**Министерство науки и высшего образования Российской Федерации** ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**ITMO University**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА GRADUATION THESIS**

**Программный модуль на основе NeRF и гауссова сплэттинга для решения задачи генерации трёхмерных объектов**

**Обучающийся / Student** Чечетко Владислав Владимирович **Факультет/институт/кластер/ Faculty/Institute/Cluster** факультет программной инженерии и компьютерной техники

**Группа/Group** P34201

**Направление подготовки/ Subject area** 09.03.04 Программная инженерия **Образовательная программа / Educational program** Нейротехнологии и программирование 2021

**Язык реализации ОП / Language of the educational program** Русский

**Квалификация/ Degree level** Бакалавр

**Руководитель ВКР/ Thesis supervisor** Кугаевских Александр Владимирович, кандидат технических наук, Университет ИТМО, факультет программной инженерии и компьютерной техники, доцент (квалификационная категория "ординарный доцент")

Обучающийся/Student

|  |  |
| --- | --- |
| Документ  подписан |  |
| Чечетко Владислав  Владимирович |
| 18.05.2025 |

Чечетко Владислав

Владимирович

(эл. подпись/ signature) (Фамилия И.О./ name and surname)

Руководитель ВКР/ Thesis supervisor

|  |  |
| --- | --- |
| Документ  подписан |  |
| Кугаевских Александр  Владимирович |
| 18.05.2025 |

Кугаевских Александр

Владимирович

(эл. подпись/ signature) (Фамилия И.О./ name and surname)

**Министерство науки и высшего образования Российской Федерации** ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

# НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

**ITMO University**

# ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ / OBJECTIVES FOR A GRADUATION THESIS

**Обучающийся / Student** Чечетко Владислав Владимирович **Факультет/институт/кластер/ Faculty/Institute/Cluster** факультет программной инженерии и компьютерной техники

**Группа/Group** P34201

**Направление подготовки/ Subject area** 09.03.04 Программная инженерия **Образовательная программа / Educational program** Нейротехнологии и программирование 2021

**Язык реализации ОП / Language of the educational program** Русский

**Квалификация/ Degree level** Бакалавр

**Тема ВКР/ Thesis topic** Программный модуль на основе NeRF и гауссова сплэттинга для решения задачи генерации трёхмерных объектов

**Руководитель ВКР/ Thesis supervisor** Кугаевских Александр Владимирович, кандидат технических наук, Университет ИТМО, факультет программной инженерии и компьютерной техники, доцент (квалификационная категория "ординарный доцент")

**Характеристика темы ВКР / Description of thesis subject (topic)**

**Тема в области фундаментальных исследований / Subject of fundamental research:** нет / not

**Тема в области прикладных исследований / Subject of applied research:** да / yes

**Основные вопросы, подлежащие разработке / Key issues to be analyzed**

Разработать механизм генерации трехмерных объектов для выявления ограничений в их применении.

**Форма представления материалов ВКР / Format(s) of thesis materials:**

отчет 40 - 60 страниц. Презентация 10 - 15 слайдов

**Дата выдачи задания / Assignment issued on:** 03.03.2025

**Срок представления готовой ВКР / Deadline for final edition of the thesis** 19.05.2025

# СОГЛАСОВАНО / AGREED:

Руководитель ВКР/ Thesis supervisor

|  |  |
| --- | --- |
| Документ подписан |  |
| Кугаевских |

(эл. подпись)

|  |  |
| --- | --- |
| Александр  Владимирович |  |
| 13.05.2025 |

Кугаевских Александр Владимирович

Задание принял к исполнению/ Objectives assumed BY

(эл. подпись)

|  |  |
| --- | --- |
| Документ  подписан |  |
| Чечетко Владислав  Владимирович |
| 14.05.2025 |

Чечетко Владислав

Владимирович

Руководитель ОП/ Head of educational program

|  |  |
| --- | --- |
| Документ  подписан |  |
| Лисицына Любовь  Сергеевна |
| 14.05.2025 |

(эл. подпись)

Лисицына Любовь

Сергеевна

**Министерство науки и высшего образования Российской Федерации** ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

# НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

**ITMO University**

# АННОТАЦИЯ

**ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ SUMMARY OF A GRADUATION THESIS**

**Обучающийся / Student** Чечетко Владислав Владимирович **Факультет/институт/кластер/ Faculty/Institute/Cluster** факультет программной инженерии и компьютерной техники

**Группа/Group** P34201

**Направление подготовки/ Subject area** 09.03.04 Программная инженерия **Образовательная программа / Educational program** Нейротехнологии и программирование 2021

## Язык реализации ОП / Language of the educational program Русский

**Квалификация/ Degree level** Бакалавр

**Тема ВКР/ Thesis topic** Программный модуль на основе NeRF и гауссова сплэттинга для решения задачи генерации трёхмерных объектов

**Руководитель ВКР/ Thesis supervisor** Кугаевских Александр Владимирович, кандидат технических наук, Университет ИТМО, факультет программной инженерии и компьютерной техники, доцент (квалификационная категория "ординарный доцент")

# ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ DESCRIPTION OF THE GRADUATION THESIS

## Цель исследования / Research goal

Разработать механизм генерации трехмерных объектов для выявления ограничений в их применении

## Задачи, решаемые в ВКР / Research tasks

1. Выполнить анализ существующих методов для генерации трехмерных объектов. 2. Собрать и подготовить данные, позволяющие обучить и оценить методы. 3. Реализовать модуль, в том числе обучив модели на разных данных. 4. Выполнить тестирование и оценку обученных моделей, выявив ограничение и недостатки технологий .

## Краткая характеристика полученных результатов / Short summary of results/findings

1. Изучена предметная область генерации трехмерных объектов, проведен обзор существующих технологий и моделей. 2. Обучены модели. Реализован модуль для генерации трехмерных объектов с помощью NeRF и гауссова сплеттинга. 3. Выполнена оценка обученных моделей. 4. Выявлены ограничения и недостатки в применении NeRF и гауссова сплеттинга для генерации трехмерных объектов.

Обучающийся/Student

|  |  |
| --- | --- |
| Документ  подписан |  |
| Чечетко Владислав |

Чечетко

Владислав Владимирович

|  |  |
| --- | --- |
| Владимирович |  |
| 18.05.2025 |

(эл. подпись/ signature) (Фамилия И.О./ name and surname)

Руководитель ВКР/ Thesis supervisor

Кугаевских Александр

|  |  |
| --- | --- |
| Документ  подписан |  |
| Кугаевских Александр  Владимирович |
| 18.05.2025 |

Владимирович

(эл. подпись/ signature) (Фамилия И.О./ name and surname)

**сОДЕРЖАНИЕ**

[**СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ 7**](#_Toc198742288)

[**Термины и определения 8**](#_Toc198742289)

[**введение 9**](#_Toc198742290)

[**1 Теоретический обзор 11**](#_Toc198742291)

[1.1 Актуальность разработки 11](#_Toc198742292)

[1.2 Формы представления 3D-данных 13](#_Toc198742293)

[*1.2.1* *2.5D представление 13*](#_Toc198742294)

[*1.2.2* *Полигональные модели 13*](#_Toc198742295)

[*1.2.3* *Воксели 15*](#_Toc198742296)

[*1.2.4* *Облака точек 16*](#_Toc198742297)

[*1.2.5* *Твердотельные/функциональные модели 18*](#_Toc198742298)

[1.3 Обзор аналогов 20](#_Toc198742299)

[*1.3.1 HSP 20*](#_Toc198742300)

[*1.3.2 NeuralRecon 22*](#_Toc198742301)

[*1.3.3 Plenoxels 24*](#_Toc198742302)

[1.4 NeRF 26](#_Toc198742303)

[1.5 GS 29](#_Toc198742304)

[**2** **Сбор данных и Проектирование 32**](#_Toc198742305)

[2.1 Сбор данных 32](#_Toc198742306)

[2.2 Проектирование модуля 35](#_Toc198742307)

[**3** **реализация 37**](#_Toc198742308)

[**4** **Оценка и анализ полученных результатов 47**](#_Toc198742309)

[**Заключение 50**](#_Toc198742310)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 51**](#_Toc198742311)

**СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ**

*NeRF* – Neural Radiance Fields

*GS* – Gaussian Splatting

*GT* – Ground Truth

*AI* – Artificial Intelligence

*ML* – Machine Learning

*PSNR* – peak signal-to-noise ratio

*SSIM* – structural similarity index measure

*SfM* – Structure from Motion

**Термины и определения**

*NeRF* – метод в области AI и ML. Он предлагает способ создания

высоко детализированных, фотореалистичных 3D-представлений

сложных сцен из набора 2D-изображений

*GS* – это технология в мире 3D-графики, которая позволяет превращать

фотографии или видео, сделанные с обычного смартфона, в

фотореалистичные 3D-модели, основана на использования гауссиан

*GT* – это эталонные данные, проверенные и истинные данные

*AI* – это область компьютерных наук, которая разрабатывает и изучает

методы и программное обеспечение, позволяющие машинам

воспринимать окружающую среду и использовать обучение и

интеллект для выполнения действий

*ML* – раздел искусственного интеллекта, изучающий алгоритмы,

способные обучаться на данных и принимать решения

*PSNR* – это нелинейная метрика, которая сравнивает значения пикселей

исходного эталонного изображения со значениями повреждённого

изображения

*SSIM* – это нелинейная метрика полного сравнения, которая

сопоставляет яркость, контрастность и структуру исходного и

ухудшенного изображения

*COLMAP* – это программное обеспечение для 3D-реконструкции из

изображений и видео. Оно реализует современные алгоритмы для

структуры из движения (SfM)

*SfM* – это метод компьютерного зрения, который восстанавливает

трёхмерную структуру сцены из серии двухмерных изображений,

снятых с разных точек зрения

**введение**

В наше время технологии AI и ML активно развиваются и все больше активно применяются в нашей жизни. С помощью них появились рекомендательные системы, которые активно используются, например Netflix и YouTube. Мы используем ML в производстве - роботы с ML-алгоритмами автоматизируют производственные процессы, контролируют качество продукции и оптимизируют логистику. Распознания объектов, прогнозирования риска развития заболеваний и это не предел всех возможностей искусственного интеллекта. В этой работе я хотел бы затронуть использование этих технологий для генерации трехмерных объектов.

Генерация 3D объектов – это мощный инструмент, который находит применение в самых разных областях. Мы используем 3D объекты в медицине для протезов, в архитектуре для моделирования зданий, в 3D-печати. С развитием технологий машинного обучения и компьютерной графики возможности генерации контента становятся все более широкими и доступными. Все больше разных технологий, подходов и идей для решения задачи по генерации трехмерных объектов предлагаются разными компаниями и независимыми разработчиками.

Казалось бы, всё хорошо, но не все технологии идеальны, при их использовании могут выявиться проблемы, и хорошо если мы знаем о них до начала работы, а если они обнаружились уже в запущенной и работающей системе это может привести к ошибкам и результатам, которые нас не удовлетворят. Для того чтобы избежать этого или хотя бы понимать над чем работать, важно знать какими ограничениями и недостатками обладают данные технологии.

С этой целью я решил сравнить и выявить на практике недостатки двух технологий, с помощью программного модуля на их основе. Я рассматривал два подхода, которые и сейчас пользуются успехом и набирают популярность на мировом рынке– это Neural Radiance Fields и Gaussian Splatting.

Цель – Разработать механизм генерации трехмерных объектов для выявления ограничений в их применении.

Для достижения поставленной цели необходимо выполнить следующие задачи:

* Выполнить анализ существующих методов для генерации трехмерных объектов;
* Собрать и подготовить данные, позволяющие обучить и оценить методы;
* Спроектировать архитектуру и функциональность модуля;
* Реализовать модуль;
* Обучить модели на разных данных;
* Выполнить тестирование и оценку обученных моделей выявив ограничение и недостатки технологий;
* Формирования и предоставление результатов;
* Выложить результаты в открытый доступ.

Реализованный программный модуль позволит выявить на практике ограничение и недостатки технологий Neural Radiance Fields и Gaussian Splatting. Ещё это позволит понять и сравнить какая из технологий лучше в своей основе NeRF или GS. А также даёт шанс любому пользователю самому с помощью модуля обучить легко модели на их данных и получить метрики качества по ним. Из чего каждый сам сможет оценить эффективность и результаты работы этих двух технологий.

**1 Теоретический обзор**

* 1. **Актуальность разработки**

Как говорилось выше, в наше время важно тестировать под разными углами и проверять функциональность технологий на практике. К 2025 технологии Neural Radiance Fields и Gaussian Splatting набирают всю большую популярность среди крупных компаний и обычных обывателей интернета [1-2]. Ниже на рисунках 1 и 2 предоставлены графики в подтверждение.

Изображение выглядит как текст, линия, снимок экрана, График

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1 – Количество опубликованных статей GS и звезд GitHub

Изображение выглядит как линия, График, скат, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2 – Количество публикаций связанных с NeRF

Как можно пронаблюдать к 2025 заинтересованность и количество работ, связанных с этими технологиями, сильно возросло. И на данный момент имеет широкий спектр применений.

Изучая статьи Neural Radiance Fields и Gaussian Splatting, можно увидеть, что не так много, как минимум в открытом доступе, статьей связанных с тестированием, выявлением недостатков и ограничений данных технологий. Статьи, которые есть в открытом доступе, связанные с этими темами, зачастую рассматривают работу этих технологий на данных автора или на данных, которые подготовлены самими разработчиками этих технологий для их тестирования(эти данные специально подготовлены, чтобы показывать хорошие результаты при работе с ними), ограничиваясь только показателями метрик, но не рассматривая работу технологий при разных условиях и не пытаясь выявить их ограничения.

Если посмотреть на Российское пространство, то аналогов статей на эту тему нет. В своей работе я реализую программный модуль на основе статьи Neural Radiance Fields и Gaussian Splatting и попытаюсь выявить ограничения этих технологий путем тестирования их на разных данных и при разных условиях, опираясь не только на метрики, но и на то, насколько хорошо технологии работают с отражениями, с большими/маленькими объектами, как влияют лишние детали на фоне у объекта, насколько важно количество данных и их качество.

Исходя из всего выше сказанного разработка подобного модуля для выявления ограничений технологий NeRF и GS актуально. Это поможет, как и компаниям, так и людям, кто захочет использовать их индивидуально для себя.

* 1. **Формы представления 3D-данных**

Полученные 3D-данные можно представить по-разному, и каждое представление выделяется чем-то своим, например визуально. Здесь я рассмотрю формы и форматы представления пространственных данных и их особенности.

* + 1. 2.5D представление

2.5D представление - псевдотрёхмерное частичное описание поверхности. При рендеринге можно менять угол обзора от исходного в небольших пределах. Чаще всего и в основном используется в игровой индустрии. Идея в том, что графика, которая пытается имитировать трёхмерное пространство, но не является трёхмерной [3]. Делает это она с помощью RGBD – цвета и глубины. Пример на рисунки 3 иллюстрирует это.

Изображение выглядит как круг, Красочность, снимок экрана, творческий подход

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3 – 2.5D представление (цвет + глубина, RGBD)

* + 1. Полигональные модели

Полигональные модели или меш - традиционная форма представления трехмерных данных в компьютерной графике. Подход к моделированию объектов путем представления или аппроксимации их поверхностей с использованием многоугольных сеток. Также полигональная сетка может быть представлена в виде примитивов: кубики, сферы, пирамиды, цилиндры [4]. Пример можно увидеть на рисунки 4 нижи.

Изображение выглядит как Фигурка животного, статуя, млекопитающее, кролик

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4 – Полигональная модель кролика

Основным форматом работы с полигональными моделями является .obj.

Физически такие модели представляют из себя пространственный граф, и могут храниться в памяти компьютера в виде двух множеств.

К преимуществам полигональных моделей можно отнести:

* Естественный формат для использования в компьютерной графике, игровых движках;
* Позволяет лучше описывать пространственные особенности объектов (топология, форма поверхности), по сравнению с вексельными моделями низкого разрешения или с облаками точек.

К недостаткам полигональных моделей можно отнести:

* Необходимость создания специального математического аппарата для извлечения признаков из полигональных моделей, например свертки на графах;
* Чувствительность формата к выбросам в данных.
  + 1. Воксели

Воксели являются обобщением пикселей на пространственный случай. Воксельные 3D модели представляют из себя кубические пространственные сетки, в которых объекты представляются множеством маленьких кубиков (вокселей). В компьютере воксели хранятся в виде трехмерной матрицы. Существует ряд форматов, для хранения воксельных моделей, но наиболее распространённым является. vox формат.

Помимо применения в играх и компьютерной графике в качестве необычного графического стиля, воксельное представления трехмерных данных часто встречается в медицине и промышленном производстве. Отличие же вокселей от полигонов в том, что воксели - это истинный 3D кирпичик, а не 2D плоскость, которая «окружает» пустое 3D пространство и внутри эти модели не полые в отличие от полигонов, но из-за этого для их хранения требуется больше памяти. На рисунки 5 ниже можно увидеть пример.

Изображение выглядит как пиксель, игрушка

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 5 – Воксельный гриб

Представления на основе вокселей используются во многих прикладных областях. В компьютерной графике воксели используются для быстрой трассировки лучей, вокселизации поверхностей и твердых тел, анализа теней и видимости. Они в основном ориентированы на быструю визуализацию в реальном времени и, следовательно, нацелены на визуализацию только видимых вокселов. В медицине вокселные представления обычно реализуются в программном обеспечении, обрабатывающем КТ и МРТ-снимки, исследующие органы и структуру тела в трех измерениях [5].

К преимуществам воксельного подхода можно отнести:

* Естественное обобщение подходов, которые применялись для обработки изображений, например трехмерная свертка;
* Данные для обучения легко получить из любого другого типа моделей;
* Большая связь с физическими свойствами объектов по сравнению с полигональными и облачными моделями;
* Возможность представлять воксельные модели в виде обычных векторов.

К недостаткам воксельного подхода можно отнести:

* “Грубость” аппроксимации формы реальных объектов при малом разрешении;
* Объём занимаемой воксельными моделями памяти растёт кубически в зависимости от количества интервалов, на которое разбивается каждая ось.
  + 1. Облака точек

В последние годы представление в виде облака точек стало одним из направлений исследований в области компьютерного зрения и широко используется во многих сферах, таких как автономное вождение, виртуальная реальность, робототехника. Облако точек является типичной формой данных о пространстве в робототехнике и компьютерном зрении. Физически облако точек представляет собой неупорядоченное множество трехмерных радиус-векторов. Иногда для каждого такого вектора (точки) можно дополнительно указать ее цвет, как дополнительное составляющей не влияющее на геометрию [6].

Такая форма представления пространственных данных часто на практике встречается в задачах сканирования помещений или отдельно взятых объектов. Камеры глубины в качестве результата работы возвращают изображение вместе с его картой глубины (RGBD), что является частным случаем облака точек. Восстановления карты глубины для данного изображения часто называют задачами 2.5D, указывая на то, что, снятый глубинной камерой объект с одного ракурса, не может быть дополнен пространственной информацией с другого ракурса. В этом смысле облако точек содержит в себе больше информации о геометрии объектов чем RGBD изображения. Пример можно посмотреть на рисунке 6 ниже.

Изображение выглядит как текст, рисунок, диаграмма, зарисовка

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6 – Объект в облаке точек

К преимуществам облака точек можно отнести:

* Естественный формат данных в задачах пространственного сканирования;
* Данные данного формата могут быть легко получены из полигональных и функциональных моделей.

К недостаткам облака точек можно отнести:

* Неупорядоченность данных приводит к проблеме выбора функции ошибки и необходимости разработки специального математического аппарата;
* Отсутствует информация о связях между точками, что не позволяет корректно восстановить геометрию и топологию исследуемого объекта и требует сложной процедуры постпроцессинга для перевода результата в иные формы представления пространственных данных.
  + 1. Твердотельные/функциональные модели

Основной идеей функционального подхода является использование функционального описания поверхностей и функциональное описание физических свойств объектов. Например, помимо формы объекта, заданной некоторой функцией, можно хранить распределение массовой плотности, и тем самым иметь возможность не только корректно визуализировать объект на всех масштабах, но использовать эту же модель для расчета физических свойств. Для создания и работы с твердотельными моделями существуют специальные системы автоматического проектирования и моделирования (САПР), также в английском эквиваленты CAD и CAM системы.

Функция, с помощью которой описывается геометрия объекта, может иметь разный смысл: с одной стороны это может быть аналитическое описание поверхности, которая ограничивает твёрдое тело в пространстве, с другой стороны, это может быть некоторая специфическая функция, например плотность вероятности случайной величины, которая должна с наибольшей вероятностью принадлежать внутренности объекта. Пример на рисунке 7 ниже.

Изображение выглядит как зарисовка, круг, дизайн, искусство

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 7 – Объект, представленный функциональным подходом

К преимуществам функционального подхода можно отнести:

* Компактное описание сложной пространственной структуры;
* Физически корректная модель;
* Масштабируемость модели;
* Возможность получить модель любого другого формата из данной.

К недостаткам функционального подхода можно отнести:

* Малое количество датасетов (обучающих выборок), использующих данный формат;
* Любые другие форматы крайне проблематично приводить к функциональному;
* Сложно работать с текстурами и шейдерами.

**1.3 Обзор аналогов**

На данный момент актуальных аналогов NeRF и GS как таковых нет. Эти две технологии плотно укрепились на мировом рынке, и все исследования и разработки направленны на их улучшения. В этой части я рассмотрю аналоги технологий по генерации трёхмерных объектов, которые были актуальны до этого, но все равно парой используются. Взглянем на их особенности и структуру.

1.3.1 HSP

HSP или Hierarchical Surface Prediction – это метод воксельной реконструкции 3D-объектов по их 2D-изображениям. Метод позволяет предсказывать воксельные сетки высокого разрешения при реконструкции трёхмерной геометрии объекта по изображениям. Особенностью этого метода являются октодеревья. Чаще всего проблемой представления объекта с высокой детализацией с помощью вокселей является высокие затраты по памяти, и как раз октодеревья решают эту проблему [7].

Октодерево - это древовидная структура данных, в которой у каждого внутреннего узла либо ровно восемь потомков, либо ни одного. По сути, мы берём наш куб (воксель) и разбиваем его на 8 кубов (вершин), а те вершины можно разбить ещё на более меньшие кубы. Важная деталь состоит в том, что некоторые вершины помечаются, как заполненные, а некоторые - как пустые. У пустых вершин не может быть потомков, и нам не надо выделять память для их детализации, что позволяет экономить место. По итогу в центре объекта вексели разбиты на более крупные кубы, а по краям на более мелкие с пустыми вершинами. Как это выглядит и саму структуру октодерева можно посмотреть на рисунке 8 и 9 соответственно.

Изображение выглядит как мультфильм, солнечный фотоэлемент

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 8 – Объект HSP в разрезе

Изображение выглядит как диаграмма, линия, дизайн, оригами

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 9 – Структура октодерева

Если говорить про работу метода, то для начала нужно взглянуть на его схему (рисунок 10).

Изображение выглядит как снимок экрана, текст, диаграмма, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 10 – Схема HSP

На вход HSP подается RGB-изображение, которое он преобразует в вектор признаков **C** и не берётся за генерацию всей воксельной сетки разом. Вместо этого он последовательно получает пространства признаков  и на основании этого продолжает строить октодерево, решая, лежит вершина и соответствующий ей воксель внутри объекта, вне объекта или на его границе. И после этого всего воссоздаёт объект.

Таким образом, HSP - это end-to-end подход, который, используя датасет ShapeNet с полигональными 3D-моделями различных классов объектов, учится по одной фотографии объекта строить его детализированную воксельную модель.

1.3.2 NeuralRecon

NeuralRecon — это фреймворк для реконструкции 3D-сцен в реальном времени по монокулярному видео. Данная модель появилась в 2021 году и оставалась непревзойденной в своей нише несколько лет подряд. Для начала взглянем на её схему (рисунок 11).

Изображение выглядит как текст, снимок экрана, диаграмма, План

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 11 – Схема NeuralRecon

В основе NeuralRecon лежит TSDF (Truncated Signed Distance Function) - усечённая функция расстояния со знаком. В случаи NR TSDF представляет собой трехмерный массив вокселей, в котором каждому вокселю задаётся усечённое расстояние до ближайшей поверхности [8].

На вход у нас подаётся видео, разбитое на фреймы, важно заметить, что не каждый фрейм из видео попадет в нейросеть, а только определённо выделенный набор, обведенный слева жёлтыми рамками. Данный подход вычленения ключевых фреймов позволяется добиться скорости, необходимой для проведения реконструкции в реальном времени.

NeuralRecon предсказывает TSDF с помощью трехуровневого подхода от грубого к мелкому, который постепенно увеличивает плотность разреженных вокселей. Изображения ключевых кадров в локальном фрагменте сначала пропускаются через магистраль изображения для извлечения многоуровневых признаков. Эти элементы изображения затем проецируются обратно вдоль каждого луча и агрегируются в объем 3D-элементов  l представляет собой индекс уровня. На первом уровне плотный объем TSDF St1 прогнозируется. На втором и третьем уровнях повышается дискретизация с последнего уровня конкатенирован с   и используется в качестве входных данных для модулей GRU Fusion и MLP. Объем признаков, определенный в мировом фрейме, поддерживается на каждом уровне как глобальное скрытое состояние ГРУ. На последнем уровне вывод Stl используется для замены соответствующих вокселей в глобальном объеме TSDF, Окончательная реконструкция в срок t [8].

1.3.3 Plenoxels

Plenoxels (пленоптические воксели) — система для фотореалистичного синтеза изображений. Plenoxels представляют сцену в виде разреженной трёхмерной сетки со сферическими гармониками. Это представление может быть оптимизировано с помощью калиброванных изображений с помощью градиентных методов и регуляризации без каких-либо нейронных компонентов [9]. Изначально технология преподносилась как конкурент для NeRF, который сможет его заменить, но в итоге не сыскала популярности и так и не смогла приблизиться к своему предшественнику. Схему Plenoxels можно увидеть ниже (рисунок 12).

Изображение выглядит как диаграмма, линия, снимок экрана, График

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 12 – Схема Plenoxels

Модель представляет собой разреженную воксельную сетку, в которой каждый занятый угол вокселя хранит скалярную величину непрозрачности σ и вектор коэффициентов сферической гармоники (SH) для каждого цветового канала. Называют такое представление пленокселем. Непрозрачность и цвет в произвольном положении и направлении просмотра определяются путем трехлинейной интерполяции значений, сохраненных в соседних вокселах, и оценки сферической гармонии в соответствующем направлении просмотра. Имея набор откалиброванных изображений, оптимизируется модель, напрямую используя потери при рендеринге на обучающих лучах [9].

Рендеринг изображения построен на принципах томографического интегрирования по лучу (volume rendering), аналогично NeRF:

1. Для каждого пикселя изображения по направлению взгляда (camera ray) строится луч;
2. Луч дискретно семплируется вдоль своей траектории через воксельную сетку;
3. В каждом семпле (вокселе) вычисляется вклад этого вокселя в итоговый цвет пикселя, учитывая его плотность и цвет;
4. Значения по всем семплам интегрируются (примерно — накапливаются с учетом затухания), чтобы получить финальный цвет пикселя;

Дальше сцена инициализируется воксельной сеткой (обычно грубой и большой, со временем сетка может становиться более детализированной — это называется voxel pruning и voxel upsampling). Параметры всех вокселей (плотность, параметры SH) задаются как оптимизируемые переменные. С помощью обратного распространения ошибки (backpropagation) они обучаются с использованием стандартного градиентного спуска так, чтобы сымитированные картинки совпадали с набором настоящих фотографий.

На данный момент технология используется только в научных исследованиях и небольших проектах.

**1.4 NeRF**

Нейронные поля сияния (Neural Radiance Fields, NeRF) представляют собой новаторский подход в области ИИ и машинного обучения, особенно в компьютерном зрении и графике. Они предлагают способ создания высокодетализированных, фотореалистичных 3D-представлений сложных сцен из набора 2D-изображений, при этом может работать на небольшом количестве данных, дорисовывая недостающие части, используя нейронное поле сияния [10].

По своей сути нейронное поле сияния - это тип неявного нейронного представления. Оно обучается непрерывной функции, которая сопоставляет 3D-координаты и направления обзора с цветом и плотностью. Это означает, что для любой точки в 3D-пространстве и любого направления обзора модель NeRF может предсказать цвет и непрозрачность в этой точке. Для начало взглянем. Взглянем на схему NeRF на рисунке 13 ниже.

Изображение выглядит как диаграмма, линия, снимок экрана, План

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 13 – Схема NeRF

На вход например подаются фотографии объекта со всех сторон с разных ракурсов, расположение и ракурс камеры и пропускается через функцию F. Функция F представляет собой оптимизированную сеть MLP без сверточных слоев, которая как раз и принимает 3D-координаты (x, y, z) плюс направление 2D-просмотра (θ, φ) в качестве входных данных, создавая 5D-вектор, и выводит плотность (σ), которая представляет собой значения прозрачности, это некоторое число, значение которого даёт нам понять, есть ли в указанной точке пространства какой-то объект или нет, и цвета (r, g, b). По итогу нейронные поля излучения представляют непрерывные сцены математически в виде векторной функции с пятью измерениями

На практике это работает так, что мы пропускаем через наше изображение множество “лучей», задаваемые радиальными координатами θ и φ. Потом на каждом луче строится распределение σ(x), это делается для того, чтобы непрерывно моделировать объём и понимать, когда луч столкнётся с препятствием. Дальше с помощью MVS метода задаются грубые границы объекта t1 и tN и в них сэмплируется N точек. Тут интересный момент для сэмплирования оптимизируются две сети Coarse и Fine. Первая сэмплирует равномерно Nc точек в наших границах, что дает приблизительную оценку вероятного расположения таких объектов, как поверхности или более плотные области. После нормализации задают некоторое распределение, на котором Fine сэмплирует Nf точек, где вероятность нахождения объекта высока, и тем самым получать более точные значение плотности и цвета. Fine сосредотачивает вычислительные усилия на областях полученных с помощью Coarse, обеспечивая мелкие детали, необходимые для получения высококачественного изображения. Примерно как это выглядит можно увидеть на рисунке 14 ниже.

Изображение выглядит как диаграмма, снимок экрана, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 14 – На сэмплированные точки Nc и Nf

В результате у нас имеется множество, состоящие из Nc и Nf, на котором вычисляется финальное значения цвета С(r) Fine сеткой. Это требует значительно больше вычислительных ресурсов, но даёт более точную оценку распределения. В итоге вычисляется функция потерь, которая представляет из себя:

, (15)

где:

- Предсказанное значение цвета на множестве Nc,

- Предсказанное значение цвета на множестве Nf,

- Настоящие значение цвета.

В дальнейшем функция потерь оптимизируется с помощью Adam. В итоге мы имеем технологию NeRF, которая позволяет нам генерировать 3d объекты и сцены в высоком качестве, хоть для этого и требуются большие вычисления.

**1.5 GS**

Gaussian Splatting — это технология в мире 3D-графики, которая позволяет превращать фотографии или видео, в фотореалистичные 3D-модели. В основе этой технологии лежит использование специальных алгоритмов, которые берут набор изображений и преобразуют их в объёмную модель. Это происходит благодаря «размазыванию» данных («splatting») для создания плавных переходов между точками. У нас имеются: три координаты положения в пространстве, для многомерного нормального (гауссовского) распределения нужна ковариационная матрица размерности 3×3, которая показывает, насколько вытянут или сплюснут эллипсоид, альфа-канал (прозрачность), который отвечает за прозрачность гауссиан и три значения RGB(цвет) [11]. Отличительной чертой GS является то, что от большинства решений, основанных на точках, для которых требуются данные Multi-View Stereo (MVS), он достигает высококачественных результатов, используя только точки SfM в качестве входных данных. Структуру метода можно увидеть на рисунке 15 ниже.

Изображение выглядит как диаграмма, линия, текст, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 15 – структура Gaussian Splatting

Первым шагом является использование метода SfM для оценки облака точек по набору изображений. Это метод оценки 3D-облака точек по набору 2D-изображений. Это можно сделать с помощью библиотеки COLMAP. Далее каждая точка переводится в гауссиану. Этого уже достаточно для растеризации. Однако из данных SfM можно сделать вывод только о положении и цвете. Чтобы изучить представление, которое дает высококачественные результаты, нам нужно его обучить. В процедуре обучения используется Stochastic Gradient Descent, похожий на нейронную сеть, но без слоев. Этапы обучения:

1. Растрируйте гауссиавы изображения с помощью дифференцируемой гауссовской растеризации (подробнее об этом позже);
2. Расчет потерь на основе разницы между растеризованным изображением и действительным;
3. Корректируютс гауссовы параметры в соответствии с потерями;
4. Применение автоматизированного уплотнения и обрезки.

В процессе исправления ошибок гауссиану клонируют (если она мелкая) или разбивают на две (если гауссиана слишком большая). Выбор между клонированием или разбитием на две точки. Новые гауссианы появляются как в местах, где других мало так и в пространстве, которое покрывает одна крупная гауссиана. Как это выглядит можно увидеть на рисунке 16 ниже.

Изображение выглядит как текст, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 16 – схема клонированием и разбитием гаусины

Качество, которое мы получим во время обучения, зависит от количества итераций, их количество может спокойно переваливать и за 7 тысяч, чтобы оптимизировать процесс обучения сначала его ведут на изображениях меньше изначальных в четыре раза по каждой из сторон. После 250 итераций даётся картинка в два раза меньше, после 500 — оригинал.

Когда распределение гауссиан стабилизируется, и разрешение изображения достигает исходного, начинается финальный этап оптимизации. На этом этапе параметры каждой гауссианы (положение, размеры, ориентация, цвет и, если требуется, альфа-прозрачность) продолжают уточняться до тех пор, пока качество получаемых изображений перестаёт заметно расти. Итоговое распределение гауссиан максимально эффективно аппроксимирует сцену, достигая компромисса между числом элементов и качеством синтеза [11].

После завершения финальной оптимизации получившееся множество гауссиан служит компактным и эффективным представлением трёхмерной сцены. Такая модель позволяет выполнять быстрый рендеринг с различных точек обзора, обеспечивая высокое качество визуализации и реалистичную передачу освещения и цвета. Важной особенностью подхода является возможность регулировать баланс между скоростью и качеством визуализации: при необходимости для ускорения воспроизведения можно уменьшать число гауссиан, незначительно жертвуя деталями.

По итогу GS интересная технология, которая решает задачу генерации трехмерных объектов с помощью гауссиан и может генерировать объекты в прямом эфире.

1. **Сбор данных и Проектирование**
   1. **Сбор данных**

Для начала работы нам потребуются объекты, а точнее фотографии объектов, по которым мы потом будем генерировать полноценные 3d модели.

Но перед тем, как мы перейдём к сбору данных, стоит выставить для них ряд требований, к самим изображениям:

1. Количество фотографий должно быть около 100 и больше, это оптимальное количество, которое рекомендуют использовать при обучении моделей;
2. Фотографии должны быть в хорошем разрешении;
3. Изображения должны перекрывать объект со всех сторон на 360. При этом желательно чтобы фотографии были сделаны с разных углов и ракурсов;
4. Также рекомендовано, что фотографии должны перекрывать друг друга для более успешной работы технологий получения более качественных результатов.

Так как с требованиями к подбору данных мы определились можно перейти к их сбору. Начнём с того какие данные нам нужны, поскольку мы хотим выявить ограничения технологий Neural Radiance Fields и Gaussian Splatting то и данные должны быть собраны под это. Первое что мне было интересно проверить насколько модели работают с отражением, для этого были в качестве объектов взяты стакан и бокал из датасета NeRFBK [12]. Ниже на рисунке 17 можно увидеть их изображение.

Изображение выглядит как стекло, контейнер, посуда, Сосуды для напитков

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 17 – Стакан и бокал

Дальше мне было интересно посмотреть, насколько хорошо технологии справляются с объектами с не очень качественной фотографиями. На этот раз я взял за объекты мавзолей [12], где все снимки с разных ракурсов, но под одним углом, и башню из датасета DNF [13] в котором с помощью дрона облетали большие постройки и делали их фотографии, где фотографии сделаны хаотично. На рисунке 18 можно увидеть их изображения.

Изображение выглядит как небо, снимок экрана, искусство, на открытом воздухе

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 18 – Башня и мавзолей

Перед тем как продолжить стоит сказать, что фотографии для объектов выше были сделаны обычными людьми с помощью телефонной камеры и дронов из-за чего распределения изображений может быть не таким качественным. Исходя из этого я решил посмотреть, насколько хорошо подобранные ракурсы и углы фотографий объекта влияют на качество моделей и за одно посмотреть какое влияние оказывает задний фон объектов. Для проверки этих идей были взяты синтетически сгенерированная металлическая деталь [12], где вручную были сгенерированы ракурсы всех камер, ваза с цветами из датасета NeRF [14] где все фотографии тщательно подобраны и имеют высокое разрешение для хорошего результата, а также для разнообразия я самостоятельно сформировал изображения игрушечной елки, предварительно записав видео на пустой площадки – облёт камеры вокруг объекты и с помощью инструментов моделирования в частности blender смог подобрать ракурсы с качественным наложением фотографий друг на друга. На рисунке 19 предоставлены фотографии.

Изображение выглядит как ваза, Украшение из цветов, Цветочная композиция, похороны

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 19 – Объекты для проверки качества

В противовес им был взят объект с простой структурой и однотонным цветом, но с большим количеством других объектов на заднем фоне и в кадре, для создания визуального шума. Он также был взят из датасета NeRFBK [12] рисунок 20.

Изображение выглядит как инжиниринг, снимок экрана, машина, в помещении

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 20 – Деталь с визуальным шумом

Таким образом я собрал данные для обучения и тестирования моделей. Следующий шагом является проектирования самого модуля на основе NeRF и GS.

**2.2 Проектирование модуля**

Модуль должен представлять из себя структуру, на вход которой подаётся папка и фотографиями объекта, и первым делом нам нужно будет восстановить трёхмерную структуру сцены по нашим изображениям и получить информация о параметрах камер (позиции и ориентации). Для этого можно использовать подход SfM, который позволяет определить, где находилась каждая камера в момент съёмки (в мировой системе координат), как она была ориентирована, и какая трёхмерная сцена присутствует перед ней. То есть, из последовательности 2D-изображений мы восстанавливаем 3D-модель и траекторию движущейся камеры. Дальше будет идти этап с трансформацией полученных данных в формат, с которым модели смогут работать, в .json данном случае.

На следующем этапе выбранная при запуске модель, NeRF или Gaussian Splatting, будет обучаться на предоставленных трансформированных данных. После этого будет проведена оценка полученных результатов, загруженная в отдельную папку и завершиться все экспортом сцены как набор гауссовских сплатов в случае гауссовым сплэттинга или в набор мешей в случаи NeRF(рисунок 21). Запуск модуля будет через CLI команду.

Изображение выглядит как линия, диаграмма, График, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 21 – Диаграмма работы модуля

1. **реализация**

Перед началом реализации нужно определиться с требованиями к вычислительной технике. Так как обе технологии довольно-таки ресурса затратные в использовании для работы был арендован сервер на базе операционной системы Linux со следующими характеристиками:

1. Процессор: Intel Core i9-14900KS 3.2 ГГц;
2. Память: 128 ГБ DDR5 NON-ECC;
3. Диск: 2 × 2000 ГБ SSD NVMe M.2
4. GPU: RTX 4090 24 ГБ GDDR6X.

Весь код будет написан на python. Первым делом нужно было получить из наших данных трёхмерную структуру сцены по нашим изображениям и получить информация о параметрах камер с помощью SfM. Для этого я воспользовался COLMAP [15] программное обеспечение с открытым исходным кодом (open-source) для 3D-реконструкции (восстановления трехмерной структуры) из фотографий. Есть несколько причин почему я выбрал COLMAP:

1. Является бесплатным в использовании;
2. Прост в использовании;
3. Показывает хорошие результаты и высокое качество работы.

из-за этого выбор пал на COLMAP. Ниже можно увидеть часть реализованного скрипта на рисунке 22.

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 22 – Скрипт запуска SfM

Feature Extraction находит характерные точки (ключевые точки, дескрипторы) на изображениях, которые помогают сопоставлять их между собой. Использует алгоритмы вроде SIFT для обнаружения ключевых точек. Для каждой точки вычисляет дескриптор — числовой вектор, описывающий её окружение. Результат: для каждого изображения получается набор ключевых точек с дескрипторами.

Feature Matching находит соответствия между ключевыми точками на разных изображениях. Сравнивает дескрипторы точек между изображениями и фильтрует ложные совпадения.

Mapper определяет положение камер для каждого изображения**.** Сначала выбирает пару изображений с хорошими совпадениями и строит начальную 3D-сцену, по совпадающим точкам определяет 3D-координаты. оптимизирует позиции камер и 3D-точек, минимизируя ошибки. Новые изображения добавляются в реконструкцию, если они хорошо согласуются с существующей сценой. Работу скрипта можно увидеть на рисунке 23 ниже.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 23 – Работа скрипта nerf\_gs\_pipeline.py

В результате мы получаем папку colmap с пред обработанными данными и информацией о расположении камер, их параметрах, координат и реконструированную сцену в пространства (рисунок 24).

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 24 – Структура папки после работы скрипта nerf\_gs\_pipeline.py

Следующим шагом идёт написание скрипта по трансформации данных в формат. json. Скрипт будет брать папку соображениями и соотносить каждую фотографию с данными, полученными с помощью COLMAP. Код реализованного скрипта можно увидеть на рисунке 25 ниже.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 25 – Скрипт трансформации в формат .json

В ходе работы скрипта в рабочую папку копируются папка с фотографиями и ещё 3 папки с изображениями меньшего разрешения в 2, 4 и 8 раз. Это делается в первую очередь, для оптимизации скорости работы, экономии памяти и реализации техник, ускоряющих обучение. Также формируется файл transforms.json. который содержит путь к изображению, а также параметры камеры - 4х4 матрица, описывающая положение и ориентацию камеры в мире. Ниже на рисунке 26 можно увидеть новую структуру рабочей папки.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 26 – Содержимое файла transform.json

Дальше идёт реализация скрипта по обучению моделей. Для обучения была взята реализация моделей NeRF и GS из фрэймворка nerfstudio [16] – nerfacto и splatfacto. Такой выбор обусловлен тем, что даже при имеющихся вычислительных ресурсах модель NeRF все равно обучается 1–2 из-за этого дабы ускорить процесс обучения была и взята модель nerfacto, которая использует hash-грид кодировку, что позволяет сократить время обучения до нескольких часов, но даже при этом есть ограничения использования модели из-за нехватки вычислительных ресурсов. На вход модели идет путь к рабочей папке с нашими подготовленными и обработанными данными и параметры обучения. На выход в папке outputs создается папка с конфиг файлом модели Реализованный скрипт и работу модели можно увидеть на рисунке 27 и 28 ниже.

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 27 – Скрипт обучения модели

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 28 – Работа скрипта train\_nerfstudio\_model.py

После обучения нужно оценить качество работы модели и сделать экспорт обученной сцены. В качестве метрик были выбраны PSNR, SSIM, LPIPS.

PSNR(пиковое отношение сигнал/шум) — это измерение качества восстановленного изображения по сравнению с эталонным, основанное на средней квадратичной ошибке (MSE). Формула

, (3.8)

где — максимальное возможное значение пикселя изображения

SSIM(показатель структурного сходства) — это измерение, оценивающее структурное сходство между двумя изображениями, принимающее во внимание яркость, контраст и структуру. Формула

, (3.9)

где:

* где: μx, μy — средние значения изображений x и y;
* σx2, σy2 — дисперсии;
* σxy — ковариация между x и y;
* C1 и C2 — малые константы для избежания деления на ноль.

LPIPS — современная метрика, основанная на сравнении активаций известных сверточных нейросетей, отражающая перцептивное сходство между изображениями. Измеряет сходство между элементами двух изображений. Формула

, (3.10)

где:

* fl(x), fl(y) — активации на слое l для изображений x и y,;
* wl — обучаемые веса для слоя l;
* H, W — высота и ширина карты признаков.

В случае с экспортом данных, если моделью является NeRF то извлекается mesh, если GS извлекается набор гауссовских сплатов. Реализованные скрипты предоставлены на рисунках 29 и 30 ниже.

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 29 – Скрипт по оценки модели

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 30 – Скрипт по экспорту сцены

После исполнения данных скриптов мы получаем файл с метриками и экспортированную сцену, пример результатов работы предоставлены ниже на рисунке 31 и 32.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 31 – Пример полученных метрик

Изображение выглядит как безалкогольный напиток, серебряный, земля, цилиндр

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 32 – Пример результата работы nerfacto

Последние что осталось это реализовать управляющий скрипт, который будет запускать последовательно все другие, что позволит проделать всю работу одной командой. Скрипт будет запускаться CLI командой в консоли

python run\_all\_pipeline.py --images images\_dict/images\_project1 --workspace dir --model nerfacto/splatfacto --extra --max-num-iterations 50000 где:

* --images – папка с изображениями;
* --workspace – рабочая директория;
* --model – выбранная для обучения модель nerfacto/splatfacto;
* --extra – можно указать дополнительные параметры обучения, например максимальное количество итераций.

внизу на рисунке 33 представлена реализация управляющего скрипта.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 33 – Управляющий скрипт

В итоге был реализован модуль для генерации трёхмерных объектов с помощью NeRF и GS. Данные использованные для обучения, код и результаты опубликованы на github [17] в открытом доступе.

1. **Оценка и анализ полученных результатов**

Результаты метрик по генерации трехмерных объектов с помощью Nerf и GS, обученных на различные объектах показаны в таблице 4.1. Каждое обучение было проведено с 100000 количества итераций.

Таблица 4.1 - Результаты работы моделей NeRF и GS

|  |  |  |  |
| --- | --- | --- | --- |
| Объект | PSNR | SSIM | LPIPS |
| 1 | 2 | 3 | 4 |
| tree\_gs | 30.52 | 0.92 | 0.07 |
| tree\_nerf | 21.78 | 0.54 | 0.19 |
| mausoleum\_gs | 24.36 | 0.78 | 0.19 |
| mausoleum\_nerf | 21.57 | 0.61 | 0.43 |
| detail\_gs | 29.03 | 0.95 | 0.03 |
| detail\_nerf | 17.17 | 0.64 | 0.10 |
| flowers\_gs | 31.6 | 0.91 | 0.11 |
| flowers\_nerf | 23.18 | 0.73 | 0.16 |
| cup\_gs | 26.42 | 0.90 | 0.07 |
| cup\_nerf | 17.8 | 0.66 | 0.15 |
| glass\_gs | 27.17 | 0.84 | 0.05 |
| glass\_nerf | 17.48 | 0.61 | 0.12 |
| metallic\_detail\_gs | 37.72 | 0.98 | 0.02 |
| metallic\_detail\_nerf | 21.25 | 0.72 | 0.12 |
| 1 | 2 | 3 | 4 |
| tower\_gs | 22.16 | 0.72 | 0.29 |
| tower\_nerf | 17.41 | 0.48 | 0.51 |
| Mean\_gs | 28.62 | 0.87 | 0.10 |
| Mean\_nerf | 19.71 | 0.62 | 0.22 |

Как можно увидеть из таблицы выше GS показывает куда более высокие результаты чем NeRF и обучается он гораздо быстрее по сравнению со своим коллегой. Можно заметить, что обе модели проседают при работе с объектами мавзолей и башня, причем гауссов сплэттинг делает сильный скачок вниз по качеству. При этом при работе с праздничным деревом, цветком в вазе и синтетически сгенерированной металлической деталью обе модели показывают лучшие результаты, при чем в этом случае гауссов сплэттинг наоборот делает скачок вверх. Из этого можно сделать вывод, что обеих технологий качество зависит от хорошо подобранных ракурсов и углов фотографий, причем GS намного сильнее проседает при плохом подборе изображений.

Также можно увидеть, что NeRF чувствителен к заднему фону и большому количеству объектов на нем, в то время как GS показывает стабильно хорошие результаты на таких объектах

Стоит поговорить и об отражении, обе технологии имеют с ним проблемы. Хоть и на первый взгляд кажется, что результат не такой плохой по метрикам, но если посмотреть на полученные модели, невооруженным взглядом станет ясно, что технологии плохо справляются с этим рисунки 34 и 35 ниже.

Изображение выглядит как текст, искусство, в помещении, стекло

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 34 – Результат работы моделей со стаканом

Изображение выглядит как стол, в помещении

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 35 – Результат работы моделей с бокалом

Подводя итог GS выгляди более выигрышно по сравнению с NeRF и показывает более высокие результаты за меньшие время, но при этом NeRF показывает более стабильные результаты. Обе технологии нуждаются в более качественном подборе изображений для обучения, GS имеет большую чувствительность к этому. Также NeRF более восприимчива большому количеству лишних объектов и загруженному заднему фону.

**Заключение**

В рамках выполнения данной работы были достигнуты следующие результаты:

1. Была изучена предметная область генерации трёхмерных объектов, был проведен обзор существующих методов и моделей.
2. После проведения сравнения были обозначены основные ограничения и недостатки технологий. В частности, ограничение работы технологий с отражениями, зависимость от подбора данных, влияние заднего фона и объектов на нем на модели;
3. Проведен анализ существующих датасетов. Были использованы как независимые датасеты, так и датасет предоставленный самими разработчиками технологий, а также были использованы собственноручно собранные данные;
4. Реализован модуль для генерации трёхмерных объектов на основе NeRF и GS;
5. Код реализованного модуля выложен на github в открытом доступе.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Guikun Chen, Wenguan Wang. A Survey on 3D Gaussian Splatting // arXiv: электрон. науч. архив. – 2025. – URL: https://arxiv.org/pdf/2401.03890 (дата обращения: 05.03.2025).
2. Статистика по теме NeRF на Github [Электронный ресурс]. – 2025. –URL: https://repositorystats.com/topic/nerf (дата обращения: 06.03.2025).
3. Обзор решений на основе 3D Gaussian Splatting [Электронный ресурс. – 2024. – URL: https://habr.com/ru/articles/786064/ (дата обращения: 12.02.2025).
4. Abdullah Bulbul, Tolga Capin, Guillaume Lavoue, Marius Preda. Assessing Visual Quality of 3D Polygonal Models // IEEE Signal Processing Magazine. – 2021. – URL: https://www.researchgate.net/publication/224258658\_Assessing\_Visual\_Quality\_of\_3-D\_Polygonal\_Models (дата обращения: 05.03.2025).
5. Mitko Aleksandrov, Sisi Zlatanova, David J. Heslop. Voxelisation Algorithms and Data Structures: A Review // Open Access Journals Sensor. – 2021. – URL: https://www.researchgate.net/publication/356924910\_Voxelisation\_Algorithms\_and\_Data\_Structures\_A\_Review (дата обращения: 05.03.2025).
6. Huang Zhang, Changshuo Wang, Shengwei Tian, Baoli Lu, Liping Zhang, Xin Ning, Xiao Bai. Deep Learning-based 3D Point Cloud Classification: A Systematic Survey and Outlook // arXiv: электрон. науч. архив. – 2023. – URL: https://arxiv.org/pdf/2311.02608 (дата обращения: 06.03.2025).
7. Christian Hane, Shubham Tulsiani, Jitendra Malik. Hierarchical Surface Prediction for 3D Object Reconstruction // arXiv: электрон. науч. архив. – 2017. – URL: https://arxiv.org/pdf/1704.00710 (дата обращения: 07.03.2025).
8. Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, Hujun Bao. NeuralRecon: Real-Time Coherent 3D Reconstruction from Monocular Video // CVPR 2021. – 2021. – URL: https://arxiv.org/pdf/2104.00681 (дата обращения: 08.03.2025).
9. Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, Angjoo Kanazawa. Plenoxels: Radiance Fields without Neural Networks // CVPR 2022. – 2022. – URL: https://arxiv.org/pdf/2112.05131 (дата обращения: 08.03.2025).
10. Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis // ECCV 2020. – 2020. – URL: https://arxiv.org/pdf/2003.08934 (дата обращения: 05.04.2025).
11. Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering// SIGGRAPH 2023. – 2023. – URL: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/ (дата обращения: 05.04.2025).
12. Датасет NeRFBK [Электронный ресурс]. - URL: https://github.com/3DOM-FBK/NeRFBK?tab=readme-ov-file (дата обращения 21.04.25).
13. Датасет DNF [Электронный ресурс]. - URL: https://github.com/nickponline/dd-nerf-dataset?tab=readme-ov-file (дата обращения 21.04.25).
14. Датасет NeRF [Электронный ресурс]. - URL: https://drive.google.com/drive/folders/1cK3UDIJqKAAm7zyrxRYVFJ0BRMgrwhh4 (дата обращения 21.04.25).
15. Документация ПО COLMAP[Электронный ресурс]. - URL: https://colmap.github.io/ (дата обращения: 25.04.2025).
16. Документация фрэймворка nerfstudio[Электронный ресурс]. - URL: https://docs.nerf.studio/index.html (дата обращения: 25.04.2025).
17. NeRF\_GS\_module [Электронный ресурс]. - URL: https://github.com/EternalVelard/NeRF\_GS\_module/tree/main (дата обращения: 15.05.2025).