ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

УДК 004

Отчет об исследовательском проекте на тему:
Восстановление 3D сцены по фотографиям

Выполнили:

Студент группы MMOBC21_O
Исаакян Феликс Валерьевич _____ (подпись) _____ (дата)

Студент группы MMOBC21_O
Сурков Антон Юрьевич _____ (подпись) _____ (дата)

Принял руководитель проекта:

Васильковский Михаил Константинович
Machine Learning Engineer,
Snap INC

_____ (подпись) _____ (дата)

Москва 2023

Содержание

Аннотация

3D реконструкция - это активно развивающаяся область глубокого обучения, чья основная задача - восстановление пространственной геометрии и структуры объекта по имеющимся изображениям. Успешное решение этой задачи может иметь множество приложений от компьютерной графики до беспилотных автомобилей. Работа посвящена single-view 3D reconstruction - 3D реконструкции по одному изображению/фотографии объекта. В работе приведен краткий обзор классических методов 3D реконструкции. Самостоятельно произведен сбор данных, которые отвечают всем требованиям для различных моделей 3D реконструкции. На имеющихся данных обучена актуальная модель UNICORN, произведен подбор параметров модели. Также предложены методы по адаптации text-to-3d генератора DreamFusion для задачи 3D реконструкции, проведены эксперименты с данными методами.

Ключевые слова: 3D reconstruction, single-view 3D reconstruction, UNICORN, DreamFusion, stable-diffusion, NeRF, pre-training

Введение

Задача 3D реконструкции изображения объекта заключается в восстановлении пространственной геометрию и структуры объекта по имеющимся изображениям (одному или нескольким).

Ранние решения данной задачи основывались на геометрической интерпретации проблемы, а именно, решении математической задачи преобразования трехмерного объекта в двумерное изображение и последующем поиске решения обратной задачи восстановления. Для получения качественных результатов, как правило, требовалось несколько двумерных изображений, сделанных под разными углами с помощью хорошо откалиброванных камер и последующее восстановление трехмерных координат изображения с помощью триангуляции. Прочие методы, позволяющие получать форму объекта из его силуэта, требуют качественно подготовленных двумерных масок объектов, и нескольких двумерных изображений объекта под разными углами. Описанные способы достаточно трудоемки и не всегда эффективны. Следующее поколение подходов к данной проблеме базируется на использует знания об известных объектах для распознавания неизвестных, а именно, проблема реконструкции трехмерного объекта решается, как проблема распознавания. Благодаря развитию методов глубокого обучения, а также большому количеству накопленных наборов данных, восстановление трехмерных объектов стало возможным осуществлять без сложного процесса калибровки камер и без жестких требований к количеству изображений и углам, под которыми сняты объекты.

Тем не менее восстановление объекта по одному изображению (single-view/single-shot 3d reconstruction) по-прежнему является довольно сложной задачей и современное развитие данной области не позволяет говорить об удовлетворительных результатах с проработанной детализацией и/или для широкого класса объектов.

В то же самое время активно развивается смежная отрасль text-to-3d моделей. которые реконструируют 3D объекты по текстовому описанию. Модели в данной области показывают перспективные результаты. Также относительно важной проблемой остается проблема данных: существует не так много датасетов с 3D моделями, которые можно было бы использовать для обучения различных моделей.

Проблема восстановления трехмерных объектов из двумерных изображений актуальна для широкого спектра различных областей от трехмерного моделирования и анимации, медицинской диагностики и промышленного контроля до задач навигации роботов и автономного вождения.

Исходя из вышеизложенных проблем, в данной работе мы ставим перед собой следующие задачи:

1. Подготовка категориального датасета (категория - машины) изображений для обучения модели, который исследователи могли бы свободно использовать в области 3D реконструкции и смежных областях машинного обучения. В качестве набора данных для обучения модели были отобраны трехмерные модели автомобилей из актуального датасета Objaverse, далее были получены изображения из трехмерных объектов. (**Выполнено: Сурков А., см. раздел 1.2**)
2. Подготовить обзор классических и новых методов восстановления трехмерных объектов по изображению. **(Выполнено: Исаакян Ф., см. раздел 1.1.)**
3. Обучение baseline модели на основе статьи Реконструкция на основе единственного изображения с помощью согласованности между экземплярами (Share With Thy Neighbors: Single-View Reconstruction by Cross-Instance Consistency) на собранных данных. **(Выполнено: Исаакян Ф., см. раздел 1.3**)
4. Эксперименты по улучшению baseline модели. (**Выполнено: Исаакян Ф., см. раздел 1.4**)
5. Адаптация text-to-3d модели DreamFusion для задачи 3D реконструкции. (**Выполнено: Сурков А., см. Глава 2**)
6. Проведение экспериментов с предложенными методами до-обучения text-to-3d модели DreamFusion для задачи 3D реконструкции. (**Выполнено: Сурков А., см. Глава 2**)

Также мы ставим задачу по проверке трех гипотез:

1. Модель Share With Thy Neighbors: Single-View Reconstruction by Cross-Instance Consistency (далее UNICORN), обученная на собранном категориальном датасете способна успешно справляться с задачей 3D реконструкции и показывает большую стабильность (отсутствие вырожденных решений), однако не способна воспроизводить мелкие детали объектов.
2. Модель DreamFusion можно адаптировать под задачу 3D реконструкции с целью более детализированного восстановления объектов, однако, данная модель не будет обладать большой стабильностью и будет справляться только с отдельными объектами.
3. Можно преодолеть недостатки обоих методов объединив с помощью пред-обучения модели DreamFusion и UNICORN.

Глава 1. 3D Реконструкция. Классические и новые методы

1.1 Обзор классических методов восстановления трехмерных объектов по изображению

Для задачи восстановления трехмерного изображения, крайне важно определить представление данных. Выбор представления будет влиять на последующий выбор архитектуры решения, а также качество и производительность. И прежде, чем перейти к обсуждению методов, сначала опишем различные виды представлений данных [9]:

1. Объемные (volumetric) представления: повсеместно применялись в ранних методах трехмерного восстановления с помощью методов глубокого обучения. Такие представления позволяют легко параметризовать трехмерную форму объекта с помощью воксельной сетки (Воксель — это элемент объёмного изображения, содержащий значение элемента растра в трёхмерном пространстве. Грубая, но достаточно близкая аналогия, это пиксель, но в трёхмерном пространстве с дополнительной координатой); позволяют легко перейти от двумерной сверки к трехмерной свертке. Однако объемные представления требовательны к памяти и не все представления способны достичь достаточной воксельной точности.
2. Поверхностные (surface-based) представления: типичные представители — это облака точек и сетки. (Облако точек — это набор вершин в трёхмерной системе координат, определяющихся координатами X, Y и Z и, как правило, предназначены для представления поверхности объекта.) Эти представления не являются обычными структурами и сложно вписываются в различные архитектуры. Хотя их несомненный плюс — это эффективность с точки зрения памяти.
3. Посреднические (intermediation) представления: основная идея состоит в разбиении задачи на последовательные шаги, где каждый шаг предсказывает промежуточное представление.

1.1.1 Объемные представления

Объемные представления разбивают пространство вокруг трехмерного объекта в воксельную сетку V. Цель в том, чтобы восстановить сетку

$$\widehat{V} = f_0(I),$$

где I - изображение, f_0 - предиктор формы объекта, который максимально приближает предсказываемую форму к искомой.

Основное преимущество использования объемных сеток заключается в том, что многие из существующих архитектур глубокого обучения, предназначенных для анализа двумерных изображений, можно переиспользовать на трёхмерных данных, заменив массив двумерных пикселей на массив трехмерных, а затем обработав сетку с помощью операций трехмерной свертки и пулинга. Также важно отметить, что чем меньше шаг дискретизации, тем точнее будет выходное представление.

Существует 4 основных объемных представления:

1. Двоичная сетка (binary occupancy grid). Если воксель принадлежит объекту, то ему присваивается 1, иначе 0.
2. Вероятностная сетка (probabilistic occupancy grid). Каждому вокселю присваивается значение равное вероятности принадлежать объекту.
3. Функция расстояния со знаком. (The Signed Distance Function (SDF). Каждому вокселю присваивается значение до ближайшей точки поверхности объекта. Если воксель внутри объекта, то значение с отрицательным знаком, если вне - с положительным.
4. Функция усеченного расстояния со знаком (Truncated Signed Distance Function (TSDF).

Методы, использующие объемные представления:

1. MarrNet [10] (см. Рисунок 1). Модель состоит из основных трех компонентов: 2.5D оценка эскиза, оценка 3D формы и расчет функции потерь для оценки согласованности проекций. В основе модели лежит архитектура энкодер-декодер для восстановления глубины, силуэта и нормалей. Плюсы метода: значительно проще восстанавливать глубину, силуэт и нормали из двумерного изображения по сравнению с оригинальным трехмерным объектом, хорошо реконструирует объекты, снятые на неоткалиброванные камеры. Недостатки метода: недостаточно точно восстанавливает объекты сложной геометрической формы и тонкие объекты, также данный метод имеет высокую вычислительную сложность и высокие

требования к памяти.

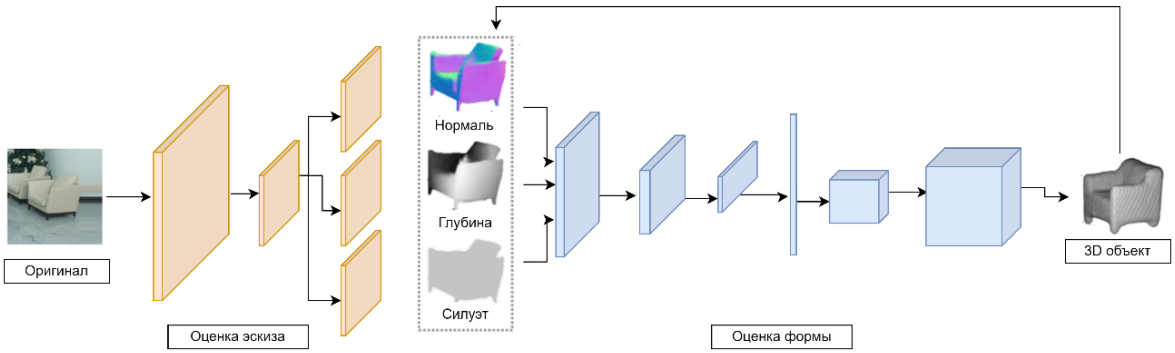


Рисунок 1. Архитектура MarrNet [10]

2. Иерархическое прогнозирование поверхностей и Сети генерации октодеревьев. (Hierarchical Surface Prediction (HSP) [11] (см. Рисунок 3) / Octree Generating Networks (OGN)) (см. Рисунок 2, 4). В основе этих методов лежит изучение структуры данных октодерева. Октодерево — тип древовидной структуры данных, в которой у каждого внутреннего узла ровно восемь «потомков».

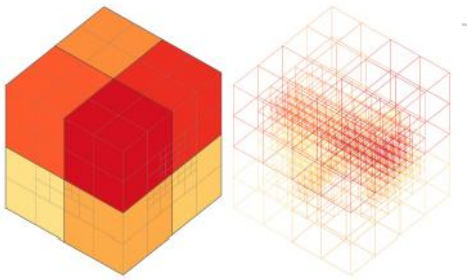


Рисунок 2. Структура данных октодерево, используемая в архитектуре OctNet. [12]

В идеальном сценарии и структура и содержимое октодерева должны оцениваться одновременно. Этого можно достичь следующим образом:

- входные данные кодируются в компактный вектор признаков используя сверточный кодировщик
- вектор признаков декодируется, используя стандартную сверточную сеть. (на этом этапе получает грубую реконструкцию входных данных с разрешением 32^3)
- реконструированный, на предыдущем шаге объем (корень октадерева), разбивается на 8 октантов. Для октантов с вокселями на границе происходит разбиение, свертка и реконструкция

- процесс обратки октантов повторяется рекурсивно до достижения нужного разрешения.

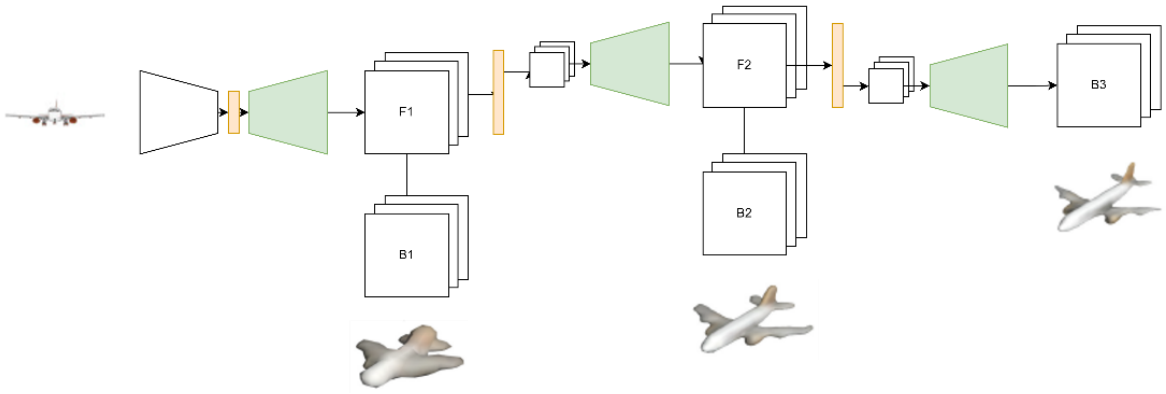


Рисунок 3. Иерархическое прогнозирование поверхностей (Hierarchical Surface Prediction) [11]

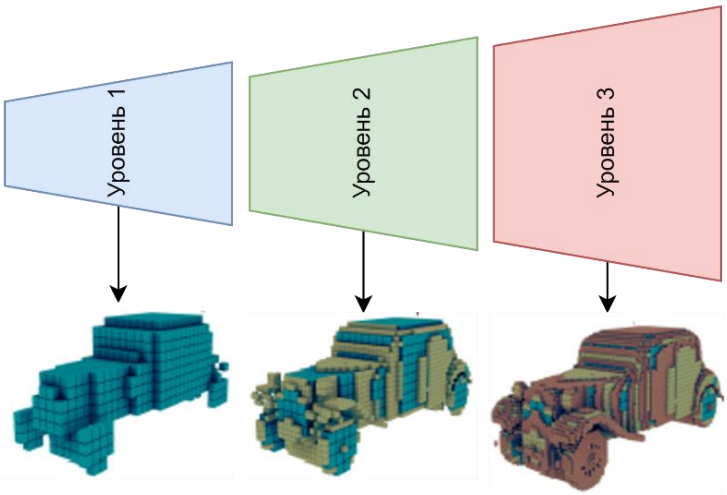


Рисунок 4. Сеть генерации октодеревьев. (Octree Generating Networks OGN) [13]

В методе HSP октодерево исследуется в глубину, а в методе OGN в ширину (см. рисунок). Эти методы позволяют реконструировать объемные сетки размером 512^3.

1.1.2 Поверхностные представления

Основная проблема при работе с поверхностями заключается в том, что как сетки или облака точек, не имеют регулярной структуры и, следовательно, их сложно использовать в моделях, основанных на глубоком обучении, особенно тех, которые используют сверточные нейронные сети. [9]

Все методы, использующие поверхностные представления, можно разбить на 2 больших группы:

1. Реконструкция трехмерных изображений, основанная на параметризации. Метод заключается в восстановлении формы, деформируемой поверхности с помощью сети с ветвями: ветвь обнаружения и ветвь оценки глубины, которые работают параллельно, и третья ветвь формы, которая объединяет маску детекции и карту глубины в параметризованную поверхность. Из ограничений методов параметризации — это возможность восстановления только поверхностей низкого рода.
2. Реконструкция трехмерных изображений, основанная на деформации. На вход принимают данные I и оценивается поле деформации Δ, которое при применении к трехмерному объекту позволяет получить реконструированную трехмерную модель.

Рассмотрим детальнее модель деформации произвольной формы (free-form deformation) на примере Mesh Deformation Network:

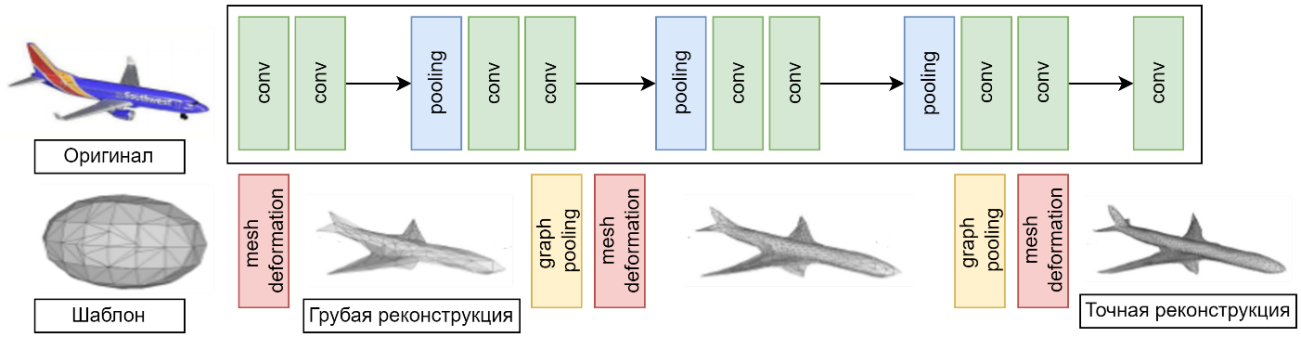


Рисунок 5. Структура каскадной Mesh Deformation Network [14]

- На вход поступает изображение
- Поиск ближайшей фигуры в базе моделей
- Деформация модели с помощью FFD Equation, до тех пор, пока она не будет соответствовать изображению на входе

1.1.3 Посреднические (intermediation) представления

В качестве примера, можно рассмотреть, использование двух блоков (см. Рисунок 6):

1. Энкодер, за которым следует декодер с тремя ветвями, который оценивает карту глубины, нормалей и маску сегментации.
2. Объединенный выход из предыдущих декодеров, поступает в блок архитектуры энкодер - 3D decoder, который восстанавливает объемную сетку и финальный объект.

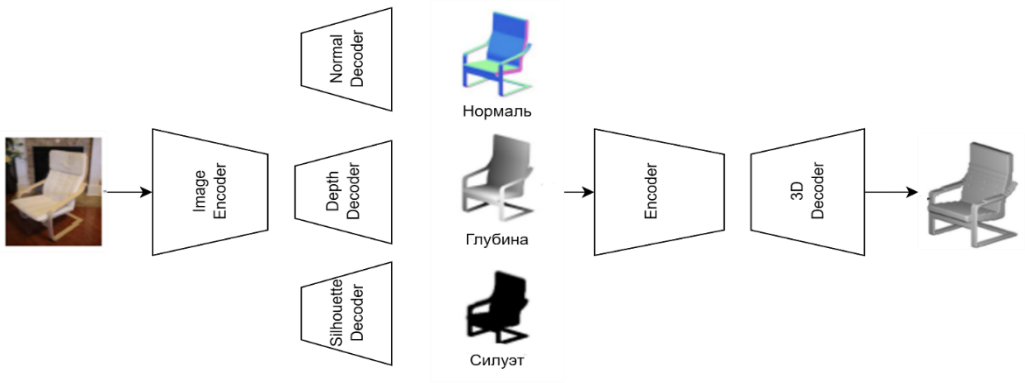


Рисунок 6. Архитектура, использующая посреднические представления, а именно 2.5D эскизы (нормаль, глубина и силуэт) [15]

1.2 Данные для 3D реконструкции

Данные, их количество, структура, входной набор и требуемое представление на выходе модели - то, что во многом определяет архитектуру нейросети и её возможности. Особенно ярким подтверждением данного утверждения является область 3d реконструкции. В этой области на одном полюсе находятся полноценные 3d модели объектов, а на другом - датасеты с 2d изображениями. В первом случае имеется несколько относительно небольших датасетов (ShapeNet, Pascal3D и др.), которые позволяют получать различные наборы данных для обучения нейросети любой архитектуры, во втором же выбор данных гораздо шире - можно использовать любой набор относительно качественных изображений, для которых определены категории (это условие не является технически обязательным для некоторых моделей, но принято использовать в основном категоризованные данные), однако в этом случае без каких либо априорных данных о 3d структуре объекта приходится придумывать архитектурные решения, которые бы имплицитно содержали в себе знания о принципах “работы” 3d мира. Таким образом, с одной стороны 3d модели являются логичным источником данных для задачи 3d реконструкции, а с другой стороны - их скудость заставляет исследователей отбрасывать явные 3d представления (прежде всего это полигональные представления и облака точек) и придумывать архитектурные решения, которые работали бы достаточно хорошо и без присутствия ground truth 3d модели объекта в loss функции. Тем не менее большинство решений все еще используют тот или иной вид изначальных знаний об отношениях 3d мира и объекта, что пока что мешает отказаться от датасетов с 3d моделями в качестве золотого стандарта в области.

Соответственно, можно выделить несколько видов данных, которые требуются для обучения модели 3d реконструкции:

1. Непосредственные изображения объекта (на inference/forward стадии в случае multi-view reconstruction используется несколько изображений с разных ракурсов, а в случае single-view reconstruction - только одно изображение).
2. Положение камеры относительно объекта для имеющихся изображений объекта.
3. Маска объекта, которая как правило представляет собой бинарную матрицу, которая выделяет объект.
4. Какие либо априорные предположения об изначальной форме объекта, симметричности и т.п.

Что касается данных о положении камеры, то они, как правило. представляются в виде двух матриц: матрицы внутренних и матрицы внешних параметров. Матрица внешних параметров переводит точки из внешней системы координат в систему координат камеры. Соответственно, эта матрица меняется при перемещении камеры вокруг статичного объекта. Матрица внутренних параметров переводит точки из системы координат камеры в 2d пиксельные координаты, учитывая фокальное расстояние и разрешение изображения. Нижеприведенный рисунок (см. Рисунок 7) демонстрирует данные преобразования. f_x и f_y обозначают соответствующие фокальные расстояния c_x и c_y - разрешение изображения. Параметры $r_{[;]}$ и $t_{[]}$ - параметры матрицы внешних параметров, которые отвечают за поворот и смещение соответственно. X; Y; Z - координаты точки, выраженные во внешней системе координат, а u; v - координаты точки в пиксельной системе координат.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Рисунок 7. Матрицы внешних и внутренних параметров [17]

Исходя из вышеперечисленных аспектов данных для 3d реконструкции, мы решили собрать собственный датасет, который бы удовлетворял уже имеющимся форматам и

стандартам и помог бы развитию области 3d реконструкции. В качестве источника данных была выбрана созданная в конце 2022 года база 3d моделей objaverse, которая содержит сотни тысяч различных 3d объектов и сцен. Как и принято в области 3d реконструкции мы решили выбрать конкретную категорию, и именно категорию машин, которая является одной из самых популярных категорий в 3d реконструкции, однако данных для этой категории не так много.

Нами было вручную отобрано 3016 моделей машин из примерно 6000 моделей, которые обозначены тэгом “car”, т. к. система аннотирования в базе данных не является строгой и тэги применяются к объектам не только напрямую, но и косвенно относящимся к предмету, что не позволяет быстро формировать качественные датасеты. С точки зрения разнообразия данных наш датасет имеет определенные преимущества перед уже имеющимися датасетами. Во-первых, наш датасет является более разнообразным с точки зрения форм и текстур объектов (в отличии от shapnet/ pascal3D, где модели приближены/являются реальными моделям машин), что позволяет предполагать, что обученные на этом датасете модели могут быть более пригодны для задач 3d моделирования в большем количестве областей, в частности в анимации и компьютерных играх (см. Рисунок 8).

Рисунок 8. Примеры изображений из датасета

Также в наших данных можно найти как более, так и менее детализированные машины (т.е. модели как с небольшим, так и с большим количеством полигонов). Такая особенность может быть полезна для пред-обучения моделей на более простых формах для последующего обучения, а также для моделей улучшения 3d моделей.

Для рендеринга датасета был использован standalone модуль программы blender для python. Для каждой модели был осуществлен рендеринг 24 изображений с различными азимутальными и зенитными углами, варьирующими от 0 до 360 и от 0 до 90 соответственно.

Также для того, чтобы исследователи могли использовать наш датасет для уже имеющихся и своих моделей мы привели его формат к стандартному формату датасета NMR сгенерированному с помощью моделей из shapenet: для каждой модели мы сохранили матрицы внутренних и внешних параметров. Также мы сохранили маски объектов для немного урезанного датасета (2712 моделей), исключив модели с фоновыми объектами, которые не позволяли получить правильную маску для объекта. Таким образом данный датасет можно использовать для большинства multi-view и single-view 3d reconstruction моделей, которые были разработаны за последние годы.

Также мы сохранили параметры камеры в формате colmap, что позволяет использовать данный датасет с различными NeRF (neural radiance field) моделями, которые стали очень популярны в последние годы и являются стандартом глубокого обучения в области 3d для различных задач.

Найти ссылку на датасет в разных разрешениях (64x64, 512x512, первое разрешение больше подходит для моделей из 3d reconstruction, а второе для различных NeRF моделей) и код для рендеринга, который также может применяться к любым моделям из objaverse для рендеринга новых датасетов, можно на github: https://github.com/ToxaSurkov/3d_reconstruction_ojaverse_cars.git.

1.3 Baseline модель Share With Thy Neighbors: Single-View Reconstruction by Cross-Instance Consistency [16]

В основе данного метода - обучение нейронной сети, восстанавливающей трехмерный объект с текстурой по единственному двухмерному изображению с помощью структурированного автоэконкодер и согласованного обучения без учителя. Следует отметить, что большинство методов трехмерной реконструкции без учителя так или иначе используют дополнительную информацию об изображениях, будь то параметры камеры или силуэты объектов (для изображений с фоном), или ограничивающие предположения о форме и структуре объектов, например симметричность, в то время как в данном подходе не используется никакая дополнительная информация и предположения.

1.3.1 Структурированный автоэнкодер

На вход энкодера подается двухмерное изображение, рассчитываются параметры, которые в декодере превращаются в явные факторы: форма, текстура, расположение и фон. Перечисленные факторы способствуют восстановлению изображения.

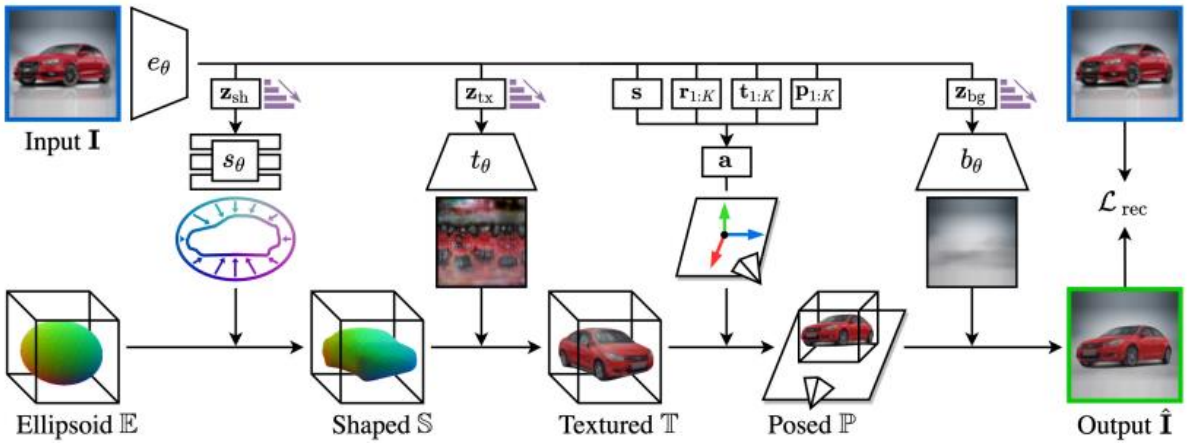


Рисунок 9. Архитектура структурированного автоэнкдера. Для подаваемого на вход изображения модель предсказывает форму, текстуру, позу и фон, которые позволяют сформировать восстановленное изображение. [16]

Рассмотрим отдельно каждый из модулей декодировщика, которые обеспечивают получение четырех ключевых факторов.

1.3.2 Форма

Для задачи реконструкции формы использовали подход, описанный в статье Implicit Mesh Reconstruction from Unannotated Image Collections и параметризацию из AtlasNet, где различные формы представлены в виде полей деформации единичной сферы. Мы применяем деформацию к икосфере, растянутой в эллипсоидную сетку и используем фиксированное анизотропное масштабирование. Пусть задана x - 3D вершина эллипсоида, тогда модуль деформации формы определяется следующим образом:

$$x + s_\theta(x, z_{sh})$$

где s_θ - многослойный перцептрон, принимающий в качестве входных данных конкатенацию трехмерной вершины x и формы z_{sh}. Применяя этот сдвиг ко всем вершинам эллипсоидной сетки мы генерируем деформированную сетку.

Текстура

Используя идеи из статьи Learning Category-Specific Mesh Reconstruction from Image Collections мы моделируем текстуры как изображение, UV-отображение на сетку через

опорный эллипсоид. Пусть z_{tx} текстура, используя сверточная сеть t_θ для создания изображения $t_\theta(z_{tx})$, которое отображается на сферу с помощью сферических координат, чтобы связать 2D точку с каждой вершиной эллипсоида и, таким образом, с каждой вершиной сетки.

1.3.3 Аффинное преобразование

Для рендеринга текстурированной сетки мы определяем ее положение относительно камеры. Кроме того, мы сочли полезным явно моделировать анизотропное масштабирование объектов. Поскольку предсказать расположение достаточно сложно, мы предсказываем K кандидатов на расположение, определяемых поворотами $r_{1:K}$, преобразованиями $t_{1:K}$, и связанные с ними вероятности $p_{1:K}$. В качестве финального расположения объекта, выбирается наиболее вероятное. Мы объединяем масштабирование и наиболее вероятную позу в единый модуль аффинного преобразования. Этот модуль параметризуется следующим образом

$$a = \{s, r, t\}$$

где s, r, t соответствуют трем величинам анизотропного масштабирования, трем углам Эйлера при повороте и трем координатам при преобразовании. Трехмерная точка x на сетке преобразуется по следующей формуле

$$A(x) = rot(r)diag(s)x + t$$

где $rot(r)$ - матрица поворота, связанная с r, а $diag(s)$ - диагональная матрица, связанная с s.

1.3.4 Рендеринг с фоном

Последним шагом нашего процесса является рендеринг сетки на фоновое изображение. Фоновое изображение генерируется из фонового блока z_{bg} с помощью сверточной сети b_θ. Модуль B_{zbg} отображает заданную сетку P на фоновое изображение $b_\theta(z_{bg})$, в результате чего получается реконструированное изображение. Рендеринг выполняется посредством мягкой растеризации сетки.

1.3.5 Согласованное обучение без учителя

Мы обучаем наш структурированный автоэнкодер, синтезируя двумерное изображения и минимизируя потери при реконструкции. Одна из проблем, такого подхода — это вырожденные решения. Для решения данной проблемы мы предлагаем две

следующие техники: прогрессивное обусловливание и реконструкция с помощью соседей. Таким образом, мы оптимизируем форму, текстуру и фон, минимизируя потери для каждого изображения I реконструированного как $\hat{I}$

$$L_{3D} = L_{rec}(I, \hat{I}) + \lambda_{nbr} L_{nbr} + \lambda_{reg} L_{reg}$$

где λ_{nbr} и λ_{reg} скалярные гиперпараметры, L_{rec}, L_{nbr} и L_{reg} соответственно функция потерь реконструкции, функция потерь реконструкции соседей и функция потерь регуляризации. Во всех экспериментах мы используем $\lambda_{nbr} = 1$ и $\lambda_{reg} = 0.01$

Функция потерь реконструкции разбивается на две составляющие: L_{pix} попиксельная квадратичный L_2 функция потерь и перцептивная функция потерь L_{perc} определяемый как L_2 на relu3_3 слое пред обученного $VGG16$. Таким образом, наш полная функция потерь реконструкции выглядит следующим образом

$$L_{rec}(I, \hat{I}) = L_{pix}(I, \hat{I}) + \lambda_{perc} L_{perc}(I, \hat{I}), \quad \lambda_{perc} = 10$$

Несмотря на то, что основанная на деформации параметризация поверхности работает достаточно хорошо, в ряде случаев, когда на поверхности есть складки, этот метод допускает ошибки. Для минимизации этого добавляем функцию потерь регуляризации,

$$L_{reg} = L_{norm} + L_{lap}$$

состоящий из функции потерь нормального состояния L_{norm} и функции потерь сглаживания (Laplacian) L_{lap}.

1.3.6 Прогрессивное обусловливание

Основная идея прогрессивного обусловливания в том, чтобы побудить модель совместно использовать информацию о форме, текстурах, фоне среди различных экземпляров, это позволит минимизировать (а в идеале избежать) вырожденных решений. Восстановление с помощью соседей.

Основная идея реконструкции с помощью соседей состоит в том, что в наборе данные существуют экземпляры с похожей формой или текстурой. Если мы правильно идентифицировали такие объекты то, перестановка их формы или текстуры на стадии генерации в нашей модели должно давать аналогичные результаты на стадии реконструкции. Пусть $\{z_{sh}, z_{tx}, a, z_{bg}\}$ параметры полученные от энкодера для изображения I, пусть Ω - блок, хранящий изображения и параметры последних M обработанных экземпляров

$$\Omega^{(m)} = \{I^{(m)}, z_{sh}^{(m)}, z_{tx}^{(m)}, a^{(m)}, z_{bg}^{(m)}\}$$

Сначала мы выбираем ближайший экземпляр из блока Ω в пространстве текстур используя L_2 метрику

$$m_t = argmin_m \, ||z_{tx} - z_{tx}^{(m)}||_2$$

в пространстве форм проделываем такую же операцию

$$m_s = argmin_m \, ||z_{sh} - z_{sh}^{(m)}||_2$$

После этого мы переставляем коды экземпляров и восстанавливаем текстуры изображения $\hat{I}_{tx}^{(mt)}$ и форму изображение $\hat{I}_{sh}^{(ms)}$ используя параметры пространства текстур и форм соответственно $\{z_{sh}^{(mt)}, z_{tx}, a^{(mt)}, z_{bg}^{(mt)}\}$ и $\{z_{sh}, z_{tx}^{(ms)}, a^{(ms)}, z_{bg}^{(ms)}\}$. В итоге мы считаем функции потерь реконструкции (для текстуры и формы соответственно) между оригинальным и восстановленным изображением $I^{(mt)}$ и $\hat{I}_{tx}^{(mt)}$ и $I^{(ms)}$ и $\hat{I}_{sh}^{(ms)}$. Итоговый вид функции потерь имеет вид

$$L_{nbr} = L_{rec}(I^{(mt)}, \hat{I}_{tx}^{(mt)}) + L_{rec}(I^{(ms)}, \hat{I}_{sh}^{(ms)})$$

1.4 Результаты экспериментов

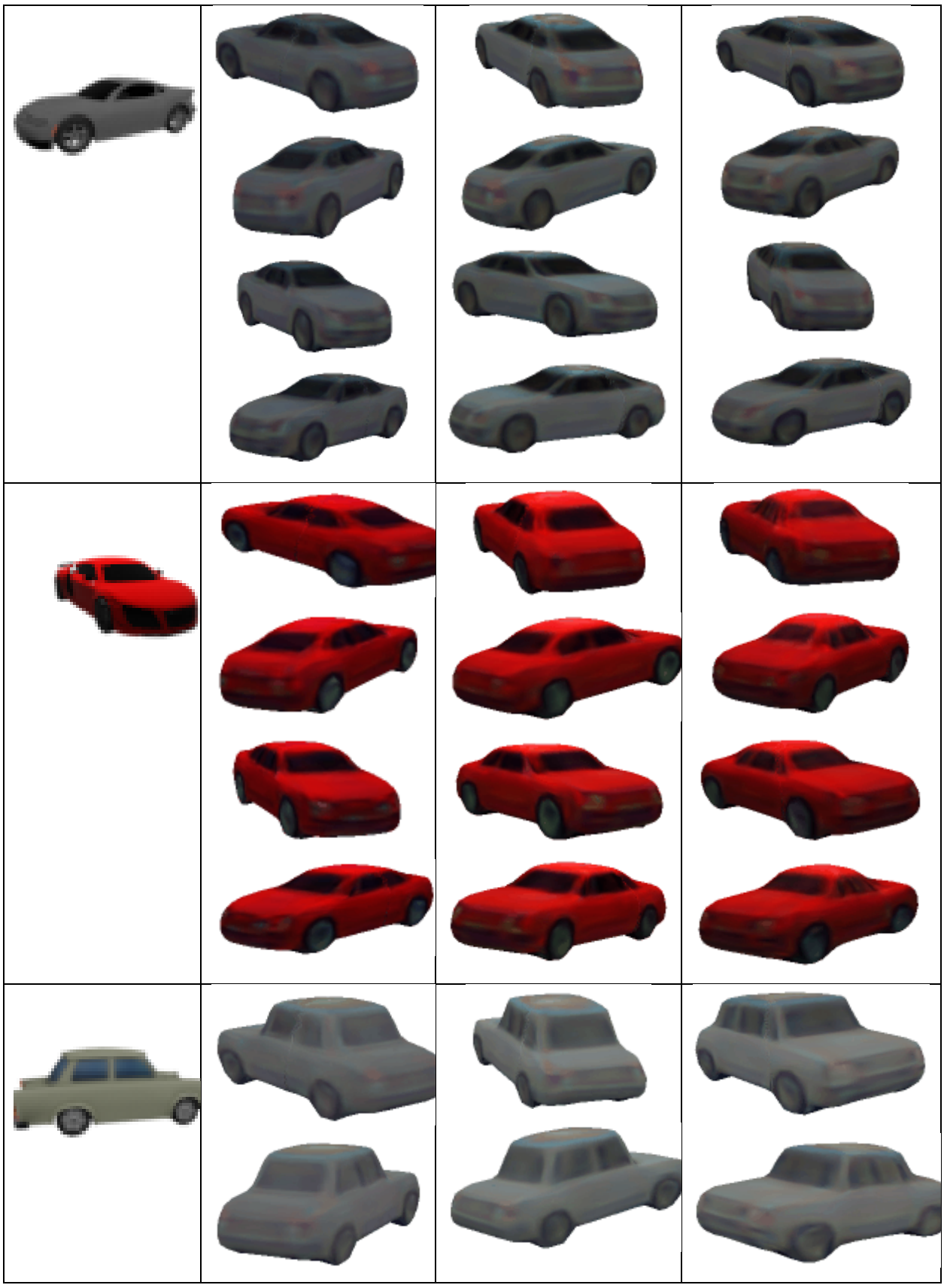
Мы обучили 3 модели с различными гиперпараметрами и реконструировали с помощью каждой модели несколько изображений. Для всех трех моделей число эпох было одинаковым (175 эпох). Ниже приведем параметры, по которым модели отличались:

- Модель 1: optimizer Adam, normal_weight – 0.01, uniform_weight – 0.02
- Модель 2: optimizer RMSProp, normal_weight – 0.05, uniform_weight – 0.1
- Модель 3: optimizer RMSProp, normal_weight – 0.1, uniform_weight – 0.2

При первых экспериментах с моделью, мы столкнулись с тем, что восстановленные изображения, получались излишне сглаженными, и в своих экспериментах мы старались минимизировать этот эффект. В основе изменения uniform_weight и normal_weight лежала гипотеза, что изменение этих весов, должно повлиять на детальность восстановленных изображений, мы провели несколько экспериментов и было обнаружено, что влияние этих параметров не так значительно, гораздо важнее для корректного предсказания – качество и полнота данных в обучающем датасете. В таблицах ниже приведены результаты удачных и неудачных реконструкций и можно отметить, что модель крайне плохо восстанавливает форму редких или уникальных автомобилей. Также мы предполагали, что порядок данных в датасете может оказать влияние на качество алгоритма (алгоритм сначала учится на простых моделях и постепенно наращивается сложность/детальность моделей), но данная гипотеза не подтвердилась и изменение порядка не влияет на итоговое качество.

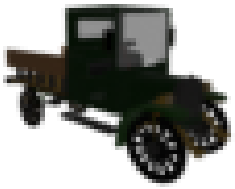
Примеры удачной реконструкции обученных моделей:

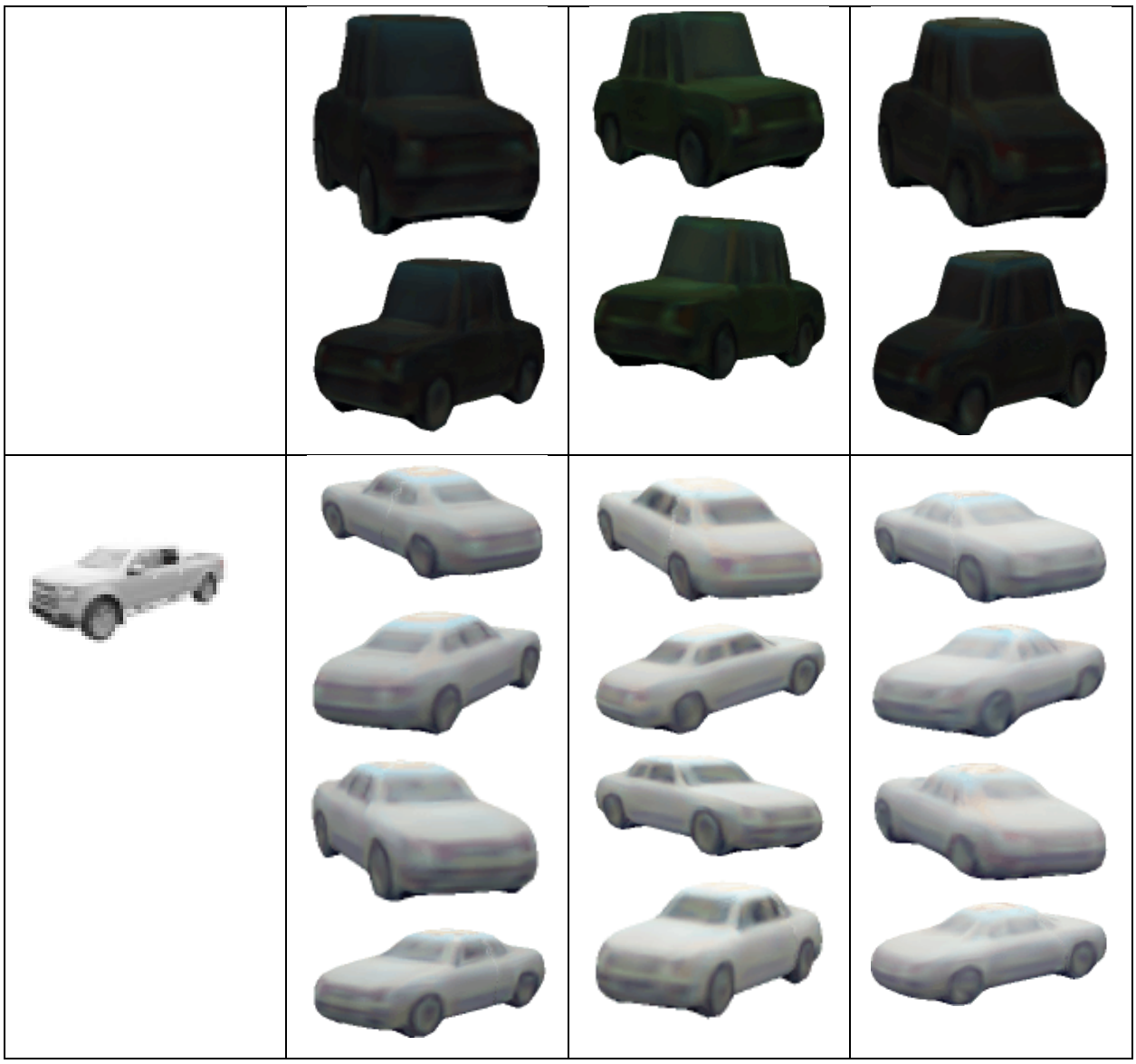
Оригинал	Модель 1	Модель 2	Модель 3
			



Примерны неточной реконструкции обученных моделей

Оригинал	Модель 1	Модель 2	Модель 3
			
			



Как можно увидеть ретро автомобиль и примитивный автомобиль, восстановлены не корректно. А последний в таблице автомобиль изменил форму кузова в процессе восстановления, что тоже не совсем корректно.

1.4.1 Выводы

По итогу обучения трех моделей с различными параметрами можно сделать следующие выводы:

- Модели хорошо восстанавливаю форму распространенных в датасете изображений, но допускают ошибки на нестандартных моделях. Увеличение разнообразия моделей позволило бы устранить этот недостаток. На ранних этапах обучения наблюдались артефакты восстановления формы на границах объектов, что в текущей версии устранено.
- Все модели неплохо справляются с восстановлением текстур.

- Моделям не хватает детальности (выступающие элементы отсутствуют, а элементы дизайна сглажены), хотя мы и старались повысить детальность восстановленных изображений, но результат все еще оставляет желать лучшего.
- Увеличение на порядок normal_weight и uniform_weight не позволили добавить детальности изображениям.

В целом можно резюмировать, что модель справляется с восстановлением формы, но все еще требует доработок.

Глава 2. DreamFusion для 3d reconstruction

2.1. Text-to-3d

Результаты и модели из смежной области, а именно из области генерации 3d объектов с условием в виде текста могут быть полезны в задаче 3d реконструкции. 3D Модели которые генерируются таким образом (как и 2d картинки, генерируемые 2d генераторами) имеют как недостатки, так и преимущества.

Основным преимуществом является качество и детализированность модели, а также способность генерировать объекты различных классов и форм без дополнительного пред-обучения на категориальном датасете (в отличии от классической single-view 3d реконструкции, где требуется достаточных размеров категориальный датасет), т. к. зачастую используются “знания” 2d генераторов, которые обучались на огромном количестве изображений. Иными словами, используется подход дистилляции знаний, т. е. для обучения одной сравнительно небольшой модели (в случае text-to-3d это имплицитная 3d репрезентация) используется другая уже обученная на большом количестве данных большая модель. Однако если мы хотим получить определенную модель, например, модель машины похожую на машину с фотографии или рисунка, то вероятностная природа этих моделей и сложность генерации исчерпывающего текстового описания желаемого объекта не позволит получить нужный результат, т.е. решить задачу 3d реконструкции.

2.2 Stable-dreamfusion

Мы рассмотрим одну из новейших моделей в этой области - DreamFusion [2], а также технику Dreambooth [5] для персонализации генеративной модели Stable-Diffusion [1], которая используется в публично доступной реализации DreamFusion [6], позволяющие обойти ограничения text-to-3d моделей и использовать их для задачи 3d реконструкции.

Также мы предложим способ объединить 3d reconstruction модель UNICORN с text-to-3d моделью DreamFusion для использования преимуществ и преодоления недостатков этих моделей для решения задачи 3d реконструкции.

Модель DreamFusion состоит из двух частей: основная модель, с помощью которой происходит рендеринг изображений (имплицитная 3d модель) и генератор изображений объекта, который называется guidance. Общий алгоритм не формально можно описать следующим образом:

1. На вход алгоритму подается текстовое описание объекта и другие гипер-параметры, регулирующие разрешение рендеринга, количество итераций и другие аспекты моделирования.

2. Далее для заданного количества итераций повторяются следующие шаги

a. Выбирается случайное расположение камеры в рамках заранее заданных пределов (например, в сферической системе координат с помощью зенитного и азимутального углов (далее θ и φ)) выбирается положение камеры и генерируется матрица внешних параметров камеры, в то время как матрица внутренних параметров задана по умолчанию (используется модель pinhole камеры).

b. После этого для сгенерированного положения камеры с помощью имплицитной 3d модели производится дифференцируемый рендеринг (как правило в небольшом разрешении, например 64 x 64 пикселя, для соблюдения не очень больших требований по видеопамяти, которая требуется для обучения модели).

c. Далее положение камеры классифицируется как переднее, боковое или заднее (front, side, back view). Это информация конкатенируются с начальным текстовым описанием (например “red sport muscle car side view”). Полученное текстовое описание переводится токенизируется и представляется в виде эмбеддингов, после чего вместе с полученным с помощью ренедеринга изображением подается на вход guidance модели и считается основной компонент функции потерь (SDS loss). Интуитивно, эта функция добавляет к изображению случайное количество шума, соответствующее временному шагу t (параметр модели guidance, выбирается случайно), и оценивает направление обновления, которое следует за функцией оценки модели диффузии, чтобы перейти к области повышенной плотности.

d. Также к SDS loss (1) добавляются различные регуляризации. При этом обновляются веса только имплицитной 3d модели, а модель guidance остается неизмененной.

$$L_{sds} = E_t[\sigma_t \, KL(q(z_t|g(\theta)l; y, t)||p_\phi(z_t; y, t))] \quad (1),$$

где z_t - это скрытое представление диффузии на шаге t, y - текстовое описание, g - генератор (3d имплицитное представление), KL - дивергенция Кульбака — Лейблера, которую можно интерпретировать как “расстояние” между распределениями.

Схематично обучение можно представить в следующем виде (см. Рисунок 10).

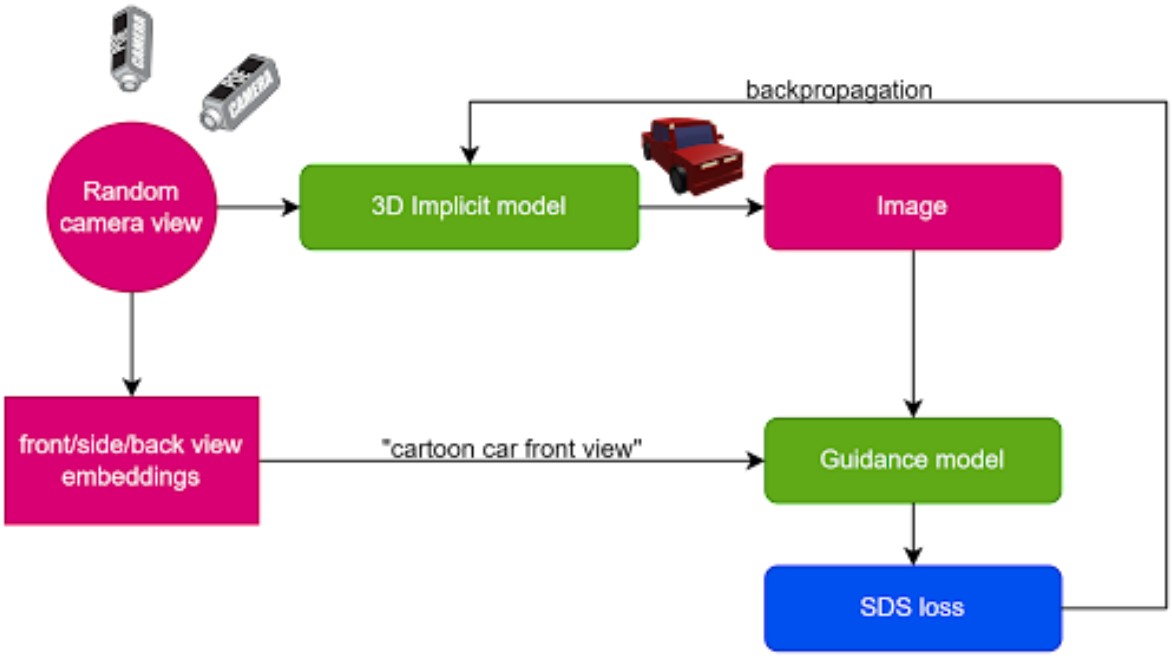


Рисунок 10. Алгоритм обучения DreamFusion

2.3 Имплицитная 3d модель: NeRF

В качестве имплицитной 3d модели выступает NeRF [3] (neural radiance field) - в последние годы один из самых популярных методов для решения различных 3d задач. Суть обучения оригинальной модели NeRF заключается в последовательной трассировке лучей из центра камеры через 3d пространство сцены и передачи сэмплированных точек (координаты x; y; z) вместе с соответствующими углами θ и φ, определяющими положение камеры, которые подаются на вход нейронной сети (обычный многослойный персептрон (MLP)), которая выдает на выходе цвета и плотности для каждой точки, после чего используются классические техники объёмного (volume rendering) рендеринга изображения.

Плотности интерпретируются как вероятность прерывания данного луча в в окрестностях точки (x, y, z). Ожидаемое значение цвета выпущенного луча r выражается как интеграл, который численно оценивается следующей формулой (2).

$$\hat{C} = \sum_{i=1}^{N} \quad T_i(1 - exp(-\sigma_i \delta_i))c_i \quad (2),$$

где c_i - цвет, σ_i - плотность, δ_i - расстояние между соседними точками луча, а T_i - кумулятивная вероятность того, что луч проследует от начальной точки до текущей (подробнее см. в статье) без столкновения с какой-либо частицей, σ_i и c_i оцениваются

нейросетью. Функция потерь - L_2 норма между предсказанным цветом и ground truth. Легко убедится, что такая функция потерь дифференцируема.

Минимизируя функцию средних квадратичных отклонений, модель обучается рендерить изображения с разных углов, являясь таким образом имплицитной моделью 3d объекта. Входные данные кодируются специальным кодировщиком, который изменяет 5-ти мерный вход (x, y, z, θ, φ) с помощью множества тригонометрических функций вида $sin(2^0\pi x), cos(2^0\pi x) \dots sin(2^n\pi x), cos(2^n\pi x)$, которые применяются последовательно к каждой переменной. Этот трюк позволяет нейросети параметризовать более высокочастотные (high-frequency) функции. Без этого этапа изображения получаются смазанными и нечеткими.

В публичной реализации DreamFusion используется улучшенный подход Instant-NGP [4] (реализация torch), в котором процесс кодирования входных данных улучшен, что позволяет быстрее обучать NeRF, не теряя в качестве репрезентации. Процесс кодирования заключается в кодировании координат пикселя с помощью хэш-таблицы, для каждого значения которой существует вектор переменных, описывающий это значение.

Также вместо обычной инференции NeRF используются дополнительные техники шейдинга (симуляция влияния света на восприятие цвета): RGB цвет на выходе нейросети интерпретируется как цвет материала ρ. Для получения итогового цвета вычисляется вектор нормали, описывающий локальную геометрии объекта. Этот вектор к поверхности может вычисляться нормированием отрицательного градиента плотности τ относительно трехмерной координаты μ (x ; y; z) : $n = -\nabla_\mu \tau / ||\nabla_\mu \tau||$. Итоговый цвет вычисляется по следующей формуле (3).

$$c = \rho \circ (\lambda_\rho \circ max(0, n \circ (\lambda - \mu)/||\lambda - \mu||) + \lambda_a) \quad (3),$$

Где λ - Это 3d координата точки источника цвета, λ_ρ - цвет этой точки, а λ_a - свет окружения. Также иногда случайным образом ρ заменяется белым цветом (1;1;1) чтобы получить “безтекстурный” рендеринг с шейдингом, что помогает модели избегать вырожденных решений, когда вместо 3d объекта получается плоское изображение в 3d пространстве.

2.4 Guidance: stable-diffusion

В качестве guidance модели используется публично доступная обученная модель Stable-DreamFusion [1] (в оригинальной статье используется Imagen). Процесс обучения происходит следующим образом. К входным картинкам, сжатым кодировщиком до

размерности латентного пространства, для которых известно текстовое описание, применяется несколько шагов зашумления (процесс диффузии, являющийся марковским случайным процессом), при этом разрешение изображения уменьшается. Обратной задачей является восстановление изображения при известном текстовом описании таким же образом, т.е. итерационно, с помощью U-Net, усиленной механизмом cross-attention предсказывается шум, который был добавлен к изображению на соответствующем шаге и вычитается из него. На последнем шаге декодировщик переводит вектор размерности латентного пространства в итоговое изображение. Используется MSE функция потерь (4).

$$L_{LDDM} = E_{\varepsilon(x),\epsilon\sim N(0,1),t}[(\epsilon - \epsilon_\theta(z_t, t, \tau_\theta(y)))^2] \text{ (4)},$$

где ϵ - случайный нормальный шум, z_t сжатое зашумленное представление изображение на шаге t, τ - закодированное представление текстового описания y.

Существует несколько обученных публично доступных моделей stable-diffusion, которые можно до-обучать для наших целей. Ввиду ограниченных ресурсов проекта мы будем использовать чуть более раннюю версию 1.5 [7], которая обучена для меньшего разрешения изображений (512x512), что позволяет сэкономить на видеопамяти и проводить до-обучение на видеокартах с 12GB видеопамяти.

2.5 Эксперименты

Для того, чтобы модель была способна генерировать необходимые объекты, имея небольшое количество изображений этого объекта (в нашем случае только одно), существует способ до-обучить guidance модель stable-diffusion с помощью техники Dreambooth, передав специальный токен (он может быть произвольным, но важно подбирать такой токен, который бы не ассоциировался у до-обучаемой модели с каким то объектом “очень сильно”, однако лучше подбирать токен из словаря т.к. не имеющийся в словаре токен модель может токенезировать на такие токены, которые имеют сильную ассоциацию с какими то объектами) и обобщенное название объекта, например класс (например, имея фотографию машины, можно использовать следующий текст: “sks car” или “a photo of sks car”). Существует несколько вариантов обучения: аналогичное стандартному обучению stable diffusion обучение под с MSE функцией потерь, но только с предоставленными изображениями и текстом, и обучение с prior preservation loss (можно перевести как сохранение априорного знания модели), когда перед обучением генерируется несколько сотен изображений класса и модель одновременно обучается на переданных

изображениях объекта и этих сгенерированных изображениях (которые в качестве условия имеют текст класса). Второй подход требуется во избежание переобучения модели, когда она выдает те же самые изображения которые получила на вход без каких либо изменений, что является актуальным для нашей задачи т. к. мы имеем только одно изображение объекта и при переобучении модель не будет иметь никаких ракурсов кроме переднего вне зависимости от вида (front, side или back).

Для подбора необходимых параметров мы выбрали несколько машин и одно ее изображение. Насколько мы осведомлены, это первая попытка провести подробное исследование применения Dreambooth вместе с DreamFusion

В ходе экспериментов мы выяснили несколько важных аспектов для обучения.

1. Во-первых требуется небольшое количество итераций Dreambooth. В противном случае модель переобучается и не способна выдавать альтернативные ракурсы, которые необходимы для обучения DreamFusion (см. таблицу). В случае слишком маленького количества итераций модель способна генерировать разные ракурсы, однако изображения слишком далеки от оригинального (см. Таблицу 2.1). При оптимальном количестве итераций до-обучение занимает примерно 30 минут на видеокарте GTX1080TI, занимая примерно 10GB видеопамяти при половинной точности (16 bit float point) и других оптимизациях.

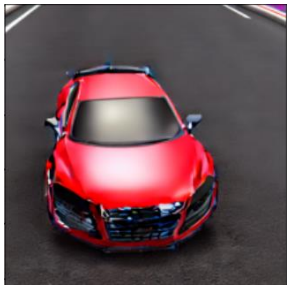
	Переобучение (800 итераций)	Недообучение (200 итераций)	Правильное обучение (400 итераций)
front view			
back view			
side view			

Таблица 2.1 Примеры переобучение и недо-обучения stable-diffusion

2. Во-вторых, непосредственно во время обучения dreamfusion, параметр guidance-scale, который регулирует степень следования заданному тексту (чем больше этот параметр, тем более строго генератор придерживается текста, степень случайности и вероятность появления неожиданных объектов уменьшаются), должен быть не слишком низким и не слишким высоком (оптимальное значение колеблется от 30 до 70 (проверялся guidance scale от 10 до 130 с шагом 10) для разных изображений). В противном случае модель не сможет либо выучить какую форму совсем (пустое изображение) либо разные ракурсы (один и тот же вид с разных углов) (см. Таблицу 2.2)..

3. В тоже время интересно, что способ обучения Dreambooth достаточно сильно влияет на результаты моделирования, как ожидалось, т.е. prior preservation loss для класса объекта оказывает положительное влияние, однако и без prior loss preservation модель способна создавать правдоподобные модели. Остальные параметры, которые нам удалось

проверить, оказываются не такими значимыми и либо ухудшают либо не дают каких либо заметных изменений по сравнению с параметрами по умолчанию.

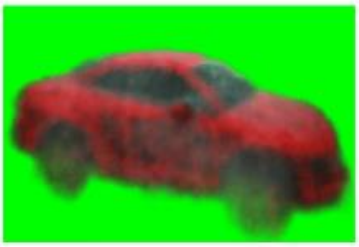
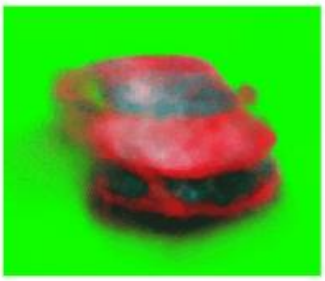
	Слишком высокий guidance scale	Правильно подобранный guidance scale
Вид спереди		
Вид сбоку		
Вид сзади		

Таблица 2.2 Пример удачного и неудачного обучения DreamFusion

Из результатов можно сделать вывод о том, что модель способна применять знания Dreambooth, однако качество рендеринга с точки зрения гладкости поверхности и разрывов оставляет желать лучшего. Также для того, чтобы получить подобный результат требуется обучать модель на tesla t4 16gb около двух часов в течении 150 итераций. Учитывая, что модель чувствительна к параметру guidance scale, на получение качественной модели может понадобиться до 24 часов.

Также мы предлагаем способ совмещения моделей UNICORN и DreamFusion, а именно пред-обучение NeRF из DreamFusion с помощью картинок сгенерированных UNICORN для конкретного объекта. Мы предполагаем, что данная предобработка может

существенно сократить время обучения DreamFusion и добавить ровности результатам, т. е. быть своеобразным регуляризатором. Идея заключается в том. что мы генерируем большое количество картинок с помощью UNICORN для различных позиций камеры в той же системе координат, что применяется и при основном обучении DreamFusion, а далее обучаем NeRF из DreamFusion для рендеринга этих картинок с помощью функции потерь MSE. Таким образом в начале основного обучения DreamFusion у имплицитной модели есть априорное знание о форме объекта.

Данный подход требует аккуратного подбора позиций камеры при пред-обучении, чтобы передние, задние и боковые позы примерно совпадали с параметрами при основном обучении. Отличие от обычной схемы обучения DreamFusion можно видеть на диаграмме. Шестигранниками выделены части обучения, которые происходят до основного обучения и позволяют решать text-to-3d модели решать задачу 3d реконструкции по заданному изображению (см. Рисунок 11).

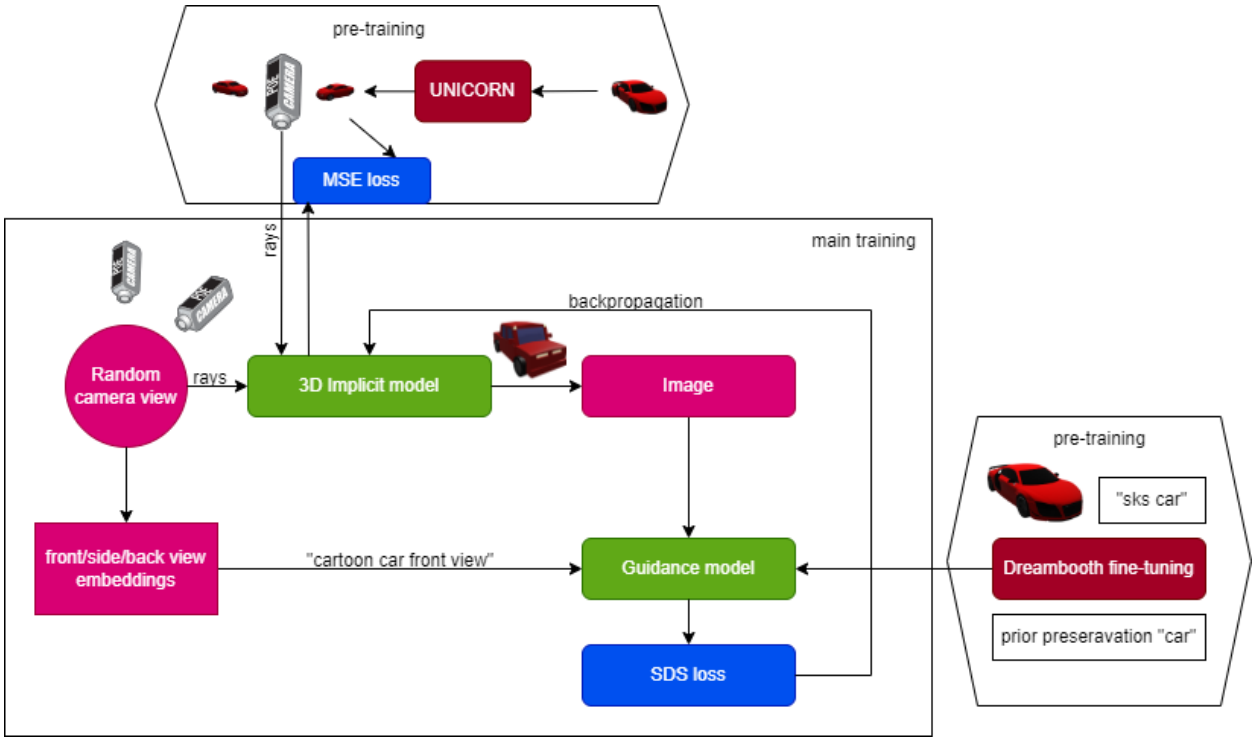


Рисунок 11. Пред-обучение DreamFusion

Сравнение результатов можно видеть в таблице ниже (см. Таблицу 2.3). В таблице представлены результаты рендеринга с разных ракурсов для лучших моделей. найденных для соответствующего способа пред-обучения. Код, ссылки на докер-файлы, которые позволяют создать окружение, готовое для обучения (требуется Docker, а также все необходимые дополнения от nvidia для Docker), инструкцию по воспроизведению

некоторых результатов и демонстрацию результатов в виде gif файлов можно найти на GitHub по ссылке: https://github.com/ToxaSurkov/dreamfusion_pretrain.git.

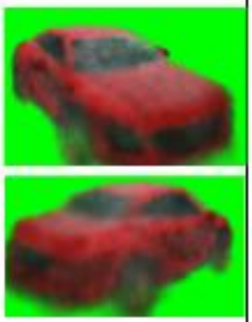
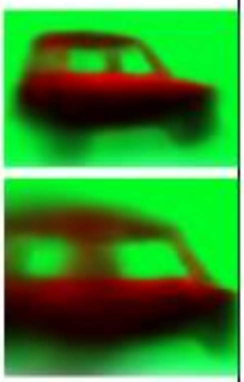
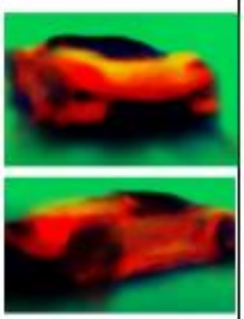
Оригинальное изображение	UNICORN	150 итераций Dreambooth	150 итераций Dreambooth + prior preservation loss	150 итераций Dreambooth + prior preservation loss + пред-обучение NeRF
				
				
				
				

Таблица 2.3 Сравнительная таблица результатов обучения

Можно заметить, что в таблице приведены довольно реалистичные современные машины, т. к. для других видов машин модели не удается генерировать невырожденные решения. Это объясняется тем, что stable-diffusion 1.5 достаточно сложно генерировать альтернативные ракурсы для произвольных машин. Также из таблицы видно, что пред-обучение nerf не дает принципиально нового уровня качества однако позволяет сэкономить время обучения, т.к. пред-обучение занимает примерно 5 минут, в то время как дополнительные 100 итераций stable-DreamFusion обходятся на порядок дороже - примерно 60 минут.

Тем не менее, несмотря на нестабильность и недостатки, в случае с нашими данными, предложенный подход справляется лучше чем альтернативный вариант обучения Image-to-3d [8], доступный в репозитории Stable-DreamFusion, который решает задачу 3D реконструкции (см. Таблицу 2.4).

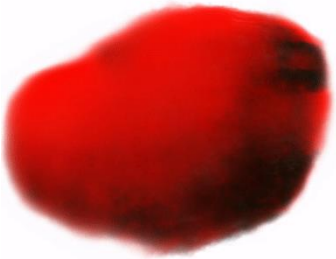
	Dreambooth + пред-обучение NeRF	Zero-1-to-3D
Вид спереди		
Вид сбоку		

Таблица 2.4 Сравнение нашего подхода и Zero-1-to-3D

2.6. Выводы

Итак, исходя из наших экспериментов можно сделать следующие выводы:

1. Возможно применять text-to-3d модели, а именно stable-DreamFusion, для задачи 3d реконструкции. Для этого можно использовать технику до-обучения guidance модели stable-diffusion - Dreambooth, для которой требуется подбирать необходимое

количество итераций во избежание недообучения или переобучения модели. Наиболее важным параметром для обучения является параметр guidance scale.

2. Пред-обучение 3d имплицитной модели Nerf может ускорить обучение Stable-DreamFusion, однако не позволяет достичь принципиально другого уровня качества.

3. 3d модели получаются гораздо более детализированные нежели в случае с моделями 3d реконструкции, однако подбор параметров и обучение для одной модели может занимать большое количество времени (сутки и более), что является большим минусом по сравнению с моделями 3d реконструкции, которые обучаются дольше, однако после обучения могут быть использованы для реконструкции любых подходящих объектов за секунды. Также для многих машин нам не удалось подобрать параметры, позволяющие получить хорошую 3d модель.

Таким образом, можно сделать вывод, что предложенный нами подход является перспективным и уже способен демонстрировать хорошие результаты, однако ограничение в вычислительных ресурсах не позволило нам проверить все возможные параметры, например, разрешение рендеринга для NeRF: возможно, более высокое разрешение позволит быстрее обучать более детализированные модели и увеличить стабильность обучения. Также помимо этого, мы предлагаем следующие направления исследований и доработок, которые могут быть интересны исследователям:

1. Альтернативные способы до-обучения stable-diffusion, которые позволят сохранять/усиливать возможность модели генерировать различные ракурсы одного и того же объекта. Также до-обучение text-encoder во время процедуры до-обучения DreamBooth (наши вычислительные мощности не позволили провести такие эксперименты).

2. Создание новой guidance модели, которая будет в лучше степени способна генерировать разные ракурсы объектов имея одно изображение объекта. Недавнее появление такой базы 3D данных как objaverse может способствовать исследованиям в данном направлении.

3. Использование других NeRF, например BARF, который является менее чувствительным к плохо специфицированным параметрам камеры для конкретного вида.

Все эти эксперименты требуют большого количества вычислительных мощностей и времени исследователей, однако мы надеемся, что наша работа может стать одним из первых шагов к созданию моделей 3D реконструкции нового уровня.

Заключение

Итак, все поставленные нами в начале работы задачи достигнуты:

1. Мы собрали датасет, который может быть в дальнейшем использован исследователями для задачи 3D реконструкции и различных смежных задач.
2. Также на собранном датасете мы обучили одну из последних 3D reconstruction моделей - Unicorn, и улучшили base-line модель.
3. Мы адаптировали text-to-3d модель DreamFusion для задачи 3D реконструкции и провели серию экспериментов с различными параметрами.

В ходе наших экспериментов мы смогли подтвердить первые две гипотезы: в то время как модель UNICORN показала большую стабильность на нашем датасете, но смогла восстановить мельчайшие детали автомобилей, адаптированная для задачи 3D реконструкции модель DreamFusion была гораздо менее стабильной и часто не позволяла получить невырожденные решения, но была способна восстановить подробные детали в случае невырожденных решений.

Мы не смогли подтвердить третью гипотезу о том, что пред-обучение модели DreamFusion с помощью модели UNICORN может преодолеть проблемы обеих моделей, т.к. предложенная нами схема пред-обучения NeRF не дала существенного увеличения стабильности.

Таким образом, в завершении нашей работы можно сказать, что в области 3D реконструкции существует еще много нерешенных проблем как с данными, так и с методами моделирования, однако в последние годы данная область значительно продвинулась на пути к получению качественно новых результатов, и наша работа вносит вклад в решение этих проблем.

Список источников

1. Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. arXiv preprint, arXiv:2112.10752v2, version 2. 2022.
2. Ben Poole, Ajay Jain, Jonathan T. Barron, Ben Mildenhall. DreamFusion: Text-to-3D using 2D Diffusion. arXiv preprint, arXiv:2209.14988v1, version 1. 2022.
3. Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. arXiv preprint, arXiv:2003.08934v2, version 2. 2020.
4. Thomas Muller, Alex Evans, Christoph Schield, Alexander Keller. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. URL: https://nvlabs.github.io/instant-ngp/assets/mueller2022instant.pdf
5. Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, Kfir Aberman. DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation. arXiv preprint, arXiv:2208.12242v2, version 2. 2022.
6. Kiui-Jiaxiang Tang. Stable-Dreamfusion. URL:https://github.com/ashawkey/stable-dreamfusion.
7. Stable Diffusion v1-5. URL: https://huggingface.co/runwayml/stable-diffusion-v1-5.
8. Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, Carl Vondrick. Zero-1-to-3: Zero-shot One Image to 3D Object. arXiv preprint, arXiv:2303.11328v1, version 1. 2023.
9. Xian-Feng Han, Hamid Laga, Mohammed Bennamoun. Image-based 3D Object Reconstruction: State-of-the-Art and Trends in the Deep Learning Era. IEEE Transactions on Pattern Analysis and Machine Intelligence, Nov. 2019.
10. J. Wu, Y. Wang, T. Xue, X. Sun, B. Freeman, and J. Tenenbaum. MarrNet: 3D shape reconstruction via 2.5D sketches. NIPS, 2017.
11. C. Hane, S. Tulsiani, and J. Malik, “Hierarchical Surface Prediction” IEEE PAMI, no. 1, pp. 1–1, 2019.
12. G. Riegler, A. O. Ulusoy, and A. Geiger, “OctNet: Learning deep 3D representations at high resolutions,” in IEEE CVPR, vol. 3, 2017.
13. M. Tatarchenko, A. Dosovitskiy, and T. Brox, “Octree generating networks: Efficient convolutional architectures for highresolution 3D outputs,” in IEEE CVPR, 2017, pp. 2088–2096.

14. N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, “Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images,” in ECCV, 2018.
15. X. Z. Xingyuan Sun, Jiajun Wu and Z. Zhang, “Pix3D: Dataset and Methods for Single-Image 3D Shape Modeling,” in IEEE CVPR, 2018.
16. Tom Monnier, Matthew Fisher, Alexei A. Efros, Mathieu Aubry. Share With Thy Neighbors: Single-View Reconstruction by Cross-Instance Consistency. arXiv:2204.10310, version v3, 25 Jul 2022.
17. OpenCv. URL: https://opencv.org/