Научная визуализация, 2025, том 17, номер 1, страницы 65 - 85, DOI: 10.26583/sv.17.1.06

Применение инструментов компьютерного зрения PyTorch3D и NERF для построения облака точек трехмерной модели и определения положения камеры фотоснимков в пространстве

В.В. Коньков[1], А.Б. Замчалов[2]

Институт интеллектуальных кибернетических систем, Национальный исследовательский ядерный университет "МИФИ", Москва, Россия

[1] ORCID: 0009-0005-1197-2248, vlad.konkov.7145@gmail.com
[2] ORCID: 0009-0006-0955-1062, andreizam@yandex.ru

Аннотация

В последнее время компьютерная графика играет ключевую роль в решении задач компьютерного зрения. Проблема преобразования 2D изображений в 3D модели продолжает оставаться актуальной, так как требует точного определения положения камеры и построения точных трёхмерных моделей объектов. Традиционные методы зачастую ограничены в применении и не предлагают комплексного решения. В данном исследовании рассматривается использование библиотек PyTorch3D и NERF для определения положения камеры в 3D пространстве и создания трёхмерной модели объекта по одному 2D изображению. В качестве метода подготовки данных был использован аппаратно-программный комплекс, включающий устройство для управления шаговым двигателем, обеспечивающее ручное и последовательное позиционирование камеры и её возврат в исходное положение, систему управления съёмкой для формирования комплексного набора фотоснимков на каждой позиции камеры, и механизм отправки данных на удалённый компьютер для дальнейшей обработки. В ходе исследования была выбрана библиотека PyTorch3D для изучения возможностей преобразования 2D изображений в 3D модели или определения положения объекта на фотоснимках. Процесс обработки включал в себя несколько шагов: построение облака точек для генерации объёмной 3D модели объекта, определение положения камеры в 3D пространстве по одному 2D изображению с использованием алгоритмов обратной задачи, а также построение 3D объекта с помощью дифференцируемой отрисовки, создание 3D вокселей и 3D мешей. Результаты исследования показали успешное определение положения камеры в 3D пространстве и построение трёхмерной модели объекта по одному 2D изображению, что демонстрирует преимущества использования библиотеки PyTorch3D по сравнению с другими существующими моделями. Эти данные могут быть применены в разработке программных и аппаратных систем для создания трёхмерных изображений на основе 2D фотографий. Исследование подтвердило актуальность и эффективность применения библиотеки PyTorch3D для решения задач преобразования 2D изображений в 3D модели. В дальнейшем работа будет направлена на расширение функциональных возможностей системы и её использование в различных областях компьютерного зрения.

Ключевые слова: компьютерное зрение; PyTorch3D; NERF; 3D-моделирование; позиционирование камеры, облако точек; глубокое обучение; 3D-реконструкция; дифференцированный рендеринг.

1. Введение

Компьютерное зрение является одной из наиболее динамично развивающихся областей искусственного интеллекта, находя применение во множестве сфер от медицинской диагностики до автономного вождения автомобилей. В последние годы проблема преобразования двухмерных (2D) изображений в трёхмерные (3D) модели стала особенно актуальной. Это связано с тем, что трёхмерные модели позволяют более точно анализировать и интерпретировать окружающую среду. В этом контексте определение точного положения камеры и построение высококачественных трёхмерных моделей объектов являются ключевыми задачами. Традиционные методы, используемые для этих целей, нередко сталкиваются с ограничениями, связанными с точностью и масштабируемостью.

Введение в эксплуатацию новых технологий и подходов, таких как библиотека PyTorch3D, открывает новые возможности для решения указанных проблем. Данное исследование направлено на изучение потенциала использования PyTorch3D для определения положения камеры и создания трёхмерных моделей объектов на основе одного двухмерного изображения. В рамках работы был использован аппаратно-программный комплекс, включая управление шаговым двигателем для точного позиционирования камеры, систему управления съёмкой и механизм передачи данных на удалённый сервер для дальнейшей обработки.

Основной задачей данного исследования является оценка точности и эффективности предложенного метода в сравнении с традиционными подходами, а также выявление потенциальных областей применения разработанного решения.

Цель работы: исследование методов обработки 2D и 3D изображений с использованием возможностей библиотеки PyTorch3D.

Дополнительно были решены следующие задачи:

В части подготовки исходных данных (фотографий) объекта без использования синтетических датасетов:

- задача управления шаговым двигателем с возможностью ручного и последовательного позиционирования камеры возле объекта съемки и возврата камеры в исходное положение;
- задача управления съемкой для формирования комплексного набора исходных данных (фотографий) на каждой позиции камеры возле объекта для последующей обработки;
- задача отправки на удаленный компьютер для обработки;
- Генерация .obj модели объекта.

В части определения объекта на 2D снимке на основе OBJ модели с помощью PyTorch3D (обратная задача):

- построение облака точек для генерации объемной 3D модели;
- определение положения камеры в 3D-пространстве по одной фотографии;
- определение положения 3D объекта на фотографии с использованием дифференцируемой отрисовки;

2. Литературный обзор

В последние годы наблюдается значительный интерес к разработке методов компьютерного зрения, используемых для анализа пространственного расположения камеры на изображениях с целью создания трехмерных моделей. В растущем числе исследований акцент сделан на использовании PyTorch3D, библиотеки, предоставляющей инструменты для трехмерного моделирования в компьютерном зрении. Эти инструменты позволяют обрабатывать и интегрировать данные, полученные из фотографий, в точные трехмерные облака точек. Несмотря на то, что существует значительное количество литературы по методам компьютерного зрения,

использование PyTorch3D для точной регистрации положения камеры на фотографиях было менее широко исследовано. Этот обзор направлен на анализ существующих исследований, посвященных данной теме, и выявление потенциала PyTorch3D в создании точных трехмерных моделей.

В области компьютерного зрения, особенно касающейся Neural Radiance Fields (NeRF), авторы работы [1] предложили Depth-supervised Neural Radiance Fields (DS-NeRF) для решения задач получения точных геометрий с ограниченным количеством входных изображений. Реализовав функцию потерь, руководствуемую глубиной, которая использует данные, полученные из структуры движения, DS-NeRF не только улучшает качество изображения, но и ускоряет обучение. [2]

Продвигая разработки в области 3D-технологий, авторы работы [3] разработали PyTorch3D, инструментальный набор, предназначенный для обработки сложных 3D-данных и содействующий эффективным дифференцируемым графическим операциям. Этот набор инструментов поддерживает различные приложения, от автономного вождения до виртуальной реальности, подчеркивая продолжающуюся эволюцию в глубоком обучении 3D и потенциал улучшения методами, усиленными NeRF.

Рассматривая сохранение культурного наследия, где традиционные методы часто сталкиваются с препятствиями из-за дорогостоящего оборудования, необходимого для высококачественного 3D-сканирования, авторы работы [1] также предложили экономичное решение с использованием PyTorch3D, предназначенное для реставраторов художественных музеев. Этот подход позволяет обнаруживать мелкие детали поверхности произведений искусства с помощью таких доступных устройств, как планшеты, демонстрируя улучшенные возможности PyTorch3D в точном позиционировании камеры, необходимом для детализированной визуализации. Эта интеграция не только повышает точность и доступность высококачественных технологий 3D-моделирования, но и подчеркивает значимость PyTorch3D как потенциально ключевого инструмента для 3D-реконструкции в контекстах, требующих точности при ограниченных ресурсах.

Рассматривая эти достижения и принимая во внимание дополнительные исследования, такие как также результаты авторов [4], которые изучают различные методы 3D-реконструкции, включая NeRF, складывается комплексное повествование. В частности, авторы работы предоставляют доказательства того, что NeRF превосходит традиционные методы фотограмметрии в обработке нетекстурированных объектов или отражающих поверхностей. Это открытие предполагает, что будущие применения PyTorch3D могут выиграть от включения методологий NeRF для повышения качества реконструкций.

На основе описанного в исследованиях [1] и [2], авторы работы [5] представили модифицированную модель NeRF−−, которая устраняет необходимость в известных параметрах камеры в настройках NeRF. Эта модель подчеркивает простоту в рендеринге трехмерных сцен и расширяет доступность. Их метод, проверенный с помощью Blender Forward-Facing Dataset, демонстрирует высококачественный синтез и динамическую оптимизацию параметров камеры во время обучения.

В текущих методологиях, используемых для создания новых видов сцен из одного изображения, наблюдаются постоянные вызовы, особенно в плане достижения реализма без обширных 3D-данных. Противостоя этим препятствиям, авторы работы [6] разработали новаторскую модель, которая способна синтезировать новые перспективы сцены всего лишь из одного входного изображения. Значимость этой модели в области визуализации и 3D-реконструкции подчеркивается тем, что она обучается непосредственно на реальных изображениях и функционирует независимо от заранее заданных 3D-данных. В основе её функциональности лежит дифференцируемый визуализатор облака точек, который эффективно преобразует скрытые 3D-облака точек в требуемую точку обзора. Кроме того, она включает в себя

сеть уточнения, которая декодирует проецируемые особенности для устранения пробелов и создания более реалистичных изображений.

Авторы работы [6] представили инновационный подход, используя прогрессивный рендеринг для эффективной работы с облаками точек, что дает большую гибкость и масштабируемость по сравнению с традиционной фотограмметрией или техникой, обсуждаемой в [4]. Эта гибкость упрощает визуализацию сцен в высоком разрешении, открывая новые возможности в области анимации и интерактивных приложений дополненной и виртуальной реальности. Модель Synsin, описанная в их исследовании, связана с достижениями в области технологий нейронных полей излучения (NeRF), что отмечено авторами [5] и сокращает зависимость от заранее известных параметров камеры. Кроме того, метод [6] находит отклики в инициативах PyTorch3D, упомянутых в [3] и [1], которые нацелены на упрощение 3D-реконструкции и визуализации при минимальных начальных данных.

Исследование [6] знаменует собой значительный шаг вперед в разработке алгоритмов 3D-рендеринга и реконструкции, отражая тенденции, соответствующие применению NeRF и PyTorch3D в самых разных областях: от музейных выставок до промышленной химии. Интеграция методов Synsin, упомянутых в предыдущих статьях, может открыть новые возможности для автоматизации визуального контента, повышения интерактивности и повышения точности 3D-моделей. Такое постоянное технологическое развитие открывает новые горизонты для понимания и воспроизведения 3D-пространства из минимального количества входных данных, добиваясь при этом максимального реализма в создаваемых сценах.

Несмотря на эти достижения в 3D-моделировании с ограниченными данными, возникает широкий спектр вопросов: могут ли такие техники, как NeRF и Synsin, обсуждаемые в [5] и [6], стабильно предоставлять точные и реалистичные модели в различных условиях. Эти методы значительно расширяют применения 3D-технологий, однако постоянство их результатов в разных условиях остается спорным. Более того, такие технологии, как PyTorch3D, на которые указывают [3] и [1], раздвигают границы быстрого и экономичного 3D-сканирования и анализа материалов. Тем не менее, их производительность в различных эксплуатационных средах по-прежнему окружена неопределенностью. Исследования [4] и [7] предполагают настоятельную необходимость дальнейших исследований для разработки новых методологий, которые улучшат точность и реалистичность 3D-реконструкций. Хотя шаги были сделаны, проблема остается недостаточно решенной. Кроме того, эти инновационные подходы не только облегчают реконструкцию сцен, но и поддерживают манипулирование и анимацию объектов в реальном времени в 3D-пространстве, что представляет собой постоянные исследовательские вызовы относительно осуществимости и надежности этих методов в практических сценариях.

В заключение, несмотря на значительные достижения в компьютерном зрении и 3D-реконструкции, многие вопросы, касающиеся осуществимости, надежности и применения этих технологий в различных условиях, остаются нерешенными. Это подчеркивает продолжающуюся необходимость в дальнейших инновациях и критической оценке этих методов в различных областях.

3. Методы обработки данных

В результате обзора литературы были определены следующие инструменты для обработки данных:

3.1 PyTorch3D

PyTorch3D — это библиотека для работы с 3D-данными, разработанная компанией Facebook AI Research (FAIR). PyTorch3D является популярным выбором по нескольким причинам:

- PyTorch3D тесно интегрирован с PyTorch, что позволяет использовать все преимущества этого популярного фреймворка машинного обучения. Это особенно полезно для разработчиков, которые уже знакомы с PyTorch и хотят добавить обработку 3D-данных в свои проекты.

- PyTorch3D поддерживает дифференцируемую отрисовку, что позволяет использовать методы оптимизации, такие как градиентный спуск, для задач, связанных с 3D. [8]

Дифференцируемая отрисовка является важным инструментом для задач, связанных с восстановлением 3D-моделей из 2D-изображений, как и другие операции, которые могут потребовать градиентного спуска или других методов оптимизации. Когда мы формируем изображение, это можно представить как процесс отображения 3D-моделей в 2D-плоскость. Например, если у нас есть две 3D-сферы, их положение в пространстве определяет, как они будут выглядеть на 2D-изображении. Разные конфигурации сфер в 3D приведут к разным 2D-картинкам.

Многие задачи 3D-компьютерного зрения требуют обратной задачи: реконструкции 3D-структур из 2D-изображений. Например, если у вас есть 2D-изображение сцены, вам необходимо определить 3D-структуру объекта, чтобы получить такое изображение. Решение этой проблемы можно рассматривать как задачу оптимизации, где переменные являются параметрами, описывающими 3D-объекты (например, расположение центра сферы). Мы ищем значения этих параметров такие, чтобы создаваемое изображение было максимально близко к заданному 2D-изображению. [1]

Для этого мы определяем функцию стоимости, которая измеряет, насколько сгенерированное изображение похоже на реальное. В этом случае, например, можно использовать среднеквадратическую ошибку на пиксель. Далее нам нужно рассчитать градиент этой функции стоимости относительно параметров 3D-объекта. Знание этих градиентов позволяет нам итеративно изменять параметры в направлении, которое снижает стоимость, что в конечном итоге приводит к лучшему сопоставлению изображений. [8]

Для этого функция, которая отображает параметры в функцию стоимости, должна быть дифференцируемой. Это означает, что весь процесс рендеринга должен быть дифференцируемым. Это позволяет использовать методы оптимизации, такие как градиентный спуск, для эффективного поиска параметров, которые минимизируют функцию стоимости и обеспечивают наилучшее соответствие между 3D-моделью и 2D-изображением. [8]

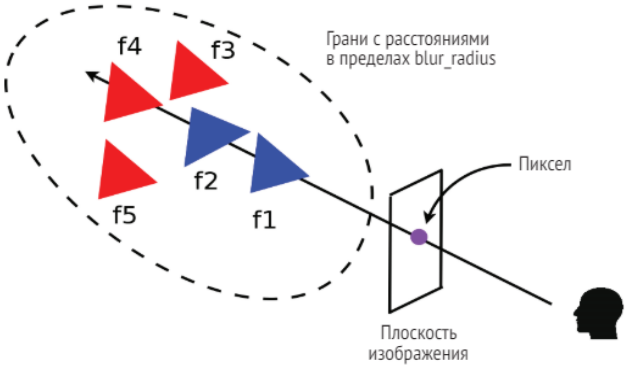

Рис. 1. Дифференцируемая отрисовка путем взвешенного усреднения всех подходящих граней полигональной сетки

Традиционные алгоритмы растрирования сталкиваются с проблемами при расчете цветовых градиентов. Основная причина заключается в том, что процесс проверки является отдельным процессом. В нем для каждого пикселя изображения создается луч, который проходит через пиксель и пересекается с различными полигонами 3D-сцены. Поскольку этот алгоритм выбирает только ближайшую к камере область сетки,

процесс по сути является ступенчатой функцией и, следовательно, не дифференцируем.

PyTorch3D решает эту проблему, используя principles описанные в подходе "Мягкий растеризатор" (Soft Rasterizer) [9]. Главная идея заключается в том, чтобы сделать процесс растеризации «мягким», позволяя учитывать несколько потенциально подходящих граней полигональной сетки при определении цвета каждого пикселя.

Вместо того чтобы выбирать одну ближайшую грань, рассматриваются все грани, расстояние до которых от луча меньше некоторого порога. В PyTorch3D этот порог настраивается через атрибут blur_radius в RasterizationSettings.

Для каждой выбранной грани рассчитывается вероятность, представляющая вероятность пересечения данной грани с лучом. Формула включает гиперпараметр, который контролирует "размытие". В PyTorch3D этот параметр можно настроить через BlendParams.sigma. [8] Далее отрисовщик должен вычислить вероятность, ассоциируемую с каждой гранью полигональной сетки, как показано ниже (1). Здесь dist представляет собой расстояние между гранью и лучом, а σ (сигма) является гиперпараметром. Проще говоря, вероятность для каждой грани обозначает вероятность того, что эта грань полигональной сетки покрывает определенный пиксель изображения. При этом расстояние может быть отрицательным, если луч пересекает грань полигональной сетки.

$$D_{j=sigmoid\,(\frac{-dist_j}{\sigma})} \tag{1}$$

Вес, w_b, – это малый вес цвета фона. Здесь параметр γ (гамма) – это гиперпараметр.

$$W_j = D_j \exp\left(\frac{z_j}{\gamma}\right); W_b = \exp\left(\frac{\epsilon}{\gamma}\right); \tag{2}$$

Таким образом, окончательный цвет пиксела можно определить с помощью следующего ниже уравнения

$$I = \frac{(\sum_j w_j c_j) + w_b c_b}{(\sum_j w_j) + w_b} \tag{3}$$

При реализации дифференцируемой отрисовки в библиотеке PyTorch3D для каждого пикселя изображения дополнительно вычисляется значение α (альфа). Данное значение представляет вероятность того, что пиксель изображения находится на переднем плане и что луч пересекает хотя бы одну грань полигональной сетки. [8]

В мягком растеризаторе значение α вычисляется из соответствующих граням вероятностей, как показано ниже.

$$a = 1 - \prod_J (1 - Dj) \tag{4}$$

Цвет пикселя определяется средневзвешенными значениями затенения всех выбранных граней, причем веса зависят от рассчитанных вероятностей.

Дифференцируемая отрисовка значительно расширяет возможности компьютерного зрения и машинного обучения, позволяя эффективно использовать методы оптимизации, такие как градиентный спуск, для решения сложных задач, включая восстановление 3D-структур из 2D-изображений. PyTorch3D реализует дифференцируемую отрисовку, используя подход мягкой растеризации, который делает процесс плавным и математически дифференцируемым. [8]

С помощью дифференцируемого рендернига PyTorch3D можно решить задачу оценивания поз объекта. Задача будет заключаться в оценивании позы объекта по одному единственному изображению, полученному в результате наблюдения. В дополнение к этому мы будем исходить из допущения, что у нас трехмерная сеточная модель объекта.

Дополнительно PyTorch3D предлагает множество модулей для работы с различными аспектами 3D-данных, включая рендеринг, деформацию сеток, реконструкцию 3D-моделей и т.д. Это позволяет разработчикам легко собирать собственные пайплайны обработки. А также поддерживает работу с мешами

(полигональными сетками), точечными облаками и вокселями, что делает её универсальной для различных приложений.

- Пиксели в 2D используются для рендеринга плоских изображений или спрайтов в 3D-сценах. Это полезно для создания иллюзии глубины при минимальных вычислительных затратах.
- Полигоны (Треугольники и Квадраты) используются для моделирования поверхностей 3D-объектов. Такой подход позволяет создавать сложные фигуры с высокой детализацией при сравнительно небольшом объеме данных.
- Воксели: используются для моделирования объемных данных, включая внутреннюю структуру объектов. Воксельные модели состоят из множества маленьких кубиков, заполняющих 3D-пространство.

Воксели, сокращение от "volume element" (объемный элемент), являются аналогом пикселей в трехмерном пространстве. Если пиксели представляют собой 2D-элементы изображения, то воксели — это объемные кубические элементы, которые заполняют 3D-пространство. Воксельные представления часто используются для визуализации объемных данных, например, в медицинской визуализации или в игровых приложениях. [8]

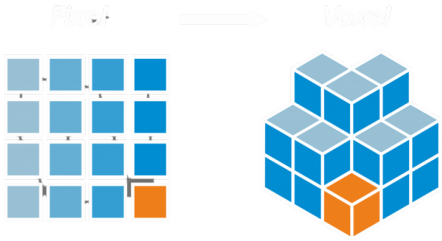

Рис. 2. Представление вокселей

Воксели могут представлять не только поверхность объекта, но и его внутреннюю структуру, а также воксели легко дискретизировать и обрабатывать, что делает их удобными для алгоритмов объемной визуализации. Но воксельные представления могут занимать много памяти, особенно для больших разрешений и для сохранения деталей требуется огромное количество вокселей, что делает такие представления неэффективными для высокодетализированных моделей. [8]

Дополнительно в исследовании используются файлы OBJ и MTL. Файл OBJ и связанный с ним файл MTL часто используются для хранения 3D-моделей:

- OBJ файл описывает геометрию объекта. Он содержит список вершин (точек) и плоскостей (граней), образующих полигональную сетку. Это один из наиболее распространенных форматов для 3D-моделей из-за своей простоты и поддержки различными программами для 3D-моделирования.
- MTL файл содержит информацию о материалах, таких как цвет, текстуры, отражательные свойства и т.д. MTL-файл обычно ссылается в OBJ-файле и позволяет назначить материалы полигонам.

3.2 Neural Radiance Fields

NeRF — это метод, использующий нейронные сети для представления и рендеринга 3D-объектов на основе серии 2D-изображений. Основная идея заключается в представлении непрерывной сцены с помощью функции плотности и цвета в 5-мерном пространстве, которое интерпретирует видимую сцену в трех пространственных измерениях и двух угловых измерениях. [4]

Концепция нейронных полей яркости излучения (NeRF) представляет собой подход к моделированию 3D-сцен с использованием нейронных сетей. Основная суть задачи

заключается в синтезировании новых ракурсов сцены на основе ограниченного числа доступных 2D-снимков. Это является непростой задачей, поскольку необходимо учитывать множество факторов, таких как артефакты объектов, источники света, отражения, непрозрачность материалов, текстура поверхностей и загораживания другими объектами. Эти аспекты могут существенно влиять на итоговое изображение при смене ракурса. [5]

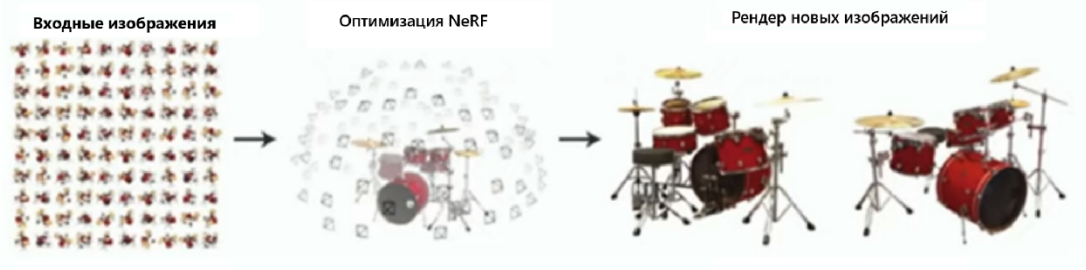

Рис. 3. Алгоритм работы NeRF

Для успешного применения NeRF (Neural Radiance Fields) требуются специфичные входные данные, которые включают изображения сцены, параметры камер (внутренние и внешние), а также, возможно, маски или другие вспомогательные данные. [8] Ниже приведено, как структуры этих данных обычно организованы.

Изображения (images) - это могут быть цветные или черно-белые изображения сцены, выполненные с различных ракурсов. Обычно они представлены в формате PNG или JPEG.

- Формат: .png или .jpg
- Содержимое: набор изображений сцены, каждое из которых имеет свое уникальное имя.

Внутренние параметры камеры (intrinsics) - эти параметры описывают внутреннюю калибровку камеры, такую как фокусное расстояние, коэффициенты искажений и координаты оптического центра.

- Формат файла: текстовый файл (например, intrinsics.txt)
- Содержимое: может включать фокусное расстояние (fx, fy), координаты главной точки (cx, cy) и параметры дисторсии, если они есть.

Метод NeRF использует нейронные сети для глубокой и точной моделировки сцены, применяя неконвенциональный подход. Этот метод был предложен исследовательской группой из Калифорнийского университета в Беркли, Google Research и Калифорнийского университета в Сан-Диего. [8] Благодаря такому уникальному использованию нейронных сетей и высокой точности усваиваемых моделей, метод NeRF выпустил на свет ряд новых изобретений в области синтеза изображений, определения глубины и 3D-реконструкции. Таким образом, освоение этой концепции становится критически важным для дальнейшего изучения указанных тем.

Вся "физика" сводится к формуле объемного рендеринга:

$$C(r) = \int_{t_n}^{t_f} T(t)\sigma\big(r(t)\big)c(r(t), d)dt, \text{ где } T(t) = \exp\left(-\int_{t_n}^{t} \sigma\big(r(s)\big)ds\right) \quad (5)$$

здесь (C) - результирующий цвет луча, (T) -коэффициент пропускания, (c) -локальный цвет, (σ) -его плотность, (r) -координата на луче, а(d) -направление.

NERF кодирует непрерывную объемную функцию что дает хорошее качество и требует мало места для хранения.

$$5D(x, y, z, theta, phi) \rightarrow 3D(r, g, b) \quad (6)$$

Поле яркости излучения, лежащее в основе NeRF, описывает распределение света в 3D-пространстве и позволяет детально моделировать взаимодействия света с объектами сцены. Нейронные сети служат для улавливания и представления этой сложной информации, что делает NeRF способным справляться с неполной и зашумленной информацией, создавая точное представление сцены. [8] Эта концепция

предлагает мощный инструмент для решения задач синтеза новых ракурсов и моделирования сложных 3D-сцен, что делает её важной для всех специалистов, работающих в области компьютерного зрения.

Поле яркости излучения описывает распределение световой энергии в пространстве и времени. Под яркостью излучения понимается интенсивность света в данной точке пространства при наблюдении в определённом направлении. Этот параметр измеряется в единицах силы света на единицу площади на единицу телесного угла. В контексте компьютерного зрения и графики, яркость излучения чаще всего представляется в системе RGB, где три компоненты (красный, зелёный и синий) описывают цветовую информацию. [8]

Важно понимать, что яркость излучения определяется рядом факторов. Во-первых, к этим факторам относятся источники света, которые освещают точку в пространстве. Положение, мощность и цвет источников света влияют на то, как освещается и воспринимается сцена. Во-вторых, это наличие поверхностей или объёмов, которые могут изменять направление, отражать или поглощать свет. Эти объекты могут создавать тени, блики или диффузные отражения, внося свой вклад в общую картину яркости излучения. В-третьих, текстура и материал поверхности также играют важную роль: гладкие или блестящие поверхности отразят свет иначе, чем грубые или матовые поверхности.

Поле яркости излучения является ключевой концепцией, лежащей в основе методов моделирования и рендеринга 3D-сцен, таких как NeRF. Оно позволяет нам моделировать, как распространяется свет в сложной сцене, учитывая взаимодействия света с объектами и материалами. Сохранение и использование этой информации позволяет создавать фотореалистичные изображения и точные реконструкции трёхмерных объектов, что является целью таких методов.

Представление полей яркости излучения с помощью нейронных сетей, таких как NeRF (Neural Radiance Fields), глубоко меняет подход к задачам 3D-воссоздания сцены и её рендеринга. В отличие от традиционных методов, NeRF использует нейронные сети не для классификации или генерирования изображений напрямую, а для представления сцены в форме объемной функции. [8] Ниже приводится краткое описание этого подхода.

NeRF использует многослойный персептрон (MLP), который по сути является полносвязной нейронной сетью, для моделирования сцены. Эта сеть не является сверточной нейронной сетью (CNN), что заметно выделяет её на фоне большинства современных моделей в компьютерном зрении.

Входные данные сети включают пять координат:

- Три пространственные координаты (x, y, z) для определения точки в 3D пространстве.
- Два угла обзора (θ, Ø), которые могут быть конвертированы в единичный вектор направления d в декартовой системе координат. Это помогает задать направление, под которым наблюдается точка.

Для каждой точки и направления сеть предсказывает:

- Объемную плотность σ в рассматриваемой точке (x, y, z), которая определяет, сколько света может поглотить или рассеять данный объем.
- Цвет (r, g, b), который определяет цвет испущенного света в этой точке при данном направлении обзора (θ, Ø). Этот цвет служит косвенным индикатором яркости излучения.

Каждая входная точка является 5-мерным вектором. Было обнаружено, что тренировка модели непосредственно на этих входных данных плохо справляется с представлением высокочастотных вариаций цвета и геометрии. Это объясняется тем, что нейронные сети склонны лучше усваивать низкочастотные функции. Эффективным решением данной проблемы является преобразование входного пространства в пространство более высокой размерности и использование его для тренировки. Это преоб-

разование представляет собой набор синусоидальных функций с фиксированными, но уникальными частотами.

$$\gamma(p) = (\sin(2^0\pi p), \cos(2^0\pi p), \ldots, \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p)) \quad (7)$$

Модель NeRF обучается на множестве изображений одной и той же сцены, снятых под разными углами. Это означает, что каждая отдельная модель оптимизирована только для одной сцены, и, несмотря на это, она может обобщать информацию о новых точках обзора внутри сцены.

Для получения окончательного изображения модель использует технику объемометрической отрисовки (volume rendering). Это включает интеграцию предсказанных плотностей и цветов вдоль лучей, исходящих из точки обзора, для создания двухмерного изображения сцены.

Для каждого пикселя изображения выпущенный луч от камеры пересекает сцену, и по длине этого луча выбираются точки. Каждой точке назначаются пространственные координаты и соответствующий ей вектор направления камеры.

В каждой выбранной точке (x, y, z) и под углом обзора (θ, Ø) нейронная сеть предсказывает плотность σ и цвет (r, g, b).

Используя объемометрическую отрисовку, значения плотностей и цветов специфическим образом агрегируются вдоль каждого луча, чтобы получить итоговое изображение. Это включает в себя взвешивание предсказанных цветов на основе плотностей и дальнейшую суммирование этих взвешенных цветов для получения пикселя изображения. [8]

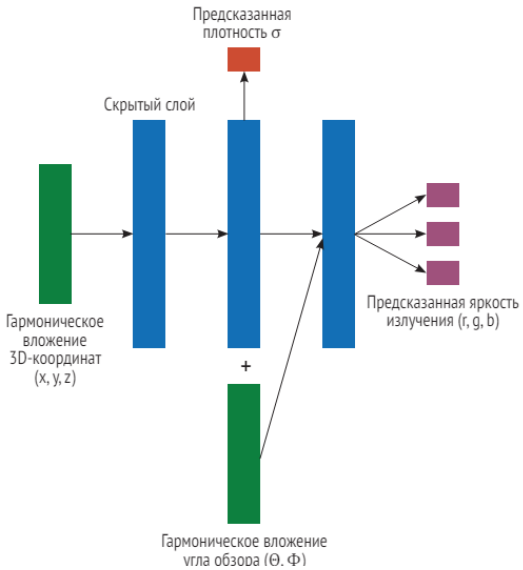

Рис. 4. Упрощенная архитектура модели NERF

3.3 Mesh R-CNN и Mask R-CNN

Этот раздел описывает модель Mesh R-CNN, которая объединяет две важные задачи в одну сквозную модель: сегментацию изображений и предсказание 3D-структуры. Модель Mesh R-CNN сочетает в себе известную Mask R-CNN, предназначенную для сегментации экземпляров на основе обнаружения объектов, с новой моделью, предсказывающей трехмерные структуры. [12]

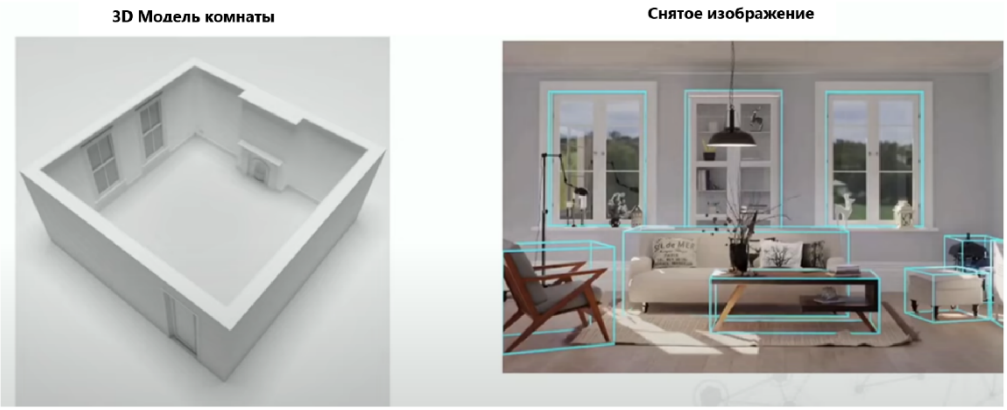

Рис. 5. Результат обработки Mesh R-CNN в части распознавания объектов

Mask R-CNN основывается на алгоритме, который обеспечивает высокую прецизионность в эталонных тестах, и действует в рамках семейства R-CNN-сетей как двухэтапная модель для обнаружения объектов. Однако Mesh R-CNN расширяет эту функциональность, предлагая не только 2D-сегментацию, но и возможность генерировать 3D-полигональные сетки для обнаруженных объектов. Таким образом, Mesh R-CNN стремится имитировать человеческое восприятие, которое воспринимает мир в трехмерном пространстве, и делает шаг вперед, выводя объекты в 3D. [8]

Рис. 6. Результат работы модели Mesh R-CNN в части создания вокселей.

Модель Mask R-CNN обрабатывает на входе RGB-изображение и выдает на выходе ограничительные рамки, метки категорий и маски сегментации экземпляров. Вначале изображение проходит через каркасную сеть, обычно основанную на ResNet, такой как ResNet-50-FPN. Эта сеть генерирует карту признаков, которая затем передается в сеть предложений участков (RPN). RPN выдает предложения, которые обрабатываются ветвями классификации объектов и предсказания масок, в результате чего на выходе получаются классы и маски. [8]

Эта архитектура Mask R-CNN сохраняется и в модели Mesh R-CNN, но с добавлением предсказателя полигональной сетки. Этот новый модуль включает две ветви: ветвь вокселов и ветвь уточнения полигональной сетки.

Ветвь вокселов обрабатывает предложенные и выровненные признаки, выдавая на выходе грубые предсказания вокселов. Эти грубые предсказания затем передаются на вход ветви уточнения полигональной сетки, которая генерирует окончательную полигональную сетку. Потери ветви вокселов и ветви уточнения полигональной сетки добавляются к потерям рамки и маски, и вся модель тренируется сквозным образом от

начала до конца. [12] Mesh R-CNN не только сохраняет возможности Mask R-CNN по сегментации и классификации 2D-объектов, но и добавляет возможность предсказания их трехмерной структуры, делая модель более комплексной и соответствующей реальному восприятию окружающего мира в 3D.

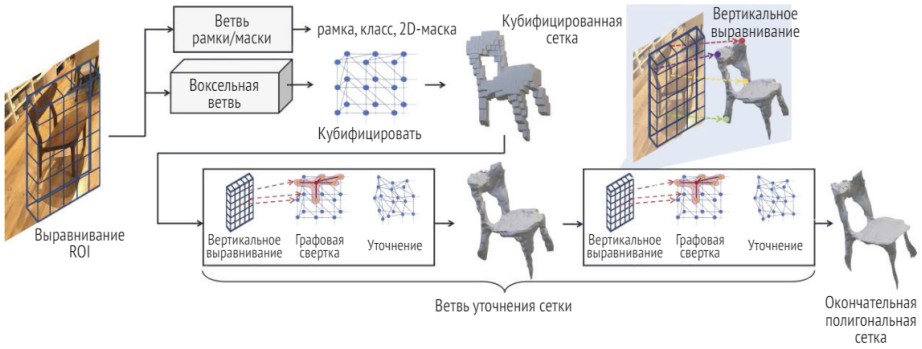

Рис. 7. Архитектура модели Mesh R-CNN

Модуль предсказания полигональной сетки создан для обнаружения 3D-структуры объекта. Он представляет собой логическое развитие модуля RoIAlign и отвечает за прогнозирование и выдачу окончательной полигональной сетки.

Поскольку нам нужно получать трехмерные полигональные сетки из реальных изображений, использование фиксированных шаблонов сетки с определёнными топологиями невозможно. Поэтому предсказатель полигональной сетки состоит из двух ветвей. Совместное использование воксельной ветви и ветви уточнения полигональной сетки помогает справиться с проблемой фиксированных топологий.

Воксельная потеря представляет собой двоичную перекрестную энтропию, которая минимизирует предсказанные вероятности занятости воксела, сравнивая их с истинными значениями занятости.

Ветвь уточнения полигональной сетки включает последовательность из трех операций: выравнивание вершин, графовую свертку и уточнение вершин. Выравнивание вершин аналогично выравниванию ROI; оно находит признаки для каждой вершины полигональной сетки, выровненные по изображению.

Для этого используется следующий метод отбора точек из полигональной сетки: при заданных вершинах и гранях точки выбираются равномерно из вероятностного распределения поверхности полигональной сетки. Вероятность каждой грани является пропорциональной её площади.

С помощью этих методов отбора, из достоверных данных формируется облако точек Q, а из предсказания – облако точек P. Далее, вычисляется DP,O, то есть множество пар (p, q), где q – это ближайший сосед точки p в Q. Затем вычисляется фасочное расстояние между P и Q.

$$L_{\text{cham}}(P,Q) = |P|^{-1} \sum_{(p,q)\in\Delta_{P,Q}} |p-q|^2 + |Q|^{-1} \sum_{(q,p)\in\Delta_{Q,P}} |p-q|^2 \quad (8)$$

Далее вычисляется абсолютное нормальное расстояние

$$L_{\text{norm}}(P,Q) = -|P|^{-1} \sum_{(p,q)\in\Delta_{P,Q}} |u_p \cdot u_q|^2 - |Q|^{-1} \sum_{(q,p)\in\Delta_{Q,P}} |u_p \cdot u_q|^2 \quad (9)$$

3.4 Проприетарные решения предобработки данных

Исходя из обзора литературы было выявлено, что практически в каждом исследовании используется готовый набор данных. В рамках данного исследования для подготовки данных использован программно-аппаратный комплекс, который формирует изображение объекта с разных ракурсов в виде фотоизображений. В

результате возникает необходимость в получении метаданных для обработки самостоятельно, для чего было решено использовать нижеперечисленное программное обеспечение:

3DF Zephyr — это передовое программное обеспечение для фотограмметрии, которое позволяет пользователям автоматически преобразовывать серии фотографий в трехмерные модели. Этот инструмент использует мощные алгоритмы для обработки изображений и создания точных и детализированных 3D-реконструкций. [10] Программа обладает широким спектром функций, включая возможность редактирования моделей, измерения расстояний и площадей, а также экспорта в различные форматы. 3DF Zephyr применяется во многих областях, таких как архитектура, инженерия, культурное наследие и игровая индустрия, благодаря своей гибкости и точности в воссоздании физических объектов в цифровом виде.

COLMAP — это мощный инструмент фотограмметрии с открытым исходным кодом, который предоставляет пользователям комплексные возможности для создания трехмерных моделей из фотографий. Программа сочетает в себе автоматизированные алгоритмы структуры из движения (Structure from Motion, SfM) и плотного стереосопоставления для точного воссоздания трехмерных сцен и объектов. [11] COLMAP идеально подходит для исследователей и энтузиастов 3D-моделирования, предлагая функции, такие как автоматическое выравнивание фотографий, оптимизация геометрии и текстурирование моделей. Этот инструмент широко используется в областях, где требуется высокая точность воссоздания реальности, включая археологию, архитектуру и создание контента для виртуальной реальности

4. Методика исследования

В ходе выполняемого исследования было решено осуществить сбор и подготовку фотоснимков объекта вместо использования готовых наборов данных, осуществить преобразование 2D фотографий объекта в 3D модель с помощью проприетарными решений Colmap и 3D Zephyr, а также выполнить работку обработку ранее перечисленными в статье методами, в частности PyTorch3D и оценить полученные результаты.

4.1 Подготовка данных

Вместо заранее подготовленных данных, которые обычно используются в NERF, для создания и подготовки данных применялся интегрированный программно-аппаратный комплекс, детальный описанный авторами в работе [13], который специализирован на приеме, передаче и хранении серий фотографических снимков. Этот комплекс обеспечивает возможность всесторонней фотографии объекта, что позволяет зафиксировать его со всех сторон и сформировать полноценный датасет. Полученные изображения затем могут быть загружены на компьютер, где они используются для создания точной трехмерной модели объекта, обеспечивая высокую степень детализации и реалистичности визуализации. На рисунке 8 представлена установка программно-аппаратного комплекса. В основе его лежит:

- Установка, способная вращаться на 360 градусов, с подвижной кареткой, которая может перемещаться на 90 градусов;
- Шаговые двигатели и соответствующие драйверы, с зубчатыми перфорированными роторными частями диаметром 450 мм, обеспечивающие движение по зубчатым рельсам установки;
- Плата и камера, закрепленные на подвижной каретке и защищённые корпусом (Рис. 9);
- В качестве камеры выступала Raspberry Pi Camera v2.1 Данная камера имеет сенсор Sony IMX219 Exmor, разрешение в 5 мегапикселей, максимальное разрешение фото 2592×1944 пикселей, имеет фиксированное фокусное расстояние [14];

- Хромакей, который значительно улучшает качество и точность создаваемой модели, сокращая время обработки и обеспечивая создание качественного фона для моделей, который можно использовать в дальнейшем.

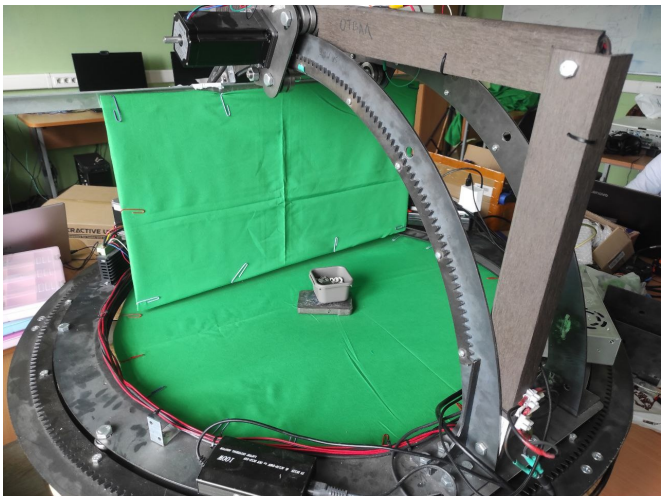

Рис. 8. Установка для съемки фотоизображений

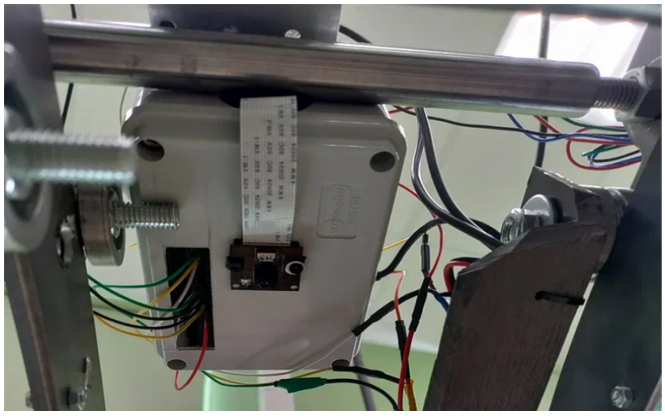

Рис. 9. Подвижная каретка с установленной камерой и платой в корпусе

На рисунке 10 представлен пользовательский интерфейс для работы с данным программно-аппаратный комплексом, в рамках которого можно указать количество остановок (позиций для съемки фотографий), а на 11 готовый датасет (коллекция фотографий одного конкретного объекта). Необходимое и достаточное количество фотографий было выбрано эмпирическим путем каждого объектов в количестве 50.

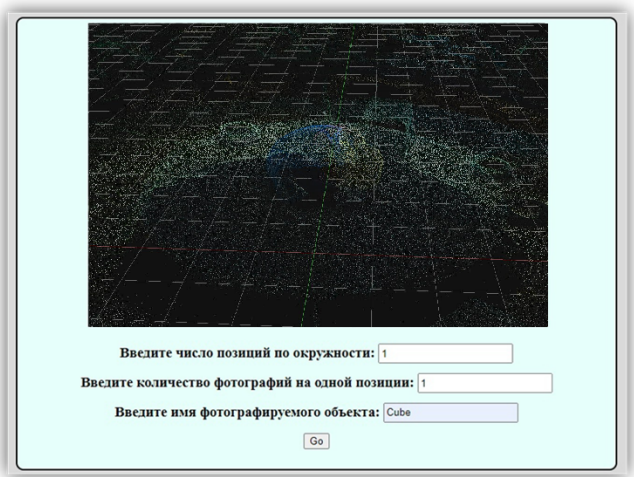

Рис. 10. Пользовательский интерфейс программно-аппаратного комплекса

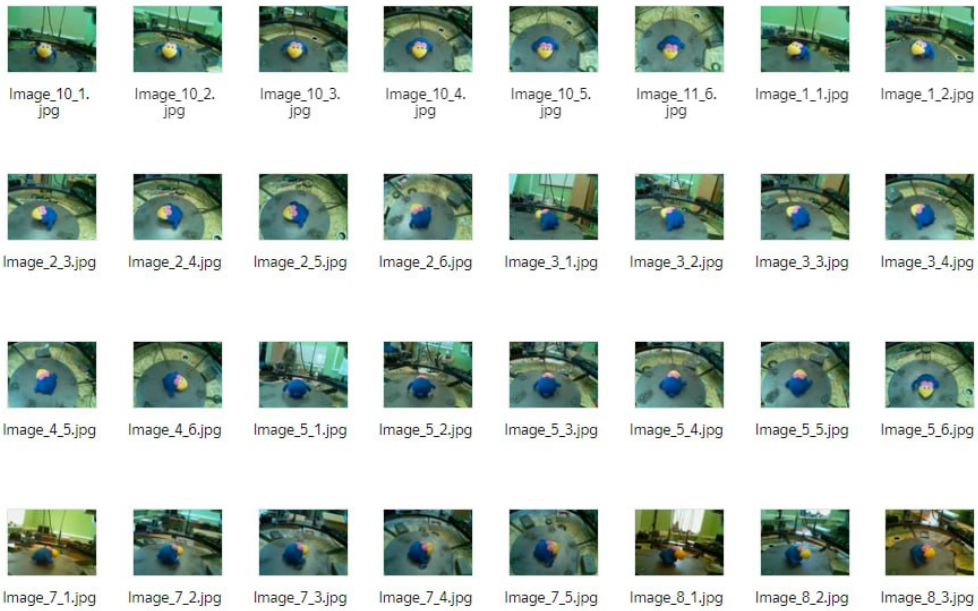

Рис. 11. Пример датасета

4.2 Преобразование в 3D модель

Следующий этап - преобразование датасета фотографий в 3D-модель формата .obj. Для этого использовались программы 3DF Zephyr и COLMAP.

В рамках исследования была выполнена подготовка данных для нескольких объектов, среди которых были простейшие геометрические фигуры и более сложные, среди которых были объекты со сложной геометрией и зеркальной поверхностью. Результат визуализации каждого из перечисленных объектов предоставлен на рисунке 12.

Рис. 12. Пример обработанных 3D-объектов в ПО 3DF Zephyr

Пример целевого 3D-объекта, который был выбран для продолжения исследования, представлен на рисунке 13. Пример обработанного 3D-объекта представлен на рисунке 14.

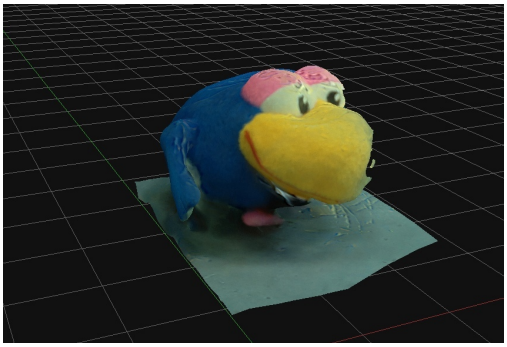

Рис. 13. Пример обработанного 3D-объекта в ПО 3DF Zephyr

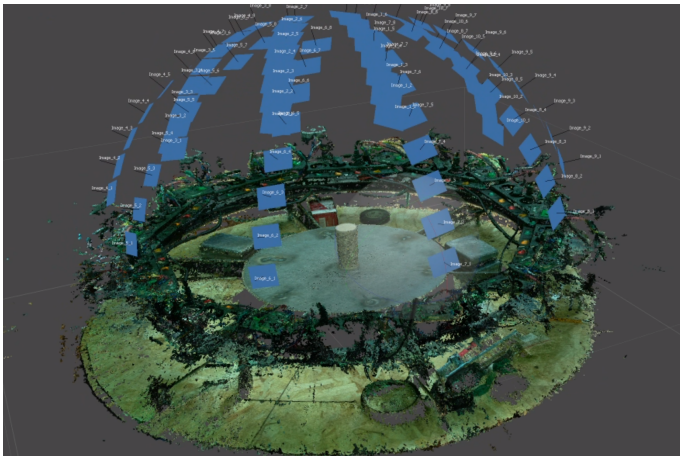

Рис. 14. Пример обработанного 3D-объекта в ПО COLMAP

5. Результаты исследования

На предыдущих этапах исследования с помощью программно-аппаратного комплекса были получены 50 изображений объектов с разных ракурсов. Один из объектов эмпирическим путем был выбран в качестве тестируемого (рис. 15).

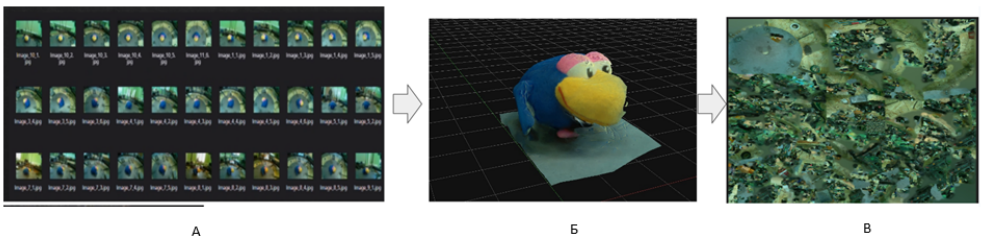

Рис. 15. Подготовленный датасет. А. – Подготовленный датасет, Б. – 3D модель, созданная в 3D Zephyr из датасета, В. – Текстура объекта в формате .mtl.

Обработка 3D файлов включает в себя несколько этапов: создание модели, текстурирование, а затем обработку этой модели с помощью разных программ для получения конечного результата. В этом процессе используются различные форматы файлов, каждый из которых имеет своё назначение и структуру:

- Формат .obj широко используется для хранения информации о 3D моделях. Он представляет собой текстовый файл, который содержит данные о вершинах, текстурных координатах, нормалях и индексах.
- Формат .mtl используется для описания материала для .obj объекта. Он определяет такие свойства, как цвет материала, текстура, коэффициент отражения и т.д.
- Формат .npz — это контейнер для хранения нескольких массивов данных в формате NumPy, который является стандартом для научных и числовых вычислений на

языке Python. Хранит большие объемы численных массивов данных, таких как облака точек, что удобно для обработки и анализа данных в научных и технических приложениях.

Имея 50 фотографий одного объекта с разных ракурсов, мы использовали 3D Zephyr и COLMAP для получения метаданных из этих фотографий, включая файлы форматов obj, mtl и npz. (рис. 16).

Рис. 16. Перевод датасета в .npz-формат

После этого мы приступили к решению задачи оценивания поз объекта. В процессе решения задачи оценивания поз объекта была применена дифференцируемая отрисовка, в результате которой трехмерная модель сравнивалась с однократным изображением с одного ракурса. Однократное изображение с одного ракурса было получено из исходных данных датасета. Сравнение выполнялось с помощью средней квадратической ошибки. В этом разделе мы продемонстрировали конкретный пример использования дифференцируемой отрисовки для задач трехмерного компьютерного зрения. Наша цель — оценить позу объекта по одному изображению, полученному в результате наблюдения. Исходя из того, что у нас есть трёхмерная модель объекта, мы можем выполнить сравнение RGB-изображения объекта и силуэтного изображения – рис. 17. Задача заключается в оценке ориентации и положения объекта в момент съемки этих изображений.

Рис. 17. Преобразование 3D-объекта. А. – Облака точек объекта (3D модель), Б. – Однократные изображения объекта с разных ракурсов.

Поскольку поворачивать и перемещать полигональные сетки затруднительно, вместо этого мы зафиксировали их ориентацию и местоположение, решив оптимизировать ориентации и местоположения камеры. Исходя из предположения, что камера всегда направлена на полигональные сетки, задачу можно упростить до оптимизации местоположения камеры.

Таким образом, мы сформулировали задачу оптимизации, где переменными являются координаты камеры. Используя дифференцируемую отрисовку, можно получить синтетические RGB-изображения и силуэтные изображения полигональной сетки объекта. Затем эти синтетические изображения сравниваются с наблюдаемыми, что позволяет рассчитать функции потерь между ними. В качестве функции потерь используются среднеквадратичные ошибки. Поскольку процесс дифференцируем, можно вычислить градиенты функций потерь по отношению к переменным оптимизации. Затем с помощью алгоритма градиентного спуска можно найти оптимальные позиции камеры, при которых синтетические изображения максимально совпадают с наблюдаемыми. В результате была определена позиция положения камеры - Рис. 18.

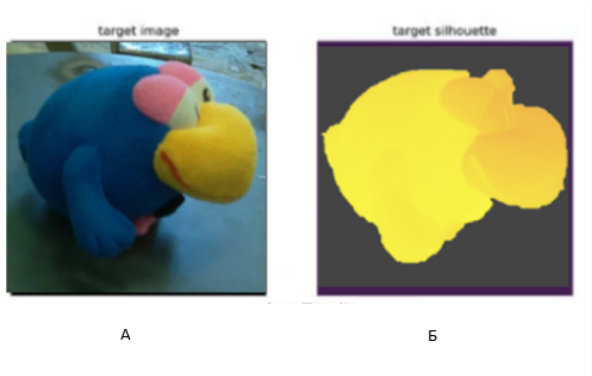

Рис. 18. Сравнение 3D-модели. А – Однократное изображение с одного ракурса. Б. – Целевое изображение ориентации объекта.

На рисунке 19 изображена среднеквадратичная ошибка.

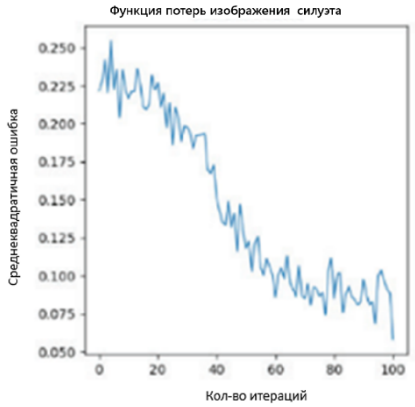

Рисунок 19. Среднеквадратичная ошибка

На дальнейших этапах исследования планируется использовать NERF для решения обратной распознаванию задачи - создания 3D моделей и усовершенствованную реализацию Mesh R-CNN для формирования полигональной сетки с целью определения объектов на фотографиях с детальным предсказанием геометрии и топологии.

В задаче реконструкции 3D-объекта и всей сцены из множества снимков, сделанных с разных сторон, есть несколько ключевых шагов и технологий, которые могут существенно облегчить её выполнение.

Во-первых, мы начинаем с этапа сбора данных. Установка делает снимки объекта с разных углов, и первый шаг здесь — это правильно сегментировать объект на этих снимках. Для этого мы можем использовать Mask R-CNN, известную своей точностью в

задаче сегментации изображений. Это позволит нам автоматически найти и выделить объект, что критически важно для последующих шагов реконструкции. Ведь точность сегментации определяет, насколько «чистыми» будут данные, переданные в следующую стадию алгоритма. Далее переходим к реконструкции самого объекта. Существует несколько методов, но MESH R-CNN выглядит как особо перспективный выбор, когда речь идет о 3D-реконструкции из 2D-изображений. Эта технология позволяет нам получать детализированные сеточные модели, которые могут быть использованы для дальнейшего анализа или визуализации. Однако стоит помнить, что подобные методы зачастую требуют значительной вычислительной мощности и могут быть довольно сложны в реализации. Но если целевым критерием является высокая точность модели, то это вполне оправданно. После успешной реконструкции объекта переходим к важному аспекту — реконструкции всей сцены. Здесь нам может очень помочь NeRF (Neural Radiance Fields). Эта технология замечательна тем, что позволяет воспроизводить не только форму, но и освещение, текстуру, что делает её идеальной для создания фотореалистичных моделей сцены. NeRF моделирует сцену на основе множества изображений, создавая объёмное представление, которое легко интегрируется в виртуальные и дополненные реальности. Но опять же, за высокое качество мы платим тем, что NeRF требует значительных ресурсов для обучения модели. Теперь, когда мы получили модели объекта и сцены, возникает вопрос их интеграции и визуализации. Тут на помощь приходит PyTorch3D. Он поддерживает разнообразные форматы и возможности визуализации, что позволяет легко объединить полученные результаты и получить конечный продукт. Возможность интеграции также важна для конечного этапа верификации результатов и их последующего использования, будь то в игровых движках или в инженерных CAD-программах.

Как итог, для задачи реконструкции 3D-объекта и сцены из снимков, сделанных с разных сторон, необходимо интегрировать несколько технологий для достижения наилучших результатов. Mask R-CNN поможет в начальной сегментации, MESH R-CNN обеспечит точную реконструкцию объекта, NeRF создаст детализированную 3D-сцену, а PyTorch3D позволит проводить интеграцию и визуализацию моделей. Совокупный сравнительный анализ методов обработки данных указан в таблице 1.

Таблица 1. Сравнительный анализ методов обработки данных.

Характеристика	**PyTorch3D**	**NeRF (Neural Radiance Fields)**	**MESH R-CNN**	**Mask R-CNN**
Тип данных	3D-данные	2D-изображения для 3D-сцены	2D-изображения для создания 3D-моделей	2D-изображения
Цель в данной задаче	Визуализация и обработка 3D-моделей	Фотореалистичная реконструкция 3D-сцен	Реконструкция 3D-моделей из изображения	Выявление объектов и их сегментация
Основная задача	Воссоздание 3D-объектов	Генерация фотореалистичных 3D-сцен	Сегментация и создание 3D-моделей объектов	Выделение объектов для дальнейшей обработки
Преимущества	Визуализация моделей, быстрая обработка	Высокая детализация и точность	Точная реконструкция форм и контуров	Быстрое выделение объектов, подготовка данных
Ограничения	Требуется знание PyTorch, сложная настройка	Высокая вычислительная нагрузка	Зависимость от качества сетчатых данных	Ограничен 2D-сегментацией
Сложность	Средняя	Высокая	Высокая	Средняя
Область применения	Обработка и визуализация 3D-моделей	Виртуальная/дополненная реальность, высокоуровневые реконструкции	Компьютерное зрение, медицина	Предварительная сегментация изображений
Оптимальное использование	Постобработка и визуализация данных	Задачи, требующие фотореалистичной визуализации	Точная реконструкция с фокусом на формы	Подготовка данных для последующей 3D-реконструкции

Подводя итог, в ходе исследования были полученные следующие результаты:

- выполнен обзор литературы в части обработки 3D изображений средствами PyTorch3D, NERF и MESH R-CNN;
- В результате модификации программно-аппаратного комплекса для автоматизированных обмеров относительно малых объектов снаружи и помещения изнутри с применением как фотофиксации, так и лидарного сканирования был добавлен хромакей для улучшения качества изображения и выполнена подготовка фотографий разных типов объектов для последующего анализа;
- получены фотоснимки объекта с разных ракурсов и обработаны 3D Zephyr и COLMAP для получения метаданных фотоснимков, в частности файлов .obj, .mtl и .npz;
- полученные данные были обработаны с использованием библиотеки PyTorch3D с целью определения положения объекта на снимках;
- были проанализированы в ходе литературного анализа и обзора методов обработки данных методы PyTorch3D, NERF и MESH R-CNN, Mask R-CNN в качестве инструментов для обработки изображения, а также, сделаны выводы по использованию реализации MESH R-CNN для дальнейшего исследования с целью определения положения объектов на фотографиях.

Заключение и обсуждение

Введение в эксплуатацию новых технологий и подходов, таких как библиотека PyTorch3D, открывает новые возможности для решения указанных задач. Данное исследование было направлено на изучение потенциала использования PyTorch3D для определения положения камеры и создания трехмерных моделей объектов на основе одного двухмерного изображения. В рамках работы был использован аппаратно-программный комплекс, включающий управление шаговым двигателем для точного позиционирования камеры, систему управления съемкой и механизм передачи данных на удаленный сервер для дальнейшей обработки. Основной задачей исследования являлась оценка точности и эффективности предложенного метода по сравнению с традиционными подходами, а также выявление потенциальных областей применения разработанного решения. Цель работы заключалась в исследовании методов обработки 2D и 2D изображений с использованием библиотеки PyTorch3D.

В процессе работы были решены следующие задачи: подготовка исходных данных (фотографий) объекта с помощью управления шаговым двигателем и последовательного позиционирования камеры, формирование комплексного набора исходных данных на каждой позиции камеры, передача данных на удалённый компьютер для обработки, а также генерация .obj модели объекта. На следующем этапе была выполнена задача определения объекта на 2D снимке на основе OBJ модели с помощью PyTorch3D. Это включало построение облака точек для генерации объемной 3D модели, определение положения камеры в 3D-пространстве по одной фотографии, определение положения 3D объекта на фотографии с использованием дифференцируемой отрисовки и создание 3D вокселей и 3D мешей.

В ходе исследования был выполнен обзор литературы в части обработки 3D изображений средствами PyTorch3D, NERF и MESH R-CNN. Модифицирован программно-аппаратный комплекс для автоматизированных обмеров относительно малых объектов снаружи и помещений изнутри, с применением как фотофиксации, так и лидарного сканирования. Получены фотоснимки объекта с разных ракурсов и обработаны с помощью 3D Zephyr и COLMAP для получения метаданных фотоснимков, в частности файлов .obj, .mtl и .npz. Полученные данные были обработаны с использованием библиотеки PyTorch3D для определения положения объекта на снимках. Были выявлены возможности и ограничения NERF и MESH R-CNN. Сделаны выводы о продолжении исследования в части применения NERF для решения обратной задачи распознавания, которое позволит создать 3D-модели объектов. Для

формирования полигональной сетки и детального предсказания геометрии и топологии объектов на фотографиях будет использоваться усовершенствованная реализация Mesh R-CNN.

Список литературы

1. Cossairt, O., Willomitzer, F., Yeh, C. K., & Walton, M. (2020, November). Low-budget 3D scanning and material estimation using PyTorch3D. 2020 54th Asilomar Conference on Signals, Systems, and Computers (pp. 1316-1317).
2. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
3. Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W. Y., Johnson, J., & Gkioxari, G. (2020). Accelerating 3d deep learning with pytorch3d. arXiv preprint arXiv:2007.08501.
4. Remondino, F., Karami, A., Yan, Z., Mazzacca, G., Rigon, S., & Qin, R. (2023). A critical analysis of NeRF-based 3d reconstruction. Remote Sensing, 15(14), 3585.
5. Wang, Z., Wu, S., Xie, W., Chen, M., & Prisacariu, V. A. (2021). NeRF--: Neural radiance fields without known camera parameters. arXiv preprint arXiv:2102.07064.
6. Wiles, O., Gkioxari, G., Szeliski, R., & Johnson, J. (2020). Synsin: End-to-end view synthesis from a single image. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 7467-7477).
7. Wu, R., Mildenhall, B., Henzler, P., Park, K., Gao, R., Watson, D., ... & Holynski, A. (2023). Reconfusion: 3d reconstruction with diffusion priors. arXiv preprint arXiv:2312.02981.
8. Ма К., Хегде В., Йольан Л. М12 Трехмерное глубокое обучение на Python / пер. с англ. А. В. Логунова. – М.: ДМК Пресс, 2023. – 226 с.: ил.
9. Liu S. et al. Soft rasterizer: A differentiable renderer for image-based 3d reasoning //Proceedings of the IEEE/CVF international conference on computer vision. – 2019. – С. 7708-7717.
10. Miyake K. Evaluating the Reliability of Three-dimensional Models Constructed Photogrammetry Software 3DF Zephyr by Measuring Joint Angles of Fingers: A Comparison to a Conventional Goniometer //Journal of Plastic and Reconstructive Surgery. – 2024. – Т. 3. – №. 1. – С. 34-38.
11. Fisher A. et al. COLMAP: A memory-efficient occupancy grid mapping framework //Robotics and Autonomous Systems. – 2021. – Т. 142. – С. 103755.
12. Gkioxari G., Malik J., Johnson J. Mesh r-cnn //Proceedings of the IEEE/CVF international conference on computer vision. – 2019. – С. 9785-9795.
13. Коньков В.В., Замчалов А.Б., & Жабицкий М.Г. (2023). Программно-аппаратный комплекс получения фотоизображений на основе технологии IIoT и анализ точности различных алгоритмов цифровой генерации 3d моделей на основе принципа фотограмметрии. International Journal of Open Information Technologies, 11 (8), 32-51.
14. Камера для Raspberry Pi «Модель D». — Текст: электронный // Amperka: [электронный ресурс]. — URL: http://wiki.amperka.ru/products:camera-raspberry-pi-model-d (дата обращения: 15.04.2023).

Scientific Visualization, 2018, volume 10, number 2, pages 70 - 83, DOI: 10.26583/sv.10.2.06

Application of PyTorch3D and NERF Computer Vision Tools for Building a Point Cloud of a Three-Dimensional Model and Determining the Camera Position of Still Images in Space

V.V. Konkov[1], A.B. Zamchalov[2]

Institute of Intelligent Cybernetic Systems, National Research Nuclear University MEPhI, Moscow, Russian Federation

[1] ORCID: 0009-0005-1197-2248, vlad.konkov.7145@gmail.com
[2] ORCID: 0009-0006-0955-1062, andreizam@yandex.ru

Abstract

Recently, computer graphics plays a key role in solving computer vision problems. The problem of converting 2D images into 3D models continues to be urgent, as it requires precise determination of camera position and construction of accurate 3D models of objects. Traditional methods are often limited in application and do not offer a comprehensive solution. This study examines the use of PyTorch3D and NERF libraries to determine the camera position in 3D space and create a 3D model of an object from a single 2D image. As a method of data preparation, a hardware and software system was used, including a stepper motor control device that provides manual and sequential positioning of the camera and its return to the initial position, a shooting control system to generate a comprehensive set of photos at each camera position, and a mechanism for sending data to a remote computer for further processing. The PyTorch3D library was selected during the study to explore the possibilities of converting 2D images into 3D models or determining the position of an object in the photos. The processing process included several steps: building a point cloud to generate a 3D volumetric model of the object, determining the camera position in 3D space from a single 2D image using inverse problem algorithms, and constructing a 3D object using differentiable rendering, creating 3D voxels and 3D meshes. The results of this study showed successful determination of camera position in 3D space and construction of a 3D object model from a single 2D image, demonstrating the advantages of using the PyTorch3D library over other existing models. These findings can be applied in the development of software and hardware systems for creating 3D images from 2D photographs. The study confirmed the relevance and effectiveness of using PyTorch3D library to solve the problems of converting 2D images into 3D models. Further work will be aimed at expanding the functionality of the system and its use in various areas of computer vision.

Keywords: computer vision; PyTorch3D; NERF; 3D modeling; camera positioning, point cloud; deep learning; 3D reconstruction; differentiated rendering.

References

15. Cossairt, O., Willomitzer, F., Yeh, C. K., & Walton, M. (2020, November). Low-budget 3D scanning and material estimation using PyTorch3D. 2020 54th Asilomar Conference on Signals, Systems, and Computers (pp. 1316-1317).

16. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

17. Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W. Y., Johnson, J., & Gkioxari, G. (2020). Accelerating 3d deep learning with pytorch3d. arXiv preprint arXiv:2007.08501.

18. Remondino, F., Karami, A., Yan, Z., Mazzacca, G., Rigon, S., & Qin, R. (2023). A critical analysis of NeRF-based 3d reconstruction. Remote Sensing, 15(14), 3585.

19. Wang, Z., Wu, S., Xie, W., Chen, M., & Prisacariu, V. A. (2021). NeRF--: Neural radiance fields without known camera parameters. arXiv preprint arXiv:2102.07064.

20. Wiles, O., Gkioxari, G., Szeliski, R., & Johnson, J. (2020). Synsin: End-to-end view synthesis from a single image. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 7467-7477).

21. Wu, R., Mildenhall, B., Henzler, P., Park, K., Gao, R., Watson, D., ... & Holynski, A. (2023). Reconfusion: 3d reconstruction with diffusion priors. arXiv preprint arXiv:2312.02981.

22. Ma K., Hegde W., Yolan L. M12 Three-dimensional deep learning in Python / translated by A. V. Logunov. - Moscow: DMK Press, 2023. - 226 p.: ill.

23. Liu S. et al. Soft rasterizer: A differentiable renderer for image-based 3d reasoning //Proceedings of the IEEE/CVF international conference on computer vision. – 2019. – С. 7708-7717.

24. Miyake K. Evaluating the Reliability of Three-dimensional Models Constructed Photogrammetry Software 3DF Zephyr by Measuring Joint Angles of Fingers: A Comparison to a Conventional Goniometer //Journal of Plastic and Reconstructive Surgery. – 2024. – Т. 3. – №. 1. – С. 34-38.

25. Fisher A. et al. COLMAP: A memory-efficient occupancy grid mapping framework //Robotics and Autonomous Systems. – 2021. – Т. 142. – С. 103755.

26. Gkioxari G., Malik J., Johnson J. Mesh r-cnn //Proceedings of the IEEE/CVF international conference on computer vision. – 2019. – С. 9785-9795.

27. Konkov, V.V., Zamchalov, A.B., & Zhabitsky, M.G. (2023). Complex software and hardware for IIoT-based acquisition of photographic images and analysis of accuracy of different algorithms for digital generation of 3D models based on the principle of photogrammetry. International Journal of Open Information Technologies, 11 (8), 32-51.

28. Camera for Raspberry Pi “Model D”. - Text: electronic // Amperk: [electronic resource]. - URL: http://wiki.amperka.ru/products:camera-raspberry-pi-model-d (address date: 15.04.2023).