

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
1. Задача об условной генерации интерьеров и методы ее решения	7
1.1. Термины и определения	7
1.2. Постановка задачи	8
1.3. Существующие решения и их недостатки	9
1.4. SceneFormer	10
1.5. Atiss	12
Выводы по главе 1	14
2. Предложенное решение поставленной задачи	15
2.1. Создание синтетического тренировочного набора данных	15
2.2. Генерация изображений с ортогональными проекциями помещений с мебелью	16
2.3. Визуализация комнат с мебелью в трехмерном пространстве	17
Выводы по главе 2	18
3. Детали реализации предложенного решения	19
3.1. Этапы создания синтетического набора изображений	19
3.1.1. Получение изображений с помощью библиотеки Trescope	20
3.1.2. Реализованный метод получения изображений	22
3.2. Генерация изображений на основе уже существующих	25
3.2.1. Модель OASIS	26
3.2.2. Модель SEAN	27
3.3. Трехмерная визуализация	28
3.3.1. Распознавание объектов на изображении с проекцией комнаты	29
3.3.2. Рендеринг	32
Выводы по главе 3	34
4. Экспериментальное сравнение	35
4.1. Генерация изображений	35
4.2. Сравнение с существующими решениями	35
Выводы по главе 4	38
ЗАКЛЮЧЕНИЕ	39
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	40
ПРИЛОЖЕНИЕ А. Трехмерные визуализации комнат	42

ВВЕДЕНИЕ

Проблема создания трехмерных и реалистичных интерьеров жилых помещений существует в течение длительного промежутка времени. На данный момент, автоматическая генерация трехмерных представлений помещений и другого рода объектов имеют довольно широкую область применения, в связи с постоянным ростом использования таких визуализаций как и в сферах виртуальной и дополненной реальности, так и в других задачах, требующих создания трехмерной среды.

Люди проводят значительную часть своей повседневной жизни внутри различных жилых помещений: в спальнях, кухнях, гостиных. Поэтому, несомненно, важной задачей является создание дизайнерских решений для вышеописанных помещений. При этом, данные решения должны удовлетворять набору требований, установленному заказчиком дизайн-проекта заранее. Чтобы выполнить такого рода задачу требуется привлечение квалифицированных специалистов в этой области, которые обладают знаниями в дизайне интерьеров и навыками использования графических приложений для последующей визуализации. Более того, существенным недостатком такого подхода является тот факт, что процесс создания дизайн-проекта помещения является крайне дорогостоящим как в финансовом аспекте, так и в плане затраченного времени.



Рисунок 1 – Пример трехмерного дизайн-проекта спальни [1].

Цель настоящей работы заключается в создании сервиса, способного выполнять условную генерацию интерьеров с помощью методов машинного обучения (автоматически генерировать реалистичную расстановку указанной мебели внутри требуемого помещения с его последующей трехмерной визуализацией). Перед тем, как приступить к разработке вышеописанного сервиса, был сформулирован ряд подзадач, поэтапная реализация которых требовалось для создания финального решения:

1. Поиск и подготовка набора данных, представляющего собой множество из трехмерных моделей различной мебели, используемой в жилых помещениях.
2. Создание синтетического набора данных, представляющего собой множество из изображений жилых помещений с мебелью. Все изображения должны придерживаться ряда строгих правил, чтобы впоследствии была возможность корректно их использовать.
3. Создание и обучение модели машинного обучения, способной генерировать изображения жилых помещений с расстановкой требуемой мебели по заданной форме жилого помещения.
4. Разработка алгоритма, способного по изображению жилого помещения с мебелью построить его трехмерную визуализацию. При этом:
 - Пользователь должен иметь возможность совершать навигацию в трехмерном пространстве, чтобы рассматривать помещение и мебель, находящуюся внутри него, с разных ракурсов;
 - Пользователь должен иметь возможность менять конфигурацию визуализации — включать или отключать отображение стен, задавать собственный цвет для стен помещения, задавать собственную текстуру для пола помещения и т.д.;
 - Алгоритм должен подбирать различную мебель, соответствующей мебели, полученной на этапе генерации расстановки, и стилю, который задал пользователь сервиса.

ГЛАВА 1. ЗАДАЧА ОБ УСЛОВНОЙ ГЕНЕРАЦИИ ИНТЕРЬЕРОВ И МЕТОДЫ ЕЕ РЕШЕНИЯ

В настоящей главе приведена постановка задачи об условной генерации интерьеров и предложено решение. Помимо этого, рассмотрены существующие методы решения данной задачи, являющиеся наиболее актуальными на момент реализации предложенного решения, а также их недостатки. Для методов, показавших наиболее качественные результаты в рамках решения данной задачи, приведено более подробное описание их архитектуры, алгоритма создания расстановки мебели и изображения, иллюстрирующие интерьеры, созданные данными методами. Также в данной главе представлены определения для терминов, фигурирующих в настоящей работе.

1.1. Термины и определения

1. **Порождающие (генеративные) модели** — класс моделей, обучающих совместное распределение данных $p(x, y)$.
2. **Авторегрессионная модель** — модель временных рядов, в которой значения временного ряда в данный момент линейно зависят от предыдущих значений этого же ряда.
3. **Трансформер** — архитектура глубоких нейронных сетей, в которой используется механизм самовнимания, по-разному оценивающий важность каждой части входных данных.
4. **Сверточная нейронная сеть** — особая архитектура нейронных сетей, которая изначально нацелена на эффективное распознавание изображений и образов.
5. **Датасет** — набор данных, обработанная и структурированная информация, представленная в некотором виде.
6. **Генеративная состязательная сеть** — алгоритм машинного обучения, являющийся порождающей моделью. Генеративная состязательная сеть состоит из двух нейронных сетей: генеративная модель G , которая строит приближение распределения данных, и дискриминативная модель D , оценивающая вероятность, что образец пришел из тренировочных данных, а не был сгенерирован моделью G .
7. **Image-to-image translation** — задача, заключающаяся в трансформации изображений, принадлежащих определенной области, в другие изобра-

жения так, чтобы полученные изображения принадлежали новой области определения.

8. **Mesh** — совокупность вершин, ребер и полигонов, задающих определенный трехмерный объект.
9. **Рендеринг** — процесс получения изображения по модели с помощью компьютерной программы.
10. **RGB** — цветовая модель, описывающая способ кодирования цвета посредством трех основных цветов (красный, зеленый, синий).
11. **Grayscale** — способ представления изображения в одноканальном формате, где каждое значение задает определенный оттенок серого.
12. **HSV** — цветовая модель, в которой координатами цвета являются *hue* (цветовой тон), *saturation* (насыщенность) и *value* (значение цвета).
13. **Латентное пространство** — эмбеддинг (вложение) некоторого набора элементов внутри многообразия, в котором элементы, схожие друг с другом, находятся близко друг к другу.
14. **Алгоритм отрезания ушей** — один из алгоритмов триангуляции выпуклого многоугольника.
15. **Метод k -ближайших соседей** — метрический алгоритм для автоматической классификации объектов или регрессии [2].
16. **Расхождение Кульбака-Лейблера** — неотрицательнозначный функционал, являющийся несимметричной мерой удалённости друг от друга двух вероятностных распределений, определённых на общем пространстве элементарных событий [3].
17. **FID** — Frechet inception distance, метрика для оценки качества изображений, созданных генеративными моделями.

1.2. Постановка задачи

В настоящей работе поставлена следующая задача: разработать сервис, способный по заданной пользователем комнате создать ее трехмерную визуализацию, добавив в комнату указанную в условии мебель. Для того, чтобы предоставить более наглядное представление интерьера, пользователю должна предоставляться возможность рассматривать созданную визуализации интерактивно. Генерация расстановки мебели внутри комнаты является ключевой подзадачей в настоящей работе. Формулировка данной подзадачи может варьироваться в зависимости от ее входных и ожидаемых выходных данных.

Однако, можно обобщить формулировку задачи следующим образом: требуется разработать алгоритм, который, имея в качестве входных данных — информацию о форме комнаты (ортогональная проекция комнаты на плоскость ее пола) и расположении окон и дверей внутри нее, должен генерировать информацию о расстановке мебели — в качестве выходных данных. Каждый элемент расстановки должен содержать некоторую информацию о себе, а именно — координаты и вектор ориентации, однозначно задающих расположение элемента в трехмерном пространстве.

Важно отметить, что решение данной задачи имеет ряд ограничений. Возможно, самым существенным ограничением является тип комнаты - в подавляющем большинстве случаев это жилые помещения, такие как спальни и гостиные комнаты. Применять описанный ранее алгоритм для, например, генерации расстановки мебели внутри кухни не является целесообразным, ведь кухонный гарнитур, хоть и состоит из некоторых модулей, является одним цельным объектом, занимающим большую часть помещения, и не обладает многочисленными вариантами расположения. Поэтому генерация расстановки мебели для такого рода помещений является тривиальным случаем, который можно выделить в отдельную подзадачу, не обладающей высокой сложностью реализации. Подробнее об ограничениях рассмотрено в разделе 1.3.

1.3. Существующие решения и их недостатки

Работы в этой области решают поставленную задачу разными способами. Были предложены *порождающие модели*, которые решают задачу о генерации расстановки мебели путем создания графов [10], в чьих вершинах содержится информацию об элементах расстановки, а на ребрах — семантические связи между элементами. Также существует ряд работ [11, 15], которые представляют комнату в виде упорядоченной последовательности объектов мебели. Основываясь на *сверточных нейронных сетях* или *трансформерной архитектуре*, такие подходы последовательно добавляют объект мебели внутрь создаваемой комнаты (при этом порядок добавления может задаваться, например, вероятностью появления определенного типа мебели в комнате, начиная от наибольшей и заканчивая наименьшей). Одно из последних решений [7] предлагает отказаться от рассмотрения расстановки мебели в комнате как упорядоченной последовательности и сделать ее неупорядоченной, избавляясь от

возможных ограничений на генерацию менее часто встречающихся типов мебели.

Возможности генерации мебели для этих подходов различаются. Например, в методе, предложенном в [15], есть возможность генерации настенных объектов, когда в подходе, изложенном в [7], есть возможность добавлять наиболее подходящие объекты внутрь комнаты в указанной пользователем области. Однако, важно понимать, что все вышеописанные подходы, несмотря на значительные успехи в создании реалистичных комнат с расставленной мебелью, в значительной степени зависят от набора данных, используемого при обучении модели. Для их корректного обучения требуется такой набор данных, в котором для каждого помещения, помимо его формы и информации про стены и окна, обязательно присутствовать координаты и ориентация каждого трехмерного объекта мебели, используемого внутри соответствующего помещения. Подавляющая часть работ в качестве обучающего набора данных использовала *датасет SUNCG* [14], являющегося одним из наиболее крупных датасетов, содержащих в себе информацию о синтетических трехмерных помещениях. Но, в связи с обращением правообладателей, он больше не находится в открытом доступе, что приводит либо к невозможности повторного использования и обучения представленных ранее методов для генерации расстановки мебели, либо заставляет реализовывать дополнительный функционал, позволяющий производить интеграцию с другими датасетами. Создание же набора данных такого рода вручную является крайне трудоемким процессом, требующего привлечения специалистов и большого количества времени из-за требования про трехмерные аннотации объектов. Помимо этого, некоторые методы, реализующие генерацию интерьеров, не обладают возможностью создавать их по заданному условию (типы мебели в комнате, стиль используемой мебели). Таким образом, разработка сервиса, способного создавать трехмерные визуализации интерьеров по заданным заранее условиям и не требующего для обучения информации о расположении объектов мебели в трехмерном пространстве, является актуальной задачей.

1.4. SceneFormer

Метод, предложенный в [15], позволяет создавать трехмерные визуализации интерьеров, которые задаются либо по плану комнаты (чертеж комнаты, т.е. информация о форме пола), либо по текстовому описанию (требуемые объ-

екты мебели и то, как они расположены относительно друг друга). В данном решении комнату рассматривают как последовательность атрибутов мебели, использованный в ней. Таким образом, задача создания интерьера сводится к задаче создания определенной последовательности.

Каждая комната представляется как последовательность объектов мебели, которая упорядочена по частоте f типов мебели c в тренировочном наборе данных. Положение мебели представляется как (x, y, z) , размеры — (l, w, h) , ориентация в пространстве — θ . Таким образом, для каждой комнаты $\{o_i\}$ создается восемь последовательностей $(\{c_i\}, \{x_i\}, \{y_i\}, \{z_i\}, \{\theta_i\}, \{l_i\}, \{w_i\}, \{h_i\})$. Модель, использованная в [15], представлена на рисунке 2. Для генерации атрибутов объекта мебели используется декодер трансформера. Для каждого из атрибутов объекта мебели обучается отдельная модель, предсказывающая соответствующий атрибут мебели. В решении используется авторегрессионный подход: атрибуты текущего объекта предсказываются на основе предсказанных ранее атрибутов.

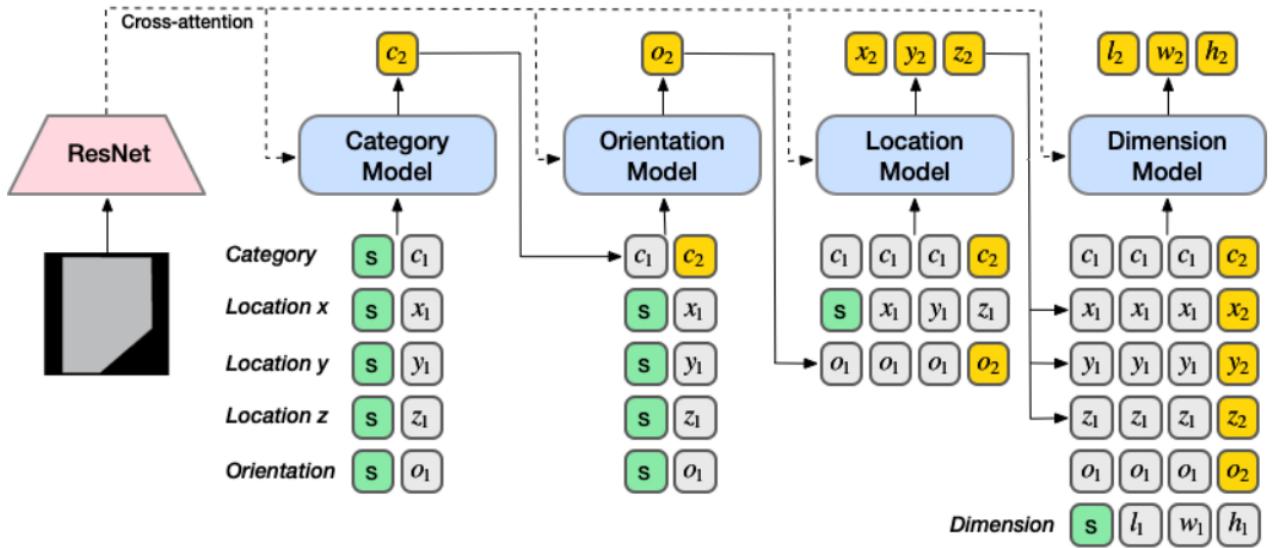


Рисунок 2 – Модель SceneFormer. План комнаты представляется в виде изображения с полом. Токены, обозначающие начало последовательности, изображены зеленым цветом, для существующих объектов — серым, для новых — желтым. Каждая модель принимает на вход три последовательности — тип мебели, ориентацию и координаты в трехмерном пространстве. Выходные данные каждой модели (кроме модели, отвечающей за размеры объекта мебели) добавляются к существующей последовательности перед выводом.

С помощью предложенного решения возможно создавать трехмерные визуализации интерьеров, в которых объекты мебели могут быть установлены не только на полу, но и на стенах комнаты. Используя языковые модели на трансформерной архитектуре, возможно создавать интерьер по его текстовому описанию. Однако, в таком случае невозможно использовать форму пола комнаты в качестве условия, что означает неприменимость данного метода в задачах условной генерации интерьеров, где требуется создавать интерьеры одновременно как и по форме пола, так и по требованиям к необходимой мебели.

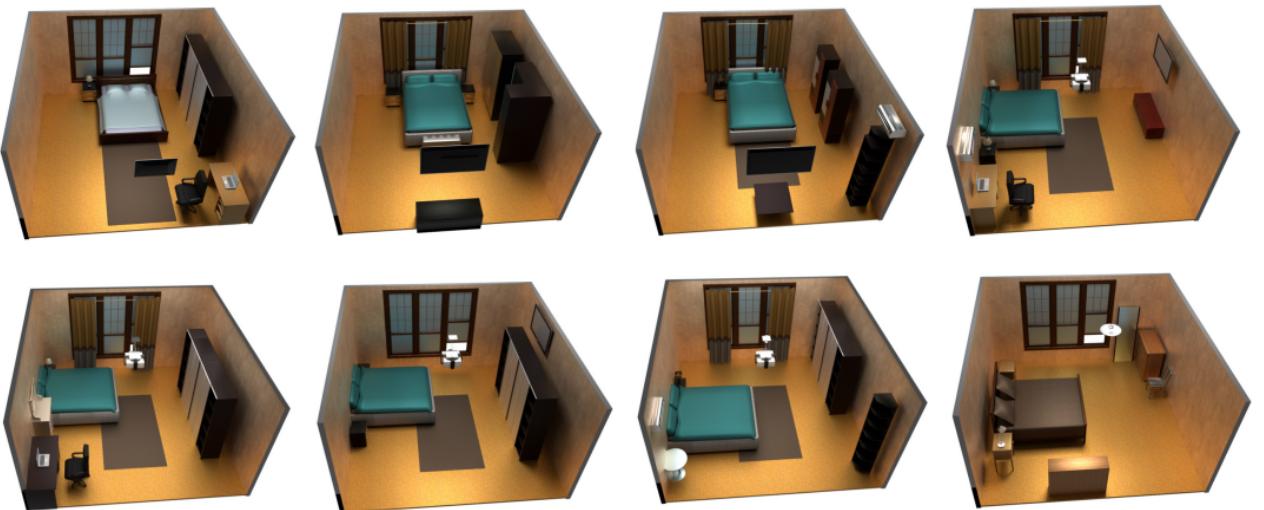


Рисунок 3 – Интерьеры, созданные SceneFormer. На данном рисунке представлена возможность создавать разнообразные расстановки мебели внутри одного и того же помещения.

1.5. Atiss

Решение, предложенное в [7], представляет из себя сервис, основывающийся на авторегрессионной трансформерной архитектуре, способный по плану комнаты создавать ее трехмерную визуализацию, содержащую трехмерные модели мебели. Утверждается, что полученные таким образом расстановки мебели являются разнообразными и правдоподобными.

На этапе обучения выбирается случайная комната из подготовленного набора. Для набора объектов мебели в выбранной комнате строится случайная перестановка, в чем и заключается подход в представлении мебели внутри комнаты как неупорядоченной последовательности. Из перестановки выбирается T первых элементов, на их основе высчитывается контекстный эмбеддинг

C. На основе C и информации о форме пола комнаты F модель предсказывает распределение атрибутов добавляемого в комнату объекта мебели, обучаясь путем максимизации функции максимального правдоподобия от $T + 1$ объекта мебели. Архитектура модели представлена на рисунке 4. Получая на вход форму пола в виде изображения, энкодер кодирует его в набор признаков F . Из последовательности объектов мебели получается контекстный эмбеддинг . Далее, F , C и вектор запроса q поступает на вход энкодеру трансформера, который предсказывает вектор \hat{q} . Используя \hat{q} , предсказывается распределение атрибутов мебели, которые используется чтобы выбрать следующий объект мебели, добавляемый в комнату.

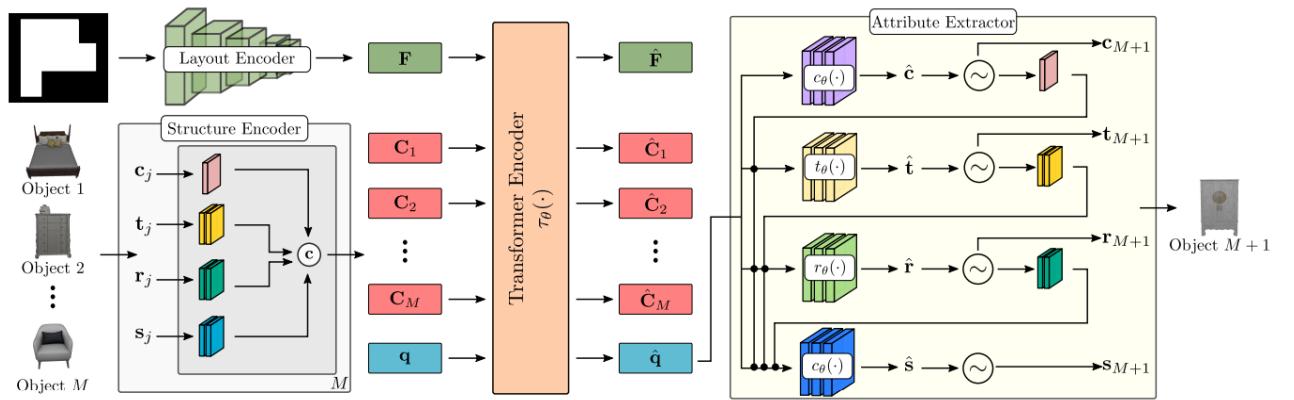


Рисунок 4 – Модель ATISS.

Использованный подход позволяет создавать достаточно разнообразные интерьеры с большим количеством различных объектов мебели. Помимо этого, представление мебели в виде неупорядоченной последовательности, позволяет использовать данную модель для решения задачи о добавлении новых объектов мебели в комнату, которая уже содержит некоторую мебель.

Однако, предложенное в [7] решение, не использует в качестве условия ничего, кроме форма пола комнаты. С одной стороны можно использовать в обучающем наборе больше комнат, ведь не накладываются требования на двери и окна. С другой стороны — такой подход не может быть применим в задачах, когда пользователю важно, чтобы при генерации расстановки мебели учитывалось расположение дверей и окон. Также в данном подходе отсутствует возможность генерировать только заданные типы мебели.



Рисунок 5 – Интерьеры, созданные ATISS. Справа представлен процесс дополнения комнаты, в которой уже присутствовала некоторая мебель.

Выводы по главе 1

В данной главе приведена формальная постановка ключевой задачи настоящей работы. Описаны современные и наиболее актуальные методы решения поставленной задачи, представлены их преимущества и недостатки, которые доказывают необходимость реализации нового метода решения, исправляющего их. Также приведен список определений и терминов, встречающихся в настоящей работе.

ГЛАВА 2. ПРЕДЛОЖЕННОЕ РЕШЕНИЕ ПОСТАВЛЕННОЙ ЗАДАЧИ

В настоящей главе представлено предложенное решение задачи об условной генерации интерьеров, а также дано обоснование, почему оно позволяет устранить недостатки уже существующих методов. Для решения поставленной задачи было выделено три подзадачи, без реализации которых невозможно получить полноценный, работающий сервис. Далее для каждой подзадачи приведено ее описание и предложенный метод решения.

2.1. Создание синтетического тренировочного набора данных

Чтобы реализовать решение задачи, поставленной в настоящей работе, необходимо уметь генерировать расстановку мебели внутри комнаты, которая задается формой своего пола. Для этого была использована соответствующая генеративная модель, описанная в разделе 2.2. Поэтому важным этапом является создание тренировочного набора данных для генеративной модели. В разделе 1.3 было описано, почему использование трехмерных координат и векторов ориентации в качестве аннотаций для объектов мебели внутри набора данных, используемого при обучении модели, является недостатком. Чтобы устраниТЬ его, предложен следующий подход: использовать лишь изображения, содержащие ортогональные проекции помещения с мебелью на плоскость его пола. Чтобы правильно реализовать трехмерную визуализацию помещения с мебелью, используемые в тренировочном наборе изображения должны соответствовать набору требований:

- Размер каждого изображения (ширина и высота) совпадает.
- Каждое изображение должно содержать фиксированный набор цветов — задать единый цветовой стиль для пола, пустого пространства вне комнаты, объектов мебели (например, каждая кровать на изображении будет фиолетового цвета, каждый стул — синего, и т.д.).
- Каждой проекции объекта мебели на плоскость должен быть сопоставленный соответствующий ограничивающий прямоугольник. Поскольку для проекции существует лишь единственный такой прямоугольник, можно уменьшить число различных геометрических форм на изображении без потери важной информации. Помимо этого, такой подход значительно упростит реализацию последующих подзадач, что описано в разделе 3.1.2.

- Допускать, что стены комнаты находятся на границе ортогональной проекции помещения на пол. Также, информацию о стенах и окнах можно добавлять путем использования соответствующего цвета на границе проекции.
- Считать, что проекция на любом изображении пропорциональна проекции на любом другом изображении. Другими словами, поскольку каждое изображение обладает фиксированным размером по вертикали и горизонтали, то чем большую площадь изображения занимает проекция помещения, тем большую площадь она занимает в реальном, трехмерном пространстве.

Таким образом, внутри одного изображения с проекцией помещения можно хранить информацию о расположении каждого объекта мебели на плоскости пола, его типе и ориентации в пространстве (можно считать, что любой объект мебели может иметь вращение лишь относительно оси, перпендикулярной плоскости пола). Несмотря на то, что процесс создания такого уникального набора изображений более гибкий и значительно проще, чем создание наборов данных, которые используются в уже существующих решениях из раздела 1.3, в рамках настоящей работы принято решение создавать синтетические изображения на основе уже существующего набора данных о трехмерных жилых помещениях. Таким образом, можно за более короткий промежуток времени создать требуемые изображения алгоритмически, без привлечения квалифицированных специалистов в области дизайна и дополнительной ручной работы.

2.2. Генерация изображений с ортогональными проекциями помещений с мебелью

В отличие от методов, описанных в разделе 1.3, предлагается первоначально генерировать не информацию о расположении мебели внутри заданной комнаты, а изображения с ортогональными проекциями помещений на плоскость пола, а затем, с помощью алгоритмов обработки изображений, получать информацию о расположении и ориентации каждого объекта мебели в трехмерном пространстве. В качестве алгоритма, который будет генерировать требуемые изображения помещений, выбраны *генеративные состязательные сети* [9]. Данная модель состоит из двух сетей: генератора и дискrimинатора. Задача генератора заключается в создании изображения, которое будет как можно более походить на изображения из тренировочного набора изображений. В

свою очередь, дискриминатор обучается отличить реальные изображения от сгенерированных, передавая результаты различения генератору. Таким образом, генератор пытается повысить процент ошибок, допускаемых дискриминатором, а дискриминатор, наоборот, пытается улучшить работу генератора — в этом заключается концепция состязательных сетей.

Поскольку в настоящей работе рассматривается задача условной генерации, то целесообразно использовать условные генеративные состязательные сети, где в качестве условия будет использоваться план комнаты и требования на необходимую мебель. Чтобы сделать генеративную состязательную сеть условной, необходимо передать дополнительные данные y как дискриминатору, так и генератору. Таким образом, оптимизационную задачу можно задать следующим образом:

$$\mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})} [\log D(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(\mathbf{z}|\mathbf{y}))].$$

Тренировочный набор для модели состоит из изображений ортогональных проекций комнат с мебелью, которые придерживаются строгим правилам. Используя этот факт, для каждого изображения можно однозначно создать его сегментацию, в которой будут представлены пол, окна и двери соответствующей комнаты, хотя в общем случае создавать сегментации для некоторого типа изображений является крайне трудной задачей. Поэтому было принято решение использовать модель, основанную на архитектуре условных генеративных состязательных сетей, решающую задачу *Image-to-Image Translation*, где условием является сегментация изображения, описанная ранее. Актуальные решения [12, 13, 16], реализующие такие модели, позволяют создавать разнообразные и реалистичные изображения по их сегментациям.

2.3. Визуализация комнат с мебелью в трехмерном пространстве

Неотъемлемой частью разрабатываемого сервиса является этап визуализации комнаты с содержащейся внутри нее мебелью. Для решения этой задачи требуется подготовить набор трехмерных моделей, которые будут использоваться при отображении конечному пользователю. Самым простым и эффективным решением является использование готовой библиотеки, предоставляющей доступ к визуализации трехмерных объектов. Помимо выбора моделей мебели, которые будут соответствовать размерам ограничивающих прямо-

угольников на изображении с проекцией комнаты и стилю, заданным пользователем, необходимо восстановить стены, окна, двери и пол комнаты в формате, пригодном для отображения произвольных трехмерных объектов выбранной библиотекой. Чтобы определять положение этих объектов в трехмерном пространстве был разработан ряд алгоритмов, определяющих их координаты и ориентации на изображении с проекцией комнаты, и, впоследствии преобразующих их к необходимому масштабу. Тем не менее, при проекции трехмерного помещения на плоскость часть информации будет утеряна: ориентацию объекта мебели можно восстановить лишь с точностью до 180° для ограничивающего прямоугольника, 90° — для квадрата. Также из-за проекции невозможно восстановить высоту объекта мебели. Для исправления возникших трудностей был разработан ряд алгоритмов, о реализации которых подробно рассказано в разделе 3.3.1.

Выводы по главе 2

1. Дано описание процесса создания тренировочного набора, содержащий изображения ортогональных проекций комнат на плоскость их пола.
2. Предложен алгоритм для условный генерации расстановки мебели внутри комнаты.
3. Предложен алгоритм для визуализации комнаты с расставленной в ней мебелью.

ГЛАВА 3. ДЕТАЛИ РЕАЛИЗАЦИИ ПРЕДЛОЖЕННОГО РЕШЕНИЯ

В настоящей главе подробно описывается детали реализации предложенного решения. Объединение решений, полученных на каждом этапе, образуют единый сервис, который был описан во введении. При этом, в случае, если какая-то подзадача имела несколько возможных решений, приведены все варианты реализаций данных решений. Каждой реализации сопутствует ее характеристика: почему она была или не была использована в финальном разработанном сервисе, какими преимуществами и недостатками она обладает. Предложенное решение было реализовано языке Python версии 3.9.7.

3.1. Этапы создания синтетического набора изображений

Поскольку датасет SUNCG [14] больше не находится в открытом доступе, за основу был взят наиболее крупный и новый датасет, предоставляющий открытый доступ — 3D-FRONT [4]. Датасет включает в себя 6813 файлов с расширением `.json`, которые суммарно содержат информацию о 18797 комнатах, основанных на чертежах существующих в реальности жилых домов Китая. Каждое помещение сопровождается информацией об объектах мебели, мэшах стен и пола, а также текстур, принадлежащих данному помещению. Помимо самого датасета, его создатели предоставляют набор утилит по взаимодействию с ним, в том числе легковесную библиотеку для его визуализации — Trescope. Пример трехмерной визуализации дома из датасета 3D-FRONT [4], выполненной с помощью Trescope можно увидеть на рисунке 6. Помимо вышеописанного датасета также использовался набор трехмерных, фотoreалистичных моделей мебели 3D-FUTURE [5]. Модель мебели задается тремя файлами: трехмерная геометрия и правила наложения на нее текстуры задается `.obj` и `.mtl`-файлом соответственно, когда используемая текстура хранится в смежном `.png`-файле. Стоит отметить, что оба датасета связаны друг с другом при помощи уникальных идентификаторов объектов мебели.

Каждый `*.json`-файл из датасета содержит информацию о конкретном доме, при этом имеет структуру, представленную на листинге 1. Каждый дом и каждая комната внутри дома обладает глобальным уникальным идентификатором `uid`. Список всех мэшей, принадлежащих дому, находится по ключу `mesh`, список мебели — `furniture`, список комнат — `room`. Каждый объект из любого из списков также обладает уникальным локальным идентификатором в пределах дома, к которому он принадлежит. У объекта мебели из



Рисунок 6 – Результат визуализации квартиры с помощью Trescope.

furniture указано, на какую модель из датасета 3D-FUTURE [5] он ссылается.

Листинг 1 – Структура содержания .json-файла из датасета 3D-FRONT.

```
{
  "uid": "6781a219-c0f0-42b0-8c40-4ef700a5dacb",
  "design_version": "0.1",
  "code_version": "0.4",
  "north_vector": [...],
  "furniture": [ {...} , ... ],
  "mesh": [ {...} , ... ],
  "material": [ {...} , ... ],
  "extension": [ {...} , ... ],
  "scene": [
    "room": [ {...} , ... ],
    ...
  ],
  "version": "2.0"
}
```

3.1.1. Получение изображений с помощью библиотеки Trescope

Поскольку требуется получать изображения проекций не всего дома, а каждого отдельного помещения — то, .json-файл необходимо предваритель-

но отфильтровать, убрав из него все объекты, не принадлежащие текущей комнате. Затем, необходимо убрать все мэши стен, карнизов, потолков и т.д., оставив лишь информацию про пол, двери и окна. Файл с полученным содержанием уже можно передавать в Trescope для его дальнейшего *рендеринга*. Однако, чтобы полученные изображения могли соответствовать требованиям, установленным в разделе 2.1, для каждой модели мебели из 3D-FUTURE [5] .png-файл с текстурой был изменен таким образом, чтобы для всей мебели, принадлежащей к одной и той же категории, текстуры имели один и тот же цвет. Чтобы при рендеринге получить ортогональную проекцию комнаты на плоскость пола, требовалось переместить внутреннюю камеру, которая используется в библиотеке для визуализации, следующим образом:

1. Высчитать координаты геометрического центра комнаты, переместить камеру в него и поднять на нужную высоту, одинаковую для каждой комнаты из набора.
2. В качестве точки, на которую будет ориентирована камера, выбрать точку с координатами лежащими на плоскости пола строго под местом нахождения камеры.

Таким образом, при фиксированном угле обзора можно будет добиться пропорциональности изображений. Пример полученного изображения можно видеть на рисунке 7.

Несмотря на кажущуюся простоту подхода, у него есть несколько существенных недостатков:

1. Из-за того, что изображение получается при рендеринге трехмерных объектов, то появляется эффект перспективы: некоторые объекты мебели будут выходить за границы пола комнаты, что в свою очередь будет мешать как и обучению модели, генерирующей картинки на следующем этапе, так и восстановлению информации о расположении объектов по изображению.
2. Поскольку объекты мебели в трехмерном пространстве находятся в своем исходном состоянии, то, например, может возникать ситуация, представленная на рисунке 7 — проекции стульев (синий цвет) не обязательно будут являться односвязными множествами. Это, опять же, не приносит дополнительной полезной информации и будет затруднять по-

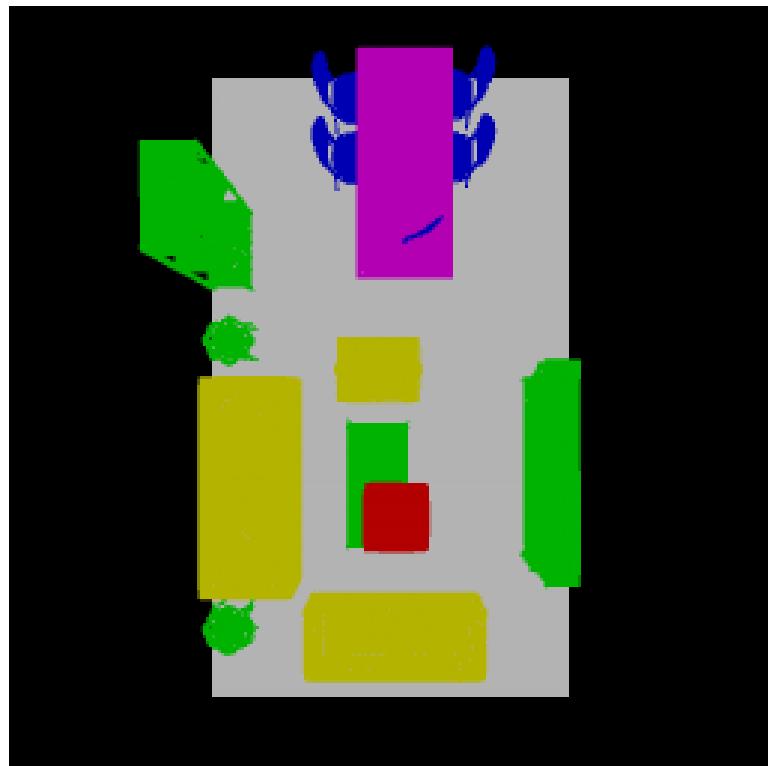


Рисунок 7 – Пример ортогональной проекции комнаты, полученной с помощью библиотеки Trescope.

следующий процесс извлечения данных о положении объекта мебели в пространстве.

Анализируя недостатки данного подхода, были получены требования к изображениям, сформулированные в разделе 2.1. Подробнее про формат изображений, требования к ним а также про фильтрацию подходящих комнат будет рассказано далее.

3.1.2. Реализованный метод получения изображений

Чтобы удовлетворить ранее поставленные требования в изображениям в обучающем наборе был разработан альтернативный метод получения изображений с проекциями помещений. Для этого были использованы доработанные методы, предоставленные утилитой 3D-FRONT-Toolbox. Был разработан алгоритм, который для каждого исходного .json-файла создает новый файл, в котором собрана обработанная информация по каждой комнате внутри данного дома. Это информация включает в себя:

- Уникальные идентификаторы комнаты и соответствующего дома, а также название комнаты.

- Список мебели, присутствующей в комнате. Для каждого элемента мебели из этого списка известен его уникальный идентификатор в 3D-FUTURE [5], координаты вершин ограничивающего прямоугольника на плоскости пола комнаты, категория и стиль.
- Координаты вершин многоугольника, задающего пол комнаты.
- Координаты окон и дверей, принадлежащих комнате.

Каждый объект используемой мебели может принадлежать к одной из шести категорий: Cabinet/Shelf/Desktop, Bed, Chair, Table, Sofa или Lighting (шкаф/тумбочка, кровать, стул, стол диван или освещение). Затем, для каждого дома происходит фильтрация его комнат:

- Удаляются комнаты, чьи размеры слишком маленькие или слишком большие.
- Если в комнате нет дверей, она считается неправильной.
- Если число объектов мебели внутри комнаты меньше двух, комната не рассматривается.
- Если тип комнаты не соответствует требуемому, комната не рассматривается.

Подходящими типами комнат являются LivingDiningRoom, MasterBedroom, SecondBedroom, Bedroom, LivingRoom и DiningRoom (другими словами, в настоящей работе рассматриваются только комнаты, схожие со спальнями и гостинными).

Следующим этапом является создание изображений на основе имеющихся данных о расположении и типах объектов расстановки мебели. Каждое изображение имеет размер 256^2 . Все координаты вершин геометрических объектов даны на координатной плоскости, где единичный отрезок равен 1 метру, а также могут быть отрицательными. Для каждой точки, задающей вершину объекта из расстановки мебели (или же для вершины многоугольника, задающего пол комнаты) применяется операция переноса со сдвигом таким образом, чтобы на итоговом изображении ортогональная проекция находилась в его центре и была верным образом масштабирована (длина наибольшей стороны комнаты должна составлять

$$\frac{\max (room_w, room_h)}{room_{max}}$$

от размера стороны изображения — 256 пикселей, где $\max(\text{room}_w, \text{room}_h)$ — длина наибольшей стороны комнаты в метрах, room_{\max} — максимально допустимая длина стороны комнаты, которую можно представить на изображении).

Несмотря на то, что при ортогональной проекции на плоскость пола теряется вся информация про объекты по оси, перпендикулярной полу, ее можно частично восстановить. Для этого важно задать порядок добавления ограничивающих прямоугольников на изображение. Для каждого используемого типа мебели введем соответствующий цвет в *RGB*-формате:

Cabinet/Shelf/Desk	: [0, 255, 0], лайм
Bed	: [128, 0, 128], пурпурный
Chair	: [0, 0, 255], синий
Table	: [255, 0, 255], маджента
Sofa	: [255, 255, 0], желтый
Lighting	: [255, 0, 0], красный

Затем, следуя заданному порядку на изображение добавляются все требуемые объекты — сначала пол, двери и окна. Затем, следуя порядку: Bed, Table, Sofa, Cabinet/Shelf/Desktop, Chair, Lighting. Таким образом, при наложениях и других спорных ситуациях (стул нарисован поверх стола, т.к. был задвинут, освещение находится над кроватью и т.д.) можно, следуя порядку, восстановить исходную информацию. Поскольку на результирующем изображении все координаты являются целыми, неотрицательными числами, могли возникать случаи, когда некоторые объекты, такие как стулья, из-за небольшого расстояния между друг другом, соприкасались сторонами ограничивающих прямоугольников. В таком случае, невозможно было бы восстановить их исходные размеры и определить количество объектов на изображении. Для избежания таких ситуаций, произошла пост-обработка, позволяющая нивелировать проблему.

Для каждого изображения также создавалось дополнительное изображение, содержащее сегментацию ортогональной проекции. Этот этап необходим для корректного обучения модели, описанной далее. Каждое изображение с сегментацией имеет размер 256^2 , но, в отличие от предыдущих изображений, находится в *Grayscale*-формате — на каждый пиксель приходится один канал. Для каждого пикселя, отождествляющего пол, приходится значение 1, дверь — 2, окно — 3, пустое пространство вне комнаты — 0. Таким образом, каждое

изображение представляется в виде матрицы $256 \times 256 \times 3$, каждое изображение с сегментацией — 256×256 . Из общего числа комнат, представленных в датасете 3D-FRONT [4], после ручного удаления нереалистичных комнат и алгоритмической фильтрации, получился новый набор, содержащий 5072 изображений. Пример пары изображений, полученных описанным методом, можно видеть на рисунке 8.

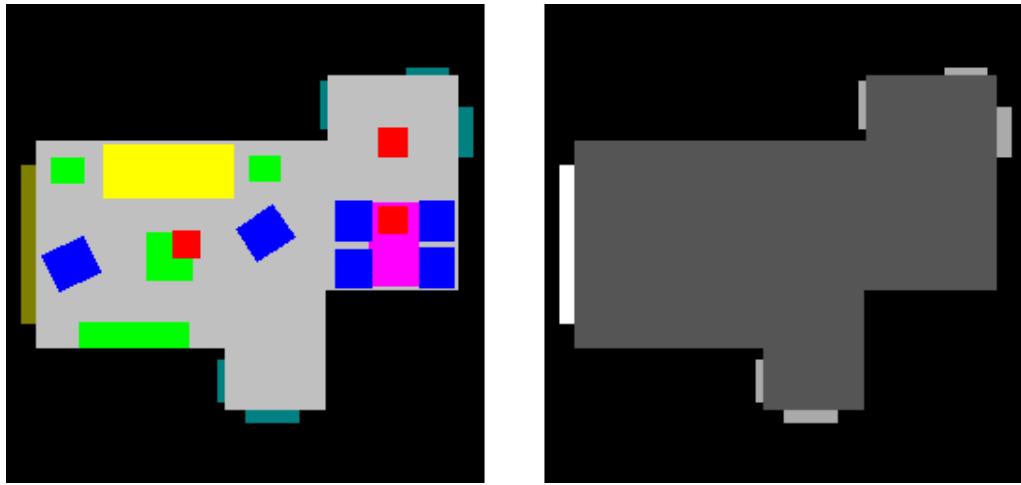


Рисунок 8 – Пример изображения, содержащего проекцию комнаты и соответствующую сегментацию. Изображение с сегментацией переведено в трехканальный формат и нормировано для наглядности.

3.2. Генерация изображений на основе уже существующих

Первоначально задумывалось использовать обход латентного пространства генеративной состязательной сети, чтобы добиться условной генерации изображений с расстановкой мебели. Однако, при таком подходе возникают трудности, ведь необходимо дополнительно реализовывать дополнительный энкодер, занимающийся кодировкой информации про стены, двери и окна комнаты. Поскольку изображение с ортогональной проекцией комнаты уже в какой-то степени является сегментацией, то было принято решение поменять подход и использовать генеративные модели, решающие задачу *Image-to-Image Translation*. Обычно трудность решения такого рода задачи состоит в том, что создавать для изображений их сегментации является трудной задачей. Однако, в подходе, изложенном в настоящей работе, получить сегментацию для изображения не составляет никакого труда — ведь, грубо говоря, сегментацией является проекция комнаты без мебели. Далее будут приведены подходы, которые применялись для решения задачи о генерации изображений.

3.2.1. Модель OASIS

Модель OASIS [16], в отличие от других методов, решающих задачу о синтезе фотореалистичных изображений из сегментационных масок, нуждается лишь в состязательном надзоре для обучения генератора. Используя в качестве основы модель SPADE [13], авторы статьи предлагают свой вариант дискриминатора, сделанного в виде семантической сегментационной сети, которая использует изображения с сегментацией в качестве истинных значений. Кроме того, обновленный генератор, на основе результатов от дискриминатора (уточненных как семантически, так и пространственно), позволяет эффективно создавать мультимодальные изображения с помощью изменения трехмерного шума. Архитектура модели OASIS представлена на рисунке 9.

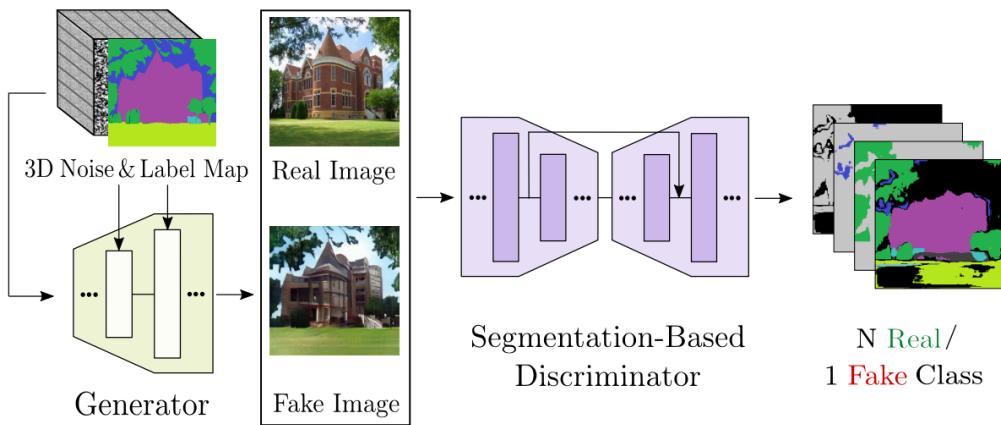


Рисунок 9 – Архитектура модели OASIS.

Чтобы использовать OASIS для решения поставленной задачи, необходимо видоизменить трехмерный шум. В исходном варианте модели, на этапе обучения генератор принимает конкатенацию изображения с сегментацией и трехмерного шума с размерностями $h \times w \times z$, где z — количество слоев шума, h, w — ширина и высота изображения соответственно. Поскольку требуется использовать в качестве условия для генерируемого изображения не только форму пола, но и требуемую мебель, было необходимо видоизменить шум. Для этого, был реализован алгоритм, который на каждом этапе обучения создавал для каждого тренировочного изображения уникальный шум. Чтобы добиться такого функционала, каждому изображению из тренировочного набора был присвоен массив длины 6. Каждый элемент массива отвечал за присутствие или отсутствие на изображении определенного типа мебели (например,

массив $[0, 0, 0, 4, 0, 6]$ означает присутствие на изображении кровати и шкафа). Каждый такой массив предварительно был нормализован, и, затем, произо-дилась конкатенация к шуму в каждой точке $h \times w$.

Обучение производилось с гиперпараметрами, предложенными в статье, за исключением тех, которые требовалось изменить для соответствия постав-ленной в настоящей работе задаче. Чтобы избежать ситуаций, когда модель при минимизации функции ошибки попадает в локальный минимум, регуляр-но делались перезапуски из мест, где сохранялись текущие веса модели. Но, спустя достаточный промежуток времени обучения, получить удовлетвори-тельные результаты генерируемых картинок так и не получилось. Поскольку даже повторное обучение с измененными гиперпараметрами не дало желае-мых улучшений, был сделан вывод что такая конкатенация дополнительной информации к шуму не является правильной (изначально шум получался из стандартного нормального распределения) и принято решение использовать другой подход для генерации изображений.

3.2.2. Модель SEAN

Авторы статьи [12] предлагают новый метод, позволяющий эффектив-но переносить, кодировать и синтезировать стиль изображения в задачах ге-нерации изображений по сегментации. В качестве основы модели была взя-та модель SPADE, в которую добавили нормализационный блок SEAN, кото-рый может получить информацию о стиле из различных регионов сегментации изображения и, затем, преобразовывать ее к форме пространственных норма-лизационных параметров. На рисунке 10 изображена схема работы генератора данной модели вместе с энкодером стиля изображения.

В отличие от подхода, описанного в разделе 3.2.1, модель на основе SEAN не требует дополнительного внедрения информации о требуемых ти-пах мебели, которые должны присутствовать на изображении — при данном подходе в качестве этой условной информации выступает стиль изображения. Таким образом, чтобы обученная модель могла синтезировать новые изобра-жения, в качестве входных данных ей требуется передать сегментацию (услов-вие на форму комнаты) и нужный стиль (условие на требуемую мебель).

Обучение производилось согласно рекомендованным параметрам в ста-тье. В течение первых 50 эпох обучения на полученном ранее тренировочном наборе темп обучения, используемый в алгоритме оптимизации, оставался по-

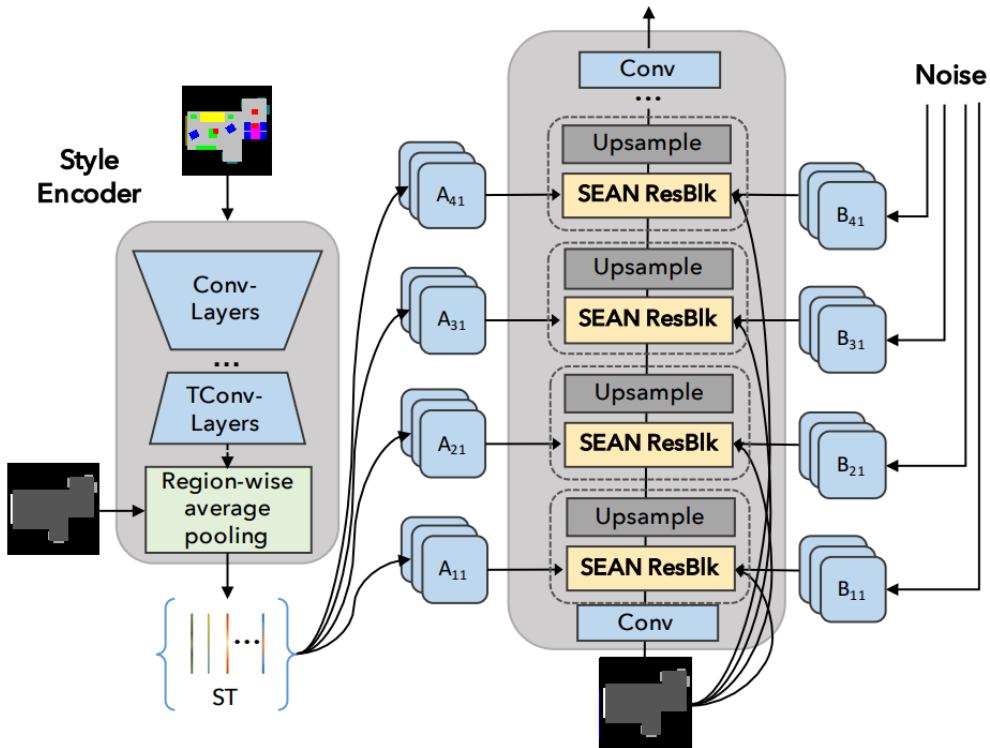


Рисунок 10 – Архитектура генератора модели SEAN.

стоянным. Следующие 50 эпох обучения этот параметр постепенно уменьшался. Так же, как и при обучении модели в разделе 3.2.1, производились перезапуски из контрольных точек. После завершения обучения, для каждого изображения из тренировочного набора был найден его стиль. Впоследствии, чтобы создать изображение проекции комнаты с нужным набором мебели, выбирается случайный стиль, который соответствует указанным требованиям. Суммарно обучение длилось приблизительно 96 часов на одной видеокарте NVIDIA Tesla K80, имеющей 12 гигабайт памяти.

3.3. Трехмерная визуализация

Задача о трехмерной визуализации заключается в том, чтобы по изображению с ортогональной проекцией комнаты построить ее трехмерную визуализацию. Важно, чтобы была возможность отображать стены комнаты, сохранять конфигурацию мебели, использованную при визуализации, выбирать желаемые цвета для стен и файл, содержащий текстуру для пола. Подразумевается, что конечный пользователь также будет обладать возможность свободно перемещать камеру, чтобы рассматривать восстановленную квартиру с разных сторон. Данная задачу следует разбить на несколько подзадач: распо-

знавание объектов мебели на изображении (тип и координаты на плоскости пола), восстановление стен вместе с дверьми и окнами, расстановка требуемых трехмерных моделей и их ориентирование. Для реализации этих подзадач использовалась библиотека simple-3dviz [6]. Подробнее про каждый этап будет рассказано далее.

3.3.1. Распознавание объектов на изображении с проекцией комнаты

Для решения этой подзадачи использовались методы обработки изображений, предоставляемые библиотекой OpenCV [8]. Области с требуемыми цветами получаются с помощью метода `inRange`, которой в качестве параметров передается массив с изображением и границы требуемого цвета. Затем метод `findContours` возвращает контуры объектов в изображении, полученном предыдущим методом. Для `findContours` в качестве режима аппроксимации используется `CHAIN_APPROX_SIMPLE`. Помимо обнаруженных контуров, которые задаются координатами вершинам многоугольника, данный метод также возвращает иерархию полученных контуров. Это позволяет определить и убрать ненужные контуры, например — вложенные. Координаты полученных многоугольников необходимо преобразовать, ведь они являются целыми неотрицательными числами, которые принадлежат промежутку $[0, 256]$. Для этого достаточно каждую координату умножить на $length_{max} / 256$.

Чтобы восстановить многоугольники, задающие пол, окна и двери, необходимо применить метод `findContours` к изображению с сегментацией. Цвета этих объектов известны, для сегментации они постоянны. Предполагается, что проекции стен являются отрезками и совпадают со сторонами многоугольника, задающего пол комнаты. Далее, чтобы найти отрезки, задающие окна и двери, необходимо найти пересечения соответствующих многоугольников с многоугольником пола. Как и в случае стен, отрезки для окон и дверей лежат на сторонах многоугольника пола. Результат работы этого этапа представлен на рисунке 11.

На следующем этапе происходит распознание мебели. В отличие от изображений в синтетическом тренировочном наборе, изображения, созданные генеративной состязательной сетью, не будут обладать ожидаемым набором цветов (например, фиолетовый цвет для ограничивающего прямоугольника кровати может иметь несколько оттенков). Чтобы находить контуры для

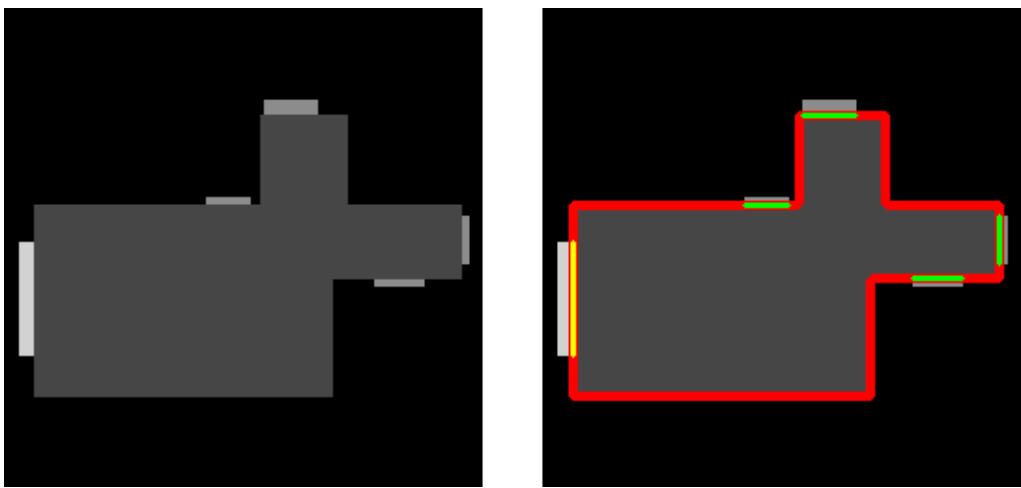


Рисунок 11 – Обнаружение формы пола (красный), дверей (зеленый) и окон (желтый) на изображении с сегментацией.

таких объектов необходимо перевести картинку из RGB формата в HSV формат с помощью метода `cvtColor`. Таким образом проще задавать диапазон для каждого цвета мебели. Границы цвета задаются следующим образом (`hue` и `value` берутся из HSV-цвета требуемой мебели):

```
нижняя : [ (hue - 10), 100, value - 25]
верхняя : [ (hue + 10), 255, value + 25]
```

Для каждого контура с помощью метода `minAreaRect` находится минимальный прямоугольник, описывающий данный контур. Вышеописанный метод не работает в случаях, когда пересечение двух прямоугольников разных объектов мебели не пустое. Чтобы проверить это, необходимо посчитать отношение площади контура к площади минимального описывающего прямоугольника. Если соотношение значительно меньше единицы — произошло пересечение. В зависимости от типа объектов, задействованных в пересечении, применяется определенный подход для разрешения конфликта. Чтобы определить требуемый подход, необходимо понять, если ли внутри контура цвета, относящиеся к полу. Для этого, с помощью методов `bitwise_and`, `bitwise_or` создаются маски изображения в нужных областях, объединение которых позволяет получить часть изображения внутри конфликтного контура.

Если внутри этой части изображения достаточное содержание пикселей с цветом пола или пустого пространства вне комнаты, то произошло пересечение двух прямоугольников одного и того же цвета. В таком случае, для изображения несколько раз применяется морфологическая операция эрозии, `erode`.

На каждой итерации ядро структурирующего элемента данного метода увеличивается. Как только контур будет разъединен на несколько контуров, площадь которых совпадает с площадью описывающих их прямоугольников, процесс эрозии прекращается. В зависимости от того, как сильно было увеличено ядро, будет увеличен описывающих прямоугольник. Пример работы данного алгоритма представлен на рисунке 12.

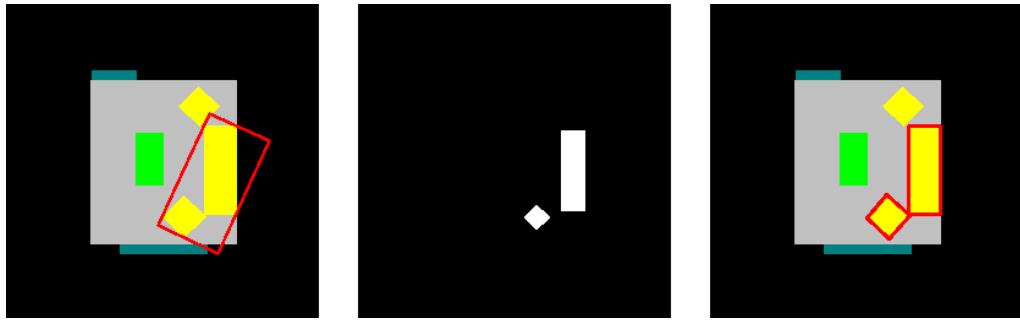


Рисунок 12 – Случай с пересекающимися прямоугольниками одного цвета.

Как видно на первом изображении, контур был обнаружен для двух прямоугольников, внутри него есть пол. На втором изображении результат применения метода `erode`. На последнем изображении — результат после разрешения конфликта.

В случае, когда на части изображения отсутствует пиксели, принадлежащие полу, произошло пересечение с другим типом мебели. В таком случае, описывающий контур прямоугольник является искомым.

Отдельно рассматривалось восстановление координат описывающего прямоугольника для столов. Поскольку на картинках зачастую стулья были задвинуты под столы, то, на изображениях с проекциями их ограничивающие прямоугольники сильно закрывали прямоугольники столов, потенциально разбивая его на множество более маленьких прямоугольников. Для восстановления исходного прямоугольника применяется морфологическая операция закрытия (расширение и последующая эрозия) — `morphologyEx`, где в качестве параметра операции используется `MORPH_CLOSE`. В качестве структурирующего элемента была выбран круг с радиусом в 9 пикселей. С помощью данного подхода, если стол был разбит на несколько близлежащих контуров, то они будут соединены в один. Пример работы данного алгоритма представлен на рисунке 13.

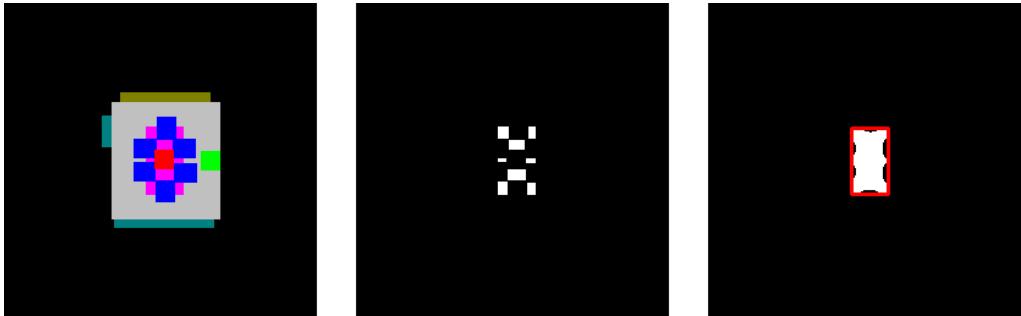


Рисунок 13 – Восстановление описывающего прямоугольника для стола. Как видно на изображении, на стол несколько раз был наложен стул, из-за чего он потерял связность. Благодаря описанному методу, на правом изображении видно как был восстановлен верный описывающий прямоугольник.

3.3.2. Рендеринг

После того, как для каждого объекта были найдены его описывающие прямоугольники, можно приступать к этапу трехмерной визуализации. Для каждого прямоугольника известны его координаты центра и угол поворота, который рассчитывается между наибольшей стороной прямоугольника и горизонтальной осью координат. Далее, необходимо подобрать для каждого такого прямоугольника требуемую мебель: подходящую по размерам и соответствующую заданному пользователем стилю. Предварительно, для всего набора моделей 3D-Future [5] были вычислены их размеры по трем координатным осям. Подбор мебели осуществляется методом k -ближайших соседей — поскольку восстановленные размеры мебели на изображении имеют некоторую погрешность и не обязательно должны соответствовать конкретной модели, для них устанавливаются пороги, в пределах которых будет вестись поиск. В случае, если по результатам поиска не было найдено подходящей модели, поиск будет повторен, но без учета стиля. Если же и в таком случае ничего не было найдено, то будет выбрана случайная модель с соответствующим типом, чью размеры отличаются от требуемых на как можно меньшую величину.

Поскольку некоторые объекты мебели в комнате могут повторяться (тумбочки по обе стороны от кровати, стулья вокруг обеденного стола и т.д.), необходимо реализовать функционал, определяющий, имеют ли объекты мебели в восстанавливаемом изображении одну и ту же модель. Для этого каждый описывающий прямоугольник представляется как пара из минимальной и максимальной стороны $\langle side_{min}, side_{max} \rangle$. Для этих пар можно применить алгоритм

кластеризации DBSCAN, который сгруппирует близко расположенные пары. Далее для каждого полученного кластера будет использоваться одна модель. Пример результата работы данного алгоритма представлен на рисунке 14.

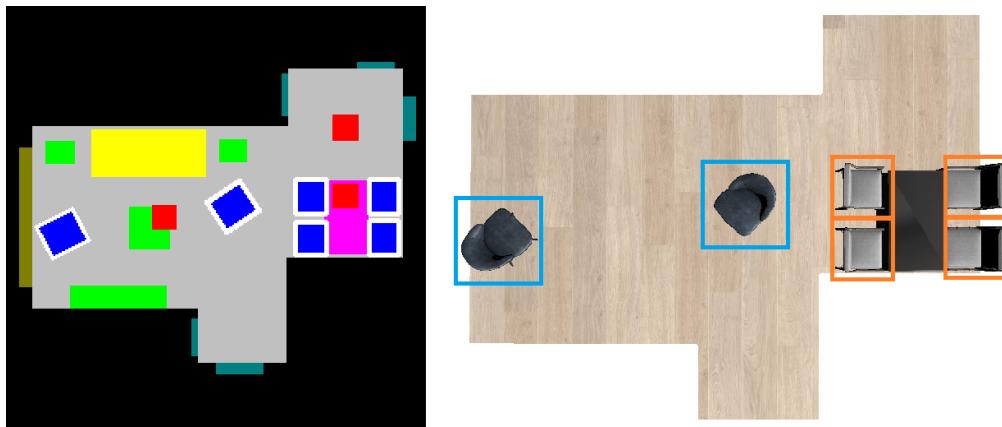


Рисунок 14 – Пример кластеризации мебели. На изображении слева видно, что имеется 4 стула одного и 2 стула другого размера. В результате работы алгоритма им присваиваются различные модели.

Восстановленные трехмерные модели мебели дополнительно требуется ориентировать, т.к. зачастую угол между горизонтальной осью координат и стороной описывающего многоугольника недостаточно. В таком случае применялся следующий ряд правил:

- Если объект мебели находится возле стены, то своей лицевой стороной он должен быть развернут от нее.
- Если объект мебели находится в углу комнаты, то его ориентация производится в зависимости от того, к какой стороне угла принадлежит наибольшая сторона описывающего объект мебели прямоугольника.
- В случае, когда несколько объектов мебели принадлежат одному кластеру, то их надо ориентировать лицевой стороной к опорному объекту. Так, для нескольких стульев, расположенных в пределах допустимого расстояния от одного стола, данный стол и будет являться опорным объектом. Отдельно стоит описать процесс визуализации пола, стен, окон и дверей.

В отличие от моделей мебели, для которых файлы, содержащие описание трехмерной геометрии и правила наложения текстур, существовали заранее, задавать произвольные объекты требовалось с нуля. Для того, чтобы однозначно описать произвольный многоугольник в трехмерном пространстве, необходимо произвести его триангуляцию и задать набор вершин `vertices`, граней

`faces`, нормалей к граням `normals` и развертку `uv` — набор значений (значения лежат на отрезке $[0, 1]$), которые задают соответствие между вершинами граней и координатами на добавляемой объекту текстуре. Если для многоугольника требуется текстура, то дополнительно задается ее характеристики `mtl`, цвет — RGB-значение цвета. Поскольку пол является многоугольником без отверстий, а стены, содержащие окна или двери можно разбить на более маленькие сегменты стен (тем самым избавиться от отверстий), то для триангуляции в данном случае применим алгоритм отрезания ушей. Нормали для многоугольников были направлены вовнутрь комнаты, если многоугольник задавал сегмент стены, и вверх, если многоугольник задавал пол. Благодаря этому освещение данных объектов будет естественным.

С помощью методов `show` и `render` производится отображение восстановленной комнаты в трехмерном пространстве либо в отдельном окне, либо путем сохранения ряда изображений в `.gif`-файл. Пользователь сервиса может указать в качестве параметра сохранение конфигурации — в таком случае будет создаваться `.json`-файл, содержащий в себе необходимою информацию для восстановления комнаты: ориентации объектов мебели, используемые модели.

Выводы по главе 3

1. Приведена реализация метода создания синтетического набора изображений, задающих комнаты с мебелью.
2. Рассмотрены различные варианты генеративных моделей, создающие требуемые синтетические изображения.
3. Приведена реализация алгоритма для визуализации комнаты с расставленной в ней мебелью.

ГЛАВА 4. ЭКСПЕРИМЕНТАЛЬНОЕ СРАВНЕНИЕ

В настоящей главе приведены результаты работы предложенного решения. Поскольку синтетический набор изображений с ортогональными проекциями комнат был создан на основе 3D-FRONT [4], а большая часть существующих методов, решающих задачу об условной генерации интерьера, для обучения использовали [14], то производить сравнение между ними и предложенным решением было бы некорректно. Чтобы избежать этого, потребовалось заново обучить модели на датасете 3D-FRONT [4], что является крайне трудоемкой задачей, как в плане реализации, так и в плане затрачиваемого времени. Поэтому в сравнении были использованы результаты, полученные в [7] — авторы статьи обучили как и свою, так и другие модели на датасете, используемом в настоящем решении.

4.1. Генерация изображений

Реализованное решение позволяет создавать правдоподобные изображения, содержащие расстановку мебели внутри комнаты. При этом, выполняются поставленные требования, а именно форма пола и необходимая мебель. На рисунке 15 приведен пример созданных изображений по требуемой сегментации (левый верхний угол). В качестве требуемой мебели была выбрана кровать, шкаф и освещение. Как можно видеть, расстановка мебели значительно отличается от изображения к изображению. При этом, не нарушается логика расстановки мебели: как и в случае с изображениями, используемыми в тренировочном наборе, никакой объект мебели не закрывает дверь, ведущую в комнату, ничего не перекрывает окно сверху. Несмотря на это, также существуют некоторые недостатки: количество объектов в изображении с комнатой не превышает 10, для некоторых экстремальных случаев (попытка создать изображение с мебелью, в котором на сегментации пол представлен не в виде выпуклого многоугольника, например - в виде буквы Т). Данные проблемы возникают из-за относительно небольших размеров обучающего набора, что было вызвано наличием большого числа неправильно заданных комнат в [4].

4.2. Сравнение с существующими решениями

Чтобы показать, что предложенное решение позволяет условно генерировать качественные интерьеры с мебелью, для трех комнат, использовавшихся в сравнении [7] были восстановлены их трехмерные визуализации, которые изображены на рисунке 16. Если не брать в расчет различный рендеринг

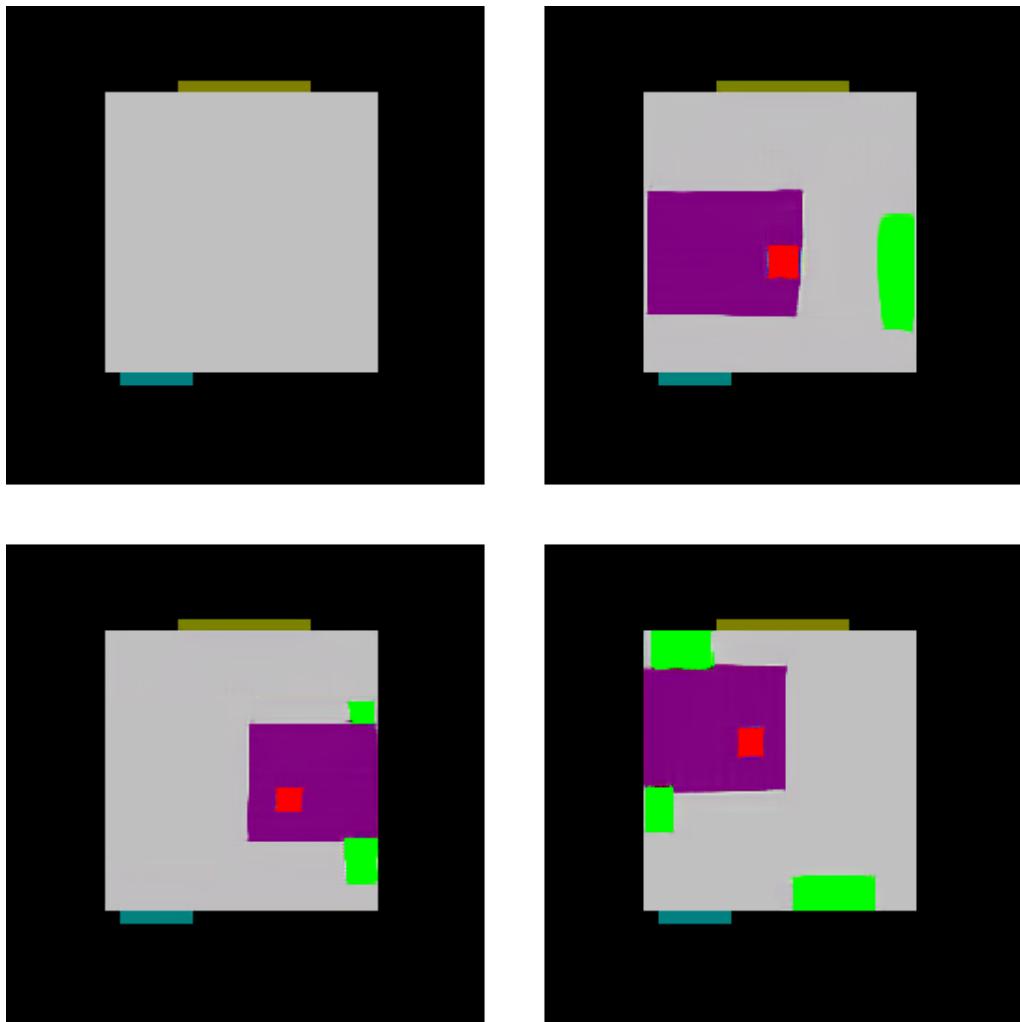


Рисунок 15 – Пример разнообразных расстановок мебели, генерация которых происходила по заданному условию.

(для визуализации существующих решений использовался сервис NVIDIA Omniverse), то можно судить о приемлемом качестве визуализации. Несмотря на то, что подход, реализованный в [7], создает более качественные интерьеры, предложенный в настоящей работе сервис не отстает по качеству от остальных решений, а в некоторых ситуациях не допускает неправильных расстановок мебели (например, мебель не находится вне комнаты). Чтобы привести количественный анализ, также был подсчитан ряд метрик (на таком же наборе данных, что и в статье [7]). Для измерения реалистичности созданных интерьеров, подсчитывается *расхождение Кульбака-Лейблера* между распределениями типов мебели в исходных комнатах и в комнатах, которые были созданы на основе изображений, полученных генеративной моделью. Помимо этого, было вычислено расстояние *FID* между ортогональными проекциями визуали-

заций комнат (визуализации получены для изображений 256² из тестового набора и для сгенерированных по соответствующим сегментациям). Результаты приведены в таблицах 1 и 2. Сравнение проводилось на комнатах, являющихся спальнями, гостиными и кухнями (под кухней понимается комната, в которой расположен стол и несколько стульев). Несмотря на то, что предложенное решение не является лучшим в рамках данных метрик, оно показало достойные результаты и по качеству находится на одном уровне с некоторыми актуальными решениями. Из-за относительно небольшого количества комнат, отличных от спален, в тренировочном наборе, получать разнообразные расстановки похожих типов мебели затруднительно.

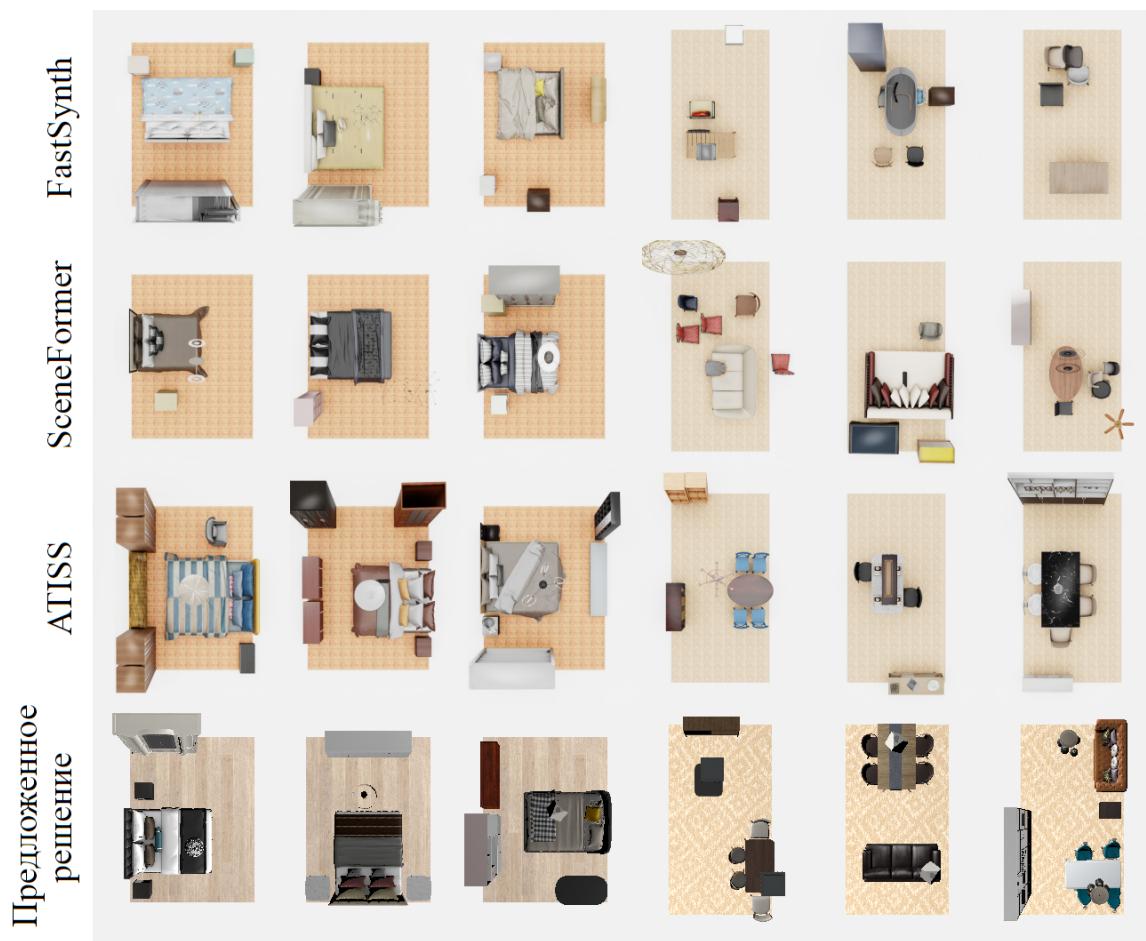


Рисунок 16 – Проекции трехмерных визуализаций интерьеров, полученные разными методами. В каждом ряду для двух форм пола представлено три различные расстановки мебели. В предложенном решении подобраны схожие текстуры пола. Для рендеринга первых трех рядов был задействован сервис NVIDIA-Omniverse.

Таблица 1 – Метрика FID для интерьеров, полученных различными методами. Значения, приведенные в таблице, являются средним результатом, полученным после 10 подсчетов данной метрики.

FID (\downarrow)				
Тип комнаты	FastSynth	SceneFormer	Atiss	Предложенное решение
Спальни	40,89	43,17	38,39	42,55
Гостиные	61,67	69,54	33,14	61,23
Кухни	55,83	67,04	29,23	53,82

Таблица 2 – Расхождения Кульбака-Лейблера между распределениями типов мебели.

KL Divergence (\downarrow)				
Тип комнаты	FastSynth	SceneFormer	Atiss	Предложенное решение
Спальни	0,0064	0,0052	0,0085	0,0081
Гостиные	0,0176	0,0313	0,0034	0,0283
Кухни	0,0518	0,0368	0,0061	0,0437

Выводы по главе 4

1. Приведен количественный и качественный анализ предложенного решения.
2. Приведено сравнение с существующими альтернативными решениями.
3. По результатам сравнения можно судить о том, что предложенное решение позволяет создавать реалистичные интерьеры, не уступающие в качестве интерьерам, созданным уже существующими решениями.

ЗАКЛЮЧЕНИЕ

В настоящей работе был разработан сервис, позволяющий по заданным условиям генерировать интерьер комнаты методами машинного обучения. В ходе работы были решены следующие задачи:

1. Приведен алгоритм, создающий синтетический набор изображений ортогональных проекций комнат с мебелью. Используя его в качестве основы, можно получать аналогичные изображения, пользуясь другими датасетами с трехмерными аннотациями помещений.
2. На основе генеративной состязательной сети создана модель, позволяющая генерировать изображения с ортогональными проекциями комнат с мебелью.
3. Разработан алгоритм для визуализации комнат на основе соответствующих изображений с ортогональными проекциями. Данный алгоритм получился автономным: для него не требуется использовать изображения, полученные генеративной моделью. С помощью любого графического редактора можно создать изображение комнаты, которое также возможно визуализировать.

По результатам экспериментального сравнения сделан вывод о применимости предложенного метода для создания реалистичных интерьеров. Предложенное решение также исправляет некоторые недостатки в уже существующих решениях.

Таким образом, цель работы достигнута, а задачи выполнены.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 *Freepik*. Bedroom interior top view [Электронный ресурс]. — 2022. — URL: https://www.freepik.com/free-vector/bedroom-interior-top-view-modern-3d-home-room_20626055.htm (дата обр. 20.02.2022).
- 2 *Википедия*. Метод k-ближайших соседей [Электронный ресурс]. — 2022. — URL: <https://ru.wikipedia.org/?curid=2477330&oldid=122480367> (дата обр. 20.02.2022).
- 3 *Википедия*. Расстояние Кульбака — Лейблера [Электронный ресурс]. — 2022. — URL: <https://ru.wikipedia.org/?curid=663021&oldid=121498117> (дата обр. 20.02.2022).
- 4 3d-front: 3d furnished rooms with layouts and semantics / H. Fu [и др.] // Proceedings of the IEEE/CVF International Conference on Computer Vision. — 2021. — С. 10933–10942.
- 5 3d-future: 3d furniture shape with texture / H. Fu [и др.] // International Journal of Computer Vision. — 2021. — С. 1–25.
- 6 Angelos K., Paschalidou D. simple-3dviz. — 2020. — <https://simple-3dviz.com>.
- 7 ATISSL: Autoregressive Transformers for Indoor Scene Synthesis / D. Paschalidou [и др.] // Advances in Neural Information Processing Systems (NeurIPS). — 2021.
- 8 Bradski G. The OpenCV Library // Dr. Dobb's Journal of Software Tools. — 2000.
- 9 Generative Adversarial Networks / I. J. Goodfellow [и др.]. — 2014. — DOI: 10.48550/ARXIV.1406.2661. — URL: <https://arxiv.org/abs/1406.2661>.
- 10 Grains: Generative recursive autoencoders for indoor scenes / M. Li [и др.] // ACM Transactions on Graphics (TOG). — 2019. — Т. 38, № 2. — С. 12.
- 11 Ritchie D., Wang K., Lin Y.-A. Fast and Flexible Indoor Scene Synthesis via Deep Convolutional Generative Models // 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). — 2019. — С. 6175–6183.

- 12 SEAN: Image Synthesis With Semantic Region-Adaptive Normalization / P. Zhu [и др.] // IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). — 06.2020.
- 13 Semantic Image Synthesis with Spatially-Adaptive Normalization / T. Park [и др.] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. — 2019.
- 14 Semantic Scene Completion from a Single Depth Image / S. Song [и др.] // Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition. — 2017.
- 15 Wang X., Yeshwanth C., Nießner M. SceneFormer: Indoor Scene Generation with Transformers // 2021 International Conference on 3D Vision (3DV). — 2021. — С. 106–115.
- 16 You Only Need Adversarial Supervision for Semantic Image Synthesis / E. Schönfeld [и др.] // International Conference on Learning Representations. — 2021. — URL: <https://openreview.net/forum?id=yvQKLaqNE6M>.

ПРИЛОЖЕНИЕ А. ТРЕХМЕРНЫЕ ВИЗУАЛИЗАЦИИ КОМНАТ

Рисунок А.1 – Пример визуализации комнаты. В конфигурации выбрана визуализация без стен, в качестве стиля — модерн.



Рисунок А.2 – Пример визуализации комнаты со стенами. В комнате присутствуют двери и окно, цвет стен был задан заранее.