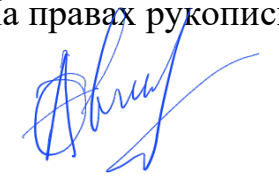
Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(МАИ)

На правах рукописи



Александров Александр Александрович

ИНФОРМАЦИОННО-ИЗМЕРИТЕЛЬНАЯ СИСТЕМА ИДЕНТИФИКАЦИИ ДЕТАЛЕЙ НА КОНВЕЙЕРНОЙ ЛИНИИ НА ОСНОВЕ ЛАЗЕРНОГО 3D-СКАНИРОВАНИЯ

2.2.11. Информационно-измерительные и управляющие системы

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель:
кандидат технических наук, доцент
Васильев Федор Владимирович

Москва 2026

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность работы.

В настоящее время наблюдается тенденция к повышению эффективности производства за счет его цифровизации. В производственные процессы внедряются комплексные цифровые системы, направленные на повышение качества изготавливаемой продукции, автоматизацию или оптимизацию производственных процессов. Системы автоматического контроля на различных стадиях производственного цикла являются неотъемлемой частью таких систем.

Особенно актуальной проблема внедрения систем автоматического контроля становится на авиастроительных и машиностроительных производствах, так как эти производства используют огромную номенклатуру деталей. В связи с повышением требований к качеству продукции и скорости работы машиностроительных и авиационных производств появляется потребность в идентификации типов деталей на производственной линии, в том числе на конвейере. Проблема обостряется, тем, что на многих производственных участках нет возможности наносить метки на детали, при этом детали могут иметь схожую форму, но разные размеры, что обуславливает необходимость использования специализированных методик идентификации.

Возможность автоматической идентификации деталей на производственной линии, позволит снизить количество ошибок, обусловленных влиянием человеческого фактора и сократить время производства продукции. К текущему моменту предложено большое количество методик идентификации деталей по облакам точек. Большой вклад в область научного и практического обеспечения методик идентификаций деталей по облакам точек внесли советские и российские ученые: А.Л. Павлов, С.Ю. Желтов, Е.Н. Сосенушкин, Ю.Б. Блохинов и др.; а также зарубежные ученые Й. Гуо (Y. Guo), Ш. Чанг (H. Zhang), Р.Б. Русу (R.B. Rusu) и др.

Однако, несмотря на обилие существующих методик, из-за специфики условий эксплуатации, универсальные решения могут не обеспечивать приемлемое качество идентификации. Поэтому разработка методик идентификации для работы в конкретных условиях производственного процесса остается актуальной.

Целью диссертационной работы является улучшение характеристик информационно-измерительных систем контроля за счет повышения точности идентификации деталей по облаку точек, получаемому с помощью лазерного 3D-сканирования.

Для достижения поставленной цели требуется решить следующие **задачи**:

1. Разработать инструмент создания виртуальных облаков точек, эквивалентных полученным с систем лазерного сканирования, для создания набора облаков точек тестовых деталей, который будет использоваться в процессе исследования.
2. Разработать алгоритм подготовки облаков точек, полученных с системы лазерного 3D-сканирования.
3. Разработать методику идентификации деталей сложной формы по облакам точек, полученных с нескольких ракурсов.
4. Провести экспериментальное исследование для определения характеристик разработанной системы.

Объектом исследования является информационно-измерительная система для идентификации объекта по облаку точек, полученного с устройства лазерного 3D-сканирования.

Предметом исследования являются методики и алгоритмы идентификации объектов по облакам точек, методики и алгоритмы подготовки облаков точек.

Научная новизна работы:

1. Разработана методика эмуляции системы лазерного сканирования детали, движущейся по конвейеру, позволяющая воспроизводить структуру облака точек реального сканирования, отличающаяся возможностью воспроизведения ошибок, связанных с обработкой изображений луча лазера на поверхности детали без необходимости отрисовки всей сцены сканирования.
2. Разработан новый дескриптор Laser Scanner Feature Histogram (LSFH), предназначенный для информационно-измерительной системы лазерного 3D-сканирования, отличающийся расчетом средней точки по параметрам объектно-ориентированного ограничивающего параллелепипеда и вводом дополнительной компоненты дескриптора, основанной на геометрических параметрах детали, и позволяющий осуществлять идентификацию детали сложной формы по набору облаков точек, полученных с разных ракурсов.
3. Разработана методика формирования эталонных дескрипторов по цифровым моделям деталей для информационно-измерительных систем контроля на конвейерных линиях, основанная на построении карты дескрипторов LSFH и её последующей кластеризации, отличающаяся адаптивным выбором числа значимых дескрипторов в зависимости от геометрической сложности детали и отсутствием необходимости проведения натурного сканирования эталонных образцов.
4. Предложена методика идентификации детали в составе информационно-измерительной системы на основе голосования по нескольким дескрипторам LSFH, вычисленным для двух и более ракурсов сканирования, отличающаяся совместным использованием метрик соответствия дескриптора и габаритных параметров детали и обеспечивающая повышение точности идентификации до уровня, близкого к 0,996.

Теоретическая значимость работы заключается в том, что разработанная методика формирования эталонных дескрипторов и методика построения дескриптора, а также содержащиеся в них математические модели образуют научные основы для разработки комплексных методик идентификации деталей по облаку точек, что способствует повышению точности идентификации.

Практическая значимость работы:

1. Разработанная методика эмуляции системы лазерного сканирования позволяет получить облако точек, отличающееся от облака точек с реального устройства сканирования не более чем на 10%, что позволяет подобрать оптимальную конфигурацию параметров в алгоритмах путем тестирования на большом количестве различных деталей и различных конфигурациях систем сканирования без необходимости проведения реального сканирования.
2. Разработанный дескриптор, адаптированный под систему контроля с использованием устройств лазерного сканирования, а также методика идентификации с использованием разработанного дескриптора позволяют идентифицировать детали различного размера и сложной формы по облаку точек, с точностью 0,996 на тестовом наборе деталей с возможностью повышения точности путем увеличения ракурсов сканирования.
3. Разработанная методика формирования эталонных дескрипторов позволяет сократить число эталонных дескрипторов, используемых в процессе идентификации, что сокращает время идентификации детали.
4. Разработанное программное обеспечение для эмуляции лазерного сканирования, позволяющее получить облака точек по заданным параметрам устройства сканирования, а также программное обеспечение для идентификации детали, которое позволяет идентифицировать деталь по ее облаку точек, обеспечивают

возможность внедрения разработанных методик для повышения точности идентификации.

Методы исследования. Проведенное исследование опирается на методы математического моделирования, статистические методы, методы компьютерного зрения, методы обработки изображений, методы линейной алгебры.

Положения, выносимые на защиту:

— разработанная методика эмуляции системы лазерного 3D-сканирования позволяет получать сымитированные облака точек без проведения натурных измерений с ошибкой не более 10% за счет учета ширины видимого лазерного луча на поверхности детали;

— разработанный дескриптор позволяет увеличить точность идентификации деталей до 5 раз на тестовом наборе за счет учета геометрии детали при поиске средней точки, а также ввода дополнительной компоненты дескриптора, основанной на геометрических параметрах детали;

— разработанная методика формирования эталонных дескрипторов по цифровым моделям деталей позволяет сократить время идентификации более чем в 40 раз за счет нового алгоритма выбора значимых ракурсов в зависимости от сложности геометрии детали, путем представления группы похожих дескрипторов одним эталонным;

— разработанная методика идентификации детали с помощью голосования позволяет обеспечить точность идентификации 0,996 за счет использования вновь введенных дескрипторов, полученных с разных устройств сканирования, и геометрических параметров детали.

Достоверность полученных результатов подтверждается обоснованностью используемого математического аппарата и допущений в разработанных методиках, а также экспериментальной проверкой точности идентификации разработанных методик.

Внедрение результатов работы. Результаты работы внедрены в ООО «ЦПР РТСофт», использованы в Московском авиационном институте при выполнении государственного задания Минобрнауки России.

Апробация работы. Основные научные и прикладные результаты докладывались и обсуждались на: 20-ой, 22-ей Международных конференциях «Авиация и космонавтика (2021, 2023; Москва, МАИ); LI, LII Международных конференциях «Гагаринские чтения» (2025, 2026; Москва, МАИ).

Публикации. По теме диссертации опубликовано 10 работ [1-10], в том числе 5 в рецензируемых изданиях Перечня ВАК РФ [1, 2, 3, 4, 5] по специальности 2.2.11, 1 свидетельство о регистрации программы для ЭВМ [9], 1 патент на полезную модель [10].

Структура и объем работы. Работа состоит из введения, четырех глав с выводами, заключения, списка литературы, четырех приложений, изложенных на 169 страницах, в том числе 142 страницах — основной части, 4 приложениях на 27 страницах. Работа содержит 52 рисунка и 19 таблиц. Список литературы содержит 96 наименований.

Глава 1. Общий анализ информационно-измерительной системы идентификации деталей

1.1. Структура информационно-измерительной системы идентификации деталей

Типовая структура информационно измерительной системы идентификации деталей представлена на рисунке 1 и состоит из следующих компонентов:

— устройств сканирования;

— системы идентификации.

Устройства сканирования осуществляют сканирование движущейся по конвейеру детали и выполняют построение облака точек этой детали. Облако точек, полученное с описанных устройств, подается на вход системы идентификации, которая осуществляет идентификацию детали. Результатом идентификации является уникальный идентификатор 3D-модели детали.

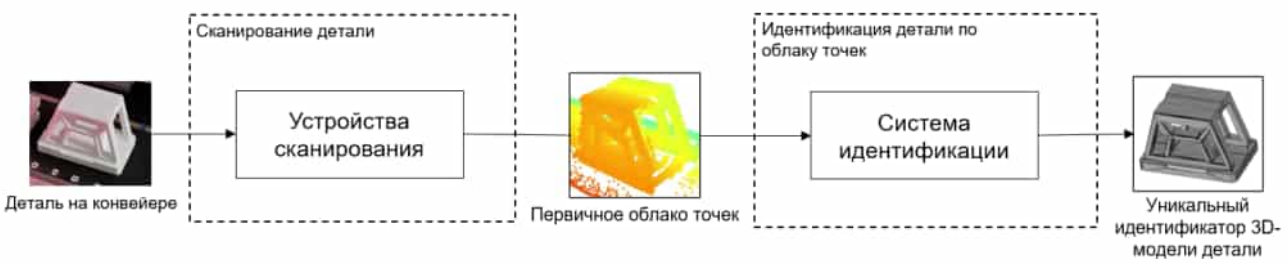


Рисунок 1 — Общая схема структуры информационно измерительной системы идентификации

Задача идентификации с помощью описанной системы формулируется следующим образом. Задана база данных эталонных цифровых моделей деталей, которые необходимо идентифицировать. По конвейеру движутся детали, с этих деталей с нескольких ракурсов снимаются облака точек с помощью устройств лазерного 3D сканирования. По облаку точек, полученному с устройств сканирования, необходимо определить какая деталь движется по конвейеру. Схема процесса идентификации изображена на рисунке 2.

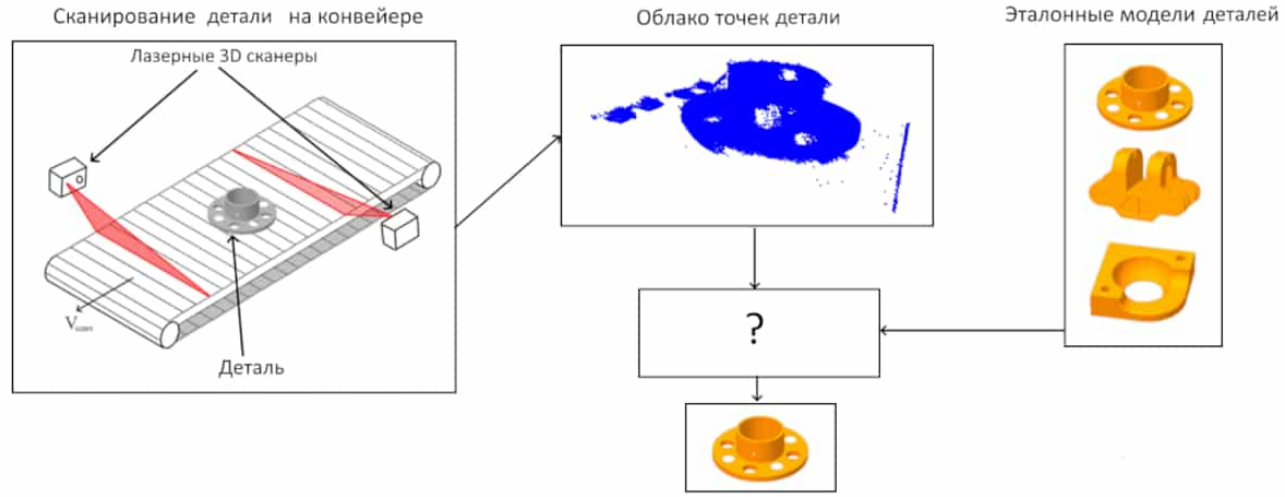


Рисунок 2 — Общая схема процесса идентификации

В работе рассматривается система идентификации деталей на конвейерной линии по облаку точек, полученному в результате лазерного сканирования, поэтому в главе будет проведен анализ существующих методов идентификации деталей по облакам точек и методов подготовки облаков точек для идентификации. Так как выбор метода подготовки облака зависит от используемых методов идентификации, сначала будет проведен анализ существующих методов идентификации, выделены их преимущества и недостатки применимо к описанной схеме идентификации деталей. После этого будет проведен анализ существующих методов подготовки облаков точек для работы с рассмотренными методами идентификации детали.

1.2. Анализ существующих методов идентификации объекта по облакам точек

Идентификация объекта по облакам точек широко представлена в научных работах и делится на две большие категории [11]: идентификация с применением методов глубокого обучения [12, 13, 14, 15] и с использованием классических методов (используются дескрипторы созданные вручную) [16, 17]. Классификация методов идентификации детали по облаку точек изображена на рисунке 3.

Рисунок 3 — Классификация методов идентификации деталей по облаку точек

Методы с использованием глубокого обучения зачастую используют подходы из классических методов для повышения точности идентификации. Методы идентификации с помощью методов глубокого обучения можно разделить на следующие классы:

— методы с выделением признаков (Feature based);

— методы, основанные на представлении облака точек вокселями (Voxel based);

— методы, использующие виды с нескольких ракурсов (Multi view);

— методы, основанные на использовании неизмененного облака точек (Point based);

— методы, использующие представление облака точек в виде графов (Graph based).

Методы с выделением признаков используют предварительно извлечённые глобальные или локальные признаки облака точек. Такие признаки извлекаются и описываются с помощью классических методов. Примерами таких подходов могут служить 3DmFV-Net [18], показавший 91,4% и CurveNet [19], с результатом точности идентификации 94,2% на наборе данных ModelNet40.

В работе [18] авторы используют представления облака точек в виде трехмерной сетки, после чего рассчитывают модифицированные вектора Фишера [20] (3D Modified Fisher Vectors). По полученным векторам выполняется идентификации с помощью сверточной нейронной сети.

В работе [19] использовалась методика, основанная на поиске последовательностей (кривых) в облаке точек и использовании их в качестве геометрических особенностей. В работе предложен оператор для выделения и агрегации выделенных последовательностей точек. Полученные кривые подаются на вход в сверточную нейронную сеть, с помощью которой и выполняется идентификация.

Методы, основанные на представлении облака точек вокселями, предварительно вокселезируют неструктурированные облака точек, после чего выполняют идентификацию. В работе [21] рассматривается методика с применением VoxNet, являющейся одной из первых сверточных нейронных сетей для работы с трехмерными объектами. В рассматриваемой методике воксельная сетка, полученная по облаку точек, подается непосредственно на вход сверточной нейронной сети, которая пытается предугадать класс объекта.

Методика с использованием кодировщика PointPillars [22] используется с лидаром для детектирования объектов и выделения ограничивающих рамок значимых объектов в облаке точек. Процесс обработки состоит из трех этапов: на первом этапе облако точек организовывается в вертикальные столбцы (авторы выделяют, что столбец является вокселем с неограниченным размером оси Z), после чего с помощью нейронной сети преобразовывается в разреженное псевдоизображение, на втором этапе двумерная сверточная нейронная сеть преобразовывает полученное псевдоизображение в высокоуровневое представление, после чего на третьем этапе с помощью детектора обнаруживаются ограничивающие рамки объекта.

В *методах, использующих виды с нескольких ракурсов,* облако точек представляется набором 2D изображений. Лучшие результаты (94,1% на наборе данных ModelNet40) в этом классе показала методика с использованием сети MHBN (Multi-View Harmonized Bilinear Network) [23]. В этой методике сначала вычисляются локальные признаки с помощью сверточной нейронной сети, после чего происходит их объединение с помощью билинейного слияния. При этом различные компоненты, входящие в состав билинейного признака, согласовываются, для создания более информативного представления. Чтобы обеспечить возможность сквозного обучения билинейное слияние включено в качестве слоя сверточной сети.

В *методах, основанных на использовании неизмененного облака точек,* облако точек в исходном виде передается в нейронную сеть. Методика [24], показавшая 93,2% идентификации на наборе данных ModelNet40, с использованием PointASNL, относится этому классу методов. PointANSL сквозная нейронная сеть, предназначенная для работы с неорганизованным облаком точек, содержащим шумы и выбросы. Отличительной особенностью методики является использование модуля адаптивной выборки (Adaptive Sampling), который корректирует координаты и характеристики точек, выбранных с помощью алгоритма выборки самых удаленных точек (Farthest Point Sampling). Для обеспечения устойчивости процесса обучения к шуму, авторами разработан модуль «локальный-нелокальный» (local-nonlocal), позволяющий объединять локальные и не локальные признаки в выбранных точках.

В *методах, использующих представление в виде графов,* облако точек представляется в виде графов вида точка — ближайшие соседи. Методика PointView-GCN [25] показала результаты 95,4% точности идентификации на наборе данных ModelNet40. Эта методика реализует методы представления в виде графов и использования нескольких ракурсов. В нем происходит объединение признаков, полученных из частичных облаков точек, снятых с

различных ракурсов вокруг объекта, для повышения точности идентификации. В процессе идентификации используется многоуровневая графовая сверточная нейронная сеть, позволяющая фиксировать геометрические признаки объекта, полученные с разных ракурсов, и отношения положений ракурсов. Извлечение геометрических признаков объектов происходит с помощью слоев графовой сверточной сети DGCNN [26].

Алгоритм идентификации детали с помощью классических подходов изображен на рисунке 4 и состоит из следующих шагов:

1. На подготовительном этапе формируется набор эталонных дескрипторов деталей для идентификации, по которым будет осуществляется идентификация.
2. Облако точек, полученное с устройства сканирования подготавливается тем или иным образом, чтобы наиболее точно построить дескриптор.
3. По подготовленному облаку точек строится один или несколько дескрипторов в зависимости от выбранной методики идентификации.
4. По полученным дескрипторам и эталонным дескрипторам выполняется идентификация детали.

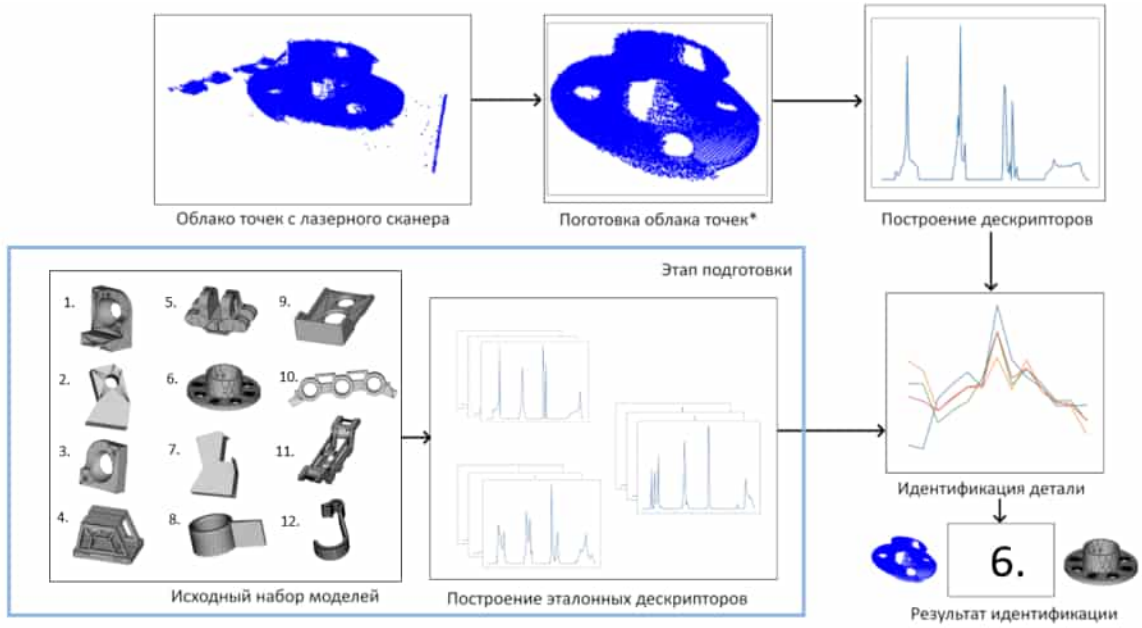


Рисунок 4 — Схема процесса идентификации детали классическими алгоритмами

Классификация методов идентификации классическими алгоритмами основывается на типе алгоритмов извлечения ключевых признаков, которые используются для построения дескриптора. Такие методы можно разделить на две категории: методы с использованием глобальных и локальных признаков [27].

Методы с использованием глобальных признаков. Эта категория методов обрабатывает облако точек как единый объект, выделяя набор глобальных признаков, по которым выполняется построение глобального дескриптора, эффективно и кратко описывающий весь объект. Эти методы широко используются для поиска и классификации трехмерных объектов в облаке точек. В работе [11] предлагается следующая классификация классических методов с использованием глобальных признаков:

— методы, использующие геометрические признаки для построения глобального дескриптора;
— методы, использующие пространственные признаки для построения глобального дескриптора.

В методах, использующих геометрические признаки для построения глобального дескриптора, дескрипторы составляются на основе геометрических данных о поверхности объекта. Используются геометрические свойства или взаимосвязи между всеми точками для описания всего облака точек с помощью дескриптора. Чаще всего дескриптор составляется в виде гистограммы. Среди таких типов дескрипторов можно выделить Global RSD (GRSD) [28], Viewpoint Feature Histogram (VFH) [29], Clustered Viewpoint Feature Histogram (CVFH) [30], The Oriented, Unique, and Repeatable Clustered Viewpoint Feature Histogram (OUR-CVFH) [31].

Global RSD — является глобальной версией дескриптора RSD (Radius-based Surface Descriptor), использующего локальные признаки [32]. В предложенной методике построения дескриптора объединяются методы по обработке 3D и 2D данных. Сначала входное облако вокселизируется, после чего

для каждого вокселя с помощью дескриптора RSD определяется класс поверхности (плоскость, сфера, цилиндр, ребро, кромка). С помощью GRSD выполняется идентификация геометрически идентичных кластеров точек. Как только выполнена идентификация, из двумерного изображения с камеры выделяется область, содержащая идентифицированный объект, для которого с помощью дескриптора SURF [33] рассчитывается 2D вектор признаков. С помощью полученного вектора выполняется окончательная идентификация и позиционирование объекта. Авторами заявлена точность идентификации 98,63% на объектах, используемых в работе (тарелка, коробка, кастрюля и тд.).

Viewpoint Feature Histogram— расширяет дескриптор FPFH [34], добавляя к нему компоненту ракурса, позволяющую выполнять грубое 6DoF позиционирование. Представляет собой вектор, размерностью 263 элемента, состоящей из нормализованных гистограмм распределения угловых характеристик между всеми точками облака и средними точкой и нормалью.

Clustered Viewpoint Feature Histogram — вектор размерностью 308 элементов, является доработкой глобального дескриптора VFH с целью более стабильного расчета средней точки и средней нормали в облаках точек частично перекрывающих поверхность объекта. Для этого похожие поверхности в облаке точек объединяются в кластеры, для каждого из которых рассчитывается средняя точка и средняя нормаль. После чего для каждого кластера рассчитывается дескриптор VFH и дополнительная компонента распределения точек Shape Distribution Component (SDC). Для неявного кодирования размеров объекта в дескрипторе авторы предложили вокселизировать облако точек, а в гистограмме использовать абсолютное число точек, а не нормализованное, как в дескрипторе VFH.

The Oriented, Unique, and Repeatable Clustered Viewpoint Feature Histogram — вектор размерностью 303 элемента, являющийся развитием дескриптора CVFH. Имеет те же компоненты, что и дескриптор CVFH, однако компонента ракурса сокращена до 64 элементов. Добавлены гистограммы расстояний между

точками поверхности кластера и средней точки, для замены компоненты SDC, полученных с помощью Semi-Global Unique Reference Frame (SGURF).

В методах, использующих пространственные признаки для построения глобального дескриптора, сначала производится преобразования входного облако точек в соответствии с заданным алгоритмом в сетку, ячейки или столбцы. После этого дескриптор формируется путем сбора характеристик пространственного распределения точек внутри каждой из единиц преобразования. Примерами таких методов можно выделить Ensemble of Shape Functions (ESF) [35], The Global Orthographic Object Descriptor (GOOD) [36].

Ensemble of Shape Functions — дескриптор, разработанный для работы с облаком точек, частично покрывающим поверхность объекта, являющийся вектором, размерностью 640 элементов. Вектор состоит из 10 гистограмм по 64 элемента в каждой: три гистограммы распределения углов, три гистограммы распределения площадей, три гистограммы расстояний и одну гистограмму отношений расстояний. В методике построения дескриптора для девяти первых гистограмм по алгоритму, описанному в работе [37], для каждого из случаев ON, OFF и MIXED строятся функции A3 — угол, образованный тремя случайными точками, D3 — площадь треугольника, образованного тремя случайными точками и D2 — длина линии между двумя случайными точками. Гистограмма отношения расстояний строится на основе длины отрезков из функции формы D2.

Методика построения The Global Orthographic Object Descriptor представлена следующими шагами: сначала определяется локальная система координат с помощью метода анализа главных компонент; облако точек ортогонально проецируется на плоскости XoY, YoZ и ZoX; плоскости разделяются на заданные интервалы в которых производится расчет матриц распределения, путем суммирования точек, попадающих в каждый интервал; для полученных трех матриц распределения рассчитываются статистические

показатели — энтропия и дисперсия; с помощью рассчитанных показателей матрицы объединяются в единый дескриптора объекта.

В отличие от методов построения дескрипторов по глобальным признакам, методы с использованием локальных признаков состоят из двух этапов: нахождение ключевых точек и описание дескрипторами признаков в окрестностях этих точек. Ключевая точка — это такая точка, которая определяет положение окрестности для построения локального дескриптора, а дескриптор описывает признаки в окрестности ключевой точки. Поэтому сначала рассмотрены методы поиска ключевых точек.

Методы поиска ключевых точек делятся на две группы: методы обнаружения ключевых точек фиксированного масштаба и методы обнаружения ключевых точек адаптивного масштаба.

Методы обнаружения ключевых точек фиксированного масштаба — это методы, выполняющие поиск и определение точек, которые являются отличительными в пределах заданной окрестности. Размер окрестности является входным параметром алгоритма. Отличительными особенностями могут быть:

— параметры кривизны поверхности;

— показатели изменения формы поверхности.

Методы на основе кривизны поверхности используют различные измерения кривизны в качестве мер отличия для определения ключевых точек. Например, в работе [38] производится предварительное сглаживание перед расчётом ключевых точек, для устранения шума и уменьшения количества ключевых точек. Далее для каждой точки рассчитываются Гауссова и средняя кривизна. Локальные максимумы Гауссовой и средней кривизны выделяются в качестве ключевых точек.

В работе [39] была введена численная мера формы в окрестности точки. Окрестности каждой точки аппроксимировались квадратичной поверхностью с помощью метода наименьших квадратов. После чего для рассчитанной окрестности в точке вычислялась нормаль, Гауссова и средняя кривизна, главная

кривизна. По полученным значениям для каждой точки рассчитывалась мера формы. Точка помечалась ключевой, если полученная мера была локальным экстремумом.

Другие методы на основе измерения изменения формы поверхности. Такие методы используют отличные от измерения кривизны в точке показатели изменения поверхности. К этому классу методов можно отнести работу [40] — в ней величина наименьшего собственного значения матрицы ковариации, составленная из точек в заданной окрестности, принималась как мера изменения поверхности. В работе [41] предложен трехмерный оператор Харриса, адаптированный для работы с трехмерными данными.

Методы обнаружения ключевых точек с адаптивным масштабированием. Этот класс методов в заданном диапазоне создает масштабные пространства для облака точек. В качестве ключевых точек выбираются те, которые обладают высокой степенью различимости как в масштабной, так и в пространственной окрестности. Для такого класса методов характерны следующие:

— методы на основе сглаживания координат;

— методы на основе изменения поверхности;

— методы, основанные на сглаживании геометрических признаков;

— методы на основе преобразований.

В *методах на основе сглаживания координат* масштабные пространства создаются путем последовательного сглаживания трехмерных координат точек в облаке. Примером таких методов могут служить работы [42, 43] — развитие алгоритма масштабно-инвариантной трансформации признаков (SIFT). Для создания масштабно-пространственного представления использовались методы Гауссова сглаживания и пирамиды разности Гауссовых функций (DOG). Точки экстремума выделялись в пространстве DOG, после чего проверялись путем анализа отношения главных значений кривизны с заданным пороговым значением.

В *методах на основе изменения поверхности* сначала вычисляется изменения поверхности в наборе окрестностей различного размера, после чего находят ключевые точки путем поиска максимального изменения поверхности в заданных окрестностях различного размера. Эти методы основаны на предположении, что размер окрестности может рассматриваться как дискретный параметр масштаба. В работе [44] предложена методика для поиска ключевых точек и оценки их качества, представленная следующими шагами: для каждой точки и ее окрестности выполняется поворот, таким образом, чтобы нормаль в обрабатываемой точке совпала с положительной Z осью системы координат; после этого для окрестности с помощью анализа главных компонент рассчитываются направления главных осей (x и y); вычисляется отношение длин облака в окрестности по этим осям, которое и является мерой изменения поверхности; производится выбор ключевых точек путем сравнением меры изменения поверхности с пороговым значением.

Методы, основанные на сглаживании геометрических признаков, создают масштабное пространство путем последовательного сглаживания геометрических признаков поверхности. Например, в работе [45] выполнялось уменьшение разрешения поверхности по мере увеличения масштаба с одновременным построением масштабно-пространственного представлением на поверхности с помощью Гауссова ядра. В качестве ключевых точек выбирались точки экстремума в масштабно-пространственном представлении.

В методах на основе преобразований поверхность рассматривают как риманово многообразие и преобразуют его из пространственной области в другую (например, в спектральную). Обнаружение ключевых точек происходит в преобразованной области. В работе [46] был предложен Heat Kernel Signature (HKS). Авторы ограничили ядро теплопроводности временной составляющей и осуществляли поиск ключевых точек в точках локальных максимумов ограниченной функции теплопроводности.

После того, как ключевые точки обнаружены тем или иным методом, в окрестности ключевой точки выполняется построение локального дескриптора по локальным признакам. Методы построения локального дескриптора по локальным признакам представлены следующими классами:

1) методы на основе сигнатур;

2) методы на основе:

а) гистограмм направленных градиентов;

б) гистограмм распределения в пространстве;

в) гистограмм геометрических признаков;

3) методы на основе преобразований.

Методы на основе сигнатур описывают окрестность вокруг ключевой точки путем кодирования одной или нескольких геометрических величин вычисляемых для каждой из точек подмножества окрестности. К такой категории методов можно отнести работу [47] — дескриптор Binary Robust Appearance and Normal Descriptor (BRAND). В методике, предложенной в работе, для ключевой точки в RGB-D пространстве выбиралась окрестность, точки в окрестности выравнивалась по доминантному направлению и масштабировались с учетом глубины (D канала). В дескрипторе объединена информация об интенсивности цвета и геометрии в окрестности, так как он формировался по смещению нормали и изменению интенсивности цвета. В работе [48] проведена экспериментальная проверка идентификации с помощью описанного дескриптора, и указана средняя точность идентификации 64% при использовании с различными методами выделения ключевых точек.

Методы на основе гистограмм. Такие методы описывают локальную окрестность ключевой точки собирая геометрические или топологические величины в гистограммы. Эти методы зачастую схожи с методами выделения глобальных признаков на основе гистограмм, однако дескриптор вычисляется только для точек в окрестности ключевой точки, а не для всего облака точек.

Методы на основе гистограмм направленных градиентов. Методы этой группы описывают локальную окрестность гистограммой на основе направленных градиентов поверхности в локальной окрестности ключевой точки. Такой метод используется в работе [49] и реализуется с помощью дескриптора MeshHOG. Для построения дескриптора на вершинах сетки определяется функция, для которой рассчитываются градиенты. Функция может описывать нормали, кривизну, текстуру или глубину. Для каждой ключевой точки рассчитывается локальная система координат, на каждую ортонормированную плоскость которой проецируются векторы градиента. Каждая плоскость разделяется на четыре полярных региона, для которых генерируются гистограммы размерностью восемь элементов в соответствии с направлениями векторов градиента. Дескриптор составляется из всех полученных гистограмм.

Методы на основе гистограмм распределения в пространстве — описывают локальную окрестность ключевой точки гистограммой на основе пространственного распределения поверхности в окрестности точки. Такой вид гистограмм используется в дескрипторе Rotational Projection Statistics (RoPS) [50]. Дескриптор строится путем последовательных поворотов облака точек вокруг оси X после расчета локальной системы координат. Точки проецируются на плоскости XoY, YoZ, ZoX после каждого поворота. Для каждой плоскости вычисляется матрица распределения точек. Дескриптор составляется из четырех центральных моментов и энтропии всех матриц.

Методы на основе гистограмм геометрических признаков — описывают локальную окрестность ключевой точки гистограммой в соответствии с геометрическими признаками локальной поверхности в окрестности (например, нормали, кривые). Одним из наиболее известных дескрипторов, использующийся в методах этой категории, является дескриптор SHOT (Signature of Histograms of Orientations) [51], разработанный для идентификации по RGB-D изображением. Методика построения дескриптора состоит из

следующих шагов. Сначала в ключевой точке рассчитывается локальная система координат, после этого окрестность вокруг ключевой точки разделяется на сферические сектора. Для каждого сектора рассчитывается гистограмма, путем суммирования количества точек в зависимости от углов между нормалью в ключевой точке и нормалями в соседних точках. Авторы выделяют, что SHOT имеет высокую степень описательности, показывает себя эффективным и устойчивым к шуму.

Дескриптор LDFH [52] также используется в методах на основе гистограмм геометрических признаков. Для построения дескриптора в ключевой точке, рассчитывается локально минимальная ось (LMA) [53]. После чего для всех точек в окрестности рассчитываются угловые характеристики относительно рассчитанной LMA и ключевой точки. По полученным характеристикам строятся гистограммы распределения, которые объединяются в единый дескриптор с применением весовых коэффициентов.

Методы на основе преобразований. Такие методы сначала преобразовывают облако точек из пространственной области в другую (например, в спектральную), после чего описывают трехмерную окрестность ключевой точки, кодируя информацию в преобразованной области. Примером использования может стать методика, описанная в работе [54]. В роле дескриптора в нем выступает адаптированный для трехмерного пространства дескриптор Speeded Up Robust Feature (SURF) [55], именуемый в работе 3D SURF. Сначала облако точек вокселизировалось, после чего к нему применялось преобразование на основе вейвлета Хаара. Для каждой ключевой точки определялась локальная система координат. Окрестность ключевой точки разделялась на определенное количество секторов, для каждого из которых рассчитывался вектор градиентов на основе проведенного преобразования. Полученные вектора градиентов объединялись в дескриптор.

Сравнение рассмотренных методик представлено в таблице 1.

Таблица 1 — Сравнение методик идентификации

Категория методов	**Класс методов**	**Методика, описанная в работе**	**Точность**	**Недостатки применимо к работе**
Методы с использованием глубокого обучения	Методы с выделением признаков	CurveNet [19]	94,2% на синтетическом ModelNet40	Необходимость переобучения при добавлении новых классов. Тестирование на объектах не промышленного назначения.
	Методы, основанные на представлении облака точек вокселями	VoxNet [21]	85,9% на синтетическом ModelNet40	Необходимость переобучения при добавлении новых классов. Тестирование на объектах не промышленного назначения.
	Методы, использующие виды с нескольких ракурсов	MHBN [23]	94,1% на синтетическом ModelNet40	Необходимость большого количества ракурсов. Переобучение при добавлении новых классов. Тестирование на объектах не промышленного назначения.
	Методы, основанные на использовании неизмененного облака точек	PointASNL [24]	93,2% на синтетическом ModelNet40	Необходимость переобучения при добавлении новых классов. Тестирование на объектах не промышленного назначения. Долгое вычисление.
	Методы, использующие представление облака точек в виде графов	PointView-GCN [25]	95,2% на синтетическом ModelNet40	Необходимость большого количества ракурсов. Переобучение при добавлении новых классов. Тестирование на объектах не промышленного назначения.

Классические методы	Методы, использующие геометрические признаки для построения глобального дескриптора	GRSD [28]	82% на реальных данных лазерного сканирования [17]	Низкие показатели точности идентификации на реальных данных. Необходимо изображение с камеры, которое тяжело получить при использовании системы лазерного сканирования, рассматриваемой в работе.
		VFH [29]	72% на реальных данных лазерного сканирования [17]	Низкие показатели точности идентификации на реальных данных. Проблема в определения средней точки и средней нормали для облаков точек, частично покрывающих поверхность объекта.
		CVFH [30]	95% на реальных данных лазерного сканирования [17]	Выделяет большое количество кластеров на поверхностях сложной формы. Сложная методика формирования набора эталонных дескрипторов.
		OUR-CVFH [31]	92% на реальных данных лазерного сканирования [17]	Более низкая точность идентификации на реальных данных по сравнению с дескриптором CVFH, при этом более сложная методика построения дескриптора.
	Методы, использующие пространстве	ESF [35]	88% на реальных данных лазерного сканирования [17]	Низкие показатели точности идентификации. Не описана методика

	нные признаки для построения глобального дескриптора			формирования эталонных дескрипторов. Большая размерность вектора ведет к увеличению времени идентификации.
		GOOD [36]	58% на реальных данных лазерного сканирования [17]	Самые низкие показатели точности идентификации из рассматриваемых дескрипторов на реальных данных. Зависимость от определения локальной системы координат.
	Методы на основе сигнатур (локальные признаки)	BRAND [47]	64% на Freiburg RGB-D [48]	Низкие показатели точности идентификации. Необходимость RGB-D изображения, что не применимо к системе сканирования, рассматриваемой в работе.
	Методы на основе гистограмм распределения в пространстве (локальные признаки)	RoPS [50]	90%-100% в среднем на разных наборах данных и в зависимости от перекрытия [50]	Чувствителен к помехам и перекрытиям объекта. Зависим от выбора размеров окрестности.
	Методы на основе гистограмм направленных градиентов (локальные признаки)	SHOT [50]	Различные данные. В работе [56] показал характеристики сравнимые с RoPS	Чувствителен к помехам и перекрытиям объекта.
	Методы на основе гистограмм	LDFH [52]	Показал лучшие чем SHOT результаты [52]	Используется большой радиус окрестности при

	геометрических признаков (локальные признаки)			построении LMA. Часть точек при лазерном сканировании будет скрыта, что даст неустойчивое построение дескриптора.
	Методы на основе преобразований (локальные признаки)	3D SURF [54]	65,4% в среднем [54]	Чувствителен к помехам и перекрытиям.

В классе методов с применением глубокого обучения наилучшие показатели точности идентификации получены у методов с пользованием извлеченных признаков и представления облака точек графом. Однако тестирование производилось на наборах данных, состоящих из сильно отличающиеся классов объектов (самолет, бутылка, машина, стул, и т.д. [57]), и зачастую не тестировалось на наборах данных промышленных объектов, имеющих сложную форму и геометрию.

Отдельной проблемой является необходимость переобучения или дообучения сети при добавлении нового типа деталей, требовательность к вычислительным ресурсам и ограниченная скорость идентификации.

На основании описанных недостатков, выбор методов идентификации осуществляется в классе методов с использованием классических алгоритмов, разработанных для работы с облаком точек, полученного с устройств, аналогичным рассматриваемому в работе.

Анализ существующих методов показал, что методы с использованием локальных признаков плохо применимы для решения задачи, рассматриваемой в работе. Основной причиной является особенности структуры облака точек полученного с рассматриваемой в работе точки контроля. Часть поверхности

детали скрыта в местах резкого изменения геометрии поверхности, а такие места в являются областями, содержащими ключевые точки, в которых выполняется построение локальных дескрипторов. Из-за особенностей структуры облака точек будет происходить некорректное определение ключевых точек или некорректное построение дескрипторов. Поэтому в работе выбраны методы идентификации с использованием глобальных дескрипторов.

Наилучшие показатели на реальных данных показал глобальный дескриптор CVFH [29], являющийся развитием глобального дескриптора VFH [29], адаптированный для работы с облаком точек, которое частично покрывает поверхность детали.

VFH (Viewpoint Feature Histogram) — дескриптор, позволяющий описывать информацию об объекте по виду с определенного ракурса, вектор размерностью 263 элемента, состоящий из двух компонентов: дескриптора FPFH (Fast Point Feature Histogram) [34], размерностью 135 элементов, и компоненты ракурса, размерностью 128 элементов, впервые введенной в дескрипторе VFH для грубой оценки 6DoF позиционирования.

На вход алгоритма расчета дескриптора VFH подается облако точек $S = \{p_i, n_i\}_{i=1}^{n}$, $\|n_i\| = 1$, для каждой из точек которого была аппроксимирована нормаль (p_i — координаты точки, n_i — вектор аппроксимированной нормали в точке p_i). Для облака точек рассчитывается центр облака точек p_c (1) и средняя нормаль n_c (2):

$$p_c = \frac{1}{n}\sum_{i=1}^{n} p_i, \tag{1}$$

$$n_c = \frac{1}{n}\sum_{i=1}^{n} n_i. \tag{2}$$

Для каждой точки $s_i \in S$, рассчитывается трехгранник Дарбу с осями (u_i, v_i, w_i):

$$u_i = n_c, \tag{3}$$

$$v_i = \frac{p_i - p_c}{\|p_i - p_c\|} \times u_i, \tag{4}$$

$$w_i = u_i \times v_i. \quad (5)$$

После чего рассчитываются угловые компоненты отклонения нормали $\cos(\alpha_i)$, $\cos(\phi_i)$, θ_i и компонента ракурса $\cos(\beta_i)$:

$$\cos(\alpha_i) = v_i \cdot n_i, \quad (6)$$

$$\cos(\phi_i) = u_i \cdot \frac{p_i - p_c}{\|p_i - p_c\|}, \quad (7)$$

$$\theta_i = arctg(w_i \cdot n_i, u_i \cdot n_i), \quad (8)$$

$$\cos(\beta_i) = n_i \cdot \frac{p_c}{\|p_c\|}. \quad (9)$$

Для каждой из величин угловых отклонений нормали и для компоненты ракурса строятся гистограммы, содержащие 45 и 128 ячеек, с интервалами $\frac{2\pi}{45}$ и $\frac{2\pi}{128}$ соответственно, где значение каждой ячейки равнялось проценту точек, для которых рассчитанные значения угловых отклонений нормали и компоненты ракурса попадали в заданный интервал. С помощью этих гистограмм формируется дескриптор VFH вида $(\cos(\beta_i), \cos(\alpha_i), \cos(\phi_i), \theta_i)$. Вид дескриптора изображен на рисунке 6.

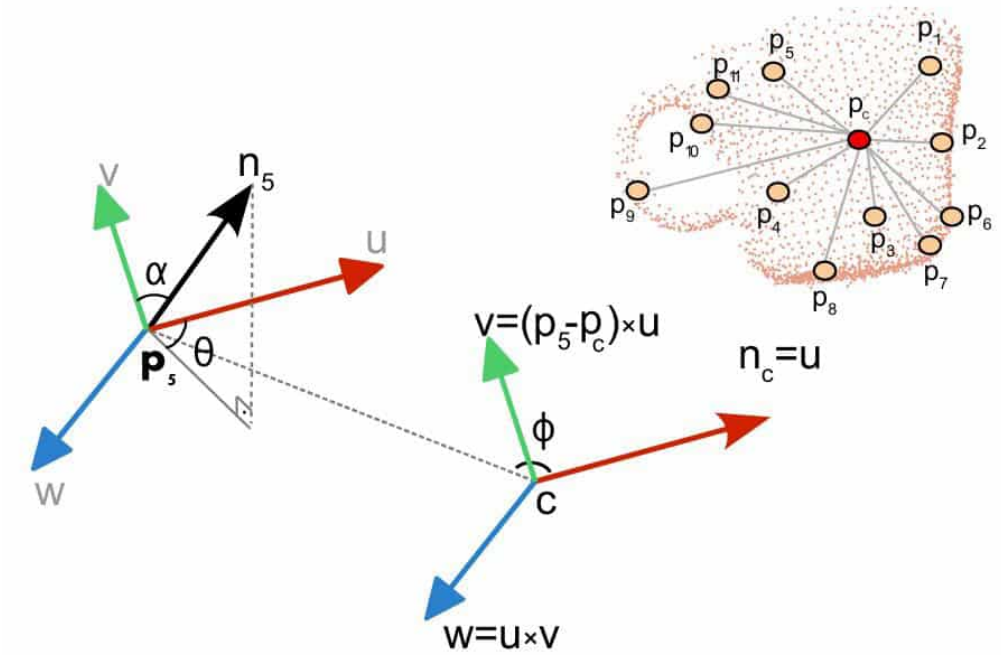


Рисунок 5 — Расчет угловых компонент для точки p_5 во время расчёта дескриптора VFH [29]

При разработке дескриптора CVFH (Clustered Viewpoint Feature Histogram) были учтены недостатки дескриптора VFH: чувствительность к шуму и выбросам, из-за расчета центра и средней нормали, а также отсутствие количественной информации о точках, что могло вызывать коллизии при распознавании объектов одинаковых форм, но разных размеров. Чтобы устранить коллизии авторы изменили процесс расчета гистограмм и подготовки облака точек.

Сначала по входному облаку точек строилась воксельная сетка, чтобы количество точек в дескрипторе могло отражать информацию о площади поверхности объекта. Затем была доработана методика построения гистограмм — гистограммы не нормализировались. Таким образом авторы учитывали параметры размера детали в дескрипторе, путем неявного кодирования площади поверхности объекта.

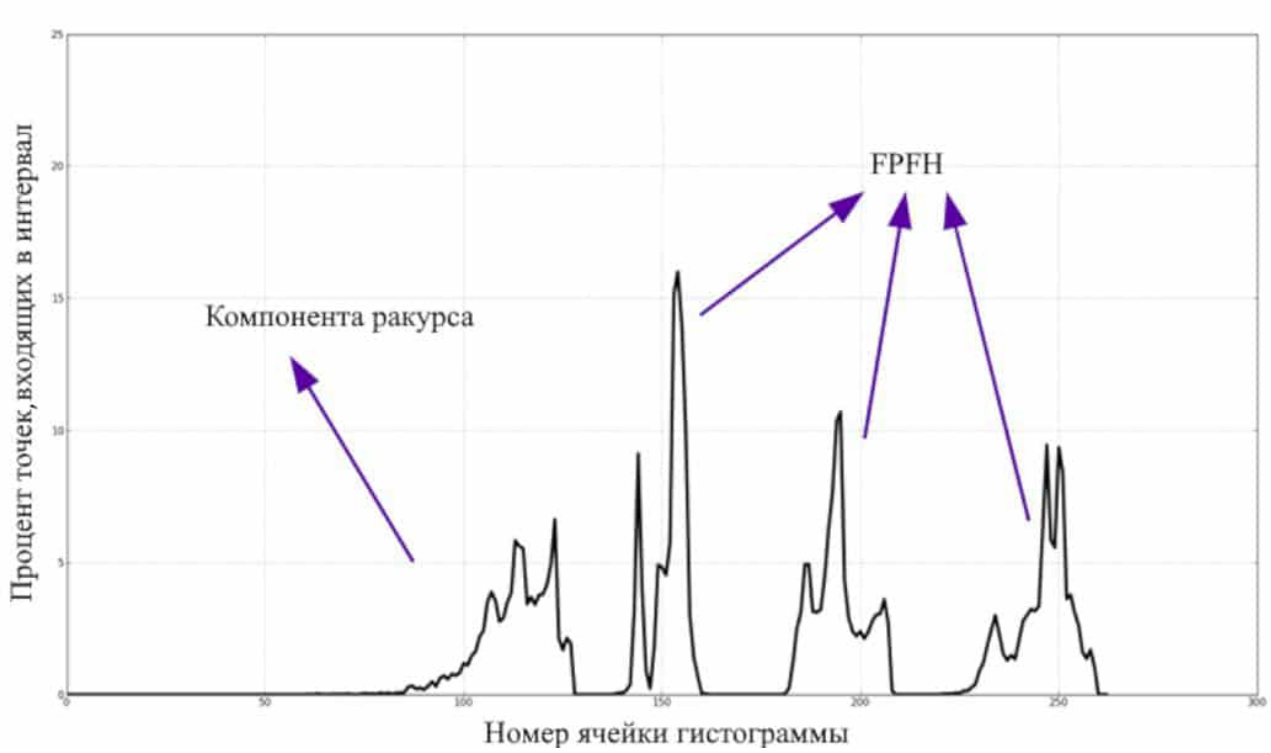


Рисунок 6 — Дескриптор VFH [29]

Чтобы устранить неточность при определении центра и расчета средней нормали, авторы предложили следующий алгоритм:

1. Из облака точек, используемого для расчета средней точки и средней нормали, удаляются точки со значением кривизны больше порогового.
2. Затем облако точек разделяется на кластеры с похожей геометрией с помощью алгоритма мягкого расширения областей. После чего по точкам $S_j = \left\{p_{j_i}, n_{j_i}\right\}_{i=1}^{n_j}$, где j — индекс кластера, для каждого кластера рассчитывать центр и нормаль:

$$p_{c_j} = \frac{1}{n_j}\sum_{i=1}^{n_j} p_{j_i}, \tag{10}$$

$$n_{c_j} = \frac{1}{n_j}\sum_{i=1}^{n_j} n_{j_i}. \tag{11}$$

3. Далее по всем точкам входного облака точек для каждого кластера рассчитывался дескриптор по методике построения дескриптора VFH, где $p_c = p_{c_j}$, а $n_c = n_{c_j}$.

Для кодирования формы в дескриптор был добавлена компонента SDC (Shape Distribution Component) — гистограмма размерностью 45 ячеек, показывающая распределение точек по удаленности от рассчитанного центра (12):

$$SDC_j = \frac{\left(p_{c_j} - p_i\right)^2}{\max\left(\left(p_{c_j} - p_i\right)^2\right)}. \tag{12}$$

В результате размерность дескриптора CVFH составила 308 элемента.

На вход алгоритма построения дескриптора CVFH подается облако точек объекта, снятого с определенного ракурса, на выходе рассчитывается j дескрипторов, где j — количество кластеров выделенных для заданного облака точек.

Отдельно стоит отметить, что авторы предложили алгоритм поиска похожих дескрипторов во время этапа формирования эталонных дескрипторов, которые используются для идентификации. Для этого предлагалось сопоставлять выровненное облако точек модели для каждого из уже

сгенерированных дескрипторов с облаком точек модели полученного с определенного ракурса, для которого был построен дескриптор. Если было найдено совпадение, то ракурс игнорировался, а дескриптор не попадал в финальную выборку.

1.3. Анализ существующих методов подготовки облака точек

Применение различных методов предварительной обработки облака перед построением дескриптора позволяет повысить точность идентификации путем фильтрации шумов, удалением выбросов и более точным описанием поверхности детали облаком точек и нормалями, аппроксимированными в точках облака. В работах [59, 60] предлагается следующая классификация методов фильтрации облака точек:

— статистические методы фильтрации облака точек;
— методы фильтрации облака точек на основе окрестностей точек;
— методы фильтрации облака точек на основе проекций;
— методы фильтрации облака точек на основе подходов к обработке сигналов;
— методы фильтрации облака точек с использованием дифференциальных уравнений с частными производными;
— специальные методы.

Статистические методы фильтрации облака точек — основаны на адаптации статистических принципов для работы с облаком точек. Методики фильтрации, описанные в работах [61, 62] можно отнести к этому классу методов. В работе [60] представлена методика фильтрации на основе оптимизации с помощью минимизации нормы L_0, которая напрямую удаляет шумы, оценивает нормали в точках и перемещает точки вдоль направления нормалей. Такой подход предлагался авторами с целью более точного сохранения формы объекта.

Методы фильтрации облака точек на основе окрестностей точек — основаны на определении результирующего положения отфильтрованной точки

используя показатели сходства между точкой и ее окрестностью. Такие методы используются в работах [63, 64]. Методика, описанная в [63] основана на алгоритме фильтрации с помощью сдвига среднего значения, который изначально применялся для обработки изображений. Алгоритм фильтрации был расширен, для работы с трехмерными поверхностями, для этого он использовал: аппроксимированную нормаль в точках, значение кривизны в качестве компоненты расстояния, а трехмерные координаты точки в качестве компоненты положения.

Методы фильтрации облака точек на основе проекций — построены на проецировании точек для коррекции их положения. Примерами реализации таких методов могут служить методики из работ [65, 66]. В работе [65] предложена методика с использованием оператора Locally Optimal Projection (LOP). Основная идея методики состоит в итеративном проецировании набора точек в исходное облако точек с целью удаления шумов и выбросов.

Методы фильтрации облака точек на основе подходов к обработке сигналов — для фильтрации используют методы для обработки сигналов. К этому типу методов можно отнести работу [67], в которой предложен метод аппроксимации облака точек дискретным оператором Лапласа. Для устранения сжатия в методе предложен метод с аппроксимацией изменения объема в окрестностях и компенсирования его путем смещения соседних точек.

Методы фильтрации облака точек с использованием дифференциальных уравнений с частными производными. Такие методы используются в работах [68, 69]. В работе [68] строились взвешенные графы, представляющие облако точек, с учетом окрестностей облака точек. Для фильтрации применялся адаптированный функционал дифференциальных уравнений в частных производных под взвешенные графы [70], такие как оператор p-Лапласиан и морфологические операторы с использованием дифференциальных уравнений в частных производных.

Специальные методы — методы в которых применяется два и более описанных выше метода, или они не подходят под выбранные классы [71, 72, 73, 74]. В работе [74] описан фильтр, который принимает на вход необработанное облако точек с шумами и выбросами, получая на выходе чистый, однородный и сохраняющий геометрию поверхности набор точек, с рассчитанными нормалями к поверхности, для каждой из полученных точек. Описанный фильтр разделен на две фазы. На первой фазе происходит реорганизация и расчет нормалей для точек, находящихся на удалении от граней. На второй фазе происходит увеличение плотности облака точек, для заполнения пространства вблизи граней. Пример работы данного фильтра изображен на рисунке 7.

Из рассмотренных методов фильтрации наибольший интерес для работы представляет первая фаза фильтра, описанного в работе [74]. Первая фаза предоставляет механизм расчета нормалей и сглаживания облака точек для восстановления исходной геометрии поверхности. Правильность расчета нормалей точек в облаке, а также определение геометрии поверхности — играет важную роль в построении дескрипторов.

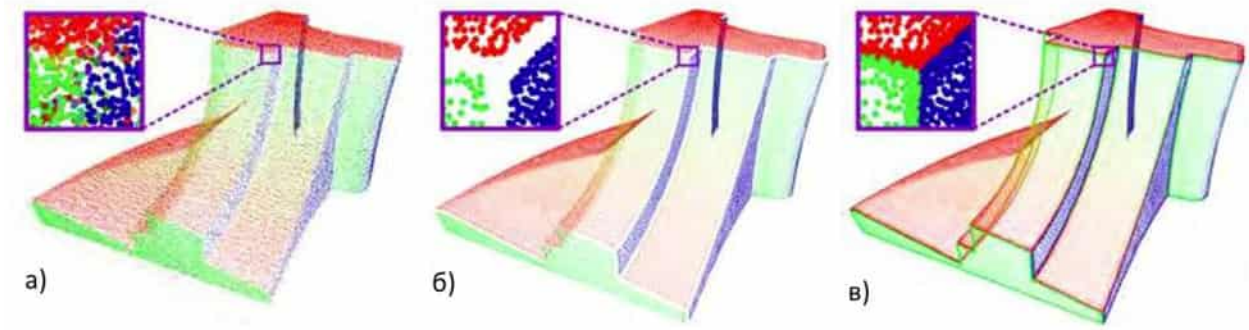


Рисунок 7 — Результат работы фильтра [74]: а — исходное облако точек; б — фильтрация облака на расстоянии от граней (первая фаза); в — увеличение плотности облака точек, для заполнения пространства вблизи граней (вторая фаза)

Первая фаза фильтра состоит из нескольких этапов. Сначала происходит грубая аппроксимация нормалями для точек по их окрестностям с помощью метода анализа главных компонент (PCA) [75]. Так как этот метод подвержен

ошибкам из-за шума и выбросов, производится итеративное сглаживание нормалей с помощью механизма билатерального сглаживания нормалей [76] и реорганизация облака точек с помощью метода LOP.

Для точки $s_i = (p_i, n_i)$, где p_i-координаты точки, а n_i- нормаль, вычисляется функция расхождения между нормалью и нормалями соседних точек в заданной окрестности σ_p:

$$f(p_i, n_i) = \sum_{s_j \in O_{s_i}} \|n_i - n_j\| \theta(\|p_i - p_j\|) \psi(n_i, n_j), \quad (13)$$

где $O_{s_i} = \{s_j | s_j \in S \cap \|p_i - p_j\| < \sigma_p\}$.

Пространственная и весовая функция нормали заданы как:

$$\theta(r) = e^{-\frac{r^2}{{\sigma_p}^2}}, \quad (14)$$

$$\psi(n_i, n_j) = e^{-\left(\frac{1-{n_i}^T n_j}{1-\cos(\sigma_n)}\right)^2}, \quad (15)$$

где $\sigma_n = 15°$.

Задача механизма сглаживания состоит в минимизации функции расхождения нормалей между всеми точками на поверхности и их соседями в заданных окрестностях (16):

$$\underset{n}{\operatorname{argmin}} \sum_{i \in I} f(p_i, n_i). \quad (16)$$

Данная задача решается с помощью метода наименьших квадратов с пересчетом весов, итеративно рассчитывая следующее значение нормали n_i (17):

$${n_i}^{k+1} \leftarrow \frac{\sum_{s_j \in O_{s_i}} n_j \theta(\|p_i - p_j\|) \psi({n_i}^k, n_j)}{\sum_{s_j \in O_{s_i}} \theta(\|p_i - p_j\|) \psi({n_i}^k, n_j)}. \quad (17)$$

Метод LOP итеративно проецирует множество точек $P = \{p_i\}_{i \in I} \subset R^3$, на множество исходных точек $Q = \{q_j\}_{j \in J} \subset R^3$, для каждой k-й итерации $P^{k+1} = G(P^k), k = 0,1 \dots m$, таким образом, чтобы выполнялось (18):

$$G(P^k) = \underset{P=\{p_i\}}{\operatorname{argmin}} \{\sum_{i \in I} \sum_{j \in J} \|p_i - q_j\| \psi(n_i, {p_i}^k - q_j) + \sum_{i \in I} \lambda_i \sum_{i' \in I \setminus \{i\}} \eta(\|p_i - {p_{i'}}^k\|) \theta(\|{p_i}^k - {p_{i'}}^k\|)\}, \quad (18)$$

где $\eta(r) = -r$, $\theta(r) = e^{-\frac{r^2}{\sigma_p{}^2}}$, а $\psi\left(n_i, {p_i}^k - q_j\right) = e^{-\frac{({n_i}^T({p_i}^k - q_j))^2}{\sigma_p{}^2}}$.

Параметр λ_i — является балансирующим критерием, зависящим от μ, и вычисляется во время итеративного решения задачи минимизации. Решение сводится к итеративному вычислению координат точек P^{k+1} по формуле (19):

$$ {p_i}^{k+1} = \sum_{j \in J} q_j \frac{\alpha_{ij}^k}{\sum_{j \in J} \alpha_{ij}^k} + \mu \sum_{i' \in I \setminus \{i\}} ({p_i}^k - {p_{i'}}^k) \frac{\beta_{ii'}^k}{\sum_{i' \in I \setminus \{i\}} \beta_{ii'}^k}, \quad (19) $$

где $\alpha_{ij}^k = \frac{\psi(n_i, {p_i}^k - q_j)}{\|{p_i}^k - q_j\|}$, $\beta_{ii'}^k = \frac{\theta(\|{p_i}^k - {p_{i'}}^k\|)}{\|{p_i}^k - {p_{i'}}^k\|} \frac{\partial \eta(\|{p_i}^k - {p_{i'}}^k\|)}{\partial r}$.

Результаты применения одной итерации рассмотренного фильтра изображены на рисунке 8.

Первая фаза рассмотренного алгоритма позволяет восстановить геометрию и направление нормалей в заданном облаке точек, однако потерять некоторую информацию о геометрии на гранях и изменить структуру облака точек. Фильтрация описанным методом может быть полезна при работе с глобальными дескрипторами.

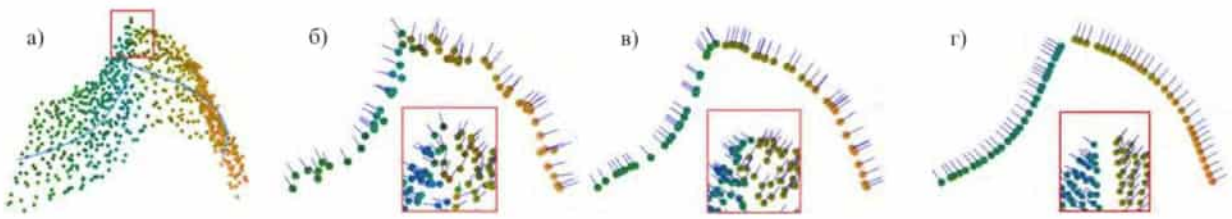


Рисунок 8 — Результат работы первой фазы фильтра [74]: а — исходное облако точек; б — применение алгоритмов PCA для расчета нормалей; в — применение алгоритма сглаживания нормалей; г — реорганизация облака точек с помощью LOP

1.4. Выводы по главе

В главе проведено исследование существующих методов идентификации деталей по облаку точек и методов подготовки облака точек, в ходе которого:

1. На основе проведенного исследования выбрана методика идентификации, использующая глобальный дескриптор CVFH [29],

применимая к решению задачи идентификации детали по облаку точек, полученному в результате лазерного сканирования.

2. В результате исследования выбрана методика подготовки облака точек, основанная на фильтре WLOP и алгоритме сглаживания нормалей [74], позволяющая осуществить сглаживание нормалей и реорганизацию облака, с целью лучшего описания поверхности детали.

Точность идентификации может зависеть от типов деталей, используемых для идентификации. При этом процесс сбора облаков точек деталей с точки контроля является трудоемким и затратным по времени. Отдельной задачей является учет угла поворота детали, с которой было получено облако, для определения причин ошибок при идентификации. Поэтому первой задачей является создание методики эмуляции лазерного сканирования, позволяющей создавать облако точек детали близкое к облаку точек, полученному в результате реального сканирования. Эта задача будет рассмотрена во второй главе.

В выбранных методиках идентификации авторы заявляют хорошие результаты идентификации на реальный облаках точек. Поэтому второй задачей является оценка воспроизводимости результатов применимо к разрабатываемой информационно-измерительной системе и к идентифицируемым типам деталей, анализ полученных результатов и разработка методик их устранения. В третьей главе будет решаться задача идентификации детали по облаку точек.

Третьей задачей является создание информационно-измерительной идентификации детали на конвейерной линии на основе лазерного 3D-сканирования на основе разработанных в работе методик. Описание этой задачи будет представлено в главе 3.

Глава 2. Получение облака точек в информационно измерительных системах идентификации деталей

2.1. Принципы лазерного 3D-сканирования детали, движущейся по конвейеру

Лазерное сканирование широко используется в составе систем контроля на производстве [77]. Рассматриваемая в работе информационно-измерительная система предназначена для идентификации деталей, движущихся по конвейеру, и состоит из нескольких устройств лазерного сканирования, направленных под разными ракурсами к детали. Схема точки контроля, использующей описанную систему, изображена на рисунке 9.

Точность идентификации деталей по облакам точек, зависит от структуры облака точек, сложности геометрии деталей, их расположения и характеристик устройств сканирования в составе ИИС. Возможность эмуляции облака точек, полученного с устройства лазерного сканирования с конкретными характеристиками и взаимным расположением детали и устройства, будут использованы для проверки методик идентификации на заданном наборе деталей без необходимости создания точки контроля и проведения сканирований тестовых деталей.

3D-сканирование с помощью лазера может осуществляться различными методами. Например, в работе [5] лазер используется как шаблон для сопоставления изображений во время сканирования с помощью стереопары. В рассматриваемой точке контроля и далее в работе под лазерным 3D-сканированием подразумевается методика сканирования с использованием триангуляции [2, 4, 6]. Идея данного метода состоит в том, что камера и лазер расположены под некоторым углом друг к другу, направлены на конвейер и жестко зафиксированы в пространстве. Параметры их взаимного расположения в пространстве известны, также известна скорость и вектор движения ленты конвейера. Камера детектирует видимое пятно луча лазера на детали. По известным параметрам взаимного расположения камеры и лазера

восстанавливаются трехмерные координаты точек пятна луча лазера на детали в пространстве. Параметры перемещения ленты конвейера и частота получения кадров с камеры позволяют итеративно восстановить полное облако точек детали.

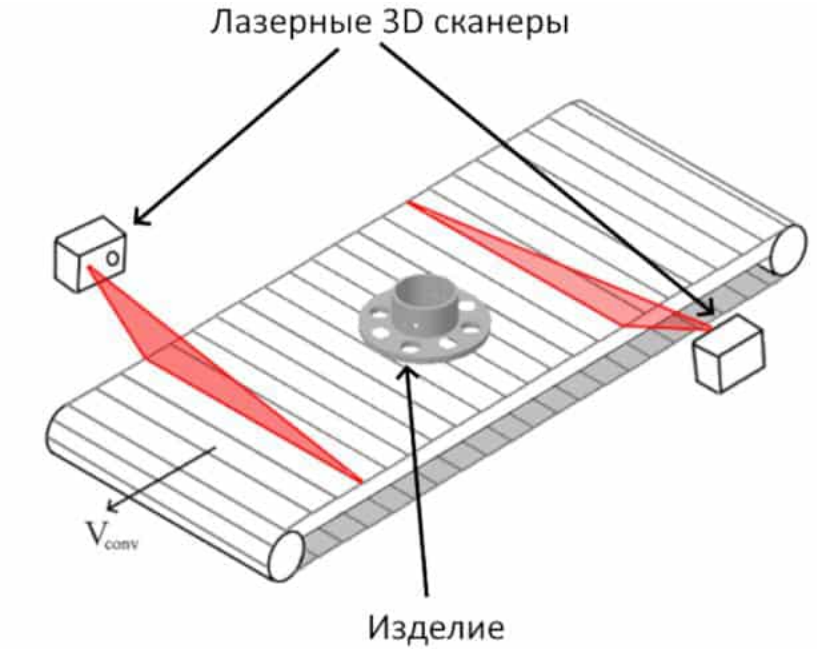


Рисунок 9 — Схема точки контроля с использованием устройств лазерного сканирования

Основными элементами системы сканирования являются камера, представленная моделью «камера-обскура» (pinhole camera) [78], и лазер, проецирующий шаблон-линию, расположенные таким образом, что расстояние $b \in R$ между камерой и лазером известно, а также известны угол наклона плоскости лазера к плоскости XoZ камеры $\eta \in R$ и угол наклона лазера к оптической оси камеры $\theta \in R$. Параметры камеры-обскура представлены четырьмя величинами, заданными в пикселях (f_x, f_y, c_x, c_y), где $f_x \in R$ и $f_y \in R$ — фокусные расстояния по оси X и Y соответственно, а $c_x \in R$ и $c_y \in R$ координаты расположения оптической оси камеры на изображении. Калибровка [79] выполняется таким образом, чтобы $f_x = f_y = f$. Схема модели лазерного сканера изображена на рисунке 10.

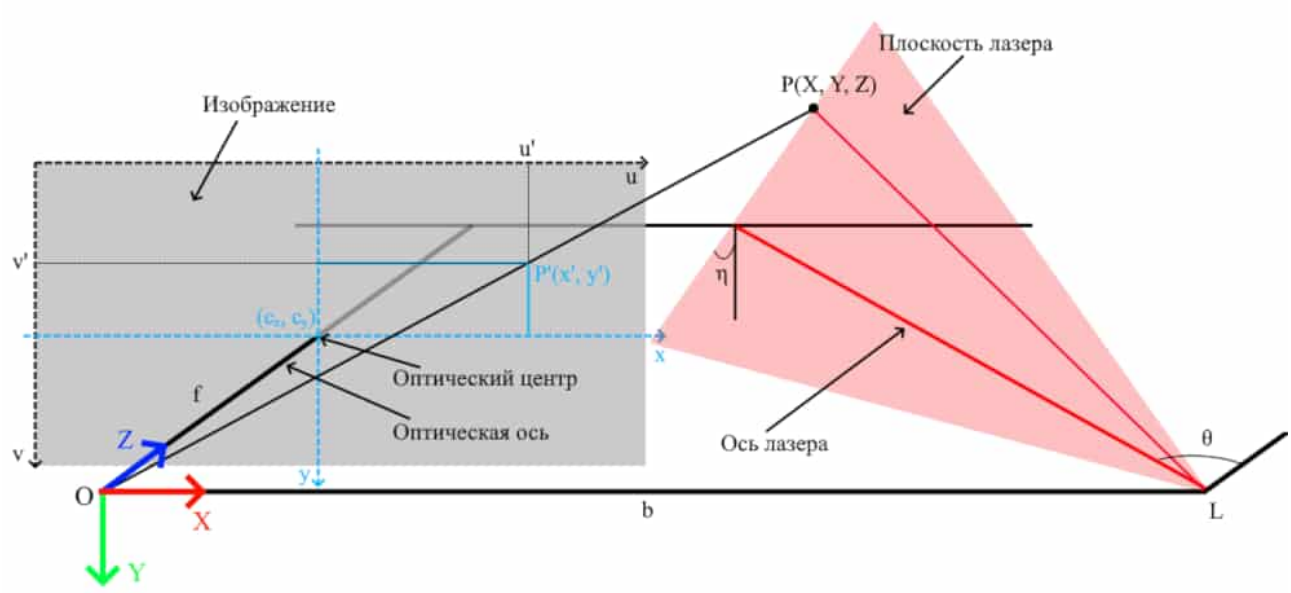


Рисунок 10 — Схема модели лазерного сканера

Из параметров камеры формируется матрица A (20), с помощью которой осуществляется взаимный переход от трехмерных координат точки $(X, Y, Z) \in R^3$ в системе координат камеры к пикселям на изображении камеры $(u, v) \in R^2$ (21):

$$\mathrm{A} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{20}$$

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}. \tag{21}$$

Пусть центр лазера выделен на изображении в точке с координатами $T(u', v')$, тогда задача восстановления трехмерных координат точки, сводится к поиску точки пересечения вектора $OP'(u', v')$ и плоскости лазера, где согласно модели камеры:

$$OP'(u', v')^T = \begin{bmatrix} \mathrm{x} \\ \mathrm{y} \\ f \end{bmatrix} = \begin{bmatrix} u' - c_x \\ v' - c_y \\ f \end{bmatrix}. \tag{22}$$

Нормаль плоскости лазера $N \in R^3$ рассчитывается путем последовательного поворота вектора $L = [1, 0, 0]$ вокруг оси Z на угол η и вокруг оси Y на угол θ (23):

$$N = [N_x \quad N_y \quad N_z] = R_y(\theta)R_z(\eta)L, \tag{23}$$

где $R_z(\eta) = \begin{bmatrix} \cos\eta & -\sin\eta & 0 \\ \sin\eta & \cos\eta & 0 \\ 0 & 0 & 1 \end{bmatrix}$, а $R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$.

Точка $P(u', v') \in R^3$ пересечения плоскости лазера и вектора $OP'(u', v')$ рассчитывается по формуле (24). Эта точка пересечения и является восстановленными координатами луча лазера на поверхности детали:

$$P(u', v') = \frac{bN_x}{N \cdot OP'(u', v')^T} \cdot OP'(u', v'), \tag{24}$$

где · — матричное произведение.

Поиск пятна луча лазера на детали на изображении осуществляется с помощью следующего алгоритма:

1. С помощью фильтров и адаптивного порогового преобразования выделяется маска пятна лазерного луча, после чего исходное изображение преобразуется в формат оттенков серого.
2. По полученной маске на исходном изображении удаляются пиксели, не содержащие видимое пятно лазерного луча.
3. Для каждой строки пикселей, содержащей пятно луча лазера, рассчитывался центр масс. Полученный центр масс являлся координатой u', а номер столбца координатой v' центра пятна лазерного луча на изображении. После чего по формуле (24) рассчитывались трехмерные координаты точек луча в пространстве.
4. Полученные точки переводились в координатную систему точки контроля. По известной скорости и вектору перемещения детали полученные точки размещались в общем облаке точек детали.

Результат работы описанного алгоритма изображен на рисунке 11.

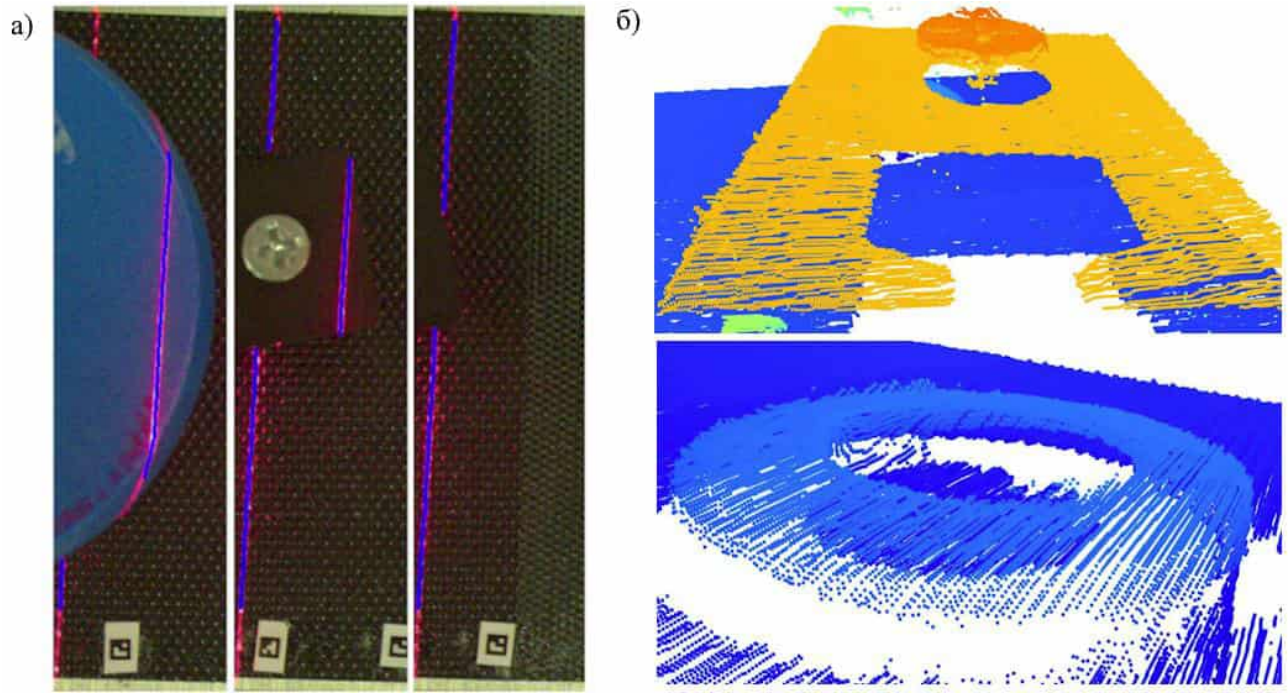


Рисунок 11 — Результат работы алгоритма лазерного сканирования: а — процесс выделения луча лазера на изображении (синие точки — центры луча лазера); б — структура общего облака точек

В результате исследования структуры облака точек, полученного по описанным принципам лазерного сканирования детали, движущейся по конвейеру, выделены следующие особенности, которые необходимо учитывать при анализе существующих методик эмуляции сканирования:

1. Облако точек представлено «срезами», поэтому зачастую разрежено по оси движении детали. Расстояние между «срезами» зависит от частоты получения кадров и скорости движения конвейера.
2. Облако точек покрывает только видимую поверхность детали с камеры, на которую попадает лазер. Выпуклые части детали могут перекрывать луч или обзор с камеры.
3. Значимое число шумов и выбросов связанно с неточностью определения центров луча лазера. Они в основном возникают на границах детали, в связи с уменьшением ширины видимого луча, или в местах резкого перехода поверхностей, когда части лазерного луча попадают на разные поверхности.

Выделенные особенности процедуры лазерного сканирования будут учтены при проведении анализа существующих методов эмуляции процесса лазерного 3D-сканирования.

2.2. Анализ существующих методов эмуляции лазерного 3D-сканирования

Методы эмулирования облака точек для проверки алгоритмов достаточно широко рассмотрены в работах. Из общего набора были выделены следующие методы.

В библиотеке Open3D [80], написанной на языках программирования C++ и Python, являющейся одной из наиболее популярных библиотек для обработки данных в трехмерном пространстве, представлен метод для генерирования облака точек по полигональной модели. Этот метод, размещает заданное количество точек на поверхности детали, используя распределение Пуассона [81], что позволяет построить облако точек, покрывающее всю поверхность детали.

В работе [82] рассмотрен метод эмуляции лазерного сканирования с помощью виртуального сканера установленного на виртуальной руке робота. В качестве объекта сканирования использовалась полигональная модель объекта. В качестве излучателя использовался виртуальный лазер, который проецировал луч в виде линии. После этого происходил расчет пересечения полигональной модели с виртуальным лучом лазера. Собранные точки на модели компоновались, учитывая смещение руки робота, для получения единого массива точек сканирования.

В работе [83] рассмотрен метод эмулирования получения облака точек в процессе сканирования с помощью ToF камеры или сенсоров, использующих модели с триангуляцией. Основная идея заключалась в поиске пересечений лучей излучателя с полигональной моделью объекта [84], после чего осуществлялась проверка видна ли найденная точка в приемнике (камере). Далее к координатам найденной точки добавлялась ошибка сенсора, указанная в

процентах. Координата точки пересечения луча излучателя с полигоном перечитывались по формуле (25):

$$P = 0 + \vec{d} \cdot h \cdot rand\left(1 - \frac{e}{100}, 1 + \frac{e}{100}\right), \quad (25)$$

где h — расстояние до точки пересечения, e — ошибка сенсора в процентах, $rand$ — функция выбора случайного значения из заданного диапазона, $\vec{d}$ — вектор направления на точку пересечения, 0 — точка начала луча.

На рисунке 12 изображены результаты работы описанного алгоритма для виртуального сканнера с различными заданными величинами процента ошибки сенсора.

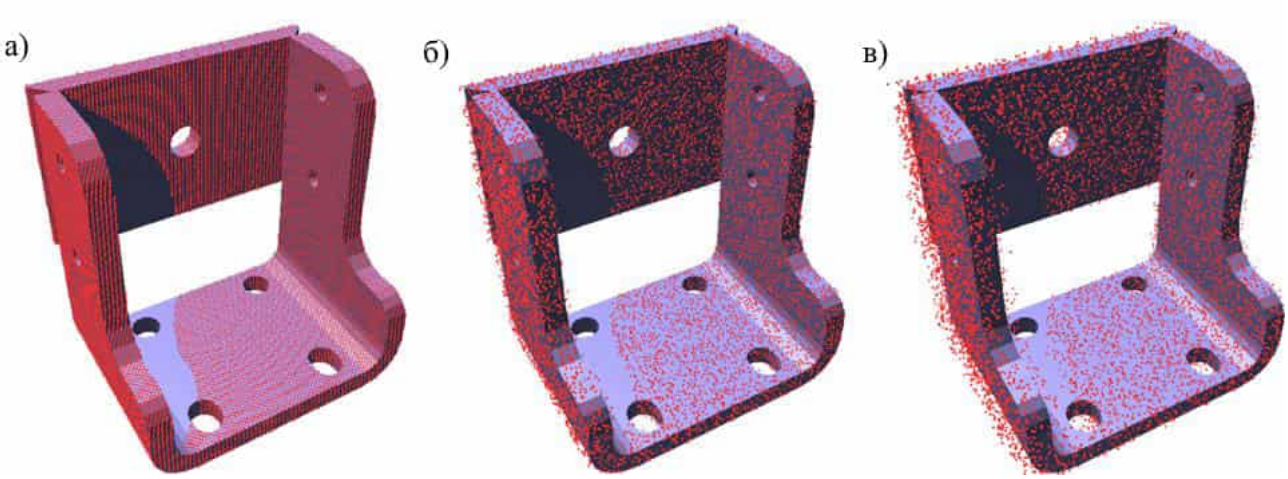


Рисунок 12 — Результат работы алгоритма эмуляции сканирования [83]: а — 0% ошибки; б — 1% ошибки; в — 3% ошибки

Отдельным классом методов эмуляции можно выделить работу [85]. Идея метода заключатся в полном рендеринге сцены сканирования с помощью программы, для работы с трехмерной графикой Blender. Авторы описывают процесс полной эмуляции сканирования, при которой задаются все источники освещения, цвета и свойства поверхности, параметры окружения и заднего фона. По заданным параметрам рассчитывается изображение с камеры входящей в состав системы лазерного сканирования, содержащее луч лазера, по которому уже восстанавливается трехмерное положение точек лазера в пространстве. В этом методе эмуляции воспроизводится ошибка определения центра луча и структура облака, связанного с перемещением детали по конвейеру.

Сравнение рассмотренных методов эмуляции лазерного сканирования представлено в таблице 2.

Таблица 2 — Сравнение методов эмуляции лазерного 3D-сканирования

Метод эмуляции	**Недостатки применимо к работе**
Методы из библиотеки Open3D [80]	Покрывают облаком точек всю поверхность модели детали, что зачастую очень затруднительно получить во время реального сканирования, или вообще невозможно, если производится сканирование объекта с одного ракурса. В полученных облаках точек отсутствуют выбросы и шумы, нельзя задать структуру точек на поверхности, обусловленную выбранным методом 3D-сканирования
Виртуальный сканер на виртуальной руке робота [82]	Не учитываются ошибки измерения или шумовые эффекты, вызванные шириной лазерного луча и ошибками определения его центров.
Метод эмулирования с помощью трассировки лучей [83]	Описанный метод хоть и учитывает ошибку сенсора на разных дистанциях, но геометрия этих ошибок не является естественной. Описанный метод не воспроизводит перемещение детали в пространстве, а также не учитывает толщину лазерного луча, а следовательно ошибки, связанные с определением его центра.
Полная отрисовка сцены сканирования [85]	Избыточная сложность настройки, скорость работы, необходимость проведения калибровки виртуальной системы камера-лазер, что не позволит осуществлять эмуляцию большого количества сканирований

Рассмотренные методы обладают недостатками, которые не позволяют использовать их без доработок. Подход к формированию изображения с камеры в работе [85], позволяющий воспроизвести ошибки позиционирования центра видимого пятна луча лазера, является избыточным и сложным в настройке для проведения сканирования, при этом накладывает необходимость в разработке алгоритмов выделения луча, который напрямую влияет на полученные результаты эмуляции. В работе [83], несмотря на скорость обработки, невозможно достоверно воспроизвести облако точек реального сканирования, полученное с устройства сканирования, рассмотренного ранее. Необходимо разработать методику эмуляции сканирования, использующую подходы из рассмотренных работ, позволяющую при этом воспроизводить облако точек близкое к полученному с реального устройства сканирования без необходимости отрисовки всей сцены сканирования и проведения виртуальных калибровок оборудования.

2.3. Разработка методики эмуляции лазерного 3D-сканирования

Для устранения недостатков, выделенных в результате анализа существующих методов, разработана методика эмуляции сканирования, основная идея которой заключается в формировании изображения с камеры устройства лазерного сканирования, содержащего только видимое пятно луча лазера на поверхности детали, с последующим выделением центра пятна лазерного луча и восстановлением положения его центров в пространстве по алгоритму, описанному в главе 2.1.

Для проведения эмуляции используются параметры, описанные в таблице 3. Схема эмулируемой системы сканирования с параметрами из таблицы изображена на рисунке 13.

Таблица 3 — Параметры эмуляции лазерного сканирования

Параметр	**Описание**
Свойства модели детали:	
V	Матрица координат вершин цифровой модели
P	Матрица полигонов (наборов вершин) из матрицы V
$m_d \in [0,1]$	Свойство материала отражать диффузную составляющую (модель Блинна-Фонга).
$m_s \in [0,1]$, $\beta \in R$	Свойство материала отражать бликовую составляющую и параметр блеска (модель Блинна-Фонга)
Свойства камеры:	
$\mathrm{h}_{\boldsymbol{p}} \in N, w_p \in N$	Количество пикселей по вертикали и горизонтали
$f_r \in R$	Фокусное расстояние (мм)
$l_p \in R$	Размер пикселя (мм)
$O_{cam} \in R^3$	Положение камеры в координатной системе сцены
$\overrightarrow{V_{cam}} \in R^3$	Вектор направления оптической оси камеры в координатной системе сцены
$\eta_{cam} \in [0,2\pi)$	Угол поворота камеры вокруг оптической оси
$\vartheta \in R$, $\delta \in R$	Коэффициент шума матрицы и чувствительности сенсора
Свойства системы камера-лазер:	
$b \in R$	Расстояние между осями камеры и лазера (мм)
$\theta \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$	Угол поворота плоскости камеры к оси камера-лазер
$\eta \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$	Угол поворота плоскости лазера вокруг оси лазера относительно Y оси камеры

$\sigma_l \in R$	Среднеквадратическое отклонение яркостей точек луча лазера (в качестве допущения принята за константу)
Параметры эмуляции:	
$i_t \in N$	Количество итераций эмуляции
$\overrightarrow{V_{step}} \in R^3$	Вектор шага перемещения модели детали между итерациями в координатной системе сцены (мм).

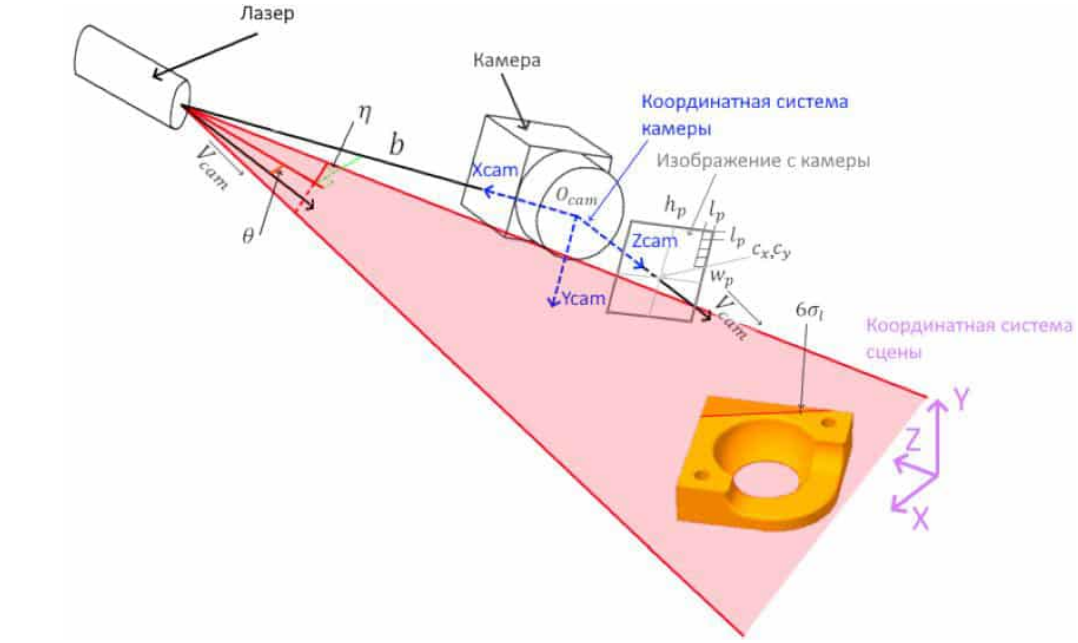


Рисунок 13 — Схема эмулируемой системы сканирования

В основе разработанной методики лежит алгоритм, состоящий из десяти последовательных шагов, схема которого представлена на рисунке 14.

На первом этапе согласно модели камеры-обскура составляются уравнения лучей $L(t)$ (30), проходящие через центр каждого пикселя на изображении. Для этого строятся лучи $J(t)$ направленные к центрам пикселей (26), при условии, что ось камеры совпадает с осью X координатной системы сцены, после чего выполняется поворот построенных лучей на угол η_{cam} вокруг оси X с помощью матрицы поворота $R(\eta_{cam})$, далее выполняется поворот лучей так, чтобы оптическая ось камеры совпала с заданным вектором $\overrightarrow{V_{cam}}$. После поворотов лучи перемещаются в точку O_{cam}:

$$J(t) = \left[f_r \quad -\left(j - \frac{h_p}{2}\right) * l_p \quad \left(i - \frac{w_p}{2}\right) * l_p\right]_{n\times 3} \cdot t, \tag{26}$$

где $i = 1..w_p$, $j = 1..h_p$, а $n = w_p \cdot h_p$.

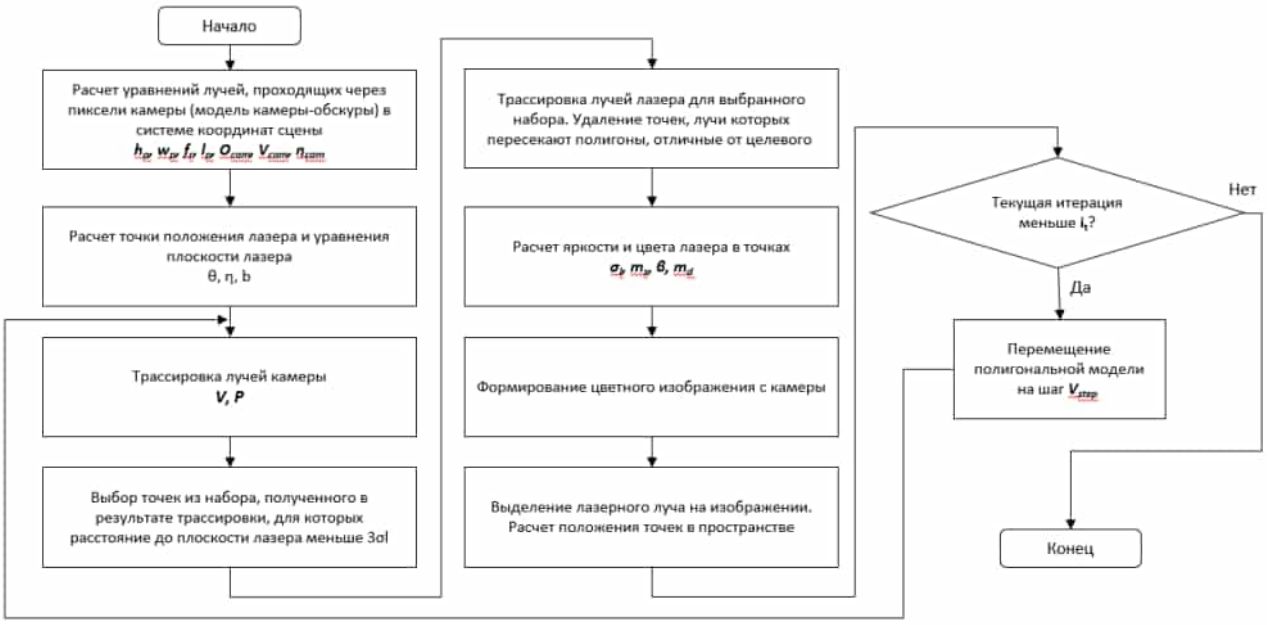


Рисунок 14 — Алгоритм эмуляции лазерного сканирования

Матрица поворота M (29) оптической оси для совмещения оптической оси с вектором $\overrightarrow{V_{cam}}$, рассчитывается как матрица поворота вокруг вектора $\overrightarrow{V_{rot}}$ (27) на угол γ (28):

$$\overrightarrow{V_{rot}} = [1 \quad 0 \quad 0] \times \overrightarrow{V_{cam}} = \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix}, \tag{27}$$

$$\gamma = \operatorname{arctg} \frac{|\overrightarrow{V_{rot}}|}{[1 \quad 0 \quad 0] \cdot \overrightarrow{V_{cam}}^T}, \tag{28}$$

$$M = \begin{bmatrix} \cos\gamma + (1-\cos\gamma){V_x}^2 & (1-\cos\gamma)V_xV_y - V_z\sin\gamma & (1-\cos\gamma)V_xV_z + V_y\sin\gamma \\ (1-\cos\gamma)V_xV_y + V_z\sin\gamma & \cos\gamma + (1-\cos\gamma){V_y}^2 & (1-\cos\gamma)V_yV_z - V_x\sin\gamma \\ (1-\cos\gamma)V_xV_z - V_y\sin\gamma & (1-\cos\gamma)V_zV_y + V_x\sin\gamma & \cos\gamma + (1-\cos\gamma){V_z}^2 \end{bmatrix}. \tag{29}$$

Уравнения лучей $L(t)$:

$$L(t) = O_{cam} + MR(\eta_{cam}) \cdot J(t). \tag{30}$$

На втором шаге рассчитывается положение лазера (34) и составляется уравнение плоскости лазера в координатной системе сцены (35). В отличие от

(23), описанной в главе 2.1., нормаль к плоскости лазера (33) рассчитывается через направляющие вектора L_1 и L_2 (31, 32), так как эти вектора будут использоваться при фильтрации облака точек:

$$L_1 = MR(\eta_{cam})R_y(\theta)R_x(\eta)\begin{bmatrix}1\\0\\0\end{bmatrix}, \tag{31}$$

$$L_2 = MR(\eta_{cam})R_y(\theta)R_x(\eta)\begin{bmatrix}0\\1\\0\end{bmatrix}. \tag{32}$$

Нормаль плоскости лазера вычисляется по формуле (33):

$$N = \begin{bmatrix}N_x\\N_y\\N_z\end{bmatrix} = L_1 \times L_2. \tag{33}$$

где × — векторное произведение.

Положение лазера в координатной системе сцены рассчитывается по формуле (34):

$$O_l = \begin{bmatrix}O_{l_x}\\O_{l_y}\\O_{l_z}\end{bmatrix} = MR(\eta_{cam})\begin{bmatrix}0\\0\\b\end{bmatrix} + O_{cam}. \tag{34}$$

Уравнение плоскости лазера примет вид:

$$A_{las}x + B_{las}y + C_{las}z + D_{las} = 0, \tag{35}$$

где $D_{las} = -A_{las}O_{l_x} - B_{las}O_{l_y} - C_{las}O_{l_z}$.

На третьем шаге с помощью алгоритма трассировки лучей [84], выполняется поиск пересечения лучей камеры (30), полученных на первом шаге, с полигонами цифровой модели детали. Если луч пересекает несколько полигонов, то выбирается ближайший к точке начала луча. В результате работы алгоритма формируются три матрицы: $T'_{m\times 3} = \{t'_i : t'_i \in R^3\}$ — матрица координат точек пересечения лучей с полигонами модели, $Q'_{m\times 2} = \{q'_i =$

(u, v): $q'_i \in N^2$ } — матрица координат пикселей, лучи которых пересекают полигоны цифровой модели в точках t'_i, $I' \in N^m$ — вектор индексов полигонов, которые пересекает луч для пикселя с координатами q'_i в точке t'_i.

На четвёртом шаге выбираются те видимые точки матрицы T', которые расположены не дальше, чем $3\sigma_l$ от плоскости лазера $T_{n\times 3} = \{t_i = t'_i : dist(t'_i) < 3\sigma_l\}$. Для этих точек формируются матрицы $Q_{n\times 2} = \{q_i = (u, v) : q_i \in N^2\}$ и $I \in N^n$ из матриц Q' и I', где n — число точек, расположенных на расстоянии менее $3\sigma_l$ от плоскости лазера, а расстояние до плоскости рассчитывается по формуле (36):

$$dist(t'_i) = \left| [t'_i \mid 1] \cdot \left[\frac{A_{las}}{|N|} \quad \frac{B_{las}}{|N|} \quad \frac{C_{las}}{|N|} \quad \frac{D_{las}}{|N|} \right]^T \right|. \tag{36}$$

На пятом шаге выполняется трассировка лучей лазера $TL(t)$ (37), составленных для точек $T_{n\times 3}$ (Рисунок 15). Это необходимо, чтобы определить, попадает ли луч лазера в выбранные на четвертом шаге точки T. Уравнение лучей $TL(t)$ принимает вид:

$$TL_i(t) = O_l + (T_i - O_l) \cdot t, \tag{37}$$

где $i = 1..\text{n}$.

Аналогично третьему шагу выполняется трассировка лучей $TL(t)$, для которых формируются матрицы: $T_{laser_{n\times 3}}$, $Q_{laser_{n\times 2}}$ и $I_{laser} \in N^n$. Из набора точек $T_{n\times 3}$ и соответствующих ему $Q_{n\times 2}$, $I \in N^n$ формируется наборы $T_{f_{k\times 3}}$, $Q_{f_{k\times 2}}$ и $I_f \in N^k$ путем удаления из исходного набора тех точек, для которых не выполняется условие (38):

$$I_{laser_i} = I_i, \tag{38}$$

где $i = 1..n$, а k — число точек, для которых выполняется условие (38).

В результате получен набор видимых точек на поверхности объекта и соответствующие этим точкам лучи камеры и лазера, координаты пикселей и

индексы полигонов цифровой модели в которых расположены полученные точки.

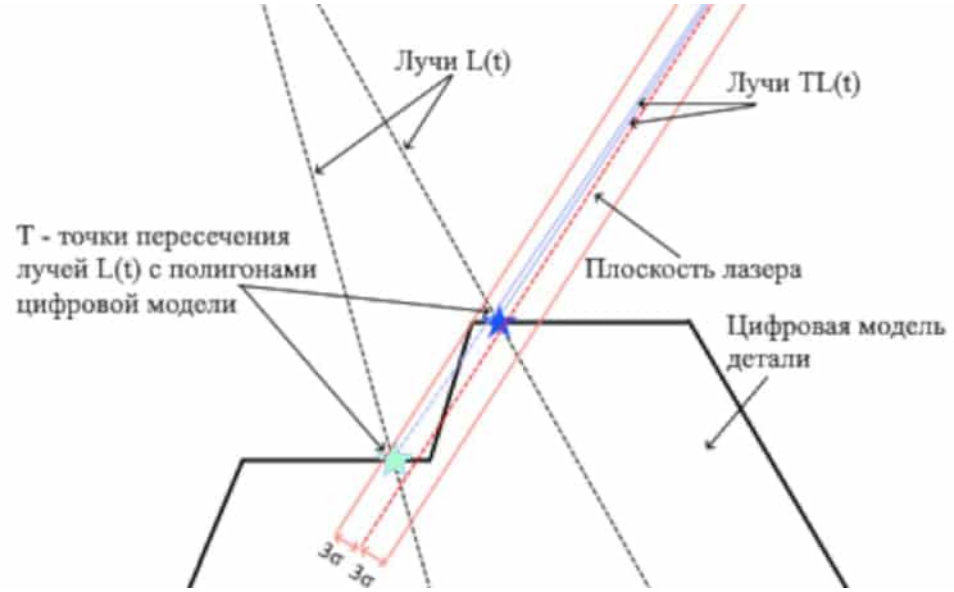


Рисунок 15 — Выбор точек после трассировки лучей камеры и лазера

На шестом шаге для каждой точки T_f рассчитывается яркость. В методике используется модель трехкомпонентного освещения RGB [86] и модель освещения Блинна-Фонга [87], для эмуляции эффектов бликов. Предполагается, что свет с интенсивностью F падает и отражается от поверхности с интенсивностью I. Отражение каждого источника освещения может быть представлен в виде суммы трех составляющих освещения: диффузной, фоновой и зеркальной:

$$I_b = I_a + I_d + I_s, \tag{39}$$

где I_a- яркость, рассчитанная по модели фонового освещения, I_d — яркость, рассчитанная по диффузной модели освещения и I_s — яркость, рассчитанная по бликовой модели освещения.

Если на сцене присутствует несколько источников освещения, то яркость рассчитывается как сумма яркостей всех источников освещения (40):

$$IL = \sum_{j=1}^{n} I_{b\,j}, \tag{40}$$

где n — число источников освещения на сцене.

В методике применяется следующее допущение: если в процессе обработки удалось выделить пятно луча лазера на изображении, значит удалось выделить составляющую освещения лазера $I_L \in IL$ из (40). Это значит, что форма луча на изображении определяется именно этой составляющей. Поэтому при моделировании рассчитываться яркость в пикселе принадлежащих лазерному лучу только с учетом влияния лазера как источника освещения, при этом лазер не вносит в освещение фоновую составляющую, поэтому яркость пикселя рассчитывается на основе бликовой (43) и диффузной (42) составляющих лазера по формуле (41):

$$I_L(i) = I_d(i) + I_s(i). \tag{41}$$

Диффузная составляющая рассчитывается по формуле (42):

$$I_d = m_d \cdot L_d \cdot \max\{0, (\vec{s}, \vec{n})\}, \tag{42}$$

где $\vec{s}$ — единичный вектор направления на источник света (обратный вектор направления советующего луча из $TL_i(t)$ (37)), $\vec{n}$ — единичный вектор нормали поверхности, $L_d \in [0, 1]$ — интенсивность диффузной составляющей источника освещения, $m_d \in [0, 1]$ — свойство материала отражать диффузную составляющую освещения (таблицу значений коэффициента m_d для типовых материалов представлена в работах [86, 88]).

Бликовая составляющая рассчитывается по формуле (43):

$$I_s = m_s \cdot L_s \cdot \max\left\{0, \left(\vec{h}, \vec{n}\right)^{\beta}\right\}, \tag{43}$$

где $\vec{n}$ — единичный вектор нормали поверхности, $L_s \in [0, 1]$ — интенсивность бликовой составляющей источника освещения, $m_s \in [0, 1]$ — свойство материала отражать бликовую составляющую освещения, β — параметр блеска (таблицу значений коэффициента m_s, β для типовых материалов представлена в

работах [86, 88]). Вектор $\vec{h}$ — единичный вектор полупути, рассчитывается по формуле (44):

$$\vec{h} = \frac{\vec{v}+\vec{s}}{\|\vec{v}+\vec{s}\|}, \quad (44)$$

где $\vec{v}$ — вектор направления на наблюдателя (вектор направления советующего луча из $L(t)$ (30).

Интенсивность источника освещения рассчитывается с использованием функции Гаусса и зависит от удаленности видимой точки лазерного луча от плоскости лазера и заданного коэффициента шума. Яркость для i-й видимой точки лазера рассчитывается по формуле (45):

$$L_s(i) = L_d(i) = e^{-\frac{T_f(i)^2}{2\sigma_l{}^2}} * (1 + 2\vartheta(rand - 0{,}5)), \quad (45)$$

где rand — случайная величина из интервала [0, 1], а ϑ — коэффициент шума.

На седьмом шаге происходит формирование цветного изображения с камеры. Для этого вычисляется яркость каждого пикселя $Y(u, v)$ изображения с камеры по формуле (46), которая зависит от полученного финального набора видимых точек лазера и коэффициента чувствительности матрицы:

$$Y(u,v) = \begin{cases} 0, \text{если } (u,v) \notin Q_f \\ 0, \text{если } I_L(i) < \delta \\ I_L(i) \end{cases}, \quad (46)$$

где i — позиция вектора (u, v) в матрице Q_f. В результате на седьмом шаге будет сформировано изображение с камеры, содержащая видимый луч лазера на поверхности детали (Рисунок 16).

На восьмом шаге с помощью алгоритма, описанного в главе 2.1. для каждой строки пикселей v', содержащей пятно луч лазера, вычисляется центр масс видимого пятна u'. По заданным параметрам эмуляции по формуле (24) рассчитывается положение центра пятна лазерного луча $P(u', v')$ в координатной системе камеры. Согласно модели камеры, используемой в

модели лазерного сканера переход к системе координат сцены с учетом смещения модели на текущей итерации, производится по формуле (47):

$$P_{scene}(u', v') = MR(\eta_{cam}) \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix} P(u', v') + O_{cam} - \overrightarrow{V_{step}} \cdot iter, \quad (47)$$

где $iter \in i_t$ — номер текущей итерации эмуляции. Полученные точки добавляются в общий набор точек модели.

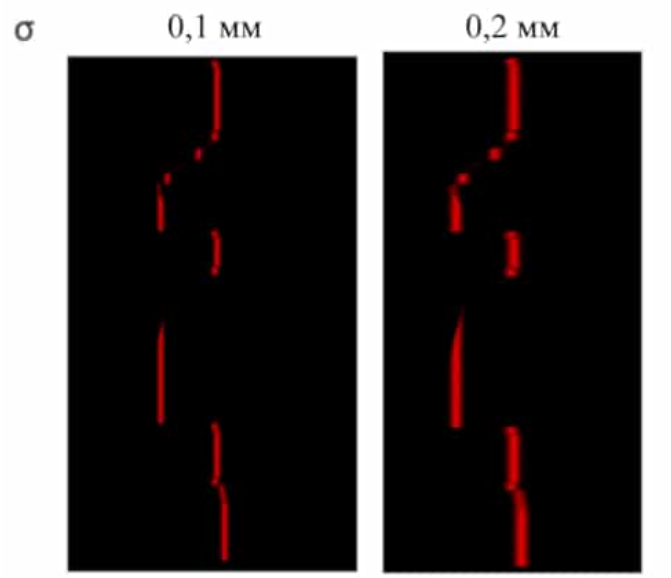


Рисунок 16 — Видимый луч лазера на детали для различных параметров σ_l

На девятом шаге проверятся номер текущей итерации эмуляции. При достижении целевого количества итераций процедура эмуляции завершается.

На десятом шаге производится смещение цифровой модели детали на сцене на заданный вектор $\overrightarrow{V_{step}}$. После этого шаги алгоритма эмуляции повторяются.

В результате работы разработанной методики эмуляции формируется набор точек поверхности детали в координатной системе сцены, воспроизводящее структуру облака точек и ошибки, вызванные определением центров лазерного луча на изображении. Пример облака точек, полученного в результате работы описанной методики, изображен на рисунке 17.

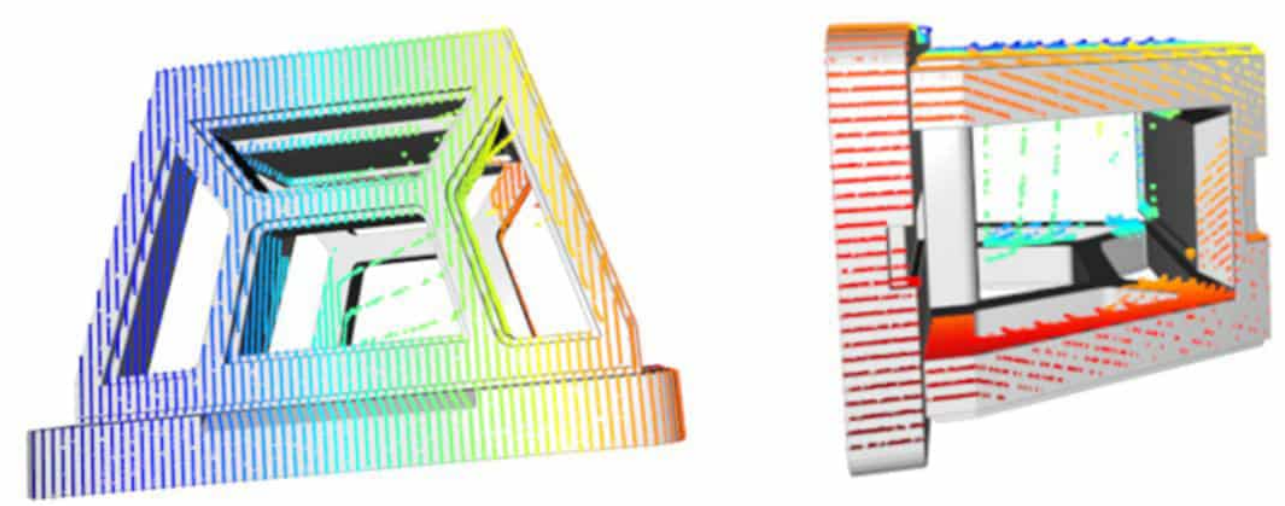


Рисунок 17 — Облако точек, полученное с помощью разработанной методики эмуляции лазерного сканирования, наложенное поверх цифровой модели детали

В главе разработана методика эмуляции сканирования, использующая подходы из работ [83, 85], позволяющая воспроизвести выделенные в главе 2.2. особенности облака точек реального сканирования, а именно:

— представить облако точек «срезами» с заданным шагом, характеризующимся параметром $\overrightarrow{V_{step}}$;

— сформировать точки на поверхности модели объекта по только видимой поверхность с камеры, на которую попадает лазер, что достигается путем повторной трассировки лучей лазера (37);

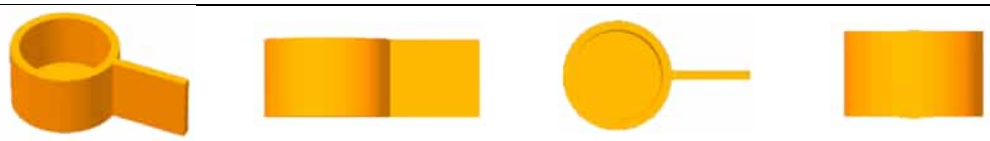
— воспроизвести значимое число шумов и выбросов, которые связаны с неточностью определения центров пятна луча лазера, что достигается путем формирования изображения, содержащее пятно луча лазера на поверхности детали, с выделением его центра и последующим расчётом координат его центров в пространстве.

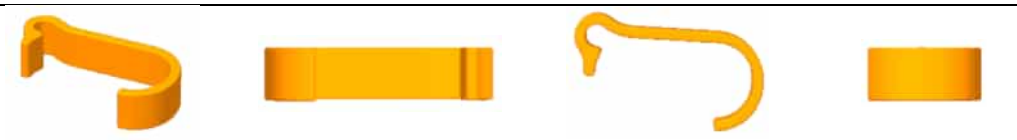
2.4. Тестовый набор деталей для разрабатываемой информационно-измерительной системы идентификации деталей

Для проведения тестирования точности идентификации с помощью информационно-измерительной системы выбран набор из двенадцати моделей

различных деталей с различной геометрией и габаритами. Некоторые детали выбраны таким образом, чтобы с определенных ракурсов они были неотличимы друг от друга. Полный список деталей и их габаритные характеристики приведены в таблице 4.

Таблица 4 — Тестовые детали и их габариты

№	Изображения детали	Габариты мм
1		94×13×25
2		93×21×23
3		73×45×49
4		78×32×49
5		77×46×23
6		48×28×17
7		56×56×17
8		49×38×25

9		38×38×11
10		55×32×13
11		87×24×40
12		89× 62×24

Выбранные детали будут использоваться для проверки точности идентификации разрабатываемой ИИС идентификации деталей на конвейерной линии на основе лазерного 3D-сканирования.

2.5. Прототип системы лазерного сканирования

Для проведения оценки точности идентификации на реальных данных был спроектирован и построен прототип системы лазерного сканирования, состоящий из конвейера, двух устройств лазерного сканирования, направленных под разными ракурсами перпендикулярно к оси движения конвейеру и вычислительного устройства, обрабатывающего данные с устройств сканирования (Рисунок 18). Устройства лазерного сканирования состояли из камеры и лазера, направленного под углом к камере.

Схема взаимодействия устройств в составе прототипа системы сканирования изображена на рисунке 19. ПК в составе точки контроля взаимодействует с камерами лазерных сканеров по USB, получая кадры с меткой времени. По COM порту передаются данные о текущей скорости конвейерной линии. Скорость замеряется с помощью устройства на базе ESP32 с помощью радиального энкодера. Сопоставляя данные о скорости конвейера и метках

времени для кадров с камер, рассчитывается текущее положение детали на конвейере, необходимое для построения общего облака точек.

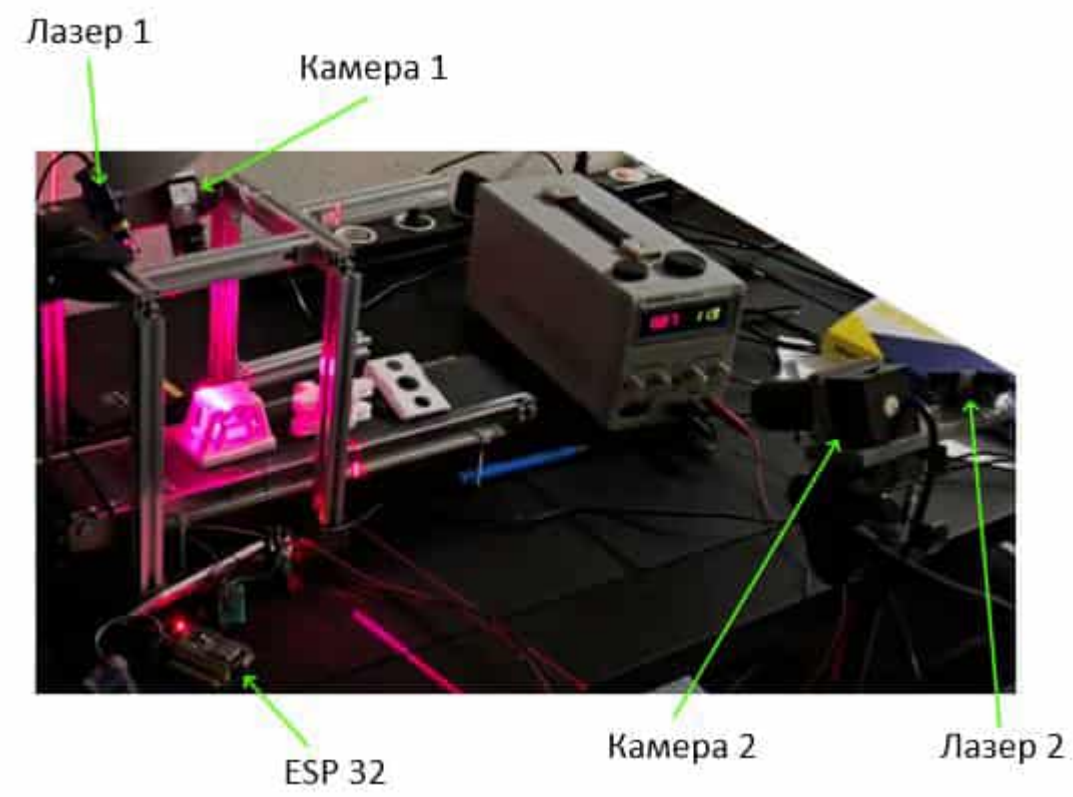


Рисунок 18 — Фотография прототипа системы лазерного сканирования

Для описанного прототипа системы лазерного сканирования разработано ПО, необходимое для калибровки лазерных сканеров и системы в целом, а также для управления процессом сканирования. Программное обеспечение состоит из нескольких отдельных компонентов (схема взаимодействия компонентов в составе ПО для прототипа системы лазерного сканирования изображена на рисунке 19):

1. ПО для калибровки камеры, реализующее алгоритм [79] предназначенное для приведения изображения с камеры к модели «камеры-обскура» путем компенсации линейных и нелинейных искажений. Разработано на языке Python 3 с использованием библиотеки OpenCV 4.
2. ПО для калибровки системы камера-лазер. Принимает на вход калибровочные матрицы и матрицу параметров камеры, рассчитывает

угол θ между лазером и оптической осью камеры и угол наклона плоскости лазера η. Разработан на языке Python3 3 с использованием библиотеки OpenCV 4, реализует алгоритм, описанный в работе [2]. Вид программы калибровки изображен на рисунке 21.

3. ПО для калибровки системы сканирования. Принимает на вход параметры, рассчитанные на прошлых этапах. По калибровочной доске на конвейерной линии восстанавливает плоскость конвейера, по видимой проекции луча лазера находит центр конвейерной линии, рассчитывает матрицу перехода к системе координат конвейера и вектор перемещения конвейерной линии. Рассчитывает расстояние между системами координат устройств сканирования.
4. ПО для управления системой сканирования. Разработан модуль для среды исполнения forte на языке программирования C++ и среды конфигурирования 4diac [89] с использованием подходов из [6, 7, 8]. Модуль получает на вход изображения с камер, параметры скорости конвейера и калибровочные параметры системы, по которым с помощью алгоритма, описанного в главе 2.1. рассчитывает облака точек с каждого устройства сканирования в координатной системе устройства сканирования, удаляет точки, расположенные близко к плоскости конвейера, и сохраняет их в формате CSV.

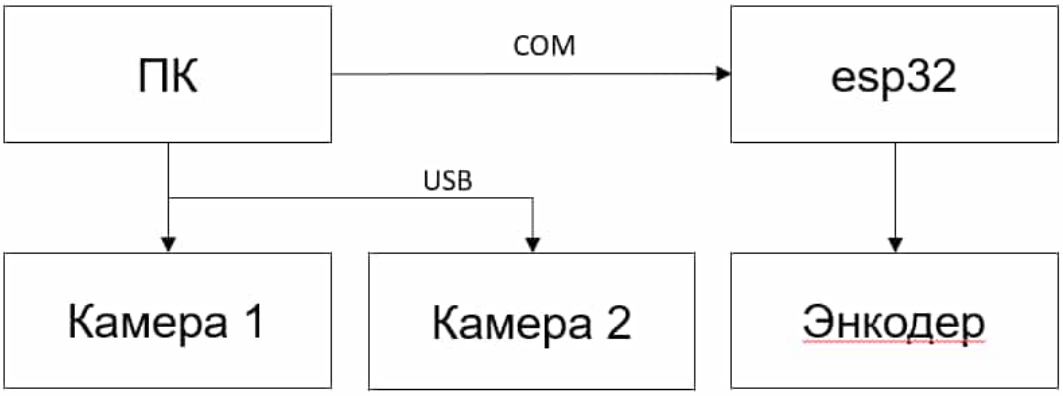


Рисунок 19 — Схема взаимодействия компонентов прототипа системы сканирования

Рисунок 20 — Схема взаимодействия компонентов ПО для прототипа системы сканирования

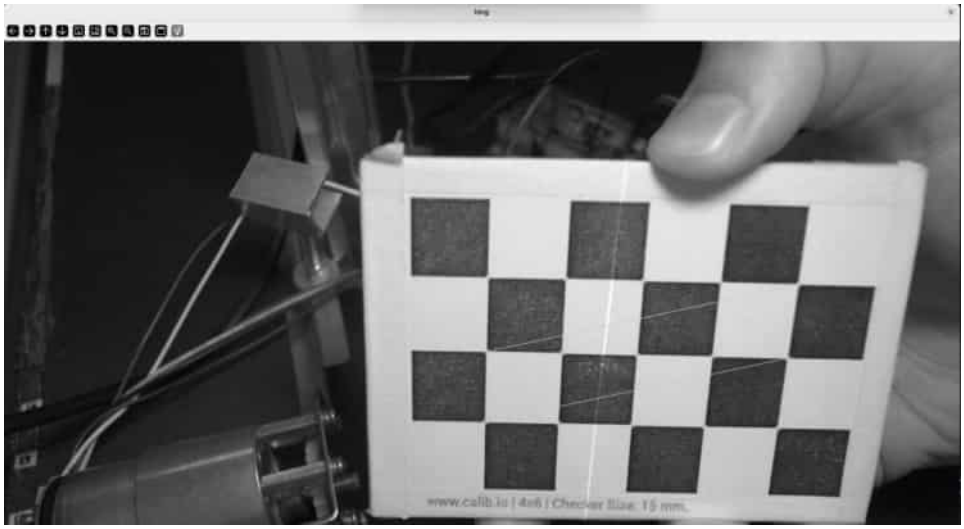


Рисунок 21 — Окно ПО для калибровки системы камера-лазер

Первичное облако точек — облако точек, полученное системы сканирования, до этапа удаления точек, расположенных близко к плоскости конвейера, изображено на рисунке 22.

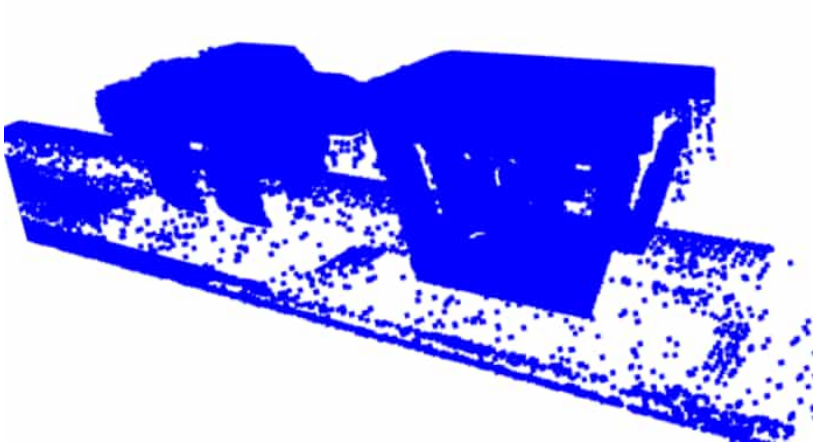


Рисунок 22 — Первичное облако точек, полученное с прототипа системы сканирования

Облака точек, полученные с прототипа системы сканирования будут использоваться как для проверки точности идентификации разрабатываемых методик на реальных данных.

2.6. Оценка качества эмуляции сканирования

Необходимо оценить, насколько облако точек, полученное с помощью разработанной методики эмуляции, соответствует реальному облаку точек, полученному с устройства лазерного 3D-сканирования.

Экспериментальная проверка осуществляется с помощью следующего алгоритма:

1. Параметры устройства сканирования при эмуляции устанавливаются идентичными калибровочным параметрам реального устройства сканирования. Среди этих параметров: размер пикселя, размер матрицы, фокусное расстояние, положение на сцене, углы поворота и наклона лазера, расстояние между оптической осью и осью лазера, вектор перемещения детали на сцене. Описанные параметры известны после проведения калибровки устройства сканирования. Шаг между итерациями рассчитывается как средний шаг между кадрами в процессе реального сканирования.
2. Выполняется поиск положения реальной детали на сцене сканирования, чтобы разместить цифровую модель на сцене во время эмуляции в том же положении как она была расположена во время сканирования. Для этого цифровая модель детали размещается в облаке точек, полученного с устройства реального сканирования. Осуществляется это в четыре этапа. На первом этапе происходит грубое ручное размещение цифровой модели в облаке точек реального сканирования. На втором этапе поверхность цифровой модели представляется с помощью облака точек. На третьем этапе исходное облако точек и облако точек, полученное на втором этапе, вокселизируются с одинаковым размером вокселя. На

четвертом этапе с помощью алгоритма Generalized ICP [90] выполняется точное сопоставление. Процесс поиска положения реальной детали изображен на рисунке 23.

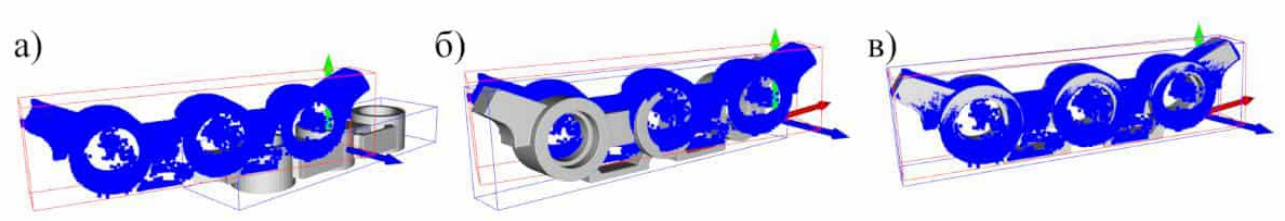


Рисунок 23 — Поиск положения детали на сцене сканирования

3. Для найденного положения детали и заданных параметров сканирования проводится эмуляция сканирования по методике, описанной в главе 2.3. После перехода к координатам сцены, положение полученного облака точек должно совпадать с облаком реального сканирования. Облако точек, полученное с устройства сканирования в сравнении с эмулированным облаком точек изображено на рисунке 24.

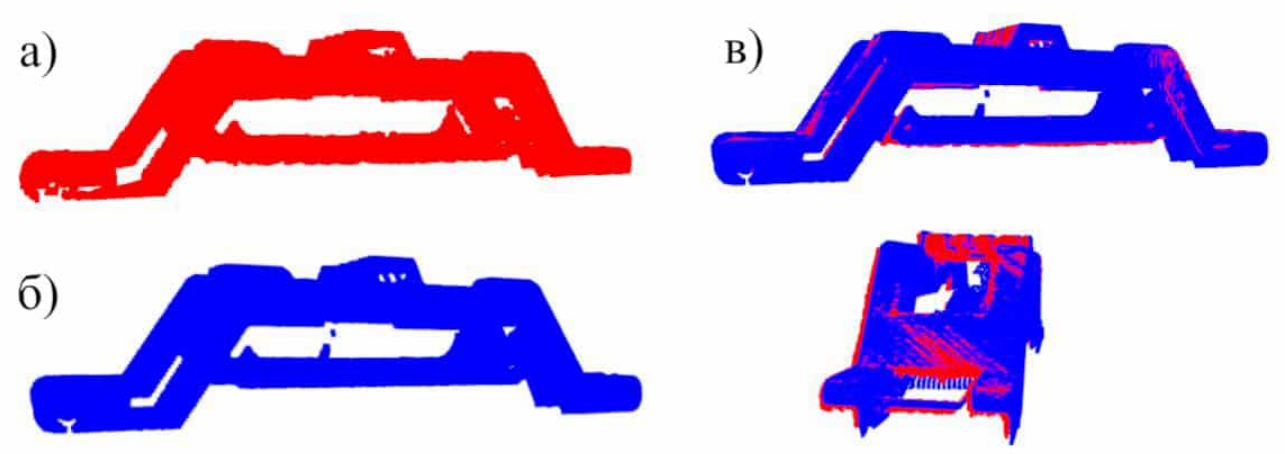


Рисунок 24 — Облако точек реального сканирования в сравнении с эмуляцией: а — Облако точек реального сканирования; б — облако точек, полученное в процессе эмуляции сканирования; в — сравнение облаков точек

4. Производится сравнение облаков точек по двум параметрам: общее отношение числа точек (48) и совпадение точек, покрывающих поверхность объекта:

$$e_n = \frac{n_e}{n_r}, \quad (48)$$

где n_e — число точек в облаке точек, полученном с помощью эмуляции, n_r — число точек в облаке точек, полученном с помощью реального сканирования.

Так как в облаке точек присутствуют шумы и выбросы, а шаг движения детали задан как средний шаг перемещения детали между кадрами, хотя при реальном сканировании он может изменяться, оценка совпадения облаков точек производится по попаданию точек из реального облака и эмуляции в окрестности заданного размера для выбранных точек на поверхности детали.

Выбор точек и размеров окрестностей для осуществления сравнения облаков точек осуществляется с помощью следующего алгоритма:

1. Выбирается размер вокселя a мм.
2. Облака точек реального сканирования и эмуляции объединяются в одно общее облако.
3. Общее облако точек вокселизируется с заданным размером вокселя a мм.
4. Для каждого вокселя, содержащего хотя бы одну точку, общего облака точек с помощью алгоритма k-d деревьев осуществляется поиск ближайших соседей к центру вокселя в эмулированном и реальном облаке точек в окрестности радиуса a мм. Такой радиус выбран для того, чтобы компенсировать расстояние между центром вокселя и точками, расположенными близко к его границам (Рисунок 25).
5. Если для вокселя был найден хотя бы один сосед в обоих облаках точек в окрестности радиуса a мм, то воксель обозначается как «совпадающий», если сосед был найден только для одного из облаков, то обозначается как «несовпадающий».
6. Рассчитывается отношение «совпадающих» вокселей к общему числу вокселей, содержащих хотя бы одну точку (49):

$$R_n(a) = \frac{v_n(a)}{v_n(a)+\bar{v}_n(a)}, \quad (49)$$

где $v_n(a)$ — число «совпадающих» вокселей для выбранного радиуса a, а $\bar{v}_n(a)$ — число «несовпадающих» вокселей.

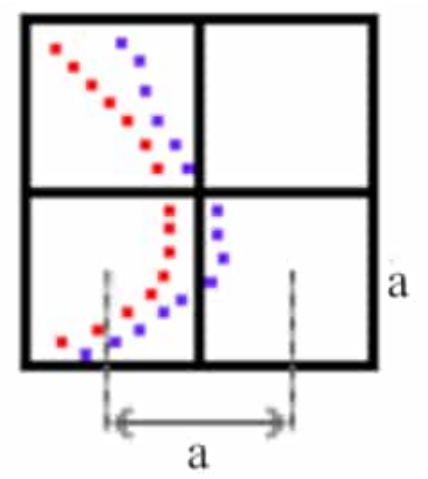


Рисунок 25 — Возможное положение вокселей в общем облаке точек: красные — точки эмуляции сканирования, синие — точки реального облака точек

Результаты сравнения количества точек, представлены в таблице 5. Графики $R_n(a)$ для каждой детали изображены на рисунке 26.

Таблица 5 — Отношение общего числа точек $\boldsymbol{e_n}$

№	1	2	3	4	5	6	7	8	9	10	11	12
$\boldsymbol{e_n}$	0,96	1,02	0,91	1,04	1,07	0,95	0,94	1,03	1,04	1,08	1,04	1,08

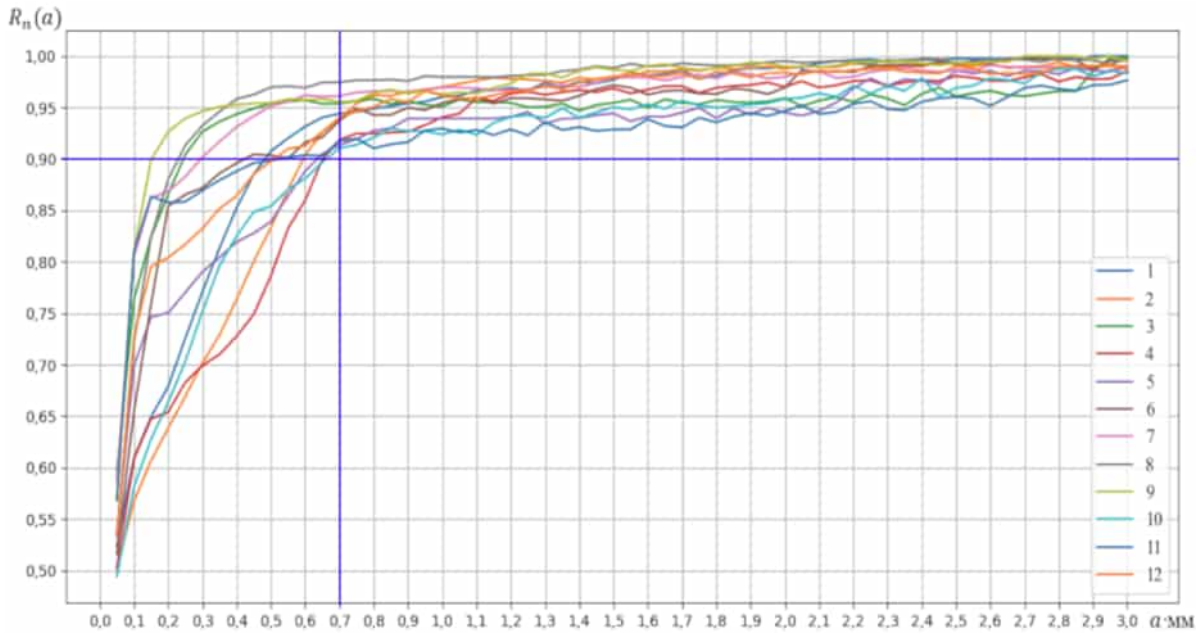


Рисунок 26 — Графики значений $R_n(a)$ для тестовых деталей

Выбранные детали не могут отличаться менее чем на 0,7 миллиметра, поэтому, облака точек выбранных деталей могут быть представлены с помощью воксельной сетки с размером вокселя 0,7 мм. При такой вокселизации максимальное отличие облаков точек, полученных с помощью эмуляции, от облаков точек реальных деталей составит менее 10%, при среднем отличии 6%. Такой процент отличий позволяет использовать облака точек, полученные в результате проведения эмуляции сканирования по разработанной методике для тестирования разрабатываемой ИИС.

С помощью разработанной методики эмуляции получено 2448 облаков точек для 12 тестовых деталей.

2.7. Выводы по главе

В главе описан процесс разработки методики эмуляции лазерного 3D-сканирования для проведения тестирования разрабатываемой ИИС. В ходе исследования:

1. Проведено исследование принципов лазерного сканирования деталей, движущихся по конвейеру, в результате которого выделены

особенности облака точек, получаемого при таком сканировании, а именно:

— представление облака точек «срезами» и его разреженность по оси движения детали;

— покрытие облаком точек только видимой части детали с камеры;

— шумы и выбросы вызванные ошибкой определения центром луча лазера.

2. По результатам исследования существующих методов эмуляции лазерного сканирования выделена необходимость разработки методики эмуляции лазерного 3D-сканирования для воспроизведения описанных особенностей облака точек.
3. Разработана методика эмуляции лазерного сканирования, позволяющая воспроизвести описанные особенности облака точек, основанная на формировании изображения с камеры устройства лазерного сканирования, содержащего только видимый луч лазера на поверхности детали, с последующим выделением центра лазерного луча и восстановлением положения его центров в пространстве.
4. Выбраны тестовые детали для проведения тестирования разработанной ИИС. Разработан прототип системы лазерного сканирования и программное обеспечение для него. Проведено экспериментальное исследование соответствия облака точек, полученному с помощью разработанной методики эмуляции, облаку точек, полученного при реальном сканировании, в результате которого получено отличие облаков не более 10% и 6% в среднем при использовании вокселя 0,7 мм. Сделан вывод, что при полученных отличиях является адекватным использование облака точек, полученного с помощью разработанной методики эмуляции для тестирования разрабатываемой ИИС.
5. С помощью разработанной методики эмуляции получены 2448 облаков точек тестовых деталей которые будут использоваться для тестирования разрабатываемой ИИС.

Полученные в результате эмуляции облака точек принимаются за эквивалентные облакам, полученных при реальном сканировании, и будут использоваться для тестирования разрабатываемой ИИС с целью проверки точности идентификации при различных углах расположения детали на конвейере.

Глава 3. Разработка методики идентификации объекта

3.1. Анализ результатов идентификации тестовых деталей с помощью глобальных дескрипторов

Проведено экспериментальное исследование точности идентификации с использованием дескрипторов CVFH, VFH на деталях, выбранных в главе 2. Такое исследование проведено с целью проверки результатов идентификации на реальных промышленных деталях по облаку точек, полученному с устройства лазерного сканирования. Необходимо было подтвердить воспроизводимость полученных в работе [17] результатов, так как авторами рассматривались детали с простой геометрией.

Исследование дескриптора VFH проведено с целью глубокого анализа недостатков дескриптора CFVH, так как дескриптор VFH является его предшественником. Сравнение результатов идентификации позволит проверить правильность выделенных авторами дескриптора CVFH недостатков в дескрипторе VFH.

Для эксперимента использованы 2448 облаков точек тестовых деталей, полученных с помощью методики эмуляции лазерного сканирования в главе 2. Эталонные дескрипторы сгенерированы с 1226 и 7092 ракурсов для каждой детали путем ее вращения вокруг центра масс.

В таблице 6 приведены результаты тестового исследования, где A — точность идентификации, рассчитанная по формуле (50), а N — число эталонных дескрипторов для детали в наборе.

Немаловажным параметром является время идентификации, для VFH с 7082 ракурсами снятия эталонных дескрипторов время идентификации составило в среднем 0,6 секунд, а для CVFH с 7082 ракурсами 8,82 секунд. Точность идентификации считается по формуле:

$$A = \frac{TP}{TP+FN}, \tag{50}$$

где TP — количество правильных идентификаций детали, FN — количество ошибочных идентификаций детали (деталь была идентифицирована как другие).

Таблица 6 — Результаты идентификации тестовых деталей с помощью дескрипторов VFH, CVFH

Деталь №	VFH 1226		CVFH 1226		VFH 7082		CVFH 7082	
	A	N	A	N	A	N	A	N
1	0,39	1226	0,3	9416	0,39	7082	0,54	53342
2	0,15	1226	0,1	5551	0,19	7082	0,15	31876
3	0,17	1226	0,34	14014	0,17	7082	0,49	81128
4	0,27	1226	0,514	9275	0,42	7082	0,58	54190
5	0,19	1226	0,15	7059	0,17	7082	0,19	41259
6	0,56	1226	0,39	3991	0,67	7082	0,41	23104
7	0,31	1226	0,2	2227	0,31	7082	0,24	12961
8	0,17	1226	0,3	5847	0,25	7082	0,34	33490
9	0,24	1226	0,2	4150	0,2	7082	0,36	24010
10	0,89	1226	0,68	3777	0,87	7082	0,79	21907
11	0,26	1226	0,4	6092	0,3	7082	0,42	35198
12	0,38	1226	0,42	8201	0,5	7082	0,48	48001
среднее	0,33	1226	0,33	6694	0,37	7082	0,41	38722

Из экспериментального исследования видно, что хоть дескриптор CVFH и показывает лучшие средние результаты относительно дескриптора VFH, они все равно остаются неудовлетворительными для использования в исходном виде в работе. При этом на части деталей лучшие результаты идентификации показывает дескриптор VFH.

По результату анализа выдвинута гипотеза, что один из основных недостатков дескриптора VFH, влияющих на точность идентификации, является ошибка при определении средней точки и расчете средней нормали, необходимых для корректного построения дескриптора. Эта ошибка вызвана тем, что при определении средней точки и расчета средней нормали, шумы и выбросы в облаке точек вносят существенную ошибку. Для подтверждения гипотезы проведена визуализация результатов определения средней точки и нормали в облаке точек при построении дескриптора VFH. Визуализация подтвердила гипотезу, применимо к деталям сложной формы — шумы, выбросы и небольшое изменения ракурса приводили к возникновению сильной ошибки определения центра и средней нормали относительно эталонных дескрипторов и, как следствие, к ошибочному построению дескриптора.

Визуализация результатов выделения кластеров в облаке точек, расчета нормалей и средней точки в них показала, что при построении CVFH сложная геометрия детали приводит к выделению большого числа кластеров, а, следовательно, и получению большого числа дескрипторов, так как дескриптор CVFH рассчитывается для каждого кластера. Все полученные с одного ракурса дескрипторы являются равносильными, и даже если один из них построен правильно, другие дескрипторы, построенные с некорректно определенным центром и средней нормалью, вносят ошибку в процесс идентификации, так как могут более точно совпадать с эталонными дескрипторами других деталей.

Из-за сложной геометрии детали во время эксперимента для некоторых ракурсов деталей было получено 14 кластеров, по которым рассчитывались 14 дескрипторов. В среднем для тестовых деталей было выделено по 6 кластеров на ракурс.

Подход к увеличению порогового значения минимального числа точек, входящих в один кластер, не решал описанную проблему. При повышении такого значения все точки на поверхности определялись в один кластер, следовательно, дескриптор CVFH, становился дескриптором VFH, с разницей в методике формирования гистограмм и компоненты SDC.

На основании проведенного исследования сделан вывод, что стабильность определения центра и средней нормали является важным фактором для построения дескриптора и напрямую влияет на результаты идентификации. Поэтому необходимо разработать подход, который позволит рассчитать среднюю точку и среднюю нормаль в облаке точек детали со сложной геометрией, полученной с рассмотренной в главе 2 точки контроля.

Другим недостатком является методика формирования эталонных дескрипторов (ФЭД), предложенная в работе CVFH. Она является вычислительно сложной и не позволяет использовать большое количество ракурсов для формирования эталонных дескрипторов. При этом для деталей

сложной формы тяжело подобрать коэффициент похожести облаков, что ведет к получению большого числа похожих дескрипторов в наборе эталонных дескрипторов.

Авторы использовали сложную методику ФЭД по нескольким причинам. Первая причина — наличие компоненты ракурса в дескрипторе (Viewpoint component). Эта компонента предложена в дескрипторе VFH для того, чтобы дескриптор грубо отражал ракурс, с которого он был получен. С помощью такого подхода предполагалось выполнять грубое позиционирование детали в пространстве. Поэтому, дескриптор каждого ракурса нес в себе уникальную компоненту, которая не позволяет динамически объединять дескрипторы в группы. Вторая — получение нескольких дескрипторов на ракурс. Из-за чего тяжело выполнять поиск похожих дескрипторов среди соседних.

Отказ от компоненты ракурса позволит разработать методику формирования эталонных дескрипторов, в которой будет выполняться объединение похожих дескрипторов в группы, вместо сопоставления исходных облаков точек. Это допустимо так как в диссертационном исследовании не рассматривается проблема грубого позиционирования детали по дескриптору. Такая методика позволит осуществлять динамический выбор ракурса, а также ускорить процесс ФЭД, так как сравнение дескрипторов вычислительно проще и быстрее, чем сравнение облаков точек.

3.2. Разработка методики построения глобального дескриптора для использования в составе разрабатываемой информационно измерительной системы

По результатам анализа результатов идентификации с помощью дескрипторов CVFH, VFH, к разрабатываемому дескриптору Laser Scanner Feature Histogram (LSFH) выдвигаются следующие требования:

1. Как и CVFH, VFH новый дескриптор должен быть инвариантен к повороту облака вокруг оптической оси камеры, что необходимо для методики формирования эталонных дескрипторов.
2. В отличие от CVFH и VFH новый дескриптор не должен нести информации о ракурсе, с которого он был рассчитан.
3. Должен рассчитываться один дескриптор на один ракурс.
4. Дескриптор должен нести в себе информацию о геометрических параметрах детали.
5. Должен осуществляться стабильный расчет средней точки и средней нормали в облаках точек детали со сложной геометрией для рассматриваемой точки контроля.

В работе проанализированы облака точек, полученные с точки контроля, состоящей из нескольких устройств лазерного сканирования. По результатам анализа выдвинута гипотеза, что при описанном расположении устройств сканирования, а также на основе информации о положении конвейера, объектно-ориентированный ограничивающий параллелепипед (minimal oriented bounding box или MOBB) может быть стабильно рассчитан с помощью алгоритма [91]. Стабильный расчет MOBB позволяет использовать его как дополнительный признак в процессе идентификации, так и непосредственно во время построения дескриптора. Пример построенного MOBB для полигональной модели тестовой детали №1 и облака точек этой детали, полученной с помощью разработанной методики эмуляции сканирования изображен на рисунке 27.

Стабильность расчета MOBB подтверждена экспериментально на облаках точек тестовых деталей, полученных в главе 2. В результате эксперимента получено, что после подготовки облака точек, средняя ошибка определения длин сторон MOBB и его центра в облаках точек относительно полигональной модели детали составляет в среднем 1,45 мм. Такая ошибка позволяет использовать центр MOBB в качестве средней точки.

Еще одной гипотезой при построении дескриптора LSFH является возможность модификации компоненты SDC (12) с учетом параметра MOBB. В исходном дескрипторе в качестве нормализующего параметра использовалось расстояние до самой удаленной точки в облаке. Однако этот параметр, является не показательным, так как часть облака может быть скрыта. В работе предлагается заменить нормализующий параметр на половину длины диагонали рассчитанного MOBB. Это позволит рассчитать компоненту распределения точек относительно всей детали, независимо от ракурса, с которого было получено облако точек. В работе доработанная компонента называется LSSDC (Laser Scanner Shape Distribution Component).

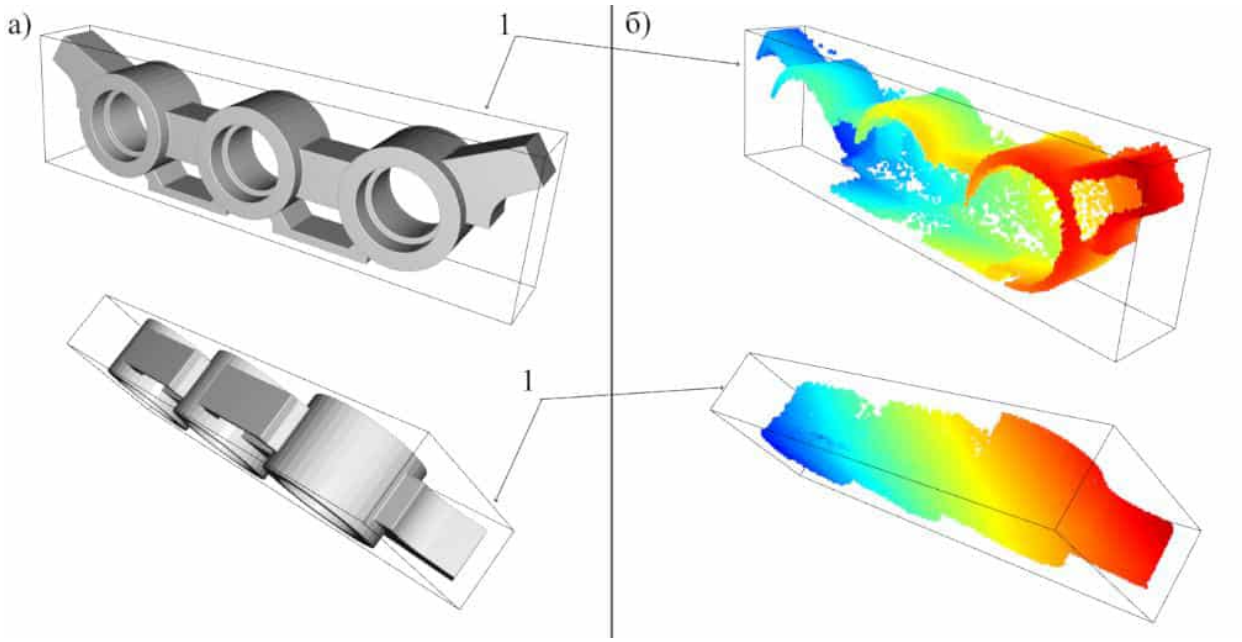


Рисунок 27 — Результат расчета объектно-ориентированного ограничивающего параллелепипеда для: а — полигональной модели детали №1; б — облако точек, полученное с помощью эмуляции сканирования детали №1

Тогда с учетом требований к разрабатываемому дескриптору и выдвинутых гипотез, из методики расчёта дескриптора CVFH использованы:

— этап удаления точек с большим значением кривизны при расчёте средней нормали;

— этап вокселизации;

— учет абсолютного количества точек в гистограммах, для кодирования геометрических параметров детали (площади видимой поверхности) в дескрипторе.

Из методики расчёта дескриптора VFH использован алгоритм расчета компонент угловых распределений точек, обеспечивающий инвариантность дескриптора к повороту вокруг оптической оси камеры, что позволит оптимизировать процесс формирования эталонных дескрипторов.

Для методики расчета дескриптора LSFH использованы новые:

— алгоритмы расчета средней точки, основанные на использовании параметров MOBB для общего облака точек детали, полученного с нескольких устройств сканирования;
— компонента LSSDC, являющаяся компонентой SDC дескриптора CVFH, адаптированная под новую методику построения дескриптора LSFH.

В основе методики построения дескриптора лежит алгоритм представленный четырьмя последовательными шагами. На первом шаге облака точек, полученные с каждого устройства сканирования, преобразуются в воксельную сетку. В работе экспериментально выбран размер вокселя 0,7 мм. В качестве точки представляющей воксель используется его центр, а нормаль для этой точки рассчитывается как средняя нормаль всех точек, находящихся в вокселе. В результате на первом шаге получены облака точек $S_k = \{p_{k_i}\, n_{k_i}\}_{i=1}^{a_k}$, где k индекс устройства сканирования, $p_{k_i} \in R^3$ координаты точки (центры вокселей), а $n_{k_i} \in R^3$ — аппроксимированная нормаль поверхности в заданной точке $||n_{k_i}|| = 1$, i — индекс точки в облаке точек для k-го устройства сканирования, a_k — число точек в облаке полученного с k-го устройства сканирования .

На втором шаге облака точек объединяются в общее облако точек в одной заданной системе координат. Для общего облака точек с помощью алгоритма

[91] рассчитываются параметры MOBB: $C = (c_x, c_y, c_z)$, $C \in R^3$ — центр объектно-ориентированного ограничивающего параллелепипеда и $D = (d_1, d_2, d_3)$, $D \in R^3$ — длины ребер параллелепипеда, при этом вектор D составлен таким образом, что выполняется условие $d_1 \leq d_2 \leq d_3$.

На третьем шаге производится расчет средней точки и средней нормали для каждого облака точек S_k. Средняя точка p_{c_k} облака точек S_k, полученного с k-го устройства сканирования, рассчитывается по формуле (51).

$$p_{c_k} = C_k, \tag{51}$$

где C_k — центр MOBB в координатной системе k-го устройства сканирования.

Для расчета средней нормали используется только часть облака точек. Подход к выбору точек для построения нормали взят из дескриптора CVFH. Этот подход позволяет удалить точки в местах резкого изменения поверхности, так как в этих точках достаточно тяжело точно аппроксимировать нормаль. Формирование облака $\overline{S_k}$ для расчета средней нормали осуществляется по следующему алгоритму:

1. Для каждой точки из облака S_k рассчитывается значение кривизны поверхности (53) в окрестности σ этой точки [92]. В работе $\sigma = 3v$, где v — размер вокселя. Расчет кривизны осуществляется с помощью матрицы ковариации (52):

$$Cov(p_{k_i}) = \begin{bmatrix} q_1 - p_{k_i} \\ \cdots \\ q_t - p_{k_i} \end{bmatrix}^T \begin{bmatrix} q_1 - p_{k_i} \\ \cdots \\ q_t - p_{k_i} \end{bmatrix}, \tag{52}$$

где t — число точек в окрестности σ точки p_{k_i}. Для матрицы ковариации в точке p_{k_i} находятся собственные значения и выбираются таким образом, что $\lambda(p_{k_i})_1 \geq \lambda(p_{k_i})_2 \geq \lambda(p_{k_i})_3 \geq 0$. Тогда значение кривизны $\gamma(p_{k_i})$ поверхности в точке p_{k_i} вычисляется по формуле (53):

$$\gamma(p_{k_i}) = \frac{\lambda(p_{k_i})_3}{\lambda(p_{k_i})_1 + \lambda(p_{k_i})_2 + \lambda(p_{k_i})_3}. \quad (53)$$

2. Из облака точек S_k удаляются точки со значением кривизны поверхности в данной точке больше порогового значения ρ, в результате чего формируется облако точек $\overline{S_k} = \{\overline{p_{k_i}}, \overline{n_{k_i}}\}_{i=1}^{I_k}$, где I_k — число точек в k-м облаке, оставшихся после удаления точек со значением кривизны больше порогового. В работе значение $\rho = 0{,}035$.
3. Средняя нормаль n_{c_k} для k-го облака точек рассчитывается по формуле (54):

$$n_{c_k} = \frac{\sum_{i=1}^{I_k} \overline{n_{k_i}}}{I_k}. \quad (54)$$

Первые три шага расчёта дескриптора изображены на рисунке 28.

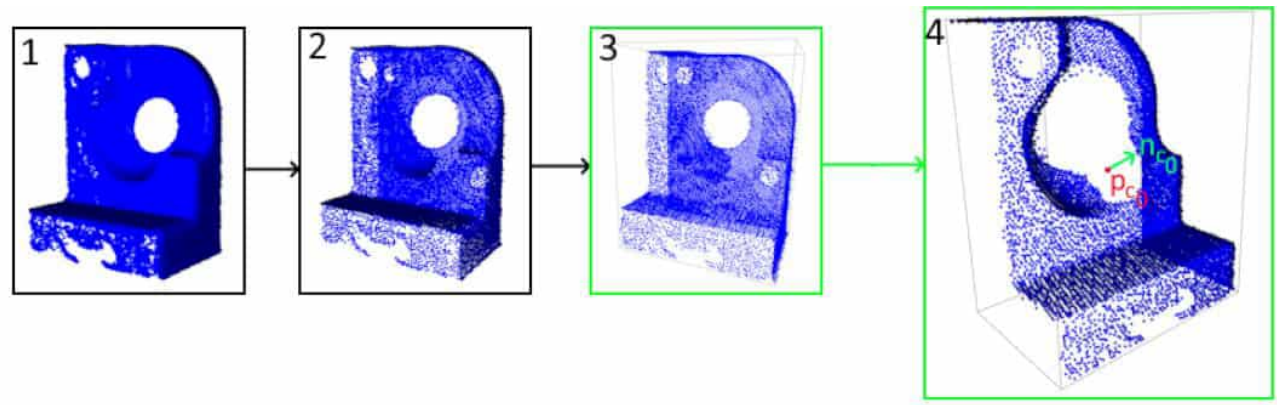


Рисунок 28 — Этапы расчета LSFH: 1 — исходное облако точек; 2 — вокселизация облака точек; 3 — расчет MOBB для общего облака точек; 4 — расчет средней точки и средней нормали

На четвёртом шаге производится построение дескриптора для каждого облака точек S_k, полученного с k-го устройства сканирования. Для этого с помощью рассчитанного центра и средней нормали для каждой i-й точки облака S_k рассчитываются угловые распределения α_{k_i} (58), ϕ_{k_i} (59), θ_{k_i}(60), а так же компонента распределения точек LSSDC, являющаяся адаптированной компонентой SDC дескриптора CVFH (61):

$$u_{k_i} = n_{c_k}, \quad (55)$$

$$v_{k_i} = \frac{p_{k_i} - p_{c_k}}{\|p_{k_i} - p_{c_k}\|} \times u_{k_i}, \quad (56)$$

$$w_{k_i} = u_{k_i} \times v_{k_i}, \quad (57)$$

$$\alpha_{k_i} = arccos(v_{k_i} \cdot n_{k_i}), \quad (58)$$

$$\phi_{k_i} = arccos\left(u_{k_i} \cdot \frac{p_{k_i} - p_{c_k}}{\|p_{k_i} - p_{c_k}\|}\right), \quad (59)$$

$$\theta_{k_i} = arctg(w_{k_i} \cdot n_{k_i}, u_{k_i} \cdot n_{k_i}). \quad (60)$$

В отличие от компоненты SDC дескриптора CVFH, в котором в знаменателе использовалось расстояние от центра кластера до самой дальней точки облака, что не учитывало реальных габаритов объекта, в компоненте LSSDC в знаменателе используется половина диагонали рассчитанного MOBB, что должно лучше представлять распределение точек поверхности объекта с учетом его габаритов:

$$LSSDC_{k_i} = \frac{2\|p_{k_i} - p_{c_k}\|}{\sqrt{({d_1}^2 + {d_2}^2 + {d_3}^2)}}, \quad (61)$$

где d_1, d_2, d_3 — длины сторон MOBB.

Расчет угловых параметров и компоненты $LSSDC_{k_i}$ для i-й точки изображен на рисунке 29.

По полученным распределениям строится дескриптор LSFH — вектор, размерностью 180 элементов, составленный из четырех гистограмм количественного распределения значений θ_k, α_k, ϕ_k с шагом $\frac{2\pi}{45}$ и значений $LSSDC_k$ с шагом $\frac{1}{45}$, содержащих по 45 элементов в каждой гистограмме (Таблица 7).

Таблица 7 —Структура дескриптора LSFH

Глобальный дескриптор LSFH (Laser Scanner Feature Histogram)			
45 ячеек с шагом $\frac{2\pi}{45}$	**45** ячеек с шагом $\frac{2\pi}{45}$	**45** ячеек с шагом $\frac{2\pi}{45}$	**45** ячеек с шагом $\frac{1}{45}$

Гистограмма распределения значений $\boldsymbol{\theta_k}$	Гистограмма распределения значений $\boldsymbol{\alpha_k}$	Гистограмма распределения значений $\boldsymbol{\phi_k}$	Гистограмма распределения значений $\boldsymbol{LSSDC_k}$

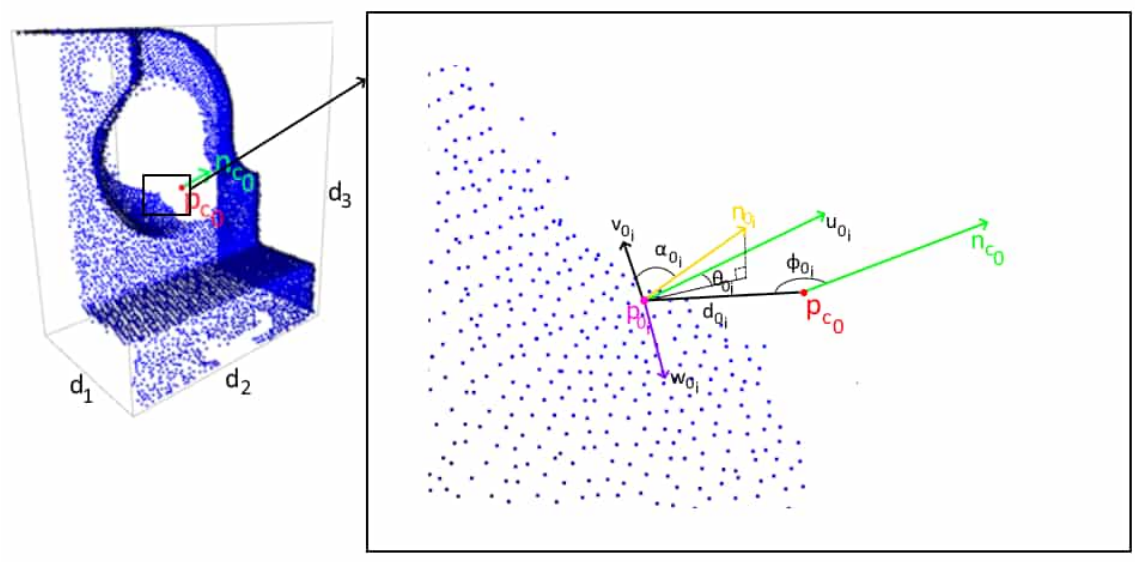


Рисунок 29 — Расчёт параметров дескриптора LSFH для i-й точки

Дескриптор LSFH является инвариантным к повороту облака точек вокруг оптической оси камеры: инвариантность угловых распределений θ_k, α_k, ϕ_k рассмотрены в работе [29], компонента LSSDC инвариантна к повороту, так как в ней используются расстояния от средней точки до точек облака, а средняя точка и средняя нормаль будут рассчитаны с учетом поворота облака.

В работе в качестве метрики сравнения дескрипторов используется евклидово расстояние между ними (62):

$$d(LSFH_a, LSFH_b) = \frac{\sqrt{\sum_{i=1}^{180}(LSFH_a(i) - LSFH_b(i))^2}}{180}. \quad (62)$$

Чем меньше расстояние $d(LSFH_a, LSFH_b)$, тем больше дескрипторы соответствуют друг другу. Пример дескрипторов LSFH изображен на рисунке 30.

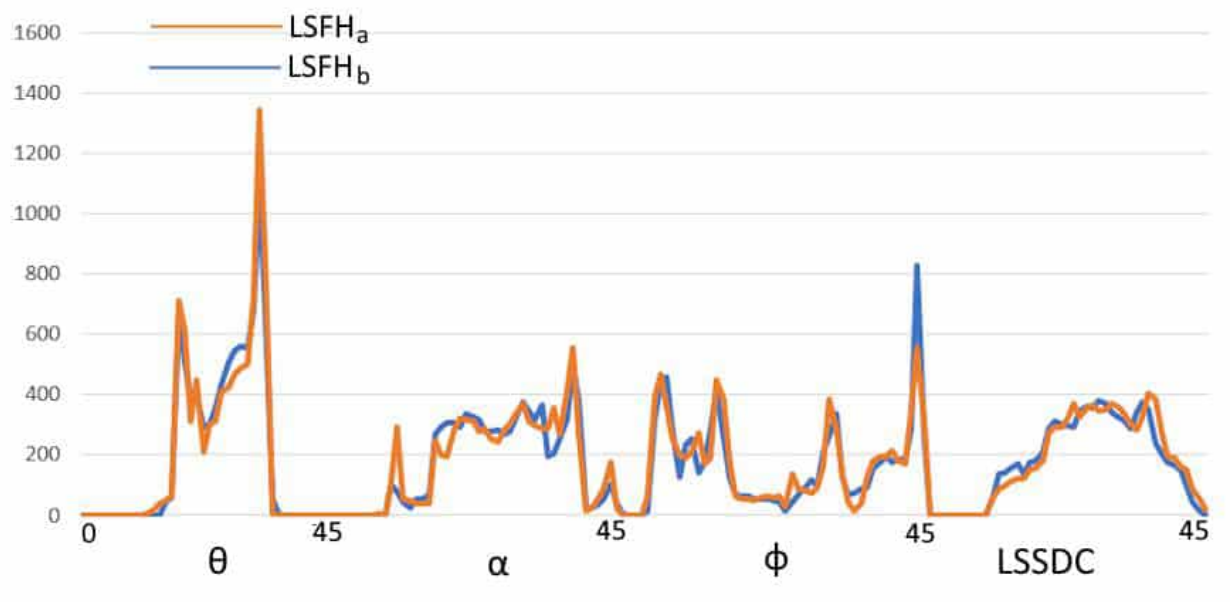


Рисунок 30 — Графическое представление дескрипторов LSFH

Описанная методика расчета дескриптора принимает на вход подготовленные и отфильтрованные облака точек, полученные с нескольких устройств сканирования в составе точки контроля, на выходе формирует несколько дескрипторов LSFH, количество которых соответствует числу устройств сканирования, и параметры MOBB для общего облака точек, по которым осуществляется дальнейшая идентификация.

3.3. Разработка методики подготовки облака точек для построения дескриптора

Анализ облаков точек, полученных в результате эмуляции, а также облаков точек реального сканирования показал, что в облаке точек содержатся единичные выбросы и шумы, полученные из-за бликов или неточного определения центра луча лазера. Такие выбросы представлены единичными точками в пространстве, однако их наличие вносит ошибку в определение параметров MOBB, так как он будет построен с учетом этих точек, а, следовательно, влияет и на соответствие дескриптора эталонному. Поэтому сначала необходимо удалить такие точки из облака. В работе для решения этой проблемы предлагается использовать статистические фильтры.

Немаловажным является то, что в облаках точек реального сканирования содержатся точки заднего фона, в случае с рассмотренной точкой контроля — точки на поверхности конвейера. Такой тип точек тоже должен быть удален из исходного облака, так как он влияет на расчет параметров MOBB. Такие точки могут быть удалены геометрическим фильтром, так как до начала сканирования известна рабочая область точки контроля.

При этом необходимым условием для построения дескриптора является аппроксимированные нормали в каждой точке облака. От точности аппроксимации нормалей и повторения точками реальной поверхности детали, зависит корректность построения дескриптора.

Для решения задачи аппроксимации нормалями и реорганизации облака для лучшего соответствия поверхности детали предназначен фильтр WLOP, рассмотренный в главе 1. Поэтому в работе был проведен эксперимент с применением рассмотренного фильтра к облакам точек. Применение этого фильтра хоть и позволило более точно описать поверхность детали и аппроксимировать нормали, но при этом нарушало структуру облака точек и стягивало точки к центру детали. Это влияло на точность построения MOBB, так как он получался меньше эталонного, а также на построение дескриптора, так как из-за стягивания точек появлялись дыры в облаке, в которые не попадали воксели. Поэтому необходимо доработать фильтр для устранения выделенных недостатков.

Из проведенного исследования структуры облаков точек следует, что для стабильного построения дескриптора LSFH облака точек необходимо предварительно подготовить. К разрабатываемой методике подготовки облака точек выдвигаются следующие требования:

— методика должна позволять аппроксимировать нормали в каждой точке облака, при этом они должны быть направлены из детали;

— при подготовке облака должны удаляться единичные выбросы и точки, расположенные вне рабочей области устройства сканирования;

— нормали должны фильтроваться, а точки реорганизовываться для лучшей аппроксимации поверхности детали, при этом не должна нарушаться структура исходного облака точек.

С учетом требований методика подготовки облака точек состоит из пяти этапов (Рисунок 31), проводимых для каждого из облаков, полученных с устройств сканирования.

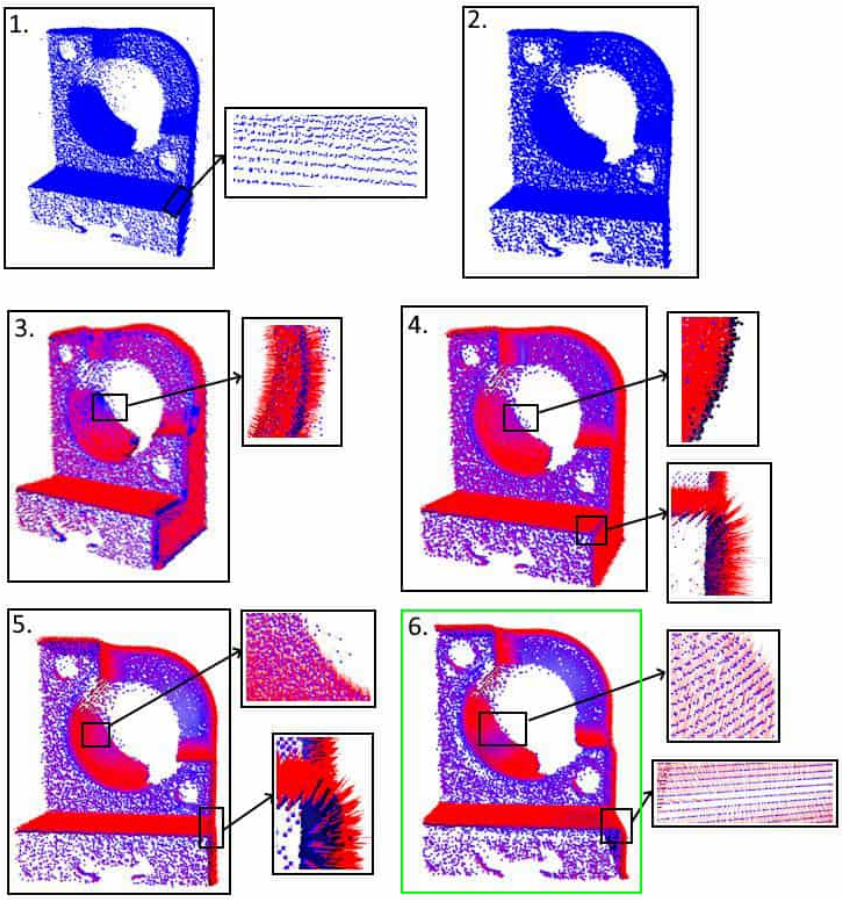


Рисунок 31 — Подготовка облака точек: 1 — исходное облако; 2 — статистическая и геометрическая фильтрация; 3 — предварительная аппроксимация нормалями; 4 — восстановление направления нормалей; 5 — фильтрация нормалей; 6 — реорганизация облака точек

Первый этап — статистическая и геометрическая фильтрация. Геометрическая фильтрация заключается в удалении точек, расположенных вне области сканирования. Область сканирования известна из взаимного

расположения устройств сканирования и конвейера. Основная идея статистической фильтрации состоит в удалении тех точек, в окрестностях радиуса R которых расположено количество точек меньше заданного порогового значения δ. Основная задача такой фильтрации — удаление единичных выбросов и шумов. Основной сложностью данного метода является выбор параметров R и δ. В работе параметры выбраны экспериментально: $R = 4 \cdot step$ и $\delta = \frac{R}{2d_x}$, где $step$ — длина среднего вектора шага конвейера, d_x — среднее расстояние между точками на срезе на ровной поверхности детали. В результате применения описанной фильтрации будет получено облако точек $P = \{p_i \in R^3\}_{i=1}^{I}$, где I — количество точек в облаке, оставшихся после применения описанного фильтра.

На втором этапе в полученном облаке точек с помощью метода анализа главных компонент (PCA) [75] производится грубый расчет нормалей. Для каждой точки $p \in P$ и ее фиксированной окрестности σ рассчитывается матрица ковариации (52). Для полученной матрицы вычисляются собственные вектора. За вектор нормали n_{r_i} в точке p_{r_i} принимается собственный вектор, соответствующий минимальному собственному значению матрицы $Cov(p_{r_i})$. В качестве параметра окрестности для расчета матрицы ковариации экспериментально получено значение $\sigma = 6 \cdot step$. После аппроксимации описанным методом нормали могут быть направлены как от поверхности детали, так и в обратном направлении. В результате применения описанного алгоритма получено облако точек $\overline{P} = \{p_i \in R^3, \overline{n_i} \in R^3\ \}_{i=1}^{I}$, где n_i — аппроксимированная нормаль в точке p_i с помощью метода PCA.

На третьем этапе выполняется восстановление направления нормалей и дополнительная фильтрация точек, нормали которых направлены под углом близкому к прямому к видовому вектору $\vec{v}_{p_i}$. Идея восстановления направления нормалей заключается в том, что камера, не может видеть поверхности, нормали которых направлены под острым углом к видовому вектору $\vec{v}_{p_i}$ к точке p_i (63).

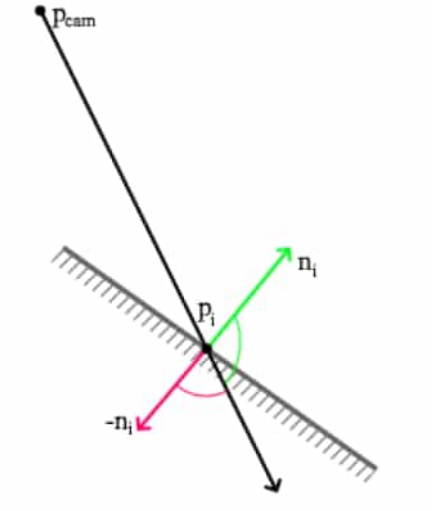


Рисунок 32 — Выбор направления аппроксимированной нормали в точке p_i

Так как общее облако точек состоит из набора срезов плоскостей лазера, то точка положения камеры над каждым срезом будет иметь разные координаты, поэтому видовой вектор для точки p_i рассчитывается по формуле (63):

$$\vec{v}_{p_i} = p_i - p_{cam_{id}}, \tag{63}$$

где $p_{cam_{id}}$ — точка положения камеры над срезом с индексом id, $p_i \in E_{id}$, а E_{id} — набор точек, принадлежащих срезу с индексом id.

Тогда выбор направления нормали осуществляется по формуле (65):

$$\varphi_i = \arccos\left(\frac{\vec{v}_{p_i} \cdot \overline{n_i}}{\left\|\vec{v}_{p_i}\right\| \|\overline{n_i}\|}\right), \tag{64}$$

$$n_i = \begin{cases} \overline{n_i}, \text{если } \varphi_i \geq \frac{\pi}{2} \\ -\overline{n_i} \end{cases}. \tag{65}$$

Дополнительная фильтрация заключается в удалении точек, для которых $\left|\varphi_i - \frac{\pi}{2}\right| < \vartheta$ (64). В работе выбрано пороговое значение $\vartheta = 5°$.

В результате третьего этапа будет получено облако точек $S_p = \{p_i, n_i\}_{i=1}^{I_p}$, где I_p — число точек в итоговом облаке, полученном на третьем этапе, после проведения фильтрации.

На четвертом этапе происходит фильтрация нормалей в полученном на третьем этапе облаке S_p. Фильтрация нормалей осуществляется с помощью алгоритма, описанного в работе [74], идея которого заключается в минимизации функции расхождения нормалей между всеми точками в окрестности O_{p_i} радиуса σ_p каждой точки p_i облака точек S_p. Минимизация выполняется итеративным перерасчетом нормалей с помощью формулы (17), где значение $\sigma_p = 6 \cdot step$, а значение $\sigma_n = 15°$, согласно работе [74].

Пятый этап необходим для реорганизации облака точек, чтобы оно лучше соответствовало поверхности детали. Для этого использовался доработанный алгоритм WLOP.

В работе предпринята попытка доработать рассмотренный фильтр с учетом принципов лазерного сканирования, описанных в главе 2. Цель доработок алгоритма — произвести реорганизацию облака точек, для лучшего описания поверхности, сохранив при этом структуру исходного облака точек.
Для этого выдвинута гипотеза, что так как облако точек собирается из срезов, то точки, принадлежащие одному срезу, могут перемещаться только в плоскости лазера. Поэтому при ограничении перемещения точек плоскостями срезов которым они принадлежат получится сохранить структуру облака, а также уменьшить эффект стягивания точек к центру детали.

В отличии от эталонного алгоритма в доработанном реорганизация осуществляется для всех точек $Q \to P$. Значит $Q = \{q_i = p_i\}_{i=1}^{I_p}$, где $p_i \in S_p$. В качестве начальных значений выбирается $P^0 = Q$, так как все точки Q изначально лежат в плоскостях лазера (при фильтрациях и сглаживании на первых четырех этапах изменялись направления нормалей и точки удалялись из исходного набора, перемещение точек в пространстве не производилось). При вычислении проекции точки, положение проецируемых точек множества P ограничивается их перемещением на плоскости среза луча лазера для точки q, являющейся обратным отображением для точки p. Тогда итеративный алгоритм

(19), с условием, что каждая точка $p_i \in E_{id}$, где E_{id} — набор точек, принадлежащих срезу с индексом id, принимает вид (66):

$$p_i^{k+1} = et_i * \overrightarrow{e_{las}} + ut_i * \overrightarrow{u_{las}} + l_{id}, \quad (66)$$

где $\overrightarrow{n_{laser}}$ — единичный вектор нормали плоскости лазера, $\overrightarrow{e_{las}}$ — единичный вектор сонаправленный оси лазера, $\overrightarrow{u_{las}}$ — единичный вектор образованный векторным произведением векторов $\overrightarrow{n_{laser}}$ и $\overrightarrow{e_{las}}$, l_{id} — точка начала лазерного луча в координатной системе устройства сканирования на срезе с индексом id.

$$et_i = eavg_i + \mu * ereps_i, \quad (67)$$

$$ut_i = uavg_i + \mu * ureps_i, \quad (68)$$

где параметр $\mu = 0{,}45$, по рекомендациям авторов работы [74].

С учетом проекции на плоскость среза лазера коэффициенты рассчитываются по формулам:

$$eavg_i = \sum_{j \in J} \overrightarrow{e_{las}} \cdot (q_j - l_{id}) \frac{\alpha_{ij}^k}{\sum_{j \in J} \alpha_{ij}^k}, \quad (69)$$

$$ereps_i = \sum_{j \in J} \overrightarrow{e_{las}} \cdot ({p_i}^k - {p_{i'}}^k) \frac{\beta_{ii'}^k}{\sum_{i' \in I \setminus \{i\}} \beta_{ii'}^k}, \quad (70)$$

$$uavg_i = \sum_{j \in J} \overrightarrow{u_{las}} \cdot (q_j - l_{id}) \frac{\alpha_{ij}^k}{\sum_{j \in J} \alpha_{ij}^k}, \quad (71)$$

$$ureps_i = \sum_{j \in J} \overrightarrow{u_{las}} \cdot ({p_i}^k - {p_{i'}}^k) \frac{\beta_{ii'}^k}{\sum_{i' \in I \setminus \{i\}} \beta_{ii'}^k}, \quad (72)$$

где коэффициенты $\alpha_{ij}^k = \frac{\psi(n_i, {p_i}^k - q_j)}{\|{p_i}^k - q_j\|}$, $\beta_{ii'}^k = \frac{\theta(\|{p_i}^k - {p_{i'}}^k\|)}{\|{p_i}^k - {p_{i'}}^k\|}$, $\theta(r) = e^{-\frac{r^2}{{\sigma_p}^2}}$, $\psi\left(n_i, {p_i}^k - q_j\right) = e^{-\frac{({n_i}^T({p_i}^k - q_j))^2}{{\sigma_p}^2}}$ использованы из исходного алгоритма.

В описанном алгоритме наборы точек в областях J и I (69, 70, 71, 72) выбираются для каждой i-й точки, путем поиска ближайших соседей в окрестностях радиусом $\delta = 6step$ для облаков Q и P соответственно. Параметр

${\sigma_p}^2 = \frac{\delta^2}{4}$ взят из существующих реализаций алгоритма WLOP, согласно размеру окрестности поиска ближайших соседей. На рисунке 33 изображен срез лазера, точка l_{id}, вектора $\overrightarrow{e_{las}}$, $\overrightarrow{u_{las}}$ и точки, принадлежащие этому срезу, а также результат применения доработанного алгоритма фильтрации для точек на срезе.

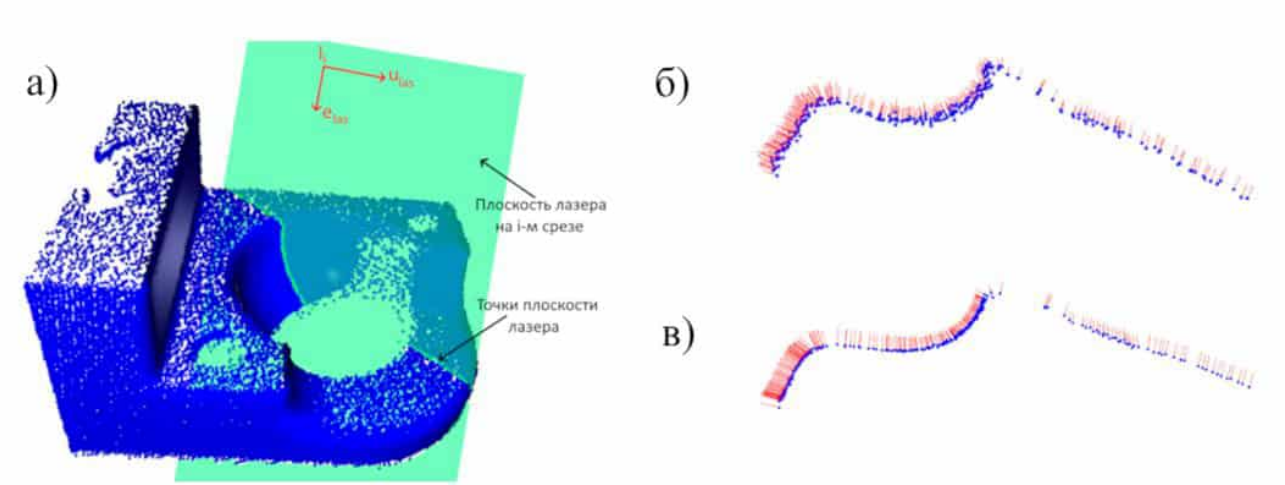


Рисунок 33 — Адаптированный алгоритм реорганизации точек: а — положение плоскости лазера и точек на заданной плоскости; б — точки и их нормали для выбранной плоскости лазера до сглаживания нормалей и реорганизации; в — результат работы четвертого и пятого этапов для точек и их нормалей для выбранной плоскости лазера

Четвертый и пятый этапы могут выполняться несколько раз для достижения оптимального результата, где в качестве входных данных четвертого этапа будет использоваться облако, полученное на пятом. Эксперименты показали, что одной — двух итераций достаточно для решения задачи реорганизации облака точек.

Результатом подготовки облака точек по предложенной методике является отфильтрованное и сглаженное облако точек с аппроксимированной нормалью к поверхности детали в каждой точке данного облака с сохранением первоначальной структуры, которое можно использовать для построения разработанного дескриптора LSFH.

Оценка методики подготовки облака точек проводится на тестовых данных, полученных во второй главе, с помощью поиска среднего расстояния до

ближайшего эталонного дескриптора (73) и среднего расстояния длин ребер до эталонного (74) для каждой детали:

$$L_{avg} = \frac{1}{\mathrm{n}} \sum_{i=1}^{\mathrm{n}} L_{LSFH\,i}, \tag{73}$$

где $L_{LSFH\,i}$ — расстояние до ближайшего дескриптора (62) в наборе эталонных дескрипторов детали, для i-го тестового дескриптора $LSFH_i$.

$$D_{avg} = \frac{1}{\mathrm{n}} \sum_{i=1}^{\mathrm{n}} \left\| \overline{D}_i - \overline{D} \right\|, \tag{74}$$

где $\overline{D}_i$ — параметры MOBB для i-го общего облака точек детали, а $\overline{D}$ — эталонные параметры MOBB для детали.

Было рассчитаны 102 параметра MOBB (по одному на каждое общее облако точек) и 204 дескриптора (для каждого облака точек) с применением описанной методики подготовки облака точек и с применением только этапов аппроксимации нормалей и восстановления их направления (стандартная методика подготовки). Результаты эксперимента представлены в таблице 8.

Таблица 8 — Результаты расчета L_{avg} и D_{avg} для облаков точек с применением методики подготовки облака точек и без нее

Деталь №	**Стандартная методика подготовки**		**Методика подготовки**	
	L_{avg}	D_{avg} мм	L_{avg}	D_{avg} мм
1	1181,01	2,53	987,74	1,34
2	1469,41	2,31	1338,34	0,99
3	2510,88	2,89	2134,49	1,39
4	2282,14	3,10	1690,29	1,16
5	2106,67	2,64	1710,6	1,32
6	792,26	2,37	615,67	1,74
7	1591,33	1,98	1254,05	1,24
8	1417,67	2,83	973,88	1,49
9	1016,85	2,94	766,60	1,92
10	733,06	2,13	555,35	1,29
11	2489,85	2,50	1852,74	1,63
12	2899,99	2,88	2104,87	1,96

среднее	1707,59	2,59	1332,05	1,45

Из результатов сравнения видно, что среднее значение расстояния до ближайшего дескриптора и средняя ошибка определения параметра $\overline{D}$ меньше у дескриптора, построенного с использованием методик фильтрации, в среднем на 22% и на 43% соответственно, что подтверждает целесообразность использования дополнительных шагов фильтрации облака точек перед построением дескриптора.

3.4. Разработка методики формирования эталонных дескрипторов по цифровым моделям деталей

Задача методики формирования эталонных дескрипторов (ФЭД) по цифровым моделям деталей — подготовить набор дескрипторов для каждой детали, с которыми будет производиться сравнение полученных дескрипторов во время процесса идентификации. Размер реальной детали и ее цифровой модели должны совпадать.

При формировании эталонных дескрипторов возникают следующие задачи: выбор ракурсов для формирования дескриптора и определение количества дескрипторов в наборе эталонных дескрипторов.

Из исследования, проведённого в главе 3.1, был сделан вывод, что методика ФЭД, предложенная в работе [29], не позволяет работать с большим числом ракурсов, а также является вычислительно сложной. С учетом особенностей дескриптора LSFH, а именно инвариантности к повороту вокруг оптической оси и отсутствие компоненты ракурса, представляется возможным осуществлять группировку похожих дескрипторов в большом количестве дескрипторов, снятых с маленьким шагом поворота детали, представляя их одним дескриптором. На основе такого подхода и предлагается разработать методику формирования эталонных дескрипторов.

Первой задачей при разработке методики является выбор метода получения облака точек по полигональной модели детали, которое будет использоваться для построения дескрипторов. Проводить эмуляцию сканирования для получения облака точек в этом случае является нецелесообразно и вычислительно сложно так как необходимо обработать огромное количество ракурсов. Для ускорения процесса формирования эталонных дескрипторов необходимо использовать способ, позволяющий получить облако точек с ракурса за одну итерацию.

Так как из всего набора дескрипторов будут выбраны только некоторые значимые, то можно пренебречь точностью построения единичного дескриптора. Поэтому в работе предлагается получить плотное облако точек поверхности детали, отражающее максимально возможное облако точек, которое может быть получено во время лазерного сканирования, и по этому облаку выполнить построение дескриптора. Такое облако точек будет называться базовым облаком точек, а дескриптор, построенный по базовому облаку точек — базовым дескриптором.

Построение базового дескриптора осуществляется по следующему алгоритму:

1. Вокруг полигональной модели детали рассчитывается ограничивающий параллелепипед, выровненный по координатным осям (axis-aligned bounding box или AABB). AABB выбран с целью ускорения и упрощения расчетов. AABB построен таким образом, что верхняя грань, параллельная плоскости XoZ, пересекает ось Y в точке $(0, y_t, 0)$, а нижняя грань, параллельная верхней пересекает ось Y в точке $(0, y_b, 0)$.
2. Верхняя грань AABB разделяется на $n \times m$ квадратов, с длиной стороны d (Рисунок 34). Параметр d, выбирается таким образом, чтобы он был на порядок меньше размера вокселя, используемого в процессе построения дескриптора $d = \frac{1}{10} v$, где v — размер вокселя используемый в процессе построения дескриптора LSFH.

3. Через центры квадратов строятся отрезки, параллельные оси Y, таким образом, чтобы Y координата начала отрезков равнялась $y_t + 10$, а Y координата конца отрезка совпадала с Y координатой нижней грани AABB. К Y координате начала отрезков добавляется отступ, так как верхняя грань будет совпадать с одной или несколькими точками облака, а значит при использовании точки без отступа трассировка лучей будет выполнена неверно. Лучи, строятся параллельно оси Y, проходящими через вершины квадратов, а не исходящими из одной точки, как описано в работе [29] для эмуляции ToF сенсора или стереокамеры. Такой подход позволяет лучше учесть особенности облака точек, полученного с устройства лазерного сканирования при движении детали по ленте конвейера.
4. С помощью алгоритма трассировки лучей, для каждого из полученных отрезков ищется пересечение с полигонами модели детали. Точки пересечения отрезков с полигонами модели и являются идеальным облаком точек детали, полученным с заданного ракурса. На рисунке 34 изображен процесс построения такого облака для детали №1.

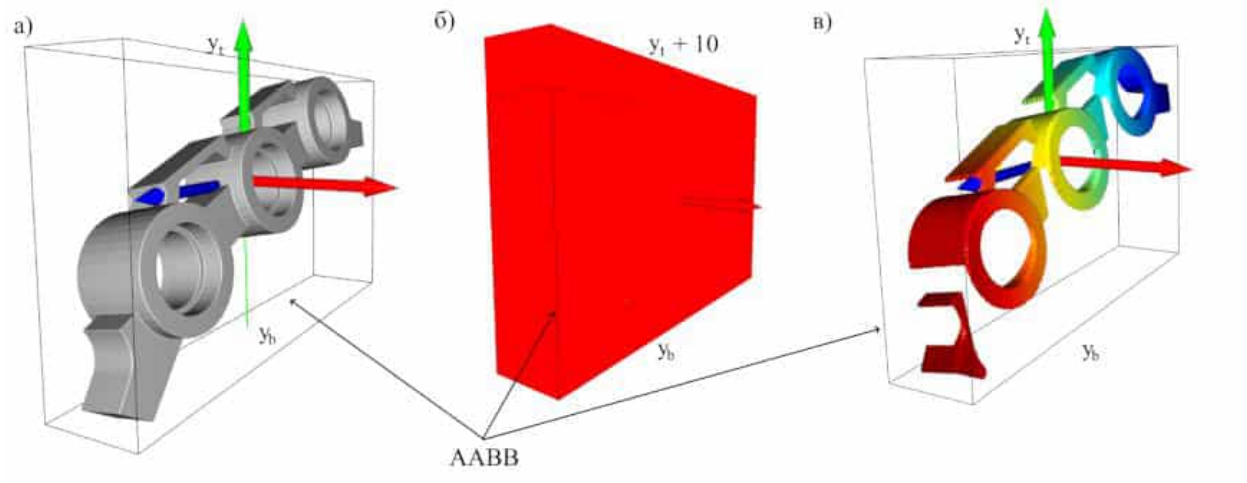


Рисунок 34 — Построение базового облака точек для цифровой модели детали

5. По полученному базовому облаку точек строится дескриптор $LSFH$ по методике, описанной в главе 3.2. Построенный дескриптор и будет базовым дескриптором.

Следующей задачей является выбор ракурсов, с которых будут формироваться базовые дескрипторы. Дескриптор LSFH является инвариантным к повороту вокруг оптической оси камеры, а значит генерировать базовые дескрипторы по ракурсам, полученным после поворота модели вокруг оси Y, не имеет смысла, так как они все будут одинаковыми. Дескрипторы будут иметь отличия только с ракурсов при повороте детали вокруг осей X и Z. В работе используется шаг поворота в 1°.

Генерирование базовых дескрипторов для полигональной модели детали, будет проводится для 65160 ракурсов, с помощью алгоритма, изображенного на рисунке 35:

1. Полигональная модель детали переносится в пространстве таким образом, чтобы ее центр масс совпадал с началом координат.
2. На каждой итерации выполняется поворот модели на заданные углы $\theta \in \left[-\frac{\pi}{2};\frac{\pi}{2}\right]$, $\varphi \in [0, 2\pi)$ вокруг осей Z и X соответственно.
3. Проводятся шаги для построения базового дескриптора, описанные ранее.
4. Полученный дескриптор сохраняется в карту дескрипторов.

После получения набора базовых дескрипторов возникает задача в объединении их в группы. Для объединения в группы необходимо выбрать критерий похожести дескрипторов. Однако в зависимости от размера детали и сложности его геометрии такой критерий будет отличаться. В работе выдвинута гипотеза, что выделить такой критерий можно статистически после первоначального разделения на группы с помощью алгоритма водораздела.

Так как дескрипторы строятся путем поворота модели вокруг осей X и Z, то представляется возможным перейти от трехмерного пространства к двумерному, представив эти дескрипторы с помощью карты M(i,j), где в ячейке M содержится дескриптор, соответствующей ракурсам съемки дескриптора, для значений угла поворота детали i и j при построении набора базовых

дескрипторов (углам поворота модели θ(i)=i-90°, φ(j)=j° вокруг осей X, Z соответственно). На рисунке 34 показано соответствие ракурсов съемки дескриптора положению дескриптора на карте M(i,j).

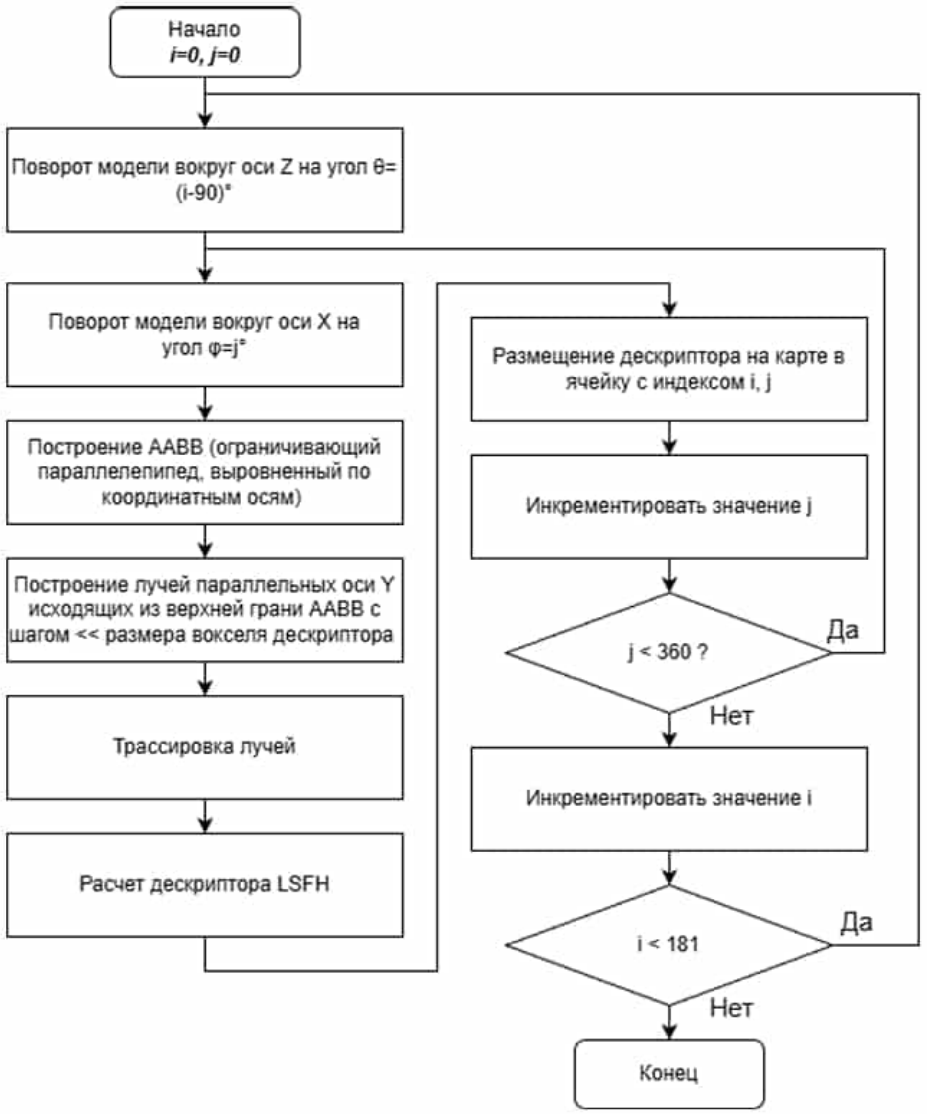


Рисунок 35 — Алгоритм формирования базовых дескрипторов

В главе 3.2 определена метрика численного сравнения дескрипторов (62). А значит к полученной карте можно применить методы обработки изображений, а именно оператор Собеля, позволяющий выделить границы областей на основе абсолютных значений градиентов и алгоритм водораздела (watershed algorithm [93]), позволяющий выделить похожие области на изображении. Однако, применить описанные методы в исходном виде не получится, так как карта

дескрипторов представляет положение ракурсов на сфере. А значит эти методы необходимо доработать.

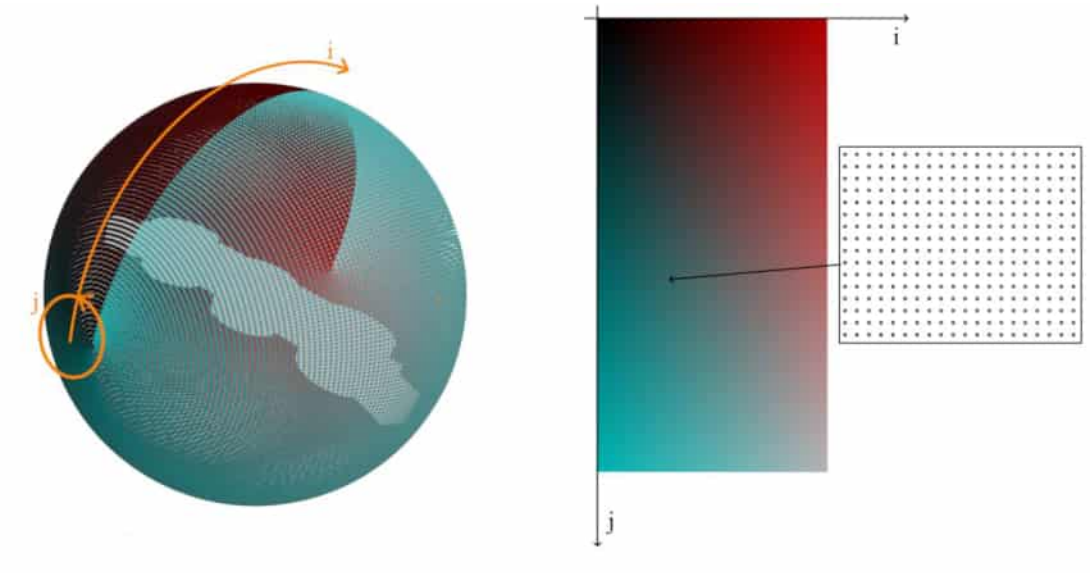


Рисунок 36 — Схема соответствия ракурсов карте дескрипторов $M(i,j)$

Оператор Собеля позволяет рассчитать карту изменения дескрипторов. Такая карта будет показывать абсолютную величину изменения дескрипторов в соседних ячейках. Однако из-за особенностей расположения дескрипторов на сфере, ракурсы $i = 0, j = 0..359$ и $i = 180, j = 0..359$, представлены ракурсами $i = 0, j = 0$ и $i = 180, j = 0$ соответственно, а дескрипторы $\mathrm{M}(i,-1) = \mathrm{M}(i,359)$, $\mathrm{M}(i,360) = \mathrm{M}(i,0)$.

Карта изменения дескрипторов строится по следующему алгоритму:

1. Для всех точек кроме i=0 и i=180 рассчитывается величина $\bar{G}(i,j)$ (75):

$$\bar{G}(i,j) = \sqrt{I_i(i,j)^2 + I_j(i,j)^2}, \tag{75}$$

где:

$$I_i(i,j) = \|M(i+1,j+1) - M(i-1,j-1)\| + 2\|M(i+1,j) + M(i-1,j)\| + \|M(i+1,j-1) - M(i-1,j+1)\|, \tag{76}$$

$$I_j(i,j) = \|M(i+1,j+1) - M(i-1,j-1)\| + 2\|M(i,j+1) - M(i,j-1)\| + \|M(i+1,j-1) - M(i-1,j+1)\|, \tag{77}$$

Для точек с индексами i=0 и i=180 значение $\bar{G}(i,j)$ рассчитывается по формуле (78):

$$\bar{G}(0,j) = \bar{G}(180,j) = 0. \quad (78)$$

2. Значение $\bar{G}(i,j)$ нормализуется к максимальному значению так, чтобы оно могло быть представлено в виде изображения в оттенках серого с глубиной цвета 8 бит (79):

$$G(i,j) = 255\frac{\bar{G}(i,j)}{\max(\bar{G}(i,j))}. \quad (79)$$

Чем меньше значение $G(i,j)$, тем меньше величина изменения дескриптора относительно соседних дескрипторов. На рисунке 37 изображен пример карты изменения дескриптора $G(i,j)$, где большая яркость соответствует большему значению величины G.

Для полученной карты $G(i,j)$ выполняется выделение областей с помощью алгоритма водораздела. С помощью этого алгоритма на карте изменения дескрипторов $G(i,j)$ выделяются области, разделенные границами — локальными максимумами области. Однако, чтобы области были выделены корректно на алгоритм накладываются ограничения, в связи с тем, что карта изменения дескрипторов также представляет положение ракурсов на сфере.

В работе выполнена доработка алгоритма водораздела, которая состоит в измененной процедуре выбора ближайших соседей для обрабатываемой точки. Ближайшие соседи выбираются исходя из того, что карта представляет положение точек на сфере (Рисунок 36). Особенными точками являются $G(0,0)$, $G(180,0)$. Для них ближайшими соседями являются все точки с индексами $i = 1, j = 0..359$ и $i = 179, j = 0..359$ соответственно. Для остальных точек ближайшими соседями является набор точек $G(i \pm 1, j \pm 1)$, $G(i, j \pm 1)$, $G(i \pm 1, j)$. Однако:

— если $i - 1 = 0$, то вместо $G(i-1,k), k = \text{j} - 1..j + 1$, используется точка $G(0,0)$;

— если $i+1=180$, то вместо $G(i+1,k), k=j-1..j+1$, используется точка $G(180,0)$;

— если $j-1=-1$, то $G(i,j-1)=G(i,359)$;

— если $j+1=360$, то $G(i,j+1)=G(i,0)$.

Точки $G(0,k)$, $G(180,k)$, где $k=1..359$ — не используются в явном виде при переборе в алгоритме водораздела.

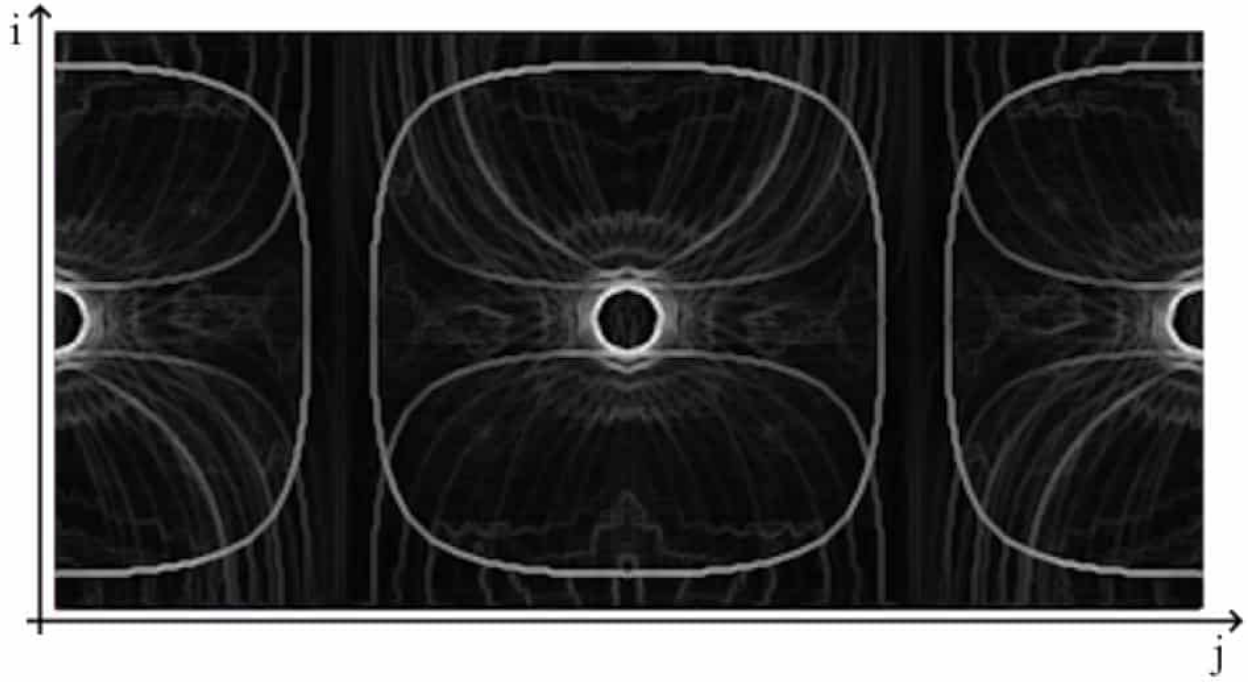


Рисунок 37 — Карта изменения дескриптора $G(i,j)$

Тогда с учетом доработок алгоритм водораздела может быть описан следующей последовательностью действий. Этап подготовки начинается с разделения всех точек на k наборов точек $M_k=(i,j)_k$ с близкими значениями $G(i,j)$. В работе экспериментально было выбрано значение $k=50$. Вводится понятия матрицы меток $L(i,j)$, где точки с одинаковыми значениями будут принадлежать одной области, а точки со значением $L=0$ — являться водоразделом, $L=-2$ — являться выделенными для обработки на текущем уровне (маркированными), $L=-3$ — добавленными в очередь на обработку. Вводится счетчик меток $l=1$. Матрица меток заполняется первоначальными

значениями $L(i,j) = -1$. Счетчик уровня устанавливается в значение $k_c = 0$. После этого производятся шаги:

1. Все точки текущего уровня M_{k_c} маркируются и добавляются в очередь, также в очередь добавляются ближайшие соседи для этих точек, выбранные по условиям, описанным ранее, для которых значение $L \geq 0$.
2. Производится постобработка граничных значений. Из очереди извлекается точка $g_c(i_c, j_c)$. Для этой точки производится поиск ближайших соседей. Для каждого из ближайших соседей $g_n(i_n, j_n)$ выполняется одно из действий в зависимости от условий:

   — если маркер соседа $L(i_n, j_n) > 0$ и если маркер текущей точки $L(i_c, j_c) = -3$ (находится в очереди) или данная точка уже была отмечена границей водораздела, во время обработки ближайших соседей, то ей присваивается значение метки $L(i_c, j_c) = L(i_n, j_n)$;

   — если значение метки ближайшего соседа $L(i_n, j_n) > 0$ и значение метки точки $L(i_c, j_c) > 0$, при этом значения меток не совпадают $L(i_n, j_n) \neq L(i_c, j_c)$, то текущей точке присваивается метка водораздела $L(i_c, j_c) = 0$, и это значение не может быть изменено;

   — если выбранный ближайший сосед обозначен границей водораздела $L(i_n, j_n) = 0$, а текущая точка в очереди $L(i_c, j_c) = -3$, то текущая точка помечается границей водораздела $L(i_c, j_c) = 0$;

   — если значение метки ближайшего соседа $L(i_c, j_c) = -2$, то ему присваивается маркер $L(i_c, j_c) = -3$, после чего точка g_n добавляется в очередь.

3. Производится проверка есть ли еще точки в очереди. Если есть, переход к пункту 2.
4. Счетчик меток l инкрементируется. Из текущего набора точек извлекается точка $g_{c1}(i_{c1}, j_{c1})$, если маска $L(i_{c1}, j_{c1}) = -2$, то точка

добавляется в очередь, а значению маски присваивается значение счетчика меток $L(i_{c1}, j_{c1}) = l$.

5. Из очереди извлекается точка $g_c(i_c, j_c)$. Для извлеченной точки выполняется проверка, есть ли ближайшие соседи $g_n(i_n, j_n)$, для которых значение маски равно $L(i_n, j_n) = -2$. Если такая точка есть, она помещается в очередь, а значению маски присваивается значение счетчика меток $L(i_n, j_n) = l$.
6. Производится проверка есть ли еще точки в очереди. Если есть, переход к пункту 5.
7. Производится проверка, есть ли еще необработанные точки в наборе. Если есть переход к пункту 4.
8. Производится проверка, есть ли еще необработанные наборы точек M_k. Если есть, счетчик текущего уровня k_c инкрементируется, после чего переход к пункту 1.

В результате работы алгоритма будет получен набор областей с близкими значениями дескриптора (Рисунок 38). Для каждой области задан массив индексов положения похожих дескрипторов на карте дескрипторов. Для выделенных областей по индексам дескрипторов формируется набор наборов дескрипторов $D = \{\{d_{k_e} = \mathrm{M}(i,j)\}_{e=1}^{n_k}\}_{k=1}^{m}$ где m — число выделенных областей, n_k — число дескрипторов в m-ом наборе, (i, j) — индексы точки номером e в k-й области. Полученный набор D — является первоначальным разделением на группы.

Первоначально выделенные группы можно представить одним дескриптором, однако, некоторые из них могут быть похожи, из-за симметрии детали. Поэтому после первоначального разделения на группы возникает задача обработки групп с целью объединения похожих областей, возникающих из-за ограничений алгоритма водораздела.

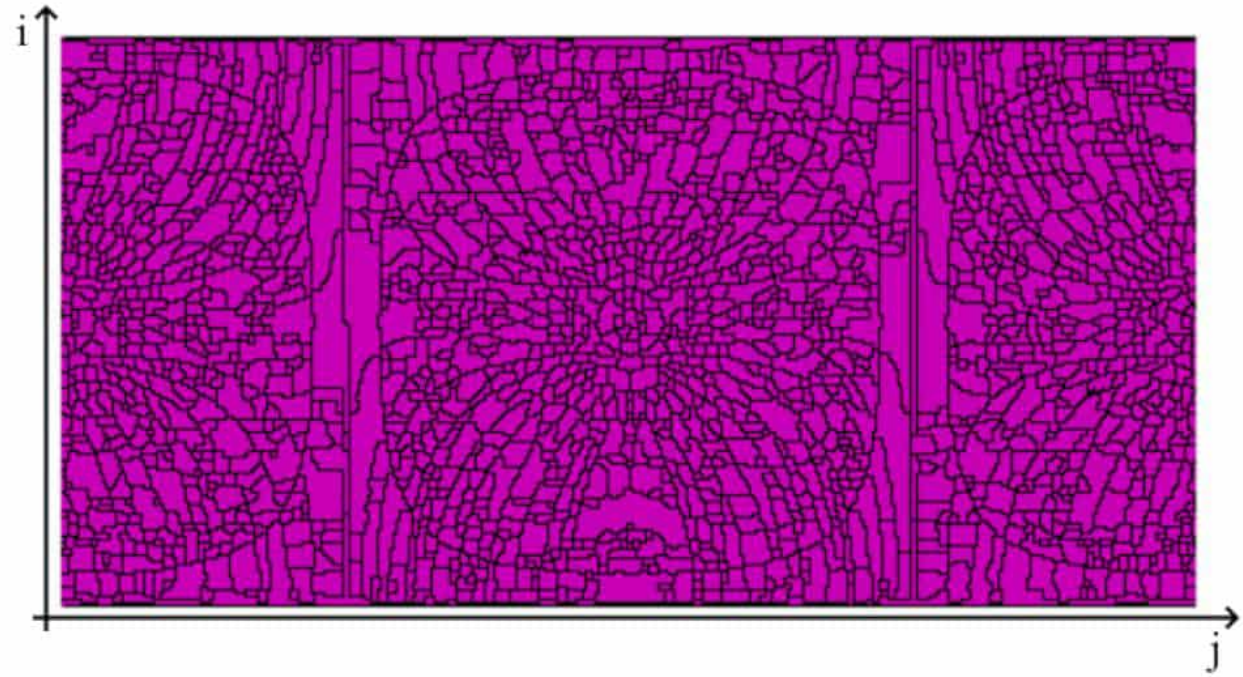


Рисунок 38 — Области, выделенные доработанным алгоритмом водораздела: черный цвет — граница областей; фиолетовый цвет — выделенные области

Для выполнения такой обработки необходимо выделить критерий похожести групп S_{thd} (84) с помощью следующего алгоритма:

1. Для каждого набора дескрипторов рассчитывается центральный дескриптор (80), а также среднее расстояние дескрипторов в группе до него (81):

$$D_{c_k} = d_{k_e}, \tag{80}$$

где $e = \underset{i}{\operatorname{argmin}}\left(\left\|d_{k_i} - \mathrm{c}_k\right\|\right)$, а $\mathrm{c}_k = \frac{1}{n_k}\sum_{i=1}^{n_k} d_{k_i}$.

Среднее расстояние dst_k до центрального дескриптора в группе рассчитывается по формуле (81):

$$dst_k = \frac{\sum_{i=1}^{n_k}\left\|D_{c_k} - d_{k_i}\right\|}{n_k - 1}. \tag{81}$$

На рисунке 39 показан пример гистограммы распределения значений dst_k.

2. Для первоначальных групп рассчитывается среднее значение c_{dst} (82) расстояния dst_k в группах и стандартное отклонение sd_c (83) значений

dst_k. После чего определяется пороговое значение похожести областей S_{thd} (84), являющийся критерием похожести групп, как величина среднего значения расстояния в группе плюс его стандартное отклонение:

$$c_{dst} = \frac{1}{n_k}\sum_{i=1}^{n_k} dst_k, \tag{82}$$

$$sd_c = \sqrt{\frac{\sum_{i=1}^{n_k}(c_{dst}-dst_k)^2}{n_k-1}}, \tag{83}$$

$$S_{thd} = c_{dst} + sd_c. \tag{84}$$

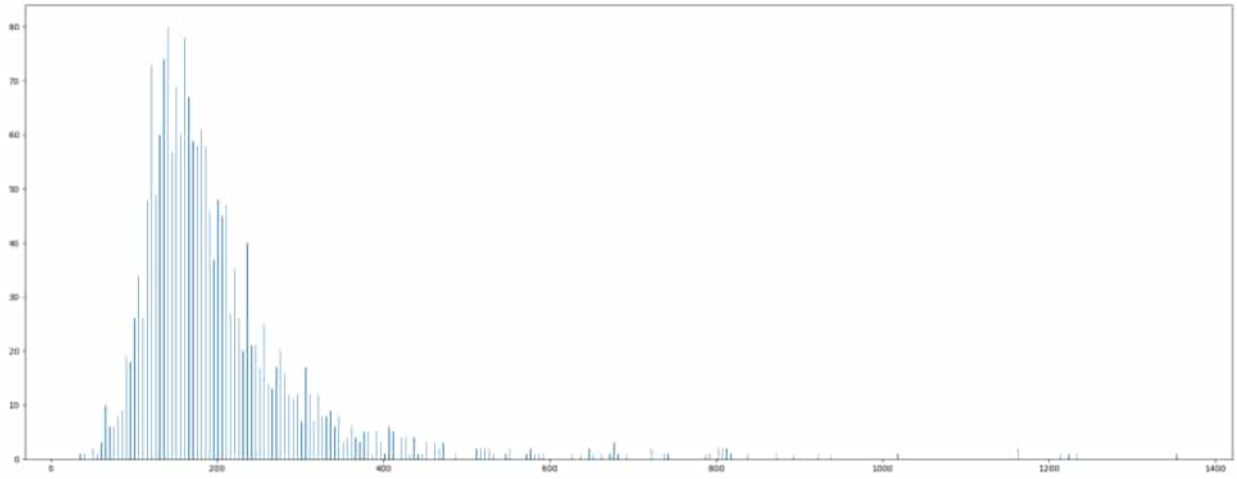


Рисунок 39 — Гистограмма значений dst_k для наборов дескрипторов D

3. По рассчитанному значению похожести групп, выполняется объединение похожих групп с помощью иерархической кластеризации и представление их одним дескриптором, расположенным ближе всех к центру группы. Набор таких дескрипторов представляет собой набор эталонных дескрипторов для детали.

Описанный алгоритм позволяет динамически выделять области похожих базовых дескрипторов, выбирать критерий похожести групп и формировать эталонные дескрипторы.

С учетом описанных задач проведения процесса ФЭД и предложенных подходов к их решению в работе предложена методика, основана представлении

групп похожих дескрипторов одним центральным дескриптором, где выбор порога похожести зависит от геометрии и размеров детали. Алгоритм, лежащий в основе разработанной методики ФЭД представлен следующими шагами:

1. Для полигональной модели детали генерируются базовые дескрипторы с 65160 ракурсов. После чего для полигональной модели детали происходит расчет объектно-ориентированного ограничивающего параллелепипеда для получения эталонного вектора длин ребер $D = (d_1, d_2, d_3)$ которые будут использоваться в процессе идентификации, где $D \in R^3$ — длины ребер параллелепипеда, при этом вектор D составлен таким образом, что выполняется условие $d_1 \leq d_2 \leq d_3$. Процесс производится аналогично расчету MOBB для облака точек, путем представления полигональной модели в виде облака точек, составленного из вершин полигонов модели, и подается на вход алгоритма расчета MOBB.
2. Из сгенерированных на первом шаге базовых дескрипторов строится карта изменения дескрипторов.
3. Производится первоначальное выделение групп дескрипторов с помощью доработанного алгоритма водораздела.
4. Производится выбор критерия похожести групп S_{thd} (84).
5. Группы со значением $dst_k > S_{thd}$ обозначаются «ошибочными» и разделяются на подгруппы, для которых $dst_k < S_{thd}$, заменяющие «ошибочные» наборы в исходной выборке.

   Для реорганизации групп используется алгоритм иерархической кластеризации (агломеративная кластеризация), со стратегией, которая минимизирует максимальное расстояние внутри кластеров. На вход алгоритма иерархической кластеризации подаются дескрипторы, входящие в «ошибочный» набор. В качестве порогового значения максимального расстояния, при котором кластеры не могут быть соединены устанавливается значение S_{thd}.

После разделения «ошибочной» группы на несколько групп, удовлетворяющим условию кластеризации, для каждой из полученных групп по формуле (80) рассчитывается средний дескриптор. В результате будут получены $\overline{D} = \{\{d_{\bar{k}i}\}_{i=1}^{n_{\bar{k}}}\}_{\bar{k}=1}^{\bar{m}}$ валидных кластеров и $\overline{\mathrm{C}} = \{c_{\bar{k}}\}_{\bar{k}=1}^{\bar{m}}$ средних дескрипторов для этих кластеров, где $\bar{k} > k$, $\bar{m}$ — результирующее число кластеров, после разделения. На рисунке 40 изображена гистограмма средних значений расстояния $dst_{\bar{k}}$ набора дескрипторов $\overline{D}$.

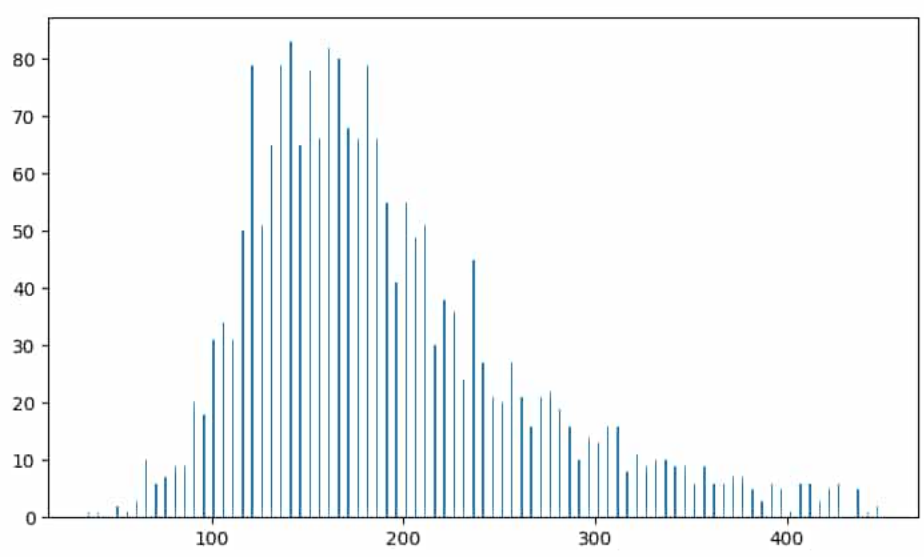


Рисунок 40 — Гистограмма значений $dst_{\bar{k}}$ для наборов дескрипторов $\overline{D}$ 1

6. Производится объединение полученных центральных дескрипторов в группы по выделенному критерию похожести. Шаг производится с целью объединить похожие дескрипторы, которые невозможно сгруппировать алгоритмом водораздела, например, вызванные симметрией детали. Каждый набор дескрипторов $\overline{D}$ представляется средним дескриптором $c_{\bar{k}}$ из набора $\overline{\mathrm{C}}$. Набор средних дескрипторов $\mathrm{C}_{\bar{k}}$ подается на вход алгоритма кластеризации с пороговым значением S_{thd}. В результате на выходе будут получены кластеры дескрипторов $E = \{\{e_{j\,i} \in \mathrm{C}_{\bar{k}}\}_{i=1}^{n_j}\}_{j=1}^{d}$, составленные из центральных дескрипторов, где n_j — число дескрипторов в j-м наборе, d — итоговое число наборов.

7. Для каждого кластера из набора E, полученного на шестом шаге, по формуле (80) рассчитываются d центральных дескрипторы C_{fin}. Набор центральных дескрипторов C_{fin} — набор эталонных дескрипторов, который будет использоваться в процессе идентификации детали.

В результате работы методики ФЭД для каждой детали будет получено d эталонных дескрипторов и параметры МОВВ.

Для детали №1 в результате работы алгоритма водораздела выделено 1918 групп D_k. После применения всего алгоритма ФЭД получено 637 эталонных дескриптора C_{fin}. На рисунке 41 показана карта эталонных дескрипторов для детали №1, и ракурсы, с которых были получен эти дескрипторы на модели детали.

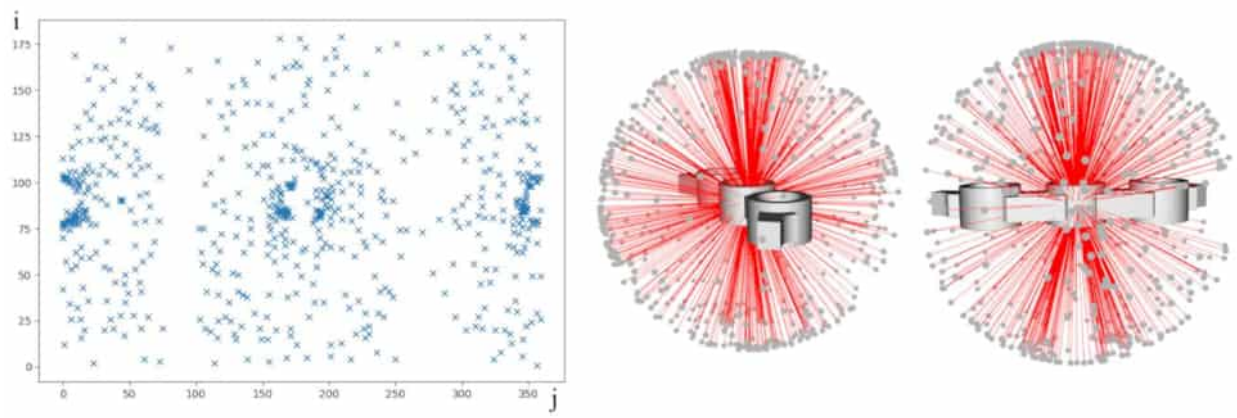


Рисунок 41 — Карта эталонных дескрипторов и ракурсы, с которых они были получены для детали №1

С помощью описанной методики формирования эталонных дескрипторов сформированы эталонные дескрипторы для всех цифровых моделей деталей. Количество эталонных дескрипторов, полученных для каждой детали приведено в таблице 9. Промежуточные результаты ФЭД для всех тестовых деталей представлены в приложении В.

Таблица 9 — Количество добавленных дескрипторов для деталей

№	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**19**	**11**	**12**	**среднее**
N	637	461	288	307	635	305	142	653	418	644	404	535	447

3.5. Разработка методики идентификации детали

В предыдущих главах описаны методики построения дескрипторов, подготовки облака точек и формирования эталонных дескрипторов. После применения описанных методик сформирован набор эталонных дескрипторов, а для каждого облака точек, которое необходимо идентифицировать получено несколько дескрипторов LSFH и параметры MOBB.

Детали выбраны так, что они не отличимые друг от друга с некоторых ракурсов. Поэтому возникает задача разработки методики идентификации, которая бы позволяла проводить идентификацию используя все полученные из общего облака точек дескрипторы и параметры MOBB. В работе выдвигается гипотеза, что использование нескольких дескрипторов LSFH, построенных по облакам точек, полученных с разных ракурсов и параметров MOBB позволит повысить точность идентификации.

Для решения описанной задачи предложена методика, которая использует несколько дескрипторов LSFH и параметры MOBB, принимая решение о соответствии облака точек конкретной детали с помощью голосования. Для этого производится расчет метрик соответствия дескрипторам и параметрам MOBB каждой из идентифицируемых деталей, в качестве результата идентификации выбирается та деталь, величина метрики соответствия к которой наибольшая. Алгоритм расчета метрик соответствия с помощью голосования, лежащий в основе методики идентификации с помощью голосования, представлен четырьмя последовательными шагами и изображен на рисунке 42:

1. Производится расчет соответствия дескрипторов каждой детали для идентификации. После этапа ФЭД для каждой детали сформирован набор эталонных дескрипторов. Для каждого дескриптора $LSFH_i$, полученного с устройства сканирования, по формуле (85) осуществляется поиск ближайшего дескриптора с расстоянием $d_i(j)$ в наборе j-й детали:

$$d_i(j) = \min\left(\left\|LSFH_i - LSFH_{j_e}\right\|\right), \quad (85)$$

где, $e = 1..m(j), m(j)$ — число эталонных дескрипторов для j-й детали. Чем меньше значение $d_i(j)$, тем больше дескриптор похож на эталонный, а значит должна быть выше метрика соответствия детали. Поэтому метрика соответствия дескриптора LSFH j-й детали $p_{\mathrm{L}_i}(j)$ рассчитывается по формуле (86):

$$p_{\mathrm{L}_i}(j) = 1 - \frac{d_i(j)}{\sum_{j'=1}^{k} d_i(j')}, \quad (86)$$

где k — число деталей для идентификации.

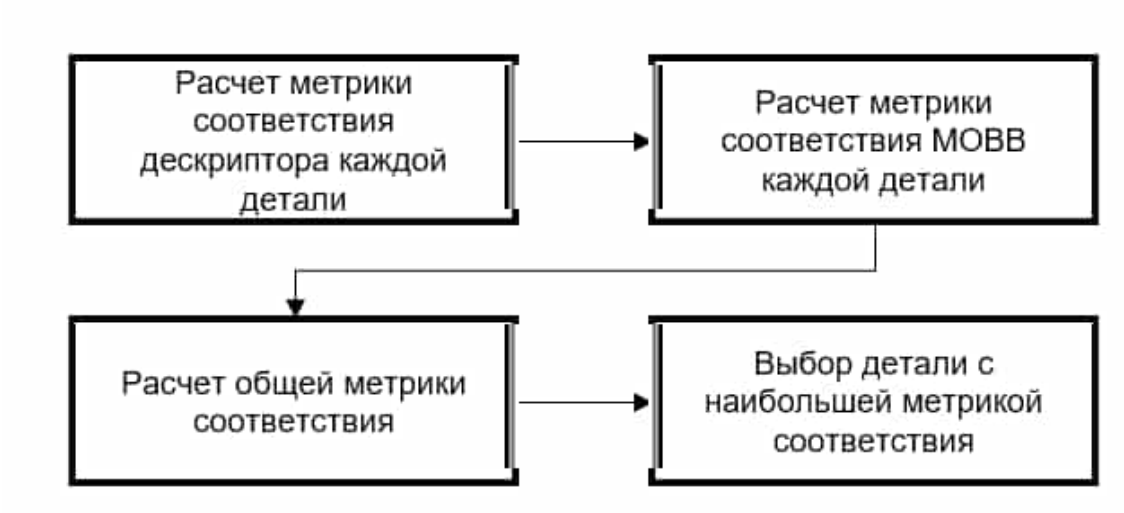


Рисунок 42 — Последовательность действий для расчета метрик соответствия с помощью голосования

2. Производится расчет соответствия полученных параметров MOBB каждой детали для идентификации. После этапа ФЭД для каждой j-й детали рассчитаны эталонные вектора длин ребер MOBB $D(j) \in R^3$, причем организованны таким образом, что $D = (d_1, d_2, d_3)$, где $d_1 \leq d_2 \leq d_3$. Для каждой детали для идентификации по формуле (87) рассчитывается расстояние между эталонным вектором длин ребер MOBB данной детали и вектором длин ребер MOBB для облака точек, полученным в процессе построения дескриптора LSFH:

$$l(j) = \|D - D(j)\|, \quad (87)$$

где $D(j)$ — длины ребер MOBB j-й детали, D — параметры MOBB, полученные во время построения дескриптора.

Чем меньше значение $l(j)$, тем больше MOBB похож на эталонный, а значит метрика соответствия детали $p_M(j)$ может быть рассчитана по формуле (88):

$$p_M(j) = 1 - \frac{l(j)}{\sum_{j'=1}^{k} l(j')}, \quad (88)$$

где k — число деталей для идентификации.

3. С помощью формулы (89) производится расчет общей метрики соответствия для каждой детали по всем рассчитанным на первом и вторым шагах метриках соответствия:

$$p(j) = \frac{p_M(j) + \sum_{i=1}^{n} p_{\mathrm{L}i}(j)}{n+1}, \quad (89)$$

где n — число дескрипторов (полученных с различных ракурсов), используемых в процессе идентификации.

4. Производится идентификация детали. Как результат идентификации выбирается та деталь, итоговая метрика соответствия к j-й детали $p(j)$ наибольшая (90):

$$J_{res} = argmax\big(p(j)\big). \quad (90)$$

На рисунке 43 изображен пример значений метрик соответствия в процессе идентификации детали №8 с помощью описанной методики.

Разработанная методика расчета метрик соответствия с помощью голосования принимает на вход два и более дескрипторов LSFH и параметры MOBB, по полученным ранее наборам эталонных дескрипторов деталей выполняется поиск соответствий, в результате чего на выходе выдается индекс детали с наибольшим значением соответствия к входным данным.

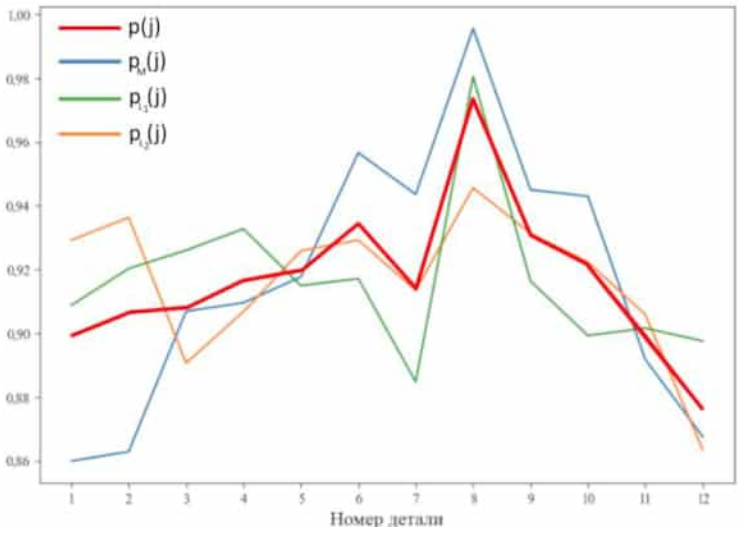


Рисунок 43 — Пример значений метрик соответствия в процессе идентификации детали №8

С учетом разработанных ранее методик, алгоритм идентификации состоит из следующих шагов и изображен на рисунке 44:

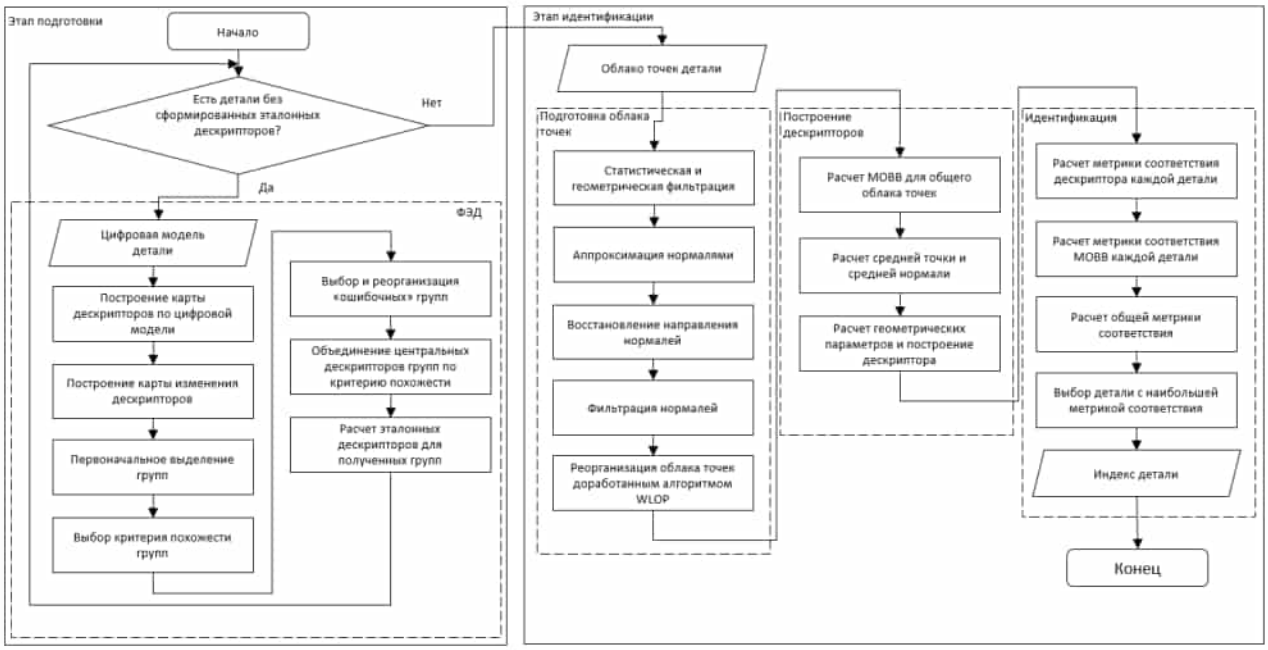


Рисунок 44 — Блок-схема методики идентификации детали

1. На этапе подготовки выполняется формирование эталонных дескрипторов для цифровых моделей деталей по методике, описанной в главе 3.4.
2. На этапе идентификации для полученных облаков точек выполняется последовательно:

— подготовка облаков по методике, описанной в главе 3.3. ;

— построение дескрипторов LSFH по методике, описанной в главе 3.2. ;

— идентификация детали по методике идентификации с помощью голосования, описанной в данной главе.

3.6. Оценка точности идентификации с помощью разработанных методик

Для проведения экспериментального исследования было разработано программное обеспечение, реализующее описанные ранее методики: подготовки облака точек, идентификации и ФЭД. Разработанное ПО будет подробно рассмотрено в главе 4.2. С помощью этого ПО проведено экспериментальное исследование точности работы методик на облаках точек, полученных с помощью эмуляции и для реальных данных. Всего было использовано 2448 облаков точек, полученных с помощью эмуляции и 240 облаков точек с реального устройства сканирования. В таблице 10 отражены результаты сравнения точности идентификации с помощью дескриптора LSFH по одному ракурсу (в таблице обозначено буквой «А») и с помощью разработанной методики голосования (в таблице обозначено «Агол») для облаков точек, полученных с помощью эмуляции с лучшими показателями экспериментального исследования точности идентификации с помощью дескрипторов VFH, CVFH, проведенных в главе 1. Параметр точности оценивался по формуле (50).

В таблице 11 приведено сравнение времени идентификации с помощью описанной методики голосования с идентификациями с помощью дескрипторов CVFH, VFH для набора облаков точек, полученных с помощью эмуляции. В таблицах 12, 13 представлена матрица ошибок для полученных результатов идентификации по одному ракурсу и с помощью разработанной методики голосования соответственно.

Таблица 10 — Сравнение точности идентификации с помощью дескрипторов VFH, CVFH, LSFH на облаках точек, полученных с помощью эмуляции

	VFH	**CVFH**	**LSFH**	
Деталь №	**А**	**А**	**А**	**Агол**

1	0,39	0,54	0,86	1
2	0,19	0,15	0,6	1
3	0,17	0,49	0,96	1
4	0,42	0,58	0,98	1
5	0,17	0,19	0,8	1
6	0,67	0,41	0,97	1
7	0,31	0,24	0,8	0,98
8	0,25	0,34	0,94	1
9	0,2	0,36	0,96	1
10	0,87	0,79	1	1
11	0,3	0,42	0,95	1
12	0,5	0,48	0,91	1
среднее	**0,37**	**0,41**	**0,89**	**0,998**

Таблица 11 — Сравнение времени идентификации с помощью дескрипторов VFH, CVFH, LSFH на облаках точек, полученных с помощью эмуляции

	VFH	**CVFH**	**LSFH**
Деталь №	**T c**	**T c**	**T c**
1	0,3872	7,532	0,0142
2	0,533	13,994	0,0175
3	0,464	7,321	0,0154
4	0,458	8,127	0,0155
5	0,459	9,21	0,0174
6	0,445	11,27	0,0126
7	0,496	8,19	0,014
8	0,464	6,13	0,0132
9	0,461	8,91	0,0131
10	0,451	7,192	0,0124
11	0,43	8,759	0,0145
12	2,3	9,218	0,0151
среднее	**0,61235**	**8,821083333**	**0,014575**

Таблица 12 — Матрица ошибок при идентификации по одному ракурсу с помощью дескриптора LSFH на облаках точек, полученных с помощью эмуляции

№	1	2	3	4	5	6	7	8	9	10	11	12
1	**176**	1	0	0	0	4	0	0	9	14	0	0
2	33	**122**	0	0	0	25	0	0	16	8	0	0
3	0	0	**196**	4	0	0	0	4	0	0	0	0
4	0	0	3	**201**	0	0	0	0	0	0	0	0
5	0	1	0	0	**165**	3	1	20	0	11	3	0
6	0	2	0	0	0	**198**	0	0	0	4	0	0
7	2	0	0	0	0	34	**163**	0	0	3	0	2
8	0	0	1	0	1	1	0	**192**	0	6	0	3
9	0	0	0	0	0	0	0	2	**196**	6	0	0
10	0	0	0	0	0	0	0	0	0	**204**	0	0
11	0	1	0	1	2	0	0	0	0	2	**194**	4
12	0	1	0	2	5	2	0	5	0	0	2	**187**

Таблица 13 — Матрица ошибок при идентификации с помощью методики идентификации с помощью голосования на облаках точек, полученных с помощью эмуляции

№	1	2	3	4	5	6	7	8	9	10	11	12
1	**102**	0	0	0	0	0	0	0	0	0	0	0
2	0	**102**	0	0	0	0	0	0	0	0	0	0
3	0	0	**102**	0	0	0	0	0	0	0	0	0
4	0	0	0	**102**	0	0	0	0	0	0	0	0
5	0	0	0	0	**102**	0	0	0	0	0	0	0
6	0	0	0	0	0	**102**	0	0	0	0	0	0
7	0	0	0	0	0	2	**100**	0	0	0	0	0
8	0	0	0	0	0	0	0	**102**	0	0	0	0
9	0	0	0	0	0	0	0	0	**102**	0	0	0
10	0	0	0	0	0	0	0	0	0	**102**	0	0
11	0	0	0	0	0	0	0	0	0	0	**102**	0
12	0	0	0	0	0	0	0	0	0	0	0	**102**

Из сравнения видно, что использование дескриптора LSFH позволило повысить точность идентификации до 0,89 по одиночному дескриптору в сравнении с методиками идентификации с использованием дескрипторов VFH, CVFH, а методика идентификации с помощью голосования позволила еще больше повысить точность идентификации до 0,998. При этом время необходимое на идентификацию уменьшилось в 42 раза по сравнению с идентификацией с помощью дескриптора VFH и в более чем 600 раз по сравнению с идентификацией с помощью CVFH.

Для подтверждения полученных результатов проведен повторный эксперимент на произвольных деталях. Для этого выбраны 5 деталей вид и габариты которых представлены в таблице 14. Для выбранных деталей проведена эмуляция сканирования и получено 1020 облаков точек. По описанной методике проведено ФЭД, промежуточные результаты которого представлены в приложении В. Результаты идентификации с одного ракурса и использованием разработанной методики идентификации с помощью голосования, а также количество эталонных дескрипторов для каждой детали отражены в таблице 15. Матрицы ошибок для полученных результатов идентификации произвольных деталей по одному ракурсу и с помощью разработанной методики голосования приведены в таблицах 16 и 17 соответственно.

Таблица 14 — Вид и геометрия деталей для дополнительного тестирования методик

№	Изображения детали	Габариты мм
1		19×22×19
2		39×39×11

3		39×24×52
4		42×48×42
5		45×17×6

Таблица 15 — Точность идентификации с помощью дескриптора LSFH по одному ракурсу и с помощью разработанной методики идентификации с помощью голосования на облаках точек произвольных деталей, полученных с помощью эмуляции

Деталь №	**A**	**Агол**	**Эталонные дескрипторы**
1	0,86	1	251
2	0,84	1	327
3	0,76	0,98	621
4	0,79	1	687
5	0,68	1	983
среднее	**0,786**	**0,996**	**574**

Таблица 16 — Матрица ошибок при идентификации по одному ракурсу с помощью дескриптора LSFH на облаках точек произвольных деталей, полученных с помощью эмуляции

№	**1**	**2**	**3**	**4**	**5**
1	175	0	0	0	29
2	30	172	0	0	2
3	25	0	156	9	14
4	32	3	0	162	7
5	24	3	0	38	139

Таблица 17 — Матрица ошибок при идентификации с помощью методики идентификации с помощью голосования на облаках точек произвольных деталей, полученных с помощью эмуляции

№	1	2	3	4	5
1	102	0	0	0	0
2	0	102	0	0	0
3	0	1	101	0	0
4	0	0	0	102	0
5	0	0	0	0	102

Использование методики идентификации с помощью голосования на произвольных деталях позволило повысить точность идентификации с 0,786 до 0,996 по сравнению с идентификацией по одному дескриптору. Целесообразность использования разработанной методики идентификации подтверждается повышением точности при экспериментальном исследовании на тестовых и произвольных деталях.

В таблице 18 представлено сравнение точности идентификации с помощью методики идентификации с помощью голосования для облаков точек, полученных в результате эмуляции и реального сканирования. В таблице 19 приведена матрица ошибок для полученных результатов идентификации на реальных данных.

Таблица 18 — Сравнение точности идентификации с помощью методики идентификации с помощью голосования на данных, полученных в результате эмуляции и реального сканирования

	Эмуляция	**Реальное сканирования**
Деталь №	**A**	**A**
1	1	1
2	1	0,9
3	1	1
4	1	1
5	1	1
6	1	1

7	0,98	1
8	1	1
9	1	1
10	1	1
11	1	1
12	1	1
среднее	**0,998**	**0,99**

Таблица 19 — Матрица ошибок при идентификации с помощью методики идентификации с помощью голосования на облаках точек, полученных с реального устройства сканирования

№	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	**10**	0	0	0	0	0	0	0	0	0	0	0
2	0	**9**	0	0	0	0	0	0	0	0	0	1
3	0	0	**10**	0	0	0	0	0	0	0	0	0
4	0	0	0	**10**	0	0	0	0	0	0	0	0
5	0	0	0	0	**10**	0	0	0	0	0	0	0
6	0	0	0	0	0	**10**	0	0	0	0	0	0
7	0	0	0	0	0	0	**10**	0	0	0	0	0
8	0	0	0	0	0	0	0	**10**	0	0	0	0
9	0	0	0	0	0	0	0	0	**10**	0	0	0
10	0	0	0	0	0	0	0	0	0	**10**	0	0
11	0	0	0	0	0	0	0	0	0	0	**10**	0
12	0	0	0	0	0	0	0	0	0	0	0	**10**

Полученные результаты на облаках точек, полученных с помощью методики эмуляции, а также на реальных данных, полученных с макета системы лазерного сканирования, подтверждают целесообразность использования разработанной методики идентификации.

3.7. Выводы по главе

В главе описан процесс разработки методики идентификации детали на конвейерной линии по облакам точек, полученных с помощью лазерного 3D сканирования. В ходе которого:

1. Проведено исследование результатов идентификации с помощью глобальных дескрипторов CVFH, VFH. Проанализированы причины полученных результатов. Выделены недостатки существующих методик применимо к деталям сложной геометрии, а именно ошибки определения средней точки, расчета средней нормали и неоптимальная методика ФЭД.
2. Разработана методика построения дескриптора LSFH на базе глобальных дескрипторов CVFH, VFH, предназначенная для описания деталей сложной геометрии, путем определения средней точки по параметрам MOBB и адаптированной компоненте LSSDC.
3. Проведено исследование влияния шумов на построение дескриптора LSFH, на основании которого сформированы требования к разрабатываемой методики подготовки облака точек. Разработана методика подготовки облака точек по описанным требованиям, включающая в себя этапы статистической фильтрации, фильтрации нормалей и реорганизации облака точек для лучшего соответствии поверхности. Методика позволила уменьшить расстояние до ближайшего эталонного дескриптора на 22% в среднем и повысить точность расчета параметров MOBB на 43% в среднем.
4. Разработана методика формирования эталонных дескрипторов на основе свойств разработанного дескриптора и подходов к обработке изображения, которая позволяет динамически выбирать ракурсы на основе сложности геометрии детали. Методика ФЭД применена к тестовым деталям в результате чего получено в среднем 447 эталонных дескриптора для каждой детали, что позволило уменьшить время

идентификации в 42 и 600 раз по сравнению с идентификацией с помощью дескрипторов VFH и CVFH соответственно.

5. Разработана методика идентификации с помощью голосования, позволяющая повысить точность идентификации путем использования дескрипторов, полученных с разных ракурсов, и информацию о геометрических параметрах детали одновременно. С помощью разработанной методики была повышена точность идентификации с 0,89 до 0,998 на эмулированных облаках точек тестовых деталей и с 0,786 до 0,996 на эмулированный облаках точек произвольных деталей.
6. Проведена оценка точности идентификации на реальных облаках точек тестовых деталей, в результате которой было получена точность 0,99.

Разработанная методика идентификации реализует в себе все этапы необходимые для осуществления процесса идентификации с помощью классических алгоритмов (Рисунок 4), а именно:

— этап формирования набора эталонных дескрипторов деталей;

— этап подготовки облака точек;

— этап построения дескриптора;

— этап идентификации детали по дескрипторам.

Необходимо интегрировать разработанную методику в состав ИИС.

Глава 4. Разработка программного обеспечения и применение разработанных методик

4.1. Программное обеспечение для эмуляции лазерного сканирования

Для разработанной методики эмуляции лазерного сканирования разработано ПО [9] на языках программирования C++ и CUDA с использованием библиотек OpenCV 4 [94], Eigen и Open3D [95].

Язык программирования C++ выбран из-за производительности, а CUDA используется для реализации параллельных матричных вычислений на графическом процессоре ПК. С помощью CUDA реализованы алгоритмы трассировки лучей.

Библиотека Eigen, написанная на языке программирования C++, предназначена для оптимизации матричных вычислений, с помощью нее реализуются все операции над матрицами, описанные в методике эмуляции сканирования.

Библиотека OpenCV 4, написанная на языке программирования C++, предоставляет широкий функционал для работы и обработки изображений. В ПО используется для формирования изображения с камеры устройства сканирования, обработки полученного изображения, и выделения лазерного луча на полученном изображении.

Библиотека Open3D, также, как и библиотека OpenCV 4 написана на C++, использует внутри библиотеку Eigen. Эта библиотека предназначена для работы с цифровыми моделями и облаками точек. Предоставляет широкие возможности по обработке облаков точек и цифровых моделей, а также возможности их сохранения, загрузки и визуализации. В ПО используется для осуществления перемещения и поворотов цифровых моделей, их загрузки и визуализации.

Разработанное ПО состоит из следующих компонентов, схема взаимодействия которых изображена на рисунке 45:

1. ПО визуализации схемы сканирования. Оконное приложение, разработанное на языке C++ с использованием библиотеки Open3D. Предназначено для пошагового просмотра конфигурации сцены для каждой итерации процесса эмуляции сканирования. Используется для проверки правильности расположения детали на сцене сканирования, а также корректность заданных параметров устройства сканирования. В окне отображается цифровая модель детали, положение камеры и лазера, а также видимая область камеры и плоскость лазера. С помощью нажатия клавиш на клавиатуре осуществляется переход между итерациями сканирования, что позволяет визуализировать расположение детали для всего процесса сканирования. Такая визуализация позволяет проверить виден ли луч лазера на камере устройства сканирования во время той или иной итерации. На вход программа принимает конфигурацию процесса эмуляции сканирования в формате JSON. Изображение вывода программы, а также пример конфигурации изображен на рисунке 46.

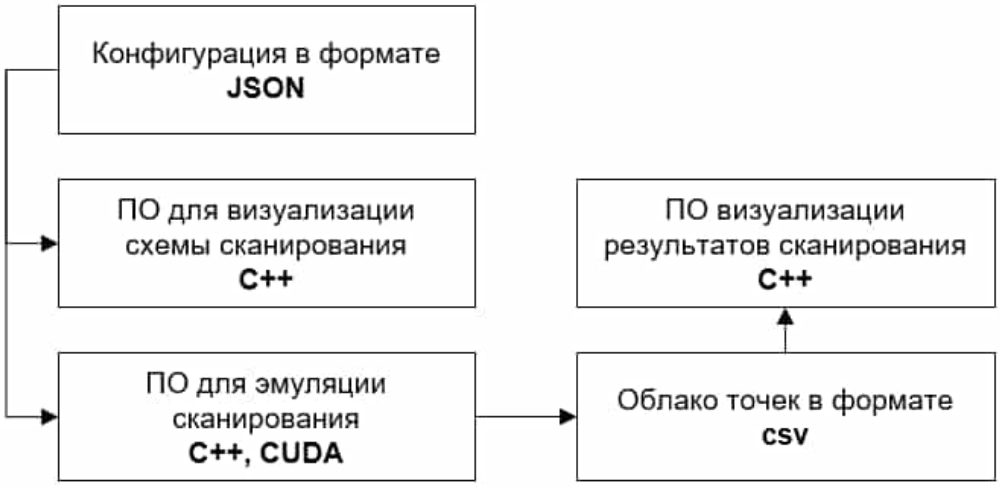


Рисунок 45 — Схема взаимодействия компонентов ПО для эмуляции лазерного сканирования

2. ПО для эмуляции сканирования. Консольное приложение, разработанное на языках C++, CUDA с использованием библиотек OpenCV 4, Open3D и подходов из библиотеки [96] для реализации

алгоритма трассировки лучей на графическом процессоре ПК. Предназначено для проведения эмуляции сканирования по разработанной методике. На вход принимает конфигурацию процесса эмуляции сканирования в формате JSON, в которой указаны параметры системы, путь к цифровой модели детали, а также ее габариты, положение на сцене, на выходе генерирует облако точек по описанной методике эмуляции лазерного сканирования и сохраняет его в формате CSV по указанному пути. Во время работы отображает промежуточную информацию о номере обрабатываемой итерации для определения статуса выполнения эмуляции.

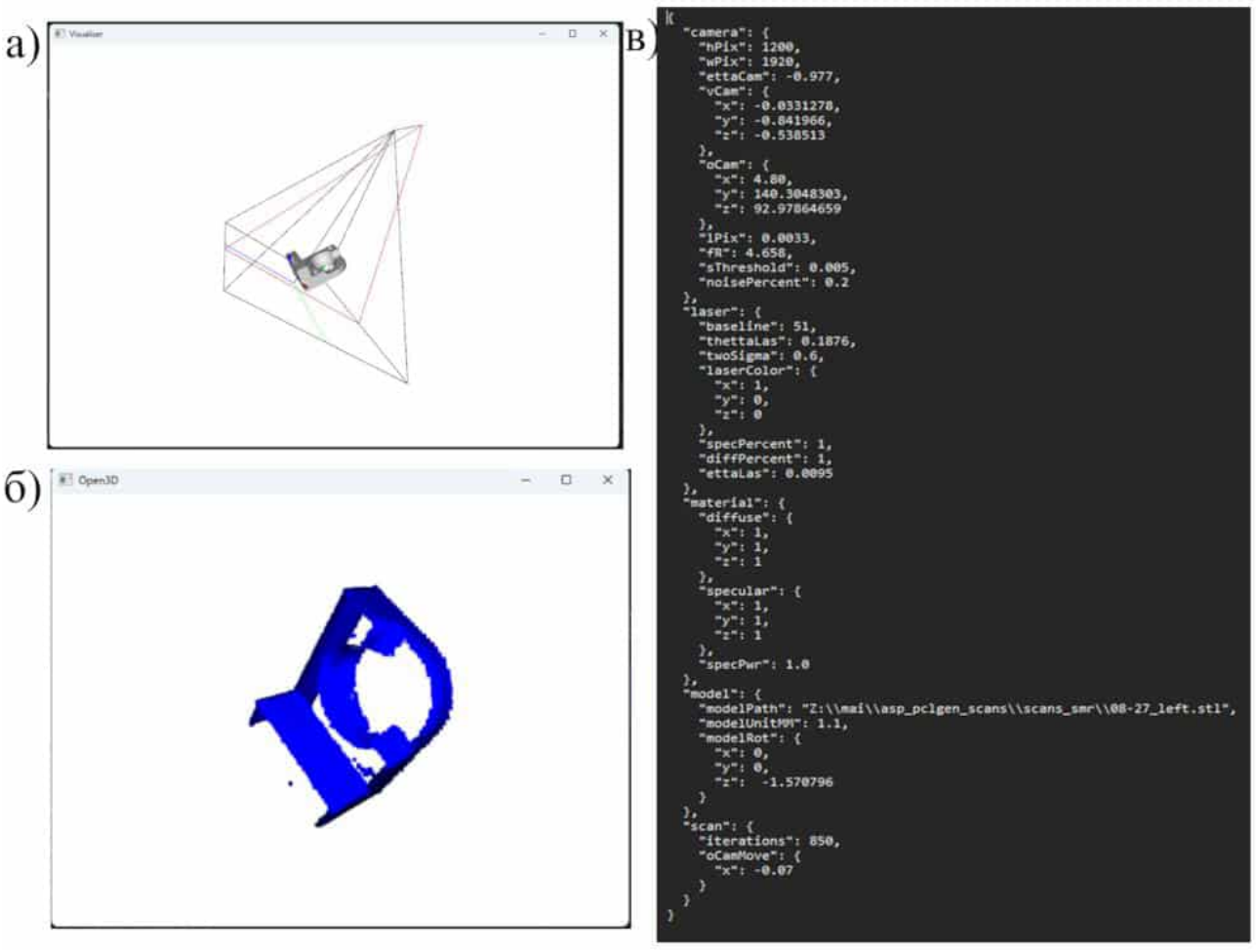


Рисунок 46 — Компоненты ПО эмуляции лазерного сканирования: а — окно ПО для визуализации схемы сканирования; б — окно ПО просмотра результатов сканирования; в — конфигурация процесса эмуляции в формате JSON

3. ПО визуализации результатов сканирования. Графическое приложение, разработанное на языке C++ с использованием библиотеки Open3D. Предназначено для просмотра результатов сканирования, с одного или нескольких устройств сканирования. На вход принимает облако точек, сохраненное в формате CSV. Отображает графическое окно с облаком точек. Взаимодействие производится с помощью клавиатуры и мыши. Предоставляется возможность приближения, отдаления и изменения ракурса положения камеры.

4.2. Программное обеспечение для идентификации деталей

ПО для идентификаций детали реализуют весь необходимый функционал для проведения процесса идентификации, для этого реализованы методика подготовки облака точек, методика формирования эталонных дескрипторов, методика построения дескриптора LSFH и методика идентификации с помощью голосования. Реализованы рассмотренные методики с помощью языков C++, CUDA и библиотек Eigen, Open3D, рассмотренных ранее, а также с помощью языка Python3 и библиотек numpy и scipy.

Язык Python 3 использовался с целью упрощения проведения экспериментальных исследований при работе с большим количеством дескрипторов. Библиотека numpy предназначена для оптимизации матричных операций, а также возможности сохранения и загрузки матриц в дисковое хранилище. Библиотека scipy реализует различные научные и инженерные алгоритмы. В разработанном ПО из библиотеки scipy используется реализация алгоритма иерархической кластеризации.

Программное обеспечение состоит из следующих компонентов, схема взаимодействия которых изображена на рисунке 47:

1. Программное обеспечение для генерирования базовых дескрипторов. Консольное приложение, разработанное на языках C++, CUDA с использованием библиотеки Open3D и подходов из библиотеки [95] для

реализации алгоритма трассировки лучей на графическом процессоре ПК. На вход программа принимает STL модель детали в масштабе соответствующему реальному. На выходе генерирует файл в текстовом формате, содержащий 65160 дескрипторов, а также информацию о ракурсах, с которых они были получены. В процессе работы в консоль выводит информацию о текущем угле поворота модели детали.

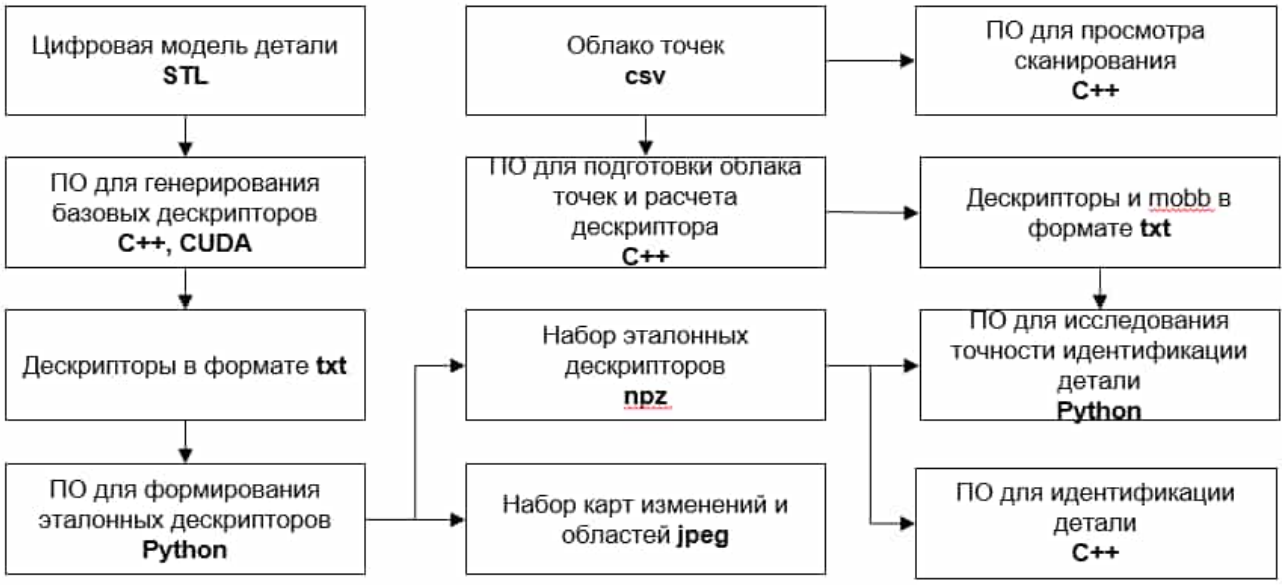


Рисунок 47 — Схема взаимодействия компонентов ПО для идентификации деталей

2. Программное обеспечение для формирования эталонных дескрипторов. Реализует шаги 2-7 разработанной методики формирования эталонных дескрипторов. Разработано на языке программирования Python 3 с использованием библиотек numpy, scipy. На вход принимает набор дескрипторов с информацией о ракурсах, с которых они были получены, в текстовом формате, полученных в результате работы ПО для генерирования базовых дескрипторов, на выходе генерирует набор эталонных дескрипторов, сохраняя их на диск в сжатом формате npz, используемым библиотекой numpy как стандартный формат хранения, и карт изменений и выделенных областей в формате jpeg, для визуального просмотра результатов. Вывод программы для формирования эталонных дескрипторов изображен на рисунке 48. В консоль

отображается информация о количестве выделенных дескрипторов в группах, количестве групп, информация о критерии похожести групп и финальном числу эталонных дескрипторов для детали.

Рисунок 48 — Вывод консольного приложения для формирования эталонных дескрипторов

3. ПО для просмотра результатов сканирования. Позволяет просмотреть облако точек, полученное с лазерного устройства сканирования, а также вид облака точек до и после фильтрации. Графическое приложение, разработанное на языке программирования C++ с использованием библиотеки Open3D. На вход получает облако точек, полученное с устройства сканирования в формате CSV, после чего отображает его пользователю в графическом окне с возможностью приближения, отдаления и изменения ракурса положения камеры, для детального рассмотрения облака точек. Вывод программы просмотра результатов сканирования изображен на рисунке 49.

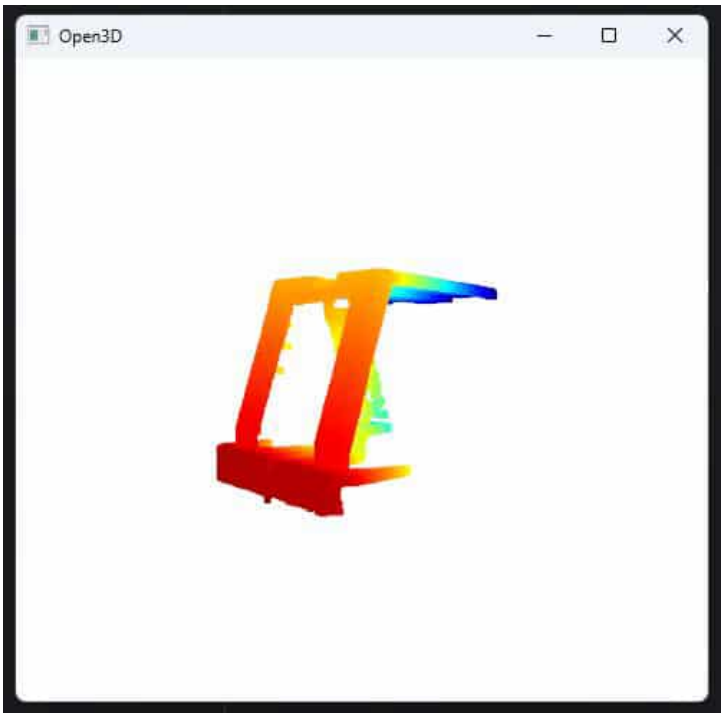


Рисунок 49 — Окно приложения для просмотра результатов сканирования

4. Программное обеспечение для подготовки облака точек и расчета дескриптора. Используется для подготовки облака точек по описанной методике подготовки и расчета дескрипторов LSFH и параметров MOBB. Консольное приложение, разработанное на языке программирования C++ с использованием библиотеки Open3D. На вход получает облако точек, полученное с устройства сканирования в формате CSV, на выходе генерирует текстовый файл, содержащий дескрипторы, а также информацию о параметрах MOBB.
5. ПО для исследования точности идентификации деталей. Реализует методику идентификации деталей с помощью голосования, разработано для оценки точности идентификации. Консольное приложение, разработанное с помощью языка программирования Python 3 с использованием библиотеки numpy. На вход получает набор эталонных дескрипторов в формате npz и набор текстовых файлов дескрипторов для набора облаков точек, полученных в результате работы ПО для подготовки облака точек и расчета дескриптора. Выполняет процесс идентификации для всех дескрипторов набора точек, сравнивает

полученные результаты с заданными. После чего рассчитывает точность идентификации для заданной детали. На выходе выдает сообщение, содержащее точность идентификации по одному дескриптору, точность идентификации с помощью методики голосования и матрицу ошибок для первого и второго типа идентификации. Вывод программы для идентификации деталей изображен на рисунке 50.

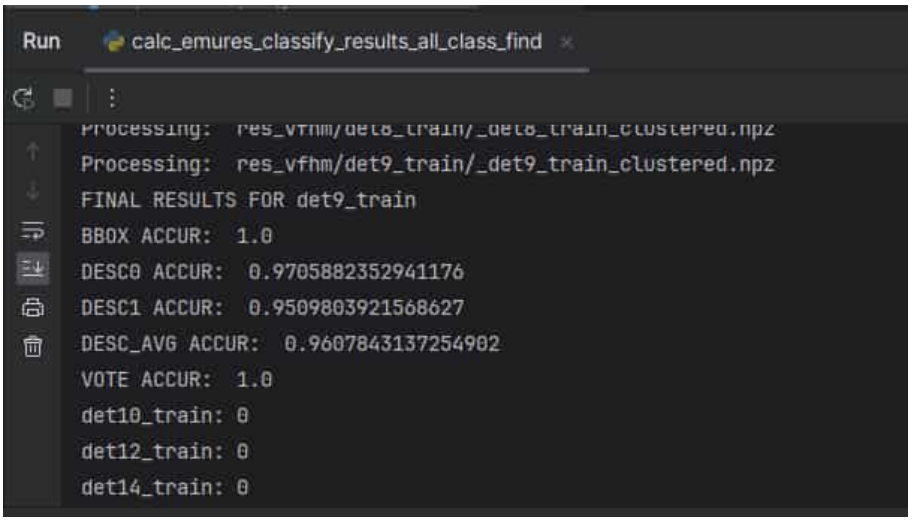


Рисунок 50 — Вывод консольного приложения для идентификации деталей

6. ПО для идентификации деталей. Полностью реализует функционал этапа идентификации из алгоритма методики идентификации деталей (Рисунок 44). В отличии от ПО для исследования точности идентификации деталей, является комплексным приложением осуществления полного процесса идентификации по облаку точек. Консольное приложение, разработанное для ОС LINUX (предполагается использование механизма PIPE для работы с потоками ввода и вывода) с помощью языка программирования C++. Из стандартного потока ввода получает облако точек с устройства сканирования, в стандартный поток вывода выдает информацию об идентифицируемой детали. Для работы программы необходимо передать набор эталонных дескрипторов.

Вырезки кода разработанного ПО приведены в приложении Г.

4.3. Информационно-измерительная система идентификации деталей

Информационно измерительная система идентификации детали на конвейере по облаку точек полученного с устройств лазерного сканирования представлена следующими модулями:

— модуль сканирования детали;

— модуль подготовки к идентификации;

— модуль идентификации детали.

Схема описанной ИИС изображена на рисунке 51.

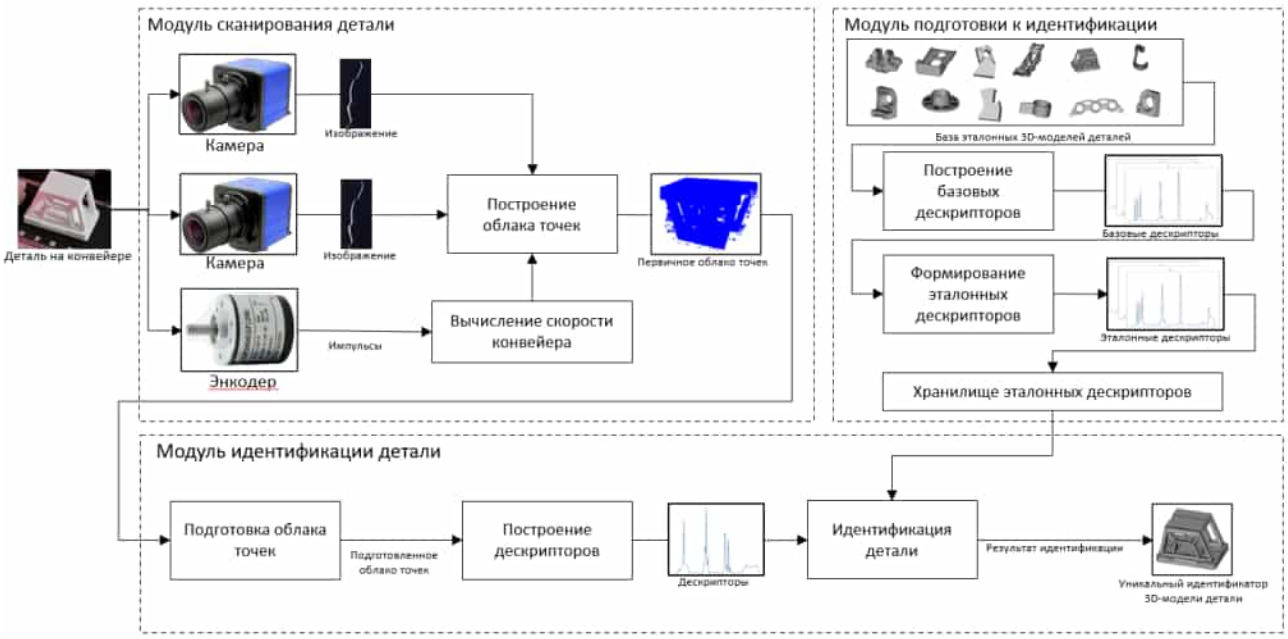


Рисунок 51 — Схема информационно измерительной системы идентификации детали

Модуль сканирования детали предназначен для проведения сканирования с целью получения облака точек детали. Модуль сканирования состоит из:

1. Камер, осуществляющих съемку детали с нескольких ракурсов, обработки изображения с целью удаления заднего фона и контрастирования лазерного луча. На выходе блоков камер формируются контрастные изображения луча лазера на поверхности детали без заднего фона.
2. Энкодера, генерирующего импульсы на движение конвейерной ленты.

3. Функционального блока вычисления скорости конвейера. Этот блок преобразует импульсы энкодера в параметры моментальной скорости движения конвейера. На выходе блока формируется скорость конвейерной ленты.
4. Функционального блока построения облака точек. Этот блок осуществляет выделение лазерного луча на изображении, построение по ним «срезов» луча лазера и размещение их в пространстве согласно информации, полученной от блока вычисления скорости конвейера. После чего выполняется сегментация, с целью выделения точек принадлежащей детали. На выходе функционального блока формируется первичное облако точек.

Модуль идентификации детали предназначен для осуществления идентификации детали по ее облаку точек. На вход модуля подается облако точек детали с модуля сканирования детали, на выходе формируется уникальный идентификатор 3D-модели детали. Модуль идентификации реализован с помощью следующих функциональных блоков:

1. Функциональный блок подготовки облака точек осуществляет подготовку облака точек по методики подготовки, описанной в главе 3.3. На вход блок принимает первичное облако точек детали, на выходе формируется подготовленное облако точек — облако точек, к которому применены фильтры, аппроксимированы и отфильтрованы нормали в каждой точке облака, проведена реорганизация облака.
2. Функциональный блок построения дескрипторов выполняет построение дескрипторов LSFH и расчет параметров MOBB по методике, описанной в главе 3.2. На вход блока подается подготовленное облако точек, на выходе формируются несколько дескрипторов LSFH, в зависимости от количества устройств сканирования и параметры MOBB детали.
3. Функциональный блок идентификации детали осуществляет идентификацию детали по методике, описанной в главе 3.5. На вход

блока подаются дескрипторы LSFH и параметры MOBB детали, на выходе формируется уникальный идентификатор 3D-модели детали, который и является результатом идентификации.

Модуль подготовки к идентификации необходим для формирования набора эталонных дескрипторов 3D-моделей деталей, с помощью которых будет осуществляться идентификация. Модуль реализует методику, описанную в главе 3.4. и представлен следующими функциональными блоками:

1. Функциональный блок построения базовых дескрипторов используется для построения базовых дескрипторов по полигональным моделям эталонных деталей. На вход блока по очереди подаются полигональные модели деталей. На выходе блока формируются набор базовых дескрипторов и параметры MOBB.
2. Функциональный блок формирования эталонных дескрипторов осуществляет формирование эталонных дескрипторов по набору базовых дескрипторов. На вход блока по очереди подаются наборы базовых дескрипторов и параметров MOBB для эталонных деталей, на выходе блока формируется наборы эталонных дескрипторов и параметров MOBB.
3. Блок хранилища эталонных дескрипторов используется для хранения и повторного использования эталонных дескрипторов.

Для ИИС идентификации деталей разработано ПО, подробно рассмотренное в главах 4.1. и 4.2. Последовательность использования компонентов ПО для реализации ИИС идентификации деталей, входные и выходные данные изображены на рисунке 52.

Этап подготовки, реализующий модуль подготовки к идентификации, выполняется один раз для используемого набора деталей. При необходимости добавления новых деталей достаточно выполнить этап подготовки только для этих деталей.

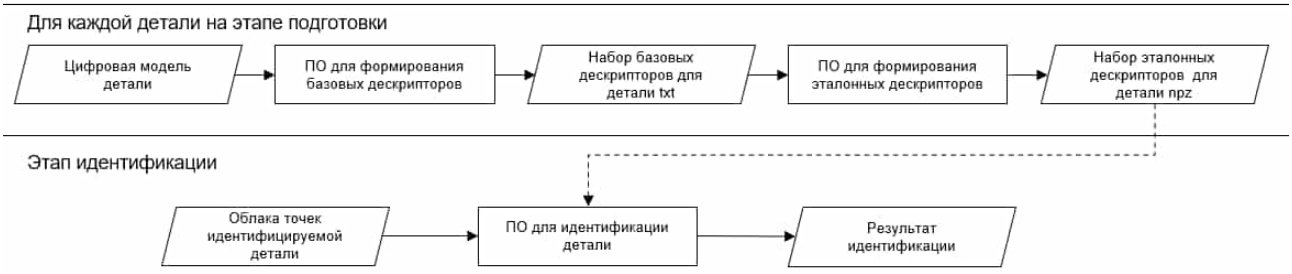


Рисунок 52 — Последовательность использования программного обеспечения для проведения идентификации детали

На первом шаге необходимо сформировать набор базовых дескрипторов с помощью ПО для формирования базовых дескрипторов (реализует блок построения базовых дескрипторов). Из командной строки осуществить запуск приложения, в аргументах командной строки указать путь в файловой системе до полигональной модели детали, представленной в формате STL, путь в файловой системе куда необходимо сохранить файл с набором базовых дескрипторов и указать размер вокселя, который используется в процессе идентификации. Во время работы ПО на экран будут выводиться сообщения о процессе генерирования базовых дескрипторов. По завершении выполнения операции в консоль будет выведено сообщение «Успешно», а по указанному пути сохранен файл в текстовом формате, содержащий набор базовых дескрипторов для заданной модели детали.

Стоит учитывать, что формирование базовых дескрипторов является затратным по времени процессом. Время расчета зависит от количества полигонов модели детали.

На втором шаге из полученных базовых дескрипторов необходимо сформировать эталонные. Этот шаг реализует функциональный блок формирования эталонных дескрипторов рассмотренной ИИС. Для этого используется ПО для формирования эталонных дескрипторов. При вызове программы в аргументах командной строки указываются путь в файловой системе до файла, содержащего базовые дескрипторы для заданной детали и путь куда необходимо сохранить архив с эталонными дескрипторами, в качестве

названия файла необходимо указать название детали. Предполагается, что все архивы с эталонными дескрипторами будут сохранены в одну директорию. В результате работы ПО будет сформирован NPZ архив, содержащий необходимую для последующей идентификации детали информацию: эталонные дескрипторы LSFH и параметры MOBB. По завершении выполнения программы в консоль будет выведена информация о количестве эталонных дескрипторов и рассчитанном параметре схожести групп. После выполнения описанного этапа файл, содержащий базовые дескрипторы можно удалить.

В описанной реализации директория, содержащая файлы с наборами эталонных дескрипторов и параметрами MOBB, является базой эталонных дескрипторов.

В результате этапа подготовки сформированы наборы необходимой для идентификации информации. На этапе идентификации осуществляется непосредственно процесс идентификации деталей по облакам точек. Для этого необходимо запустить программу для идентификации деталей, указав в аргументах командной строки директории расположения наборов эталонных дескрипторов, размер вокселя, и параметры фильтрации, включающие в себя средний шаг конвейера и среднее расстояние между точками на ровной поверхности на срезе лазера. После запуска программа загрузит все эталонные дескрипторы из архивов в указанной директории в память.

Программа для идентификации деталей полностью реализует модуль идентификации деталей из рассмотренной ИИС.

Программа для идентификации детали ожидает передачи облака точек, полученного с устройств сканирования, в стандартный поток ввода. Предполагается использование программы для идентификации в качестве подпрограммы с помощью механизма PIPE операционной системы LINUX или перенаправления потоков ввода-вывода другими способами.

В поток ввода ПО для идентификации необходимо передавать облака точек в следующем формате: сначала указывается число ракурсов и количество точек в облаках для каждого из ракурсов, направляющие вектора плоскости лазера, после чего построчно передаются координаты точек облаков и индексы среза для точки, после чего построчно передаются индексы срезов и координаты положения камеры и лазера над срезом с указанным индексом (параметры необходимые для проведение фильтрации, описанной в главе 3.3.).

Для полученного из стандартного потока ввода облака точек программа выполнит процедуру подготовки облака точек, расчет дескрипторов LSFH и идентификацию по рассчитанным дескрипторам. В поток вывода программа направит результат идентификации в виде текстового идентификатора детали. В качестве текстового идентификатора используется имя архива, содержащего эталонные дескрипторы.

4.4. Выводы по главе

В главе описано применение разработанных методик, в ходе которого:

— разработано и описано программное обеспечение для эмуляции лазерного сканирования и способы его применения;
— разработано и описано программное обеспечение для проведения процесса идентификации деталей по облакам точек;
— разработана и описана информационно-измерительная система идентификации деталей на конвейерной линии по облаку точек на основе разработанных в прошлых главах методик;
— описан алгоритм применения разработанного ПО для ИИС.

ЗАКЛЮЧЕНИЕ

В работе проведена разработка информационно-измерительной системы идентификации детали на конвейере по облаку точек, полученному с устройства лазерного 3D-сканирования. В ходе которой получены следующие результаты:

1. Проведено исследование существующих методов идентификации деталей по облаку точек, позволившее выделить методы, использующие глобальные дескрипторы, как подходящие для решения задачи идентификации детали по облаку точек, полученному при движении детали по конвейеру. Проведено исследование методик подготовки облаков точек к построению глобального дескриптора, в результате которого выделены методики, применимые для решаемой задачи.
2. Разработана методика эмуляции лазерного 3D-сканирования, отличающаяся от известных способом построения облака точек, а именно формированием изображения с камеры устройства лазерного сканирования, содержащего только видимое пятно лазера на поверхности детали, с последующим выделением центра лазерного луча и восстановлением положения его центров в пространстве. В ходе исследования соответствия облака точек, полученного с помощью разработанной методики эмуляции, облаку точек, полученному при реальном сканировании, отличие облаков составило не более 10% и 6% в среднем при использовании вокселя 0,7 мм.
3. Разработана методика построения глобального дескриптора, названного Laser Scanner Feature Histogram, на базе дескрипторов CVFH, VFH, предназначенная для описания деталей сложной геометрии, отличающаяся методом определения средней точки по параметрам MOBB и адаптированной компоненте LSSDC, позволившая повысить точность идентификации до 5 раз по сравнению с дескриптором CFVH на тестовых деталях.

4. Разработана методика подготовки облака точек, включающая в себя этапы статистической фильтрации, фильтрации нормалей и реорганизации облака точек для лучшего соответствии поверхности, позволившая уменьшить расстояние до ближайшего эталонного дескриптора на 22% и повысить точность расчета параметров MOBB на 43% в среднем.
5. Разработана методика формирования эталонных дескрипторов с учетом свойств дескриптора LSFH и подходов к обработке изображений, которая позволяет динамически выбирать значимые дескрипторы на основе сложности геометрии детали, что позволило уменьшить число дескрипторов с 65160 до 447 в среднем на деталь, за счет чего более чем в 40 раз уменьшилось время на идентификацию детали по сравнению с идентификацией с помощью дескриптора VFH и в более чем 600 раз по сравнению с идентификацией с помощью дескриптора CVFH.
6. Разработана методика идентификации деталей с помощью голосования, позволяющая повысить точность идентификации до 0,99 на реальных данных путем одновременного использования дескрипторов, полученных с разных ракурсов, и информации о геометрических параметрах детали.
7. Разработано программное обеспечение для эмуляции лазерного сканирования и для идентификации деталей по облаку точек, реализующее разработанные методики.
8. Разработана информационно-измерительная система идентификации деталей на конвейерной линии по облаку точек, полученному с помощью лазерного сканирования.

Таким образом цель работы, а именно повышение точности идентификации деталей, а также уменьшение времени идентификации в составе ИИС достигнуто.

СПИСОК ЛИТЕРАТУРЫ

1. Александров А.А. Идентификация деталей по облакам точек, полученных с системы лазерного сканирования, с помощью глобального дескриптора LSFH // Мягкие измерения и вычисления. — 2026. № 4. Т. 101. С. 7-16.

2. Александров А.А., Максимов А.Н. Система лазерного 3D сканирования изделий, движущихся по ленточному конвейеру // Мягкие измерения и вычисления. — 2025. № 3. Т. 88. С. 5–14.

3. Александров А. А., Максимов А. Н. Метод эмуляции получения облака точек объекта по его полигональной модели с учетом параметров сканирующего устройства // Мягкие измерения и вычисления. — 2024. — Т. 82, № 9. — С. 39-50.

4. Александров А.А., Максимов А.Н. Метод сегментации объектов с использованием принципов лазерного 3D сканирования // Мягкие измерения и вычисления. — 2024. — Т. 85, № 12. — С. 59-71.

5. Александров А. А., Максимов А.Н. Метод измерения диаметра изделий с помощью стереозрения при их изготовлении на токарном станке // Мягкие измерения и вычисления. — 2024. — Т. 78, № 5. — С. 5-14.

6. Александров А.А., Павлов О.В. Оценка параметров трехмерного объекта по облаку точек в программах на языках IEC 61499 // Гагаринские чтения – 2025: сборник тезисов докладов Международной молодёжной научной конференции, Москва, 15–18 апреля 2025 года. – Москва: Издательство «Перо», 2025.

7. Максимов А.Н., Александров А.А., Лавриненко С.А. Метод построения киберимунной архитектуры на языках IEC 61499 // Авиация и космонавтика: тезисы 22-й Международной конференции, Москва, 20–24 ноября 2023 года. – Москва: Издательство «Перо», 2023.

8. Максимов А.Н., Александров А.А., Романов В.Д. Метод интеграции алгоритмов обработки изображения в программы на языках IEC 61499 // Авиация

и космонавтика: Тезисы 20-ой Международной конференции, Москва, 22–26 ноября 2021 года. – Москва: Издательство «Перо», 2021.

9. Свидетельство о государственной регистрации программы для ЭВМ № 2026661733 Российская Федерация. Программное обеспечение для эмулирования процесса лазерного 3D сканирования на ленточном конвейере : заявл. 30.03.2026: опубл. 22.04.2026 / А.А. Александров.

10. Патент на полезную модель № 230531 U1 Российская Федерация, МПК G01B 11/08, G01B 11/10, G01B 11/22. Автоматизированное устройство технического контроля геометрических параметров тел вращения : № 2024104364 : заявл. 19.02.2024 : опубл. 09.12.2024 / Ф.В. Васильев, М.А. Коробков, О.И. Овсянников, А.А. Александров и др.; заявитель АКЦИОНЕРНОЕ ОБЩЕСТВО "СЕРОВСКИЙ МЕХАНИЧЕСКИЙ ЗАВОД".

11. Han X. F. et al. 3D point cloud descriptors: state-of-the-art //Artificial Intelligence Review. – 2023. – Т. 56. – №. 10.

12. Zhang H. et al. Deep learning-based 3D point cloud classification: A systematic survey and outlook //Displays. — 2023. — Т. 79. — С. 102456.

13. Liu W. et al. Deep learning on point clouds and its application: A survey //Sensors. — 2019. — Т. 19. — №. 19. — С. 4188.

14. Guo Y. et al. Deep learning for 3d point clouds: A survey //IEEE transactions on pattern analysis and machine intelligence. — 2020. — Т. 43. — №. 12. — С. 4338-4364.

15. Liu Y. et al. Point cloud-based deep learning in industrial production: A survey //ACM Computing Surveys. — 2025. — Т. 57. — №. 7. — С. 1-36.

16. Guo Y. et al. 3D object recognition in cluttered scenes with local surface features: A survey //IEEE transactions on pattern analysis and machine intelligence. — 2014. — Т. 36. — №. 11. — С. 2270-2287.

17. Himri K., Ridao P., Gracias N. 3D object recognition based on point clouds in underwater environment with global descriptors: A survey //Sensors. — 2019. — Т. 19. — №. 20. — С. 4451.

18. Ben-Shabat Y., Lindenbaum M., Fischer A. 3d point cloud classification and segmentation using 3d modified fisher vector representation for convolutional neural networks //arXiv preprint arXiv:1711.08241. — 2017.

19. Xiang T. et al. Walk in the cloud: Learning curves for point clouds shape analysis //Proceedings of the IEEE/CVF international conference on computer vision. — 2021. — С. 915-924.

20. Sánchez J. et al. Image classification with the fisher vector: Theory and practice //International journal of computer vision. — 2013. — Т. 105. — №. 3. — С. 222-245.

21. Maturana D., Scherer S. Voxnet: A 3d convolutional neural network for real-time object recognition //2015 IEEE/RSJ international conference on intelligent robots and systems (IROS). — Ieee, 2015. — С. 922-928.

22. Lang A. H. et al. Pointpillars: Fast encoders for object detection from point clouds //Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. — 2019. — С. 12697-12705.

23. Yu T., Meng J., Yuan J. Multi-view harmonized bilinear network for 3d object recognition //Proceedings of the IEEE conference on computer vision and pattern recognition. — 2018. — С. 186-194.

24. Yan X. et al. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling //Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. — 2020. — С. 5589-5598.

25. Mohammadi S. S., Wang Y., Del Bue A. Pointview-gcn: 3d shape classification with multi-view point clouds //2021 IEEE International Conference on Image Processing (ICIP). — IEEE, 2021. — С. 3103-3107.

26. Phan A. V. et al. Dgcnn: A convolutional neural network over large-scale labeled graphs //Neural Networks. — 2018. — Т. 108. — С. 533-543.

27. Georgiou T. et al. A survey of traditional and deep learning-based feature descriptors for high dimensional data in computer vision //International Journal of Multimedia Information Retrieval. — 2020. — Т. 9. — №. 3. — С. 135-170.

28. Marton Z. C. et al. Hierarchical object geometric categorization and appearance classification for mobile manipulation //2010 10th IEEE-RAS International Conference on Humanoid Robots. — IEEE, 2010. — С. 365-370.

29. Rusu R. B. et al. Fast 3d recognition and pose using the viewpoint feature histogram //2010 IEEE/RSJ international conference on intelligent robots and systems. — IEEE, 2010. — С. 2155-2162.

30. Aldoma A. et al. CAD-model recognition and 6DOF pose estimation using 3D cues //2011 IEEE international conference on computer vision workshops (ICCV workshops). — IEEE, 2011. — С. 585-592.

31. Aldoma A. et al. OUR-CVFH–oriented, unique and repeatable clustered viewpoint feature histogram for object recognition and 6DOF pose estimation //Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2012. — С. 113-122.

32. Marton Z. C. et al. General 3D modelling of novel objects from a single view //2010 ieee/rsj international conference on intelligent robots and systems. — IEEE, 2010. — С. 3700-3705.

33. Bay H. et al. Speeded-up robust features (SURF) //Computer vision and image understanding. — 2008. — Т. 110. — №. 3. — С. 346-359.

34. Rusu R. B., Blodow N., Beetz M. Fast point feature histograms (FPFH) for 3D registration //2009 IEEE international conference on robotics and automation. — IEEE, 2009. — С. 3212-3217.

35. Wohlkinger W., Vincze M. Ensemble of shape functions for 3D object classification //2011 IEEE international conference on robotics and biomimetics. — IEEE, 2011. — С. 2987-2992.

36. Kasaei S. H. et al. An orthographic descriptor for 3D object learning and recognition //2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — IEEE, 2016. — С. 4158-4163.

37. Ip C. Y. et al. Using shape distributions to compare solid models //Proceedings of the seventh ACM symposium on Solid modeling and applications. — 2002. — С. 273-280.

38. Mokhtarian F., Khalili N., Yuen P. Multi-scale free-form 3D object recognition using 3D models // Image and Vision Computing. — 2001. — T. 19. — №. 5. — C. 271-281.

39. Chen H., Bhanu B. 3D free-form object recognition in range images using local surface patches // Pattern Recognition Letters. — 2007. — T. 28. — №. 10. — C. 1252-1262.

40. Matei B. et al. Rapid object indexing using locality sensitive hashing and joint 3D-signature space estimation // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2006. — T. 28. — №. 7. — C. 1111-1126.

41. Sipiran I., Bustos B. Harris 3D: a robust extension of the Harris operator for interest point detection on 3D meshes // The Visual Computer. — 2011. — T. 27. — C. 963-976.

42. Lo T. W. R., Siebert J. P. Local feature extraction and matching on range images: 2.5 D SIFT // Computer Vision and Image Understanding. — 2009. — T. 113. — №. 12. — C. 1235-1250.

43. Darom T., Keller Y. Scale-invariant features for 3-D mesh models // IEEE Transactions on Image Processing. — 2012. — T. 21. — №. 5. — C. 2758-2769.

44. Mian A., Bennamoun M., Owens R. On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes // International Journal of Computer Vision. — 2010. — T. 89. — C. 348-361.

45. Hou T., Qin H. Efficient computation of scale-space features for deformable shape correspondences //European conference on computer vision. – Berlin, Heidelberg : Springer Berlin Heidelberg, 2010. — C. 384-397.

46. Sun J., Ovsjanikov M., Guibas L. A concise and provably informative multi-scale signature based on heat diffusion // Computer graphics forum. — Oxford, UK : Blackwell Publishing Ltd, 2009. — T. 28. — №. 5. — C. 1383-1392.

47. do Nascimento E. R. et al. On the development of a robust, fast and lightweight keypoint descriptor // Neurocomputing. — 2013. — T. 120. — C. 141-155.

48. do Nascimento E. R. Um descritor robusto e eficiente de pontos de interesse: desenvolvimento e aplicações. — 2012.

49. Zaharescu A. et al. Surface feature detection and description with applications to mesh matching //2009 IEEE conference on computer vision and pattern recognition. – IEEE, 2009. — C. 373-380.

50. Guo Y. et al. Rotational projection statistics for 3D local surface description and object recognition // International journal of computer vision. — 2013. — T. 105. — C. 63-86.

51. Salti S., Tombari F., Di Stefano L. SHOT: Unique signatures of histograms for surface and texture description // Computer vision and image understanding. — 2014. — T. 125. — C. 251-264.

52. Jia L. et al. A local discrete feature histogram for point cloud feature representation //Applied Sciences. — 2025. — T. 15. — №. 5. — C. 2367.

53. Zhao B., Le X., Xi J. A novel SDASS descriptor for fully encoding the information of a 3D local surface //Information Sciences. – 2019. – T. 483. – C. 363-382.

54. Knopp J. et al. Hough transform and 3D SURF for robust three dimensional classification // Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part VI 11. — Springer Berlin Heidelberg, 2010. — C. 589-602.

55. Bay H., Tuytelaars T., Van Gool L. Surf: Speeded up robust features //European conference on computer vision. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2006. — C. 404-417.

56. Guo Y. et al. Performance evaluation of 3D local feature descriptors //Asian Conference on Computer Vision. — Cham : Springer International Publishing, 2014. — C. 178-194.

57. Wu Z. et al. 3d shapenets: A deep representation for volumetric shapes //Proceedings of the IEEE conference on computer vision and pattern recognition. — 2015. — C. 1912-1920.

58. Rusu R. B. et al. Detecting and segmenting objects for mobile manipulation //2009 IEEE 12th international conference on computer vision workshops, ICCV workshops. — IEEE, 2009. — C. 47-54.

59. Schall O., Belyaev A., Seidel H. P. Adaptive feature-preserving non-local denoising of static and time-varying range data // Computer-Aided Design. — 2008. — Т. 40. — №. 6. — С. 701-707.

60. Han X. F. et al. A review of algorithms for filtering the 3D point cloud // Signal Processing: Image Communication. — 2017. — Т. 57. — С. 103-112.

61. Orts-Escolano S. et al. Point cloud data filtering and downsampling using growing neural gas // The 2013 international joint conference on neural networks (IJCNN). — IEEE, 2013. — С. 1-8.

62. Sun Y., Schaefer S., Wang W. Denoising point sets via L0 minimization // Computer Aided Geometric Design. — 2015. — Т. 35. — С. 2-15.

63. Hu G., Peng Q., Forrest A. R. Mean shift denoising of point-sampled surfaces // The Visual Computer. — 2006. — Т. 22. — С. 147-157.

64. Wang J. et al. Consolidation of low-quality point clouds from outdoor scenes // Computer graphics forum. — Oxford, UK : Blackwell Publishing Ltd, 2013. — Т. 32. — №. 5. — С. 207-216.

65. Lipman Y. et al. Parameterization-free projection for geometry reconstruction //ACM Transactions on Graphics (ToG). — 2007. — Т. 26. — №. 3. — С. 22-1 — 22-6.

66. Preiner R. et al. Continuous projection for fast L1 reconstruction //ACM Trans. Graph. — 2014. — Т. 33. — №. 4. — С. 47:1-47:13.

67. Pauly M., Kobbelt L., Gross M. Multiresolution modeling of point-sampled geometry //D-INFK Technical Report. — 2002. — Т. 378.

68. Lozes F., Elmoataz A., Lézoray O. Partial difference operators on weighted graphs for image processing on surfaces and point clouds // IEEE Transactions on Image Processing. — 2014. — Т. 23. — №. 9. — С. 3896-3909.

69. Xiao C. et al. A dynamic balanced flow for filtering point-sampled geometry // The Visual Computer. — 2006. — Т. 22. — С. 210-219.

70. Ta V. T., Elmoataz A., Lézoray O. Nonlocal PDEs-based morphology on weighted graphs for image and data processing //IEEE transactions on Image Processing. — 2010. — Т. 20. — №. 6. — С. 1504-1516.

71. Liu S., Chan K. C., Wang C. C. L. Iterative consolidation of unorganized point clouds // IEEE computer graphics and applications. — 2011. — Т. 32. — №. 3. — С. 70-83.

72. Zaman F., Wong Y. P., Ng B. Y. Density-based denoising of point cloud // 9th International conference on robotic, vision, signal processing and power applications: Empowering research and innovation. — Singapore : Springer Singapore, 2016. — С. 287-295.

73. Digne J. Similarity based filtering of point clouds // 2012 IEEE computer society conference on computer vision and pattern recognition workshops. — IEEE, 2012. — С. 73-79.

74. Huang H. et al. Edge-aware point set resampling //ACM transactions on graphics (TOG). — 2013. — Т. 32. — №. 1. — С. 1-12.

75. Huang H. et al. Consolidation of unorganized point clouds for surface reconstruction //ACM transactions on graphics (TOG). — 2009. — Т. 28. — №. 5. — С. 1-7.

76. Öztireli A. C., Guennebaud G., Gross M. Feature preserving point set surfaces based on non-linear kernel regression //Computer graphics forum. — Oxford, UK : Blackwell Publishing Ltd, 2009. — Т. 28. — №. 2. — С. 493-501.

77. Svedberger J., Andersson J. Laser scanning in manufacturing industries //Master of Science Thesis IIP. — 2013.

78. Dawson-Howe K. M., Vernon D. Simple pinhole camera calibration // International Journal of Imaging Systems and Technology. — 1994. — Т. 5. — №. 1. — С. 1-6.

79. Zhang Z. A flexible new technique for camera calibration //IEEE Transactions on pattern analysis and machine intelligence. — 2000. — Т. 22. — №. 11. — С. 1330-1334.

80. Zhou Q. Y., Park J., Koltun V. Open3D: A modern library for 3D data processing // arXiv preprint arXiv:1801.09847. — 2018.

81. Yuksel C. Sample elimination for generating poisson disk sample sets // Computer Graphics Forum. — 2015. — Т. 34. — №. 2. — С. 25-32.

82. Danhof M. et al. A virtual-reality 3D-laser-scan simulation // Proc. SINCOM'15. — 2015. — С. 68-73.

83. Kot T. et al. Using virtual scanning to find optimal configuration of a 3D scanner turntable for scanning of mechanical parts //Sensors. — 2021. — Т. 21. — №. 16. — С. 5343.

84. Möller T., Trumbore B. Fast, minimum storage ray/triangle intersection // ACM SIGGRAPH 2005 Courses. — 2005.

85. Grans S., Tingelstad L. Blazer: laser scanning simulation using physically based rendering //arXiv preprint arXiv:2104.05430. — 2021.

86. Задорожный А. Г. Модели освещения и алгоритмы затенения в компьютерной графике. — 2020.

87. Blinn J. F. Models of light reflection for computer synthesized pictures //Proceedings of the 4th annual conference on Computer graphics and interactive techniques. — 1977. — С. 192-198.

88. McReynolds T., Blythe D. Advanced graphics programming using OpenGL. — Elsevier, 2005.

89. Christensen J. H. et al. The IEC 61499 function block standard: Software tools and runtime platforms //ISA Automation Week. — 2012. — Т. 2012.

90. Segal A. et al. Generalized-icp //Robotics: science and systems. — 2009. — Т. 2. — №. 4. — С. 435.

91. O'Rourke J. Finding minimal enclosing boxes //International journal of computer & information sciences. — 1985. — Т. 14. — №. 3. — С. 183-199.

92. Hackel T., Wegner J. D., Schindler K. Joint classification and contour extraction of large 3D point clouds //ISPRS Journal of Photogrammetry and Remote Sensing. — 2017. — Т. 130. — С. 231-245.

93. Vincent L., Soille P. Watersheds in digital spaces: an efficient algorithm based on immersion simulations //IEEE Transactions on Pattern Analysis & Machine Intelligence. — 1991. — Т. 13. — №. 06. — С. 583-598.

94. Bradski G. The opencv library //Dr. Dobb's Journal: Software Tools for the Professional Programmer. — 2000. — Т. 25. — №. 11. — С. 120-123.

95. Zhou Q. Y., Park J., Koltun V. Open3D: A modern library for 3D data processing // arXiv preprint arXiv:1801.09847. — 2018.

96. Leung R. GPU implementation of a ray-surface intersection algorithm in CUDA (Compute Unified Device Architecture) //arXiv preprint arXiv:2209.02878. — 2022.

ПРИЛОЖЕНИЕ А. Акты о внедрении на предприятии и использовании результатов работы в научном процессе

УТВЕРЖДАЮ
Генеральный директор
ООО «ЦПР РТСофт»
_____ В.Ю. Синенко
«13» мая 2026 г.

АКТ

о внедрении результатов диссертационной работы Александрова Александра Александровича на тему «Информационно-измерительная система идентификации деталей на конвейерной линии на основе лазерного 3D-сканирования» в работы, проводимые в ООО «ЦПР РТСофт»

Мы, нижеподписавшиеся, технический директор Денисов Е.А., начальник отдела перспективных разработок Максимов А.Н., составили настоящий акт о том, что материалы диссертационной работы Александрова Александра Александровича на тему «Информационно-измерительная система идентификации деталей на конвейерной линии на основе лазерного 3D-сканирования», а именно:

- методика идентификации деталей, движущихся по конвейерной линии, по их облакам точек, полученных с устройств лазерного сканирования;
- программная реализация методики идентификации деталей и алгоритмов калибровки устройств сканирования

используются в рабочем процессе отдела перспективных разработок ООО «ЦПР РТСофт» для отработки перспективных технологий контроля геометрии деталей в АСУ ТП, что позволило повысить эффективность алгоритма контроля путем увеличения показателя точности идентификации деталей сложной формы по их облакам точек.

Технический директор ООО «ЦПР РТСофт» _____ Е.А. Денисов

Начальник отдела перспективных разработок ООО «ЦПР РТСофт» _____ А.Н. Максимов

Рисунок А.1 — Акт о внедрении результатов диссертационной работы в ООО «ЦПР РТСофт»

УТВЕРЖДАЮ

Проректор по научной работе федерального государственного автономного образовательного учреждения высшего образования «Московский авиационный институт (национальный исследовательский университет)»,
доктор технических наук

_____ **А.В. Иванов**

_____ 202_ г.

АКТ
об использовании результатов диссертационной работы Александрова Александра Александровича на тему «Информационно-измерительная система идентификации деталей на конвейерной линии на основе лазерного 3D-сканирования», представляемой на соискание ученой степени кандидата технических наук по специальности 2.2.11 «Информационно-измерительные и управляющие системы (технические науки)» в научной деятельности.

Настоящим актом подтверждается, что материалы диссертационной работы Александрова Александра Александровича на тему «Информационно-измерительная система идентификации деталей на конвейерной линии на основе лазерного 3D-сканирования», представляемой на соискание ученой степени кандидата технических наук по специальности 2.2.11, а именно:

– методика эмуляции системы лазерного сканирования, позволяющая получить облака точек объектов по их цифровым моделям;

– программные реализации методики эмуляции системы лазерного сканирования и методики идентификации объекта по облаку точек.

использовались при выполнении государственного задания Минобрнауки России, номер темы FSFF-2023-0005, в прототипах систем навигации БЛА и позволили осуществлять предварительное детектирование и сегментацию объектов по их облакам точек, для дальнейшей идентификации методами искусственного интеллекта.

Директор дирекции института № 3,
доцент, кандидат технических наук _____ Ю.Г. Следков

Заведующий кафедрой № 307, доцент,
кандидат технических наук _____ Ф.В. Васильев

Доцент кафедры № 307, доцент,
кандидат технических наук _____ С.В. Ванцов

Рисунок А.2 — Акт об использовании результатов диссертационной работы в научной деятельности МАИ

ПРИЛОЖЕНИЕ Б. Свидетельства о публикации объектов интеллектуальной собственности

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2026661733

Программное обеспечение для эмулирования процесса лазерного 3D сканирования на ленточном конвейере

Правообладатель: ***Александров Александр Александрович (RU)***

Автор(ы): ***Александров Александр Александрович (RU)***

Заявка № **2026618581**
Дата поступления **30 марта 2026 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***22 апреля 2026 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Ю.С. Зубов

Рисунок Б.1 — Свидетельство о государственной регистрации программы для ЭВМ №2026661733 от 22.04.2026: Александров А.А. Программное обеспечение эмулирования процесса лазерного 3D сканирования на ленточном конвейере

РОССИЙСКАЯ ФЕДЕРАЦИЯ

ПАТЕНТ

НА ПОЛЕЗНУЮ МОДЕЛЬ

№ 230531

Автоматизированное устройство технического контроля геометрических параметров тел вращения

Патентообладатель: ***АКЦИОНЕРНОЕ ОБЩЕСТВО "СЕРОВСКИЙ МЕХАНИЧЕСКИЙ ЗАВОД" (АО "СМЗ") (RU)***

Авторы: ***Васильев Федор Владимирович (RU), Коробков Максим Андреевич (RU), Овсянников Олег Игоревич (RU), Александров Александр Александрович (RU), Ванцов Сергей Васильевич (RU), Шмаков Александр Сергеевич (RU), Аджиумер Роман Маратович (RU), Одинцова Мария Александровна (RU), Нестов Александр Сергеевич (RU), Галимов Роберт Ришатович (RU)***

Заявка № **2024104364**
Приоритет полезной модели **19 февраля 2024 г.**
Дата государственной регистрации в Государственном реестре полезных моделей Российской Федерации **09 декабря 2024 г.**
Срок действия исключительного права на полезную модель истекает **19 февраля 2034 г.**

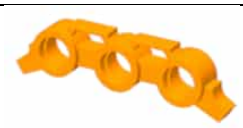
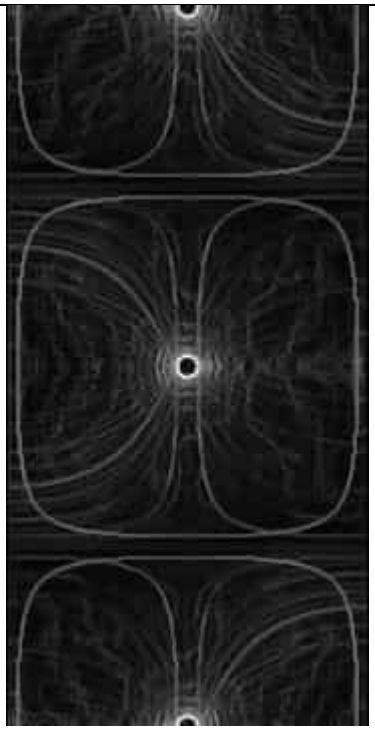
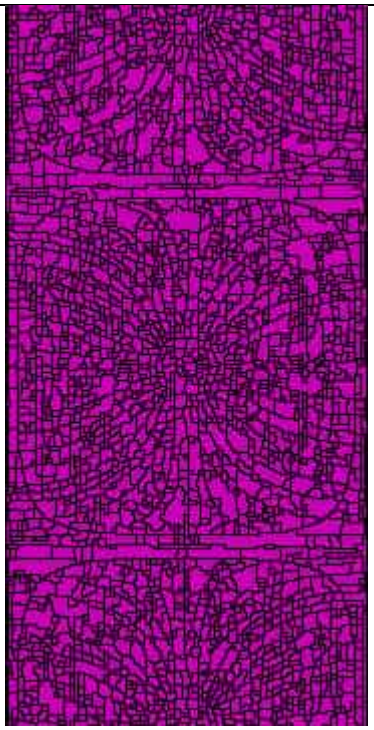
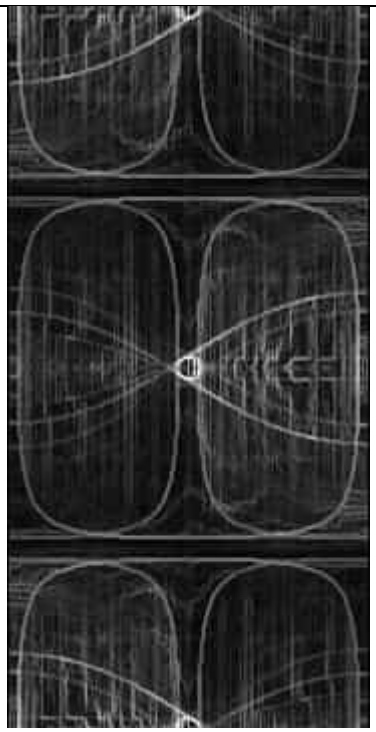
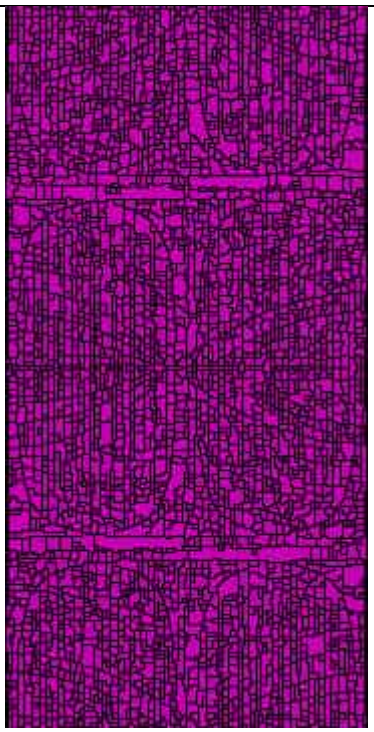
Руководитель Федеральной службы по интеллектуальной собственности

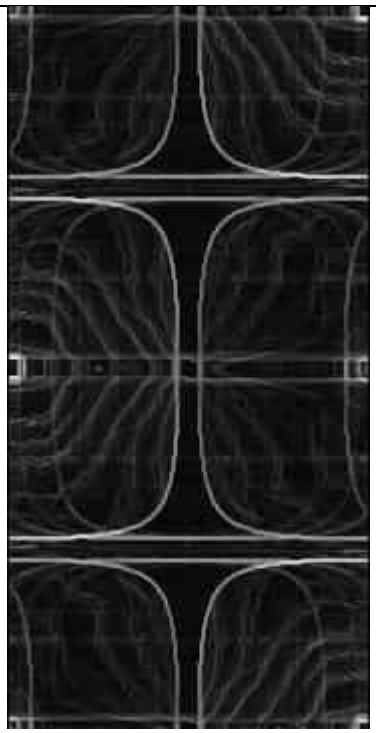
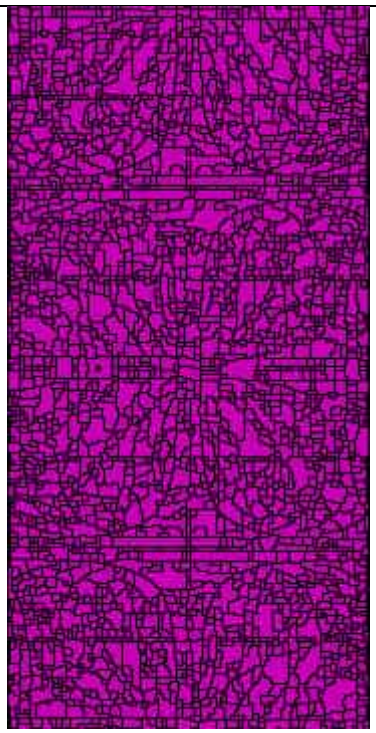
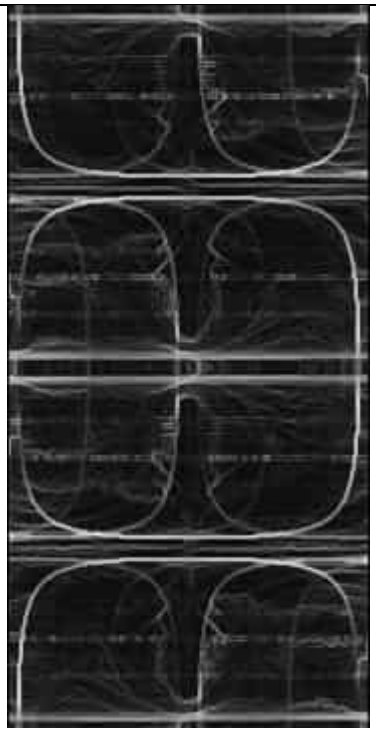
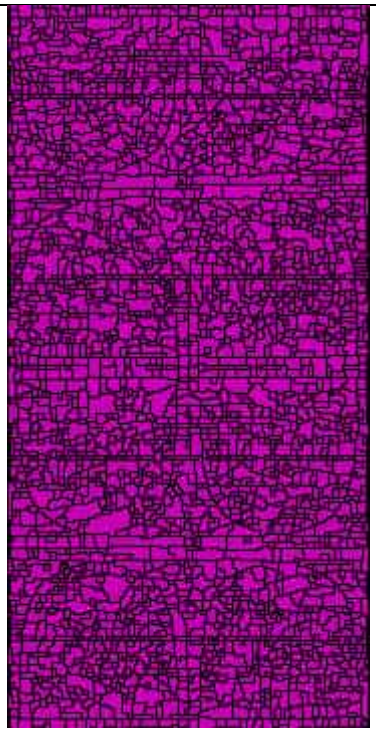
Ю.С. Зубов

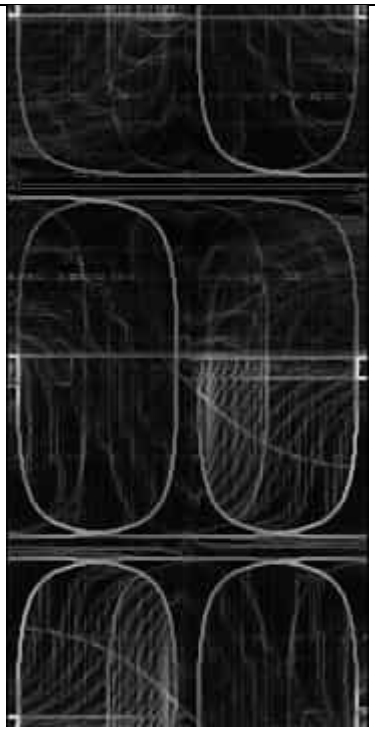
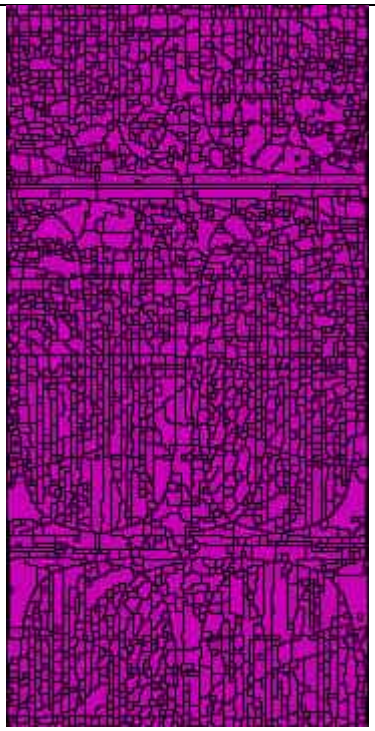
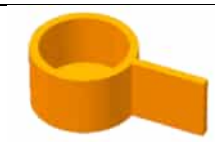
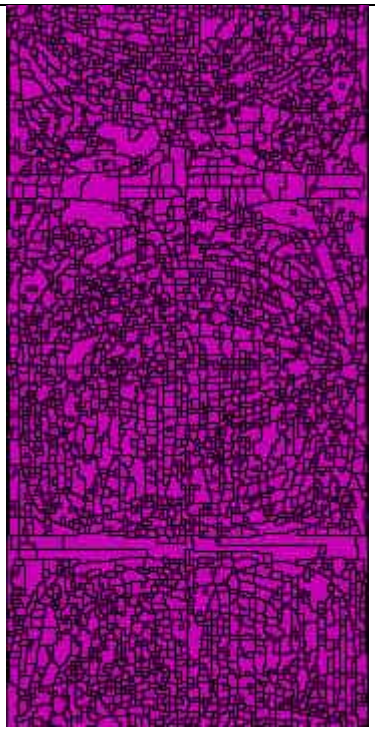
Рисунок Б.2 — Патент на полезную модель № 230531 U1 Российская Федерация: Ф.В. Васильев, М.А. Коробков, О.И. Овсянников, А.А. Александров и др. Автоматизированное устройство технического контроля геометрических параметров тел вращения. Заявитель АКЦИОНЕРНОЕ ОБЩЕСТВО "СЕРОВСКИЙ МЕХАНИЧЕСКИЙ ЗАВОД"

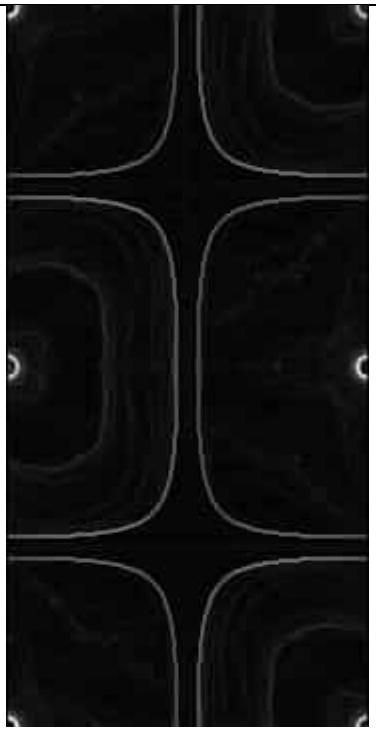
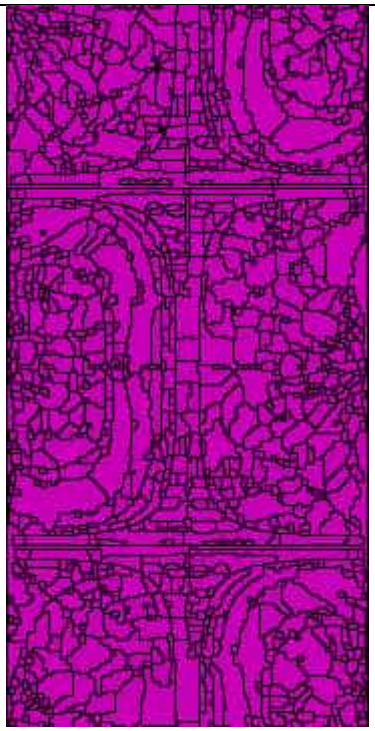
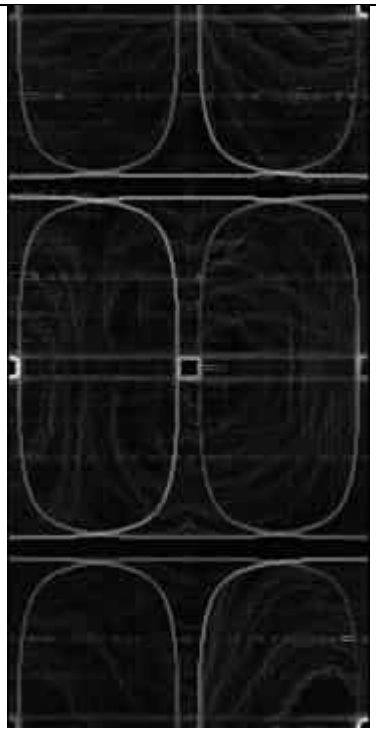
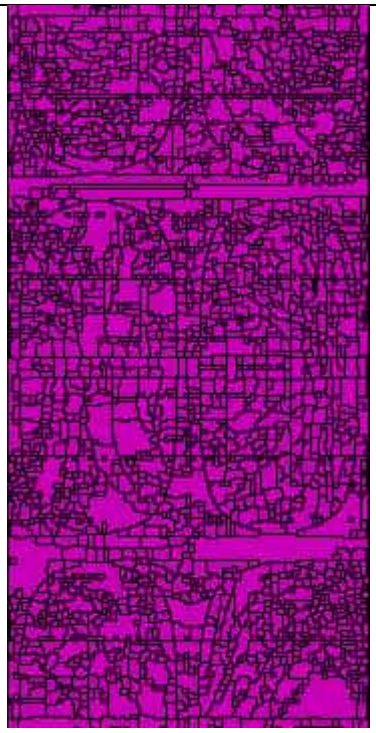
ПРИЛОЖЕНИЕ В. Промежуточные результаты процесса ФЭД

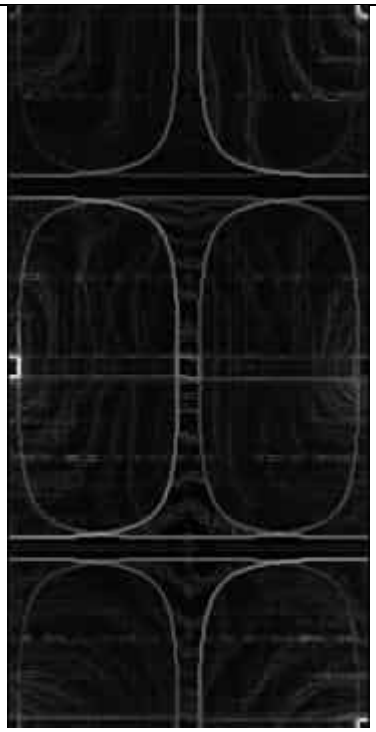
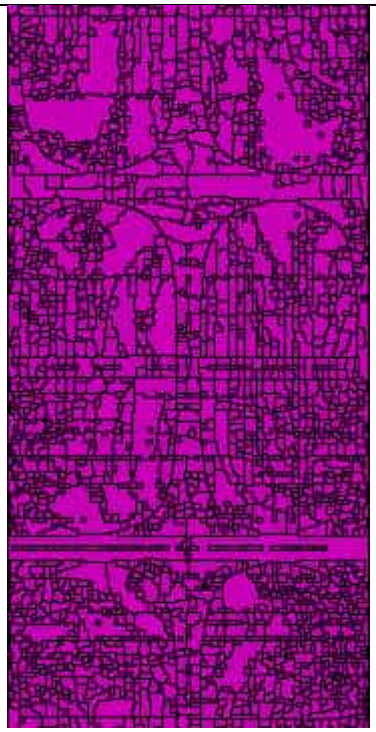
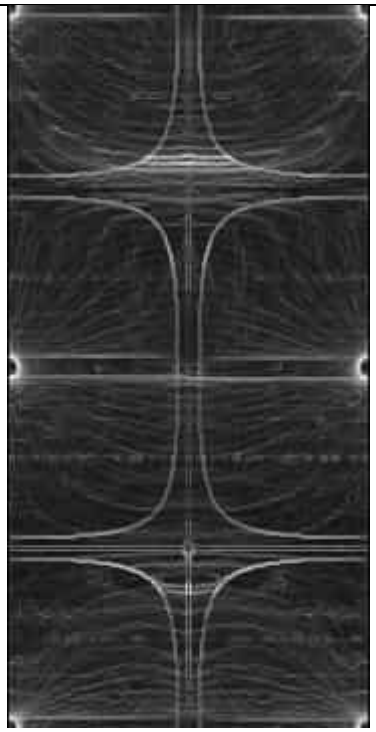
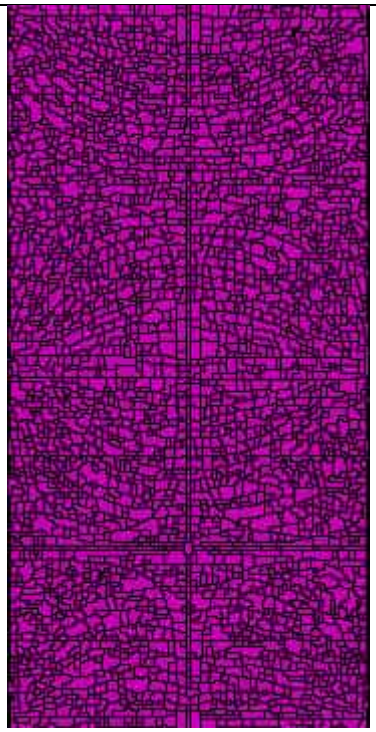
Таблица В.1 — Промежуточные результаты ФЭД тестовых деталей

№ Детали	Вид детали	Карта изменения дескрипторов	Карта выделенных областей
1			
2			

3			
4			

5			
6			

7			
8			

9			
10			

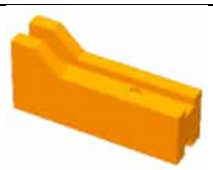
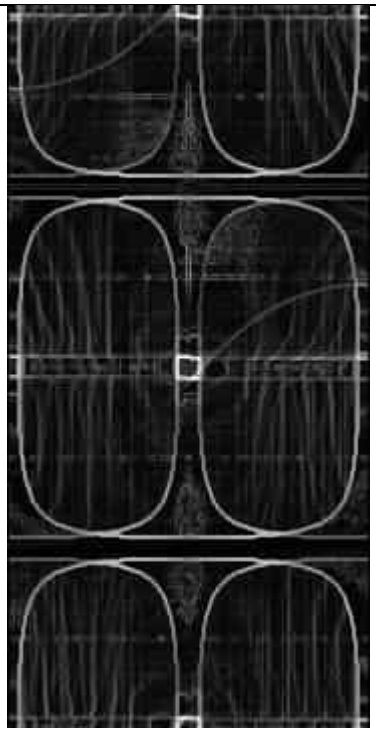
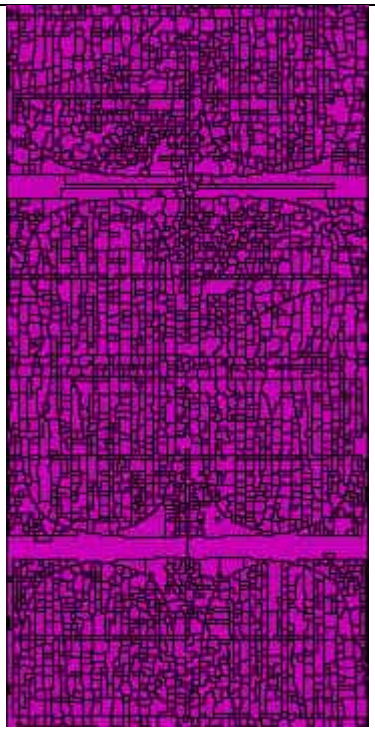
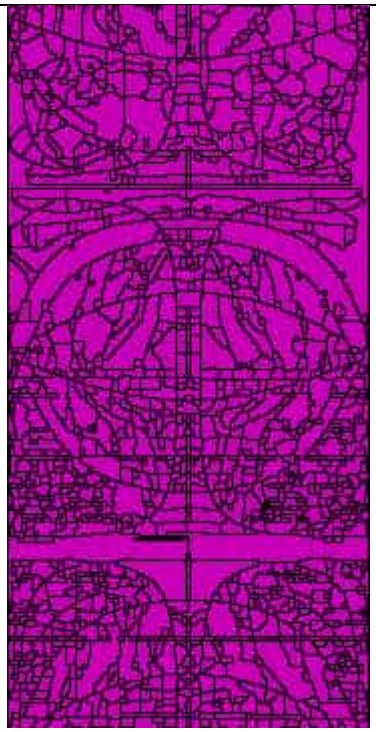
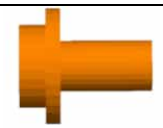
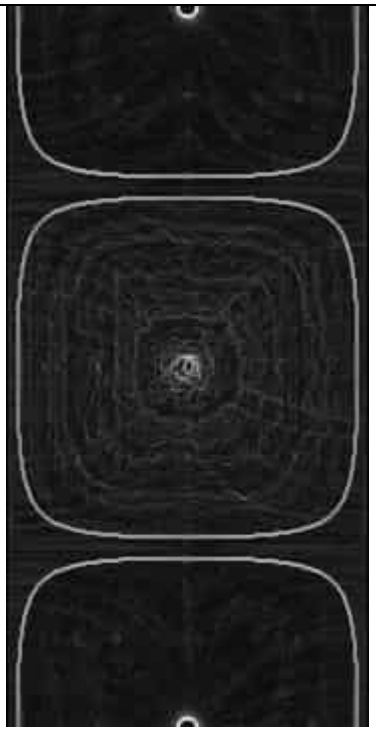
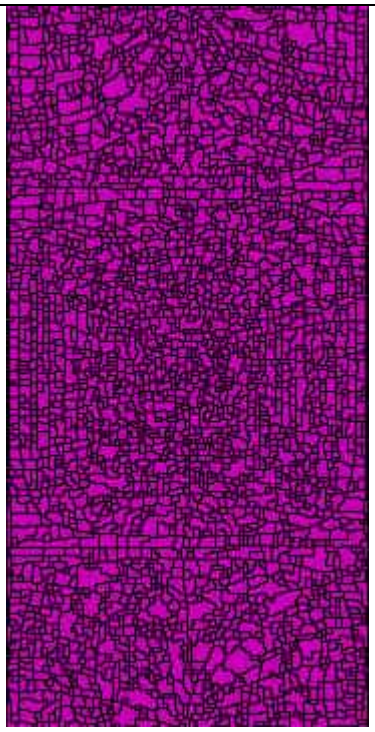
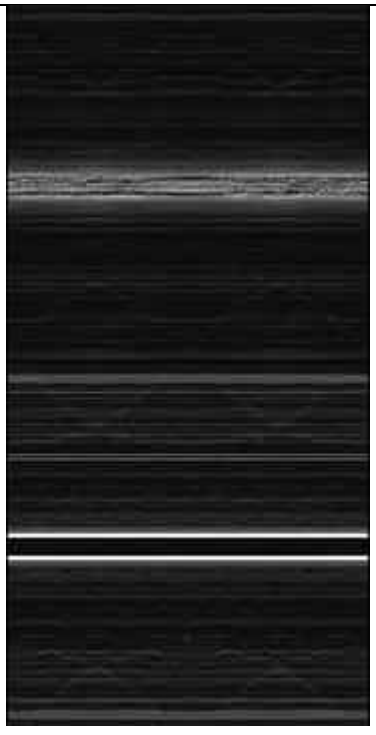
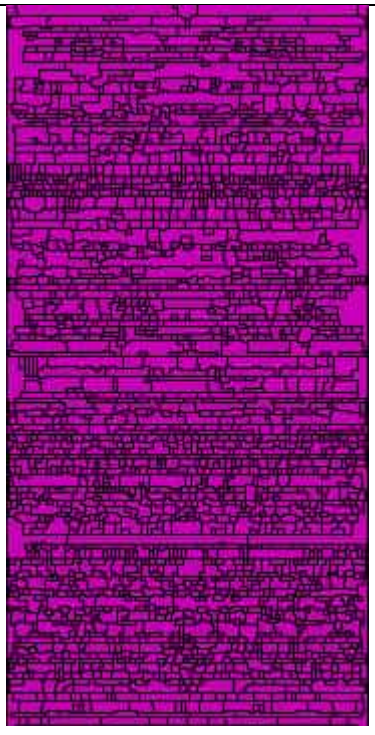
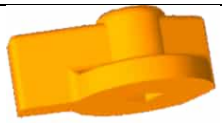
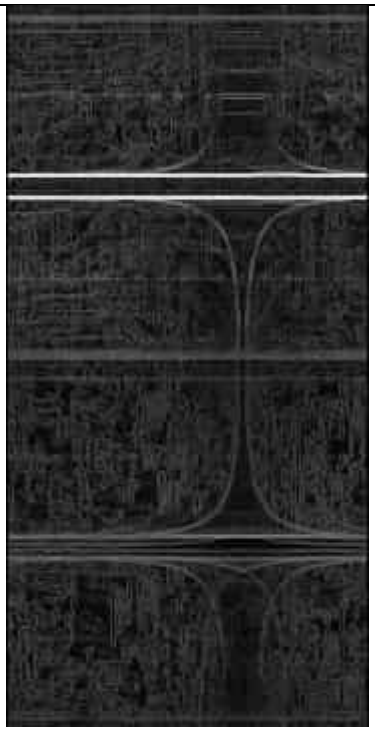
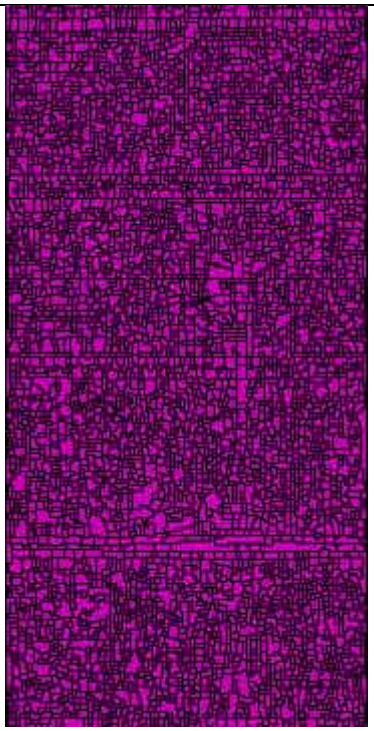
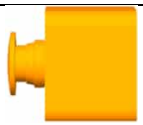
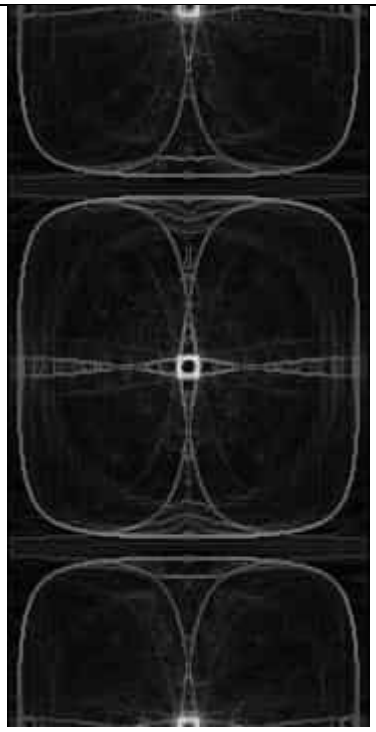
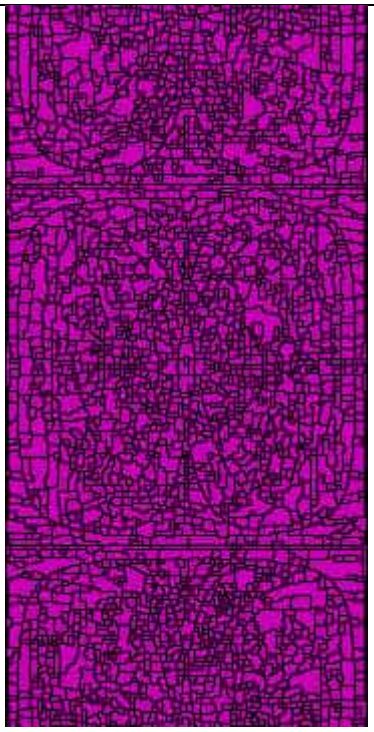
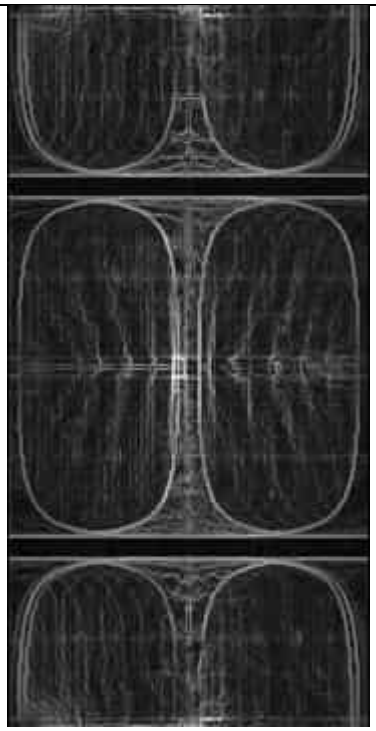
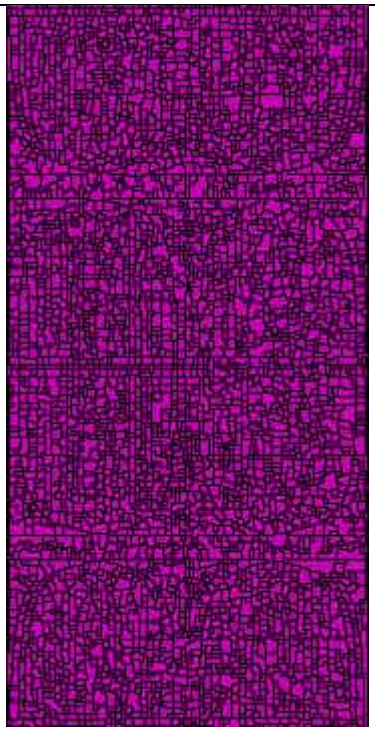
11			
12			

Таблица В.2 — Промежуточные результаты ФЭД произвольных деталей

№ Детали	Вид детали	Карта изменения дескрипторов	Карта выделенных областей
1			
2			

3			
4			

5			

ПРИЛОЖЕНИЕ Г. Программное обеспечение

Структура конфигурации эмуляции лазерного сканирования.

```
typedef struct {
    double x, y, z;
} Vec3d;

// Конфигурация сцены сканирования
class SceneConfigBase {
public:
    struct {
        Vec3d oCam; // Camera pos
        Vec3d vCam; // view vector
        double ettaCam; // rotate over focal axis
        unsigned int wPix; // pix width
        unsigned int hPix; // pix height
        double lPix; // pix size mm
        double fR; // focus distance
        double sThreshold; // sensor threshold
        double noisePercent;
    } cameraConfig;

// Конфигурация системы камера-лазер
    struct {
        double baseline; // cam laser dist
        double thettaLas; // laser/camera angle
        double ettaLas; // laser/ XZ plane
        double twoSigma;
        Vec3d laserColor;
        double specPercent, diffPercent;
    } laserConfig;

// Конфигурация поверхности модели
    struct {
        Vec3d diffuse;
        Vec3d specular;
        double specPwr;
    } materialConfig;

// Конфигурация положения модели
    struct {
        std::string modelPath;
        std::string modelRotSeq;
        double modelUnitMM;
        Vec3d modelMove;
        Vec3d modelRot;
        Vec3d rotCenter;
    } modelConfig;
};

// Конфигурация итераций сканирования (перемещение камеры = -перемещение модели,
для простоты расчетов)
class SceneIterationConfig {
public:
    Vec3d oCamMove;
    double thettaLasDiff;
    unsigned int iterations;
};
```

Основные функции класса эмуляции лазерного сканирования.

```
// Функция запуска эмуляции
void EmulationOptimized::Start() {
    iter_ = 0;
    if (!GetSceneConfigForIteration(sceneConfig0Iter_, iter_, scb_, sic_)) {
        std::cerr << "Error creating config for iteration: " << iter_ << std::endl;
        return;
    }
    sceneConfig0Iter_.mesh->Rotate(addRot_,{
                        scb_.modelConfig.rotCenter.x,
                        scb_.modelConfig.rotCenter.y,
                        scb_.modelConfig.rotCenter.z});

    bbox_ = sceneConfig0Iter_.mesh->GetAxisAlignedBoundingBox();
    std::vector<std::unique_ptr<std::thread>> workers(worker_cnt_);
    for (int i = 0; i < worker_cnt_; ++i) {
        workers[i] = std::make_unique<std::thread>([this]() {
             DoCalc();
        });
    }

    for (int i = 0; i < worker_cnt_; ++i) {
        workers[i]->join();
    }
}

// Функция процесса циклической эмуляции
void EmulationOptimized::DoCalc() {
    while (true) {
        const unsigned int iter = iter_.fetch_add(1);
        {
            std::lock_guard<std::mutex> l(gl);
            std::cout << "[" << iter << "] / " << sic_.iterations << std::endl;
            if (iter >= sic_.iterations) {
                return;
            }
        }

        SceneConfig sc = sceneConfig0Iter_;

// перемещение камеры = -перемещение модели
        sc.config_base.cameraConfig.oCam.x += sic_.oCamMove.x * iter;
        sc.config_base.cameraConfig.oCam.y += sic_.oCamMove.y * iter;
        sc.config_base.cameraConfig.oCam.z += sic_.oCamMove.z * iter;

// Расчет изображения с камеры лазерного сканера
        const auto img = calcOptimized(sc, bbox_);

// Передача изображения в модуль лазерного сканирования
        const auto pts = proc_(sc, img);

        Eigen::Vector3d oCam = {sc.config_base.cameraConfig.oCam.x,
            sc.config_base.cameraConfig.oCam.y,
sc.config_base.cameraConfig.oCam.z};

// Сохранение результатов сканирования
        int i = 0;
        std::lock_guard<std::mutex> l(lock_);
        for (const auto& pt : pts.scan) {
            pts_.push_back(pt);
            view_vectors_.emplace_back(pt - oCam);
            scan_num_.push_back((int)iter);
```

```
            y_indexes_.push_back(pts.yIndex[i++]);
        }
        if (pts.scan.empty()) {
            continue;
        }
        scanv_.push_back((int)iter);
        v1v2v0_.push_back(pts.v1v2v0);
    }
}
// Оптимизированная функция изображения с камеры лазерного сканера
cv::Mat EmulationOptimized::calcOptimized(const SceneConfig &scene_config, const
open3d::geometry::AxisAlignedBoundingBox &bbox) const {
    Eigen::Matrix3Xd m;
// Расчет лучей камеры
    m = generateRays(scene_config);
// Расчет положения лучей в пространстве
    rotateCameraRays(scene_config, m);

    Eigen::Vector3d a;
    Eigen::Vector4d b;
// Расчет положения плоскости лазера
    getLaserPlaneRot(scene_config, b, a);
    const auto bp = bbox.GetBoxPoints();
    std::vector<double> pts(bp.size());
    for (int i = 0; i < bp.size(); ++i) {
        pts[i] = b[0]*bp[i][0] + b[1]*bp[i][1] + b[2]*bp[i][2] + b[3];
    }
// Фильтрация лучей не попадающих в BBOX модели
    double signV = pts[0];
    bool found = false;
    for (double pt : pts) {
        if ((signV >= 0 && pt < 0) || (signV < 0 && pt >= 0)) {
            found = true;
            break;
        }
    }
    if (!found) {
        return cv::Mat::zeros(cv::Size(scene_config.config_base.cameraConfig.wPix,
            scene_config.config_base.cameraConfig.hPix), CV_8UC3);
    }

    std::vector<int> intercectingTriangles;
    std::vector<int> intercectingRaysId;
    std::vector<double> intercectDist;
    std::vector<Eigen::Vector3d> intercetPts;
// Трассировка лучей камеры/лазера
    if (traceCameraLaserRaysOptimized(scene_config, m, b, a, bbox,
        intercectingTriangles, intercectingRaysId, intercectDist, intercetPts) ==
0) {
        return cv::Mat::zeros(cv::Size(scene_config.config_base.cameraConfig.wPix,
            scene_config.config_base.cameraConfig.hPix), CV_8UC3);
        }
// Расчет цветного изображения с камеры
    return          computeLightImage(scene_config,          intercectingTriangles,
intercectingRaysId, intercectDist, intercetPts, a);
}
// Функция расчета цветного изображения с камеры
cv::Mat computeLightImage(const SceneConfig& config,
        const std::vector<int>& intercectingTriangles,
        const std::vector<int>& intercectingRaysId,
        const std::vector<double>& intercectDist,
```

```
        const std::vector<Eigen::Vector3d>& intercetPts,
        const Eigen::Vector3d& laserPos) {

    const auto& scene_config = config.config_base;
    config.mesh->ComputeTriangleNormals(true);
    const double sigm = scene_config.laserConfig.twoSigma / 2;
    Eigen::Vector3d cameraPosition;
    cameraPosition << scene_config.cameraConfig.oCam.x,
        scene_config.cameraConfig.oCam.y,
        scene_config.cameraConfig.oCam.z;

    const auto& triNormals = config.mesh->triangle_normals_;
    cv::Mat img = cv::Mat::zeros(
        cv::Size(scene_config.cameraConfig.wPix,  scene_config.cameraConfig.hPix),
CV_8UC3);

    const auto arrSize = intercetPts.size();
    for (auto i = 0; i < arrSize; ++i) {
        if (intercectingTriangles[i] < 0) {
            continue;
        }

        unsigned int u_, v_;
        getUVByPos(intercectingRaysId[i], config, u_, v_);

// Расчет яркости лазера
        const double lp = getLightningPwr(intercectDist[i], sigm);
// Яркость по Блинну-Фонгу
        const Eigen::Vector3d s = (laserPos - intercetPts[i]).normalized();
        const double d_coef = s.dot(triNormals[intercectingTriangles[i]]);
        const   double   diffuse   =   d_coef   >   0   ?   lp   *   d_coef   *
scene_config.laserConfig.diffPercent: 0;
        const Eigen::Vector3d v = (cameraPosition - intercetPts[i]).normalized();
        const Eigen::Vector3d h = (v + s).normalized();
        const  double  s_coef  =  pow(h.dot(triNormals[intercectingTriangles[i]]),
scene_config.materialConfig.specPwr);
        const   double   specular   =   s_coef   >   0   ?   lp   *   s_coef   *
scene_config.laserConfig.specPercent : 0;

// Добавление случайного шума
        double  noice_coef  =  2.0  *  ((double)(rand()  %  1000)  -  500)  /  1000  *
scene_config.cameraConfig.noisePercent;
        double   r   =   (diffuse   *   scene_config.laserConfig.laserColor.x   *
scene_config.materialConfig.diffuse.x
            +     specular     *     scene_config.laserConfig.laserColor.x     *
scene_config.materialConfig.specular.x) * (1 + noice_coef);
        double   g   =   (diffuse   *   scene_config.laserConfig.laserColor.y   *
scene_config.materialConfig.diffuse.y
            +     specular     *     scene_config.laserConfig.laserColor.y     *
scene_config.materialConfig.specular.y) * (1 + noice_coef);
        double   b   =   (diffuse   *   scene_config.laserConfig.laserColor.z   *
scene_config.materialConfig.diffuse.z
            +     specular     *     scene_config.laserConfig.laserColor.z     *
scene_config.materialConfig.specular.z) * (1 + noice_coef);

        if (r < scene_config.cameraConfig.sThreshold) {
            r = 0;
        } else if (r > 1) {
            r = 1;
        }
```

```
        if (g < scene_config.cameraConfig.sThreshold) {
            g = 0;
        } else if (g > 1) {
            g = 1;
        }

        if (b < scene_config.cameraConfig.sThreshold) {
            b = 0;
        } else if (b > 1) {
            b = 1;
        }

// Формирование цветного изображения
        unsigned int x, y;
        getUVByPos(intercectingRaysId[i], config, x, y);
        auto& color = img.at<cv::Vec3b>(y, x);
        color[0] = (uchar)(b * 255);
        color[1] = (uchar)(g * 255);
        color[2] = (uchar)(r * 255);
    }

    return img;
}
```

Основные функции программного обеспечения для построения дескриптора LSFH.

```
// Функция расчета дескриптора LSFH (VHHModern – рабочее название)
VFHModernPayload                                        CalculateVFHModern(const
std::shared_ptr<open3d::geometry::PointCloud> &pcl,
                                const Eigen::Vector3d &normal, const BBoxAndCenter
&bbc) {

    VFHModernPayload mpl {};
    for (int i = 0; i < VFHM_DESCRIPTOR_LEN; ++i) {
        mpl.descriptor[i] = 0;
    }
    mpl.extent = bbc.extent;

    double extentDist = bbc.extent.norm() / 2;

    const auto &points = pcl->points_;
    const auto &normals = pcl->normals_;
    const double dPi = (double)1.0 / (2.0 * static_cast<double> (M_PI));

    Eigen::Vector3d center = bbc.center;
    const Eigen::Vector3d& avgNorm = normal;

    for (int ptc = 0; ptc < points.size(); ++ptc) {
        Eigen::Vector4d pfhM;
// Расчет компоненты FPFH
        if (!computePairFeatures(center, avgNorm, points[ptc], normals[ptc],
                                 pfhM[0], pfhM[1], pfhM[2], pfhM[3])) {
            continue;
       }

        int pos = 0;
        for (int hi = 0; hi < 3; ++hi) {
            int raw_index;
            if (hi > 0) {
                raw_index = static_cast<int> (std::floor (45 * ((pfhM[hi] + 1) /
2)));
```

```
            } else {
                raw_index = static_cast<int> (std::floor (45 * ((pfhM[hi] + M_PI)
* dPi)));
            }
            const int h_index = std::max<>(std::min<>(raw_index, 45 - 1), 0);
            mpl.descriptor[pos + h_index] += 1;
            pos += 45;
        }
        int h_index;
        h_index = static_cast<int> (std::floor (45 * (pfhM[3] / extentDist)));
        h_index = std::max<> (std::min<> (h_index, 45 - 1), 0);
        mpl.descriptor[pos + h_index] += 1;
    }

    return mpl;
}

// Функции расчета параметров MOBB для полигональной модели
BBoxAndCenter                                  GetBBoxAndCenterForTriangleMesh(const
std::shared_ptr<open3d::geometry::TriangleMesh> &mesh) {
    BBoxAndCenter bcntr;

    open3d::geometry::TriangleMesh copy;
    copy.vertices_ = mesh->vertices_;
    copy.triangles_ = mesh->triangles_;

    // robust make noice in points coords, so use a copy
    auto mobbox = copy.GetMinimalOrientedBoundingBox(true);
    bcntr.center = mobbox.GetCenter();

    std::array<double,      3>      sarr{mobbox.extent_[0],      mobbox.extent_[1],
mobbox.extent_[2]};
    std::sort(sarr.begin(), sarr.end());

    bcntr.extent = {sarr[0], sarr[1], sarr[2]};

    bcntr.obb = std::make_shared<open3d::geometry::OrientedBoundingBox>();
    bcntr.obb->extent_ = mobbox.extent_;
    bcntr.obb->center_ = mobbox.center_;
    bcntr.obb->color_ = {0, 0, 0};
    bcntr.obb->R_ = mobbox.R_;

    return bcntr;
}

// Функции расчета параметров MOBB для облака точек
BBoxAndCenter                                   GetBBoxAndCenterForPointCloud(const
std::shared_ptr<open3d::geometry::PointCloud> &pcl) {
    BBoxAndCenter bcntr;
    open3d::geometry::PointCloud copy;
    copy.points_ = pcl->points_;

    auto mobbox = copy.GetMinimalOrientedBoundingBox(true);
    bcntr.center = mobbox.GetCenter();

    std::array<double,      3>      sarr{mobbox.extent_[0],      mobbox.extent_[1],
mobbox.extent_[2]};
    std::sort(sarr.begin(), sarr.end());

    bcntr.extent = {sarr[0], sarr[1], sarr[2]};
```

```
    bcntr.obb = std::make_shared<open3d::geometry::OrientedBoundingBox>();
    bcntr.obb->extent_ = mobbox.extent_;
    bcntr.obb->center_ = mobbox.center_;
    bcntr.obb->color_ = {0, 0, 0};
    bcntr.obb->R_ = mobbox.R_;

    return bcntr;
}

// Модифицированная функция расчета компоненты FPFH (эталон из pcl)
bool computePairFeatures (const Eigen::Vector3d &p1, const Eigen::Vector3d &n1,
                          const Eigen::Vector3d &p2, const Eigen::Vector3d &n2,
                          double &f1, double &f2, double &f3, double &f4) {
  Eigen::Vector3d dp2p1 = p2 - p1;
  f4 = dp2p1.norm();

  if (f4 == 0.0){
    std::cerr << "computePairFeatures Euclidean distance between points is 0!\n";
    f1 = f2 = f3 = f4 = 0.0;
    return false;
  }

  Eigen::Vector3d n1_copy = n1,
                  n2_copy = n2;
  double angle1 = n1_copy.dot (dp2p1) / f4;

  // Make sure the same point is selected as 1 and 2 for each pair
  double angle2 = n2_copy.dot (dp2p1) / f4;
  if (std::acos (std::fabs (angle1)) > std::acos (std::fabs (angle2))) {
    // switch p1 and p2
    n1_copy = n2;
    n2_copy = n1;
    dp2p1 *= (-1);
    f3 = -angle2;
  }
  else {
    f3 = angle1;
  }
  // Create a Darboux frame coordinate system u-v-w
  // u = n1; v = (p_idx - q_idx) x u / || (p_idx - q_idx) x u ||; w = u x v
  Eigen::Vector3d v = dp2p1.cross(n1_copy);
  double v_norm = v.norm ();
  if (v_norm == 0.0) {
    std::cerr << "computePairFeatures Norm of Delta x U is 0!\n";
    f1 = f2 = f3 = f4 = 0.0;
    return false;
  }
  // Normalize v
  v /= v_norm;
  Eigen::Vector3d w = n1_copy.cross(v);
  f2 = v.dot (n2_copy);
  f1 = std::atan2 (w.dot (n2_copy), n1_copy.dot (n2_copy)); // @todo optimize this

  return true;
}
```

Основные функции программы расчета базовых дескрипторов:

```
// Расчет лучей и генерирование базового дескриптора
VFHModernPayload                                        genVFHForModel(const
std::shared_ptr<open3d::geometry::TriangleMesh> &mesh) {
    auto finDescBBoxes = GetBBoxAndCenterForTriangleMesh(mesh);
```

```
    const auto bbox = mesh->GetAxisAlignedBoundingBox();
    const auto points = bbox.GetBoxPoints();
    Eigen::Vector3d startPoint = points[2] + Eigen::Vector3d{0, 10, 0};
    Eigen::Vector3d yzVec = (points[5] - points[2]).normalized() * rayGSizeMM;
    int rayCountYZ = (int) ((points[2] - points[5]).norm() / rayGSizeMM + 1);
    Eigen::Vector3d yxVec = (points[7] - points[2]).normalized() * rayGSizeMM;
    int rayCountYX = (int) ((points[7] - points[2]).norm() / rayGSizeMM + 1);

    Eigen::Vector3d dstVec = points[0] - points[2] - Eigen::Vector3d{0, 10, 0};
    std::vector<Eigen::Vector3d> raysTo(rayCountYX * rayCountYZ);
    std::vector<Eigen::Vector3d> raysFrom(rayCountYX * rayCountYZ);
    for (int i = 0; i < rayCountYZ; i++) {
        for (int j = 0; j < rayCountYX; j++) {
            const Eigen::Vector3d ptf = startPoint + ((double) i) * yzVec +
((double) j) * yxVec;
            raysFrom[i * rayCountYX + j] = ptf;
            raysTo[i * rayCountYX + j] = ptf + dstVec;
        }
    }
    std::vector<int> intersectionTri;
    std::vector<Eigen::Vector3d> pts;
    for (int k = 0; k < 3; ++k) {
        if (traceRays(mesh, raysFrom, raysTo, intersectionTri, pts) == 0) {
            std::cerr << mesh->vertices_.size() << "  " << mesh->triangles_.size()
<< "  " <<
                raysFrom.size() << "   " << raysTo.size() << std::endl;
            std::cerr << "Something went wrong with trace rays... Try again..." <<
std::endl;
            continue;
        }
        break;
    }

    mesh->ComputeTriangleNormals(true);
    mesh->NormalizeNormals();

    auto bbx = std::make_shared<open3d::geometry::AxisAlignedBoundingBox>();
    *bbx = bbox;
    bbx->color_ = Eigen::Vector3d{0,0,0};
    // auto coord = open3d::geometry::TriangleMesh::CreateCoordinateFrame(30);
    // open3d::visualization::DrawGeometries({mesh, coord, bbx});

    auto pcl = std::make_shared<open3d::geometry::PointCloud>();
    for (int i = 0; i < intersectionTri.size(); i++) {
        if (intersectionTri[i] < 0) {
            continue;
        }
        const double prod = viewVector.dot(mesh-
>triangle_normals_[intersectionTri[i]]);
        const double angle = std::acos(prod);
        if ((angle - filterDeg) < M_PI / 2) {
            continue;
        }
        pcl->points_.push_back(pts[i]);
        pcl->normals_.push_back(mesh->triangle_normals_[intersectionTri[i]]);
    }

    auto tPcl = pcl->VoxelDownSample(voxelSizeMM);

    open3d::geometry::KDTreeFlann kdtree;
```

```
    kdtree.SetGeometry(*pcl);

    const int ptSize = tPcl->points_.size();

#pragma omp parallel for schedule(static)
    for (int pt = 0; pt < ptSize; ++pt) {
        std::vector<int> indices;
        std::vector<double> distance2;
        kdtree.SearchKNN(tPcl->points_[pt], 1, indices, distance2);
        tPcl->normals_[pt] = pcl->normals_[indices[0]];
    }

    tPcl->NormalizeNormals();
    auto remCurvPcl = RemovePointsWithHighCurvature(tPcl, pRadiusClusterSmoothMM,
curvThresh);
    auto norm = CalculateWeightedYCenterNormal(remCurvPcl);
    return CalculateVFHModern(tPcl, norm, finDescBBoxes);
}

// Точка входа программы расчета базовых дескрипторов
int main(int argc, char *argv[]) {
    if (!parseArgs(argc, argv)) {
        return 1;
    }

    SceneConfigBase scene_config{};
    SceneIterationConfig iter_config{};
    if (!loadEmulationScenario(scanPath, scene_config, iter_config)) {
        std::cout << "Error reading config" << std::endl;
        return 2;
    }

    SceneConfig sc{};
    unsigned int iter = 0;
    if (!GetSceneConfigForIteration(sc, iter, scene_config, iter_config)) {
        std::cout << "Error generating scene config for 0 iteration" << std::endl;
        return 3;
    }

    std::ofstream outfile;
    outfile.open (saveRes);
    if (!outfile.is_open()) {
        std::cerr << "CANNOT OPEN FILE: " << saveRes << std::endl;
        return 1;
    }

    auto center = sc.mesh->GetCenter();
    for (auto &pt: sc.mesh->vertices_) {
        pt -= center;
    }

// Поворот цифровой модели для расчета базовых дескрипторов
    for (int i = 0; i < 181; ++i) {
        for (int j = 0; j < 360; ++j) {
            double xp = (double) (i - 90) / 180. * M_PI;
            double yp = (double) j / 180. * M_PI;

            std::cout << "I: " << i << "  J: " << j << "  xp: "<< xp << "   yp: "
<< yp << std::endl;
```

```
            const                Eigen::Matrix3d                M                =
open3d::geometry::Geometry3D::GetRotationMatrixFromYXZ({0, yp, xp});

            auto model = std::make_shared<open3d::geometry::TriangleMesh>();
            // model->triangles_ = sc.mesh->triangles_;
            // model->vertices_ = sc.mesh->vertices_;
            *model = *sc.mesh;
            const int vSize = model->vertices_.size();
            for (int o = 0; o < vSize; ++o) {
                model->vertices_[o] = M * model->vertices_[o];
            }
            const auto desc = genVFHForModel(model);
// Сохранение базовых дескрипторов в файл
            outfile << i << " " << j << " " << xp << " " << yp;
            outfile << " " << desc.extent[0] << " " << desc.extent[1] << " " <<
desc.extent[2];
            for (int p = 0; p < 180; ++p) {
                outfile << " " << desc.descriptor[p];
            }
            outfile << std::endl;
        }
    }
    outfile.close();
    std::cout << << "Успешно" << std::endl;
}
```

Основные функции программы ФЭД (вырезка кода из этапа расчеты карты дескрипторов и первоначального разделения на группы).

```
mlen = 0
npz_index = 0
with open(scan_res, mode='r') as file:
    while True:
        line = file.readline()
        if not line:
            break
        content.append(line)

xm = 0
ym = 0
for line in content:
    t_arr = line.split(" ", 3)
    xp = int(t_arr[0])
    yp = int(t_arr[1])
    if xm < xp:
        xm = xp
    if ym < yp:
        ym = yp

ym += 1
xm += 1

desc_count = 0
bbox_size = np.zeros((3, 1), dtype=np.float64)

desc_tmp = []
for i in range(ym):
    d = []
    for j in range(xm):
        d.append(np.zeros((180,), dtype=np.float64))
    desc_tmp.append(d)
```

```
for line in content:
    t_arr = line.split(" ")
    xp = int(t_arr[0])
    yp = int(t_arr[1])
    bbox_size[0] = float(t_arr[4])
    bbox_size[1] = float(t_arr[5])
    bbox_size[2] = float(t_arr[6])
    for i in range(180):
        desc_tmp[yp][xp][i] = float(t_arr[7 + i])
    desc_count += 1

x_step = 180 / (xm - 1)
y_step = 360 / ym
print(x_step, y_step, xm, ym)
print("Descriptors count: ", desc_count)
print("bbox size: ", bbox_size)

desc_map_res_sq_sm = np.zeros((ym, xm), dtype=np.float64)
desc_map_res_sq_res = np.zeros((ym, xm), dtype=np.float64)

for i in range(ym):
    for j in range(1, xm - 1):
        def get_cnt_weighted(psy_f, psx_f, psy_t, psx_t, c):
            if psy_t < 0:
                psy_t = ym - 1
            if psy_f < 0:
                psy_f = ym - 1
            if psy_f == ym:
                psy_f = 0
            if psy_t == ym:
                psy_t = 0
            f_vec = desc_tmp[psy_f][psx_f]
            t_vec = desc_tmp[psy_t][psx_t]
            t_val = np.linalg.norm(f_vec - t_vec)
            return c * t_val

        ps = get_cnt_weighted(i - 1, j - 1, i + 1, j + 1, 1) + \
             get_cnt_weighted(i, j - 1, i, j + 1, 2) + \
             get_cnt_weighted(i + 1, j - 1, i - 1, j + 1, 1)

        ps1 = get_cnt_weighted(i - 1, j - 1, i + 1, j + 1, 1) + \
              get_cnt_weighted(i - 1, j, i + 1, j, 2) + \
              get_cnt_weighted(i + 1, j - 1, i - 1, j + 1, 1)

        desc_map_res_sq_sm[i, j] = math.sqrt(
            math.pow(ps, 2) + math.pow(ps1, 2)
        )

mv_sq_sm = np.max(desc_map_res_sq_sm)
img = np.zeros((ym, xm), dtype=np.uint8)
for i in range(ym):
    for j in range(xm):
        img[i, j] = int(desc_map_res_sq_sm[i, j] / mv_sq_sm * 255)
cv2.imwrite(f"res_newdet/{res_name}/map_desc_.png", img)

# Функции доработанного алгоритма водораздела
ws = WatershedSphere(COEF)
markers = ws.apply(img, None)

imgn = utils.create_grid_colored(1, markers, None)
```

```
# old was res_vfhm
cv2.imwrite(f"res_newdet/{res_name}/map_areas_.png", imgn)

vl = np.max(markers)
markers -= 1

arr_markers = []
arr_index = []
for _ in range(vl):
    arr_markers.append([])
    arr_index.append([])

for k in [(0, 0), (0, xm - 1)]:
    ky, kx = k
    index = markers[ky, kx]
    if index < 0:
        continue
    arr_markers[index].append(desc_tmp[ky][kx])
    arr_index[index].append([ky, kx])
    mlen += 1

for i in range(ym):
    for j in range(1, xm - 1):
        index = markers[i, j]
        if index < 0:
            continue
        arr_markers[index].append(desc_tmp[i][j])
        arr_index[index].append([i, j])
        mlen += 1
```

Основные функции программы ФЭД (статистическая обработка).

```
# Функция расчетов параметров в кластере
def process_arr(arr_, ps_, calc_std=False):
    ct_arr = arr_[ps_]
    n = np.zeros((180,), dtype=np.float64)
    cnt = len(ct_arr)
    if cnt == 0:
        return n, 0, 0, 0
    for i in range(cnt):
        n += np.array(ct_arr[i], dtype=np.float64) / cnt

    dst = float("inf")
    pp = 0
    for i in range(cnt):
        dt = np.linalg.norm(n - ct_arr[i])
        if dt < dst:
            dst = dt
            pp = i

    n = ct_arr[pp]

    avg_dist = 0
    if calc_std:
        for i in range(cnt):
            avg_dist += np.linalg.norm(n - ct_arr[i]) / (cnt - 1)

    return n, avg_dist, cnt, pp
```

Вырезка функции статистической обработки карты изменения дескрипторов.

```
arr_s = []
```

```
for i in range(len(arr)):
    ds = process_arr(arr, i, True)
    arr_s.append([*ds, i])

hist_arr = []
for i in range(len(arr_s)):
    if arr_s[i][2] == 0:
        continue
    hist_arr.append(arr_s[i][1])

mean = np.mean(hist_arr)
std = np.std(hist_arr)
thd = mean+std
print("Descriptors:", mtt, ", Groups:", groups)
print("Meandist:", mean, "Stddev:", std, "THD:", thd)

arr_new = []
ind_new = []
for i in range(len(arr_s)):
    orig_index = arr_s[i][4]
    orig_arr = arr[orig_index]
    orig_ind = ind[orig_index]

    if arr_s[i][1] < thd:
        arr_new.append(orig_arr)
        ind_new.append(orig_ind)
        continue

    fit_arr = []
    for k in orig_arr:
        fit_arr.append(k)

    clustering     =     AgglomerativeClustering(None,     distance_threshold=thd,
linkage='complete').fit(fit_arr)
    new_arrs = [[] for _ in range(clustering.n_clusters_)]
    new_index = [[] for _ in range(clustering.n_clusters_)]
    for k in range(len(orig_arr)):
        new_arrs[clustering.labels_[k]].append(orig_arr[k])
        new_index[clustering.labels_[k]].append(orig_ind[k])

    for k in range(len(new_arrs)):
        arr_new.append(new_arrs[k])
        ind_new.append(new_index[k])

arr_s = []
for i in range(len(arr_new)):
    if len(arr_new[i]) == 0:
        continue
    ds = process_arr(arr_new, i
                     # , True
                     )
    arr_s.append([*ds, i])

nw_arr = []
for i in range(len(arr_s)):
    nw_arr.append(arr_s[i][0])

clustering      =      AgglomerativeClustering(None,      distance_threshold=thd,
linkage='complete').fit(nw_arr)
print("FIN CLUSTERS: ", clustering.n_clusters_)
arr_new_fin = [[] for _ in range(clustering.n_clusters_)]
```

```
ind_new_fin = [[] for _ in range(clustering.n_clusters_)]
for i in range(len(arr_s)):
    if arr_s[i][2] == 0:
        continue
    orig_index = arr_s[i][4]
    orig_pp = arr_s[i][3]
    arr_new_fin[clustering.labels_[i]].append(
        arr_new[orig_index][orig_pp]
    )
    ind_new_fin[clustering.labels_[i]].append(
        ind_new[orig_index][orig_pp]
    )

arr_descriptors = []
arr_indexes = []
arr_tree_descriptors = []
arr_tree_indexes = []
#    cent_tree = []
for i in range(len(arr_new_fin)):
    if len(arr_new_fin[i]) == 0:
        continue
    n, avg_dist, cnt, pp = process_arr(arr_new_fin, i)
    arr_descriptors.append(arr_new_fin[i][pp])
    arr_indexes.append(ind_new_fin[i][pp])

    arr_tree_descriptors.append([])
    arr_tree_indexes.append([])
    for e in range(len(arr_new_fin[i])):
        cpt = len(arr_tree_descriptors) - 1
        arr_tree_descriptors[cpt].append(arr_new_fin[i][e])
        arr_tree_indexes[cpt].append(ind_new_fin[i][e])
```