ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 81 за 2020 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

В.А. Фролов, Е.Д. Феклисов, М.А. Трофимов, А.Г. Волобой

Синтез изображений интерьеров для обучения нейросетей

Рекомендуемая форма библиографической ссылки: Синтез изображений интерьеров для обучения нейросетей / В.А. Фролов [и др.] // Препринты ИПМ им. М.В.Келдыша. 2020. № 81. 20 с.
https://doi.org/10.20948/prepr-2020-81
https://library.keldysh.ru/preprint.asp?id=2020-81

О р д е н а Л е н и н а
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М. В. Келдыша
Р о с с и й с к о й а к а д е м и и н а у к

В.А. Фролов, Е.Д. Феклисов, М.А. Трофимов, А.Г. Волобой

Синтез реалистичных изображений интерьеров для обучения нейросетей

Москва — 2020

Фролов В.А., Феклисов Е. Д., Трофимов М.А., Волобой А.Г.

Синтез изображений интерьеров для обучения нейросетей

В работе предложен ряд методов, которые могут быть использованы для синтеза изображений интерьеров в целях обучения искусственного интеллекта. Предложенные методы решают проблему генерации обучающих выборок в комплексе, начиная от автоматической генерации 3D контента и заканчивая непосредственно рендерингом. Одна из основных целей разработанной системы – обеспечить достаточную производительность при генерации наборов фотореалистичных изображений интерьеров при помощи GPU.

Ключевые слова: *синтез изображений, обучение нейросетей, генерация интерьеров.*

Vladimir Alexandrovich Frolov, Egor Dmitrievich Feklisov, Maxim Alexandrovich Trofimov, Alexei Gennadievich Voloboy.

Synthesis of images of interiors for training neural networks

The paper proposes a number of methods that can be used to synthesize images of interiors in order to train artificial intelligence. The proposed methods solve the problem of generating training samples in a complex, starting from automatic generation of 3D content and ending with rendering directly. One of the main goals of the developed system is to provide sufficient performance when generating sets of photo-realistic images of interiors via using GPUs.

Key words: *Synthesis of images of interiors for training neural networks, interior sampling.*

Работа выполнена при поддержке Российского фонда фундаментальных исследований, 18-31-20032 мол_а_вед.

1. Введение

Синтез изображений для обучения нейросетей в настоящий момент является чрезвычайно перспективным направлением исследований. По сравнению с традиционным сбором и разметкой данных у синтезированных изображений есть ряд преимуществ. Это высокая точность разметки (что очень трудно добиться на изображениях, размеченных вручную), практически произвольный объём данных и возможность моделировать редко встречающиеся явления, которые в реальности трудно или невозможно собрать в большом количестве.

Однако на практике в этом подходе есть и сложности: низкая реалистичность синтезируемых изображений либо высокая трудоёмкость при создании новых обучающих выборок, а также низкая производительность рендеринга. В нашей работе мы постарались решить эти проблемы при помощи создания автоматизированных средств генерации и синтеза изображений интерьеров.

2. Предыдущие работы

Создание и рендеринг интерьеров используются во многих областях компьютерной графики: дизайн, архитектура, игры, виртуальная реальность и кино; обучение искусственного интеллекта. Компьютерная модель, которая умеет создавать новые интерьеры или хотя бы вариации одного и того же интерьера, таким образом, нужна во многих приложениях. Поэтому эта проблема актуальна и активно изучается. Создание интерьера можно разделить на 5 основных задач:

1. Генерация плана интерьера.
2. Создание схемы расположения мебели
3. Создание и настройка 3D контента для наполнения интерьера (мебель, мелкие предметы, люди и т.д.).
4. Последующее наполнение интерьера созданным 3D контентом.
5. Обеспечение реалистичности изображения при помощи корректного задания материалов и освещения.

2.1 Методы генерации плана интерьера

Создание плана может быть выполнено либо с помощью машинного обучения [1], либо более традиционными методами путем создания общего списка размещения комнат, который будет использоваться для случайной сборки результата [2]. При создании плана помещения основную сложность представляет проблема реалистичного размещения комнат: площадь должна быть использована эффективно, т. к. в реальности это сильно влияет на стоимость помещения. Поэтому относительно простые методы на основе

тайлов [3], хорошо подходящие для генерации подземелий в компьютерных играх, здесь могут быть неприменимы. Кроме того, тайловые методы в чистом виде очень сильно ограничены в вариативности получаемого результата [4], что плохо для обучения искусственного интеллекта.

Метод **«роста»** [5] помещает условные центры комнат в случайное место на 2D или 3D сетке. После чего комнаты итеративно выращиваются вокруг начальной позиции, захватывая с каждым шагом границу толщиной в 1 клетку. Все неиспользуемое пространство, таким образом, так или иначе будет использовано существующими комнатами. Преимуществом метода роста является простота реализации и возможность работы с непрямоугольными формами целевого пространства. Главный недостаток алгоритма – невозможность контролировать размер и тип комнат: например, невозможно заставить метод генерировать коридоры.

Метод **«плотной упаковки»** использует помещения заданной формы и размеров и пытается оптимальным образом разместить комнаты в ограниченном пространстве [6]. В его основе лежит одноименный класс задач оптимизации. При этом сами комнаты могут состоять из тайлов, чтобы их было проще состыковывать. Преимущество этого метода в том, что он основан на известной математической задаче, и для нее разработано множество различных решений. Этот метод очень хорош также, если требуется определенный размер или форма комнат. Основным недостатком является то, что алгоритму может потребоваться регенерировать комнаты, если их размещение в зоне невозможно, и поэтому окончательное решение может занять много времени даже без учета сложности самой задачи оптимизации.

Алгоритм **«эволюции»** использует мультиагентный поиск топологии (связь комнат друг с другом), сочетая его с процессом пространственной оптимизации [7]. В этом методе комнаты представлены в виде похожих на трёхмерные пузыри агентов, соединённых между собой нитками. В зависимости от типа комнат пузыри могут быть более или менее вытянутыми и иметь фиксированное положение (если они, например, отвечают за лестничный пролёт) или свободно плавать в 3D пространстве. Пространственная оптимизация старается минимизировать длину ниток и таким образом «стягивает» все пузыри друг к другу. К достоинствам метода следует отнести возможность работы с трёхмерными помещениями. К недостаткам метода можно отнести то, что он может работать только с прямоугольными комнатами, поскольку итоговая генерация плана осуществляется на регулярной сетке.

Метод **«изнанки»** ставит задачу генерации интерьера по-другому [8]. Если в рассмотренных ранее методах комнаты необходимо было вписать в заданную площадь, то в данном методе задача ставится наоборот: внешняя форма дома будет определяться расположением комнат. Сами комнаты при этом могут размещаться различным способом. Такой метод не походит для высотных многоэтажных зданий. Другой очевидный недостаток – трудно контролируемая внешняя форма здания, которая может получаться нереалистичной.

Методы **«на основе машинного обучения»** используют так называемые генеративно-состязательные сети [1]. Преимуществом методов на основе нейросетей является учёт социальных и других логических аспектов, которые не учитываются рассмотренными ранее методами. Серьёзным недостатком, однако, является необходимость сбора обучающей выборки. С учётом того что цель нашей работы состоит в генерации обучающих выборок, мы не будем углубляться далее в эту группу методов.

Предложенный нами метод использует комбинацию методов **«плотной упаковки»** и **«изнанки»**, поскольку для нас было в первую очередь важно сделать генератор контролируемым. Например, если пользователь хочет получить помещение с длинными коридорами и/или одной большой комнатой посередине, мы должны это учитывать. Внешняя форма здания или много-этажность при этом нас интересуют в меньшей степени.

2.2 Методы расстановки мебели

В ранних работах, исследовавших проблему автоматического размещения мебели в интерьерах, используются простые статистические отношения между объектами [10]. Следующим шагом был синтез сцены на машинного обучения. Первый такой метод изучал отдельные априорные значения для возникновения и расположения [11], но ограничен небольшими сценами из-за ограниченной доступности обучающих данных и методов обучения, доступных в то время. Были предложены различные связанные методы, моделирование ориентированных на появление объектов графических моделей в сочетании с гауссовыми шаблонами расположения смесей [12] и графами отношений объектов на основе действий [13].

Благодаря наличию большого набора данных виртуальных сцен внутри помещений, такого как SUNCG [14], были предложены новые методы на основе машинного обучения. Работа [15] использует ориентированные графы для выбора объекта, но полагается на эвристику для компоновки объектов. В [16] используется вероятностная грамматика для моделирования сцен, но также требуются данные о человеческой деятельности в сценах для некоторых наборов данных и ручные аннотации для особо важных групп объектов. В новых работах используются глубокие сверхточные сети, чтобы узнать, какие объекты должны находиться в сцене и как они должны быть расположены [17,18].

Поскольку в практике обучения нейросетей важно иметь возможность генерировать выборки с заданным распределением и типами объектов, мы не использовали методы на основе машинного обучения, которые будут расставлять предметы так, как это было в исходных обучающих выборках. Расстановка мебели по некоторому шаблону с использованием настраиваемых статистических соотношений (например [10]) больше подходит для нашей цели.

2.3 Синтез изображений для машинного обучения

Перед исследователями, решающими данную проблему, на первый план почти всегда выходит один и тот же вопрос – откуда взять контент, который бы выглядел достаточно реалистично.

2.3.1 Использование существующих баз 3D моделей

Авторы [19] использовали набор 3D моделей, полученных с сайтов-агрегаторов. К сожалению, модели, загруженные с подобных интернет-ресурсов, в большинстве случаев не готовы для фотореалистичного рендеринга из-за отсутствия текстур, наличия в них неизвестных материалов (которые неправильно экспортированы в текущий формат или не поддерживаются используемой системой рендеринга) и, наконец, просто человеческих ошибок, которые были допущены при подготовке художником 3D модели для сайта агрегатора. Такие модели можно использовать, но на практике их нужно так или иначе дорабатывать.

В [20] 3D-контент был получен из набора данных SUNCG [21]. У этого подхода много проблем. Поскольку изначально 3D-модели SUNCG были собраны в программе Planner5D (которая очень проста по сравнению с традиционными инструментами создания контента, такими как Blender или 3ds Max), они не содержат реалистичных настроек материалов или освещения, ограничиваясь только диффузными текстурами в низком разрешении. То есть этот контент не был подготовлен для фотореалистичного рендеринга. Неудивительно, что другие работы, использующие SUNCG [22–24], также смогли достичь лишь базового уровня реализма.

2.3.2 Дополненная реальность

Идея использования дополненной реальности заключается в том, чтобы сократить затраты на подготовку 3D контента за счёт использования фотографий реального мира. Уже упомянутая работа [20] использует эту идею при рандомизации фона. В целом дополненная реальность даёт определённые результаты [25], но имеет и серьёзные ограничения:

1. [26] отмечает, что масштабируемость данной группы методов ограничена из-за меньшей вариативности данных в обучающей выборке, чем при обучении только на синтезированных данных за счет ограничения ракурса камеры, и в действительности это является лишь незначительным улучшением по отношению к обучению только на реальных данных (итоговое повышение точности в [25] составило несколько процентов).
2. Реалистичная дополненная реальность требует реконструкции освещения и окружения, что в случае интерьеров само по себе является нетривиальной задачей. Хотя реконструкция окружения является стандартной задачей в дополненной реальности и методы освещения [27–29] могут оказаться полезными.

3. Существенным недостатком дополненной реальности является требование автоматического правильного размещения 3D-моделей внутри некоторого реконструированного представления сцены, что в некоторых случаях может быть несложным (например, в [25] использовали методы сегментации для восстановления движущейся плоскости дороги), но является фундаментально трудной задачей в общем случае и, в частности, для интерьеров, изображения которых не поддаются простому шаблону.

2.3.3 Процедурная генерация

Процедурное моделирование – это дорогой и трудоёмкий процесс. Оно требует времени и обычно ограничивается определенной задачей (например, моделирование грязи или ржавчины), а иногда и конкретным алгоритмом рендеринга. Тем не менее один раз созданные модели изменчивы, дают нужное качество для рендеринга (детализация с бесконечным разрешением) и традиционно используются во многих конвейерах создания контента для кино.

Авторы [26] использовали метод процедурного моделирования для создания городов и продемонстрировали применимость этого подхода для тренировки нейронных сетей. Здания, дороги и план города были процедурными, в то время как другие модели (автомобили, пешеходы, растительность, велосипеды и др.) выбирались случайным образом из подготовленной базы данных. Также были рандомизированы дополнительные параметры для автомобилей: тип; количество; размещение; цвет. Перспективными выглядят подходы, основанные на процедурном моделировании с помощью машинного обучения [30;31], но здесь мы снова встречаемся с проблемой курицы и яйца: сначала нам нужно обучить эти модели.

Алгоритмы процедурной генерации планов интерьеров были рассмотрены нами в начале обзора и используются в нашей работе.

2.3.4 Рендеринг и физическая симуляция

Авторы [32] достигли высокого качества за счет использования существующего конвейера кинопроизводства на основе Autodesk Maya и системы рендеринга Arnold, высококачественных 3D-моделей и физического моделирования, которое было основным инструментом рандомизации. Очевидно, этот подход страдает основным недостатком существующих конвейеров кинопроизводства – высокая стоимость и высокая трудоемкость: [32] использовали только 6 сцен с 30 различными объектами, что принципиально отличается от предыдущих подходов в худшую сторону. Другие недостатки включают использование проприетарных инструментов (Maya и Arnold), что ограничивает внедрение этой работы. Наконец, существенное время рендеринга – от 15 до 720 секунд на 1 изображение в зависимости от настроек качества на 16-ядерных процессорах Xeon с объемом памяти 112GB оперативной памяти. Следовательно, достаточно быстрое

создание набора данных возможно только при наличии значительных вычислительных ресурсов. Важно отметить, что авторы сообщают, что точное моделирование контекста сцены было намного более значимым (+ 16% для точности CNN) по сравнению с точным расчётом освещения (+ 6% для точности CNN).

Другие работы [33] показывают, что современные игровые движки также могут успешно использоваться для обучения искусственного интеллекта. Поэтому выбор между игровым движком или высококачественной системой рендеринга должен быть сделан в первую очередь на основе того, насколько легко будет моделировать в нём целевую ситуацию.

2.3.5 Специализированные инструменты

В отличие от работ, упомянутых ранее, в [34] предлагается специализированный конвейер создания рандомизированного контента. Разработанные авторами [34] инструменты расширяют программу 3D моделирования Blender, позволяя художнику генерировать позиции для размещения объектов (камеры, источники света, 3D-модели) с различными распределениями и ограничениями. Например, можно задать, чтобы камеры не смотрели в стену, объекты не сталкивались. Инструменты художника также позволяют выбирать объекты на основе определенных пользователем условий и управлять их свойствами. Например, включать физическое моделирование.

Основным недостатком этой работы является тот факт, что на сегодняшний день она еще не применялась к какой-либо конкретной задаче обучения искусственного интеллекта, в отличие от многих ранее упомянутых работ. Тем не менее мы полагаем, что направление [34] в целом правильное. Мы строим своё решение аналогичным образом, комбинируя инструменты художника с процедурной генерацией и рендерингом на GPU.

3. Предложенные методы

Целью нашей работы было создание комплексного решения, и по аналогии с [34] мы сфокусировались на создании конвейера, с которым может работать 3D художник. Мы разбили эту задачу на несколько частей, о которых будет написано далее, а итоговое приложение выполнено нами в виде плагина для системы моделирования Autodesk 3ds Max. Мы не использовали какие-либо современные методы на основе искусственного интеллекта или автоматические методы для создания 3D-контента, потому что нашим основным требованием является высокая степень контроля над сгенерированным результатом, а это проблема для методов на основе нейронных сетей.

3.1 Генерация плана помещения

Как мы уже упоминали, предложенный нами метод использует комбинацию методов **«плотной упаковки»** и **«изнанки»**, поскольку для нас было в первую очередь важно сделать генератор контролируемым. План этажа генерируется на основе правил и размеров комнаты, задаваемых пользователем в формате JSON. На основе этих входных данных алгоритм упаковки пытается заполнить область комнатами, после чего (если это получается) формируется внешняя стена вокруг модели этажа. Исходно мы собирались генерировать несколько этажей (но не более 3). Ниже представлен предложенный алгоритм, который состоит из 2 больших частей. Сначала мы генерирует план (1), после чего собираем из него 3D модель (2):

(1) Генерация плана.
   (a) Алгоритм запускается со случайным количеством этажей и комнат разных типов.
   (b) Проверяет, достаточно ли комнат каждого требуемого типа (например, ванных комнат). В противном случае переход к шагу (a).
   (c) Помещения случайным образом распределяются по этажам, при этом им присваиваются определенные размеры и тип (ванная, гостиная, коридор). Кроме того, алгоритм добавляет лестницу на каждый этаж, кроме последнего.
   (d) Алгоритм переходит к шагу (c) если выполнено одно из условий:
      i. некоторые этажи пусты;
      ii. верхний этаж больше нижнего этажа;
      iii. на некоторых этажах все комнаты одного типа;
      iv. нарушено хотя бы одно заданное пользователем ограничение.

(2) Сборка 3D модели.
   (a) Для каждого этажа алгоритм ищет связанные комнаты и собирает их в кластеры. Для каждого кластера рассчитывается его общее пространство, а затем комнаты размещаются на основе алгоритма плотной упаковки внутри кластера.
   (b) Кластеры, отдельные комнаты и лестницы с использованием алгоритма плотной упаковки размещаются на этаже.
   (c) Внешняя стена обводится вокруг конструкции, созданной на верхнем этаже.
   (d) Алгоритм переходит на следующий этаж. На этот раз площадь пола уменьшается на пространство лестницы с предыдущего этажа, а над ней делается отверстие.

Полученные 3D модели (без пола в целях демонстрации многоэтажности) продемонстрированы на рис. 1. Таким образом, наша реализация обладает необходимой гибкостью, контролируема и может создать многоэтажное здание

с соединительными лестницами. Кроме того, мы поддерживаем непрямоугольные формы помещений и полов. Однако есть несколько ограничений:

1. Добавление пользователем новых правил для генерации плана (которые подаются на вход алгоритму) может быть трудоёмкой задачей, поскольку они друг с другом связаны, и на практике настраивать работу генератора художнику не легко.
2. Добавление окон, дверей и других геометрических деталей в нашем подходе превращается в нетривиальную и ресурсоёмкую задачу обработки мешей, поскольку необходимо на лету создавать и модифицировать меш. В этом смысле более простые подходы на основе тайлов, где окна, двери и другие геометрические детали уже пред-просчитаны внутри каждого тайла, находятся в более выигрышной позиции.

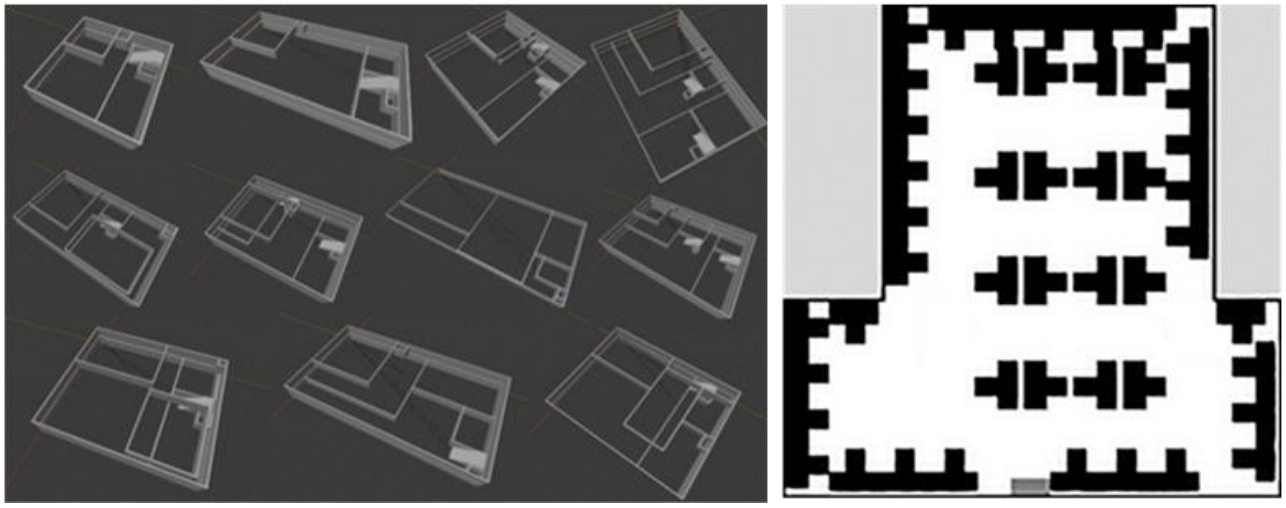

Рис. 1. Демонстрация 3D моделей сгенерированных планов помещений (справа) и демонстрация алгоритма размещения мебели по шаблону (слева).

3.2 Размещение мебели

Хотя для этой задачи существует множество передовых методов (упомянутых в обзоре), нам было важно разработать контролируемый 3D художником метод, который бы в то же время умел воспроизводить строго определённое положение мебели для одного и того же зерна генератора случайных чисел. Это важно, поскольку позволяет пользователям проверять различные гипотезы за счёт генерации специфичных выборок изображений. Например: (1) выборки, где изменялось положение мебели, но не изменялись сами 3D модели, (2) выборки, где изменялись 3D модели, но не изменялось их положение.

Поэтому мы сначала обходим края периметра каждой комнаты и расставляем мебель по одномерному шаблону. Затем, чтобы расставить мебель внутри комнаты, художник рисует несколько бинарных 2D-изображений. Во

время выборки мы случайным образом выбираем шаблон и частоту. Результат представлен на рис. 1 справа.

3.3 Размещение мелких объектов

Для размещения объектов на столах мы также использовали шаблонный метод. Проведя несколько экспериментов с физическим моделированием в начале нашего проекта аналогично [32], мы обнаружили, что на практике с его помощью чрезвычайно трудно добиться адекватного и воспроизводимого результата: объекты сваливались в кучу, проваливались под стол или находились в невозможных конфигурациях (книга на клавиатуре, мышка на мониторе, системный блок по центру на столе и т. п.).

Мы остановились на подходе, когда художник подготовил несколько шаблонов с настроенными ограничивающими рамками и углами поворота для каждого типа объекта на столе или на поверхности пола: монитор, системный блок, клавиатура и др. Если в плоскости стола происходит столкновение объектов, мы применяем правило «чистой доски», очищая всё и заново регенерируя до тех пор, пока не исчезнут столкновения [35]). Это гарантирует, что полученное распределение объектов будет такое же, как и при использовании физической симуляции [35]. Полученные изображения (вид сверху) можно увидеть на рис. 2.

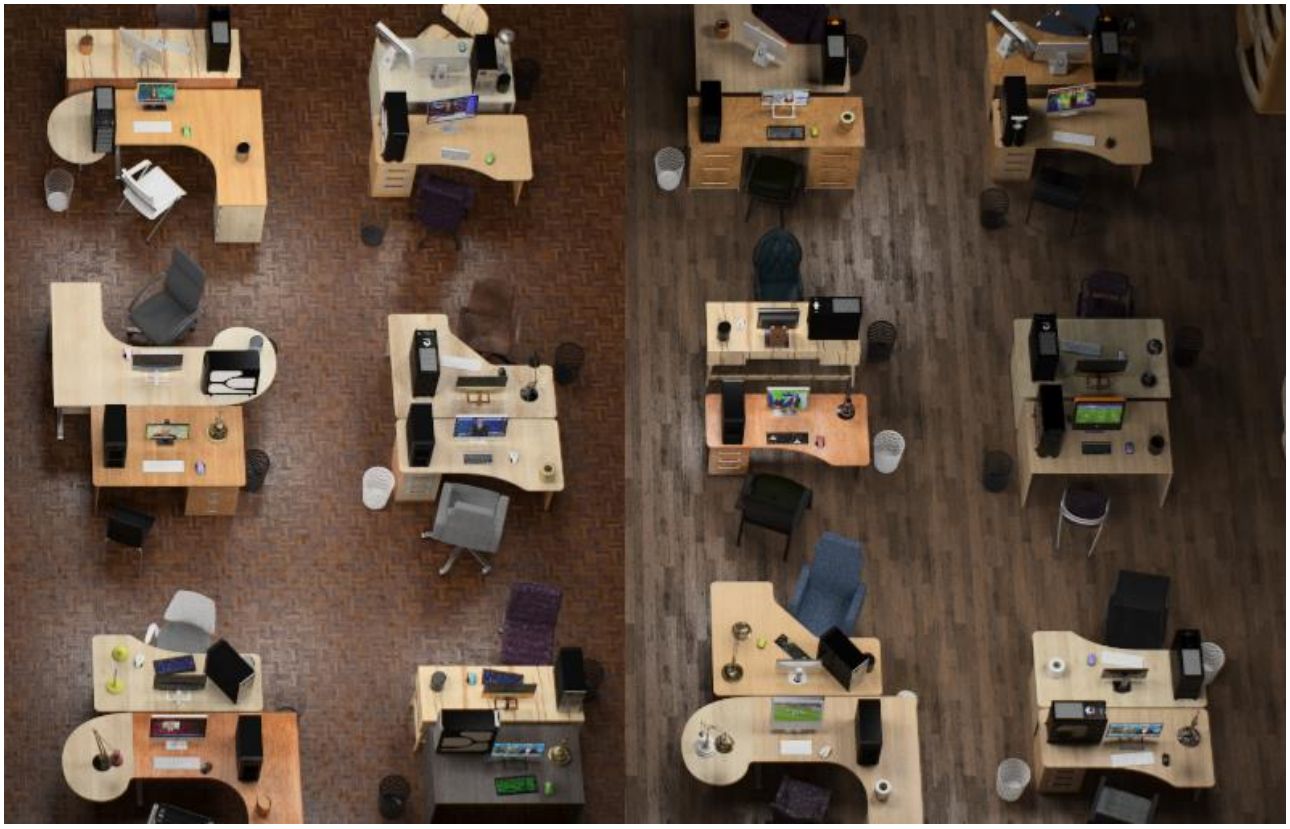

Рис. 2. Демонстрация автоматической расстановки предметов по шаблону.

3.4 Материалы и освещение

Следующая серьезная проблема заключается в том, что современные системы рендеринга используют исключительно собственные модели освещения и материалов, которые несовместимы с другими системами. Реалистично выглядящий контент компьютерной графики создается для целевой системы рендеринга и не может использоваться напрямую в других. Поэтому на практике не существует открытых баз данных реалистичных 3D-моделей, т.к. импорт / экспорт 3D-контента из одной системы рендеринга в другую – нетривиальная задача. Открытый стандарт GLTF [36] может быть решением, но пока в открытом доступе подобных моделей, к сожалению, крайне мало. Принимая во внимание факт необходимости рандомизации материалов и освещения, нам пришлось построить собственный конвейер создания контента для адаптации существующих 3D-моделей, их материалов и настройки освещения.

Мы понимали, что в процесс работы художника придётся вносить существенные изменения, поэтому знакомая инфраструктура сыграла не последнюю роль в выборе системы рендеринга. В качестве основы для нашего решения мы использовали разработанную нами собственную GPU рендер-систему Hydra Renderer [37], которая имеет развитый конвейер для создания 3D контента с возможностью преобразования материалов из других популярных систем рендеринга (VRay, Mental, Corona) и обеспечивает высокую производительность рендеринга благодаря GPU и эффективным алгоритмам расчёта освещения.

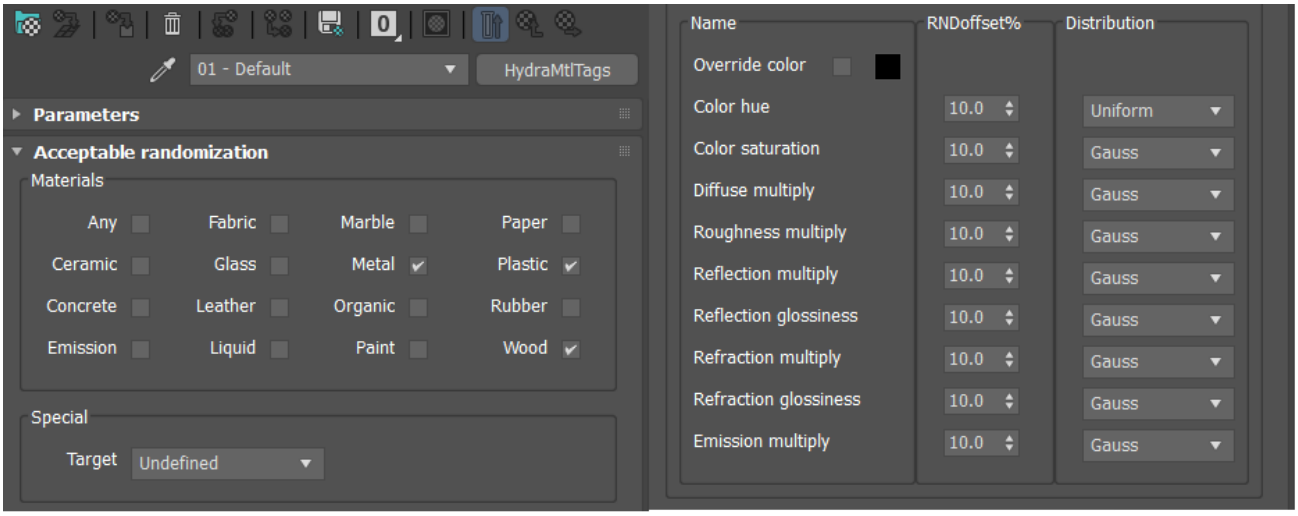

Рис. 3. Скриншот разработанного нами специализированного редактора материалов. В левой части посередине расположен выбор значений типов материалов. Слева внизу расположен параметр target, позволяющий ограничить применение данного материала только на определённый тип объектов. Справа находятся настройки распределений.

Для того чтобы 3D художник мог настраивать рандомизированный контент, мы разработали специализированный плагин материала (рис. 3), позволяющий настраивать распределение и осуществлять предпросмотр работы генератора на различных 3D моделях из базы (рис. 4). База 3D моделей была собрана нами из открытых источников. Для каждой 3D модели затем вручную 3D художник назначал рандомизированный материал для отдельных частей.

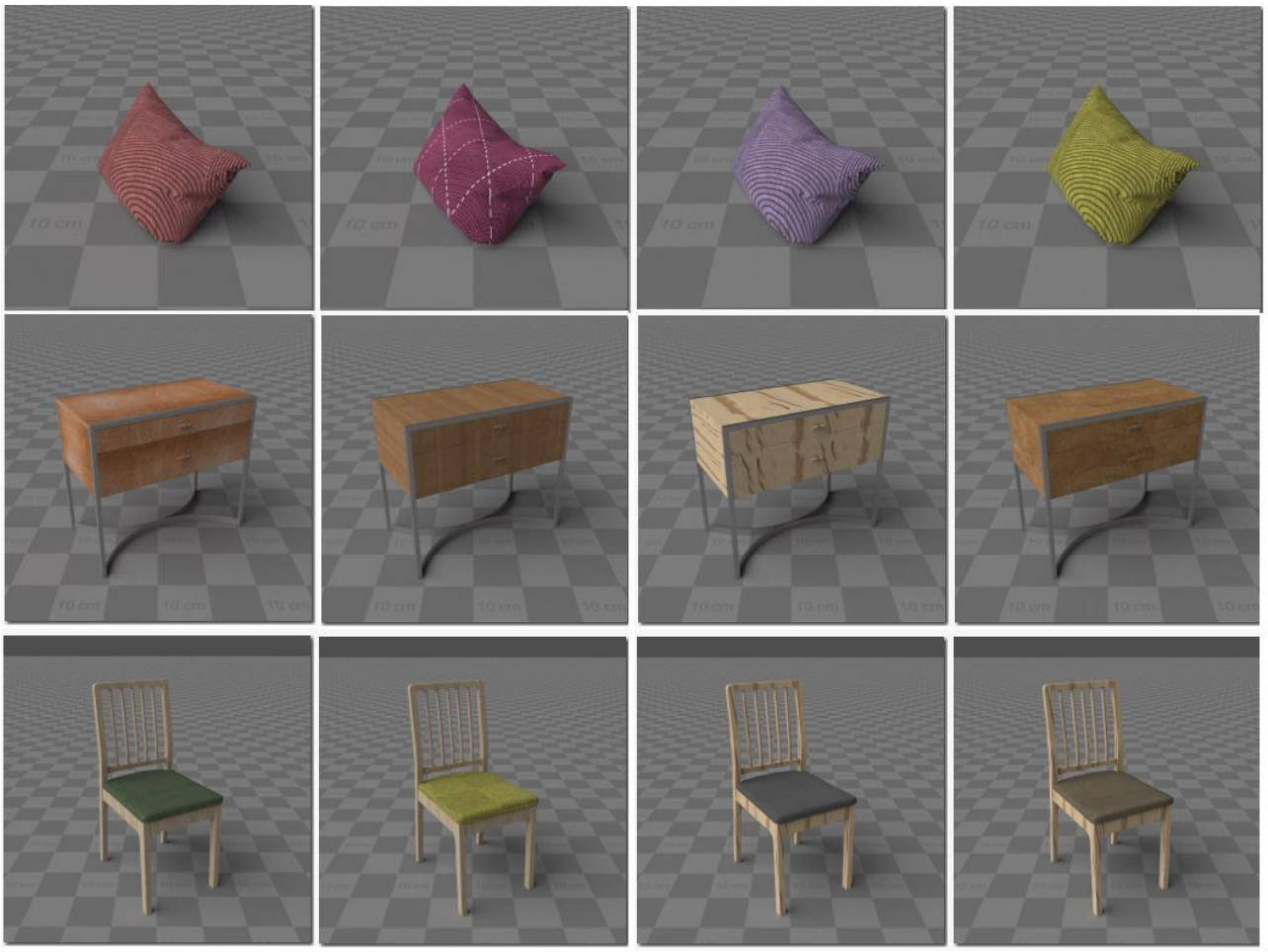

Рис. 4. Демонстрация предпросмотра работы рандомизатора материалов на 3D моделях из базы.

Художник определяет логику рандомизации, задав специальные параметры материала (рис. 2). Это позволяет ограничить рандомизацию и сделать ее реалистичной. Например, параметр «Target» (рис. 2, внизу и слева), приобретая определенное значение, позволяет использовать этот материал только на определенной части определенного класса моделей. Такое ограничение существенно, поскольку многие материалы содержат в себе текстурные матрицы – отображение текстурных координат, применяемое при рендеринге для корректного наложения текстуры на 3D модель. Это отображение всегда тщательно настраивается художниками для каждого класса

3D моделей, поэтому деревянную текстуру пола, например, нельзя применить на деревянный карандаш или деревянный стул, хотя все эти вещи и сделаны из дерева.

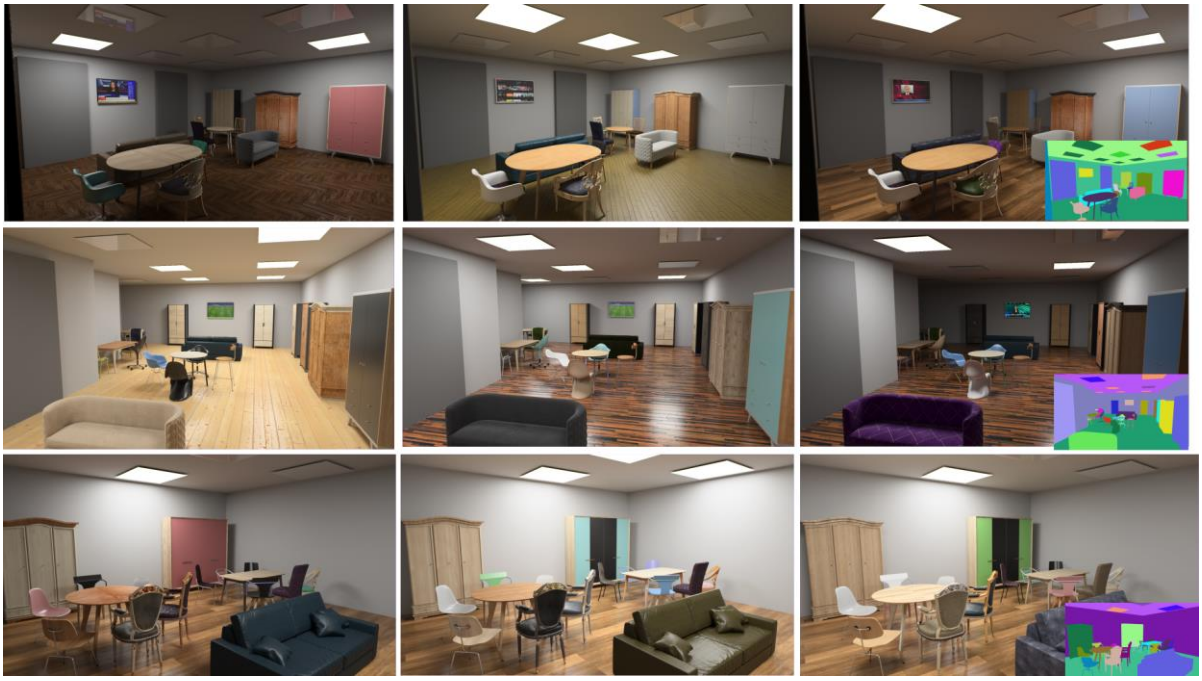

Рис. 5. Примеры синтезированных изображений сгенерированных интерьеров, полученные при помощи разработанной системы. Маленькое цветное изображение в правом нижнем углу – маска идентификаторов объектов.

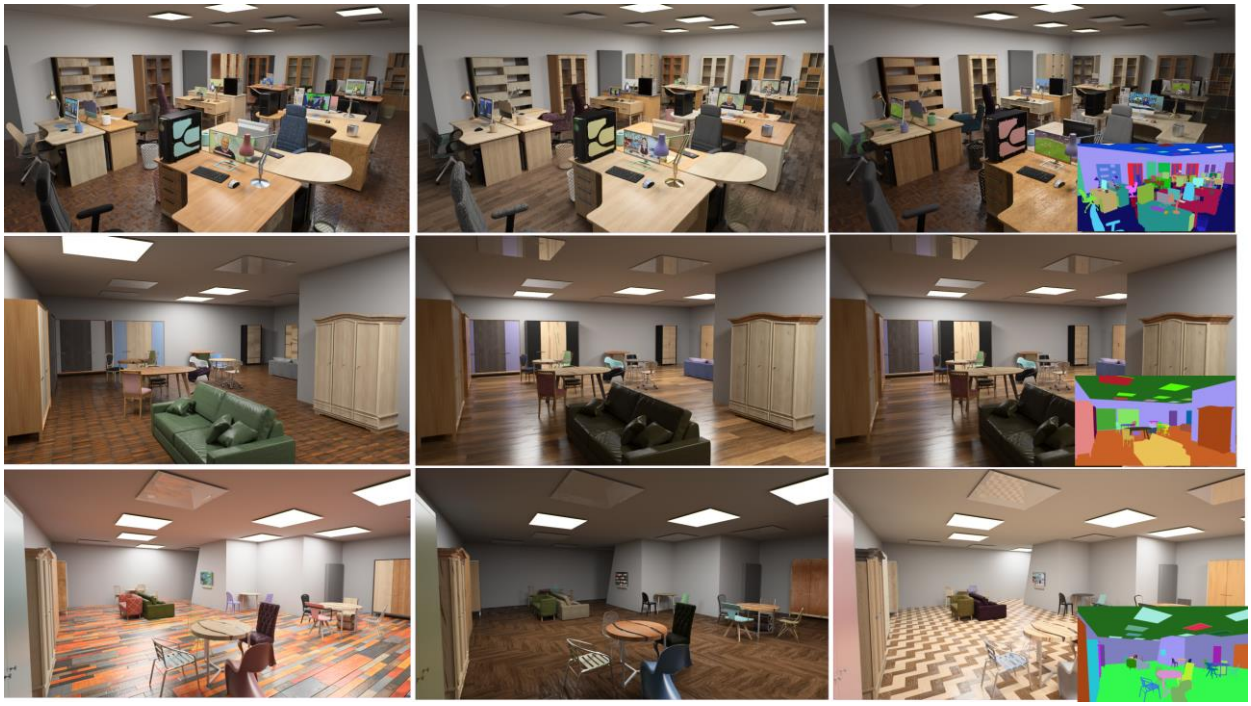

Рис. 6. Примеры синтезированных изображений сгенерированных интерьеров, полученные при помощи разработанной системы. Маленькое цветное изображение в правом нижнем углу – маска идентификаторов объектов.

3.5 Производительность и итоговое решение

На рисунках 5, 6 представлены синтезированные разработанной программной системой изображения. Наше решение способно генерировать примерно 10 изображений в час на одном графическом процессоре K100, поэтому для создания полного набора обучающих данных на 8 GPU обычно требуется около 2 дней. Тем не менее это не было узким местом в работе, поскольку настройка и отладка скриптов для специфического сценария генерации обучающей выборки могла занимать до 2 недель.

4. Заключение

Генерация рандомизированного реалистичного 3D контента для обучения нейронных сетей с высокой вариативностью является комплексной задачей, в которой нельзя выделить какое-то одно узкое место. Главное требование для подобной системы – развитая инфраструктура, позволяющая быстро и легко компоновать результаты работы различных алгоритмов и фрейм-ворков вместе, осуществлять быстрое прототипирование и настройку алгоритмов генерации. А главное требование к рендер-системе – автоматизированная настройка материалов, подвергающихся рандомизации, и непосредственная связь с базой используемых генератором 3D моделей. Последнее является сильной стороной нашего решения. Однако с точки зрения инфраструктуры, к сожалению, есть недостатки.

1. Наша система сильно фрагментирована, а всё взаимодействие организовано через файлы. Генератор планов использует API Blender, скрипт расстановки мебели выполнен в Unity, рандомизатор мелких объектов, материалов и освещения использует скрипты 3ds Max, а сам рендеринг работает на сервере и управляется скриптами на Python и Bash, которые взаимодействуют с плагином для 3ds Max через базу данных, где часть информации хранится в SQL, а часть (модели и текстуры) – просто в файлах.
2. Полностью автоматического решения не получилось, поскольку почти на каждом этапе сгенерированные данные приходилось фильтровать. Например, до 40% планов интерьеров, полученных нашим алгоритмом, 3D художник отфильтровывал, полагая их нереалистичными. Кроме того, почти любое обновление в базе 3D моделей и материалов вело к обновлению логики рандомизации в итоговых сценариях генерации обучающих выборок.
3. Написание же самих сценариев генерации было трудоёмким процессом, поскольку, для того чтобы отладить эти сценарии, необходимо просмотреть большое количество сгенерированных изображений. Причём эту процедуру нужно повторять неоднократно.

Тем не менее мы полагаем, что указанные недостатки не являются критическими, поскольку любая подобная система должна пройти долгий путь

от прототипа до индустриального применения, и возможностей разработанного нами генератора достаточно для его апробации в практике обучения искусственного интеллекта.

Список литературы

1. Merrell P., Schkufza E., Koltun V. *Computer-generated residential building layouts* //ACM SIGGRAPH Asia 2010 papers. С. 1-12.
2. Bengtsson D., Melin J. *Constrained procedural floor plan generation for game environments* // Master of Science in Game and Software Engineering. Blekinge Institute of Technology, Karlskrona, Sweden, 2016.
3. Michael Cerny Green, Ahmed Khalifa, Athoug Alsoughayer, Divyesh Surana, Antonios Liapis, and Julian Togelius. 2019. *Two-step constructive approaches for dungeon generation //* In Proceedings of the 14th International Conference on the Foundations of Digital Games (FDG '19). Association for Computing Machinery, New York, NY, USA, Article 84, 1–7.
4. Aanholt, Levi & Bidarra, Rafael. *Declarative procedural generation of architecture with semantic architectural profiles* // Proceedings of CoG 2020 - IEEE Conference on Games.
5. Lopes R. et al. *A constrained growth method for procedural floor plan generation* // Proc. 11th Int. Conf. Intell. Games Simul. – 2010. – С. 13-20.
6. Campbell M. I., Koenig R., Knecht K. *Comparing two evolutionary algorithm based methods for layout generation: Dense packing versus subdivision* // Artificial Intelligence for Engineering Design, Analysis and Manufacturing. – 2014. – Т. 28. – №. 3. – С. 285-299.
7. Guo Z., Li B. *Evolutionary approach for spatial architecture layout design enhanced by an agent-based topology finding system* // Frontiers of Architectural Research. – 2017. – Т. 6. – №. 1. – С. 53-62.
8. Martin J. *Procedural house generation: A method for dynamically generating floor plans* // Proceedings of the Symposium on Interactive 3D Graphics and Games. – 2006. – С. 1-2.
9. Marson F., Musse S. R. *Automatic real-time generation of floor plans based on squarified treemaps algorithm* // International Journal of Computer Games Technology. – 2010. – Т. 2010.
10. Yu L. F. et al. *Make it home: automatic optimization of furniture arrangement* // ACM Transactions on Graphics (TOG)-Proceedings of ACM SIGGRAPH 2011, v. 30,(4), July 2011, article no. 86. – 2011. – Т. 30. – №. 4.
11. Fisher M. et al. *Example-based synthesis of 3D object arrangements* // ACM Transactions on Graphics (TOG). – 2012. – Т. 31. – №. 6. – С. 1-11.
12. Henderson P., Ferrari V. A generative model of 3d object layouts in apartments / arXiv preprint arXiv: 1711.10939. – 2017.
13. Fu Q. et al. *Adaptive synthesis of indoor scenes via activity-associated object relation graphs* //ACM Transactions on Graphics.2017.Т.36. – №. 6. – С. 1-13.

14. Song S. et al. *Semantic scene completion from a single depth image* // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. – 2017. – С. 1746-1754.
15. Henderson P., Subr K., Ferrari V. *Automatic Generation of Constrained Furniture Layouts* / arXiv preprint arXiv:1711.10939. – 2017.
16. Qi S. et al. *Human-centric indoor scene synthesis using stochastic grammar* // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. – 2018. – С. 5899-5908.
17. Wang K. et al. Deep convolutional priors for indoor scene synthesis // ACM Transactions on Graphics (TOG). – 2018. – Т. 37. – №. 4. – С. 1-14.
18. Ritchie D., Wang K., Lin Y. *Fast and flexible indoor scene synthesis via deep convolutional generative models* // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. – 2019. – С. 6182-6190.
19. Movshovitz-Attias, T. Kanade, and Y. Sheikh. 2016. *How useful is photo-realistic rendering for visual learning* // European Conference on Computer Vision. Springer,2016, С. 202–217.
20. Zhang Y., Song S., Yumer E., Savva M., Lee J.-Y., Jin H., and Funkhouser T. *Physically-based rendering for indoor scene understanding using Krauth convolutional neural networks* // In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5287–5295.
21. Shuran Song, Fisher Yu, Andy Zeng, Angel X. Chang, Manolis Savva, Thomas Funkhouser. *Semantic Scene Completion from a Single Depth Image* / arXiv preprint, arXiv: 1611.08974, 2016.
22. Pavel Kirsanov, Airat Gaskarov, Filipp Konokhov, Konstantin Sofiiuk, Anna Vorontsova, Igor Slinko, Dmitry Zhukov, Sergey Bykov, Olga Barinova, Anton Konushin. 2019. *DISCOMAN: Dataset of Indoor Scenes for Odometry, Mapping And Navigation* / arXiv preprint arXiv: 1909.12146 (2019).
23. Li Z., Snavely N. *Cgintrinsics: Better intrinsic image decomposition through physically-based rendering* // Proceedings of the European Conference on Computer Vision (ECCV). – 2018. – С. 371-387.
24. McCormac J. et al. *SceneNet RGB-D: Can 5M synthetic images beat generic ImageNet pre-training on indoor segmentation?* // Proceedings of the IEEE International Conference on Computer Vision. – 2017. – С. 2678-2687.
25. Alhaija H. A. et al. *Augmented reality meets computer vision: Efficient data generation for urban driving scenes* // International Journal of Computer Vision. – 2018. – Т. 126. – №. 9. – С. 961-972.
26. Tsirikoglou A. et al. *Procedural modeling and physically based rendering for synthetic data generation in automotive applications* / arXiv preprint arXiv: 1710.06270. – 2017.
27. Song S., Funkhouser T. Neural illumination: Lighting prediction for indoor environments // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. – 2019. – С. 6918-6926.

28. Garon M. et al. *Fast spatially-varying indoor lighting estimation* // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. – 2019. – С. 6908-6917.
29. Sorokin M.I., Zhdanov D.D., Zhdanov A.D., Potemin I.S., Bogdanov N.N. *Restoration of Lighting Parameters in Mixed Reality Systems Using Convolutional Neural Network Technology Based on RGBD Images* // Programming and Computer Software, 2020, Vol. 46, No. 3, pp. 203-212
30. Risi S., Togelius J. *Increasing generality in machine learning through procedural content generation* // Nature Machine Intelligence – 2020. – С. 1-9.
31. Spick R. J., Cowling P., Walker J. A. P*rocedural Generation using Spatial GANs for Region-Specific Learning of Elevation Data* / 2019 IEEE Conference on Games (CoG). – IEEE, 2019. – С. 1-8.
32. Hodan T., Vineet V., Gal R., Shalev E., Hanzelka J., Connell T., Urbina P., Sinha S. N., and Guenter B. *Photorealistic image synthesis for object instance detection* / arXiv preprint arXiv:1902.03334, 2019.
33. Shah S. et al. Airsim: *High-fidelity visual and physical simulation for autonomous vehicles* // Field and service robotics. – Springer, Cham, 2018. – С. 621-635.
34. Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Youssef Zidan, Dmitry Olefir, Mohamad Elbadrawy, Ahsan Lodhi, Harinandan Katam. 2019. *BlenderProc*. / arXiv:1911.01911v1.
35. Werner Krauth. Advanced Monte Carlo algorithms / 2015. Course lecture. Pp. 11-12. http://www.lps.ens.fr/~krauth/images/5/50/BadHonnef_2.pdf (accesed 01.09.2020)
36. Robinet F. et al. *glTF: Designing an Open-Standard Runtime Asset Format Fabrice Robinet, Re mi Arnaud, Tony Parisi, and Patrick Cozzi* //GPU Pro 360 Guide to 3D Engine Design. – AK Peters/CRC Press, 2018. – С. 243-260.
37. Frolov V., Sanzharov V., Galaktionov V. Open source rendering system Hydra Renderer. https://github.com/Ray-Tracing-Systems/HydraAPI (accesed 01.09.2020)

Оглавление