Министерство образования и науки Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и кибербезопасности
Высшая школа программной инженерии

Работа допущена к защите
Директор ВШПИ
_____ П. Д. Дробинцев
«_____» _____ 2024 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
работа бакалавра

СРАВНЕНИЕ МЕТОДОВ СЖАТИЯ АТРИБУТОВ ОБЛАКОВ ТОЧЕК

по направлению подготовки (специальности)
09.03.04 Программная инженерия

Направленность (профиль)
09.03.04_1 «Технология разработки и сопровождения качественного программного продукта»

Выполнил студент гр. 5130904/00104 — Поздняков А. А.

Руководитель старший преподаватель — Фёдоров С. А.

Консультант по нормоконтролю — Локшина Е. Г.

Санкт-Петербург
2024

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ПЕТРА ВЕЛИКОГО

Институт компьютерных наук и технологий

Высшая школа программной инженерии

УТВЕРЖДАЮ

Директор ВШПИ

_____ П.Д. Дробинцев

”18” апреля 2024 г.

ЗАДАНИЕ

на выполнение выпускной квалификационной работы

студенту Позднякову Артемию Анатольевичу, группа 5130904/00104

1. Тема работы: Сравнение методов сжатия атрибутов облаков точек
2. Срок сдачи студентом законченной работы: 20.05.2024
3. Исходные данные по работе:
    - Документация на язык программирования Python;
    - Документация к библиотеке Open3D;
    - Документация к библиотеке NumPy;
4. Содержание работы (перечень подлежащих разработке вопросов):
    - Обоснование актуальности работы
    - Обзор существующих решений
    - Составление требований к разрабатываемому решению
    - Описание реализации
    - Анализ результатов
5. Перечень графического материала (с указанием обязательных чертежей):
6. Консультанты по работе:
7. Дата выдачи задания: 18.04.2024

Руководитель ВКР _____ Фёдоров С. А.

Задание принял к исполнению 18.04.2024

Студент _____ Поздняков А. А.

СПИСОК ИСПОЛЬЗУЕМЫХ СОКРАЩЕНИЙ

PCC (Point Cloud Compression) - сжатие облака точек.

MSE (Mean Squared Error) - среднеквадратичное отклонение.

PSNR (Peak Signal-to-Noise Ration) - отношение пикового сигнала к шуму.

CD (Chamfer Distance) - расстояние Чамфера.

HD (Hausdorff Distance) - метрика Хаусдорфа.

CD-PSNR (Chamfer Distance Peak Signal-to-Noise Ration) - отношение пикового сигнала к шуму для расстояния Чамфера.

MPEG (Moving Picture Experts Group) - группа специалистов, занимающихся стандартизацией в области движущихся изображений.

GPCC (Geometry Point Cloud Compression) - проект MPEG для сжатия облаков точек, представляющих собой статические объекты и сцены, а также динамически полученные объекты и сцены.

TMC13 (Test Model Codec 13) - тестовая модель кодека 13, модель кодека, разрабатываемая в рамках проекта GPCC.

XR (eXtended Reality) - расширенная реальность, набор технологий, включающий в себя альтернативную и виртуальную реальность.

CV (Computer Vision) - компьютерное зрение.

VIF (Visual Image Fidelity) - визуальное качество изображения.

SSIM (Structural Similarity Index Measure) - мера структурной похожести изображений.

BPP (Bits Per Point) - бит на точку.

CSV (Comma Separated Values) - значения, разделённые запятыми. Структурированный формат данных.

CLI (Command Line Interface) - интерфейс командной строки.

CI/CD (Continuous Integration / Continuous Deployment) - непрерывная интеграция и внедрение.

РЕФЕРАТ

На 49 с., 10 рисунков, 4 таблицы.

КЛЮЧЕВЫЕ СЛОВА: PCC, POINT CLOUD COMPRESSION, ОБЛАКО ТОЧЕК, СЖАТИЕ, БЕНЧМАРК

Тема выпускной квалификационной работы: ”Сравнение методов сжатия атрибутов облаков точек”.

Бакалаврская работа посвящена разработке подхода к оценке различных методов сжатия атрибутов облаков точек. Дан обзор существующих систем оценки качества сжатия облаков точек. Определён набор метрик для сравнения методов сжатия облаков точек и их атрибутов. В результате анализа существующих решений, определён набор требований для новой системы оценки методов сжатия облаков точек.

В рамках работы была разработана программа для вычисления качественных метрик сжатия облаков точек с использования языка Python. Предложенное решение было внедрёно в существующую платформу оценки кодеков для сжатия облаков точек PCCArena. С помощью модифицированной системы PCCArena произведён качественный сравнительный анализ кодеков Draco и TMC13.

ABSTRACT

49 pages, 10 figures, 4 tables.

KEYWORDS: PCC, POINT CLOUD COMPRESSION, POINT CLOUD, COMPRESSION, BENCHMARK

The topic of final qualifying work: ”Comparison of point cloud attribute compression methods”.

This Bachelor’s work describes the development of the new approach for comparison of different point cloud attribute compression methods. In this paper, the analysis of existing point cloud compression quality assessment systems carried out. Also, the suite of metrics for point cloud attribute compression is defined. As a result of existing approaches analysis, the suite of requirements for new point cloud compression quality assessment system is defined.

In the course of work, the software for evaluating point cloud compression qualitative metrics developed in Python programming language. The proposed approach was integrated withing PCCArena - point cloud compression benchmark. Using modified version of PCCArena, qualitative comparative analysis of Draco and TMC13 codecs carried out.

СОДЕРЖАНИЕ

Введение 7

Глава 1. Обзор подходов к сравнению методов сжатия облаков точек 10

1.1 Подходы к оценке качества сжатия изображений 10

1.1.1 Математически-обоснованные метрики качества 11

1.1.2 Высокоуровневые метрики качества 12

1.2 Подходы к оценке качества сжатия облаков точек 15

1.3 Системы оценки качества сжатия облаков точек 18

1.3.1 mpeg-pcc-dmetric 18

1.3.2 GeoCNNv1 geo_dist 19

1.3.3 PCCArena 20

1.4 Новая система оценки качества сжатия облаков точек 21

Глава 2. Обоснование выбора технологий и средств разработки 23

2.1 Язык программирования и библиотеки 23

2.2 Непрерывная интеграция 25

Глава 3. Система подсчёта метрик 28

3.1 Алгоритм поиска реконструированной точки 28

3.1.1 Алгоритм сопоставления точек 28

3.1.2 KD-дерево 29

3.2 Выбор метрик для оценки качества сжатия атрибутов 30

3.2.1 Оценка искажения геометрической структуры 30

3.2.2 Оценка искажения нормалей точек 32

3.2.3 Оценка искажения цветов 33

3.2.4 Итоговый перечень метрик 34

3.3 Архитектура разрабатываемого решения 35
Глава 4. Результаты работы 38
4.1 Консольное приложение 38
4.2 Оценка качества сжатия атрибутов 40
4.3 Интеграция с системой PCCArena 43
Заключение 47
Список источников 48

ВВЕДЕНИЕ

Облака точек. Растущая популярность (?ссылка?) технологий компьютерного зрения (CV - Computer Vision) и расширенной реальности (XR - eXtended Reality) влечёт за собой потребность в способах компактного хранения и передачи трёхмерных (далее 3Д) данных. 3Д-данные представляются в виде полигональных сеток - совокупности вершин, рёбер и граней, определяющих форму трёхмерного объекта, или облаков точек, отличающихся от последних отсутствием связей между вершинами. Дополнительно, данные о геометрической структуре объекта могут быть дополнены информацией о его внешних характеристиках.

Важной задачей является создание цифрового представления реальных объектов и сцен. 3Д-сканирование - технология, позволяющая считывать форму физического объекта и его внешних характеристик, таких как цвет или отражающая способность поверхности. В общем случае, результатом процесса 3Д-сканирования является конечное множество точек в трёхмерном пространстве[8, с. 10].

Полигональные сетки аппроксимируют непрерывную поверхность исходного объекта. В свою очередь, облака точек сохраняют мельчайшие подробности структуры поверхности объекта вплоть до миллиметра[8, с. 33]. Задача получения 3Д-модели объекта в виде полигональной сетки по имеющемуся облаку точек, считанных с его поверхности, решается методами реконструкции поверхности[8]. Для обратного преобразования достаточно удалить связи между вершинами в полигональной сетке.

Большая точность позволяет использовать облака точек для машинной обработки в системах компьютерного зрения, взаимодействующих с реальным миром. Примером подобных систем являются беспилотные автомобили, использующие лидары - локаторы, испускающие световые волны оптического

диапазона с дальнейшей регистрацией отраженных импульсов[13, с. 7]. Другим примером являются системы расширенной реальности, в данном случае облака точек используются для совмещения позиций виртуальных объектов и физических объектов, находящихся рядом с человеком[13, с. 15].

Для точного описания поверхности, облака точек должны быть достаточно плотными. Точки в облаке представляют собой дискретные образцы непрерывной поверхности, а полигональные сетки аппроксимируют данную поверхность полигонами[13, с. 4]. Всё это влечёт за собой большой размер облаков точек (может тут тоже ссылка?), в связи с чем возникает задача разработки способов компактного хранения подобных объектов.

Алгоритмы сжатия облаков точек решают задачу компактного хранения и передачи облаков точек. Популярные кодеки были реализованы в рамках проектов Draco[3] и PCL[23]. В настоящее время различными авторами предлагаются новые методы сжатия геометрической структуры облаков точек[15][22] и их атрибутов[7][18][9]. Алгоритм кодирования данных в этих решениях можно разделить на этапы реконструкции изначальной геометрической структуры объекта и кодирования атрибутов облака в соответствии с полученной структурой. Работа по стандартизации PCC-кодеков была начата MPEG в 2017 году[14], также данной группой был предложен собственный кодек и разработана тестовая модель на его основе - TMC13[4].

Большое количество постоянно появляющихся подходов к сжатию облаков точек делает актуальной задачу разработки программы для оценки работы PCC-кодеков. Подобная программа может быть использована исследователями для подсчёта метрик разрабатываемых ими кодеков.

Цель работы - разработка подхода к сравнению алгоритмов сжатия атрибутов облаков точек. В рамках данной работы необходимо решить следующие задачи:

- Проанализировать системы оценки качества сжатия облаков точек
- Изучить релевантные метрики, отображающие эффективность и качество

сжатия атрибутов облаков точек

- Разработать программу подсчёта метрик
- Получить метрики для отобранных PCC-кодеков
- Проанализировать результаты работы

ГЛАВА 1. ОБЗОР ПОДХОДОВ К СРАВНЕНИЮ МЕТОДОВ СЖАТИЯ ОБЛАКОВ ТОЧЕК

В общем случае, информация может быть сжата, если она является избыточной. Сжатие без потерь основано на уменьшения избыточности информации. В сжатии с потерями вводится новое понятие - нерелевантная информация, то есть такая информация, которая слабо влияет на восприятие изображения человеком. Сжатие с потерями, таким образом, основано не только на уменьшении избыточности, но и на определении нерелевантной информации и уменьшении количества подобной информации[1, с. 265].

1.1. Подходы к оценке качества сжатия изображений

Потребность в оценке качества сжатия визуальной информации возникла после изобретения методов сжатия подобной информации. Многие из качественных метрик сжатия 3Д-данных, были изначально разработаны для оценки качества сжатия обычных изображений.

Введем понятие реконструированного изображения. Пусть оригинальное изображение A было закодировано в формате F_1. Закодируем оригинальное изображение с использованием алгоритма сжатия и получим сжатое изображение B, закодированное в формате F_2. Теперь произведем декомпрессию изображения B и закодируем его с помощью формата F_1. Полученное изображение C в формате F_1 - реконструированное изображение.

Метрики качества для медиа могут быть классифицированы по признаку доступности исходного изображения[12]:

- Оригинальное неискаженное изображение доступно
- Доступен лишь набор признаков, извлеченных из оригинального изображения
- Оригинальное изображение недоступно

Другая классификация предполагает разделение метрик по признаку того,

на чём они основаны[17, с. 6]:

- Математически-обоснованные метрики, учитывающие лишь интенсивность искажения. К подобным метрикам относятся среднеквадратичная ошибка (MSE) и отношение пикового сигнала к шуму (PSNR)
- Низкоуровневые метрики, которые учитывают видимость искажений, как, например функции чувствительности контраста (CSF)
- Высокоуровневые метрики, которые основаны на гипотезе о том, что человеческое зрение адаптировано к извлечению структурной информации из изображения. К подобным метрикам относится индекс структурной похожести (SSIM) и визуальная точность изображения (VIF)

1.1.1. Математически-обоснованные метрики качества

Одной из стандартных метрик качества сжатия изображений является отношение пикового сигнала к шуму (Peak Signal to Noise Ratio - далее PSNR). Данная метрика достаточно проста для вычисления, однако она имеет лишь ограниченное, приблизительное отношение к ошибкам, которые воспринимает человеческий глаз. Иными словами, большее значение PSNR означает меньшее расхождение между оригинальным и сжатым изображением, но не гарантирует положительное восприятие реконструированного изображения человеком[1, с. 279].

Обозначим пиксели исходного изображения как P_i, а пиксели реконструированного изображения как Q_i (где $0 \leq i \leq n$). Для начала определим понятие среднеквадратичной ошибки (Mean Squared Error - далее MSE) между двумя изображениями как

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (P_i - Q_i)^2 \tag{1.1}$$

Иными словами, среднеквадратичная ошибка для изображений является

суммой квадратов ошибок для каждого из пикселей, разделённой на общее количество точек. Тогда метрика PSNR может быть определена как

$$\text{PSNR} = 10 \log_{10} \frac{\max_i |P_i|^2}{MSE} \tag{1.2}$$

При этом $\max_i |P_i|$ - пиковое значение сигнала. Для чёрно-белого изображения пиковым значением является 1, для изображения в оттенках серого при глубине 8 бит на пиксель данное значение равно 255. Так как используется логарифм отношения, итоговое значение измеряется в децибелах.

Другой похожей метрикой является отношение сигнала к шуму (Signal to Noise Ratio - далее SNR). В данном случае, рассматривается не пиковое значение сигнала, а среднеквадратичное.

$$\text{SNR} = 10 \log_{10} \frac{\frac{1}{n} \sum_{i=1}^{n} P_i^2}{MSE} \tag{1.3}$$

Значение отношения сигнала к шуму квантования (Signal to Quantization Noise Ratio - далее SQNR) представляет собой меру влияния квантования на качество сигнала. Данную метрику можно определить как отношение мощности сигнала к разнице между сигналом после квантования и оригинальным значением сигнала.

$$\text{SQNR} = 10 \log_{10} \frac{\text{signal power}}{\text{quantization error}} \tag{1.4}$$

1.1.2. Высокоуровневые метрики качества

Математически-обоснованные метрики основаны на предположении, что уменьшение воспринимаемого качества изображения напрямую связано с величиной шума. Так, например, MSE даёт объективную оценку мощности шума, однако два зашумленных изображения с одинаковым значением MSE могут иметь ошибки разного рода, некоторые из которых являются более

заметными чем другие. В случае, когда визуальная информация предназначена прежде всего для восприятия человеком, единственной корректной метрикой визуального качества изображения является субъективная оценка некоторой отобранной группой людей[12]. Данный подход, однако, не является машинным и, следовательно, не подлежит автоматизации. Для решения данной проблемы были разработаны метрики, помогающие автоматически предсказать *воспринимаемое качество (perceived quality)* изображения. К подобным метрикам относится метрики качества изображения SSIM и VIF.

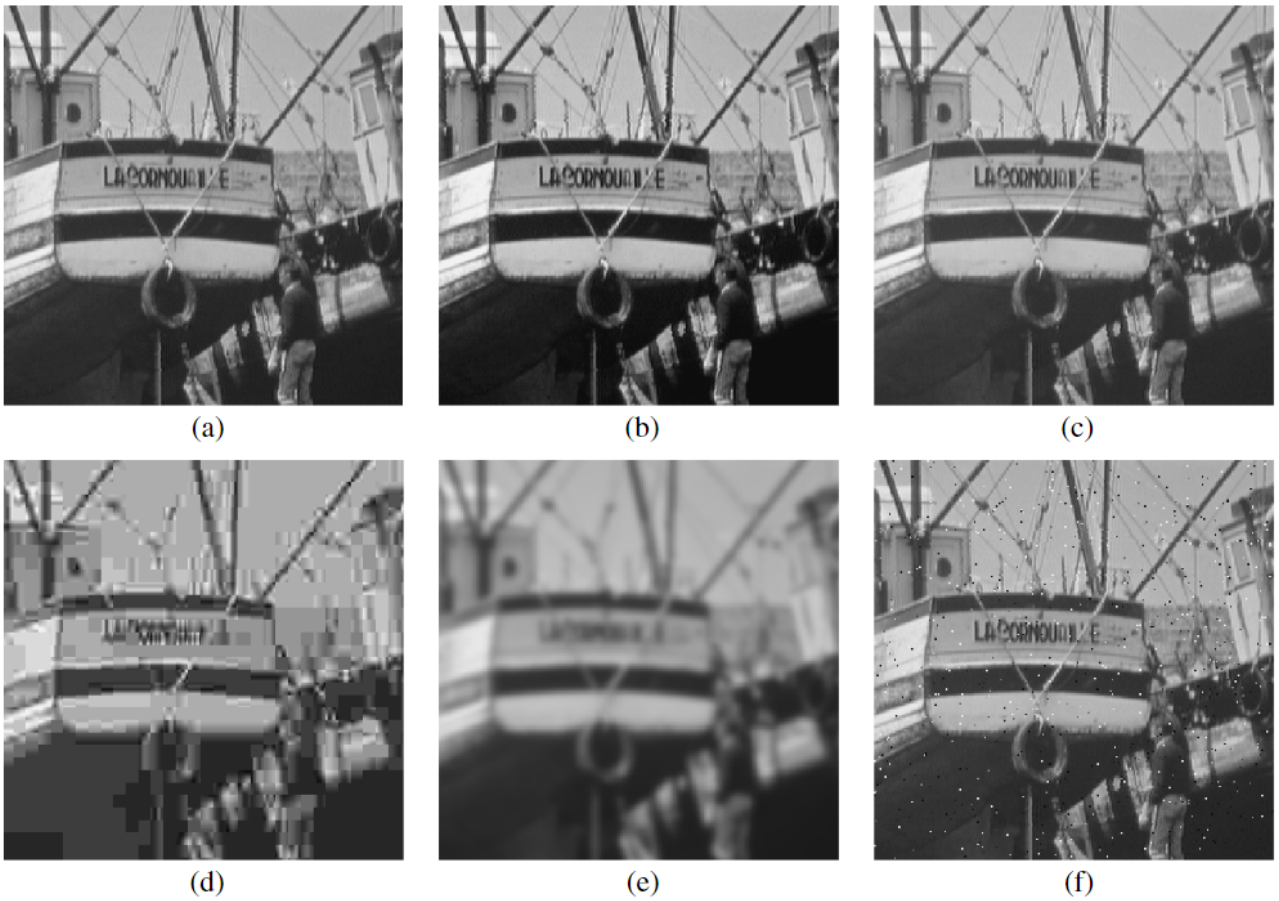


Рис. 1.1: Сравнение изображений, имеющих разный тип ошибок и одинаковое значение MSE=210[12].

Индекс структурной похожести

SSIM основан на вычислении 3 характеристик изображения: яркости, контрастности и структуры. Формулы для вычисления данных характеристик выглядят следующим образом:

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \quad c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \quad s(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3} \tag{1.5}$$

где:

- x и y - сравниваемые изображения
- μ_x и μ_y - средние значения пикселей изображений x и y
- σ_x^2 и σ_y^2 - дисперсии значений пикселей изображений x и y
- σ_{xy} - ковариация между значениями пикселей изображений x и y
- c_1 и c_2 - заданные константы

При этом SSIM является произведением данных значений, взятых с некоторым весом:

$$\text{SSIM}(x,y) = l(x,y)^\alpha + c(x,y)^\beta + s(x,y)^\gamma \tag{1.6}$$

Значения индекса находятся в диапазоне от -1 до 1, при этом большее значение индекса означает большее сходство изображений.

Данный индекс основан на предположении, что человеческое зрение адаптировано именно к извлечению структурной информации[12]. Следовательно, особенно важной компонентой данного индекса являются структура $s(x,y)$. При этом яркость и контрастность по сути также являются характеристиками структурной информации, на данном факте также основано семейство цветовых схем YCC.

Визуальное качество изображения

VIF также основан на оценке структурных особенностей изображения, однако, в отличие от SSIM, данная характеристика учитывает особенности текстуры, края и деталей изображения[11]. Визуальное качество изображение может быть вычислено по следующему алгоритму:

1. Разбиение изображения на блоки. Данный этап включает в себя

разбиение исходного изображения на квадратные блоки пикселей, имеющих некоторый заданный размер.

2. Вычисление локальных статистических характеристик. На этом этапе каждый блок подвергается обработке, в ходе которой вычисляются средние значения пикселей, а также их дисперсия и ковариация.
3. Вычисление глобальных статистических характеристик. При этом локальные статистические характеристики, вычисленные в блоках, суммируются или усредняются.
4. Вычисление взвешенной суммы сходства между локальными и глобальными статистическими характеристиками сравниваемых изображений.

Дополнительно VIF может быть нормализован. Нормализованный VIF принимает значения от 0 до 1, при этом большее значение данного показателя означает большее сходство сравниваемых изображений.

1.2. Подходы к оценке качества сжатия облаков точек

Идея многих подходов к оценке качества сжатия облаков точек взята из более глубоко проработанной области оценки качества изображений и видео. Облака точек, однако представляют собой гораздо более сложный объект, что влияет не только на то, какие стандартные математически-обоснованные качественные метрики можно к ним применить, но и на сам процесс вычисления данных метрик. Рассмотрим терминологию, связанную с наличием потерь при сжатии облаков точек[14]:

- Преобразование геометрической структуры с потерями. Координаты точек в декодированном облаке не обязательно численно совпадают с изначальными координатами. Количество точек в декодированном облаке также может не совпадать с количеством точек в изначальном облаке
- Преобразование геометрической структуры без потерь. Координаты

точек в декодированном облаке численно совпадают с изначальными координатами. Количество точек в декодированном облаке также совпадает с количеством точек в изначальном облаке

- Преобразование атрибутов с потерями. Декодированные сжатые атрибуты не обязательно численно совпадают с изначальными атрибутами
- Преобразование атрибутов без потерь. Декодированные сжатые атрибуты полностью численно совпадают с изначальными атрибутами

Очевидно, что метрики качества релевантны только для методов сжатия геометрической структуры с потерями и методов сжатия атрибутов с потерями. Рассмотрим, как некоторые из вышеупомянутых метрик можно адаптировать для оценки качества сжатия облаков точек.

Искажение геометрической структуры можно оценить, вычислив среднеквадратичную ошибку искажения. Для этого необходимо определить, что собой представляет ошибка для единичной точки. Точку оригинального облака назовём оригинальной точкой, данная точка в ходе сжатия с потерями и последующей декомпрессии определенным образом преобразуется, соответствующую точку в реконструированном облаке назовём реконструированной точкой. Возможно несколько случаев:

1. Координаты оригинальной и реконструированной точки полностью совпадают
2. Координаты оригинальной и реконструированной точки отличаются незначительно
3. Реконструированной точки не существует. Сжатия с потерями геометрической структуры не гарантирует совпадение количества точек в оригинальном и реконструированном облаке, соответственно, исходная точка может быть редуцирована или заменена несколькими другими точками

Для вычисления ошибки выбирается точка реконструированного облака, наиболее близкая к координатам оригинальной точки. В случаях 1 и 2 данной точкой будет реконструированная точка, в случае 3 некоторая другая точка. Ошибка представляет собой расстояние от оригинальной точки до наиболее близкой к её координатам точки реконструированного облака. Исходя из данного определения возможно вычислить среднеквадратичную ошибку и отношение пикового сигнала к шуму. Обозначим исходные точки как $x_1, x_2, \ldots, x_n$, а реконструированные как $x'_1, x'_2, \ldots, x'_n$, определим среднеквадратичную ошибку как

$$\text{MSE} = \frac{1}{n} \sum_{i=0}^{n} (x_i - x'_i)^2 \tag{1.7}$$

Примем за пиковое значения сигнала максимальное расстояние от точки до её ближайшего соседа внутри оригинального облака. Тогда отношение пикового сигнала к шуму можно определить как

$$\text{PSNR} = 10 \log_{10} \frac{(\text{max distance})^2}{\text{MSE}} \tag{1.8}$$

Указанные метрики вычисляются по алгоритму ”точка-точка”, для каждой из них также может быть сформулирована альтернативная метрика по алгоритму ”точка-плоскость”, при которой ошибка в координатах точек дополнительно проецируется вдоль нормалей точек.

Высокоуровневые метрики для оценки качества изображений также возможно адаптировать для оценки облаков точек. Для этого необходимо произвести рендеринг облака и получить одно или несколько обычных изображений данного облака. Далее, по полученным изображениям возможно рассчитать высокоуровневые метрики качества, такие как SSIM и VIF. Указанный подход целесообразен в случае, когда данные предназначены для восприятия человеком, например, при использовании облака для рендеринга или в системах расширенной реальности.

1.3. Системы оценки качества сжатия облаков точек

Исследователями были разработаны различные системы оценки качества сжатия облаков точек. Рассмотрим принципы их работы и выделим метрики, которые могут быть подсчитаны с помощью данных систем.

1.3.1. mpeg-pcc-dmetric

Данная система была разработана MPEG в рамках работы по стандартизации методов сжатия облаков точек. MPEG выделяет следующие категории облаков точек[14]:

- Статические объекты и сцены
- Динамические объекты
- Динамически считанные объекты и сцены

Категории 1 и 3 были объединены и работа над стандартизацией в данной области ведётся в рамках проекта G-PCC. Работа над стандартизацией методов, работающих с облаками категории 2 ведётся в рамках проекта V-PCC. MPEG также были предложены собственные методы для сжатия облаков точек и разработаны тестовые модели, реализующие данные кодеки, тестовая модель проекта V-PCC имеет название TMC2, тестовая модель проекта G-PCC имеет название TMC13. Для качественной оценки работы данных тестовых моделей MPEG была разработана утилита mpeg-pcc-dmetric, представляющая собой консольное приложение, принимающее на вход два облака точек - оригинальное и реконструированное, и вычисляющая определенные показатели искажения геометрической структуры и атрибутов для данных облаков.

Листинг 1.1: Пример использования утилиты mpeg-pcc-dmetric, параметр fileA - оригинальное облако, fileB - реконструированное облако, color=1 - также вычисляются значения искажения цветов.

```
1 ./pc_error --fileA=./oskull.ply --fileB=./pskull.ply --color=1
```

На листинге 1.1 приведен пример использования данной утилиты. В данном случае, в итоговый отчёт о работе утилиты будет включено минимальное и минимальное и максимальное расстояние до ближайшей точки в оригинальном облаке, среднеквадратичная ошибка для точек, среднеквадратичная ошибка для каждой компоненты цвета, а также отношение пикового сигнала к шуму для точек и каждой компоненты цвета.

Данное программное обеспечивание поддерживается MPEG, однако его исходный код, как и исполняемые файлы, не находятся в открытом доступе. Для доступа к данному ПО необходимо отправить запрос MPEG, при этом доступ выдаётся лишь для некоммерческого использования и только в исследовательских целях. Данные обстоятельства делают невозможным использование данной утилиты в разработке проприетарных кодеков, а непрозрачная система выдачи доступов усложняет проведение независимых исследований.

1.3.2. GeoCNNv1 geo_dist

Авторами кодека GeoCNNv1[21] была разработана собственная система для подсчета качественных метрик. Исходный код утилиты опубликован на Github[5] и написан на языке C++ с использованием библиотеки PCL.

Листинг 1.2: Пример использования утилиты geo_dist, параметр a - оригинальное облако, b - реконструированное облако.

```
pc_error -a test/sphere100.pcd -b test/noisy_sphere100.pcd
```

На листинге 1.2 приведен пример использования данной утилиты, в результате работы программы вычисляются лишь искажения геометрической структуры облака точек. Также, как и утилита от MPEG, данная программа вычисляет минимальное и максимальное расстояние до ближайшего соседа в оригинальном облаке, корень среднеквадратичной ошибки и соответствующее отношение пикового сигнала к шуму. Важным отличием является то, что данная

утилита способна вычислять указанные метрики не только только для случая точка-точка, но и для случая точка-плоскость.

Таким образом, данная система обладает большей функциональностью с точки зрения вычисления геометрического искажения облака, однако не позволяет вычислять искажение цветов и других атрибутов. Несмотря на то, что данная программа имеет открытый исходный код, данная кодовая база не поддерживается разработчиками, так как во второй версии кодека - GeoCNNv2[22], ими было принято решение использовать для оценки качества утилиту mpeg_pcc_dmetric. Другим важным обстоятельством является отсутствие лицензии у данного проекта, в таком случае к исходному коду применяются стандартный копирайт, который не даёт другим разработчикам права модифицировать и распространять данное ПО. Иными словами, несмотря на доступность исходного кода ПО в публичном доступе, оно не является ПО с открытым исходным кодом.

1.3.3. PCCArena

Другими исследователями была предложена комбинированная система для сравнения различных методов сжатия облаков точек - PCCArena[16]. Данная система использует mpeg_pcc_dmetric для оценки искажения геометрической структуры и атрибутов облака точек. Также, в PCCArena было адаптировано решение VMAF от Netflix, предназначенное для оценки качества сжатия изображений и видео с помощью высокоуровневых метрик, а также некоторых методов машинного обучения[6]. Облако точек проходит через рендеринг, в результате которого и получается изображение, которое может быть оценено системой VMAF.

Система имеет открытый исходный код, имеющий лицензию MIT, что позволяет использовать свободно распространять и модифицировать данное ПО, а также использовать в коммерческих целях. Однако, для запуска программы необходим исполняемый файл mpeg_pcc_dmetric, так как данная

утилита не включена в исходный код системы.

1.4. Новая система оценки качества сжатия облаков точек

Рассмотренные системы оценки качества сжатия имеют различные достоинства и недостатки.

Таблица 1: Метрики, вычисляемые различными рассмотренными системами.

	mpeg_pcc_dmetric	geo_dist
$d_{\max}$	+	+
$d_{\min}$	+	+
$\mathrm{MSE}_{\mathrm{geometry}}$	+	+
$\mathrm{MSE}_{\mathrm{attribute}}$	+	-
$\mathrm{PSNR}_{\mathrm{geometry}}$	+	+
$\mathrm{PSNR}_{\mathrm{attribute}}$	+	-
Точка-точка	+	+
Точка-плоскость	-	+

Таблица 2: Характеристики различных рассмотренных систем.

	mpeg_pcc_dmetric	geo_dist
Полнота метрик	+-	+-
Оценка искажения атрибутов	+	-
Поддерживаемость	+	-
Возможность расширения	-	-
Открытый исх. код	-	-

В таблицах 1 и 2 приведена сравнительная характеристика рассмотренных систем. Из проведенного анализа можно сделать вывод, что разработка новой системы оценки качества сжатия облаков точек - актуальная задача. К разрабатываемому решению можно сформулировать следующие требования:

- Возможность вычисления стандартных метрик искажения (MSE и PSNR) геометрической структуры и атрибутов облака точек по алгоритму точка-точка и точка-плоскость
- Использование библиотек, имеющих признание в сообществе разработчиков. Использование стандартных решений вместо написания собственной реализации стандартных функций упрощает разработку, защищает от ошибок, а также делает более простым освоение новыми разработчиками кодовой базы проекта
- Использование технологий разработки качественного программного обеспечения. Документированный код, спроектированный для возможности простого и удобного расширения, использование популярных в сообществе руководств по стилю кода и линтеров также положительно влияет на вовлечение новых разработчиков в работу над проектом с открытым исходным кодом
- Покрытие тестами. Тестирование кода позволяет проверить соответствие ПО требованиям, а также безопасно модифицировать его без нарушения существующей функциональности
- Использование лицензии MIT. Данная лицензия позволяет распространять и модифицировать ПО любым разработчикам, а также допускает использование в коммерческих проектах. Благодаря использованию данной лицензии, любой исследователь или организация, разрабатывающая собственный кодек, будут иметь возможность объективно сравнить его показатели с другими существующими кодеками.

ГЛАВА 2. ОБОСНОВАНИЕ ВЫБОРА ТЕХНОЛОГИЙ И СРЕДСТВ РАЗРАБОТКИ

Правильный выбор технологий и средств разработки имеет значительное влияние на простоту и удобство не только самой разработки и дальнейшей поддержки ПО, но и на качественные характеристики данного ПО. Выбор языка программирования и библиотек напрямую влияет на то, какие готовые решения будут доступны разработчику. В свою очередь, использование готовых, протестированных решений в значительной степени защищает программное обеспечение от ошибок, а также удешевляет процесс разработки.

2.1. Язык программирования и библиотеки

Существующие программы, взаимодействующие с облаками точек разрабатываются на языках программирования C++ и Python. При этом, к наиболее популярным библиотекам для работы с облаками точек можно отнести PCL и Open3D, обе библиотеки реализованы на C++, однако Open3D также имеет привязки для языка Python. Благодаря использованию привязок Open3D, с одной стороны, предоставляет клиентам простые интерфейсы на Python, абстрагирующие внутреннюю сложность данных, а с другой стороны, реализует алгоритмы эффективно благодаря управлению данными на низком уровне с помощью C++.

Для разрабатываемой библиотеки важной является простота освоения новыми разработчиками. Библиотеки для Python активно поддерживаются и хорошо документированы, большое количество исследователей используют данный язык для разработки собственных программ, а также существуют мощные инструменты управления зависимостями, позволяющие начать работу над проектом без необходимости в ручной установке требуемых библиотек. Принимая во внимание указанные особенности, в данной работе для реализации программы был выбран язык Python.

В качестве библиотеки для реализации математических операций был выбран NumPy. Данная библиотека также реализована с использованием языков низкого уровня и предоставляет клиентам интерфейс на языке Python. Например, для решения задач линейной алгебры NumPy использует библиотеку LAPACK, реализованную на языке Fortran.

Клиенты разрабатываемого решения могут использовать его в качестве библиотеки, если их программного обеспечение также разрабатывается на языке Python. Однако для возможности использования данного программного обеспечения в качестве самостоятельного приложения, необходимо реализовать интерфейс для командной строки. Библиотека click содержит готовые функции для аргументов, флагов и опций, считываемых со стандартного потока ввода. Данная библиотека также автоматически генерирует инструкции по использованию программы в виде help-сообщения.

Для модульного тестирования был выбран фреймворк pytest. Он широко поддерживается сообществом, обладает более простым и чистым синтаксисом по сравнению с фреймворком unittest, входящим в стандартную библиотеку Python. Pytest поддерживает генерацию JUnit-отчётов о тестировании, которые поддерживаются большим количеством инструментов для тестирования. Данная библиотека также предоставляет возможность внедрения зависимостей в тесты с помощью фикстур, а также инструменты для мокирования.

Другим важным инструментом, позволяющим преждевременно выявлять ошибки и поддерживать единый стиль кода, являются статические анализаторы, линтеры. В данном проекте использовались линтеры flake8 и pylint. При этом flake8 используется для поддержания стиля кода и проверяет код на соответствие официальному руководству по стилю кода на Python - PEP8. Линтер pylint служит для обнаружения потенциальных ошибок и более тонкого анализа исходного кода. Так, pylint поддерживает статический анализ типов, что помогает избежать большого количества ошибок при разработке на языке, имеющем динамическую типизацию.

Для управления данными зависимостями необходим пакетный менеджер. Пакетный менеджер решает задачи согласования версий различных библиотек между собой, а также с версией интерпретатора, обновления библиотек и установки зависимостей. В качестве пакетного менеджера в данном проекте был выбран poetry. Данный инструмент позволяет устанавливать зависимости как с Python Package Index (далее PyPI), так и из открытых git-репозиториев, группировать зависимости, необходимые в различных средах, а также управлять версиями самого проекта. Поддерживается также экспорт зависимостей в альтернативные форматы, например, в стандартный формат описания зависимостей на Python - файл requirements.txt. Помимо загрузки пакетов с PyPI, poetry способен осуществлять сборку приложения в формат необходимый PyPI и публиковать пакеты на PyPI.

2.2. Непрерывная интеграция

В области непрерывной интеграции существует два решения, являющихся бесплатными и подходящих для проектов с открытым исходным кодом: Github Actions и Gitlab CI/CD. При этом Github не ограничивает использование собственных облачных ресурсов для проектов с открытым исходным кодом, в то время как Gitlab имеет ограничение на количество минут в месяц которое их ресурсами можно пользоваться бесплатно. Другой важной особенностью платформы Github Actions является возможность переиспользования отдельных единиц конфигурации, называемых action. Action может использоваться для настройки среды, например, установки нужной версии интерпретатора Python, или для выполнения сложных действий, таких как сборка и публикация дистрибутива программного обеспечения. С учётом указанных преимуществ, в данном проекте была выбрана платформа Github Actions.

Рассмотрим этапы, которые необходимы для непрерывной интеграции кода в разрабатываемом решении. Группа задач, которая должна быть

выполнена при определенных условиях называется рабочим процессом (Workflow). Для данного проекта существует 3 триггера, срабатывание которых должно приводить к запуску скрипта непрерывной интеграции: создание коммита на главной ветке, создание запроса на слияние с главной веткой и создание тега на главной ветке, при этом все эти триггеры подразумевают выполнение одинакового набора задач и могут быть объединены в один рабочий процесс.

Сценарий интеграционного тестирования обычно разбивается на 3 этапа:

1. Запуск статических и динамических тестов
2. Сборка исходного кода в дистрибутив
3. Запуск высокоуровневых тестов, использующих собранный дистрибутив

В данном случае, этап 3 не является необходимым, поскольку отсутствуют другие компоненты системы, интеграцию с которыми необходимо проверять. Этап 1 включает в себя запуск flake8 и pylint, а также запуск модульных тестов с помощью pytest. В результате работы pytest формируется отчёт о тестировании в формате JUnit, отчёт загружается на Github в виде артефакта конкретного запуска рабочего процесса.

Для реализации этапа 2 необходимо понять, как клиент будет использовать программное обеспечение. В случае веб-приложения, дистрибутивом будет являться docker-образ, позволяющий запустить приложение на любой совместимой платформе. Разрабатываемое решение выполняется в виде исполняемого Python-модуля. Модули в Python являются составной частью пакетов. Для распространения пакетов используется формат wheel. Файл в формате wheel содержит в себе информацию о версии ПО, номере сборки, версии Python, а также платформе, для которой был собран данный дистрибутив. Пакет в формате wheel представлен в виде ZIP-архива, содержащего исходный код и необходимые для работы библиотеки служебные файлы. Сборка и публикация дистрибутива в формате

wheel могут быть осуществлены с помощью poetry, при этом в переменных среды непрерывной интеграции необходимо сконфигурировать ключ, который будет использоваться для загрузки дистрибутива в PyPI.

ГЛАВА 3. СИСТЕМА ПОДСЧЁТА МЕТРИК

3.1. Алгоритм поиска реконструированной точки

3.1.1. Алгоритм сопоставления точек

Ранее уже был описан алгоритм, по которому вычисляется среднеквадратичная ошибка и отношение пикового сигнала к шуму по двум облакам точек - оригинальному и реконструированному. На его основе можно сформулировать алгоритм сопоставления точек в двух облаках разной размерности.

Пусть X - итерируемое облако точек, а Y - облако точек поиска, пронумеруем точки в каждом облаке так, что $x_0, x_1, \ldots, x_n$ - точки облака X, где n - количество точек в облаке X, а $y_0, y_1, \ldots, y_m$ - точки облака Y, где m - количество точек в облаке Y. Каждая из точек представляет собой вектор, принадлежащий векторному пространству $\mathbb{R}^3$. Составим облако Y' состоящее из n точек по следующему алгоритму: для каждой точки $x_i \in X$ найдем точку $y_j \in Y$, наиболее близкую к x_i, добавим в облако Y' точку $y'_i = y_j$. В итоге, количество точек в облаках X и Y' будет одинаковым, при этом, для любого $i \in [1, n]$, y'_i будет ближайшим соседом точки x_i в облаке Y'. Облако Y' будем называть *облаком соседей* облака X.

В дальнейшем, метрики, вычисляемые подобным способом будем называть *направленными метриками*. Для каждой метрики, вычисляющейся путем прямого сопоставления точек двух облаков, в зависимости от конкретного способа сопоставления, можно сформулировать следующие 3 значения:

- Левостороннее значение. В данном случае, итерируемым облаком является оригинальное облако, а облаком поиска - реконструированное
- Правостороннее значение. В данном случае, итерируемым облаком является реконструированное облако, а облаком поиска - оригинальное

- Симметричное значение. Данное значение является худшим из двух предыдущих. Если значение метрики прямо (обратно) пропорционально качеству облака, то берется меньшее (большее) из значений

3.1.2. KD-дерево

Для реализации описанного алгоритма необходимо выбрать алгоритм для поиска ближайшей точки к данной. Для этой задачи может быть использована структура данных KD-дерево (K-Dimensional - имеющее размерность K)[13, с. 23]. Эта структура данных значительно упрощает операцию поиска ближайшего соседа, что является важным, поскольку данная операция должна быть выполнена для каждой точки облака.

KD-дерево представляет собой бинарное дерево, в котором каждый узел-лист является точкой в пространстве размерности k. Рассмотрим алгоритм построения KD-дерева:

1. Если количество точек в области меньше, чем заданное число - размер листа, то данные точки сохраняются в лист дерева, цикл останавливается
2. Последовательность точек в данной области сортируется по x-координате
3. Проводится гиперплоскость, разделяющая последовательность точек пополам и перпендикулярная оси x.
4. Точки, находящиеся левее гиперплоскости, добавляются в область левого поддерева узел, точки, находящиеся правее гиперплоскости - в область правого поддерева
5. Шаги 1-4 повторяются для поддеревьев, при этом ось сортировки меняется. Порядок смены может быть цикличным(x, y, z, x, $\dots$) или адаптивным

Построенное дерево можно использовать для эффективного поиска ближайшей точки к заданной по следующему алгоритму:

1. Согласно обычному алгоритму поиска в двоичном дереве, определяется

лист, к которому принадлежит заданная точка

2. Последовательным поиском среди точек, хранящихся в узле, определяется ближайшая к заданной точка
3. На основе выбранной ближайшей точки оценивается худшее расстояние
4. В случае, если худшее расстояние меньше, чем расстояние от точки до некоторой разделяющей гиперплоскости, производится повторный поиск в области, отделённой данной гиперплоскостью

Сложность поиска 1 ближайшей точки в сбалансированном KD-дереве составляет $O(\log N)$ в среднем[13, с. 25], что позволяет эффективно производить поиск при построении облака соседей. Таким образом, необходимые структуры данных включают в себя:

- Оригинальное облако точек
- Реконструированное облако точек
- KD-дерево, построенное по реконструированному облаку. Используется для построения облака соседей оригинального облака
- KD-дерево, построенное по оригинальному облаку. Используется для построения облака соседей реконструированного облака
- Облако соседей оригинального облака точек. Используется для вычисления левосторонних значений метрик
- Облако соседей реконструированного облака точек. Используется для вычисления правосторонних значений метрик

3.2. Выбор метрик для оценки качества сжатия атрибутов

3.2.1. Оценка искажения геометрической структуры

Сжатие геометрической структуры облака и сжатие его атрибутов происходит отдельно, однако кодирование атрибутов напрямую зависит от результата сжатия геометрической структуры. Таким образом, для оценки качества сжатия атрибутов облака, также необходимо оценить искажение геометрической структуры облака.

Для данной цели используются метрики, основанные на известных значениях СКО и отношения пикового сигнала к шуму[16]:

- Ассиметричное расстояние Чамфера
- Симметричное расстояние Чамфера
- Отношение пикового сигнала к шуму для расстояния Чамфера
- Метрика Хаусдорфа

Ассиметричное расстояние Чамфера является направленной метрикой и имеет левостороннее и правостороннее значения. Имеет следующую формулу:

$$\text{ACD}\left(P_1, P_2\right) = \frac{\sum_{p_1 \in P_1} \min_{p_2 \in P_2} \|p_1 - p_2\|_2^2}{|P_1|} \tag{3.1}$$

где P_1, P_2 - облака точек, $p_1 \in P_1, p_2 \in P_2$ - точки левого и правого облаков соответственно, а $|P_1|$ - количество точек в левом облаке. Легко заметить, что данная формула совпадает с ранее описанной формулой для вычисления СКО(1.7).

Симметричное расстояние Чамфера является средним между левосторонним и правосторонним значением ассиметричного расстояния Чамфера:

$$\text{CD}\left(P_1, P_2\right) = \frac{\left(\text{ACD}\left(P_1, P_2\right), \text{ACD}\left(P_2, P_1\right)\right)}{2} \tag{3.2}$$

Пусть M - максимальный диаметр оригинального облака среди осей x, y, z, диаметром облака D по некоторой оси является расстояние между двумя наиболее удалёнными точками в облаке относительно данной оси

$$
\begin{aligned}
e_{p_1,p_2} &= p_1 - p_2 \\
\text{Proj}_x &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\
D_x &= \max_{p_1,p_2 \in P} \|\text{Proj}_x e_{p_1,p_2}\| \\
M &= \max\left(D_x, D_y, D_z\right)
\end{aligned} \tag{3.3}
$$

Примем симметричное расстояние Чамфера CD за значение шума, а M - за пиковое значение сигнала. Тогда отношение пикового сигнала к шуму для расстояния Чамфера может быть выражено следующим образом

$$
\text{CD-PSNR} = 10 \log_{10} \frac{M^2}{CD} \tag{3.4}
$$

Расстояние Чамфера и производные от него метрики показывают, насколько в среднем ошибается алгоритм. Для того, чтобы оценить ошибку сжатия в худшем случае, необходимо вычислить метрику Хаусдорфа - максимальное расстояние среди всех пар точек в оригинальном и реконструированном облаке

$$
\text{HD} = \max\left(\max_{p_1 \in P_1}\left(\min_{p_2 \in P_2} \|p_1 - p_2\|_2^2\right), \max_{p_2 \in P_2}\left(\min_{p_1 \in P_1} \|p_1 - p_2\|_2^2\right)\right) \tag{3.5}
$$

3.2.2. Оценка искажения нормалей точек

Качество сжатия нормалей напрямую влияет на реконструкцию геометрической структуры и позволяет определить, в какой плоскости лежит каждая конкретная точка. Данные факторы влияют как на рендеринг облака, так и на распознавание объектов методами компьютерного зрения. Для

оценки искажения нормалей будем использовать *проективные метрики*[19]. Данный вид метрик подразумевает проекцию вектора ошибки вдоль нормали в данной точке.

$$
\begin{aligned}
e^{\text{point}}_{p_1,p_2} &= \|p_2 - p_1\| \\
e^{\text{plane}}_{p_1,p_2} &= \left|\left(e^{\text{point}}_{p_1,p_2}, n_{p_2}\right)\right|
\end{aligned}
\tag{3.6}
$$

При использовании проекции вектора ошибки вдоль нормали вместо обычного вектора ошибки, мы получаем соответствующие аналоги метрик, описанных ранее: ACD^P, CD^P, CD-PSNR^P, HD^P. Данные метрики позволяют оценить искажение поверхности объекта и не требуют больших вычислительных мощностей.

3.2.3. Оценка искажения цветов

Очевидно, что передача цвета играет большую роль в том, как люди воспринимают объекты. Системы машинного зрения также используют информацию о цвете для распознавания объектов[2]. Для оценки искажения цвета также может быть применено отношение пикового сигнала к шуму, однако важной деталью в данном случае является цветовая схема, которая используется для вычислений.

Семейство цветовых схем YCC более точно соотносится с человеческим восприятием, чем RGB[20, с. 291]. По этой причине, в разрабатываемом решении искажение цвета будем считать не только в цветовой схеме RGB, но и в Y’CbCr. Преобразование будем производить по стандарту ITU-R BT.709, используя следующую операцию

$$\begin{pmatrix} Y' \\ C_B \\ C_R \end{pmatrix} = \begin{pmatrix} 0.2126 & 0.7152 & 0.0722 \\ -0.1146 & -0.3854 & 0.5 \\ 0.5 & -0.4542 & -0.0458 \end{pmatrix} \begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} \quad (3.7)$$

Данное преобразование является достаточно простым, так как требует лишь одной операции умножения матрицы на вектор для каждого значения цвета. Вычисление ошибки в данной цветовой схеме также поддерживается утилитой mpeg-pcc-dmetric. Яркостная компонента Y’ в данном случае является наиболее важной, посколько именно оттенки серого содержат структурную информацию.

3.2.4. Итоговый перечень метрик

Таким образом, включая стандартные метрики СКО и отношение пикового сигнала к шуму, в разрабатываемом решении предлагается вычислять следующие метрики:

Таблица 3: Метрики, вычисляемые в разрабатываемом решении.

	Лев.	Прав.	Симм.	Проец.
$MSE_{коорд}$	+	+	+	+
$PSNR_{коорд}$	+	+	+	+
CD	+	+	+	+
HD	+	+	+	x
MSE_{RGB}	+	+	+	x
$PSNR_{RGB}$	+	+	+	x
$MSE_{Y'CbCr}$	+	+	+	x
$PSNR_{Y'CbCr}$	+	+	+	x

3.3. Архитектура разрабатываемого решения

Вычисляемые метрики образуют достаточно сложный граф зависимостей в сравнении с достаточно небольшой сложностью реализации каждой из них. Из-за этой особенности появляется необходимость в управлении данными зависимостями.

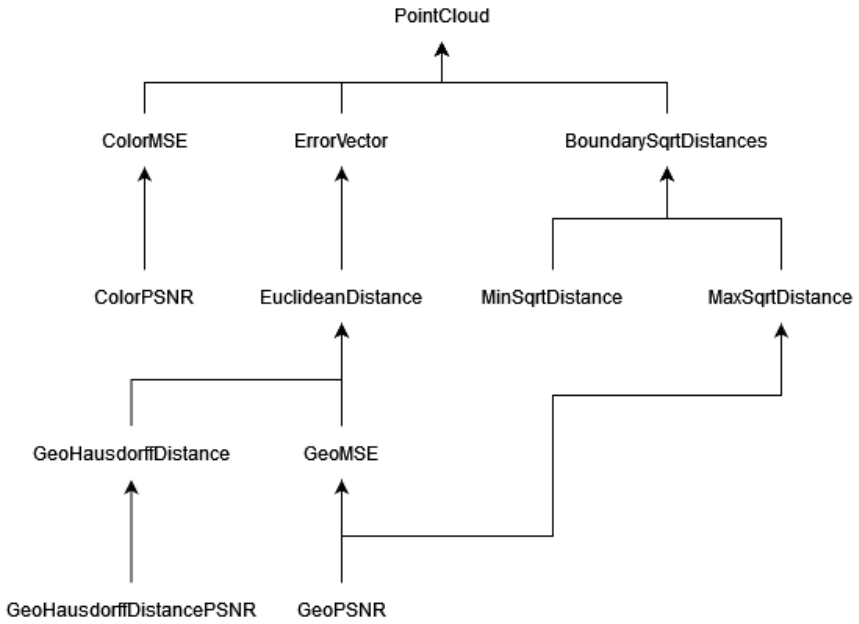


Рис. 3.1: Граф зависимостей вычисляемых метрик

Для решения задачи управления зависимостями в данном проекте были использованы шаблоны проектирования ”Стратегия” и ”Посредник”. Шаблон ”Стратегия” позволяет вынести части алгоритма, осуществляющие разные действия над схожими данными, в отдельные классы, при этом объемлющий класс выполняет роль контекста, от которого зависит, какая стратегия будет выполнена при конкретных условиях, а также решает, какие данные будут переданы стратегии.

В роли объемлющего класса выступает класс CloudPair, задача которого - сопоставление оригинального и реконструированного облаков точек. Данный класс содержит в себе сами облака точек, КД-деревья и облака соседей. В роли классов-стратегий выступают отдельные метрики, подобное решение позволяет вынести алгоритм вычисления метрик из объемлющего класса и, тем

самым, соблюсти условие единственной ответственности.

При реализации данного подхода напрямую, однако, возникают проблемы, связанные с двунаправленной зависимостью между объемлющим классом и стратегиями. В классической реализации данного шаблона все стратегии требуют одни и те же данные и при этом напрямую не зависят от родительского класса. В данном случае, стратегии зависят от объемлющего класса, поскольку вынуждены обращаться к нему напрямую из-за наличия разных зависимостей. При этом объемлющий класс также становится зависим от стратегий, поскольку в зависимости от типа стратегии ей нужно передавать различные данные.

Проблему двунаправленной зависимости в данном случае помогает решить введение класса-посредника MetricCalculator. Задача данного класса - передача необходимых данных в конкретные стратегии, при этом класс CloudPair перестаёт зависеть от стратегий. Дополнительно снизить связность классов можно, разделив все метрики на два подкласса - первичные и вторичные. Первичные метрики требуют для своего вычисления класс CloudPair, при этом передачу CloudPair в стратегии осуществляет посредник. Вторичные метрики зависят лишь от первичных метрик и других вторичных метрик. К первичным метрикам можно отнести среднеквадратичное отклонение координат точек, а к вторичным - отношение пикового сигнала к шуму, оцененного с помощью СКО координат точек. Вторичные метрики сообщают о своих зависимостях, реализуя метод _get_dependencies, данный метод возвращает список стратегий, которые будут необходимы для вычисления данной стратегии. Класс-посредник определяет зависимости стратегий посредством вызова метода _get_dependencies и осуществляет внедрение данных зависимостей. При этом MetricCalculator также содержит словарь, в котором находятся все ранее вычисленные метрики, что позволяет избежать повторного вычисления.

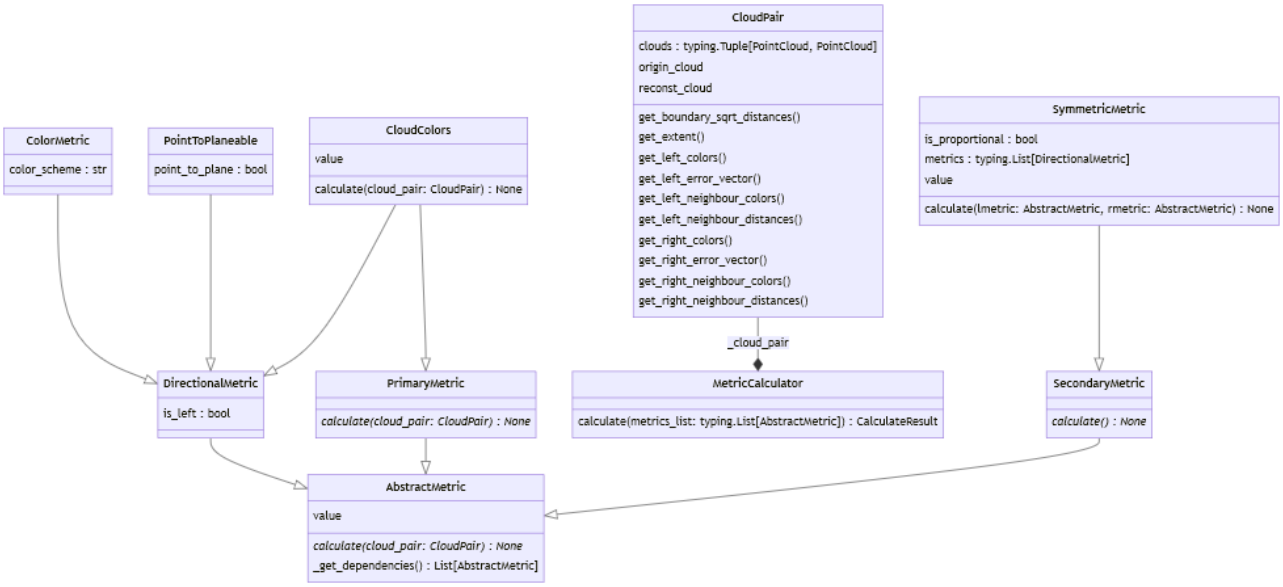


Рис. 3.2: UML-диаграма базовых классов программы

Итоговая архитектура представлена на рисунке 3.2. Данная архитектура позволяет снизить связность классов и значительно упростить тестирование. При этом для добавления новой метрики достаточно лишь определить её зависимости в методе _get_dependencies и реализовать алгоритм, преобразующий входные метрики. Подобное решение также упрощает реализацию многопоточности, класс CloudPair предоставляет клиентам доступ лишь на чтение, а метрики имеют один публичный метод - calculate. Таким образом, для внедрения многопоточности необходимо лишь разделить обработку входного списка метрик в классе MetricCalculator между потоками.

ГЛАВА 4. РЕЗУЛЬТАТЫ РАБОТЫ

4.1. Консольное приложение

В результате работы было разработано консольное приложение для оценки качества облака точек при наличии оригинального облака точек в качестве образца.

Листинг 4.1: Интерфейс разработанного консольного приложения

```
$ python3 -m open_pcc_metric --help
Usage: python -m open_pcc_metric [OPTIONS]

Options:
  --ocloud TEXT       Original point cloud. [required]
  --pcloud TEXT       Processed point cloud. [required]
  --color [rgb|ycc]   Report color distortions as well.
  --hausdorff         Report hausdorff metric as well. If --point-to-plane is
                      provided, then hausdorff point-to-plane would be reported
                      too
  --point-to-plane    Report point-to-plane distance as well.
  --help              Show this message and exit.
  --csv               Print output in csv format.
```

Интерфейс приложения приведен на листинге 4.1. Разработанная программа имеет несколько опций, позволяющих пользователю указать интересующие его метрики, которые должны быть включены в отчёт программы. При этом по-умолчанию программа подсчитывает стандартный набор метрик в который входит максимальное и минимальное расстояние между соседними точками, среднеквадратичное отклонение геометрической структуры и отношение пикового сигнала к шуму. Рассмотрим подробнее входные параметры программы:

- Опция –ocloud. Файл с оригинальным облаком.
- Опция –pcloud. Файл с реконструированным облаком.
- Опция –color. Не является обязательной и имеет два допустимых

входных значения: ”rgb” и ”ycc”. При наличии данного параметра, программа вычислит метрики искажения цветов исходных точек. Значение параметра определяет цветовую схему, в которой будут производится вычисления. При значении параметра ”rgb”, итоговые метрики будут вычислены в цветовой схеме RGB, при этом значением каждой из метрик будет вектор размерности 3, где компоненты соответствуют красной, зелёной и голубой составляющей цвета точек соответственно. При значении параметра ”ycc”, в качестве цветовой схемы будет выбрана схема Y’CbCR с преобразованием по стандарту BT.709, компоненты итогового вектора соответствуют искажениям яркости, синей и красной цветоразностным компонентам соответственно.

- Опция –hausdorff. Не является обязательной, при наличии, сообщает программе, о том, что значение метрики Хаусдорфа также должно быть вычислено и включено в отчёт.
- Опция –point-to-plane. Не является обязательной, при наличии, сообщает программе, о необходимости вычислить проективные метрики. Если данная опция передана вместе с опцией –hausdorff, то в отчёт будет также включена проективная метрика Хаусдорфа.
- Опция –csv. Не является обязательной, при наличии, сообщает программе, что вывод результата должен быть осуществлён в формате CSV.

```
● open-pcc-metric-py3.10vscode ➜/workspaces/open-pcc-metric (unittests) $ python3 -m open_pcc_metric \
  > --ocloud="./files/oskull_reduced.ply" \
  > --pcloud="./files/pskull_reduced.ply" \
  > --color="ycc"
                  label is_left point-to-plane                                    value
0        MinSqrtDistance                                          0.0005394594300728226
1        MaxSqrtDistance                                             3.3819776943374595
2                GeoMSE    True          False                    0.008200834632373888
3                GeoMSE   False          False                    0.008181548987474831
4     GeoMSE(symmetric)                                           0.008200834632373888
5               GeoPSNR    True          False                        72.4728937253675
6               GeoPSNR   False          False                       72.48311891997054
7    GeoPSNR(symmetric)                                            72.4728937253675
8              ColorMSE    True                [1.26237366e-05 2.97011418e-07 2.35043354e-07]
9              ColorMSE   False                [1.86551090e-05 4.75979959e-07 3.83567251e-07]
10  ColorMSE(symmetric)                        [1.86551090e-05 4.75979959e-07 3.83567251e-07]
11            ColorPSNR    True                          [48.98812076 65.27226855 66.28852024]
12            ColorPSNR   False                          [47.29202209 63.22411332 64.1615848 ]
13 ColorPSNR(symmetric)                                  [47.29202209 63.22411332 64.1615848 ]
```

Рис. 4.1: Пример вывода программы

4.2. Оценка качества сжатия атрибутов

Рассмотрим применение разработанного приложения на примере реальных данных. Для данной задачи была выбрана модель longdress_vox10_1051 из датасета 8iVFB v2[10]. Чтобы произвести оценку, необходимо выполнить следующие шаги:

1. Выбрать некоторый кодек для сжатия облака точек. Для данного эксперимента будем использовать кодек TMC13.
2. Осуществить сжатие облака с параметрами, слабо искажающими атрибуты точек. Назовём данное облако LD (Low Distortion - слабое искажение)
3. Повторно осуществить сжатие с параметрами, сильно искажающими атрибуты точек. Данное облако назовём SD (Severe Distortion - сильное искажение)
4. Осуществить декомпрессию облаков LD и SD
5. Вычислить метрики качества для облаков LD и SD

Для того чтобы задать необходимый уровень искажения атрибутов, например цветов, TMC13 позволяет задать параметр квантования (qp - quantization parameter). Чем выше значение данного параметра, тем больше

искажение атрибута, для которого задаётся данный параметр.

Листинг 4.2: Пример сжатия облака точек посредством кодека TMC13, используются параметры, обеспечивающие сильное искажение цвета.

```
tmc3 --uncompressedDataPath=./origin/longdress_vox10_1051.ply \
  --compressedStreamPath=./sd/longdress_vox10_1051.ply.bin \
  --mode=0 \
  --qp=51 \
  --attribute=color \
  --positionQuantizationScale=0.9375
```

Команда для сжатия облака точек с сильным искажением цвета приведена на листинге 4.2. Дополнительно задан параметр positionQuantizationScale для внесения незначительной дисторсии в координаты точек, поскольку в ином случае программа работает в режиме сжатия без потерь.

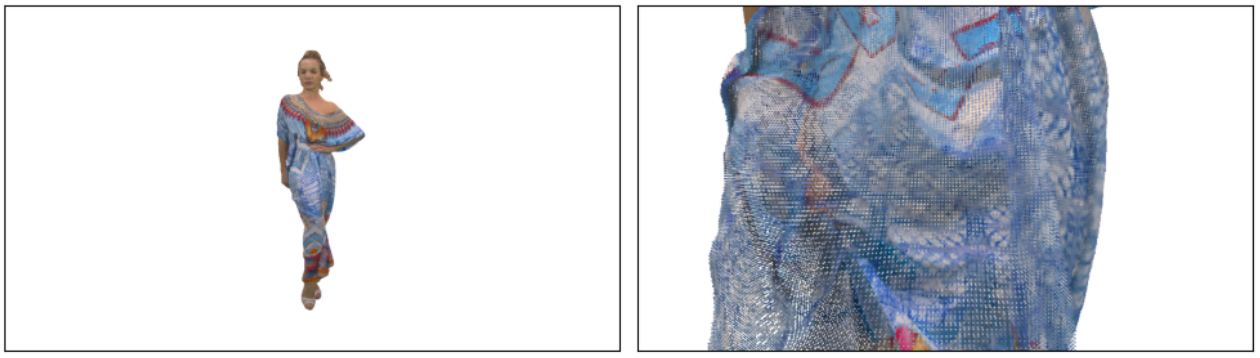


Рис. 4.2: Оригинальное облако точек и его увеличенная версия

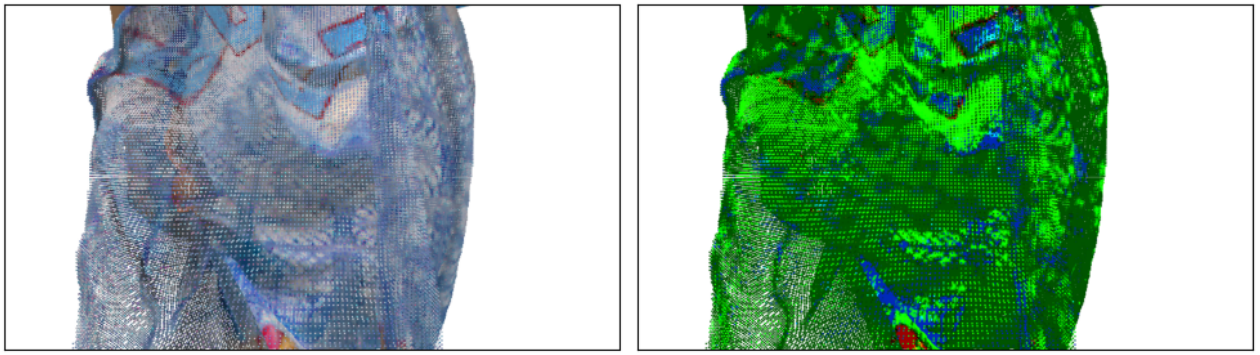


Рис. 4.3: Облака точек LD и SD

На рисунке 4.2 изображено облако точек модели longdress и его увеличенный фрагмент. На рисунке 4.3 изображен данный увеличенный фрагмент для облаков точек LD и SD.

Таблица 4: Метрики, вычисляемые различными рассмотренными системами.

Метрика / Облако	LD	SD
CD-PSNR(дБ)	65.443136	65.443136
Y' – PSNR(дБ)	83.18469189	62.45301073
C_B – PSNR(дБ)	89.59494615	62.73502595
C_R – PSNR(дБ)	78.44316297	62.24977632

Рассмотрим результаты работы программы, приведенные в таблице 4. Мы видим, что отношение пикового сигнала к шуму для расстояния Чамфера CD-PSNR является одинаковым для обоих облаков, что означает, что искажение геометрической структуры облаков LD и SD является одинаковым. Это связано с тем, что при кодировании данных использовалось одинаковое значение параметра positionQuantizationScale. Метрика Y' – PSNR отражает качество передачи яркостной компоненты, здесь мы видим, что у облака LD значение

данной метрики выше, чем у облака SD. При этом, поскольку именно яркостная компонента влияет на восприятие структурной информации, определение объекта по облаку SD может являться более затруднительным. Метрики C_B – PSNR и C_R – PSNR показывают качество передачи цветоразностных компонент цвета, данные значения не влияют на структурное содержание, однако непосредственно связаны с внешними характеристиками объекта, по значениям данных метрик в таблице 4 можно сделать вывод, что внешние характеристики объекта также искажены сильнее в облаке SD, что также наглядно видно на рисунке 4.3.

4.3. Интеграция с системой PCCArena

Система бенчмаркинга PCCArena использует для вычисления математически-обоснованных метрик утилиту mpeg-pcc-dmetric. Поскольку разработанное приложение реализует большую часть функциональности mpeg-pcc-dmetric, оно может быть интегрировано в систему PCCArena. В настоящий момент данная система не может быть использована без mpeg-pcc-dmetric, что означает, что для использования системы необходимо запрашивать данное программное обеспечение у MPEG. С помощью разработанной программы, данная система может быть запущена без наличия программного обеспечения от MPEG.

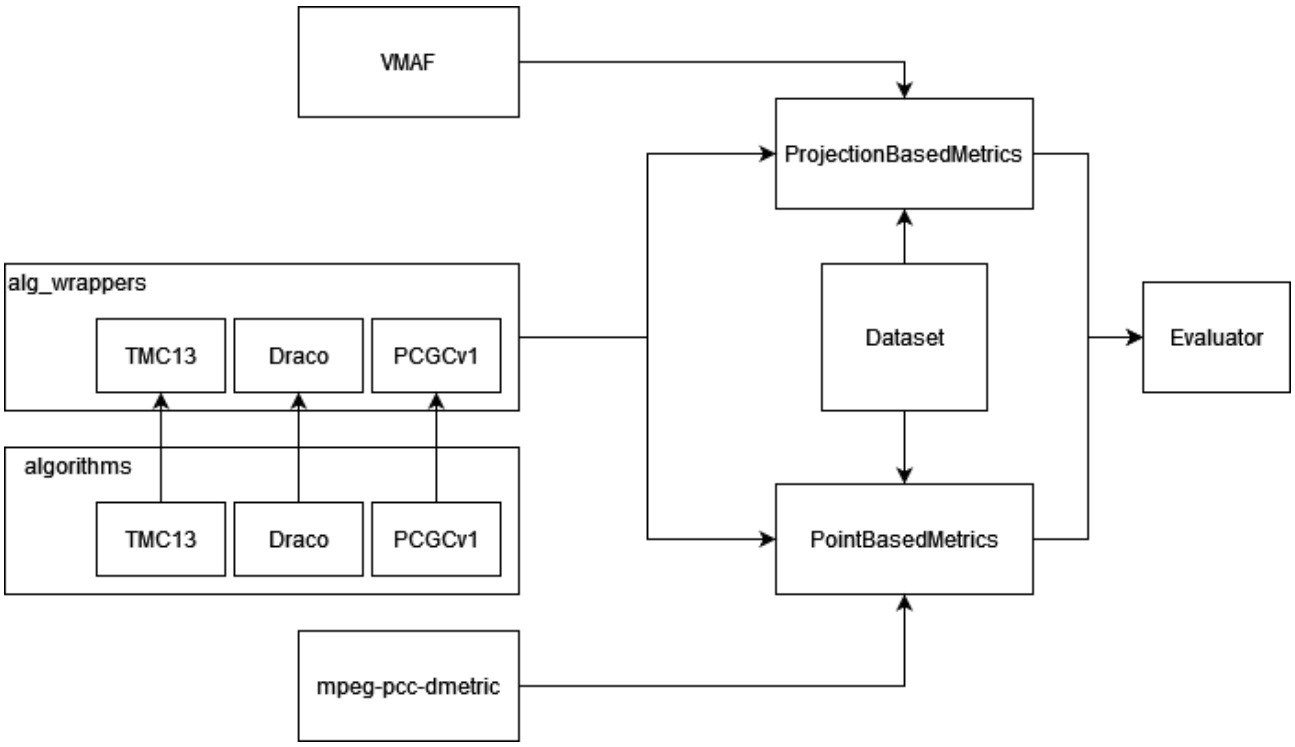
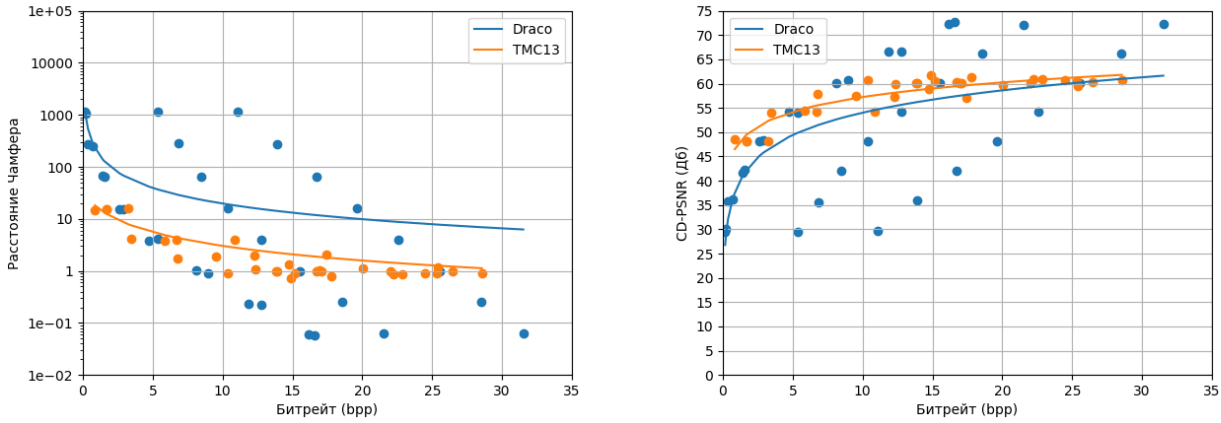


Рис. 4.4: Архитектура системы бенчмаркинга PCCArena

Архитектура PCCArena приведена на рисунке 4.4. Для внедрения разработанного приложения в данную системы, были внесены соответствующие изменения в модуль PointBasedMetrics. С помощью полученной системы было произведено качественное сравнение отобранных кодеков (Draco, TMC13). В качестве тестового набора данных использовался датасет ShapeNet Core [24].

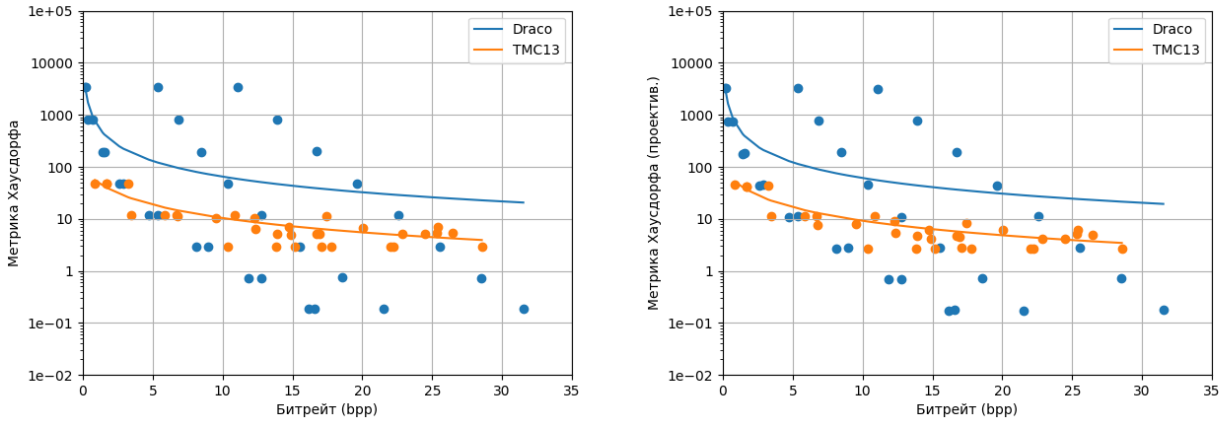
При построении графиков значение конкретной метрики целесообразно изображать в зависимости от значения битрейта - среднего количества бит, затраченных на кодирование точек (далее bpp - bits per point). Подобный подход позволяет с одной стороны, оценить эффективность конкретного метода сжатия в рамках их основной задачи - уменьшения размера информации, а с другой стороны, численно выразить искажение исходной информации в зависимости от степени сжатия.



(а) (б)

Рис. 4.5: (a) Зависимость расстояния Чамфера от битрейта. (b) Зависимость CD-PSNR от битрейта.

На рисунке 4.5 приведен график, изображающий зависимость расстояния Чамфера от битрейта. По данному графику можно сделать вывод, что кодек Draco показывает себя хуже относительно TMC13 при низком битрейте. При высоком битрейте, кодеки показывают схожий результат.



(а) (б)

Рис. 4.6: (a) Зависимость метрики Хаусдорфа от битрейта. (b) Зависимость проецированной метрики Хаусдорфа от битрейта.

На рисунке 4.6 приведен график зависимости метрики Хаусдорфа, а также значение данной метрики при проецировании вдоль нормалей точек. Данные графики показывают ошибку метода сжатия в худшем случае, при этом по графику можно сделать вывод, что соотношение ошибки в худшем

случае и в среднем слабо отличается для приведенных методов сжатия. Кроме того, можно отметить, что при проецировании ошибки вдоль нормалей, кодек TMC13 показывает незначительно большее значение метрики Хаусдорфа, при этом значения для кодека Draco остаются прежними, что может говорить о большем искажении нормалей кодеком TMC13.

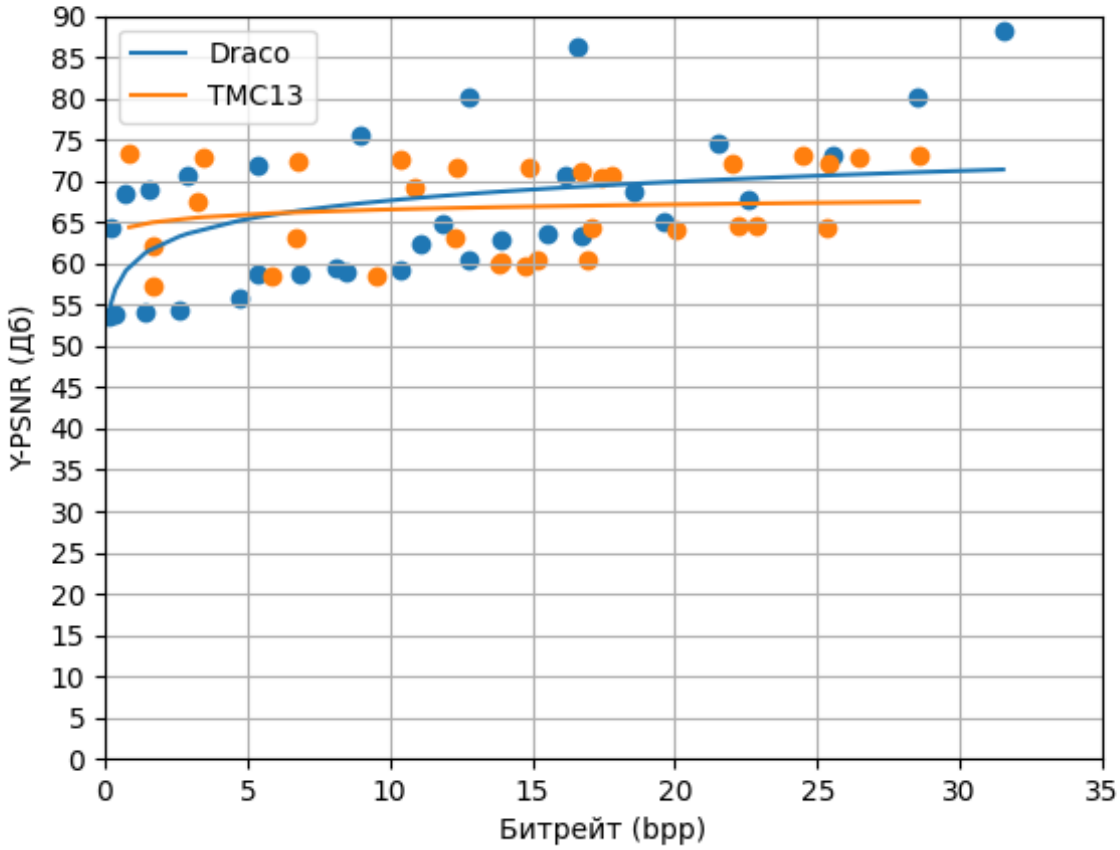


Рис. 4.7: Зависимость Y-PSNR от битрейта.

На графике 4.7 приведено отношение пикового сигнала к шуму для яркостной компоненты цветовой схемы Y’CbCr. По графику можно сделать вывод, что кодек Draco сильно искажает цвета при малом битрейте, однако, при повышении битрейта, показывает лучшие значения по сравнению с TMC13. Для TMC13 качество передачи цвета слабо меняется в зависимости от битрейта.

ЗАКЛЮЧЕНИЕ

В данной работе был проведён анализ существующих систем оценки методов сжатия облаков точек и их атрибутов, а также определён набор метрик для оценки качества сжатия атрибутов облаков точек. На основе проведённого анализа разработана программа для оценки качества облака точек при наличии оригинального облака точек. Произведён качественный сравнительный анализ кодеков Draco и TMC13. Полученные данные позволяют судить о качестве сжатия облаков точек данными кодеками при различной степени сжатия. Разработанное решение упростит оценку методов сжатия атрибутов облаков точек и может быть полезным исследователям, ведущим разработки в данной области, а также может быть использовано при разработке проприетарных кодеков.

СПИСОК ИСТОЧНИКОВ

1. *(auth.) P. D. S.* Data Compression: The Complete Reference. — Springer, 2007. — ISBN 9781846286025; 1846286026; 9781846286032; 1846286034.
2. 2020 Autonomous Vehicle Technology Report [Электронный ресурс]. — URL: https://www.wevolver.com/article/2020.autonomous.vehicle.technology.report (дата обр. 10.05.2024).
3. Draco [Электронный ресурс]. — URL: https://github.com/google/draco (дата обр. 08.05.2024).
4. TMC13 [Электронный ресурс]. — URL: https://github.com/MPEGGroup/mpeg-pcc-tmc13 (дата обр. 08.05.2024).
5. geo_dist [Электронный ресурс]. — URL: https://github.com/mauriceqch/geo_dist (дата обр. 08.05.2024).
6. Toward A Practical Perceptual Video Quality Metric [Электронный ресурс]. — URL: https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652 (дата обр. 10.05.2024).
7. Attribute compression of 3D point clouds using Laplacian sparsity optimized graph transform / Y. Shao [и др.] // 2017 IEEE Visual Communications and Image Processing (VCIP). — IEEE, 12.2017. — DOI: 10.1109/vcip.2017.8305131. — URL: http://dx.doi.org/10.1109/VCIP.2017.8305131.
8. *Bellocchio F.* 3D Surface Reconstruction: Multi-Scale Hierarchical Approaches. — 1-е изд. — Springer, 2013. — ISBN 9781461456315; 1461456312; 9781461456322; 1461456320.
9. Cluster-based two-branch framework for point cloud attribute compression / L. Sun [и др.] // The Visual Computer. — 2023. — Нояб. — ISSN 1432-2315. —

DOI: 10.1007/s00371-023-03146-9. — URL: http://dx.doi.org/10.1007/s00371-023-03146-9.

10. *Garcia D. C.*, *Fonseca T. A.*, *Queiroz R. L. de*. Example-based super-resolution for point-cloud video. — 2018. — eprint: arXiv:1803.06466.

11. Image Quality Assessment: From Error Visibility to Structural Similarity / Z. Wang [и др.] // IEEE Transactions on Image Processing. — 2004. — Апр. — Т. 13, № 4. — С. 600—612. — ISSN 1057-7149. — DOI: 10.1109/tip.2003.819861. — URL: http://dx.doi.org/10.1109/tip.2003.819861.

12. Image quality assessment: from error visibility to structural similarity / Z. Wang [и др.] // IEEE Transactions on Image Processing. — 2004. — Т. 13, № 4. — С. 600—612. — DOI: 10.1109/TIP.2003.819861.

13. *Kuo(auth.) S. L. Z. K.-C. J.* 3D Point Cloud Analysis: Traditional, Deep Learning, and Explainable Machine Learning Methods. — 1st ed. — Springer, 2021. — ISBN 9783030891794; 3030891798; 9783030891800; 3030891801.

14. *MPEG*. Call for Proposals for Point Cloud Compression V2 : тех. отч. / ISO/IEC. — Hobart, AU, 04.2017. — MPEG2017/N16763.

15. Multiscale Point Cloud Geometry Compression / J. Wang [и др.]. — 2020. — DOI: 10.48550/ARXIV.2011.03799. — URL: https://arxiv.org/abs/2011.03799.

16. PCC arena: a benchmark platform for point cloud compression algorithms / C.-H. Wu [и др.] // Proceedings of the 12th ACM International Workshop on Immersive Mixed and Virtual Environment Systems. — ACM, 06.2020. — (MMSys '20). — DOI: 10.1145/3386293.3397112. — URL: http://dx.doi.org/10.1145/3386293.3397112.

17. *Pedersen M.*, *Hardeberg J. Y.* — 2012. — DOI: 10.1561/0600000037.

18. Point Cloud Attribute Compression via Successive Subspace Graph Transform / Y. Chen [и др.] // 2020 IEEE International Conference on Visual Communications and Image Processing (VCIP). — IEEE, 12.2020. — DOI: 10.1109/vcip49819.2020.9301784. — URL: http://dx.doi.org/10.1109/VCIP49819.2020.9301784.

19. Point Cloud Quality Assessment Using Multi-Level Features / J. Lv [и др.] // IEEE Access. — 2024. — Т. 12. — С. 47755—47767. — ISSN 2169-3536. — DOI: 10.1109/access.2024.3383536. — URL: http://dx.doi.org/10.1109/ACCESS.2024.3383536.

20. *Poynton C. A.* Digital Video and HD. — Oxford, England : Morgan Kaufmann, 10.2001. — (The Morgan Kaufmann Series in Computer Graphics).

21. *Quach M.*, *Valenzise G.*, *Dufaux F.* Learning Convolutional Transforms for Lossy Point Cloud Geometry Compression // 2019 IEEE International Conference on Image Processing, ICIP 2019, Taipei, Taiwan, September 22-25, 2019. — IEEE, 2019. — С. 4320—4324. — DOI: 10.1109/ICIP.2019.8803413. — URL: https://doi.org/10.1109/ICIP.2019.8803413.

22. *Quach M.*, *Valenzise G.*, *Dufaux F.* Improved Deep Point Cloud Geometry Compression. — 2020. — arXiv: 2006.09043 [cs.CV].

23. *Rusu R. B.*, *Cousins S.* 3D is here: Point Cloud Library (PCL) // IEEE International Conference on Robotics and Automation (ICRA). — Shanghai, China : IEEE, 05.2011.

24. ShapeNet: An Information-Rich 3D Model Repository : тех. отч. / A. X. Chang [и др.] ; Stanford University — Princeton University — Toyota Technological Institute at Chicago. — 2015. — ArXiv:1512.03012 [cs.GR].