Посвящается моей жене и семье,
за их поддержку и вдохновение на всех этапах работы
– Вишах Хегде

Посвящается моей семье и друзьям,
чья любовь и поддержка были моей самой большой мотивацией
– Лилит Йольян

Содержание

От издательства 10

Об авторах 11

О рецензентах 12

Предисловие 14

Часть I. Основы обработки 3D-данных 18

Глава 1. Введение в обработку 3D-данных 19

Технические требования 20
Настройка среды разработки 20
Представление 3D-данных 21
  Представление в виде облака точек 22
  Представление в виде полигональной сетки 22
  Представление в виде воксела 23
Формат файла 3D-данных – файлы PLY 24
Формат файла 3D-данных – файлы OBJ 29
Понятие системы 3D-координат 37
Понятие модели камеры 39
Пример программирования моделей камеры и систем координат 40
Резюме 43

Глава 2. Введение в трехмерное компьютерное зрение и геометрию 44

Технические требования 45
Ознакомление с базовыми понятиями отрисовки, растеризации и затенения 45
  Понятие барицентрических координат 47
  Модели источника света 48
  Концепция модели затенения по Ламберту 48
  Концепция модели освещения по Фонгу 49

Пример программирования 3D-отрисовки 50
Использование разнородных пакетов данных в библиотеке PyTorch3D и оптимизаторов PyTorch 57
Пример программирования разнородных мини-пакетов 59
Понятия трансформации и поворота 63
Примеры программирования трансформации и поворота 65
Резюме 66

Часть II. Трехмерное глубокое обучение с использованием библиотеки PyTorch3D 68

Глава 3. Подгонка деформируемых сеточных моделей к необработанным облакам точек 69

Технические требования 70
Задача подгонки полигональных сеток к облакам точек 70
Формулирование задачи подгонки деформируемой полигональной сетки в задачу оптимизации 73
Функции потери для регуляризации 74
Функция потери с учетом лапласианова сглаживания полигональной сетки 74
Функция потери с учетом согласованности нормалей полигональной сетки 75
Функция потери с учетом длин ребер полигональной сетки 75
Реализация подгонки полигональной сетки с помощью библиотеки PyTorch3D 76
Эксперимент без использования каких-либо регуляризационных функций потери 80
Эксперимент с использованием только одной функции потери – потери с учетом длин ребер полигональной сетки 81
Резюме 82

Глава 4. Обнаружение и отслеживание позы объекта с помощью дифференцируемой отрисовки 83

Технические требования 85
Зачем нужна дифференцируемая отрисовка 85
Как сделать отрисовку дифференцируемой 86
Какие задачи можно решать с использованием дифференцируемой отрисовки 89
Задача оценивания поз объекта 90
Как это программируется 93
Пример оценивания позы объекта для подгонки силуэта и подгонки текстуры 100
Резюме 107

Глава 5. Понятие дифференцируемой объемометрической отрисовки 109

Технические требования 110
Общий обзор объемометрической отрисовки 110
Понятие отбора лучей 112
Применение отбора объемов 115
Обследование лучевого маршировщика 116
Дифференцируемая объемометрическая отрисовка 118
Реконструкция 3D-моделей по многоракурсным изображениям 118
Резюме 123

Глава 6. Обследование нейронных полей яркости излучения (NeRF) 124

Технические требования 125
Концепция нейронных полей яркости излучения (NeRF) 125
Что такое поле яркости излучения? 126
Представление полей яркости излучения с помощью нейронных сетей 127
Тренировка модели NeRF 128
Понимание архитектуры модели NeRF 136
Понимание объемной отрисовки с использованием полей яркости излучения 142
Проецирование лучей на сцену 143
Накопление цвета луча 143
Резюме 144

Часть III. Современное трехмерное глубокое обучение с использованием библиотеки PyTorch3D 145

Глава 7. Обследование контролируемых нейронных полей признаков 146

Технические требования 147
Концепция синтеза изображений на основе GAN-сети 147
Введение в композиционный 3D-информированный синтез изображений 149
Генерирование полей признаков 152
Отображение полей признаков в изображения 153
Обследование контролируемой генерации сцен 155
Обследование контролируемой генерации автомобилей 156
Обследование контролируемой генерации лиц 158
Тренировка модели GIRAFFE 160
Начальное расстояние Фреше 161
Тренировка модели 161
Резюме 162

Глава 8. Моделирование человеческого тела в 3D 164

Технические требования 165
Постановка задачи 3D-моделирования 165
Определение подходящего представления 166
Концепция техники линейно-переходного кожного покрова 168
Концепция модели SMPL 170
Определение модели SMPL 170
Форма и шаблонная полигональная сетка в зависимости от позы 171
Суставы в зависимости от формы 171
Применение модели SMPL 172
Оценивание позы и формы человека в 3D с помощью метода SMPLify 174
Определение целевой функции оптимизации 175
Обследование метода SMPLify 176
Выполнение исходного кода 177
Обследование исходного кода 178
Резюме 182

Глава 9. Сквозной синтез ракурсов с помощью модели SynSin 183

Технические требования 184
Общий обзор синтеза ракурсов 184
Сетевая архитектура модели SynSin 185
Сети пространственных признаков и глубин 186
Нейронный отрисовщик облака точек 187
Модуль уточнения и дискриминатор 190
Тренировка и тестирование модели на практике 191
Резюме 201

Глава 10. Модель Mesh R-CNN 202

Технические требования 203
Общий обзор полигональных сеток и вокселов 203
Архитектура модели Mesh R-CNN 204
Графовые свертки 207
Предсказатель полигональной сетки 209
Демонстрация модели Mesh R-CNN с помощью PyTorch3D 212
Демонстрационный пример 212
Резюме 220

Тематический указатель 221

От издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, если вы сообщите о ней главному редактору по адресу dmkpress@gmail.com. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательство «ДМК Пресс» очень серьезно относится к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Об авторах

Ксудонг Ма – штатный инженер машинного обучения в Grabango Inc. Беркли, штат Калифорния. Работал старшим инженером машинного обучения в Facebook (Meta) Oculus и тесно сотрудничал с коллективом 3D PyTorch в проектах отслеживания лица в 3D. Имеет многолетний опыт работы в области компьютерного зрения, машинного и глубокого обучения и имеет докторскую степень в области электромашиностроения и конструирования вычислительных машин.

Вишах Хегде – исследователь в области машинного обучения и компьютерного зрения. Имеет более 7 лет опыта работы в указанных областях, во время которых стал автором нескольких хорошо процитированных исследовательских работ и опубликованных патентов. Имеет степень магистра Стэнфордского университета по специализации «Прикладная математика и машинное обучение», а также степени бакалавра и магистра по физике университета IIT в Мадрасе. Ранее работал в Schlumberger и Matroid. Является старшим прикладным исследователем в Ambient.ai, где помогал разрабатывать систему обнаружения оружия, которая развернута в нескольких глобальных компаниях из списка Fortune 500. Сейчас он использует свой опыт и страсть к решению деловых задач с целью создания технологического стартапа в Кремниевой долине. Подробнее о нем можно узнать на его сайте.

Хотел бы поблагодарить исследователей в области компьютерного зрения, прорывное исследование которых мне пришлось описывать. Хотел бы поблагодарить рецензентов за их отзыв и замечательный коллектив издательства Packt Publishing за то, что он дал мне возможность проявить творческий подход. Наконец, хочу поблагодарить свою жену и семью за их поддержку и вдохновение в ситуации, когда мне это было нужно больше всего.

Лилит Йольян – исследователь машинного обучения, работающая над докторской диссертацией в университете в YSU. Ее исследования посвящены разработке технологических решений в области компьютерного зрения для умных городов с использованием данных дистанционной съемки. Имеет 5-летний опыт работы в области компьютерного зрения и работала над технически сложным решением по обеспечению безопасности водителя, планируемым к развертыванию многими известными компаниями-производителями автомобилей.

О рецензентах

Эйя Абид – студентка магистратуры в области машиностроения со специализацией «Глубокое обучение и компьютерное зрение». Занимает пост преподавателя ИИ в рамках NVIDIA и квантового машинного обучения в CERN.

Прежде всего хотела бы посвятить эту работу своей семье, друзьям и всем тем, кто помог мне в ходе работы. Особая благодарность Аймену, которому я всегда благодарна.

Рамеш Сехар – генеральный директор и соучредитель компании Dapster.ai, которая разрабатывает доступных и легко развертываемых роботов, выполняющих самые трудные задачи на складах. Рамеш работал в таких компаниях, как Symbol, Motorola и Zebra, и специализируется на разработке продуктов на пересечении компьютерного видения, ИИ и робототехники. Имеет степень бакалавра в области электромашиностроения и магистра в области вычислительных систем. Рамеш основал Dapster.ai в 2020 году. Миссия Dapster состоит в том, чтобы разрабатывать роботов, которые оказывают положительное влияние на людей, выполняя опасные и вредные для здоровья задачи. Видение компании заключается в том, чтобы открывать доступ к более качественным рабочим местам, усиливать цепочки поставок и лучше справляться с вызовами, возникающими в результате изменения климата.

Уткарш Шривастава – профессионал в области ИИ/МО, тренер, ютубер и блогер. Любит решать и разрабатывать алгоритмы МО, обработки естественного языка и компьютерного зрения, чтобы решать сложные задачи. Начал свою карьеру в науке о данных в качестве блогера в своем блоге (datamahadev.com) и на канале YouTube (Datamahadev), после чего перешел на работу старшим тренером по науке о данных в институте в Гуджарате. Кроме того, он обучил и консультировал более 1000 работающих специалистов и студентов по ИИ/МО. Уткарш выполнил более 40 внештатных работ/проектов по тренировке и разработке в области науки о данных и аналитике, ИИ/МО, разработке на Python и SQL. Он родом из Лакхнау и в настоящее время поселился в Бангалоре, Индия, в качестве аналитика в Deloitte USI Consulting.

Хотел бы поблагодарить свою мать, миссис Рупам Шривастава, за ее постоянное руководство и поддержку на протяжении трудных периодов и борьбы. Спасибо также Верховному Пара-Брахману.

Мейсон МакГоу – старший специалист НИОКР в области машиностроения и компьютерного зрения в лаборатории Lowe's Innovation Labs. Страстно увлечен визуализацией и провел более десяти лет, решая задачи компью-

терного зрения в широком спектре промышленных и академических дисциплин, включая геологию, биоинформатику, разработку игр и розничную торговлю. Совсем недавно приступил к разведывательному анализу применения цифровых близнецов и 3D-сканирования применительно к розничным магазинам.

Хотел бы поблагодарить Энди Ликоса, Джозефа Канзано, Александра Аранго, Олега Александра, Эрин Кларк и мою семью за поддержку.

Предисловие

Благодаря этому практическому руководству по трехмерному глубокому обучению разработчики в области трехмерного компьютерного зрения смогут применить свои знания на практике. В данной книге представлен практический подход к реализации вычислительных решений в указанной области и связанных с ней методологий, которые помогут вам быстро начать работу и повысить продуктивность.

Оснащенные пошаговыми объяснениями важных понятий, практическими примерами и вопросами для самопроверки, вы начнете с обследования передовых методов трехмерного глубокого обучения.

Вы познакомитесь с базовой обработкой 3D-данных полигональной сетки и облака точек с помощью библиотеки PyTorch3D, такой как загрузка и сохранение файлов PLY и OBJ, проецирование 3D-точек на координаты камеры с использованием моделей перспективной камеры и ортографической камеры, отрисовка облаков точек и полигональных сеток на изображениях и т. д. Вы также научитесь реализовывать некоторые современные алгоритмы трехмерного глубокого обучения, такие как дифференцируемая отрисовка, NeRF, SynSin и Mesh R-CNN, поскольку благодаря библиотеке PyTorch3D программирование этих моделей глубокого обучения значительно упрощается.

К концу этой книги вы сможете реализовывать свои собственные модели трехмерного глубокого обучения.

Для кого эта книга предназначена

Эта книга предназначена для всех тех, кто начинает свою карьеру в области машинного обучения, а также практиков среднего уровня, исследователей данных, инженеров машинного обучения и инженеров глубокого обучения, которые стремятся хорошо разбираться в методах компьютерного зрения, используя 3D-данные.

О чем эта книга рассказывает

Глава 1 «Введение в обработку 3D-данных» будет посвящена основам 3D-данных, например способам хранения 3D-данных и базовым понятиям полигональной сетки и облаков точек, мировой системы координат и системы

координат поля зрения камеры. В ней также дается объяснение часто используемой системы координат NDC, способов конверсии разных систем координат, перспективной и ортографической камер и моделей камеры, которые следует использовать.

Глава 2 «Введение в трехмерное компьютерное зрение и геометрию» покажет базовые понятия компьютерной графики, такие как отрисовка и затенение. Вы познакомитесь с несколькими фундаментальными понятиями, которые потребуются в последующих главах этой книги, включая 3D-трансформации геометрии, тензоры PyTorch и оптимизацию.

Глава 3 «Подгонка деформируемых сеточных моделей к необработанным облакам точек» представит практический проект применения деформируемой 3D-модели с целью подгонки шумных 3D-наблюдений, используя все знания, которые вы получили в предыдущих главах. Вы познакомитесь с часто используемыми функциями стоимости, причинами важности этих функций и ситуациями, когда эти функции стоимости обычно используются. Наконец, мы обследуем наглядный пример выбора конкретных функций стоимости под конкретную задачу и настройки цикла оптимизации, чтобы получить желаемые результаты.

Глава 4 «Обнаружение и отслеживание позы объекта с помощью дифференцируемой отрисовки» расскажет о базовых концепциях дифференцируемой отрисовки. Она поможет разобраться в основных понятиях и выяснить, в каких ситуациях следует эти методы применять для решения своих собственных задач.

Глава 5 «Понятие дифференцируемой объемометрической отрисовки» представит практический проект с использованием дифференцируемой отрисовки для оценивания позиций камеры по одному изображению и известной трехмерной сеточной модели. Вы научитесь применять библиотеку PyTorch3D на практике, чтобы настраивать камеры, отрисовщики и затенители. Вы также получите практический опыт использования разных функций стоимости, чтобы получать результаты оптимизации.

Глава 6 «Обследование нейронных полей яркости излучения (NeRF)» предоставит практический проект с использованием дифференцируемого отрисовщика для оценивания трехмерных сеточных моделей по нескольким изображениям и текстурным моделям.

Глава 7 «Обследование контролируемых нейронных полей признаков» охватывает очень важный алгоритм синтеза ракурсов под названием Nerf. Вы познакомитесь с тем, что это вообще такое, как его использовать и где он проявляет свою ценность.

Глава 8 «Моделирование человеческого тела в 3D» посвящена обследованию подгонки 3D-тела человека с использованием алгоритма SMPL.

Глава 9 «Сквозной синтез ракурсов с помощью модели SynSin» посвящена передовой модели глубокого обучения, применяемой для синтеза других ракурсов изображения.

Глава 10 «Модель Mesh R-CNN» познакомит еще с одним передовым методом предсказания трехмерных воксельных моделей по одному входному изображению под названием Mesh R-CNN.

Что нужно, чтобы извлечь максимум пользы из этой книги

Описанное в книге программное/аппаратное обеспечение	Требования к операционной системе
Python 3.6+	Windows, MacOS или Linux

Если вы используете цифровую версию этой книги, то советуем набирать исходный код самостоятельно либо обращаться к исходному коду в репозитории книги на GitHub (ссылка на репозиторий доступна в следующем разделе). Это поможет избежать любых потенциальных ошибок, связанных с копированием и вставкой исходного кода.

Для справки, пожалуйста, ознакомьтесь с перечисленными ниже статьями.

Глава 6: https://arxiv.org/abs/2003.08934, https://github.com/yenchenlin/nerf-pytorch.

Глава 7: https://m-niemeyer.github.io/project-pages/giraffe/index.html, https://arxiv.org/abs/2011.12100.

Глава 8: https://smpl.is.tue.mpg.de/, https://smplify.is.tue.mpg.de/, https://smpl-x.is.tue.mpg.de/.

Глава 9: https://arxiv.org/pdf/1912.08804.pdf.

Глава 10: https://arxiv.org/abs/1703.06870, https://arxiv.org/abs/1906.02739.

Используемые обозначения

В этой книге используется ряд текстовых обозначений.

`Исходный код в тексте`: указывает слова исходного кода в тексте, имена таблиц базы данных, имена папок, имена файлов, расширения файлов, имена путей, фиктивные URL-адреса, вводимые пользователем данные и дескрипторы Twitter. Например, «Далее нужно обновить файл `./options/options.py`».

Блок исходного кода задается, как показано ниже:

```
elif opt.dataset == 'kitti':
    opt.min_z = 1.0
    opt.max_z = 50.0
    opt.train_data_path = (
        './DATA/dataset_kitti/'
    )
    from data.kitti import KITTIDataLoader
    return KITTIDataLoader
```

Когда мы хотим привлечь ваше внимание к определенной части блока исходного кода, соответствующие строки или элементы выделяются жирным шрифтом:

```
wget https://dl.fbaipublicfiles.com/synsin/checkpoints/realestate/synsin.pth
```

Любые данные на входе или на выходе из команды командой оболочки записываются, как показано ниже:

```
bash ./download_models.sh
```

Жирный шрифт: выделяет новый термин, важное слово или слова, которые вы видите на экране. Например, слова в меню или диалоговых окнах пишутся в тексте следующим образом: «Модуль детализации (**g**) получает входные данные от нейронного отрисовщика облака точек и затем выводит окончательное реконструированное изображение».

Подсказки и важные замечания выглядят так.

Часть I

ОСНОВЫ ОБРАБОТКИ 3D-ДАННЫХ

Первая часть книги посвящена определению самых базовых понятий обработки данных и изображений, поскольку указанные понятия лягут в основу последующего изложения. Данная часть делает книгу самодостаточной, вследствие чего читателям не придется читать какие-либо другие книги, чтобы приступить к изучению библиотеки PyTorch3D.

Эта часть содержит следующие главы:

- глава 1 «Введение в обработку 3D-данных»;
- глава 2 «Введение в трехмерное компьютерное зрение и геометрию».

Глава 1

Введение в обработку 3D-данных

В этой главе мы обсудим несколько базовых понятий, весьма существенных для трехмерного глубокого обучения, которые будут часто использоваться в последующих главах. Вы начнете со знакомства с наиболее часто используемыми форматами 3D-данных, а также многими способами манипулирования ими и конвертации их в разные форматы. Мы начнем с настройки среды разработки и установкой всех необходимых программных пакетов, включая Anaconda, Python, PyTorch и PyTorch3D. Затем мы поговорим о наиболее часто используемых способах представления 3D-данных – например, облаках точек, полигональных сетках и вокселах. Затем мы перейдем к форматам файлов 3D-данных, таким как файлы PLY и OBJ. Затем обсудим системы 3D-координат. Наконец, мы обсудим модели камеры, которые в основном связаны со способом отображения 3D-данных в 2D-изображения[1].

После прочтения этой главы вы сможете легко отлаживать алгоритмы трехмерного глубокого обучения, проводя инспекцию файлов выходных данных. Благодаря четкому пониманию систем координат и моделей камеры вы будете готовы опираться на эти знания и узнать о более продвинутых темах трехмерного глубокого обучения.

В данной главе будут охвачены следующие ниже главные темы:

- настройка среды разработки и установка дистрибутива Anaconda, библиотек PyTorch и PyTorch3D,
- представление 3D-данных,
- форматы 3D-данных – файлы PLY и OBJ,
- системы 3D-координат и конверсия между ними,
- модели камеры – перспективная и ортографическая камеры.

[1] Син. соотнесение 3D-данных с 2D-изображениями. – *Прим. перев.*

Технические требования

Для выполнения примеров исходного кода этой книги в идеале понадобится компьютер с графическим процессором. Тем не менее для выполнения фрагментов исходного кода вполне будет достаточно только центрального процессора(ов).

Рекомендуемая компьютерная конфигурация включает следующее:

- GPU, такой как серия GTX или серия RTX с не менее 8 Гб памяти,
- Python 3,
- библиотеки PyTorch и PyTorch3D.

Фрагменты исходного кода к этой главе находятся по адресу https://github.com/packtpublishing/3d-deep-learning-with-python.

Настройка среды разработки

Сначала давайте создадим среду разработки для всех прилагаемых к этой книге примеров программирования. Для всех примеров исходного кода Python в этой книге рекомендуется использовать машину Linux.

1. Сначала мы настроим широко используемый дистрибутив Python под названием Anaconda, который идет в комплекте с мощной реализацией CPYTHON. Одним из преимуществ использования дистрибутива Anaconda является его система управления пакетами, позволяющая пользователям легко создавать виртуальные среды. Индивидуальная редакция дистрибутива является бесплатной для одиночных практиков, студентов и исследователей. В целях установки дистрибутива мы рекомендуем посетить его веб-сайт anaconda.com, на котором можно получить подробные инструкции. Самым простым способом установки дистрибутива Anaconda, как правило, является выполнение скрипта, который нужно скачать с веб-сайта дистрибутива. После настройки дистрибутива выполните следующую ниже команду, чтобы создать виртуальную среду Python 3.7:

   ```
   $ conda create -n python3d python=3.7
   ```

   Эта команда создаст виртуальную среду Python версии 3.7. Для того чтобы использовать эту виртуальную среду, ее нужно сначала активировать.
2. Активируйте только что созданную виртуальную среду следующей ниже командой:

   ```
   $ source activate python3d
   ```

3. Установите библиотеку PyTorch. Подробные инструкции по установке PyTorch находятся на ее веб-странице по адресу www.pytorch.org/get-

started/locally/. Например, на своем рабочем столе Ubuntu с CUDA 11.1 я установлю PyTorch 1.9.1 следующим образом:

```
$ conda install pytorch torchvision torchaudio
cudatoolkit-11.1 -c pytorch -c nvidia
```

4. Установите библиотеку PyTorch3D. Это библиотека Python с открытым исходным кодом для трехмерного компьютерного зрения, недавно выпущенная исследовательской группой Facebook AI Research. Библиотека PyTorch3D предоставляет много функций-утилит, позволяющих с легкостью манипулировать 3D-данными. Будучи спроектированной специально для глубокого обучения, она позволяет обрабатывать почти все 3D-данные мини-пакетами, такими как камеры, облака точек и полигональные сетки. Еще одной ключевой особенностью библиотеки PyTorch3D является реализация очень важной техники трехмерного глубокого обучения, именуемой *дифференцируемой отрисовкой*[1]. Тем не менее самым большим преимуществом данной библиотеки трехмерного глубокого обучения является ее тесная связь с PyTorch.
   Библиотеке PyTorch3D могут понадобиться некоторые зависимости, и подробные инструкции по установке этих зависимостей находятся на домашней странице PyTorch3D на Github по адресу github.com/facebookresearch/pytorch3d. После установки всех зависимостей, если следовать инструкциям веб-сайта, установка PyTorch3D легко выполняется следующей ниже командой:

```
$ conda install pytorch3d -c pytorch3d
```

Теперь, когда мы создали среду разработки, давайте продолжим и займемся изучением представления данных.

Представление 3D-данных

В этом разделе вы познакомитесь с наиболее часто используемыми представлениями 3D-данных. Выбор представления данных является особенно важным конструктивным решением для многих систем трехмерного глубокого обучения. Например, облака точек не имеют решетчатых структур, поэтому свертки обычно невозможно использовать для них напрямую. Представления в виде вокселов имеют решетчатые структуры, однако они, как правило, потребляют большой объем компьютерной памяти. Мы обсудим плюсы и минусы этих 3D-представлений подробнее в этом разделе. Пред-

[1] Дифференцируемая отрисовка (differentiable rendering) – это относительно новая и захватывающая область исследований в компьютерном зрении, преодолевающая разрыв между 2D и 3D, позволяющая связывать пикселы 2D-изображения с 3D-свойствами сцены. – *Прим. перев.*

ставлениями 3D-данных, получившими наиболее широкое применение на практике, обычно являются облака точек, полигональные сетки и вокселы.

Представление в виде облака точек

Облако 3D-точек – это очень простое представление 3D-объектов, в котором каждое облако точек – это просто множество 3D-точек, и каждая 3D-точка представлена одним трехмерным кортежем (*x*, *y* и *z*). Сырые мерные данные многих камер глубины обычно представляют собой трехмерные облака точек.

С точки зрения глубокого обучения облака 3D-точек являются одним из неупорядоченных и нерегулярных типов данных. В отличие от регулярных изображений, в которых по каждому отдельному пикселу можно определить соседствующие ему пикселы, в облаке точек нет четких и регулярных определений соседних точек по каждой точке – т. е. применить свертки к облакам точек обычно невозможно. И поэтому для обработки облаков точек необходимо использовать специальные типы моделей глубокого обучения, такие как PointNet: https://arxiv.org/abs/1612.00593.

Еще одной проблемой облаков точек в качестве тренировочных данных для трехмерного глубокого обучения является проблема разнородности данных – т. е. по каждому тренировочному набору данных разные облака точек могут содержать разное число 3D-точек. Один из подходов к решению проблемы разнородности данных заключается в вынужденном назначении всем облакам точек одинакового числа точек. Однако это не всегда возможно – например, число возвращаемых камерами глубины точек может отличаться от кадра к кадру.

При тренировке моделей глубокого обучения разнородные данные могут создавать некоторые трудности для мини-пакетного градиентного спуска. В большинстве систем глубокого обучения подразумевается, что каждый мини-пакет содержит тренировочные примеры одинакового размера и мерности. Предпочитаются именно такие однородные данные, поскольку их можно обрабатывать на современном оборудовании для параллельной обработки наиболее эффективным образом, таком как графические процессоры. Эффективная обработка разнородных мини-пакетов требует дополнительной работы. К счастью, PyTorch3D предоставляет целый ряд способов эффективной обработки разнородных мини-пакетов, которые очень важны для трехмерного глубокого обучения.

Представление в виде полигональной сетки

Полигональные сетки, или меши, – это еще одно широко используемое представление 3D-данных. Как и точки в облаках точек, каждая полигональная сетка содержит множество 3D-точек, именуемых вершинами. Кроме того, каждая полигональная сетка содержит множество многоугольников, именуемых гранями, которые определены на вершинах.

В большинстве основанных на данных приложений полигональные сетки являются результатом постобработки сырых мерных данных камер глубины. Нередко они создаются вручную в процессе конструирования 3D-ресурсов. По сравнению с облаками точек полигональные сетки содержат дополнительную геометрическую информацию, кодируют топологию и имеют информацию о нормалях к поверхности. Эта дополнительная информация становится особенно полезной в тренировке обучающихся моделей. Например, графовые сверточные нейронные сети обычно трактуют полигональные сетки как графы и определяют сверточные операции, используя информацию о соседстве вершин.

Подобно облакам точек, полигональные сетки также имеют схожие проблемы разнородности данных. И снова PyTorch3D предоставляет эффективные способы оперирования разнородными мини-пакетами данных полигональной сетки, что делает трехмерное глубокое обучение весьма эффективным.

Представление в виде воксела

Еще одним важным представлением 3D-данных является представление в виде воксела. Воксел – это аналог пиксела в трехмерном компьютерном зрении. Пиксел определяется путем деления прямоугольника в 2D на меньшие прямоугольники, и каждый малый прямоугольник – это один пиксел. По аналогии с этим воксел определяется путем деления трехмерного куба на кубы меньшего размера, и каждый такой куб называется одним вокселом. Соответствующие процессы показаны на следующем ниже рисунке:

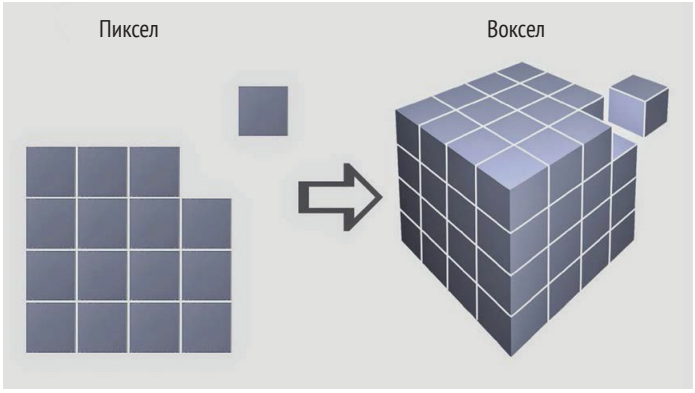

Рис. 1.1 ❖ Представление в виде воксела является трехмерным аналогом двумерного представления в виде пиксела, где кубическое пространство делится на малообъемные элементы

В представлениях в виде воксела для представления 3D-поверхностей обычно используются **функции усеченных расстояний со знаком** (**TSDF**)[1].

[1] От англ. *Truncated Signed Distance Function. – Прим. перев.*

Функция расстояния со знаком (**SDF**)[1] может быть определена на каждом вокселе в качестве расстояния (со знаком) между центром воксела до ближайшей точки на поверхности. Положительный знак в SDF указывает на то, что центр воксела находится вне объекта. Единственное различие между TSDF и SDF заключается в том, что значения TSDF усекаются, в результате чего значения TSDF всегда варьируются в интервале от –1 до +1.

В отличие от представлений в виде облаков точек и полигональных сеток представление в виде вокселов упорядочено и является регулярным. Это свойство похоже на пикселы в изображениях и позволяет использовать сверточные фильтры в моделях глубокого обучения. Одним из потенциальных недостатков представления в виде вокселов является то, что для них обычно требуется больше компьютерной памяти, но указанный недостаток можно уменьшить за счет таких методов, как хеширование. Тем не менее представление в виде вокселов является важным представлением 3D-данных.

Существуют представления 3D-данных, отличные от упомянутых выше. Например, в многоракурсных представлениях используется несколько изображений, взятых с разных точек зрения, чтобы представлять трехмерную сцену. В представлениях RGB-D используется дополнительный канал глубины, чтобы представлять 3D-сцену. Однако в этой книге мы не будем погружаться в эти 3D-представления слишком глубоко. Теперь, когда вы познакомились с основами представлений 3D-данных, самое время заняться несколькими форматами файлов, часто используемыми для облаков точек и полигональных сеток.

Формат файла 3D-данных – файлы PLY

Формат файла PLY[2] был разработан в середине 1990-х годов группой исследователей из Стэнфордского университета. С тех пор он превратился в один из наиболее широко используемых форматов файлов 3D-данных. Формат файла PLY имеет как ASCII-версию, так и двоичную версию. Двоичная версия предпочтительнее в тех случаях, когда необходимо минимизировать размеры файлов и обеспечить эффективность обработки. ASCII-версию легко отлаживать. Здесь мы обсудим базовый формат PLY-файлов и технику использования как пакета Open3D, так и библиотеки PyTorch3D для загрузки и визуализации 3D-данных из PLY-файлов.

В этом разделе мы собираемся обсудить два наиболее часто используемых формата файлов данных, чтобы представлять облака точек и полигональные сетки, формат PLY-файла и формат OBJ-файла. Мы обсудим сами форматы и способы загрузки и сохранения этих форматов файлов с помощью библиотеки PyTorch3D. Библиотека PyTorch3D предоставляет отличные функции-утилиты, поэтому с помощью этих утилит загрузка из этих форматов и сохранение в них проста и эффективна.

---

[1] От англ. *Signed Distance Function. – Прим. перев.*

[2] От англ. *Polygon File Format. – Прим. перев.*