и Java;
- содержит автоматический сборщик мусора, который работает для всех объектов Ruby, в том числе и для подключенных внешних библиотек;
- позволяет переопределять операторы, которые в действительности являются методами;
- целочисленные переменные в Ruby автоматически конвертируются между типами Fixnum (32-разрядные) и Bignum (более 32 разрядов) в зависимости от хранимого в них значения, что позволяет производить целочисленные математические расчёты со сколь угодно большой точностью;
- поддерживает замыкания с полной привязкой к переменным;
- в Ruby непосредственно в языке реализованы многие шаблоны проектирования [27];
- может динамически загружать расширения, если это позволяет сделать операционная система;
- создавать расширения для Ruby на C очень просто частично из-за сборщика мусора, частично из-за несложного и удобного API;
- имеет независимую от операционной системы поддержку невытесняющей многопоточности.

Кроссплатформенная реализация интерпретатора языка является полностью свободной, а большинство расширений могут быть использованы в любом проекте

практически без ограничений, так как распространяются под свободными лицензиями (LGPL, лицензия Ruby). [28]

C# – объектно-ориентированный язык программирования. Разработан в 1998-2001 годах в компании Microsoft в качестве языка разработки приложений для платформы Microsoft .NET Framework и впоследствии был стандартизирован как ISO/IEC 23270 и ECMA-334.

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

Переняв многое от своих предшественников – языков Pascal, C++, Модула, Smalltalk и, в особенности, Java – C#, опираясь на практику их применения, исключает некоторые модели использования, зарекомендовавшие себя как проблематичные при разработке программных систем. Так, в отличие от C++, C# не поддерживает множественное наследование классов (но между тем допускается множественное наследование интерфейсов). [29]

C# разрабатывался как язык программирования прикладного уровня для CLR и потому прежде всего зависит от возможностей самой CLR. Это касается, прежде всего, системы типов C#, которая отражает BCL. Отсутствие или присутствие каких-либо выразительных особенностей языка диктуется тем, может ли конкретная языковая особенность быть транслирована в соответствующие конструкции CLR. Так, с развитием CLR от версии 1.1 к 2.0 значительно преобразился и сам C#; подобного взаимодействия следует ожидать и в дальнейшем (стоит заметить, что данная закономерность была нарушена с выходом C# версии 3.0, представляющего собой расширения языка, не опирающиеся на расширения платформы .NET). CLR предоставляет C#, как и всем другим .NET-ориентированным языкам, многие возможности, которых лишены «классические» языки программирования. Например, сборка мусора не реализована в самом C#, а производится CLR для программ, написанных на C# точно так же, как это делается для программ на VB.NET, J# и др.

Взаимодействие между различными языками является ключевой особенностью .NET Framework. Поскольку код на промежуточном языке (IL), создаваемый компилятором C#, соответствует спецификации CTS, промежуточный код на основе C# может взаимодействовать с кодом, который создается версиями языков Visual Basic, Visual J#, Visual C++ платформы .NET Framework и еще другими более чем 20 CTS-совместимых языков. В одной сборке может быть несколько модулей, написанных на разных языках платформы .NET Framework, при этом типы могут ссылаться друг на друга как в том случае, если бы они были написаны на одном языке. [30]

Помимо служб времени выполнения, в .NET Framework также имеется богатая библиотека, состоящая из более чем 4000 классов, структурированных по пространствам имен, которые обеспечивают разнообразные полезные функции для любых действий, начиная от элементов управления Windows Forms до ввода и вывода файлов для управления строками для разбивки XML. В обычном приложении на языке C# библиотека классов .NET Framework интенсивно используется в ходе написания кода программы.

C# – объектно-ориентированный язык, который является элегантным, типобезопасным и предназначенным для разработки разнообразных мощных и безопасных приложений, выполняемых в среде .NET Framework. С помощью языка C# можно создавать обычные приложения Windows, распределенные компоненты, XML-веб-службы, приложения «клиент-сервер», приложения баз данных и т.д. [31]

Perl – динамический интерпретируемый высокоуровневый язык программирования общего назначения, который был первоначально предназначен для манипуляций с текстом, но на данный момент активно используется для выполнения широкого спектра задач, включая веб-разработку, системное администрирование, сетевое программирование, биоинформатику, игры и разработку графических пользовательских интерфейсов. [32]

Главными достоинствами языка являются поддержка различных парадигм (процедурный, функциональный и объектно-ориентированный стили программирования), встроенная поддержка обработки текста, большая коллекция модулей сторонних разработчиков, а также контроль за памятью (без сборщика мусора, основанного на циклах).

Perl унаследовал много свойств от языков C, AWK, скриптовых языков командных оболочек UNIX. Основной особенностью языка считаются его богатые возможности для работы с текстом, в том числе работа с регулярными выражениями, встроенная в

синтаксис. Perl обладает множеством встроенных функций, которые обеспечивают набор инструментов, часто используемых для программирования оболочки, например, вызов системных служб или сортировку. [33]

Все версии Perl выполняют автоматический контроль над памятью и автоматическую типизацию данных. Интерпретатор знает запросы памяти и тип каждого объекта программы, он освобождает и распределяет память, производя подсчёт ссылок. Перевод одного типа данных в другой — например, числа в строку — происходит автоматически во время исполнения, невозможные для выполнения переводы типов данных приводят к фатальной ошибке. [34]

PHP – скриптовый язык общего назначения, который активно применяется для разработки веб-приложений. В настоящее время является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов и поддерживается подавляющим большинством хостинг-провайдеров. Язык и его интерпретатор находятся в открытом доступе. Проект распространяется под собственной лицензией, которая несовместима с GNU GPL. [35]

В области веб-программирования, особенно в серверной компоненте, PHP — один из популярных сценарных языков (наряду с Perl, JSP, и языками, используемыми в ASP.NET). Популярность в области разработки веб-сайтов обосновывается наличием большого набора встроенных средств для разработки веб-приложений. Основные из них:

- автоматическое извлечение POST и GET-параметров, а также переменных окружения веб-сервера в предопределённые массивы;
- работа с HTTP-авторизацией;
- автоматизированная отправка HTTP-заголовков;
- взаимодействие с большим количеством различных систем управления базами данных (MySQLi, MySQL, SQLite, PostgreSQL, Oracle (OCI8), Oracle, Microsoft SQL Server, Sybase, SESAM, ODBC, mSQL, Apache Derby и IBM DB2, Informix, Lotus Notes, Cloudscape, dBase, DB++, DBM, DBX, FrontBase, FilePro, Ingres II, Ovrimos SQL, Firebird / InterBase, Paradox File Access, MaxDB);
- работа с cookies и сессиями;
- обработка файлов, загружаемых на сервер;
- работа с локальными и удалёнными файлами, сокетами;
- работа с XForms.

Несмотря на то что PHP не очень распространён в области создания приложений с пользовательским интерфейсом, его можно использовать для создания кроссплатформенных приложений с помощью пакетов PHP-Qt и PHP-GTK, представляющих собой обёртки для соответствующих популярных библиотек графических элементов и виджетов. Для создания графических приложений для платформы Windows существуют свободный пакет WinBinder, который написан на C и практически использует вызовы для WinAPI. Также существуют реализации PHP для JVM — JPHP и для .NET/Mono — Phalanger. JPHP поддерживает расширение Swing, почти полностью интегрированное из среды Java, результатом работы компилятора PHP-кода в Phalanger может быть любое .NET-приложение. [36]

Недостатками PHP являются:

- отсутствие поддержки многопоточности, что непозволяет разрабатывать приложения для быстрой обработки данных;
- неортогональность и несогласованный синтаксис функций, что замедляет разработку приложения;
- отсутствие обратной совместимости между версиями языка, что усложняет переход на новую версию языка и поддержку реализованных продуктов. [37]

Проведем сравнение языков программирования на основе критических параметров для разработки и работы ПМ ВИЗ, результаты приведены в таблице 2.1. Результаты нагрузочных тестов помечены как необязательные к рассмотрению, т.к. модуль обработки снимков (критическое место работы ПМ ВИЗ) реализован на языке C++ для оптимизации времени работы и эффективности. [38]

Таблица 2.1 – Сравнение языков программирования

Параметры (необязательные выделены курсивом)	Языки программирования							
	C	C++	Python + QML	Java	Ruby	C#	Perl	PHP
Возможность компиляции	Да	Да	Да	Да	Да	Да	Да	Да
Динамическая типизация	Нет	Да	Да	Да	Да	Да	Да	Да

Продолжение таблицы 2.1

Поддержка ООП	Нет	Да	Да	Да	Да	Да	Да	Да
Наличие готовых шаблонов и библиотек для решения задачи	Нет	Да	Да	Нет	Нет	Да	Нет	Нет
Возможность применения модулей, реализованных на других языках программирования	Нет	Нет	Да	Да	Да	Да	Да	Да
Время решения нагрузочной тестовой задачи (нахождение 30 чисел Фибоначчи) в секундах	0.013	0.014	1.109	0.176	0.566	0.184	2.335	1.307
Время решения нагрузочной тестовой задачи (соединение строк в цикле до 4 Мб) в формате «час:мин:сек»	0:09:15	0:28:51	0:45:20	4:37:54	0:40:55	3:58:36	0:07:17	0:31:09

[14] [16] [20] [21] [23] [27] [29] [33] [35]

На основе данных из источников [39], [40] и [41] дополним сравнение языков программирования результатами выполнения двух нагрузочных тестовых задач, включающих скорость проводимых вычислений и обработки данных на примере нахождения чисел Фиббоначи и соединения строк в цикле. Стоит отметить, что данные результаты носят обзорный характер и не являются обязательными параметрами.

В результате сравнения языков программирования оптимальным вариантом является сочетание Python и QML. Для написания модулей логики и взаимодействия будет

применяться Python, для создания пользовательского интерфейса – QML, установление связи между модулями будет обеспечено с помощью механизма слотов и сигналов и библиотеки PyQt5. [42] [43]

2.1.2. Выбор среды разработки

На сегодняшний день на рынке информационных технологий существует большое количество различных средств разработки приложений. Приведем список наиболее подходящих для разработки на языке программирования Python сред:

- MS Visual Studio;
- Qt Creator IDE;
- Eclipse IDE;
- Spider IDE.

Visual Studio 2015 — это интегрированная среда разработки, которая обладает большими возможностями для создания замечательных приложений для Android, iOS и Windows, а также современных облачных служб и веб-приложений. Visual Studio анализирует код и подсказывает программисту, какие типы и где следует использовать. Всплывающие завершения, подсказки и фрагменты кода повышают производительность работы и упрощают процесс разработки. Доступ к функциональной и удобной среде IntelliSense, преимущества простой навигации по коду и быстрой сборки и развертывания в кратчайшие сроки уменьшают время разработки. Visual Studio позволяет удобно работать в составе рабочей группы и самостоятельно и повышает продуктивность.

Преимущества Visual Studio при работе с Python:

- подробная IntelliSence;
- наличие библиотек сторонних поставщиков и средств Pip, PyPI и поддержки виртуальной среды для управления проектами и зависимостями;
- CPython, PyPy, IronPython и многое другое;
- интеграция с функциями Visual Studio;
- интерактивная отладка (визуальное пошаговое выполнение кода, взаимодействие с программой независимо от операционной системы и просмотр или изменение состояния программы) в REPL Python;

- можно использовать с открытым кодом и бесплатно, если разработка идет в исследовательских целях.

Встроенный отладчик может работать в качестве отладчика уровня исходного кода, так и в качестве отладчика машинного уровня. Остальные встраиваемые инструменты включают в себя веб-редактор, дизайнер схемы базы данных, дизайнер классов и редактор форм для упрощения создания графического интерфейса приложения. Visual Studio позволяет подключать и создавать сторонние расширения (плагины) для дополнения функциональности практически на любом уровне, включая добавление новых наборов инструментов (например, для визуального проектирования и редактирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server), также возможно добавление поддержки систем контроля версий исходного кода (например, Visual SourceSafe и Subversion). [44]

Qt Creator – кроссплатформенная свободная IDE для разработки на языках программирования C, C++ и QML. Данная среда разработана для работы с фреймворком Qt и включает в себя визуальные средства разработки интерфейса как с использованием QML, так и QtWidgets, и графический интерфейс отладчика. Поддерживаемые компиляторы: GCC, MinGW, Clang, Linux ICC, MSVC, WINSCW, RVCT, GCCE.

Основной задачей Qt Creator является упрощение разработки приложения с помощью фреймворка Qt на различных платформах. Потому среди возможностей, присущих любой среде разработки, есть и специфичные, такие как отладка приложений на QML и отображение в отладчике данных из контейнеров Qt, встроенный дизайнер интерфейсов как на QML, так и на QtWidgets.

Qt Creator поддерживает системы сборки cmake, qmake, qbs, autotools. Есть возможность редактирования этапов сборки проекта. Для проектов, созданных под другими системами, существует возможность применения в качестве редактора исходных кодов. Также IDE нативно поддерживает системы контроля версии, такие как Git, Mercurial, Subversion, Perforce, Bazaar, CVS. В Qt Creator реализовано автодополнение, в том числе ключевых слов, введённых в стандарте C++11 (начиная с версии 2.5), подсветка кода (её определение аналогично таковому в Kate, что позволяет использовать уже готовые виды подсветок или создавать свои). Также есть возможность задания отступов, стиля выравнивания и постановки скобок.

Реализован ряд возможностей при работе с сигнатурами методов, к ним относятся следующие:

- автогенерация пустого тела метода после его обновления;
- возможность автоматически поменять порядок следования аргументов;
- возможность автоматически изменить сигнатуру метода в объявлении, если она была изменена в определении, аналогичный механизм срабатывает в определении, если изменения произошли в сигнатуре объявления метода.

При навигации по коду можно использовать переход к объявлению метода и переключение между определением и объявлением метода, также возможно переименование метода как в отдельном проекте, так и во всех открытых. Присутствует возможность вызвать справку согласно текущему контексту.

Среда разработки имеет графический интерфейс для следующих популярных отладчиков: GDB, QML/JavaScript и CDB. В качестве отдельной опции реализовано отображение содержимого контейнеров, таких как QString, std::map и прочих. Поддерживаются следующие режимы отладки:

- терминал для отладки локально запущенных процессов, которым требуется консоль;
- простой для отладки локально запущенных приложений, таких как GUI приложения на Qt;
- удалённый для отладки запущенных на другой машине процессов с использованием gdbserver;
- подключенный для отладки локальных процессов, запущенных вне Qt Creator;
- ядро для отладки завершившихся аварийно процессов на Unix;
- TRK для отладки процессов, запущенных на устройстве Symbian;
- отладка исключения, приводящего к сбою программы на Windows.

Точки остановки можно задать различными удобными способами, упрощая отладку, поддерживаются следующие способы:

- остановка на заданной строчке заданного файла;
- остановка при обращении к данным по заданному адресу;
- остановка при поимке исключения;
- остановка при вызове функции с определенным именем;
- остановка при выполнении системного вызова;
- остановка при запуске или создании нового процесса;

- остановка при изменении в данных с адресами, заданными выражением. [45]

Eclipse – свободная интегрированная среда разработки, разработанная для создания кроссплатформенных модульных приложений. Поддерживается и развивается сообществом Eclipse Foundation. Наиболее известными приложениями на основе Eclipse Platform являются различные «Eclipse IDE» для разработки ПО на множестве языков (например, наиболее популярный «Java IDE», который поддерживался первоначально, не основывается на какие-либо закрытых расширениях, а использует стандартный открытый аппаратно-программный интерфейс для доступа к Eclipse Platform).

Eclipse служит в первую очередь платформой для разработки дополнений, благодаря чему он и стал популярен: любой разработчик может расширить Eclipse своими модулями, дополнив его необходимым функционалом. Уже существуют C/C++ Development Tools (CDT), разрабатываемые инженерами QNX совместно с IBM, Java Development Tools (JDT) и средства для языков Ada (GNATbench, Hibachi), FORTRAN, COBOL, X10 (X10DT), PHP и других от разных разработчиков. Множество дополнений расширяет среду Eclipse диспетчерами для работы с серверами приложений, базами данных и др.

Eclipse написана на Java, потому является кроссплатформенным продуктом, за исключением библиотеки SWT, которая разрабатывается для всех распространённых платформ. Вместо стандартной для Java библиотеки Swing используется библиотека SWT. Она полностью опирается на нижележащую платформу (операционную систему), что обеспечивает натуральный внешний вид и быстроту пользовательского интерфейса, но иногда вызывает на разных платформах проблемы устойчивости и совместимости приложений.

Основой Eclipse является RCP, в которую входят следующие широко используемые компоненты:

- ядро платформы (загрузка Eclipse, запуск модулей);
- SWT (портируемый инструментарий виджетов);
- OSGi (стандартная среда поставки комплектов);
- рабочая среда Eclipse (панели, редакторы, проекции, мастеры);
- JFace (файловые буферы, работа с текстом, текстовые редакторы).

Пользовательский интерфейс среды Eclipse создан с применением инструментария SWT. Последний, в отличие от библиотеки Swing, которая самостоятельно эмулирует графические элементы управления, использует графические компоненты данной

операционной системы, вследствие чего интерфейс работает быстрее и выглядит естественно и гармонично на фоне операционной системы. Пользовательский интерфейс Eclipse также зависит от промежуточного слоя пользовательского интерфейса, называемого JFace, который упрощает построение GUI, базирующегося на SWT.

Удобство применения Eclipse обеспечивается за счёт подключаемых модулей, благодаря чему возможна разработка не только на Java, но и на других языках, таких, как C/C++, Python Perl, Ruby, Groovy, PHP, Erlang, компонентного Pascal, Zonnon, и других. Для данной среды разработки существует целый набор коммерческих и свободных модулей. Изначально среда была разработана для языка Java, но в настоящее время существуют многочисленные дополнения для поддержки других языков. Для разработки аналитических BI-приложений, разработки и получения отчётов в Eclipse имеется BIRT Project. Также существуют различные модули для создания графических интерфейсов. В Eclipse встроена функция установки и обновления модулей расширений через сеть Интернет. [46]

Spyder – это среда разработки для Python, которая распространяется свободно. Она является кроссплатформенной и доступна для применения на операционных системах Windows, MacOS и Linux. Название Spyder является аббревиатурой и расшифровывается как Scientific PYthon Development EnviRonment, то есть научная среда разработки для языка Python. Среда разработки создавалась для проведения научных расчетов, и в этой сфере она действительно удобна.

Согласно аннотации разработчиков, Spyder является:

- интерактивной мощной средой разработки для языка программирования Python с продвинутыми возможностями редактирования, интроспекции, интерактивного тестирования и отладки;
- средой численных расчетов благодаря поддержке IPython (улучшенный интерактивный интерпретатор Python) и популярных библиотек Python, таких как SciPy (обработка данных, численные расчеты и др.), NumPy (линейная алгебра) и matplotlib (интерактивная 2D/3D визуализация);
- частью модуля spyderlib для Python, предоставляющего гибкие виджеты на PyQt4, такие как консоль Python (встраиваемая в приложения), редактор кода, редактор списков, кортежей и массивов NumPy.

Рассмотрим некоторые возможности Spyder. Одной из главных особенностей данной среды разработки является возможность удобной работы с переменными. В процессе

выполнения программы они выводятся на панели в виде списка с возможностью просмотра их значений, при этом имеется возможность строить графики по данным массива, что является очень удобной возможностью, используемой при отладке программы и поиске логических ошибок.

Spyder предоставляет гибкие возможности по работе с консолями Python и IPython, благодаря чему можно взаимодействовать с консолями как с отдельными процессами и создавать требуемое их количество. Разрабатываемые программы могут запускаться в существующей или новой консоли.

При запуске программ, использующих для визуализации библиотеку matplotlib появляется возможность редактировать параметры графика, например, параметры осей, легенду, подписи, стили линий с помощью формы, реализуемой библиотекой formlayout. Применение данной особенности позволяет не перезапускать программу чтобы просто изменить параметры оси или провести подобное изменение. С помощью pyflakes в реальном времени код программы проверяется на ошибки, и в случае их нахождения пользователь получает подсказки по их устранению. С помощью средств rope имеются возможности:

- автодополнения;
- перехода к выбранному объекту или функции;
- просмотра кода используемых модулей.

Также в среде Spyder имеется возможность просмотра исходных кодов или документации любых объектов Python (классов, функций, модулей). Также доступна онлайн документация модулей Python, которая сгенерирована в формат html. Для документации в Python используется библиотека Sphinx, которая является удобным инструментом, позволяющим уменьшить время разработки и устранить необходимость в наличии выхода в Интернет для получения информации об особенностях работы библиотечных функций. [47]

Проведем сравнение сред программирования на основе важных возможностей и средств для разработки, отладки и тестирования ПМ ВИЗ, результаты приведены в таблице 2.2.

Таблица 2.2 – Сравнение сред разработки

Параметры	MS Visual Studio	Qt Creator IDE	Eclipse IDE	Spider IDE

Встроенный редактор графического интерфейса	Да	Да	Да	Да
Встроенный отладчик	Да	Да	Да	Да
Переносимость среды на разные платформы	Нет	Да	Да	Да
Установленное ПО на предприятии и наличие лицензии	Нет	Да	Нет	Нет
Опыт работы	Да	Да	Да	Нет
Поддержка разработки QML приложений	Нет	Да	Да	Нет
Стоимость и лицензия распространения	Бесплатная, версия Community	Бесплатная, использование под лицензией LGPL 2.1	Бесплатная, использование под лицензией Eclipse Public License	Бесплатная, использование под лицензией MIT

[44] [45] [46] [47]

В результате сравнения сред разработки и отладки лучшим выбором является Qt Creator IDE. С помощью утилиты qmlscene, входящей в пакет Qt Creator IDE, отладка и тестирование модулей, описывающих пользовательский интерфейс на языке QML, будет проводиться до завершения работы над ПМ ВИЗ в целом, что позволит сразу выявить все ошибки в коде интерфейса пользователя. Вместе с тем модуль интерактивной отладки для языка Python, также встроенный в Qt Creator IDE, позволит тестировать модули логики, что в целом ускорит процесс разработки ПМ. Комплексное применение выбранных технологий позволит оптимизировать производственный процесс и увеличить эффективность проводимых работ.

2.2. Структуры входных и выходных данных

В качестве входных данных должны использоваться файлы специального формата, позволяющего организовывать блочную обработку.

Входные данные представляют собой:

- параметры для обработки поступающих данных, которые должны вводиться оператором с помощью графического интерфейса или конфигурационного файла, имеющего структуру «параметр обработки - значение»;
- файл данных специального формата, включающий размеры снимка, географическую привязку и другие поля данных, необходимые для обработки;
- данные, предназначенные для первичного ознакомления оператора с местностью проведения съемки, полученные из общедоступных источников (например, ГИС или поисково-информационных картографических служб).

Файл данных снимка может записываться в ходе работы ПМ, потому следует разработать алгоритм параллельной обработки данных, представленных в виде информационных блоков.

Выходные данные включают в себя:

- результаты работы модуля обработки данных в виде серии изображений в формате BMP, выводимой на экран оператора на соответствующие географической привязке позиции в экранной форме, а также консольные выводы работы модуля обработки, перенаправляемые в интерфейс ПМ ВИЗ (при использовании различных параметров обработки можно увеличить глубину вывода модуля обработки данных, а также вывести информацию об этапе и прогрессе обработки текущего блока информации);
- сообщения о статусе обработки данных (как отдельного блока, так и процесса в целом) и возникающих событий (например, необходимость изменения входных параметров обработки данных или невозможность применения поступивших параметров при обработке данного блока), выводимые на интерфейс ПМ ВИЗ.

На основе структур данных создадим схему данных, описывающую взаимосвязи всех структур данных на рисунке 2.1 в соответствии с ГОСТ 19.701-90. На ней отображен весь процесс работы с основными информационными структурами. [48]

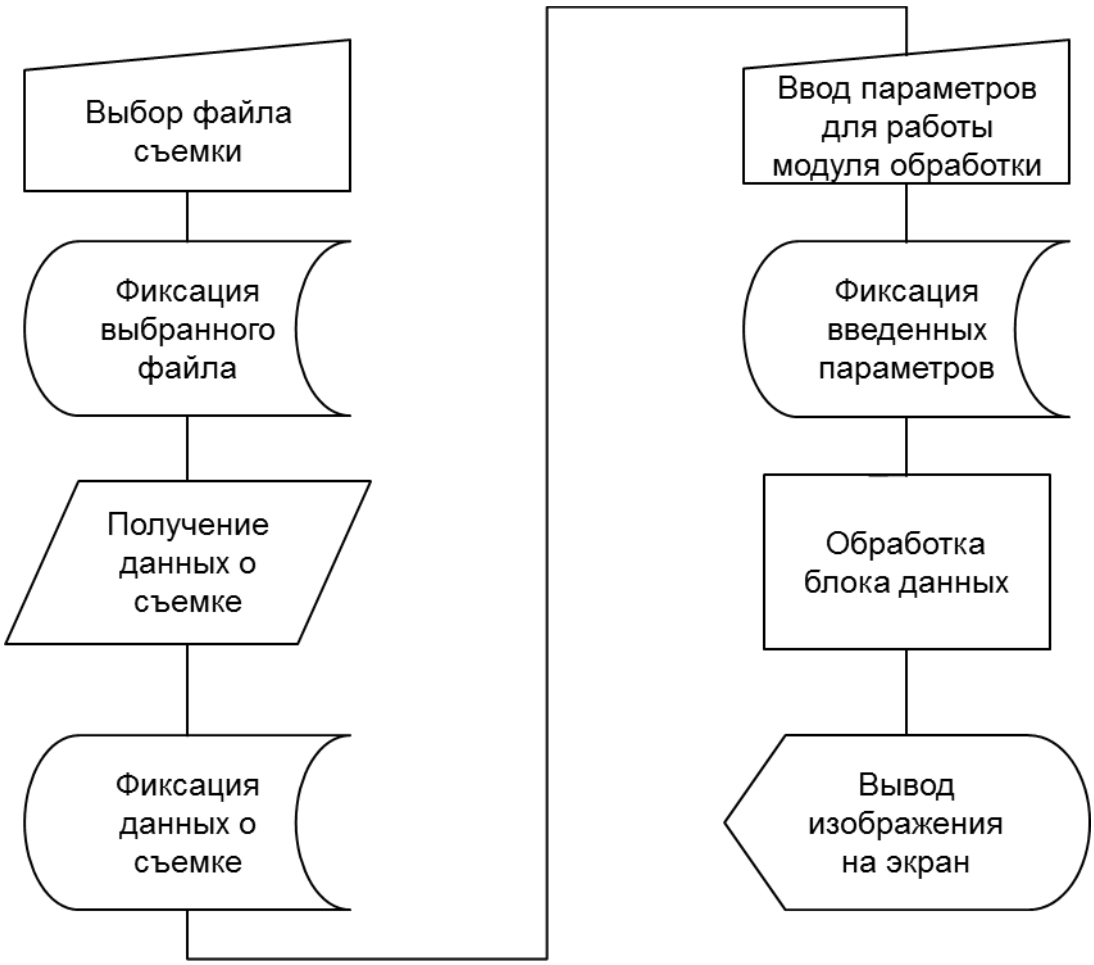

Рисунок 2.1 – Схема данных ПМ ВИЗ

Как видим, сначала производится выбор источника данных съемки, из него читаются поля географической привязки к местности, размеров и прочих атрибутов для дальнейшего вывода карты-подложки и частичного формирования команды конфигурации модуля обработки. После оператор вводит параметры обработки и запускает процесс обработки снимка. Из файла-источника ПМ получает информационный блок, который проверяется на доступность операций чтения/записи и корректность (блок должен быть полностью сформирован в ходе ведения съемки), который потом инкапсулируется в задачу обработки (направляется с параметрами конфигуратора в модуль обработки, откуда возвращается блок изображения). По ее завершении результат выводится на экран для дальнейшей обработки пользователем.

2.3. Разработка алгоритма параллельной обработки

В течение времени файл данных продолжает дописываться, что позволяет организовать блочную обработку поступающей информации. При необходимости

длительной обработки блоков применяется технология параллельного программирования, позволяющая проводить независимую обработку каждого блока и получать результаты быстрее.

Потому основной задачей в ходе разработки ПМ ВИЗ является разработка алгоритма параллельной обработки информационных блоков с применением внешнего модуля обработки. Разработаем схему алгоритма в соответствии с ГОСТ 19.701-90 (см. рис. 3).

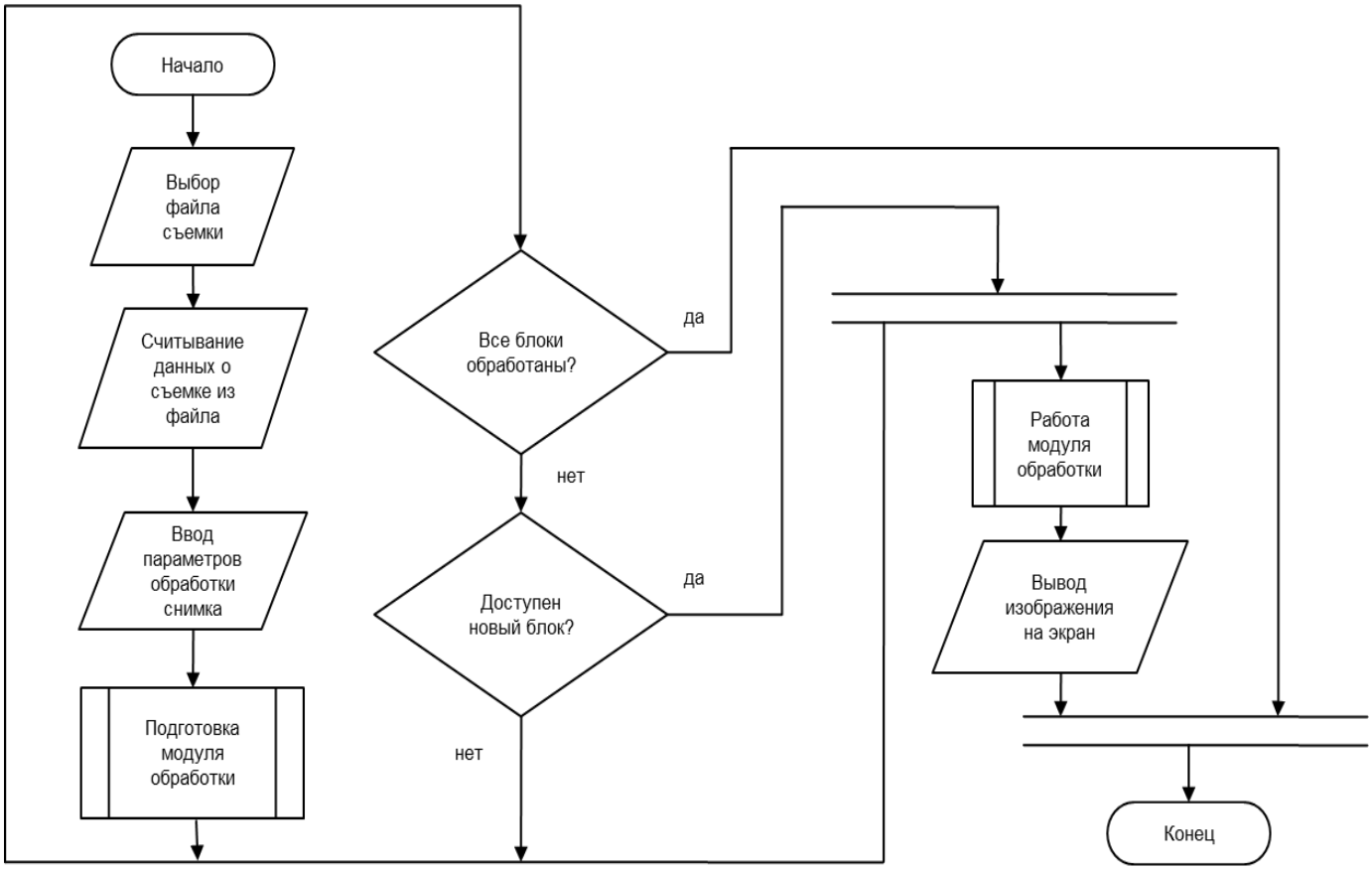

Рисунок 2.2 – Схема основного алгоритма работы ПМ ВИЗ

Как видим из рисунка 2.2, сначала пользователь производит выбор файла съемки, из которого производится чтение данных. На основе геоинформационных данных получаем первичное изображение местности, выводим его на экран. В это время пользователь вводит требуемые параметры обработки, которые подаются с информационным блоком в модуль обработки. ПМ ВИЗ анализирует состояния всех блоков (принимаемые значения – готов, записывается, обрабатывается, обработан). Как только любой блок прекращает записываться и становится готовым для обработки, запускается его параллельная обработка, по завершении которой результирующее изображение выводится на экран. ПМ ВИЗ продолжает свою работу до тех пор, пока не будут обработаны все блоки, а их изображения выведены. Ход работы модуля обработки сопровождается выводами

сопутствующих сообщений на интерфейс пользователя, на котором выводятся и состояния всех блоков. В случае возникновения сбоев в работе модуля обработки ПМ ВИЗ запускает механизм повторной обработки с универсальным набором параметров обработки, в случае если оператор не указал вторичный набор параметров. Если повторная обработка завершается неудачно, то блок в дальнейшей обработке не участвует и пропускается.

2.4. Особенности реализации используемых алгоритмов обработки и вывода данных

В ходе реализации используются следующие паттерны проектирования для реализации программного интерфейса ПМ ВИЗ:

- мост – для организации связи с удаленными сервисами карт описываются похожие модули, имеющие незначительные различия лишь в специфичных случаях;
- фасад – для взаимодействия с модулем-обработчиком данных и передачи параметров обработки, введенных пользователем.

Также для создания процесса обработки отдельного блока используется паттерн Абстрактная фабрика.

Обозначим ряд свойств разработанного решения:

- возможность быстрой замены модуля обработки данных, для чего достаточно заменить модуль интерфейса с целью обеспечения корректной передачи параметров;
- возможность выбора карты-подложки из множества общедоступных сервисов: внедрение новых сводится к созданию нового модуля-моста, имеющего идентичную с аналогичными модулями логику.

Так как для пользовательского интерфейса нет специальных требований, то для его создания используется язык QML, который на базе механизма сигналов и слотов обеспечивает работу для модулей, реализованных на языке Python, в которых находится основная логика. Через интерфейс программы оператор выбирает файл данных, полученных во время съемки, и определяет параметры для последующей обработки.

Основной функционал ПМ ВИЗ расположен во вкладках меню, расположенного на верхней части пользовательского интерфейса. С помощью мыши и специальных клавиш пользователь может менять ориентацию изображения и его масштаб. В строке состояния выводится информация о состояниях всех блоков. Прогресс обработки каждого можно

увидеть, воспользовавшись инструментом отслеживания работы параллельной обработки, где выводятся подробные сообщения из каждого потока обработки. В случае возникновения ошибки обработки конкретного блока оператор может ввести иные параметры обработки, иначе будут применены универсальные. В случае повторной ошибки на интерфейсе пользователя будет выведено сообщение об исключении блока из списка требуемых к обработке.

Выводы к конструкторскому разделу

1. Проведено сравнение языков и технологий программирования, выбраны Python3 и QML 2.0 при использовании библиотеки PyQt5, метода параллельного программирования и механизма слотов и сигналов;
2. Проведено сравнение сред разработки, выбраны среда Qt Creator IDE и утилита qmlscene;
3. Описаны структуры входных и выходных данных, уточнены работа и взаимодействие сущностей;
4. Разработан алгоритм параллельной обработки блоков информации;
5. Уточнены детали реализации используемых алгоритмов обработки и вывода данных.

3. Технологический раздел

3.1. Описание применявшихся средств отладки программы

В ходе разработки ПМ ВИЗ применялись самые различные средства отладки.

Ведение журнала возникающих ошибок с целью дальнейшей их отладки. В языке Python присутствует удобная библиотека гибкого логгирования событий. Данная библиотека представляет возможности перенаправления вывода в требуемое место назначения, фильтрации сообщений по уровням логгирования и форматирование записей для представления в конкретном формате. [49]

Использование механизма исключений и их перехвата для разрешения критических ситуаций работы модуля. Необходимо учитывать тот факт, что во время исполнения ПМ ВИЗ могут параллельно работать несколько потоков обработки данных, потому необходимо корректно отработать возникающие исключения, чтобы сохранить работоспособность программы. При возникновении исключений в ходе работы модуля обработки, описанного на языках C/C++, его тоже требуется принять и обработать, что возможно благодаря возможностям Python. [50] [51] [52]

Использование pdb (интерактивной среды отладки кода, написанного на языке Python) с целью просмотра и изменения значения переменных, осуществления навигации по стеку вызовов, добавления и изменения точек останова в коде, управления ходом выполнения программы. Интерактивная среда способна продолжать отладку даже после возникновения исключения, приводящего к сбою программы, и может управляться программным кодом. [53]

Использование утилиты qmlscene для первичной отладки пользовательского интерфейса до объединения с кодом логики, описанным на Python. Таким образом можно создать пользовательский интерфейс любой сложности, используя для отладки специальные «заглушки». [54]

Использование интерактивной среды отладки, включенной в состав Qt Creator IDE, для осуществления комплексной отладки ПМ. Среда разработки имеет графический интерфейс для следующих отладчиков: GDB, CDB и QML/JavaScript. Среда позволяет осуществлять построчный или поинструкционный переходы, прерывать исполняющиеся программы, устанавливать различные точки останова многими способами и изучать содержимое стека вызовов, значения локальных и глобальных переменных, регистров,

потоков исполнения. В то же время есть возможность расширения среды отладки с помощью собственных отладчиков и помощников отладки. В качестве отдельной опции реализовано отображение содержимого контейнеров, таких как QString, std::map и прочих.

Поддерживаются следующие режимы отладки:

- терминал для отладки локально запущенных процессов, для управления которым требуется консоль;
- простой для отладки локально запущенных приложений, таких как GUI приложения на Qt;
- удалённый для отладки запущенных на другой машине процессов (используя gdbserver);
- подключённый для отладки локальных процессов, запущенных вне Qt Creator;
- ядро для отладки завершившихся аварийно процессов на Unix;
- TRK для отладки процессов, запущенных на устройстве Symbian;
- отладка возникающих исключений, приводящих к сбою программы на Windows. [55]

Также для упрощения и уменьшения необходимости отладки применялись различные паттерны проектирования, соглашения об именовании и прозрачное поведение отдельных блоков кода — чтобы объявить себе и другим, каким образом должна вести себя та или иная функция. Широкое использование проверенных внешних библиотек позволяет уменьшить вероятность возникновения ошибок в ходе разработки.

3.2. Анализ методов и средств тестирования

Тестирование можно рассматривать, как процесс семантической отладки (проверки) программы, заключающийся в исполнении последовательности различных наборов контрольных тестов, для которых заранее известен результат. Т.е. тестирование предполагает выполнение программы и получение конкретных результатов выполнения тестов.

Тесты подбираются так, чтобы они охватывали как можно больше типов ситуаций алгоритма программы. Менее жесткое требование - выполнение хотя бы один раз каждой ветви программы.

Целью тестирования является проверка работы реализованных функций в соответствии с их спецификацией. На основе внешних спецификаций функций и

проектной информации на процессах жизненного цикла создаются функциональные тесты, с помощью которых проводится тестирование с учетом требований, сформулированных на этапе анализа предметной области. Методы функционального тестирования подразделяются на статические и динамические.

Статические методы используются при проведении инспекций и рассмотрении спецификаций компонентов без их выполнения. Техника статического анализа заключается в методическом просмотре (или обзоре) и анализе структуры программ, а также в доказательстве их правильности. Статический анализ направлен на анализ документов, разработанных на всех этапах жизненного цикла, и заключается в инспекции исходного кода и сквозного контроля программы. [56]

Инспекция ПО - это статическая проверка соответствия программы заданным спецификациям, которая проводится путем анализа различных представлений результатов проектирования (документации, требований, спецификаций, схем или исходного кода программ) на процессах жизненного цикла. Просмотры и инспекции результатов проектирования и соответствия их требованиям заказчика обеспечивают более высокое качество создаваемых ПМ.

При инспекции программ рассматриваются документы рабочего проектирования на этапах жизненного цикла совместно с независимыми экспертами и участниками разработки ПМ. На начальном этапе проектирования инспекция предполагает проверку полноты, целостности, однозначности, непротиворечивости и совместимости документов с исходными требованиями к программной системе. На этапе реализации системы под инспекцией понимается анализ текстов программ на соблюдение требований стандартов и принятых руководящих документов технологии программирования.

Эффективность такой проверки заключается в том, что привлекаемые эксперты пытаются взглянуть на проблему «со стороны» и подвергают ее всестороннему критическому анализу.

Эти приемы позволяют на более ранних этапах проектирования обнаружить ошибки или дефекты путем многократного просмотра исходных кодов. Символьное тестирование применяется для проверки отдельных участков программы на входных символьных значениях.

Кроме того, разрабатывается множество новых способов автоматизации символьного выполнения программ. Например, автоматизированное средство статического контроля для

языков ориентированной разработки, инструменты автоматизации доказательства корректности и автоматизированный аппарат сетей Петри.

Динамические методы тестирования используются в процессе выполнения программ. Они базируются на графе, связывающем причины ошибок с ожидаемыми реакциями на эти ошибки. В процессе тестирования накапливается информация об ошибках, которая используется при оценке надежности и качества ПМ.

Динамическое тестирование ориентировано на проверку корректности ПМ на множестве тестов, прогоняемых по ПМ, в целях проверки и сбора данных на этапах жизненного цикла и проведения измерения отдельных показателей (число отказов, сбоев) тестирования для оценки характеристик качества, указанных в требованиях, посредством выполнения системы на ЭВМ. Тестирование основывается на систематических, статистических, (вероятностных) и имитационных методах. Дадим краткую их характеристику. [57]

Систематические методы тестирования делятся на методы, в которых программы рассматриваются как «черный ящик» (используется информация о решаемой задаче), и методы, в которых программа рассматривается как «белый ящик» (используется структура программы). Этот вид называют тестированием с управлением по данным или управлением по входу-выходу. Цель - выяснение обстоятельств, при которых поведение программы не соответствует ее спецификации. При этом количество обнаруженных ошибок в программе является критерием качества входного тестирования.

Цель динамического тестирования программ по принципу «черного ящика» - выявление одним тестом максимального числа ошибок с использованием небольшого подмножества возможных входных данных.

Рассматривать метод «черного ящика» более детально не имеет никакого практического смысла, так как весь исходный код программы находится в доступе разработчика. Достаточно будет заметить, что методы тестирования по принципу «черного ящика» используются для тестирования функций, реализованных в программе, путем проверки несоответствия между реальным поведением функций и ожидаемым поведением с учетом спецификаций требований. Во время подготовки к этому тестированию строятся таблицы условий, причинно-следственные графы и области разбивки. Кроме того, подготавливаются тестовые наборы, учитывающие параметры и условия среды, которые

влияют на поведение функций. Для каждого условия определяется множество значений и ограничений предикатов, которые тестируются.

Метод «белого ящика» позволяет исследовать внутреннюю структуру программы, причем обнаружение всех ошибок в программе является критерием исчерпывающего тестирования маршрутов потоков (графа) передач управления, среди которых рассматриваются:

- критерий покрытия операторов - набор тестов в совокупности должен обеспечить прохождение каждого оператора не менее одного раза;
- критерий тестирования ветвей (известный как покрытие решений или покрытие переходов) - набор тестов в совокупности должен обеспечить прохождение каждой ветви и выхода, по крайней мере, один раз.

Второй рассмотренный критерий соответствует простому структурному тесту и наиболее распространен на практике. Для удовлетворения этого критерия необходимо построить систему путей, содержащую все ветви программы. Нахождение такого оптимального покрытия в некоторых случаях осуществляется просто, а в других является более сложной задачей.

Тестирование по принципу «белого ящика» ориентировано на проверку прохождения всех путей программ посредством применения путевого и имитационного тестирования.

Путевое тестирование применяется на уровне модулей и графовой модели программы путем выбора тестовых ситуаций, подготовки данных и включает тестирование следующих элементов:

- операторов, которые должны быть выполнены хотя бы один раз, без учета ошибок, которые могут остаться в программе из-за большого количества логических путей и необходимости прохождения подмножеств этих путей;
- путей по заданному графу потоков управления для выявления разных маршрутов передачи управления с помощью путевых предикатов, для вычисления которого создается набор тестовых данных, гарантирующих прохождение всех путей. Однако все пути протестировать бывает невозможно, поэтому остаются не выявленные ошибки, которые могут проявиться в процессе эксплуатации;
- блоков, разделяющих программы на отдельные части блоки, которые выполняются один раз или многократно при нахождении путей в программе, включающих совокупность

блоков реализации одной функции либо нахождения входного множества данных, которое будет использоваться для выполнения указанного пути.

«Белый ящик» базируется на структуре программы, в случае «черного ящика», о структуре программы ничего неизвестно. Для выполнения тестирования с помощью этих «ящиков» известными считаются выполняемые функции, входы (входные данные) и выходы (выходные данные), а также логика обработки, представленные в документации.

Тестирование программ может проводиться на разных уровнях и этапах готовности:

- тестирование модуля, юнит-тестирование – изолированная проверка отдельного модуля программы;

- интеграционное тестирование программ – проверка взаимодействия между модулями программы и отдельными подсистемами. Подразумевает постепенное вовлечение модулей в тестирование программы;

- комплексное, системное тестирование – проверка соответствия и работоспособности всей системы. Комплексное тестирование программ может проводиться как в изолированной, так и в реальной среде;

- альфа-тестирование – тестирование программ и информационных продуктов при полной имитации реальной среды. Процесс выполняется разработчиками или реальными потенциальными пользователями, привлеченными разработчиком;

- бета-тестирование – пробная работа программы, распространенной разным пользователям. Зачастую предлагается урезанная бета-версия, ограниченная по функциональности или же срокам использования.

Модульное тестирование — это процесс тестирования отдельных блоков, подпрограмм, классов или процедур, образующих крупную программу. Другими словами, прежде чем тестировать программу в целом, необходимо сосредоточить внимание на ее меньших по размеру компонентах. Так как модульное тестирование является одним из самых популярных и простых методов тестирования разрабатываемых систем, то остановимся на деталях. Опишем три возникающие причины применения модульного тестирования. Во-первых, при таком подходе повышается эффективность тестирования сложных объектов, поскольку на первом этапе внимание фокусируется на небольших программных блоках, которые легче тестировать. Во-вторых, при таком подходе облегчается отладка программ (точная локализация и исправление обнаруженной ошибки), поскольку, обнаружив ошибку, мы всегда точно знаем, в каком именно модуле она

содержится. Наконец, модульное тестирование позволяет распараллеливать процесс тестирования, что обеспечивает возможность одновременного тестирования нескольких модулей. Цель модульного тестирования — сравнение функций, реализуемых модулем, со спецификациями, описывающими его функциональные или интерфейсные характеристики. Вновь подчеркнем, что под этим подразумевается не доказательство соответствия модуля требованиям спецификации, а демонстрация того, что поведение модуля отличается от того, которое специфицировано.

При проектировании модульных тестов используются два источника информации: спецификация модуля и его исходный код. Типичная спецификация описывает назначение модуля, а также его входные и выходные параметры.

Модульное тестирование в основном ориентировано на использование метода «белого ящика». Объясняется это прежде всего тем, что при последующем переходе к тестированию более крупных программных единиц, например, программ в целом, применимость метода «белого ящика» снижается. Кроме того, последующие этапы процесса тестирования ориентированы на обнаружение ошибок другого типа (не обязательно связанных с программной логикой, а обусловленных, например, несоответствием программы ожиданиям пользователей). [58]

Данный программный модуль разрабатывается достаточно быстро и легко, потому создание кейс-тестов для него является задачей достаточно тривиальной. Опишем ниже кейс-тесты и причины необходимости их применения. Нас будут интересовать модульное и интеграционное тестирования.

В ходе проведения тестирования программного модуля необходимо выявить удовлетворение таким качественно-количественным параметрам, требуемым в результате постановки задачи о разработке ПМ, как:

- функциональность, т.е. обеспечивает соответствие функций тестируемой программы требованиям пользователя;
- эффективность, характеризует баланс уровня качества программы и загруженности системных ресурсов на её выполнение в заданных рамках;
- мобильность, обеспечивает свойство программы быть перенесенной в другую среду реализации;
- надежность, обеспечивает свойство программы сохранять уровень качества работы в заданных условиях;

- сопровождаемость, определяет поддержку функционирования, простоту и удобство внесения изменений;
- практичность, т.е. применимость программы, с точки зрения удобства и привлекательности для пользователя. [59]

Этот набор критериев и определяет границы тестирования разрабатываемого ПМ. При модульном тестировании функциональность программного обеспечения сравнивается со спецификациями, описывающими назначение функций. Модульное, или блочное, тестирование может служить важной частью инструментария разработчика, позволяя создавать надежные приложения, особенно при работе с такими объектно-ориентированными языками, как Java и C#. Перед модульным тестированием стоит та же цель, что и перед любым другим типом тестирования программного обеспечения: демонстрация несоответствия программы требованиям ее спецификации. Для выполнения модульного теста требуется не только спецификация, но и исходный код каждого модуля. В значительной степени модульное тестирование относится к стратегии “белого ящика”. Тщательное выполнение модульного тестирования предполагает использование инкрементных стратегий, таких как нисходящее или восходящее тестирование. Прежде чем приступить к модульному тестированию, целесообразно исследовать материал, посвященный психологическим и экономическим аспектам тестирования для создания более полноценных тестов. [60]

3.3. Составление кейс-тестов

Так как разрабатываемый ПМ своей целью ставит ускорение процесса обработки данных и вывод результатов на экран как можно скорее, то необходимо прежде убедиться в безошибочной работе пользовательского интерфейса и его базового функционала по работе с изображениями – вращение и изменение масштаба. Потому требуется составить модульный кейс-тест, который будет включать в себя открытие обработанных изображений, корректное их отображение на пользовательском интерфейсе, последовательное вращение итогового изображения на 90 градусов вправо и увеличение масштаба в 2 раза. Таким образом, будет проведено тестирование корректности расчетов отступа, вращения и размера выводимых изображений.

Запись последовательности проводимых шагов будет выглядеть так:

1. Загрузка изображения на пользовательский интерфейс;
2. Вращение изображения на 90 градусов направо;
3. Сравнение полученного результата с исходным и поиск ошибки отображения. Если подобной нет, то продолжение тестирования;
4. Увеличение масштаба изображения в 1.5 раза;
5. Сравнение полученного результата с исходным и поиск ошибки отображения. Если подобной нет, то завершение тестирования.

Для работы в целом должна быть разработана контрольно-демонстрационная задача. При этом проверяется выполнение всех функций программы. В качестве такой задачи может выступить интеграционное кейс-тестирование взаимодействия пользовательского интерфейса, программной логики и модуля обработки. В ходе этого теста будет происходить выбор файла съемки, ввод параметров обработки, запуск параллельных задач по обработке, вывод сопутствующих сообщений из модуля обработки и результирующего изображения на пользовательский интерфейс, а также обработка критических ситуаций. Данный кейс-тест должен покрывать значительную часть ПМ и проверять не только корректность работы отдельных функций, но и всего модуля в целом.

Запись последовательности проводимых шагов будет выглядеть так:

6. Выбор файла съемки через пользовательский интерфейс;
7. Ввод пользовательских параметров обработки данных через форму диалога;
8. Запуск процесса параллельной постановки информационных блоков на обработку, ожидание завершения первого процесса, возвращающего в интерфейс пользователя сопутствующие сообщения;
9. Проверка полученного результата – блок-изображение №1 должен корректно вывестись на необходимую позицию;
10. Продолжение получения результатов обработки всех оставшихся блоков и проверка их вывода на экран;
11. По завершению работы модуля обработки просмотр полученного итогового изображения на предмет соответствия действительному тестовому значению. Если ошибки нет, то завершение тестирования.

3.4. Процесс и результаты тестирования

Так как помимо известных вариантов применения ПМ (а именно: ввод данных, их параллельная обработка с выводом состояния и результата-изображения) другие функциональные возможности в ПМ ВИЗ не предусмотрены (вывод карты-подложки и ее трансформация являются подвидом работы с изображением на экране), то описанные ранее два кейс-теста обеспечивают оптимальное покрытие кода ПМ.

3.4.1. Процесс модульного тестирования ПМ ВИЗ

Тест: изменение масштаба изображения в 1.5 раза и его вращение на 90° вправо.

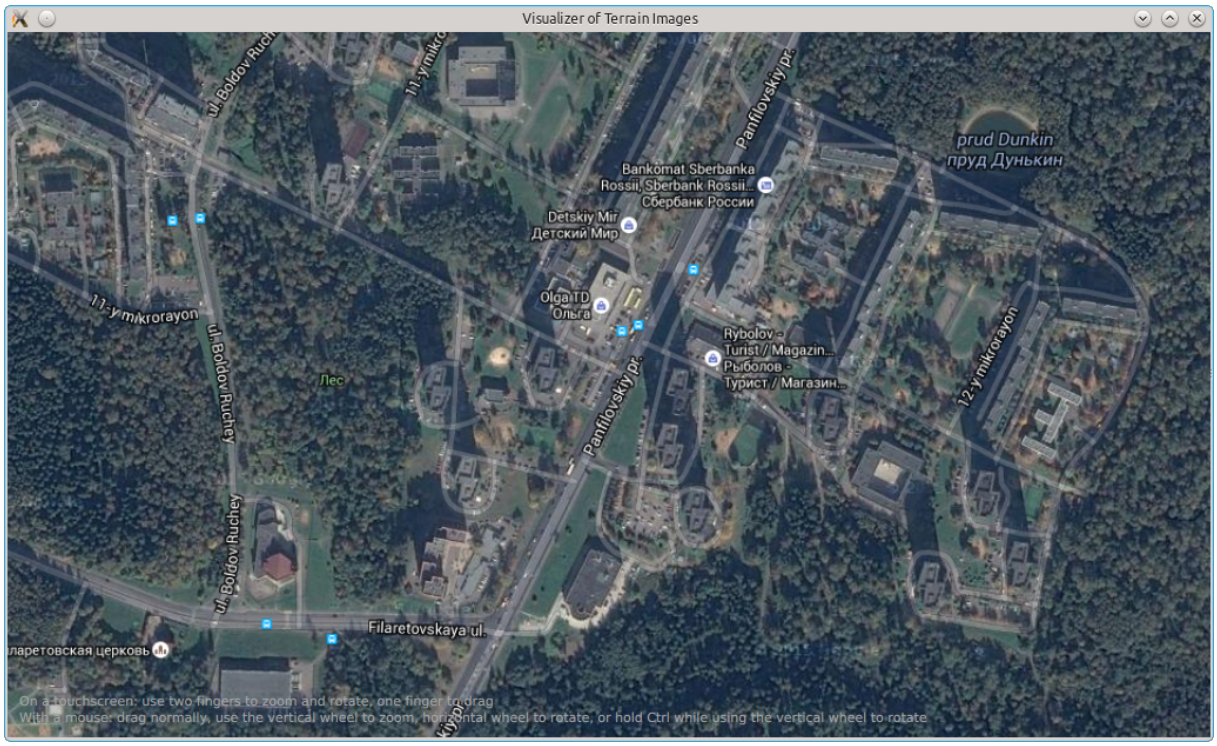

Рисунок 3.1 – Вид пользовательского интерфейса до проведения модульного тестирования

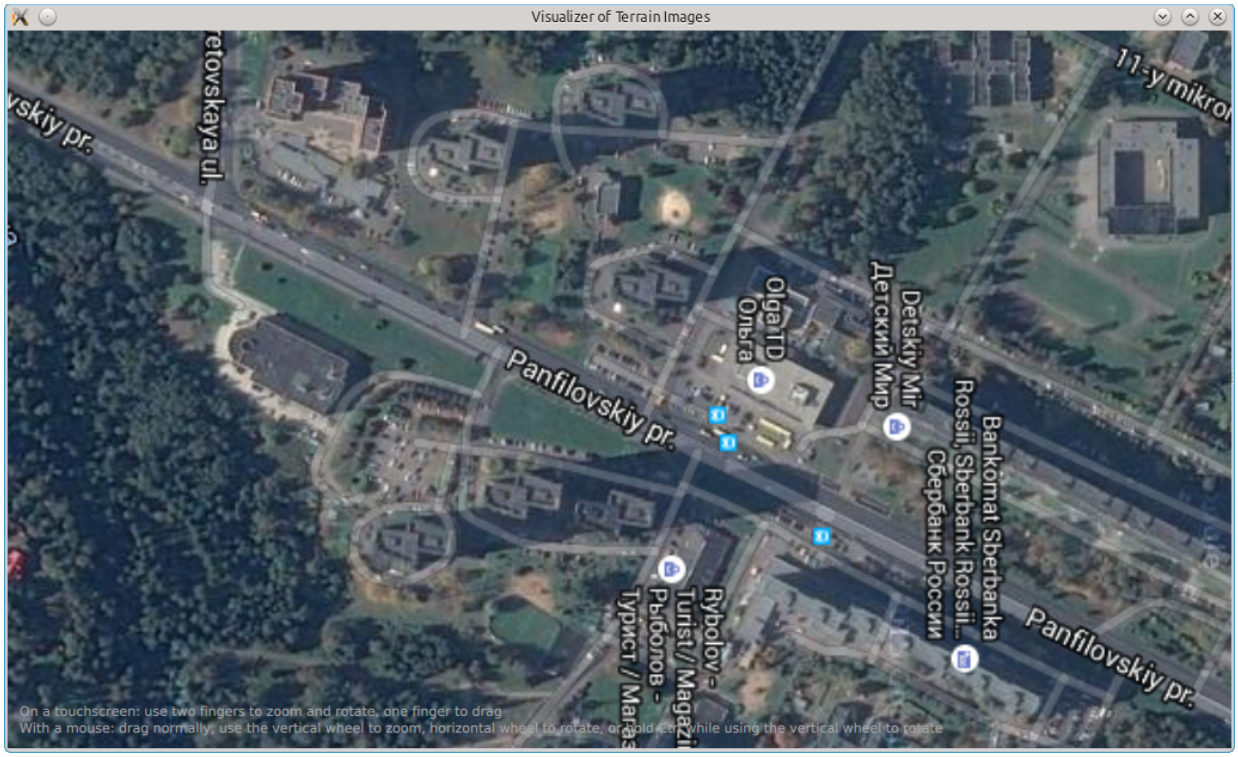

Рисунок 3.2 – Вид пользовательского интерфейса после проведения unit-тестирования

3.4.2. Процесс интеграционного тестирования ПМ ВИЗ

Тест: выбор файла съемки и вывод на интерфейс результатов работы модуля обработки.

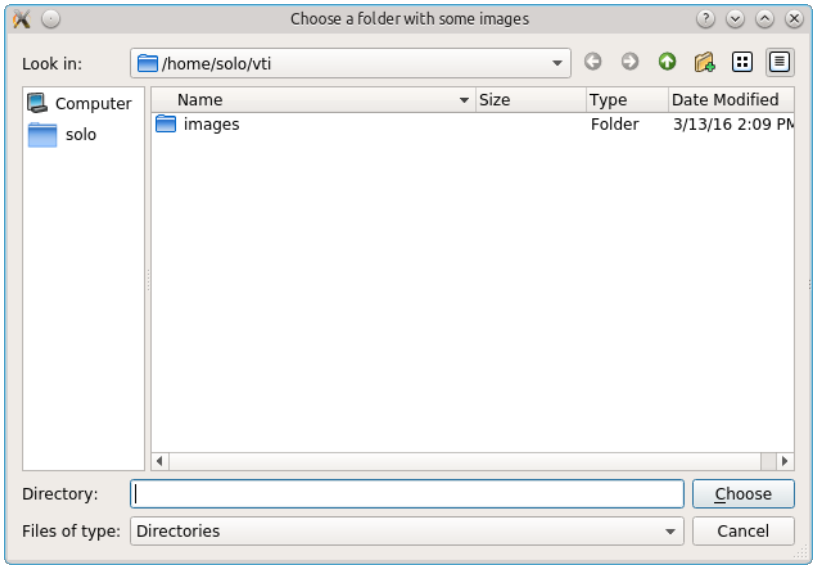

Рисунок 3.3 – Выбор файла съемки для начала работы ПМ ВИЗ

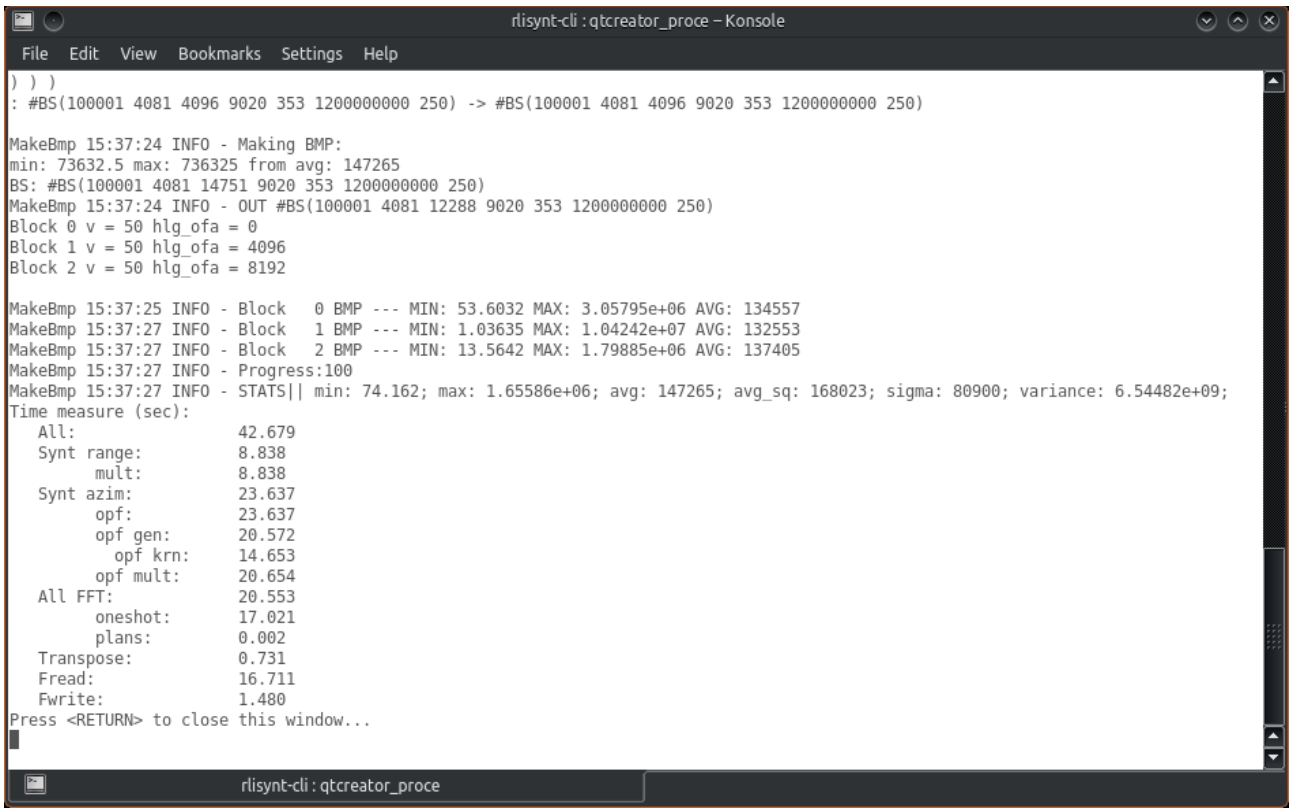

Рисунок 3.4 – Вывод сообщений из модуля обработки на пользовательский интерфейс

Рисунок 3.5 – Вид пользовательского интерфейса после завершения обработки файла съемки

3.4.3. Результаты тестирования ПМ ВИЗ

В ходе тестирования разрабатываемого модуля были выявлены незначительные отклонения в формулах вычисления размеров и отступов блоков изображений, они либо частично накладывались друг на друга, либо не состыковывались краями. В результате тестирования были скорректированы соответствующие формулы и получены верные результаты, которые отражены в рисунках 3.1-3.5.

Выводы к технологическому разделу

1) Описаны применявшиеся средства отладки программы, помимо средств используемых сред и утилит используется утилита интерактивной отладки модулей на языке Python pdb;
2) Проведен анализ методов и средств тестирования, выбраны методы «белого ящика», модульного и интеграционного тестирований;
3) Составлены кейс-тесты, обеспечивающие оптимальное покрытие программного кода и основного функционала ПМ ВИЗ;
4) Описан процесс обработки возникающих ошибок;
5) Проведены модульное и интеграционное тестирования, результаты которых проанализированы и учтены в ходе разработки программного модуля.
6)

Заключение

Разработанный программный модуль обладает рядом свойств:

- при необходимости длительной обработки блоков применяется технология параллельного программирования, позволяющая проводить независимую обработку каждого блока и получать результаты быстрее;
- возможность быстрой замены модуля обработки данных, для чего достаточно заменить модуль интерфейса с целью обеспечения корректной передачи параметров;
- возможность выбора карты-подложки из множества общедоступных сервисов: внедрение новых сводится к созданию нового модуля-моста, имеющего идентичную с аналогичными модулями логику;
- кроссплатформенность: модуль обладает способностью одинакового функционирования вне зависимости от операционной системы.

В результате разработанный программный модуль осуществляет блочную обработку потока поступающей информации параллельно съемке, в связи с чем повышена скорость и уменьшено время обработки поступающих данных, что обеспечивает высокую актуальность получаемых изображений.

Работа оператора упрощена, так как карта-подложка как способ получения первичных сведений ускоряет процесс изучения местности проведения съемки, и в дальнейшем она заменяется снимками высокого качества.

Модуль обработки данных легко заменяется для применения в различных предметных областях (обработка цифровой фотографии, теплового снимка, геодезических данных или иных).

С помощью разработанного модуля можно быстро выполнять самые различные задачи, как поиск объекта и спасение жизни.

Список литературы

1. Гагарина, Л.Г. Методические указания по подготовке выпускной квалификационной работы по направлению подготовки бакалавров 09.03.04 «Программная инженерия» [Текст] / Л.Г. Гагарина, Р.А. Касимов, Е.Л. Федотова, Б.В. Черников, Д.Г. Коваленко, 3о Е Чжо; ред. Б.В. Черников. – М.: МИЭТ, 2016. – 20 с.

2. Официальный сайт промышленного холдинга «ТСК» // Что такое GNSS? [Электронный ресурс]. – ООО «МНПО Спектр», cop. 2008–2016. – Режим доступа: http://www.mnpo-spektr.ru/articles/gnss.php [дата обращения: 14.05.2016]

3. Свободная интернет-энциклопедия Wikipedia // Rich Client Platform [Электронный ресурс]. – НКО «Фонд Викимедиа», cop. 2003–2016. – Режим доступа: https://ru.wikipedia.org/wiki/Rich_Client_Platform [дата обращения: 01.05.2016]

4. Издание о бизнесе и технологиях EquipNet.ru // Беспилотники: воздушные роботы мирного назначения [Электронный ресурс]. – ООО «Гипер-Пресс», cop. 2000-2016. – Режим доступа: http://www.equipnet.ru/articles/hi-tech/hi-tech_1478.html [дата обращения: 01.04.2016]

5. Портал журнала «Наука и жизнь» // Беспилотные самолеты: Максимум возможностей [Электронный ресурс]. – АНО Редакция журнала «Наука и жизнь», cop. 2005–2016. – Режим доступа: http://www.nkj.ru/archive/articles/4323/ [дата обращения: 01.04.2016]

6. Официальный сайт ООО «РУСГЕОКОМ» // БПЛА Supercam S350f [Электронный ресурс]. – ООО «РУСГЕОКОМ», cop. 2010–2016. – Режим доступа: http://www.rusgeocom.ru/bpla-supercam-s350f [дата обращения: 13.04.2016]

7. Официальный сайт ООО «ГЕОСалют» // Геодезический БПЛА Supercam S350f [Электронный ресурс]. – ООО «ГЕОСалют», cop. 2012–2016. – Режим доступа: http://www.geosalut.ru/bpla/bpla-s350-f/ [дата обращения: 13.04.2016]

8. The Open Source Geospatial Foundation // gvSIG info sheet [Электронный ресурс]. – gvSIG Association, cop. 2009–2016. – Режим доступа: http://www.osgeo.org/gvsig [дата обращения: 01.04.2016]

9. Официальный сайт GRASS GIS // GRASS GIS - Home [Электронный ресурс]. – GRASS Development Team, cop. 1998–2016. – Режим доступа: http://grass.osgeo.org/ [дата обращения: 01.04.2016]

10. Официальный сайт QGIS // Обзор QGIS [Электронный ресурс]. – QGIS Development Team, cop. 2002–2016. – Режим доступа: http://qgis.org/ru/site/about/index.html [дата обращения: 01.04.2016]

11. Свободная интернет-энциклопедия Wikipedia // QGIS [Электронный ресурс]. – НКО «Фонд Викимедиа», cop. 2003–2016. – Режим доступа: https://ru.wikipedia.org/wiki/QGIS [дата обращения: 05.04.2016]

12. Журнал «Системы управления базами данных» // Модель «сущность-связь» - шаг к единому представлению о данных [Электронный ресурс]. – Изд-во «Открытые системы», cop. 1992–2016. – Режим доступа: http://www.osp.ru/dbms/1995/03/13031431/ [дата обращения: 15.04.2016]

13. Свободная интернет-энциклопедия Wikipedia // Си (язык программирования) [Электронный ресурс]. – НКО «Фонд Викимедиа», cop. 2003–2016. – Режим доступа: https://en.wikipedia.org/wiki/C_(programming_language) [дата обращения: 18.04.2016]

14. Керниган, Б. Язык программирования Си [Текст] / Б. Керниган, Д. Ритчи; пер. с анг. и ред. В. Бродовой. – М.: Вильямс, 2007. – 304 с.

15. Справка по C++ // История C++ [Электронный ресурс]. – Электронная библиотека «Cppreference.com», cop. 2010–2016. – Режим доступа: http://ru.cppreference.com/w/cpp/language/history [дата обращения: 18.04.2016]

16. Страуструп, Б. Язык программирования C++. Специальное издание [Текст] / Б. Страуструп; зав. ред. Н. Н. Мартынов. – М.: Бином, 2011. – 1136 с.

17. Страуструп, Б. Программирование: принципы и практика использования C++ [Текст] / Б. Страуструп; пер. и ред. Д. Клюшин. – М.: Вильямс, 2011. – 1248 с.

18. Сообщество языка Python // О языке Python [Электронный ресурс]. – Python Software Foundation, cop. 2001–2016. – Режим доступа: https://www.python.org/about/ [дата обращения: 18.04.2016]

19. Лутц, М. Программирование на Python [Текст]: в 2-х т. / М. Лутц; пер. с анг. А. Киселевой. – 4-е изд. – СПб.: Символ-Плюс, 2011. – 2 т.

20. Лутц, М. Изучаем Python [Текст] / М. Лутц; пер. с анг. А. Киселевой, глав. ред. А. Галунов, зав. ред. Н. Макарова, вып. ред. П. Щеголев, ред. Ю. Бочина, корр. С. Минин, вер. К. Чубаров. – 4-е изд. – СПб.: Символ-Плюс, 2011. – 1280 с.

21. Qt Documentation // QML Applications [Электронный ресурс]. – The Qt Company, cop. 1994–2016. – Режим доступа: http://doc.qt.io/qt-5/qmlapplications.html [дата обращения: 18.04.2016]

22. Java и вы // Подробнее о технологии Java [Электронный ресурс]. – Oracle Corporation, cop. 2010–2016. – Режим доступа: http://www.java.com/ru/about/ [дата обращения: 19.04.2016]

23. Вязовик, Н.А. Программирование на Java [Текст] / Н.А. Вязовик;. – 2-е изд. – М.: Интуит, 2016. – 600 с.

24. Гослинг, Д. Язык программирования Java SE 8. Подробное описание [Текст] / Д. Гослинг, Б. Джой, Г.Л. Стил, Г. Брача, А. Бакли; пер. с анг. И. Карася, зав. ред. С.Н. Тригуб, лит. ред. Л.Н. Красножон, худ. ред. В.Г. Павлютин, корр. Л.А. Гордиенко, вер. М.А. Удалов. – 5-е изд. – М.: Вильямс, 2015. – 672 с.

25. Йенер, М. Java EE. Паттерны проектирования для профессионалов [Текст] / М. Йенер, А. Фидом; тех. ред. М. Санаулла. – СПб.: Питер, 2016. – 240 с.

26. Флэнаган, Д. Язык программирования Ruby [Текст] / Д. Флэнаган, Ю. Мацумото; пер. с анг. Н. Вильчинского, зав. ред А. Кривцов, рук. А. Юрченко, вед. ред. Ю. Сергиенко, лит. ред. П. Маннинен, худ. ред. Л. Адуевская, корр. В. Нечаева, вер. Л. Родионова. – СПб.: Питер, 2011. – 492 с. – (Серия «Бестселлеры O’Reilly»).

27. Руби, С. Гибкая разработка веб-приложений в среде Rails [Текст] / С. Руби, Д. Томас, Д. Хэнссон; пер. с анг. Н. Вильчинского. – 4-е изд. – СПб.: Питер, 2012. – 464 с.

28. Эдельсон, Д. Ruby на платформе Java [Текст] / Д. Эдельсон, Г. Лю; пер. с анг. А. Слинкина. – М.: ДМК Пресс, 2011. – 240 с. – (Серия «Профессиональная серия для программистов»).

29. Стиллмен, Э. Изучаем C# [Текст] / Э. Стиллмен, Дж. Грин; пер. с анг. И. Рузмайкина, зав. ред. П. Щеголев, рук. пр. А. Юрченко, вед. ред. Ю. Сергиенко, корр. Н. Викторова, вер. Н. Лукьянова. – 3-е изд. – СПб.: Питер, 2014. – 816 с. – (Серия «Head First O’Reilly»).

30. Шилдт, Г. C# 4.0: полное руководство [Текст] / Г. Шилдт; пер. с анг. и ред. И.В. Берштейна, зав. ред. С.Н. Тригуб. – М.: Вильямс, 2013. – 1056 с.

31. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C# [Текст] / Дж. Рихтер; пер. с анг. И. Радченко и И. Рузмайкина, зав. ред. А. Кривцов, рук. пр. А. Юрченко, вед. ред. Ю. Сергиенко, лит. ред. А. Жданов, худ.

ред. Л. Адуевская, корр. В. Листова, вер. В. Романов. – 3-е изд. – СПб.: Питер, 2012. – 928 с.

32. Шварц, Р.Л. Изучаем Perl [Текст] / Р.Л. Шварц, Т. Феникс, б. д фой; пер. с анг. Е. Матвеевой, глав. ред. А. Галунов, зав. ред. Н. Макарова, вып. ред. А. Пасечник, ред. А. Петухов, корр. О. Макарова, вер. Д. Орлова. – 5-е изд. – СПб.: Символ-Плюс, 2009. – 384 с.

33. Кристиансен, Т. Программирование на Perl [Текст] / Т. Кристиансен, б. д фой, Л. Уолл, Дж. Орвант; пер. с анг. А. Киселева, глав. ред. А. Галунов, зав. ред. Н. Макарова, науч. ред. М. Зислис, А. Киселев, ред. М. Зислис, корр. Т. Иванкова, В. Логунова, вер. Д. Орлова. – 4-е изд. – СПб.: Символ-Плюс, 2014. – 1048 с. – (Серия «Head First O’Reilly»).

34. Штайн, Л. Разработка сетевых программ на Perl [Текст] / Л. Штайн; пер. с анг. К. Птицына. – М.: Вильямс, 2001. – 752 с.

35. Дэвис, М.Е. Изучаем PHP и MySQL [Текст] / М.Е. Дэвис, Дж.А. Филлипс; пер. с анг. А. Киселева, глав. ред. А. Галунов, вып. ред. Л. Пискунова, ред. Т. Темкина, науч. ред. О. Цилюрик, корр. Е. Бекназарова, вер. Д. Белова. – СПб.: Символ-Плюс, 2008. – 448 с.

36. Бейли, Л. Изучаем PHP и MySQL [Текст] / Л. Бейли, М. Моррисон. – М.: Эксмо, 2010. – 800 с. – (Серия «Head First O’Reilly»).

37. Скляр, Д. PHP. Рецепты программирования [Текст] / Д. Скляр, А. Трахтенберг; пер. с анг. Е. Матвеева. – 3-е изд. – СПб.: Питер, 2015. – 784 с. – (Серия «Бестселлеры O’Reilly»).

38. Python 3.4.4 documentation // Extending Python with C or C++ [Электронный ресурс]. – Python Software Foundation, cop. 2001–2016. – Режим доступа: https://docs.python.org/3.4/extending/extending.html [дата обращения: 01.05.2016]

39. Журнал «Системы управления базами данных» // Эмпирическое сравнение семи языков программирования [Электронный ресурс]. – Изд-во «Открытые системы», cop. 1992–2016. – Режим доступа: http://www.osp.ru/os/2000/12/178361/ [дата обращения: 15.04.2016]

40. Свободная интернет-энциклопедия Wikipedia // Comparison of programming languages [Электронный ресурс]. – НКО «Фонд Викимедиа», cop. 2003–2016. – Режим доступа: https://en.wikipedia.org/wiki/Comparison_of_programming_languages [дата обращения: 21.05.2016]

41. Официальный сайт сообщества IBM developerWorks // Производительность языков программирования. Часть 1 [Электронный ресурс]. – International Business Machines

Corporation, cop. 1888–2016. – Режим доступа: http://www.ibm.com/developerworks/ru/library/ManySpeed_08_1/index.html [дата обращения: 21.05.2016]

42. Прохоренок, Н.А. Python 3 и PyQt. Разработка приложений [Текст] / Н.А. Прохоренок; глав. ред. Е. Кондукова, зам. глав. ред. Е. Рыбаков, зав. ред. Г. Добин, вып. ред. А. Пасечник, ред. Е. Капалыгина, корр. З. Дмитриева, вер. О. Сергиенко, диз. М. Дамбиева, зав. произв. Н. Тверских. – СПб.: БХВ-Петербург, 2012. – 704 с.

43. PyQt 5.5.1 Reference Guide // Integrating Python and QML [Электронный ресурс]. – Riverbank Computing Limited, cop. 2015–2016. – Режим доступа: http://pyqt.sourceforge.net/Docs/PyQt5/qml.html [дата обращения: 18.04.2016]

44. Visual Studio – Microsoft Developer Tools // Overview of Visual Studio 2015 Products [Электронный ресурс]. – Microsoft Corporation, cop. 1975–2016. – Режим доступа: https://www.visualstudio.com/vs-2015-product-editions [дата обращения: 19.04.2016]

45. Qt Wiki // About Qt [Электронный ресурс]. – The Qt Company, cop. 1994–2016. – Режим доступа: http://wiki.qt.io/About_Qt [дата обращения: 19.04.2016]

46. Eclipse Wiki // Eclipse Project [Электронный ресурс]. – The Eclipse Foundation, cop. 2001–2016. – Режим доступа: http://wiki.eclipse.org/Eclipse_Project [дата обращения: 19.04.2016]

47. Spyder IDE Wiki // About Spyder IDE [Электронный ресурс]. – GitHub Inc, cop. 2008–2016. – Режим доступа: https://github.com/spyder-ide/spyder/wiki [дата обращения: 19.04.2016]

48. ГОСТ 19.701 – 901. ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения [Текст]. – Взамен ГОСТ 19.002-80, ГОСТ 19.003-80; Введ. 01.01.92. – М.: Изд-во стандартов, 1990. – 22 с.

49. Python 3.4.4 documentation // Logging facility for Python [Электронный ресурс]. – Python Software Foundation, cop. 2001–2016. – Режим доступа: https://docs.python.org/3.4/library/logging.html [дата обращения: 01.05.2016]

50. Qt Documentation // Debugging [Электронный ресурс]. – The Qt Company, cop. 1994–2016. – Режим доступа: http://doc.qt.io/qtcreator/creator-debugging.html [дата обращения: 01.05.2016]

51. Python 3.4.4 documentation // Errors and Exceptions [Электронный ресурс]. – Python Software Foundation, cop. 2001–2016. – Режим доступа: https://docs.python.org/3.4/tutorial/errors.html [дата обращения: 01.05.2016]

52. Python 3.4.4 documentation // Exception Handling [Электронный ресурс]. – Python Software Foundation, cop. 2001–2016. – Режим доступа: https://docs.python.org/3.4/c-api/exceptions.html [дата обращения: 01.05.2016]

53. Python 3.4.4 documentation // Tasks and coroutines [Электронный ресурс]. – Python Software Foundation, cop. 2001–2016. – Режим доступа: https://docs.python.org/3.4/library/asyncio-task.html [дата обращения: 01.05.2016]

54. Qt Documentation // Prototyping with qmlscene [Электронный ресурс]. – The Qt Company, cop. 1994–2016. – Режим доступа: http://doc.qt.io/qt-5/qtquick-qmlscene.html [дата обращения: 01.05.2016]

55. Python 3.4.4 documentation // The Python Debugger [Электронный ресурс]. – Python Software Foundation, cop. 2001–2016. – Режим доступа: https://docs.python.org/3.4/library/pdb.html [дата обращения: 01.05.2016]

56. Тамре, Л. Введение в тестирование программного обеспечения [Текст] / Л. Тамре; пер. с анг. В.В. Марченко, зав. ред. А.В. Слепцов. – М.: Вильямс, 2003. – 368 с.

57. Майерс, Г. Искусство тестирования программ [Текст] / Г. Майерс, Т. Баджетт, К. Сандлер; пер. с анг. А. Гузикевича. – М.: Вильямс, 2012. – 272 с.

58. Блэк, Р. Ключевые процессы тестирования. Планирование, подготовка, проведение, совершенствование [Текст] / Р. Блэк; пер. с анг. М. Павлова. – 3-е изд. – М.: Лори, 2011. – 544 с.

59. Макгрегор, Дж. Тестирование объектно-ориентированного программного обеспечения. Практическое пособие [Текст] / Дж. Макгрегор, Д. Сайкс; пер. с анг. о. Здира. – М.: ТИД «ДС», 2002. – 438 с.

60. Канер, С. Тестирование программного обеспечения [Текст] / С. Канер, Дж. Фолк, Е. Нгуен; пер. с анг. о. Здира. – М.: ДиаСофт, 2000. – 544 с.

61. ГОСТ 19.505 – 79. ЕСПД. Руководство оператора. Требования к содержанию и оформлению [Текст]. – Введ. 01.01.80. – М.: Стандартинформ, 2010. – 99 с.

Программный модуль визуализации изображений местности на основе данных беспилотного летательного аппарата (ПМ ВИЗ)
Текст программы

Москва 2016

Программный модуль визуализации изображений местности на основе данных беспилотного летательного аппарата (ПМ ВИЗ)
Руководство оператора

Москва 2016