**«Санкт-Петербургский государственный электротехнический университет**

**«ЛЭТИ» им. В.И.Ульянова (Ленина)»**

**(СПбГЭТУ «ЛЭТИ»)**

|  |  |  |
| --- | --- | --- |
| **Направление** | 09.03.04 – Программная инженерия | |
| **Профиль** | Разработка программно-информационных систем | |
| **Факультет** | КТИ | |
| **Кафедра** | МО ЭВМ | |
| *К защите допустить* |  | |
| Зав. кафедрой |  | А.А. Лисс |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**БАКАЛАВРА**

**Тема: ИССЛЕДОВАНИЕ ВОЗМОЖНОСТЕЙ АВТОМАТИЧЕСКОГО ОБНАРУЖЕНИЯ И УСТРАНЕНИЯ ОШИБОК ТРЕХМЕРНОЙ МОДЕЛИ В ЗАДАЧЕ ТРЕХМЕРНОЙ РЕКОНСТРУКЦИИ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент |  |  |  | М.В. Пичугин |
|  |  | *подпись* |  |  |
| Руководитель | к.т.н., доцент |  |  | М.М. Заславский |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
| Консультанты | ассистент |  |  | Д.В. Иванов |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
|  | ассистент |  |  | А.И. Маловский |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Санкт-Петербург

2024

**ЗАДАНИЕ**

**НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой МО ЭВМ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ А.А. Лисс |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | Пичугин М.В. | |  | Группа | 0303 |
| Тема работы: Исследование возможностей автоматического обнаружения и устранения ошибок трехмерной модели в задаче трехмерной реконструкции | | | | | |
| Место выполнения ВКР: СПбГЭТУ «ЛЭТИ» кафедра МО ЭВМ | | | | | |
| Исходные данные (технические требования):  Программа должна быть разработана на языке программирования Python 3. | | | | | |
| Содержание ВКР:  Введение, Обзор предметной области, формулировка требований к решению и постановка задачи, Обзор программы, Применение программного обеспечения на практике, Заключение | | | | | |
| Перечень отчетных материалов: пояснительная записка, иллюстративный материал | | | | | |
| Дополнительные разделы: безопасность жизнедеятельности | | | | | |
|  | | | | | |
| Дата выдачи задания | | Дата представления ВКР к защите | | | |
| «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. | | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. | | | |
|  | |  | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент |  |  |  | М.В. Пичугин |
|  |  | *подпись* |  |  |
| Руководитель | к.т.н., доцент |  |  | М.М. Заславский |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
| Консультанты | ассистент |  |  | Д.В. Иванов |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |

**КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ**

**ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой МО ЭВМ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ А.А. Лисс |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | Пичугин М.В. |  | Группа | 0303 |
| Тема работы: Исследование возможностей автоматического обнаружения и устранения ошибок трехмерной модели в задаче трехмерной реконструкции | | | | |

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование работ | Срок выполнения |
| 1 | Обзор литературы по теме работы | 03.03 – 10.04 |
| 2 | Обзор предметной области | 10.04 – 14.04 |
| 3 | Обзор программы | 14.04 – 20.04 |
| 4 | Применение программного обеспечения на практике | 20.04 – 23.04 |
| 5 | Безопасность жизнедеятельности | 23.04 – 27.04 |
| 6 | Оформление пояснительной записки | 27.04 – 05.05 |
| 7 | Оформление иллюстративного материала | 06.05 – 11.05 |
| 8 | Предзащита | 13.05.2024 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент |  |  |  | М.В. Пичугин |
|  |  | *подпись* |  |  |
| Руководитель | к.т.н., доцент |  |  | М.М. Заславский |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
| Консультанты | ассистент |  |  | Д.В. Иванов |

**РЕФЕРАТ**

Пояснительная записка 56 стр., 21 рис., 1 табл., 17 ист.

ОБЛАКО ТОЧЕК, ТРЕХМЕРНАЯ РЕКОНСТРУКЦИЯ.

**Объектом исследования** является трехмерная модель, полученная в результате трехмерной реконструкции.

**Предметом исследования** является процесс автоматического обнаружения и устранения ошибок в трехмерных моделях.

**Цель работы:** исследование методов автоматического обнаружения и устранения ошибок в трехмерной модели в контексте задачи трехмерной реконструкции.

С увеличением интереса к трехмерной реконструкции в различных областях, таких как компьютерное зрение, графический дизайн и инженерия, возникает насущная необходимость в разработке эффективных методов автоматического обнаружения и исправления ошибок в трехмерных моделях, с особенным акцентом на их применение в строительстве. Эта актуальность проявляется в различных сферах, включая использование трехмерной реконструкции в инженерных проектах, где она может использоваться для точного моделирования крупногабаритных объектов, таких как здания. В этом контексте автоматическое обнаружение ошибок становится ключевым элементом, обеспечивающим точность и надежность результатов, что напрямую влияет на качество проектирования и строительства. В промышленности также возникает потребность в автоматизированном контроле качества, анализе деформаций и создании точных копий деталей или конструкций, подчеркивая важность разработки методов автоматического обнаружения и исправления ошибок в трехмерных моделях для крупногабаритных объектов в строительной отрасли.

**ABSTRACT**

With the increasing interest in three-dimensional reconstruction in various fields such as computer vision, graphic design and engineering, there is an urgent need to develop effective methods for automatic detection and correction of errors in three-dimensional models, with particular emphasis on their application in construction. This relevance is evident in various fields, including the use of three-dimensional reconstruction in engineering projects, where it can be used to accurately model large-sized objects such as buildings. In this context, automatic error detection becomes a key element ensuring the accuracy and reliability of the results, which directly affects the quality of design and construction. There is also a need in the industry for automated quality control, deformation analysis and the creation of exact copies of parts or structures, emphasizing the importance of developing methods for automatic detection and correction of errors in three-dimensional models for large-sized objects in the construction industry.

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ**](#_heading=) 9

[**1 ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ**](#_heading=h.gjdgxs) 11

[1.1 Трехмерная реконструкция](#_heading=h.30j0zll) 11

[1.1.1 Основные понятия](#_heading=) 11

[1.1.2 Исходные данные для трехмерной реконструкции](#_heading=h.jxs677991y1v) 12

[1.2 Характеристики облаков точек](#_heading=h.uphw4id853f) 14

[1.2.1 Плотность выборки](#_heading=h.gygx71ln4dys) 14

[1.2.2 Шум](#_heading=h.tlz34bb09byl) 15

[1.2.3 Выбросы](#_heading=h.rjw3bgja1icm) 15

[1.2.4 Смещение](#_heading=h.op7jh0yz1hqr) 16

[1.2.5 Отсутствующие данные](#_heading=h.noalsgiv084) 17

[1.3 Ввод данных в облако точек](#_heading=h.hpcyh6o17s2a) 17

[1.3.1 Неориентированные нормали](#_heading=h.p2t7s8ln1dax) 18

[1.3.2 Ориентированные нормали](#_heading=h.tc3yq1lrnx9g) 19

[1.3.3 Информация о сканере](#_heading=h.8el2o4cq4nw6) 21

[1.4 Предварительная обработка модели](#_heading=h.b3m0ht7tiws5) 22

[1.4.1 Гладкость поверхности](#_heading=h.1iw2sl76vwot) 23

[1.4.2 Видимость](#_heading=h.uajqcxvzwdya) 23

[1.4.3 Гладкость объема](#_heading=h.2cayse7vm11m) 24

[1.4.4 Примитивы](#_heading=h.qjzgmoorgnjn) 25

[1.4.5 Глобальная регулярность](#_heading=h.mambxd8n9o53) 25

[1.5 Модели сглаживания поверхности](#_heading=h.akaqb661g1a8) 27

[1.5.1 Локальные модели гладкости поверхности](#_heading=h.virn7ik3x9k4) 27

[1.5.2 Перемещающиеся наименьшие квадраты](#_heading=h.bimd4a5lhsr1) 27

[1.5.3 Иерархические методы](#_heading=h.n6xwelmnuaut) 29

[1.5.4 Локально оптимальная проекция (LOP)](#_heading=h.rjz9rpgj8ycp) 29

[**2 ФОРМУЛИРОВКА ТРЕБОВАНИЙ К РЕШЕНИЮ И ПОСТАНОВКА ЗАДАЧИ**](#_heading=h.73por31qpvmv) 30

[2.1 Постановка задачи](#_heading=h.uecfd5gurgw5) 30

[2.2 Требование к решению](#_heading=h.2jcom0s34xqw) 30

[2.3 Обоснование требований к решению](#_heading=h.x4exi9kdvv8f) 30

[**3 ОБЗОР ПРОГРАММЫ**](#_heading=h.7k6f9vvv7o68) 31

[3.1 Язык программирования](#_heading=h.3no4r4pwx13m) 31

[3.2 Выбор среды разработки](#_heading=h.bojb2a4r9qdl) 31

[3.3 Используемые библиотеки](#_heading=h.gff12jbbqm2o) 32

[3.4 Системные требования](#_heading=h.c82ey5cztshd) 32

[3.5 Разработанные функции](#_heading=h.y6mjbbc0axs3) 33

[**4 ПРИМЕНЕНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ НА ПРАКТИКЕ**](#_heading=h.6qu3y8u1l7cg) 35

[4.1 Удаление выбросов на основе статистического анализа](#_heading=h.i2jr4amjr22y) 35

[4.2 Удаление выбросов на основе радиуса](#_heading=h.77ts0mg8jr9c) 37

[4.3 Использование фильтра среднего значения](#_heading=h.g9jzg7dr8723) 41

[4.4 Использование фильтра Лапласа](#_heading=h.qjolhwdo6dv7) 42

[4.5 Использование фильтра Таубина](#_heading=h.8iq64ohpgw8l) 43

[4.6 Сравнение используемых методов](#_heading=h.8iq64ohpgw8l) 45

[**5 БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ**](#_heading=h.3sus0b52sm9q) 47

[5.1 Основные положения](#_heading=h.br5bgodqeoyg) 47

[5.2 Эргономика среды разработки](#_heading=h.mpq40qf17usr) 47

[5.2.1 Приемлемость организации диалога для выполнения производственного задания](#_heading=h.cc36sxg8qjym) 47

[5.2.2 Информативность](#_heading=h.i7x11rc1f82l) 48

[5.2.3 Соответствие ожиданиям пользователей](#_heading=h.pizo9n7ya8qz) 49

[5.2.4 Пригодность для обучения](#_heading=h.6l2v2j3o7vwm) 51

[5.2.5 Контролируемость](#_heading=h.fipyhy2rwzwm) 51

[5.2.6 Устойчивость к ошибкам](#_heading=h.179n7j3phuhr) 51

[5.2.7 Адаптируемость к особенностям пользователей](#_heading=h.suf9sr8fbjr8) 52

[5.3 Эргономика программ, написанных в PyCharm](#_heading=h.nd3dq6th1lcu) 52

[5.4 Выводы](#_heading=h.7atsh6ums37g) 52

[**ЗАКЛЮЧЕНИЕ**](#_heading=) 53

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**](#_heading=h.1fob9te) 55

**ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**

ЯП – язык программирования;

IDE – Integrated Development Environment;

## ВВЕДЕНИЕ

С ростом интереса к трехмерной реконструкции в различных областях, включая компьютерное зрение, графический дизайн и инженерное дело, возникает неотложная потребность в разработке эффективных методов для автоматического обнаружения и исправления ошибок в трехмерных моделях. Эта актуальность подкрепляется важностью точности и надежности результатов трехмерной реконструкции, которая непосредственно влияет на качество проектирования и строительства.

Точность трехмерных моделей становится критически важной, когда речь идет о применении этих технологий в критичных к безопасности секторах, таких как авиастроение, автомобилестроение и медицина. Ошибки в трехмерных моделях могут привести к серьезным последствиям, включая финансовые потери и угрозы для жизни. Поэтому разработка методов, которые могут автоматически обнаруживать и исправлять такие ошибки, становится первостепенной задачей для исследователей и разработчиков.

**Цель работы:** исследование методов автоматического обнаружения и устранения ошибок в трехмерной модели в контексте задачи трехмерной реконструкции.

**Задачи работы:**

1. Рассмотреть различные типы ошибок, возникающих в облаках точек.
2. Проанализировать существующие методы и подходы к трехмерной реконструкции и обработке облаков точек.
3. Разработать методы для автоматического устранения и обнаруженных ошибок в облаках точек.
4. Провести эксперименты для оценки эффективности разработанных методов на реальных исходных данных.

**Объектом исследования** является трехмерная модель, полученная в результате трехмерной реконструкции.

**Предметом исследования** является процесс автоматического обнаружения и устранения ошибок в трехмерных моделях.

**Практическая значимость** данного исследования выражается в возможности повышения точности и надежности трехмерных моделей, что имеет прямое применение в инженерных проектах, строительстве и промышленности. Эффективные методы автоматического обнаружения и устранения ошибок в трехмерных моделях способствуют оптимизации процессов проектирования, контроля качества и анализа деформаций, что значительно улучшает рабочие процессы и повышает безопасность при эксплуатации сооружений и производстве изделий. Кроме того, результаты исследования вносят вклад в развитие теоретических основ дисциплины, расширяя понимание сложных взаимодействий в трехмерной реконструкции и предоставляя исследователям новые инструменты для дальнейших изучений в этой области. Работа, опираясь на исследования отечественных и зарубежных авторов, а также данные научно-практических конференций и периодической печати, способствует наращиванию академического и практического потенциала в области трехмерной реконструкции.

## 1 ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

### 1.1 Трехмерная реконструкция

#### 1.1.1 Основные понятия

Трехмерная реконструкция – это процесс, при котором из набора 2D-изображений или видеороликов, создается трехмерное представления объекта или сцены. Этот процесс представляет собой сложную задачу, требующую анализа входных данных для извлечения геометрической информации, такой как глубина, форма и текстура. Конечная цель заключается в создании цифровой модели, которая максимально точно отражает структуру и внешний вид реального объекта.

Облако точек — это набор данных или структура, состоящая из множества точек в пространстве. В контексте компьютерной графики и трехмерной реконструкции, облака точек обычно получают с помощью трехмерных сканеров, которые измеряют большое количество точек на поверхности внешних объектов. Каждая точка в таком облаке представляет собой координату в трехмерном пространстве, часто сопровождаемую дополнительными данными, такими как цвет и интенсивность [1].

Прогресс в сообществе компьютерной графики радикально изменил наши возможности цифрового представления окружающего мира. Одной из подобластей, которая активно развивалась в ходе этой революции, стала реконструкция поверхностей. В основе этого процесса лежит воссоздание трехмерного объекта из набора дискретных точек, описывающих его форму. Данный обзор собирает основные направления и достижения сообщества, решающего различные аспекты этой задачи, и анализирует, как внедрение новых технологий, алгоритмические новшества и практическое применение влияют на передовые достижения в этой области.

Значение реконструкции поверхностей возросло в основном благодаря новым методам получения облаков точек. Сначала эти технологии включали активные методы, такие как оптические лазерные сканеры, сканеры со структурированным светом и лидары, а также пассивные методы, такие как мульти-видовая стереосъемка. Эти устройства кардинально изменили подходы к инженерным работам и быстрому прототипированию, развиваясь наряду с технологиями компьютерного проектирования.

Компьютерная графика проявила немедленный интерес к таким технологиям, следуя одной из своих основных целей: моделированию, распознаванию и анализу реального мира. Кроме того, современные приложения начали использовать такие сканеры во всех областях данных наук, от микро- до макроуровня. Относительно новым трендом стало массовое распространение облаков точек, полученных с помощью недорогих коммерческих сканеров в реальном времени. Это оказало влияние на множество областей, включая автомобильный дизайн, инженерию, археологию, телекоммуникации и искусство, расширяя границы применимости этих технологий и углубляя наше понимание трехмерного мира [2].

#### 1.1.2 Исходные данные для трехмерной реконструкции

В современных методиках сбора данных для трехмерной реконструкции присутствует ряд вызовов, связанных с особенностями облаков точек, которые они генерируют. Каждый метод сбора данных создает уникальные облака точек с разнообразными свойствами и неточностями. Эти особенности, наряду с характеристиками сканируемых объектов, определяют подходы к реконструкции, используемые в настоящее время. Методы варьируются от тех, которые исходят из предположения о качественно собранном облаке точек и способны адаптироваться к любым формам, создавая герметичные сетчатые модели, до тех, которые основаны на минимальных предположениях о качестве данных и ориентированы на специфические классы форм, выдавая представления объектов, не основанные на сетках.

Рассматривая алгоритмы реконструкции с точки зрения автоматического обнаружения и устранения ошибок в трехмерных моделях. Это включает анализ предположений, которые алгоритмы делают для коррекции дефектов в облаках точек и определения, какая информация о форме объекта будет восстановлена. Без таких предположений задача реконструкции является некорректно поставленной; множество потенциальных поверхностей может соответствовать одному и тому же набору точек. Предположения часто касаются самого облака точек, такие как плотность выборки, уровень шума и выравнивание. Также крайне важны предположения, касающиеся характеристик сканируемого объекта, например, локальная и объемная гладкость, отсутствие границ, симметрии, элементарные формы, общая регулярность и предположения, основанные на данных [3,4].

В некоторых случаях важно также понимание условий, при которых получены данные, таких как положение сканера и RGB-изображения объекта. Пользователи могут вносить коррективы, указывая алгоритмам высокоуровневые подсказки для реконструкции. Эти элементы помогают упорядочить иначе сложно решаемую задачу трехмерной реконструкции, особенно при работе с облаками точек, содержащими серьезные недостатки.

Разработка и классификация методов реконструкции с учетом этих предпосылок помогает ограничить ожидания и определить приоритеты для результатов реконструкции в зависимости от применения. Например, в археологии доступные плотные сканирования с хорошим покрытием позволяют использовать стандартные предположения о гладкости для детализированной реконструкции. В градостроительстве, напротив, восстановление мелких деталей может быть ненужным или невозможным из-за неполноты данных от наземного лидара. В этих контекстах предположения о глобальной регулярности могут способствовать дополнению деталей и созданию интерактивных, реалистичных заменителей отсутствующих элементов.

Исторически предположения развивались вместе с типами обрабатываемых облаков точек, стимулируя разработку специализированных подходов для динамичного захвата сцен. Этот процесс подчеркивает значимость нашего исследования в контексте разработки методов для будущих типов сбора данных, делая его ценным для понимания и улучшения процессов трехмерной реконструкции.

### 1.2 Характеристики облаков точек

В процессе трехмерной реконструкции важными факторами, влияющими на методы восстановления, являются свойства входного облака точек. Основные характеристики облаков точек, оказывающие наибольшее влияние на алгоритмы реконструкции, включают плотность выборки, шум, выбросы, несоответствие и отсутствие данных.

#### 1.2.1 Плотность выборки

Плотность выборки относится к распределению точек, выбираемых для изучения поверхности. Это критически важно для определения локальной геометрии поверхности, например, её касательной плоскости. Для этого определяется окрестность — набор точек, находящихся близко к данной точке, достаточно описывающих локальную геометрию, но при этом достаточно малых, чтобы сохранить локальные особенности. При равномерной плотности выборки окрестность может быть одинаково определена для каждой точки. Например, окрестность точки p в облаке P может быть определена с помощью ε-сферы, состоящей из точек , каждая из которых удовлетворяет условию , где ε — это константа для всех точек.

В 3D-сканах обычно наблюдается неоднородная плотность выборки из-за различного расстояния от сканируемой формы до позиции сканера, его ориентации и геометрических особенностей формы [6]. Для учета локальных вариаций в плотности выборки часто используется метод k ближайших соседей (knn) для построения окрестности. Альтернативный подход заключается в использовании пространственно изменяющейся ε-сферы, определяемой как функция от окрестности k ближайших соседей.

Более сложные техники оценки плотности выборки включают методы, основанные на оценке ошибок реконструкции и ядерных методах. Существуют методы, где определяется размер окрестности для каждой точки, ограничивая ошибку аппроксимации поверхности методом наименьших квадратов (MLS), где выбранное значение минимизирует эту оценку ошибки. Некоторые формулируют ошибку аппроксимации MLS поверхности в терминах ядерной регрессии, где оптимальный размер окрестности может быть определен через точечную ошибку или ошибку в поддерживающей области определенного ядра.

#### 1.2.2 Шум

Точки, случайно распределенные около поверхности, обычно рассматриваются как шум. Это распределение часто является результатом сканирования и связано с такими артефактами, как шум датчика, квантование глубины, а также расстояние или ориентация поверхности относительно сканера. У некоторых популярных сканеров шум появляется вдоль линии визирования и может зависеть от свойств поверхности, включая характеристики рассеяния материалов. В условиях такого шума основная цель алгоритмов реконструкции поверхности состоит в том, чтобы создать поверхность, проходящую около точек, но не приспосабливающуюся излишне к шуму. Разработка устойчивых алгоритмов, которые налагают гладкость на результат, а также методы, использующие устойчивую статистику, являются распространенными способами борьбы с шумом.

#### 1.2.3 Выбросы

Точки, значительно отклоняющиеся от истинной поверхности, классифицируются как выбросы. Выбросы в облаке точек определяются как те, что находятся на большом расстоянии от поверхности. Они часто возникают из-за импульсивного шума при практическом сканировании.

Выбросы обычно связаны со структурными артефактами в процессе сбора данных. Иногда выбросы распределены случайно в объеме, где их плотность меньше плотности точек, выборочно образующих поверхность. Однако выбросы также могут быть более структурированными, когда существуют высокоплотные кластеры точек, находящиеся далеко от поверхности. Это может происходить при многовидовом стереосканировании, где зависимые от точки зрения блики могут привести к неверным соответствиям. В отличие от шума, выбросы — это точки, которые не следует использовать для восстановления поверхности, будь то явное обнаружение или неявные устойчивые методы обработки.

#### 1.2.4 Смещение

Смещение происходит из-за неполного или некорректного совмещения диапазонов сканирования. Эта проблема особенно актуальна, когда начальная конфигурация серии сканирований значительно отличается от идеального выравнивания. В условиях, когда сканирование ограничивается одним объектом, объект обычно вращается относительно датчика для каждого скана; таким образом, степень смещения ограничена, поскольку начальное выравнивание сканов может быть оценено по известным вращениям. Однако в технологиях SLAM (одновременная локализация и картографирование), основанных на RGBD-картировании, может происходить дрейф — постепенное накопление ошибок регистрации между последовательными сканами. Это может привести к значительному смещению между сканами одного объекта, сделанными в разное время, известное как "замыкание петли"[6,7].

Для коррекции смещения применяются методы, отличные от тех, что используются для борьбы с обычным шумом. Например, при применении предположения о "манхэттенском мире", где сцена состоит из плоских примитивов, выровненных вдоль трех ортогональных осей, плоские примитивы из ошибочно повернутых сканов могут быть надежно "прищелкнуты" к одной из этих осей. В рамках предположения о повторяющихся взаимосвязях между геометрическими примитивами, смещенный скан может быть скорректирован, если он не соответствует обнаруженным повторениям.

#### 1.2.5 Отсутствующие данные

Одной из основных причин, побуждающих разработку методов реконструкции, является проблема отсутствующих данных. Эти данные могут отсутствовать по ряду причин, включая ограниченный диапазон датчика, высокое поглощение света и закрытие видимости в процессе сканирования, когда большие участки формы не подвергаются семплированию. Хотя вышеупомянутые артефакты постоянно улучшаются, отсутствующие данные часто остаются из-за физических ограничений устройства. Важно отметить, что отсутствующие данные отличаются от неоднородной выборки тем, что плотность выборки в таких регионах равна нулю.

Многие методы решения проблемы отсутствующих данных основаны на предположении, что сканируемая форма является водонепроницаемой. Цель этих методов — справляться с упомянутыми выше вызовами в местах, где данные имеются, и восстанавливать геометрию в частях поверхности, которые не были просканированы. Другие подходы направлены на решение проблемы отсутствующих данных путем попыток восстановления топологических структур исходной поверхности, даже за счет потери геометрической точности, например, нахождение поверхности, гомеоморфной исходной форме.

Для значительно отсутствующих данных другие подходы стремятся к реконструкции информации более высокого уровня, такой как скелет, базовые формы, симметричные отношения и канонические регулярности.

### 1.3 Ввод данных в облако точек

Методы реконструкции требуют разнообразных входных данных, связанных с облаком точек. Минимальное требование для всех алгоритмов — это наличие набора трехмерных точек, которые представляют собой выборки поверхности. Однако использование исключительно точек может быть недостаточным для эффективного восстановления данных определенных типов облаков точек. Дополнительные типы входных данных могут значительно улучшить процесс восстановления данных из сложных облаков точек. Важные формы входных данных, обычно ассоциируемые с облаками точек, включают нормали к поверхности, информацию со сканера и RGB изображения [2,3,6].

#### 1.3.1 Неориентированные нормали

Нормали к поверхности являются критически важными входными данными для методов реконструкции. Для гладких поверхностей каждая нормаль уникально определяется в каждой точке и направлена перпендикулярно касательной плоскости точки, которая является локальным приближением поверхности в этой точке. Ориентированные нормали предоставляют ценные подсказки для алгоритмов реконструкции, облегчая процесс восстановления.

С другой стороны, существуют неориентированные нормали, у которых нет фиксированного направления, они могут быть направлены как внутрь, так и наружу поверхности. Эти данные могут использоваться для определения плоских участков в облаке точек, проецирования точек на приближенную поверхность или создания поля беззнаковых расстояний.

Неориентированные нормали часто вычисляются напрямую из облака точек. Распространенным и простым методом вычисления нормали в заданной точке *p* в облаке является использование анализа главных компонент (PCA) в локальной окрестности точки *p*. Этот метод оценивает нормаль как наименьший собственный вектор ковариационной матрицы, созданной по локальным точкам, собранным с использованием ε-сферы или k ближайших соседей. PCA обеспечивает общую оценку соответствия касательной плоскости к плоскости наименьших квадратов и может быть чувствителен к дефектам в облаке точек, таким как плотность выборки и шум. Однако это предоставляет ценную информацию о локальной геометрии поверхности, несмотря на потенциальные несовершенства данных.

) h

где k — кривизна в точке p, r — радиус для шара r - радиуса, используемого при построении окрестности, — величина шума при нулевом среднем независимом и одинаково распределенном шуме, p — плотность выборки, а , — константы, независимые от этих величин. Здесь виден компромисс между шумом и размером окрестности: при уменьшении радиус r должен быть малым, но при введении шума r должен быть большим, чтобы противостоять воздействию шума [8].

Другие методы оценки нормалей включают использование взвешенной ковариационной матрицы и более высокого порядка аппроксимаций через оскулирующие джеты. Подобно стандартному PCA, эти методы требуют локальной окрестности точек; следовательно, наличие дефектов в облаке точек может серьезно затруднить поиск оптимальной окрестности. В результате методы оценки могут производить довольно шумные нормали, что требует от алгоритмов реконструкции высокой устойчивости к таким данным.

#### 1.3.2 Ориентированные нормали

**Ориентированные нормали** — это нормали, имеющие постоянное направление, указывающее либо внутрь, либо наружу поверхности. Знание внешней и внутренней сторон поверхности оказывается чрезвычайно полезным для реконструкции поверхностей. Это знание используется для создания поля знакового расстояния по всему окружающему пространству, где поле принимает положительные значения снаружи и отрицательные внутри поверхности. Поверхность тогда представляется как нулевое пересечение этого знакового поля расстояния. Другие методы обобщают это на неявные поля и индикаторные функции, но базовая идея построения внешней и внутренней сторон остается неизменной.

Существует множество способов вычисления ориентированных нормалей. Если известны исходные 2D сканирования, то 2D решетчатая структура предоставляет возможность выполнять последовательную ориентацию, поскольку всегда известно, как повернуться по часовой стрелке вокруг данной вершины. Например, если обозначить точку в сканировании диапазона на пикселе (x, y) как , то нормаль в можно просто взять как векторное произведение между и . Если облако точек шумное, этот метод может производить довольно шумные нормали, поскольку он не агрегирует точки в перекрывающихся сканированиях.

Если информация о сканере отсутствует, то точки должны ориентироваться исключительно на основе неориентированных нормалей. Известный метод достигает этого, создавая граф над облаком точек (например, через knn-соседство каждой точки) и взвешивая каждое ребро для точек и на основе сходства их неориентированных нормалей и как = 1 - . Затем строится минимальное остовное дерево, где, зафиксировав ориентацию нормали в одной точке, служащей корнем, ориентация нормали распространяется по дереву. Метод [HLZ\*09] корректирует веса , отдавая приоритет распространению вдоль тангенциальных направлений — если оценочное касательное пространство между двумя точками ( - ) перпендикулярно направлениям нормалей, это указывает на то, что эти две точки являются допустимыми соседями, а не принадлежат противоположным сторонам поверхности [4, 8].

Хотя эти методы способны справляться с неоднородной выборкой, шумом и несоответствием до определенной степени, они все же остаются чувствительны к несовершенствам в облаке точек и, как результат, могут оставить некоторые нормали неориентированными или указывающими в неправильном направлении. Влияние на реконструкцию поверхности в значительной степени зависит от распределения некорректных ориентаций: если они распределены случайно, методы могут рассматривать это как случайный шум, но, если неправильные ориентации сконцентрированы в больших регионах, такой структурированный шум может быть сложно обработать.

#### 1.3.3 Информация о сканере

Информация от сканера, с помощью которого было получено облако точек, может оказаться полезной для реконструкции поверхностей. Его 2D решетчатая структура позволяет оценить плотность выборки, что может использоваться для обнаружения определенных форм выбросов — точек, соседи по решетке которых находятся на значительно большем расстоянии, чем плотность выборки, что делает их вероятными выбросами. Тем не менее, следует проявлять осторожность при различении выбросов и резких особенностей.

Информация о сканере также может использоваться для определения уверенности в точках, что полезно при обработке шума. Некоторые сканеры (например, LiDAR) предоставляют меры уверенности в виде измеренной отражательной способности каждой точки. Уверенность также может быть получена из информации о линии визирования — это совокупность линейных сегментов между каждой точкой в облаке точек и положением головки сканера, с которой была получена данная точка. В активных системах сканирования, таких как лазерные сканеры, большой угол между линией визирования и нормалью к поверхности может привести к шумному оцениванию глубины, то есть к плохой оценке пика лазера, что указывает на низкую уверенность [4,5,8].

Линия визирования также определяет регион пространства, который считается находящимся за пределами формы. Комбинирование линий визирования из нескольких сканов уточняет ограничивающий объем, в котором находится поверхность — это известно как визуальный корпус. Эта информация особенно полезна при обработке неполных данных — она может указывать на наличие большой впадины в форме.

**RGB-изображения**

Различные способы получения данных, дополняющие сбор глубины, могут значительно помочь. Получение RGB-изображений — очень распространенный метод, который сопровождает множество датчиков, таких как Microsoft Kinect. В случае Kinect RGB-камера расположена в одном месте с ИК-камерой, поэтому, предполагая, что обе камеры откалиброваны, легко идентифицировать соответствующие значения глубины и RGB на уровне пикселей. RGB-изображения наиболее полезны для реконструкции, когда они могут дополнять информацию о глубине, которая либо не так описательна, как визуальный вид, либо просто не измеряется данными. Например, если цветное изображение и 3D-скан сделаны с широкого базиса, содержащего очень разные виды, то сегментированные части изображения могут быть использованы для дополнения глубинной информации.

### 1.4 Предварительная обработка модели

Разработка методов реконструкции поверхностей в значительной мере зависит от технологий сбора данных. Способы сбора данных формируют ожидания относительно возможных классов форм, которые могут быть восстановлены, и типов артефактов, связанных с полученными данными, определяя тем самым характеристики выходных данных, производимых алгоритмами реконструкции, и их точность. В этом контексте рассматривается несколько основных моделей предварительной обработки, каждая из которых адаптирована для специфических входных данных и производит соответствующие результаты, а также типичные классы форм и методы сбора данных, характерные для этих сценариев [8].

#### 1.4.1 Гладкость поверхности

Модель гладкости поверхности предполагает, что восстановленная поверхность будет обладать определенным уровнем гладкости, при этом точно соответствуя исходным данным. Часто используемая форма — это локальная гладкость, которая стремится обеспечить гладкость в непосредственной близости от данных. Такие подходы обычно создают поверхности, которые сглаживают шумы, связанные с процессом сбора данных, одновременно сохраняя граничные компоненты, где данные либо недостаточно плотные, либо отсутствуют. Благодаря своей универсальности, эти методы могут применяться для различных форм и устройств сбора данных, но без дополнительных предположений они могут столкнуться с трудностями при обработке серьезных артефактов.

В отличие от локальной, глобальная гладкость предполагает более высокий уровень гладкости и может охватывать более широкие пространственные масштабы. Высокий порядок гладкости связан с изменениями дифференциальных свойств поверхности, таких как области, касательные плоскости, кривизна и т. д. Такие методы обычно сфокусированы на реконструкции отдельных объектов и создании водонепроницаемых поверхностей, что ограничивает класс восстанавливаемых форм [9]. Для создания таких облаков точек обычно используются настольные сканеры, способные сканировать небольшие и средние объекты. Типичные устройства для таких сценариев включают лазерные сканеры оптической триангуляции, сканеры времени полета и сканеры с ИК-структурированным освещением.

Модели кусочно-гладких поверхностей направлены на сохранение резких особенностей формы. Устройства сбора данных для таких моделей мало отличаются от используемых для глобальной или локальной гладкости, однако класс форм ограничен теми, которые содержат четко выраженные резкие особенности, например, CAD-модели или искусственные структуры.

#### 1.4.2 Видимость

Модель видимости учитывает внешние пространственные данные о восстановленной сцене и использует эту информацию для борьбы с шумами, неравномерностью выборки и пропущенными данными. Видимость сканера служит мощным инструментом, так как может предоставить модель шума, зависящую от данных, и помочь определить пустые области пространства. Это позволяет фильтровать структурированный шум, что особенно важно при реконструкции водонепроницаемых объектов из данных многокамерных систем. В некоторых исследованиях модель видимости расширяется для поддержки интерактивной реконструкции сцен, смягчая требования к водонепроницаемости и сохраняя поверхность только в непосредственной близости к данным.

Сканеры, такие как Microsoft Kinect и Intel RealSense, обеспечивают интерактивное получение геометрических данных и позволяют реконструировать значительные пространства, например, интерьеры зданий, но часто за счет геометрической точности по сравнению с более статичными установками сканирования.

#### 1.4.3 Гладкость объема

Модель гладкости объема направлена на обеспечение гладкости в отношении изменений объема формы, что доказало свою эффективность при работе с большими объемами пропущенных данных. Некоторые модели объема стремятся к водонепроницаемой реконструкции отдельных объектов с акцентом на топологическую точность, предполагая, что полная выборка объекта невозможна, в основном из-за самозатенения и ограниченной мобильности датчика [8,10]. Примером могут служить искусственные объекты, изготовленные из материалов, таких как катушки или металлические проволоки, где форму можно описать как сложное расположение обобщенных цилиндров. Другие подходы сосредоточены на извлечении скелетной структуры формы при значительном отсутствии данных, что характерно для органических форм, таких как деревья, которые обычно сканируются в неконтролируемых внешних условиях с использованием лидаров, в результате чего многие ветви и листья могут быть зафиксированы только частично.

#### 1.4.4 Примитивы

Геометрическая модель примитивов основывается на предположении, что геометрия сцены может быть представлена компактным набором простых геометрических форм, таких как плоскости, кубы, сферы, цилиндры и т.д. В контексте реконструкции водонепроницаемых отдельных объектов, методы обнаружения примитивов могут использоваться для экстраполяции данных при наличии значительного количества пропущенной информации. Формы CAD, которые обычно моделируются с использованием базовых геометрических фигур, естественно вписываются в этот тип модели. Однако сканированные облака точек могут быть неполными из-за сложных самозатенений, а некоторые материалы CAD-моделей могут быть непригодны для сканирования на стандартном оборудовании, что может привести к структурированному шуму. Таким образом, модель примитивов может быть полезна для надежного выявления базовых, более простых форм в зашумленных облаках точек.

Внутренние пространства представляют собой другой класс форм, который идеально подходит для модели примитивов, так как их можно представить в виде набора плоскостей или кубов. Типичным примером применения является реконструкция планов зданий. Такие среды могут быть захвачены сканерами LiDAR, где полное сканирование может быть затруднено из-за обширного охвата сцены. Плоскости могут быть полезным инструментом для дополнения неполных данных и для устранения шума. Однако, как и в случае с формами CAD, мелкие геометрические детали могут быть потеряны. Несмотря на это, были разработаны гибридные методы, которые сохраняют детализацию, одновременно извлекая примитивы [11].

#### 1.4.5 Глобальная регулярность

Модель глобальной регулярности основывается на предположении, что многие формы, такие как модели CAD, искусственные и архитектурные формы, обладают определенным уровнем регулярности в их структуре. Регулярность может проявляться в различных формах, например, в зданиях с фасадами, состоящими из повторяющихся элементов, или в интерьерах, где пространства организованы в регулярные композиции.

Фасады зданий, например, часто характеризуются повторяющимися элементами, такими как окна, расположенные в равномерной сетке. Однако данные, полученные о фасаде, часто содержат значительный шум и неполные измерения из-за ограничений устройств сбора данных, таких как LiDAR или многокамерные стереосистемы, которые могут измерять только на больших расстояниях и подвергаться затенению от окружающих объектов. Если регулярность обнаруживается во входных данных, она может быть использована для моделирования оставшейся части фасада, устранения шума и заполнения пропущенных элементов.

Основной проблемой, которую устраняет глобальная регулярность в интерьерах зданий и механических деталях, является несоответствие сканов. Например, интерьеры зданий могут быть захвачены путем регистрации глубинных сканов в реальном времени, но несовершенства регистрации могут проявляться как дрейф. Обеспечение регулярности углов между обнаруженными плоскостями может быть весьма полезным для коррекции такого дрейфа. Данные о механических частях, полученные с помощью настольных 3D-сканеров, также могут привести к плохой регистрации глубинных сканов, но поиск канонических отношений в извлеченных геометрических примитивах может помочь исправить такие ошибки. Задачей реконструкции для механических деталей часто является создание водонепроницаемой реконструкции отдельного объекта, в то время как для интерьеров зданий основная задача заключается в обнаружении плоскостей, а не обязательно в создании водонепроницаемой реконструкции [12]. Однако из-за больших размеров интерьеров зданий может быть больше свидетельств регулярности, чем в механической детали, что ведет к компромиссу между точностью реконструкции и регулярностью.

#### 

### 1.5 Модели сглаживания поверхности

Модели гладкости поверхности можно классифицировать на локальную гладкость, глобальную гладкость и кусочно-гладкую гладкость. Эти категории отличаются по степени и способу применения ограничений гладкости.

В контексте трехмерной реконструкции облако точек P, представляющее форму S, состоит из индивидуальных точек для i -той точки. Для эффективного моделирования требуется поле нормалей N, состоящее из векторов нормалей, где каждой точке из *P* соответствует нормаль из *N*. В зависимости от метода реконструкции, нормали могут быть ориентированы или неориентированы [4, 8, 13].

#### 1.5.1 Локальные модели гладкости поверхности

Метод, влияющий на подходы к локальному ограничению гладкости, аппроксимирует подписанное поле расстояний , присваивая каждой точке x в пространстве её проекцию на касательную плоскость к ближайшей точке в *P*:

Ориентация поля нормалей N необходима для точной оценки подписанного поля расстояний. Поверхность определяется как место, где f равно нулю. Этот метод прост в реализации, но чувствителен к оценкам нормалей; шумные или неправильно ориентированные нормали могут сильно исказить оценки расстояний. Последующие методы, основанные на локальной гладкости, стараются решить эти проблемы [6, 14, 15].

#### 1.5.2 Перемещающиеся наименьшие квадраты

Этот подход аппроксимирует реконструкцию, адаптируя пространственно изменяемый полином низкой степени к входным точкам. Предполагая, что каждой входной точке присвоено скалярное значение , восстановленный сигнал в точке x определяется значением в этой точке, аппроксимированным с использованием заданной полиномиальной функции. В контексте многомерных полиномиальных аппроксимаций, метод MLS ищет функцию g(x), которая оптимально аппроксимирует окрестность точки x в смысле взвешенных наименьших квадратов:

где — гладкая убывающая весовая функция, увеличивающая влияние ближайших к оцениваемой точке образцов. Эта весовая функция выполняет роль фильтра нижних частот. Она является существенным элементом MLS для борьбы с умеренными уровнями шума, позволяя функции веса иметь большее пространственное влияние. Для неравномерной выборки необходимо определить весовую функцию, пространственная поддержка которой изменяется в зависимости от плотности выборки [4].

**Метод Левина**

В первой формулировке MLS, соседние образцы точки оценки x сначала параметризуются относительно локальной касательной плоскости H(x), полученной через взвешенный PCA, как в методах оценки нормалей. В этой параметризации, окрестность может быть воспринята как поле высот с отклонениями , которое аппроксимируется бивариативным полиномом низкой степени . Проекция x определяется как точка, ближайшая к полиномиальной аппроксимации.

Поверхность MLS в конечном итоге неявно определяется как фиксированные точки оператора проекции, соответствующим образом определенного для точек вблизи ввода.

**Плоские аппроксимации**

Метод Левина может быть значительно упрощен путем опущения шага полиномиального фитинга. Поверхность MLS тогда напрямую определяется как нулевой уровень скалярного поля, определенного расстоянием между точкой оценки и наилучшей подогнанной плоскостью. Если используются ориентированные нормали, этот подход может обеспечить точное и эффективное определение поверхности, если доступна, то можно вычислить нормаль n(x) наилучшим образом подогнанной плоскости как взвешенное среднее соседних нормалей [5].

#### 1.5.3 Иерархические методы

Иерархические методы, такие как метод многоуровневого разделения единства (MPU), используют октодеревья для аппроксимации данных в ячейках полиномами или квадриками, зависящими от ориентации нормалей. Если ошибка в ячейке слишком велика, она делится на более мелкие сегменты для уточнения аппроксимации. Локально определенные поля расстояний сглаживаются для формирования глобальной неявной поверхности. Этот метод требует ориентированных нормалей и позволяет управлять гладкостью и устойчивостью к шуму через терпимость к остаточной ошибке.

#### 1.5.4 Локально оптимальная проекция (LOP)

Методы класса LOP не стремятся к строгому восстановлению непрерывной поверхности, а скорее подгоняют произвольный набор точек *Q* к мультивариативной медиане исходного облака точек *P*, обеспечивая равномерное распределение *Q*. Эти методы не требуют информации о нормалях или локальной параметризации и не включают подгонку методом наименьших квадратов[17]. Они особенно подходят для работы с сырыми облаками точек, в которых присутствуют шум, выбросы и несоответствие. Методы LOP привлекают выборки к локальным медианам и одновременно создают равномерное распределение за счет отталкивания слишком близко расположенных точек друг к другу.

**2 ФОРМУЛИРОВКА ТРЕБОВАНИЙ К РЕШЕНИЮ И ПОСТАНОВКА ЗАДАЧИ**

### 2.1 Постановка задачи

Необходимо разработать программу на языке программирования Python для исследования методов нахождения и удаления ошибок в облаке точек.

### 2.2 Требование к решению

Программа должна обладать следующими свойствами:

1. В программе должны быть реализованы методы, которые позволяют найти и устранить ошибки в загруженном облаке точек.

2. В программе должны выводиться параметры, по которым можно сравнить выбранные методы.

3. В программе должен быть метод для визуализации получившихся результатов.

### 2.3 Обоснование требований к решению

1. В программе должны быть реализованы методы, которые позволяют найти и устранить ошибки в загруженном облаке точек, так как это позволит выявить ошибочное расположение точек и исправить его, путем их удаления или перемещения, для улучшения качества последующей трехмерной реконструкции.

2. В программе должны выводиться параметры, по которым можно сравнить выбранные методы, что позволит оценить лучшие методы для той или иной задачи.

3. В программе должен быть метод для визуализации получившихся результатов, что позволит наглядно увидеть получившийся результат действия программы.

## 

## 3 ОБЗОР ПРОГРАММЫ

### 3.1 Язык программирования

Язык программирования Python был выбран в качестве основного языка для разработки в рамках моей выпускной квалификационной работы по нескольким причинам. Python является популярным и широко используемым языком программирования, известным своей простотой, читаемостью и богатой библиотекой. Он предлагает простой в освоении синтаксис, что облегчает процесс написания и чтения кода.

Python обладает мощными возможностями обработки данных, включая поддержку различных типов данных, таких как списки, словари и наборы, что делает его идеальным выбором для работы с трехмерными моделями и их анализом. Библиотеки, доступные для Python, такие как NumPy, SciPy и Open3D, обеспечивают дополнительные инструменты для манипуляции данными, визуализации и применения сложных алгоритмов.

### 3.2 Выбор среды разработки

Выбор PyCharm в качестве интегрированной среды разработки (IDE) для моей выпускной квалификационной работы был продиктован его обширными возможностями и удобством использования. PyCharm предлагает комплексный набор инструментов, оптимизированных для языка программирования Python, такие как автодополнение кода, подсветку синтаксиса и инструменты рефакторинга. IDE обеспечивает эффективную организацию кода, полезные подсказки в режиме реального времени и мощные функции отладки.

Еще одним важным фактором выбора PyCharm является его тесная интеграция с библиотеками и технологиями, используемыми в моей работе. PyCharm обеспечивает беспроблемную поддержку библиотеки Open3D, упрощая работу с трехмерными моделями и обеспечивая беспрепятственный доступ к ее функциям.

### 3.3 Используемые библиотеки

В ходе разработки программы были использованы следующие библиотеки:

* + - Open3D – это, библиотека с открытым исходным кодом, предназначенная для ускорения разработки программного обеспечения, связанного с трехмерными данными. Она предоставляет разработчикам тщательно отобранный набор структур данных и алгоритмов, доступных как на C++, так и на Python. Этот гибридный подход позволяет сочетать высокую производительность C++ с простотой и гибкостью Python [16].
    - NumPy – это, библиотека Python, предназначенная для научных вычислений и обработки данных. Он предоставляет объект многомерного массива, позволяя выполнять быстрые и эффективные операции над массивами данных. NumPy включает в себя различные производные объекты, такие как массивы с масками и матрицы, что упрощает работу с многомерными данными.
    - time – это, библиотека в Python предоставляет инструменты для работы со временем и датами, включая получение текущего времени, форматирование дат и выполнение расчетов с временными интервалами.

### 3.4 Системные требования

Системные требования для использования программы, разработанной, в рамках моей выпускной квалификационной работы были следующими:

1. **Операционная система**: Windows 10 64-битная версия.
2. **Процессор**: Intel Core i5 или эквивалентный.
3. **Память**: 8 ГБ ОЗУ или больше.
4. **Графический процессор**: NVIDIA GeForce GTX 1050 или эквивалентный с поддержкой OpenGL 4.5 или выше.
5. **Хранилище**: не менее 3 ГБ свободного пространства на диске для установки библиотеки и дополнительных файлов.
6. **Программное обеспечение**: Python 3.7 или выше, PyCharm (рекомендуется) или другая IDE по выбору.

### 3.5 Разработанные функции

В ходе написания программы были реализованы следующие функции.

copy\_point\_cloud(point\_cloud) - функция принимает на вход облако точек и возвращает его копию, для сохранения оригинала в исходном состоянии.

load\_point\_cloud(filename) - функция принимает на вход путь к файлу и возвращает объект облака точек в формате, совместимом с Open3D.

mesh\_reconstruction(point\_cloud, depth) - функция принимает на вход облако точек и параметр depth, определяющий качество поверхности. Используется для реконструкции поверхности из облака точек с помощью алгоритма Пуассона.

estimate\_normals(point\_cloud, radius, max\_nn) - функция принимает на вход облако точек, radius, определяющий радиус поиска соседних точек, и max\_nn, определяющий максимальное количество соседних точек. Функция оценивает нормали точек, используя алгоритм поиска кд-дерева (kd-tree), и обновляет облако точек, добавляя к нему информацию о нормалях.

remove\_statistical\_outliers(point\_cloud, nb\_neighbors, std\_ratio) - принимает на вход облако точек, nb\_neighbors, определяющий количество соседних точек для расчета, и std\_ratio, определяющий пороговое значение для выявления выбросов. Используется для удаления статистических выбросов из облака точек.

remove\_radius\_outliers(point\_cloud, nb\_points, radius) - функция принимает на вход облако точек,nb\_points, определяющий количество точек, и radius, определяющий радиус для выявления выбросов. Используется для удаления выбросов из облака точек на основе радиуса.

downsample\_cloud(point\_cloud, voxel\_size) - функция принимает на вход облако точек, voxel\_size, определяющий размер вокселя. Используется для уменьшения количества точек в облаке с помощью воксельной решетки.

filter\_fun\_simple(point\_cloud, original\_size) - функция принимает на вход облако точек и количество точек, которые должны быть сохранены в упрощенной сетке. Используется для упрощения сетки, созданной из облака точек, с помощью простого алгоритма сглаживания.  
 filter\_fun\_laplacian(point\_cloud, original\_size) - функция принимает на вход облако точек и количество точек, которые должны быть сохранены в сглаженной сетке. Используется для сглаживания сетки, созданной из облака точек, с помощью алгоритма Лапласа.  
 filter\_fun\_taubin(point\_cloud, original\_size) - функция принимает на вход облако точек и количество точек, которые должны быть сохранены в сглаженной сетке. Используется для сглаживания сетки, созданной из облака точек, с помощью алгоритма Таубина.

analyze\_methods(point\_cloud) -функция принимает на вход облако точек. Используется для вывода справочной информации, о количестве проблемных точек и времени, затраченного на методы, используемые для устранения ошибок в облаке точек.

add\_noise\_in\_point\_cloud(filename) - функция принимает на вход путь к файлу, в котором лежит облако точек. Используется для добавления случайного шума в облако точек, чтобы наглядно наблюдать за действием вышеизложенных методов по устранению ошибок.

## 

## 4 ПРИМЕНЕНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ НА ПРАКТИКЕ

### 4.1 Удаление выбросов на основе статистического анализа

Для применения метода удаления выбросов на основе статического анализа был использован метод статистической фильтрации, библиотеки Open3D. В качестве параметров учитывалось 20 соседних точек и коэффициент стандартного отклонения равный 2.

Данные метод включает в себя следующие шаги:

1. Поиск ближайших соседей для каждой точки. Для увеличения скорости поиска было использовано KD-дерево.
2. Для каждой точки вычисляется среднее расстояние до ее ближайших соседей.
3. Вычисляется среднее значение и стандартное отклонение средних расстояний. Среднее значение средних расстояний всех точек определяется как:

где n – общее количество точек в облаке точек, – среднее расстояние для i-ой точки. Стандартное отклонение средних расстояний всех точек определяется как:

1. Вычисляется порог на основе среднего значения и стандартного отклонения. Порог задается как:

где m – коэффициент стандартного отклонения.

1. Удаляются точки считающимися выбросами. Точка считается выбросом если выполняется неравенство:



Рисунок 1 – Первоначальное облако точек



Рисунок 2 – Полученное облако точек

Результаты до и после применения метода представлены на рис. 1 и рис. 2 соответственно. Фильтрация выбросов позволила уменьшить количество точек с 17017 до 16085, что связано с удалением выбросов. Но при использовании другого облака точек было выявлено, что были удалены некоторые важные точки, которые не являлись выбросами. Пример с удалением важных точек в облаке точек представлен на рис. 3 и рис. 4.

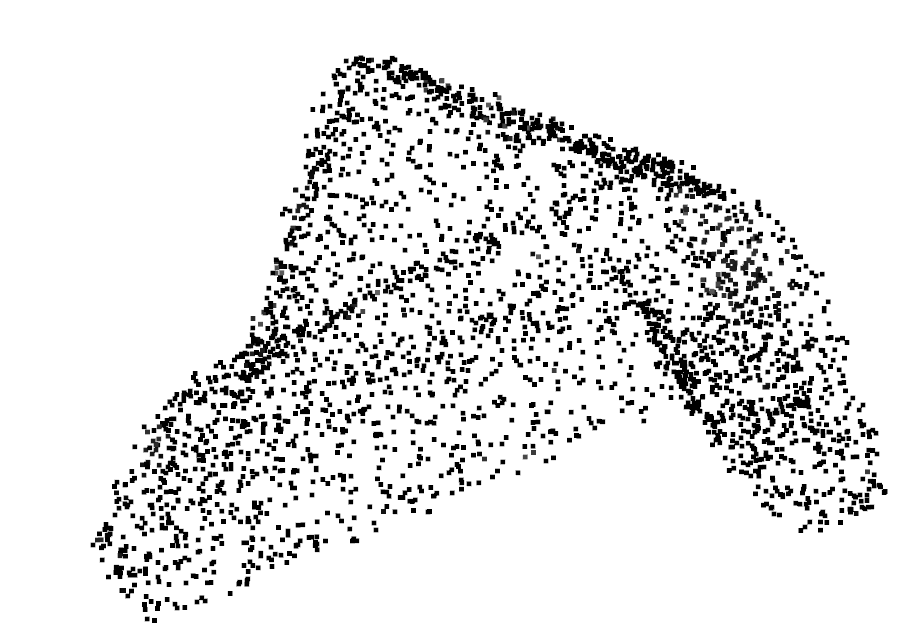


Рисунок 3 – Первоначальное облако точек



Рисунок 4 – Полученное облако точек

Метод статистического удаления выбросов продемонстрировал высокую эффективность в удалении шумов, но недостатком текущего метода является его жесткость в выборе пороговых значений и недостаточное внимание к локальным особенностям данных. Хотя удаление точек за пределами выбранных порогов помогает сформировать более плотное и чистое облако, оно может привести к потере тонких деталей и структурных особенностей объекта. Оставшиеся точки представляют собой упрощенную версию объекта, что может повлиять на точность дальнейших вычислений и анализа.

### 4.2 Удаление выбросов на основе радиуса

Для применения метода удаления выбросов на основе радиуса был использован метод из библиотеки Open3D. В качестве параметров учитывалось минимум 15 соседних точек и радиус равный 3.

Данные метод включает в себя следующие шаги:

1. Поиск всех соседей для каждой точки, которые находятся в радиусе r. Для увеличения скорости поиска было использовано KD-дерево. Функция поиска расстояния:

Поиск соседей:

1. Для каждой точки вычисляется количество соседей в заданном радиусе.
2. Удаляются точки считающимися выбросами. Точка считается выбросом если выполняется неравенство:

где k – пороговое значение количеств соседних точек.

### 

Рисунок 5 – Первоначальное облако точек



Рисунок 6 – Полученное облако точек

Результаты до и после применения метода представлены на рис. 5 и рис. 6 соответственно. Фильтрация выбросов позволила уменьшить количество точек с 17017 до 15408, что связано с удалением выбросов. Визуальный анализ показал, что большая часть удаленных точек действительно являлась шумом, но при изменении параметров метода может произойти удаление значимых данных, как показано на рис. 7.



Рисунок 7 – Полученное облако точек с изменениями параметров метода

### 

Применение метода удаления выбросов на основе радиуса позволило существенно улучшить качество облака точек, удалив большую часть шумов. Тем не менее, необходимо внимательно выбирать параметры метода, чтобы избежать удаления значимых данных.

### 4.3 Использование фильтра среднего значения

### Для применения метода фильтрации среднего значения был использован метод из библиотеки Open3D. В качестве параметров было взято количество итераций равное 1.

Данные метод включает в себя следующие шаги:

1. Поиск ближайших k соседей для каждой точки.
2. Вычисление средних значений координат этих k соседних точек для каждой точки. Вычисление средних значений координат:
3. Обновление координат каждой точки:



Рисунок 8 – Первоначальное облако точек



Рисунок 9 – Полученное облако точек

Результаты до и после применения метода представлены на рис. 8 и рис. 9 соответственно. Визуальный анализ показал, что большая часть шума была сглажена. Однако при увеличении количества итераций может произойти более сильное сглаживание, что может деформировать мелкие детали облака точек, как показано на рис. 10.

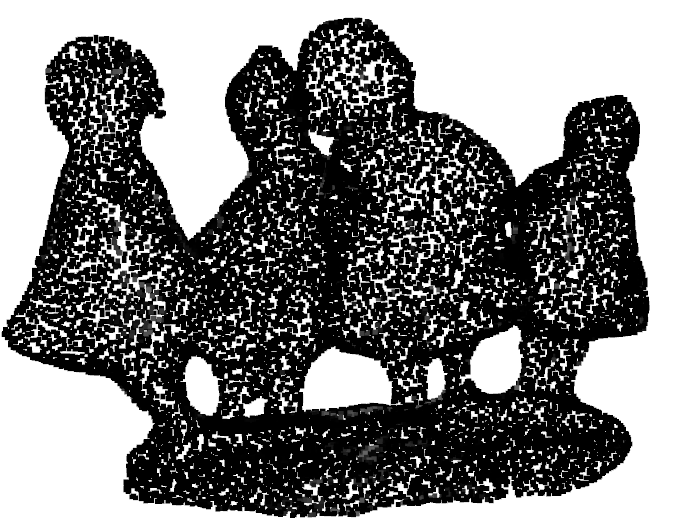


Рисунок 10 – Полученное облако точек с увеличением количества итераций

Метод простого усреднения показал свою эффективность для уменьшения шума в облаке точек. Однако во избежание потери важных данных, особенно при работе с объектами сложный формы, стоит использовать данный метод с осторожностью.

### 4.4 Использование фильтра Лапласа

### Для применения фильтра Лапласа был использован метод из библиотеки Open3D. В качестве параметров были взяты количество итераций равное 10 и коэффициент сглаживания *λ* = 0.2.

Данные метод включает в себя следующие шаги:

1. Поиск ближайших k соседей для каждой точки.
2. Вычисление лапласового оператора этих k соседних точек для каждой точки. Вычисление лапласового оператора:

где – координаты текущей точки, – координаты соседней точки.

Этот оператор вычисляет среднее смещение точки ​ относительно её соседей. Он показывает, насколько сильно точка отклоняется от локальной средней позиции.

1. Обновление координат каждой точки:

Значение *λ* выбирается так, чтобы контролировать степень сглаживания. Так при малых значениях *λ* сглаживание будет слабым, а при больших значениях – более сильным.



Рисунок 11 – Первоначальное облако точек

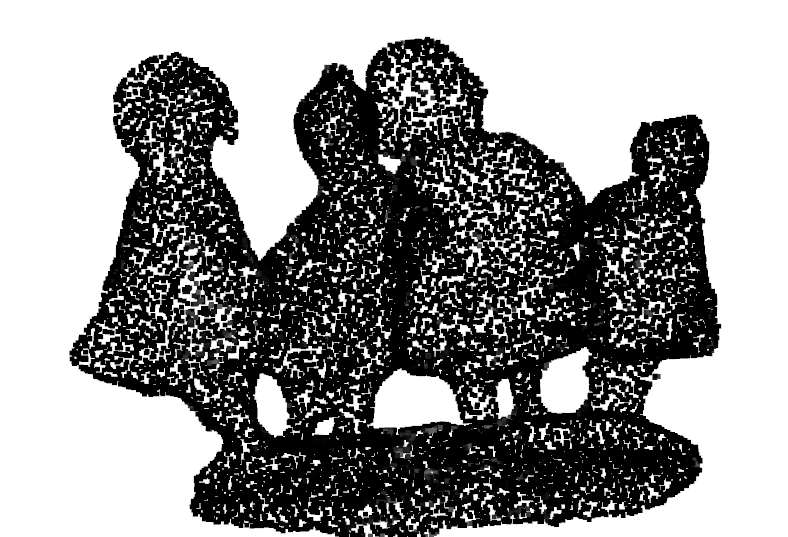


Рисунок 22 – Полученное облако точек

Результаты до и после применения метода представлены на рис. 11 и рис. 12 соответственно. Визуальный анализ показал, что большая часть шума была сглажена. Однако при увеличении количества итераций может произойти более сильное сглаживание, что может привести к утоньшению фигуры рис. 13.



Рисунок 33 – Полученное облако точек

Фильтр Лапласа показал свою эффективность для уменьшения шума в облаке точек. Однако во избежание потери важных данных, необходима тщательная настройка параметров.

### 4.5 Использование фильтра Таубина

### Для применения фильтра Таубина был использован метод из библиотеки Open3D. В качестве параметров были взяты количество итераций равное 10, коэффициент сглаживания *λ* = 0.2 и коэффициент обратного сглаживания *μ* = -0.2.

Данные метод включает в себя следующие шаги:

1. Поиск ближайших k соседей для каждой точки.
2. Вычисление лапласового оператора этих k соседних точек для каждой точки. Вычисление лапласового оператора:

где – координаты текущей точки, – координаты соседней точки.

1. Первое сглаживание выполняется аналогично фильтрации Лапласа:
2. Обратное сглаживание выполняется с отрицательным коэффициентом *μ*:



Рисунок 14 – Первоначальное облако точек

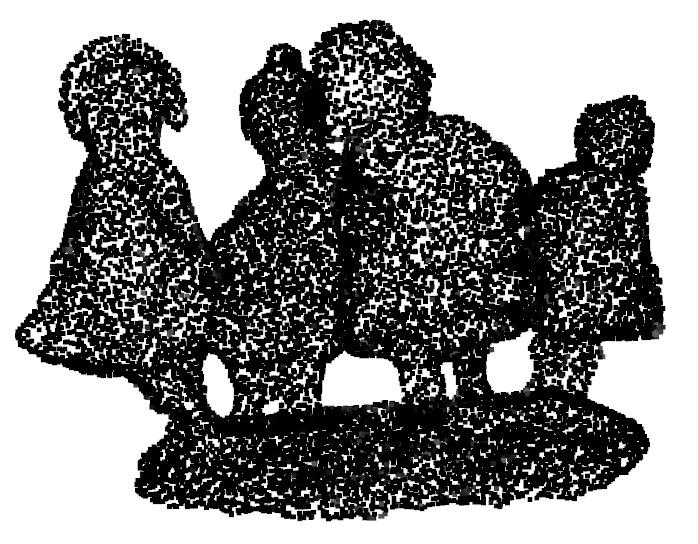


Рисунок 45 – Полученное облако точек после

Результаты до и после применения метода представлены на рис. 14 и рис. 15 соответственно. Визуальный анализ показал, что большая часть шума была сглажена. Фильтрация Таубина включает две итерации сглаживания: сглаживание с положительным коэффициентом и обратное сглаживание с отрицательным коэффициентом. Это предотвращает усадку модели.

### 4.6 Сравнение используемых методов

В табл. 1 представлено сравнение методов, их преимущества и недостатки, а также область применения.

Таблица 1 – Сравнительная таблица методов

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Преимущества** | **Недостатки** | **Области применения** |
| Статистическое удаление выбросов | Эффективен при явных выбросах | Возможность удаления важных данных | Предварительная обработка данных |
| Удаление выбросов на основе радиуса | Простота настройки | Неэффективен при неоднородной плотности | Предварительная обработка данных |
| Фильтр среднего значения | Простота реализации | Возможность сглаживания деталей | Мягкое сглаживание |
| Фильтр Лапласа | Сохранение геометрии | Вычислительно затратный | Сложные поверхности |
| Фильтр Таубина | Удаление шума без усадки | Настройка параметров | Высокоточные модели |

Как видно из табл. 1 каждый из рассмотренных методов имеет свои преимущества и недостатки. Выбор подходящего метода зависит от конкретных требований задачи и характеристик исходных данных. В практике, часто используется комбинация нескольких методов для достижения оптимальных результатов.

## 5 БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

### 5.1 Основные положения

Выпускная квалификационная работа посвящена исследованию возможностей автоматического обнаружения и устранения ошибок трехмерной модели в задаче трехмерной реконструкции на языке программирования Python. Результаты работы демонстрируют эффективность применения библиотеки данного языка для решения задач обработки и анализа трехмерных данных.

Разработка программного обеспечения для обработки трехмерных моделей требует создания надежного и удобного в использовании кода, и здесь важную роль играет интегрированная среда разработки (IDE). В данном проекте для написания кода использовался Python в сочетании с IDE PyCharm, которая обеспечивает высокий уровень эргономики и удобства для разработчиков.

### 5.2 Эргономика среды разработки

Для проведения оценки PyCharm, как удобной и практичной среды разработки, был взят ГОСТ Р ИСО 9241-110-2016, в котором описаны основополагающие принципы эргономики в контексте взаимодействия человека и компьютерной системы. Далее в работе проводится анализ соответствия PyCharm этим принципам и оценке того, насколько эффективно PyCharm удовлетворяет потребности разработчиков, обеспечивая удобство и эффективность в работе.

#### 5.2.1 Приемлемость организации диалога для выполнения производственного задания

PyCharm обеспечивает эффективную организацию диалога между пользователем и системой, облегчая выполнение задач разработки. IDE предлагает четкую и логичную структуру интерфейса с стандартными элементами, такими как меню, панели инструментов и консоль. На рис. 16 показан интерфейс PyCharm на котором можно заметить основные элементы, включая верхнюю панель меню, панель инструментов, редактор кода, панель навигации и правую панель с дополнительной информацией. PyCharm также позволяет настраивать интерфейс в соответствии с предпочтениями разработчика, обеспечивая комфортную и эффективную рабочую среду.

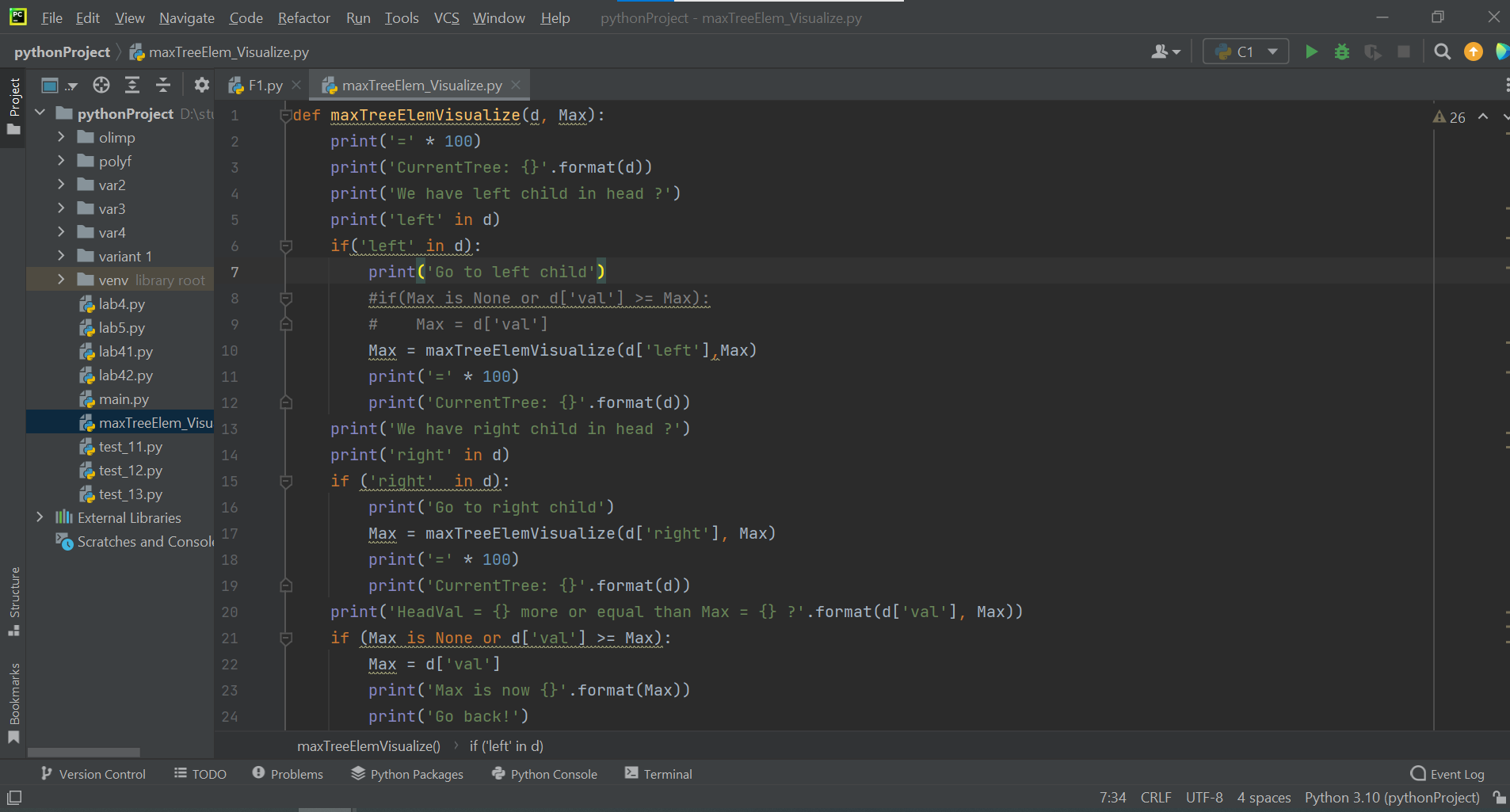


Рисунок 16 – Интерфейс PyCharm

#### 5.2.2 Информативность

PyCharm придает большое значение информативности, обеспечивая пользователей подробными и своевременными сведениями, необходимыми для эффективной работы. IDE предлагает четкие и понятные сообщения об ошибках, дополненные полезными предложениями по их устранению, как показано на рис. 17. Подсказка кода, представленная на рис. 18, служит надежным руководством для написания точного и эффективного кода. Кроме того, встроенная документация, доступная непосредственно в IDE, как показано на рис. 19, обеспечивает быстрый и удобный доступ к справочной информации.

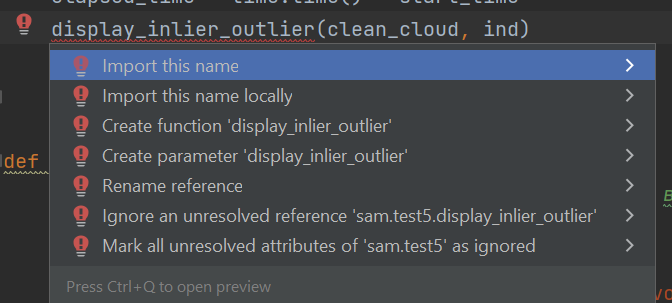


Рисунок 17 – Предложения по устранению ошибки

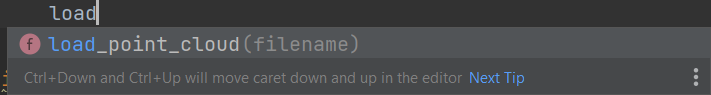


Рисунок 18 – Подсказка кода

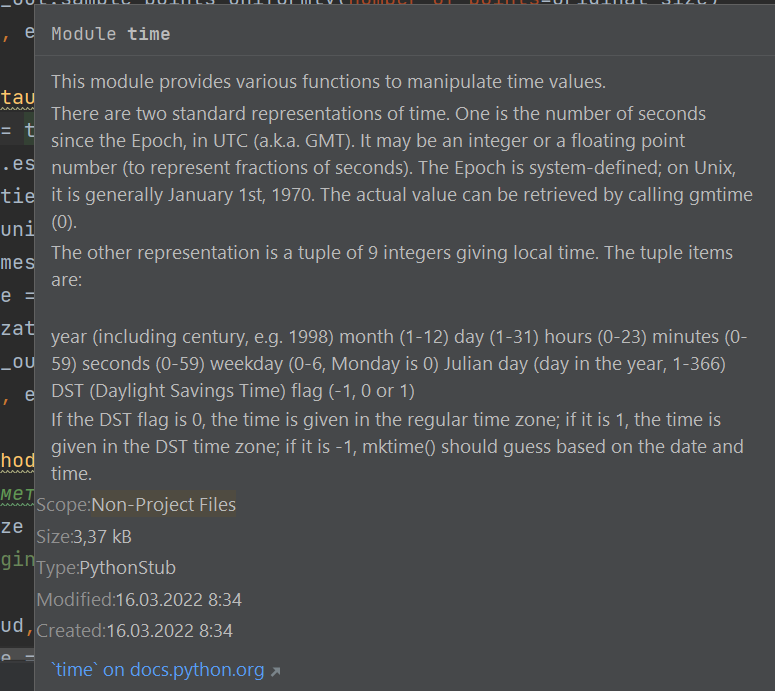


Рисунок 19 – Пример встроенной документации

Помимо текстовых сообщений, PyCharm использует визуальные индикаторы для передачи сведений о коде. Цветовая маркировка выделяет различные элементы, такие как ключевые слова, переменные и комментарии, облегчая чтение и понимание программы. PyCharm также визуально отслеживает изменения в файлах, состояние отладки и результаты тестирования, предоставляя разработчикам наглядную информацию о состоянии их проекта.

#### 

#### 5.2.3 Соответствие ожиданиям пользователей

PyCharm уделяет особое внимание соответствию ожиданиям пользователей, создавая знакомую и интуитивно понятную среду разработки. IDE тщательно спроектирована, чтобы обеспечить эффективное и комфортное взаимодействие с пользователем. PyCharm использует стандартные элементы интерфейса, такие как панели меню, инструментов и навигации, которые пользователи могут легко распознать и использовать. Это сокращает время, необходимое для освоения IDE, позволяя разработчикам сосредоточиться непосредственно на написании кода.

Рис. 20 демонстрирует панель меню PyCharm, которая соответствует стандартной структуре, знакомой большинству пользователей. Меню содержит такие стандартные пункты, как "File", "Edit", "View" и другие, обеспечивая быстрый доступ к различным командам и настройкам.



Рисунок 20 – Панель меню

Кроме того, PyCharm понимает, что каждый разработчик уникален, и предлагает широкие возможности для настройки интерфейса в соответствии с личными предпочтениями. Пользователи могут адаптировать свою рабочую среду, выбирая между различными цветовыми схемами, изменяя размер и тип шрифта или перемещая и скрывая панели. Рис. 21 иллюстрирует настраиваемость PyCharm, демонстрируя выбор цветовой схемы, которая может улучшить читаемость и снизить утомляемость глаз.

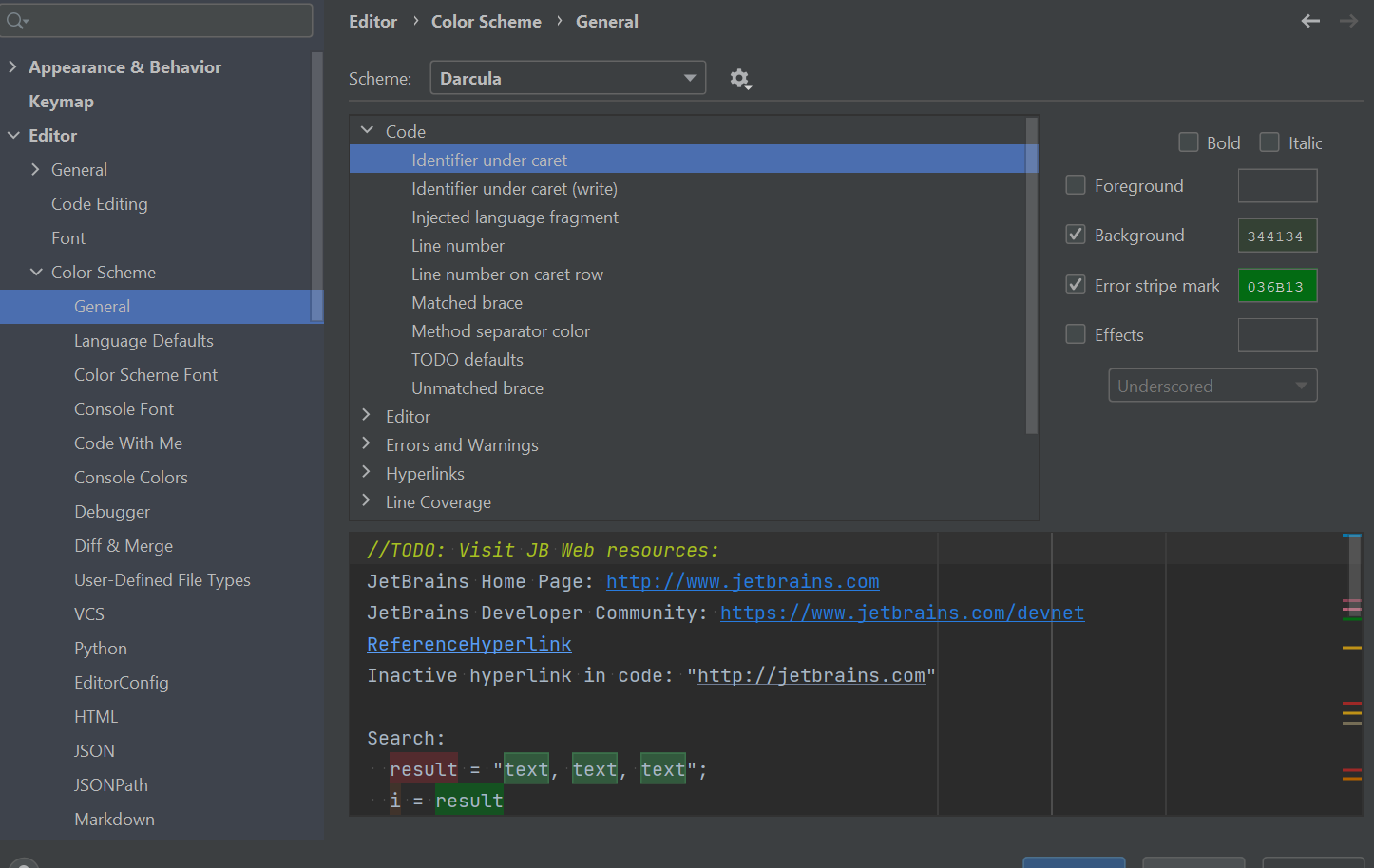


Рисунок 21 – Пример выбора цветовой схемы

PyCharm также поддерживает различные операционные системы, включая Windows, macOS и Linux, обеспечивая знакомую среду независимо от платформы. IDE предоставляет аналогичный набор функций и интерфейс на разных ОС, что облегчает работу разработчиков, использующих несколько систем.

#### 5.2.4 Пригодность для обучения

PyCharm предлагает подробную документацию и различные функции, которые делают его пригодным для обучения. Интегрированный отладчик позволяет пошагово выполнять код и отслеживать переменные, что облегчает процесс обучения и понимания структуры программы. Кроме того, PyCharm предлагает подсказки кода, автодополнение и контекстно-зависимую помощь, что ускоряет процесс обучения и делает IDE более доступной для пользователей с разным уровнем опыта.

#### 

#### 5.2.5 Контролируемость

PyCharm обеспечивает высокую контролируемость, предоставляя пользователям различные инструменты для управления процессом разработки. IDE позволяет настраивать параметры проекта, управлять зависимостями и интегрироваться с системами контроля версий. PyCharm автоматически сохраняет проекты, предотвращая потерю работы, и предлагает функции восстановления в случае непредвиденных сбоев. Разработчики могут легко отслеживать изменения в файлах, настраивать точки останова для отладки и получать визуальные индикаторы результатов тестирования. Таким образом, PyCharm предоставляет пользователям полный контроль над процессом разработки, обеспечивая эффективное управление проектами и сотрудничество в команде.

#### 5.2.6 Устойчивость к ошибкам

PyCharm демонстрирует высокую устойчивость к ошибкам, обеспечивая стабильную и надежную среду разработки. Как упоминалось ранее IDE автоматически сохраняет проекты, предотвращая потерю работы в случае сбоя системы или неожиданного закрытия программы. PyCharm также предлагает функции восстановления, позволяющие возвращаться к предыдущим версиям кода и исправлять ошибки. Разработчики могут настроить точки восстановления, создавать резервные копии проектов и использовать системы контроля версий для отслеживания изменений.

#### 5.2.7 Адаптируемость к особенностям пользователей

Как упоминалось ранее, пользователь может настроить IDE под себя: изменить шрифт, поменять цвет фона, изменить шрифт и т.д..

### 5.3 Эргономика программ, написанных в PyCharm

PyCharm обеспечивает эргономику программ, написанных с его помощью, предоставляя инструменты для создания интуитивно понятного интерфейса. Интеграция с библиотекой Open3D упростила работу с трехмерными моделями в моей выпускной работе. PyCharm ускорил разработку с помощью автодополнения и подсказок, а интегрированный отладчик позволил эффективно выявлять и устранять ошибки. IDE также обеспечивает соответствие программ стандартам эргономики взаимодействия "человек-система", что привело к созданию более удобной и эффективной программы.

### 5.4 Выводы

Анализ соответствия PyCharm принципам эргономики, изложенным в ГОСТ Р ИСО 9241-110-2016, показывает, что IDE эффективно удовлетворяет потребности разработчиков, обеспечивая удобство и эффективность в работе. PyCharm соответствует основополагающим принципам, включая приемлемость организации диалога, информативность, соответствие ожиданиям пользователей, пригодность для обучения, контролируемость, устойчивость к ошибкам и адаптируемость к особенностям пользователя. IDE обеспечивает четкую структуру кода, полезные подсказки и контекстно-зависимую помощь.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы были решены следующие задачи:

1)Проанализировать существующие методы и подходы к трехмерной реконструкции и обработке облаков точек.

Существующие методы и подходы к трехмерной реконструкции и обработке облаков точек были проанализированы, с особым вниманием к проблемам и ограничениям, связанным с неоднородностью выборки, шумом и выбросами данных. Исследование охватывает широкий спектр стратегий, от простых предположений о гладкости до сложных адаптивных моделей. Влияние технологических инноваций, таких как улучшенное сканирование и обработка данных, также рассматривается, подчеркивая непрерывное развитие в этой области.

2)Рассмотреть различные типы ошибок, возникающих в облаках точек.

В ходе работы были рассмотрены возможные ошибки, возникающие в облаках точек и причины их возникновения.

3)Разработать методы для автоматического устранения и обнаруженных ошибок в облаках точек.

Были разработаны методы для автоматического устранения ошибок в облаке точек на основе статического анализа и радиуса, а также методы, использующие фильтрацию (среднего значения, Лапласа, Таубина), для сглаживания облака точек.

4)Провести эксперименты для оценки эффективности разработанных методов на реальных исходных данных.

Были проведены эксперименты по 5 методам. На основе этих данных можно сделать следующие выводы:

1. Методы удаления точек на основе статистического анализа и радиуса меньше влияют на форму облака точек и избавляются от выбросов. Занимают мало времени, но избавляются от выбросов хуже, чем методы фильтрации, а также могут привести к удалению мелких деталей фигуры.
2. Методы фильтрации хорошо избавляются от выбросов сглаживая поверхность, но занимают больше времени и влияют на форму.
3. Для методов фильтрации можно выбрать количество итераций их применения. Для фильтра среднего значения оптимальное количество итераций равно 5, в большем количестве нет смысла. Для фильтра Лапласа оптимальное количество итераций равно 10, при большем количестве облако точек будет сжиматься и потеряет свои первоначальный размер и форму. Для фильтра Таубина нет такой проблема как у фильтра Лапласа, ведь он может предотвратить усадку сетки, что позволит сохранить форму облака точек.

## 

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Bolle R. M., Vemuri B. C. On three-dimensional surface reconstruction methods //IEEE Transactions on Pattern Analysis & Machine Intelligence. – 1991. – Т. 13. – №. 01. – С. 1-13.
2. Alexa M., Adamson A. On Normals and Projection Operators for Surfaces Defined by Point Sets //PBG. – 2004. – С. 149-155.
3. МЕТОД РЕКОНСТРУКЦИИ ТРЕХМЕРНЫХ СЦЕН, ОСНОВАННЫЙ НА ПРИМЕНЕНИИ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ, ФИЛЬТРАЦИИ ПО ДИСТАНЦИИ И С ПОМОЩЬЮ "ОКТОДЕРЕВА" / Ю. В. Дубенко, Y. V. Dubenko, Е. Е. Дышкант [и др.] // Известия высших учебных заведений. Поволжский регион. Технические науки. — 2021. — № 4 (60). — С. 43-54. — ISSN 2072-3059. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/journal/issue/342341 (дата обращения: 13.05.2024). — Режим доступа: для авториз. пользователей.
4. Berger M. et al. A survey of surface reconstruction from point clouds //Computer graphics forum. – 2017. – Т. 36. – №. 1. – С. 301-329.
5. Huang Z. et al. Surface reconstruction from point clouds: A survey and a benchmark //arXiv preprint arXiv:2205.02413. – 2022.
6. Khatamian A., Arabnia H. R. Survey on 3D surface reconstruction //Journal of Information Processing Systems. – 2016. – Т. 12. – №. 3. – С. 338-357.
7. You C. C. et al. A survey on surface reconstruction techniques for structured and unstructured data //2020 IEEE Conference on Open Systems (ICOS). – IEEE, 2020. – С. 37-42.
8. Berger M. et al. A benchmark for surface reconstruction //ACM Transactions on Graphics (TOG). – 2013. – Т. 32. – №. 2. – С. 1-17.
9. Carr J. C. et al. Reconstruction and representation of 3D objects with radial basis functions //Proceedings of the 28th annual conference on Computer graphics and interactive techniques. – 2001. – С. 67-76.
10. Chauve A. L., Labatut P., Pons J. P. Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data //2010 IEEE computer society conference on computer vision and pattern recognition. – IEEE, 2010. – С. 1261-1268.
11. Hornung A., Kobbelt L. Robust reconstruction of watertight 3 d models from non-uniformly sampled point clouds without normal information //Symposium on geometry processing. – 2006. – Т. 41. – С. 50.
12. Jiang H. et al. Neighborhood-based neural implicit reconstruction from point clouds //2021 International Conference on 3D Vision (3DV). – IEEE, 2021. – С. 1259-1268.
13. Li C. et al. Low-light image and video enhancement using deep learning: A survey //IEEE transactions on pattern analysis and machine intelligence. – 2021. – Т. 44. – №. 12. – С. 9396-9416.
14. Chibane J., Alldieck T., Pons-Moll G. Implicit functions in feature space for 3d shape reconstruction and completion //Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. – 2020. – С. 6970-6981.
15. Zhang J. et al. A review of deep learning-based semantic segmentation for point cloud //IEEE access. – 2019. – Т. 7. – С. 179118-179133.
16. Zhou Q. Y., Park J., Koltun V. Open3D: A modern library for 3D data processing //arXiv preprint arXiv:1801.09847. – 2018.
17. Alexa M. et al. Computing and rendering point set surfaces //IEEE Transactions on visualization and computer graphics. – 2003. – Т. 9. – №. 1. – С. 3-15.