МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра компьютерных технологий и систем

ПОСТРОЕНИЕ КАРТЫ ГЛУБИНЫ НА ОСНОВЕ ВИДЕОПОТОКА С МОНОКАМЕРЫ

Курсовой проект

Савицкой Елизаветы Дмитриевны
студентки 4 курса,
специальность «информатика»

Научный руководитель:
профессор кафедры КТС,
Недзьведь Александр Михайлович

Минск, 2021

РЕФЕРАТ

Курсовой проект, 32 стр., 10 рисунков, 15 источников.

ПОСТРОЕНИЕ КАРТЫ ГЛУБИНЫ НА ОСНОВЕ ВИДЕОПОТОКА С МОНОКАМЕРЫ

Ключевые слова: анализ изображений, карта глубины, монокулярные методы оценки глубины, свёрточные нейронные сети.

Объект исследования: алгоритмы и методы построения карты глубины с помощью свёрточных нейронных сетей.

Цель работы: разработка алгоритма построения карты глубины по видеопотоку с монокамеры.

Методы исследования: системный подход, изучение соответствующей литературы и электронных источников, проведение экспериментов.

Результаты: теоретическая база, сравнительные эксперименты, были разработаны алгоритмы построения карты глубины с помощью свёрточной нейронной сети.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Анализ и обработка фото / видео изображений, в том числе микроскопических, могут использоваться для реализации широкого спектра задач, в том числе, таких как реализация технологий компьютерного зрения, существенное улучшение качества обычной фото /видеосъемки по сравнению с техническими возможностями изначально имеющегося аппаратного оборудования, использоваться для исследований в области медицины и исследований спектра биологических процессов, включая клеточные.

Научная новизна исследования заключается в разработке алгоритмов, позволяющих построить карту глубины, используя всего одну камеру. На основе предложенных в работе алгоритмов разработана методика анализа видео последовательности, которая базируется на использовании понятия машинного обучения.

Тем не менее, существующие в настоящее время технологии анализа фото / видеопоследовательностей в большей степени ориентированы на перемещение отдельных объектов, а не их подвижных систем.

Цель курсового проекта – разработать алгоритм построения карты глубины по видеопотоку с монокамеры.

Для достижения цели были поставлены и /или должны быть решены следующие задачи:

- выполнен анализ существующих монокулярных методов и алгоритмов построения карты глубины;
- выполнен анализ существующих наборов данных, на которых будет обучаться нейронная сеть;
- автоматизировать процесс построения карты глубины по данным, полученным с помощью свёрточной нейронной сети;
- применить разработанные алгоритмы на практике.

Предмет исследования – алгоритмы и методы построения карты глубины на основе глубокого обучения.

В настоящем курсовом проекте предлагается методика анализа фотоизображения путем построения карты глубины по видеопотоку с монокамеры с помощью нейронной сети. Предлагается методика анализа последовательности изображений с учетом подвижности ее внутренней структуры, изменения формы и размеров, которая сочетает описание системы как целого с движением составляющих ее отдельных компонентов.

ГЛАВА 1. Современные методы оценки глубины по видеопоследовательности

1.1.1 Определение карты глубины

Карта глубины (depth map) — это изображение, на котором для каждого пикселя, вместо цвета, храниться его расстояние до камеры.

Карты глубины имеют ряд применений, в том числе:

- Имитация эффекта однородно плотных полупрозрачных материалов внутри сцены, таких как туман, дым или большие объемы воды.
- Имитация малой глубины резкости, когда некоторые части сцены кажутся не в фокусе. Карты глубины можно использовать для выборочного размытия изображения в различной степени. Малая глубина резкости может быть характеристикой макросъемки, поэтому данная техника может быть частью процесса подделки миниатюр.
- Z-буферизация и z-отбраковка - методы, которые можно использовать для повышения эффективности рендеринга 3D-сцен. Их можно использовать для идентификации объектов, скрытых от просмотра и, следовательно, игнорируемых для некоторых целей визуализации. Это особенно важно в приложениях реального времени, таких как компьютерные игры, где должна быть доступна быстрая последовательность завершенных визуализаций, чтобы они отображались с регулярной и фиксированной скоростью.
- Отображение теней - часть одного процесса, используемого для создания теней, отбрасываемых освещением в компьютерной 3D-графике. В этом случае карты глубины рассчитываются с точки зрения источников света, а не наблюдателя.
- Предоставлять информацию о расстоянии, необходимую для создания автостереограмм и в других связанных приложениях, предназначенных для создания иллюзии трехмерного просмотра посредством стереоскопии.
- В компьютерном зрении карты глубины одно- или многовидовых изображений или другие типы изображений используются для моделирования трехмерных форм или их реконструкции. Карты глубины можно создавать с помощью 3D-сканеров или восстанавливать из нескольких изображений
- Зная информацию о глубине можно генерировать 3D модели ландшафта и других природных объектов для использования в различных приложениях, таких как виртуальная реальность, симуляция полета,

робототехника. В частности, знание об удаленности точек изображения от реального прообраза позволяет делать захват опорных точек движущегося объекта для получения трехмерных координат, что важно в индустрии спецэффектов, кино и телевидении.

- Оценка глубины является неотъемлемой частью многих проблем в робототехнике, включая картирование, локализацию и предотвращение препятствий для наземных и воздушных транспортных средств, а также в компьютерном зрении, включая дополненную и виртуальную реальность.

ГЛАВА 2. СОВРЕМЕННЫЕ МОНОКУЛЯРНЫЕ МЕТОДЫ ОЦЕНКИ ГЛУБИНЫ

2.1 Оценка глубины на основе монокулярных изображений

Оценка глубины на основе монокулярных изображений является важной темой исследований в области фотометрического компьютерного зрения [1, 2, 3]. Его целью является создание пиксельных карт глубины с изображением с определённой точки обзора. Такая информация о глубине помогает лучше понять трехмерные сцены, а также облегчает выполнение многих задач компьютерного зрения, таких как локализация в помещении [3], оценка высоты, одновременная локализация и картирование (SLAM), визуальная одометрия, классификация и т.д.

На данный момент известны активные и пассивные методы восстановления информации о глубине реальной сцены. Активные методы используют ультразвуковые преобразователи или лазерное освещение рабочего пространства, дающие на выходе быструю и точную информацию о глубине. Однако у этих методов есть ограничения по отношению к диапазону измерений и стоимости аппаратных компонентов. Пассивные методы, основанные на компьютерном зрении, обычно реализуются более простыми и недорогими сенсорами, определяющими расстояние. Например, карта глубины может быть получена с помощью специальной стереокамеры, по стереопаре изображений, а также с применением метода зеркального разделения изображений. Несмотря на преимущества таких подходов, их реализация также требует наличия специфического аппаратного обеспечения, что может существенно сужать область его применения.

Обычно информацию о глубине получают с помощью коммерческих датчиков глубины, таких как различные устройства LiDAR и Kinects. Однако, помимо высокой стоимости и потребности в оперативных навыках, они также сталкиваются с недостатками, связанными с низким разрешением и коротким расстоянием восприятия, что ограничивает возможности широкого применения. Получение карты глубины из монокулярных изображений привлекает все больший интерес благодаря широкой доступности изображений RGB. Монокулярные камеры привлекательны, как они уже есть во многих ежедневных системах, таких как телефоны, dashcameras и камеры наблюдения.

Анализ методов построения карты глубины [5, 6, 7, 8], не требующих использования специального оборудования показывает, что ряд наиболее инновационных решений такого рода связан с монокулярными решениями и использованием методов машинного обучения. Эти методы направлены на оценку расстояний между объектами сцены и камерой с одной смотровой зоны (зоны захвата изображения). В настоящее время применение методов машинного обучения является решением многих задач компьютерного зрения, включая оценку глубины. Оценка монокулярной глубины широко используется в робототехнике и виртуальной реальности, которая требует развертывания на устройствах низкого уровня.

Методы машинного обучения, используемые для оценки глубины можно разбить на две группы: непараметрические методы и методы обучения. Непараметрические методы используют обширные доступные наборы данных RGBD и выводят целевую глубину на основе нескольких эталонных изображений глубины. Методы, основанные на обучении, непосредственно оценивают глубину по различным характеристикам и обычно включают в себя модели на основе графов.

Последние разработки в области оценки глубины сосредоточены на использовании **свёрточных нейронных сетей** (CNN) для выполнения 2D-3D реконструкции. Хотя эффективность этих методов неуклонно растет, по-прежнему существуют серьезные проблемы как с качеством, так и с разрешением этих расчетных карт глубины. Последние приложения в дополненной реальности, синтетические эффекты глубины поля и другие эффекты изображения требуют быстрого вычисления трехмерных реконструкций с высоким разрешением подлежащий применению. Для таких применений крайне важно точно восстановить разрывы в картах глубины и избежать больших возмущений, которые часто присутствуют в оценках глубины, вычисленных с использованием текущих CNN. Чтобы достичь этого, можно использовать предварительно обученные сети, которые изначально предназначены для классификации изображений в качестве кодировщика *(encoder)*. Ключевым преимуществом такого подхода, основанного на переходе к обучению, является то, что он позволяет создать более модульную архитектуру, в которой будущие достижения в одной области легко перенести в проблему оценки глубины. То есть, предлагается построить простую сетевую архитектуру, основанную на трансфертном обучении, которая производит углубленные оценки более высокой точности и качества. Полученные в результате карты глубин будут отражать границы объектов более точно, чем те, которые получены с помощью методов с меньшим количеством параметров и учебных итераций. Также необходимо определить соответствующую функция

потерь, стратегии обучения и проведены различные эксперименты на нескольких наборах данных для оценки эффективности и качества сети оценки глубины.

Монокулярная оценка глубины рассматривается многими методами CNN [5], где задача формулируется как регрессия карты глубины из одного изображения RGB. Хотя эффективность этих методов неуклонно растет, общие проблемы, связанные как с качеством, так и с разрешением расчетных карт глубины, оставляют много возможностей для совершенствования. Основная цель в этой работе - создание качественной карт глубины с точными границами с использованием стандартных нейронных сетей.

Сети кодировщиков-декодеров внесли значительный вклад в решение многих проблем, связанных со зрением, таких как сегментация изображения, оптическая оценка потока и восстановление изображения. В последние годы использование таких архитектур показало реальный успех как в контролируемой, так и в неконтролируемой установке задачи оценки глубины. Такие методы обычно используют одну или несколько сетей кодировщиков-декодеров как часть их более крупной сети

2.2 Наборы данных для машинного обучения

Для обучения или оценки методов определения глубины, основанных на глубоком обучении, необходимы пары изображений RGBD. Карты глубины могут быть собраны различными способами, включая камеры RGBD, лазерные сканеры или многовидовые стереосистемы (MVS). Некоторые наборы данных даже используют синтетическую карту глубины, отображаемую из виртуальных сцен.

Многие из самых впечатляющих успехов в глубоком обучении компьютерному зрению были связаны с задачами распознавания и опирались на большие, разнообразные, помеченные вручную наборы данных, такие как ImageNet, Places и COCO. В отличие от этого, наборы данных RGBD, которые соединяют изображения и глубину, не могут быть созданы с помощью краудсорсинговой аннотации, и вместо этого полагаются на 3D-диапазоны сенсоров, которые шумные, редкие, дорогие или все вышесказанное. Некоторые популярные сенсоры диапазона ограничены внутренними сценами из-за пределов дальности и технологии зондирования. Другие типы датчиков обычно размещаются только на открытом воздухе. В результате доступные наборы данных RGBD в основном включают только один из этих типов сцен.

Внутренние наборы данных RGBD обычно собираются с помощью структурированных световых камер, которые обеспечивают плотные, но шумные карты глубин до приблизительно 10 м, ограничивая их применение небольшими внутренними средами (например, домашними и офисными). Внешние наборы данных, как правило, собираются с учетом конкретного приложения (например, беспилотных транспортных средств) и, как правило, приобретаются с помощью индивидуализированных сенсорных массивов, состоящих из монокулярных камер и сканеров LiDAR. Типичные сканеры LiDAR имеют высокую частоту выборки, но относительно низкое пространственное разрешение. Следовательно, характеристики имеющихся карт глубин внутри и снаружи весьма различны, и сети, обученные на основе одного вида данных, как правило, плохо проявляют себя на другом.

2.2.1 Набор данных NYU Depth V2

NYU Depth V2 [12] — это набор данных для сегментации объектов на изображениях интерьера. Данный набор данных состоит из видеопоследовательностей из разных сцен интерьера, которые были записаны в RGB и с помощью камер глубины от Microsoft Kinect, которая обеспечивает редкую и шумную глубину. Эти результаты, как правило, окрашиваются и сглаживаются до того, как они используются для задач оценки монокулярной глубины (рисунок 1).

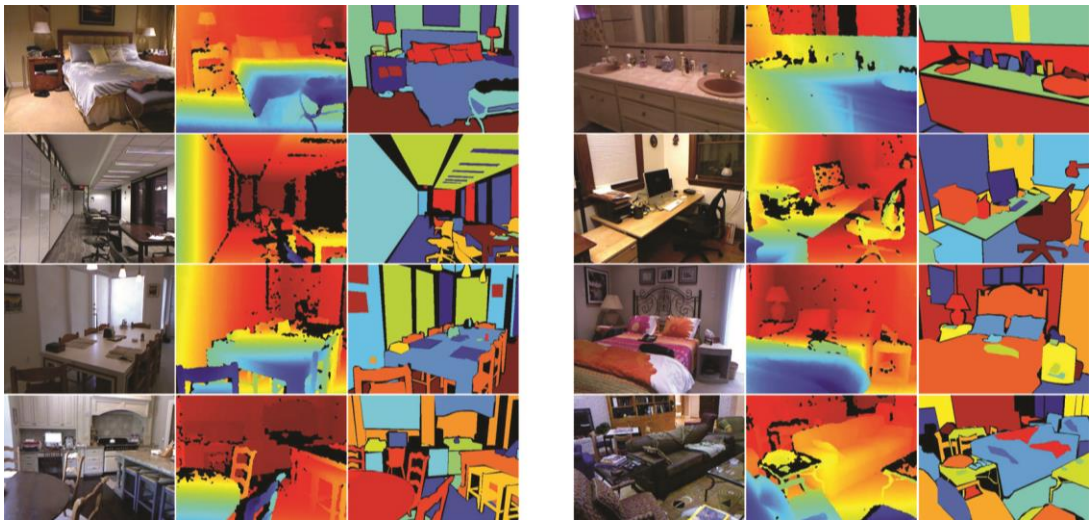

Рисунок 1. Выход RGB камеры (слева), предобработанная глубина (по центру) и размеченные классы объектов на изображении (справа)

В результате, в то время как набор данных включает достаточное количество образцов для обучения современных конвейеров машинного обучения, глубина *«ground-truth»* не обязательно соответствует истинной глубине сцены. Набор данных содержит 464 различные сцены из разных зданий, 249 сцен используются для тренировки и 215 сцен для тестирования, эти сцены включают в себя 1449 пар изображений, из которых 795 изображений используются для тренировки, а остальные 654 изображения для тестирования. Разрешение изображений составляет 640×480. Набор данных NYUv2 широко используется для оценки монокулярной глубины во внутренних средах.

2.2.2 Набор данных KITTI

Набор данных **KITTI** [10] — это набор данных, собранных на открытом воздухе с помощью мобильных картографических средств. Изображения RGB записываются стерео-калиброванной и корректируемой камерой. В настоящее время это крупнейший в мире набор данных для оценки алгоритмов компьютерного зрения в сценариях автономного вождения. Он используется для оценки эффективности технологий компьютерного зрения, таких как обнаружение целей (автотранспортных средств, немеханических транспортных средств, пешеходов и т. д.), отслеживание целей и сегментация дорог в транспортной среде. KITTI содержит реальные данные изображения, собранные в городских, сельских и автомобильных сценах. Размер изображения составляет около 1224×368. Они обычно обрезаются, чтобы уменьшить области неба. Для выполнения задач оценки глубины набор данных KITTI содержит более 93000 разреженных карт глубины и оригинальных изображений RGB. Карты плотной глубины производятся путем накопления измерений LiDAR по всей последовательности. Весь набор данных официально разделен на 86000 для обучения и 76000 для тестирования.

2.2.3 Набор данных Make3D

Make3D [11] является гибридным набором данных, содержащей 1000 наружных сцен и 50 внутренних сцен. Весь набор данных состоит из 400 тренировочных изображений и 134 тестовых изображений с разрешением

2272×1704 пикселей. Соответствующие наземные карты глубины собираются специальным 3D-сканером с разрешением всего 305×55 пикселей.

2.2.4 Набор данных DIODE

Датасет **DIODE** (Dense Indoor/Outdoor Depth) [13] — это набор данных, который содержит тысячи разнообразных цветовых изображений с высоким разрешением 1024×768 с точными, плотными, дальними измерениями глубины. DIODE является первым публичным набором данных, включающим RGBD изображения внутренних и наружных сцен, полученные с помощью одного набора датчиков. Это контрастирует с другими наборами данных, которые включают только один тип сцены и используют различные датчики, что затрудняет обобщение по доменам.

Набор данных DIODE — это попытка устранить ограничения выше упомянутых наборов данных RGBD. DIODE предоставляет массивный набор данных различных внутренних и наружных сцен, собранных с помощью лазерного сканера (FARO Focus S350). На рисунке 2 представлены несколько показательных примеров ДИОД, иллюстрирующих разнообразие сцен и качество трехмерных измерений.

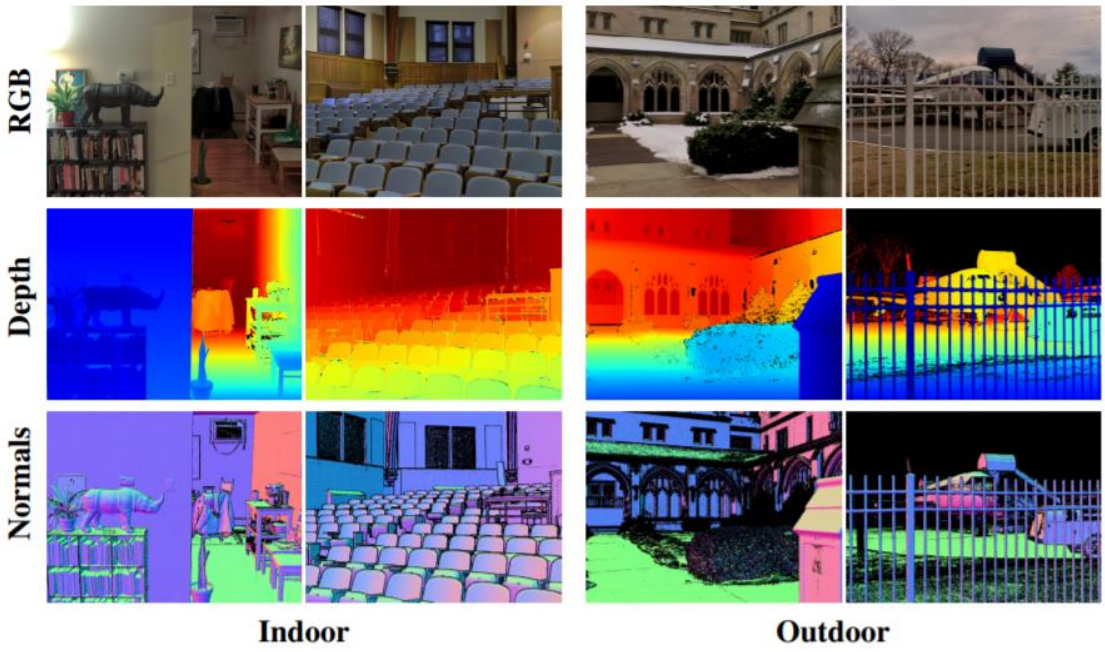

Рисунок 2. Образцы из данных из набора DIODE

Это качество позволяет не только составлять карты глубин с беспрецедентной плотностью и разрешением, но и получать нормали поверхности с уровнем точности, невозможной с другими наборами данных. Наиболее важной особенностью DIODE является то, что это первый набор данных, который охватывает как внутренние, так и внешние сцены в одной и той же системе считывания и изображения.

Набор данных DIODE дополняет NYUv2, предоставляя карты с очень высоким разрешением и низкой глубиной шума как внутренних, так и наружных сцен.

Другой внутренний набор данных, который опирается на SfM, это SUN3D, который обеспечивает приблизительную глубину без масштаба. Между тем, недавние наборы данных Matterport3D и ScanNet предлагают большое количество изображений плотной глубины внутренних сцен. Наборы данных были выведены из нескольких просмотров с использованием конвейера SLAM. В результате, карты глубин намного шумнее и с меньшим разрешением, чем DIODE, и предназначены для семантических задач, таких как 3D-сегментация, а не для точной 3D-реконструкции или оценки глубины.

Таким образом, по сравнению с другими наборами данных RGBD, DIODE предлагает более широкое разнообразие сцен; более высокое разрешение изображения и глубины карты; более высокую плотность и точность измерений глубины; и, самое главное, способность рассуждать о глубине восприятия как в помещении, так и за его пределами в подлинно единой системе.

Make3D предоставляет информацию RGB и глубину для наружных сцен, которые по своей природе похожи на DIODE, как и DIODE, он содержит разнообразные наружные сцены, которые не ограничиваются видом на улицу, но карты глубины в нём низкого разрешения. В то время как набор данных DIODE был собран с помощью сканера с гораздо более высоким разрешением и включающий множество более разнообразных сцен. Это качество позволяет не только составлять карты глубин с беспрецедентной плотностью и разрешением, но и получать нормали поверхности с уровнем точности, невозможной с другими наборами данных. Наиболее важной особенностью DIODE является то, что это первый набор данных, который охватывает как внутренние, так и внешние сцены в одной и той же системе считывания и изображения, структура набора данных представлена на рисунке 4.

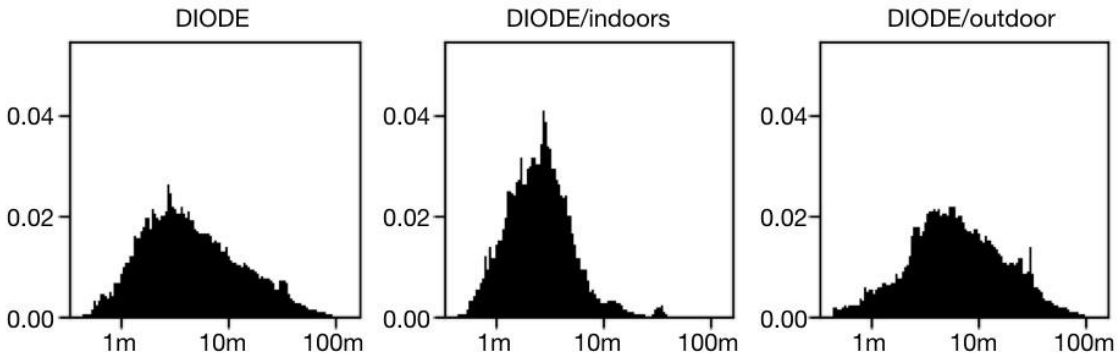

Рисунок 3. Распределение измеренных значений глубины для DIODE

Данные ДИОД организованы иерархически. Детализированная структура показана следующим образом:

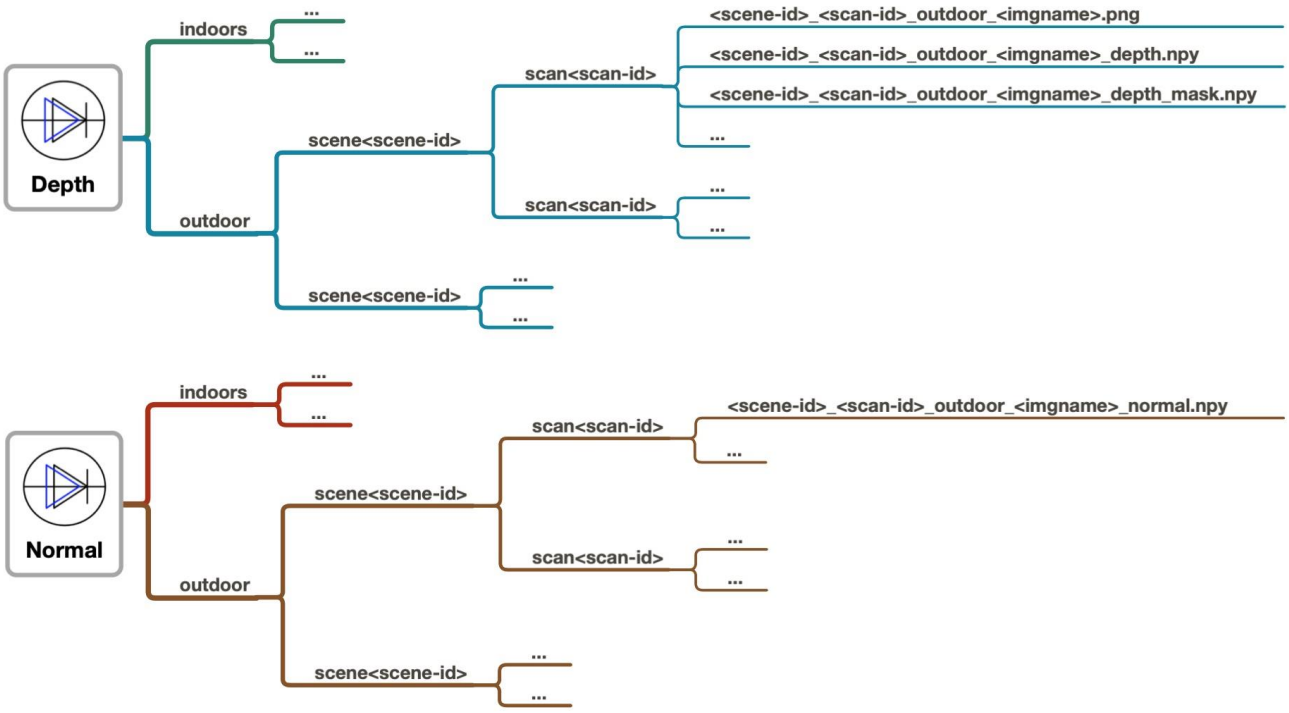

Рисунок 4. Структура набора данных DIODE

2.3 Оценка глубины с помощью свёрточных нейронных сетей (CNN)

2.3.1 Свёрточная нейронная сеть

Свёрточная нейронная сеть (англ. convolutional neural network, CNN) — специальная архитектура искусственных нейронных сетей, предложенная Яном Лекуном в 1988 году и нацеленная на эффективное распознавание образов, входит в состав технологий глубокого обучения (англ. *deep learning*). Идея свёрточных нейронных сетей заключается в чередовании свёрточных слоёв (англ. *convolution layers*) и субдискретизирующих слоёв (англ. *subsampling layers* или англ. *pooling layers*, слоёв подвыборки). Структура сети — однонаправленная (без обратных связей), многослойная. Для обучения используются стандартные методы, чаще всего метод обратного распространения ошибки. Функция активации нейронов (передаточная функция) — любая, по выбору исследователя.

Работа свёрточной нейронной сети обычно интерпретируется как переход от конкретных особенностей изображения к более абстрактным деталям, и далее к ещё более абстрактным деталям вплоть до выделения понятий высокого уровня. При этом сеть самонастраивается и вырабатывает сама необходимую иерархию абстрактных признаков (последовательности карт признаков), фильтруя маловажные детали и выделяя существенное.

Подобная интерпретация носит скорее метафорический или иллюстративный характер. Фактически «признаки», вырабатываемые сложной сетью, малопонятны и трудны для интерпретации настолько, что на практике суть этих признаков даже не пытаются понять, тем более «подправлять», а вместо этого для улучшения результатов распознавания меняют структуру и архитектуру сети. Так, игнорирование системой каких-то существенных явлений может говорить о том, что-либо не хватает данных для обучения, либо структура сети обладает недостатками, и система не может выработать эффективных признаков для данных явлений.

Архитектура CNN аналогична структуре связей нейронов в мозгу человека, учёные черпали вдохновение в организации зрительной коры головного мозга. Отдельные нейроны реагируют на стимулы только в некоторой области поля зрения, также известного как перцептивное поле. Множество перцептивных полей перекрывается, полностью покрывая поле зрения CNN.

В обычном перцептроне, который представляет собой полносвязную нейронную сеть, каждый нейрон связан со всеми нейронами предыдущего слоя, причём каждая связь имеет свой персональный весовой коэффициент. В свёрточной нейронной сети в операции свёртки используется лишь ограниченная матрица весов небольшого размера, которую «двигают» по всему обрабатываемому слою (в самом начале — непосредственно по входному изображению), формируя после каждого сдвига сигнал активации для нейрона следующего слоя с аналогичной позицией. То есть для различных нейронов выходного слоя используются одна и та же матрица весов, которую также называют ядром свёртки. Её интерпретируют как графическое кодирование какого-либо признака, например, наличие наклонной линии под определённым углом. Тогда следующий слой, получившийся в результате операции свёртки такой матрицей весов, показывает наличие данного признака в обрабатываемом слое и её координаты, формируя так называемую карту признаков (англ. *feature map*). Естественно, в свёрточной нейронной сети набор весов не один, а целая гамма, кодирующая элементы изображения (например, линии и дуги под разными углами). При этом такие ядра свёртки не закладываются исследователем заранее, а формируются самостоятельно путём обучения сети классическим методом обратного распространения ошибки. Проход каждым набором весов формирует свой собственный экземпляр карты признаков, делая нейронную сеть многоканальной (много независимых карт признаков на одном слое). Также следует отметить, что при переборе слоя матрицей весов её передвигают обычно не на полный шаг (размер этой матрицы), а на небольшое расстояние. Так, например, при размерности матрицы весов 5×5 её сдвигают на один или два нейрона (пикселя) вместо пяти, чтобы не «перешагнуть» искомый признак.

Операция субдискретизации (англ. *subsampling*, англ. *pooling*, также переводимая как «операция подвыборки» или операция объединения), выполняет уменьшение размерности сформированных карт признаков. В данной архитектуре сети считается, что информация о факте наличия искомого признака важнее точного знания его координат, поэтому из нескольких соседних нейронов карты признаков выбирается максимальный и принимается за один нейрон уплотнённой карты признаков меньшей размерности. За счёт данной операции, помимо ускорения дальнейших вычислений, сеть становится более инвариантной к масштабу входного изображения.

Рассмотрим типовую структуру свёрточной нейронной сети более подробно (рисунок 5). Сеть состоит из большого количества слоёв. После начального слоя (входного изображения) сигнал проходит серию свёрточных слоёв, в которых чередуется собственно свёртка и субдискретизация (пулинг).

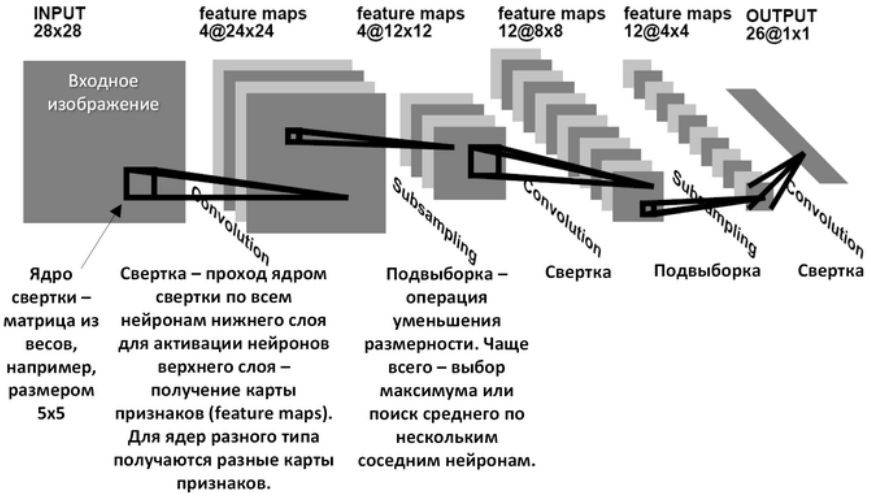

Рисунок 5. Типовая архитектура свёрточной нейронной сети

Чередование слоёв позволяет составлять «карты признаков» из карт признаков, на каждом следующем слое карта уменьшается в размере, но увеличивается количество каналов. На практике это означает способность распознавания сложных иерархий признаков. Обычно после прохождения нескольких слоёв карта признаков вырождается в вектор или даже скаляр, но таких карт признаков становятся сотни. На выходе свёрточных слоёв сети дополнительно устанавливают несколько слоёв полносвязной нейронной сети (перцептрон), на вход которому подаются оконечные карты признаков.

Основным назначением CNN является выделение из исходного изображения малых частей, содержащих опорные (характерные) признаки *(features)*, такие как ребра, контуры, дуги или грани. На следующих уровнях обработки из этих ребер можно распознать более сложные повторяемые фрагменты текстур, которые дальше могут сложиться в еще более сложные текстуры.

2.3.2 Слои модели для построения свёрточной сети

Основным элементом CNN является фильтр (матричной формы), который продвигается вдоль изображения с шагом в один пиксел (ячейку) вдоль горизонтальной и вертикальной осях, начиная от левого верхнего угла и заканчивая нижним правым. На каждом шаге – CNN выполняет вычисление с целью оценки – на сколько похожа часть изображения попавшиеся в окно фильтра с паттерном самого фильтра.

Предположим, что есть фильтр размерностью 2×2 (матрица K) и он делает проекцию на исходное изображение, которая обязательно тоже имеет размерность 2×2 (матрица N), тогда значение выходного слоя вычисляется следующим образом:

$$\begin{bmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \end{bmatrix} * \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} = n_{11}k_{11} + n_{12}k_{12} + n_{21}k_{21} + n_{22}k_{22}$$

Затем каждый пиксел в паттерне фильтра перемножается с соответствующим пикселем части исходного изображения, попавшего в проекцию фильтра на текущем шаге и суммируем эти значения. Данное выражение имеет похожую форму записи операции суммирования в полносвязанные слоях (*fully-connected, dense layers*):

$$sum = \vec{X}^T \vec{W} = \sum_{i=1}^{n=4} x_i w_i = x_1 w_1 + x_2 w_2 + x_3 w_3 + x_4 w_4$$

Единственным существенным отличием является то, что в полносвязанных слоях операции выполняются над векторо-подобными структурами, а в случае свёрточных слоях – это матрично-подобные структуры, которые учитывают влияние близко расположенных пикселов в вертикальных и горизонтальных осях и при этом игнорируют влияние далеко расположенных пикселов (вне окна фильтра).

Размерность ядра фильтра обычно выбирают квадратной формы и с нечетным количеством элементов вдоль осей матрицы – 3, 5, 7. Если мы имеем форму ядра фильтра $[k_h, k_w]$, а входное изображение размерностью $[n_h, n_w]$, то размерность выходного сверточного слоя будет:

$$c_w = n_w - k_w + 1, \qquad c_h = n_h - k_h + 1$$

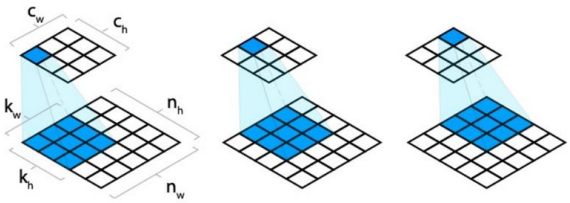

Рисунок 6. Принцип формирования свёрточного выходного слоя с размерностью ядра фильтра [3,3]

Когда выбирается достаточно малое значение ядра фильтра, то потеря в размерности выходного слоя не велика. Если же это значение увеличивается, то и размерность выходного слоя уменьшается по отношению к оригинальному размеру входного изображения. Это может стать проблемой, особенно если в модели нейронной сети подключено последовательно много свёрточных слоев.

Чтобы избежать потери разрешений выходных изображений, в настройках свёрточных слоев используют дополнительный параметр – отступ (*padding*). Это расширяет исходное изображения по краям, заполняя эти ячейки нулевыми значениями. Предположим, что мы добавляем p_h и p_w ячеек к исходному изображению, тогда размер выходного сверточного слоя будет:

$$c_w = n_w + p_w - k_w + 1, \qquad c_h = n_h + p_h - k_h + 1$$

Обычно, размерность отступов задают таким образом, чтобы размерности исходного изображения и выходного сверточного слоя совпадали, тогда они могут быть вычислены следующим образом:

$$p_w = k_w - 1, \qquad p_h = k_h - 1$$

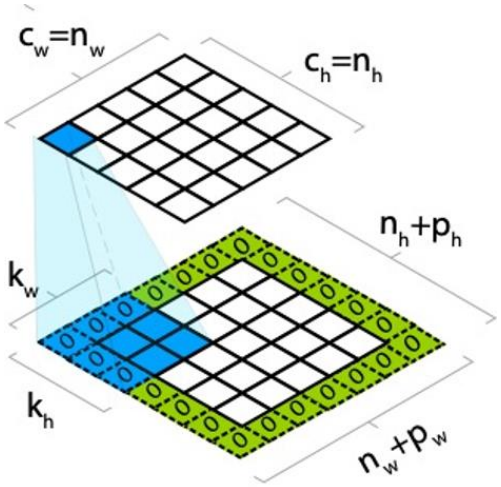

Рисунок 7. Включение отступов в исходное изображения для сохранения той же размерности для выходного сверточного слоя

Сверочный фильтр начинает свое движение с верхнего левого угла изображения и продвигается вдоль горизонтальной и вертикальной осей на одну ячейку вдоль направления движения. Однако иногда для снижения времени вычисления или же для уменьшения размерности выходного слоя, перемещение сверточного слоя может происходить вдоль направления движения больше чем

на одну ячейку (*stride*). Это еще один из параметров сверточного слоя – шаг перемещения (*stride*). Передвижение сверточного слоя с шагом (*stride*) больше единицы показано на рисунке 8.

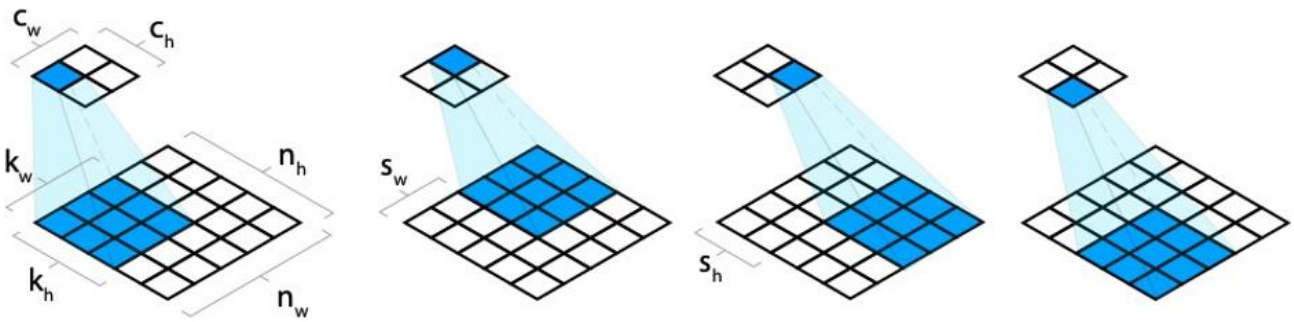

Рисунок 8. Передвижение сверточного слоя с шагом (*stride*) больше единицы

Предполагая, что размер шага вдоль горизонтальной и вертикальной осей равны соответственно s_w, s_h, тогда размер выходного свёрточного слоя составит:

$$c_w = \lfloor (n_w + p_w - k_w + s_w)/s_w \rfloor, c_h = \lfloor (n_h + p_h - k_h + s_h)/s_h \rfloor$$

Так же стоит отметить, что свёрточный слой может содержать один и более фильтров (каждый фильтр – это аналог скрытого слоя с нейронами в полносвязанном слое нейронной сети). Каждый фильтр будет ответственен за извлечение из изображения своих специфичных паттернов (признаков). Представим, что на вход первого сверточного слоя (CONV1) было подано изображение размерностью $9 \times 9 \times 1$ (изображение с одним цветовым каналом – черно-белое изображение), а свёрточный слой имеет 2 фильтра с шагом перемещения ядра 1×1 (*stride*) и отступ (*padding*) подобран таким образом, чтобы выходной слой сохранял туже размерность, что и входной. Тогда размерность выходного слоя будет $9 \times 9 \times 2$, где 2 – это количество фильтров (см. рисунок 9).

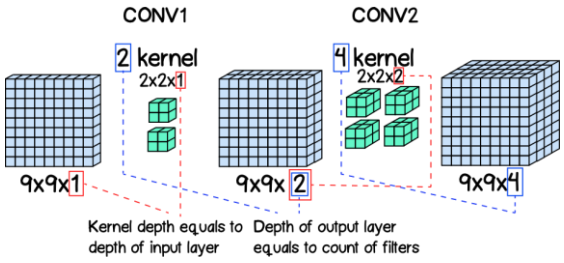

Рисунок 9. Изменение размерности тензоров после несколько последовательных свёрточных слоев

На следующем шаге в CONV2 свёрточном слое при задании размерности фильтра 2×2, его глубина определяется глубиной входного слоя, которой равен 2, тогда ядро будет размерностью $2 \times 2 \times 2$. Выходной слой после второго сверточного слоя (CONV2) тогда будет $9 \times 9 \times 4$, где 4 – количество фильтров в сверточном слое.

На рисунке 10 изображен подход при вычислениях, если на свёрточный слой подано цветное изображение с тремя каналами RGB, а свёрточный слой задан матрицей 3x3. Как было указано выше, так как глубина входного изображения равна трем, то фильтр будет иметь размерность $3 \times 3 \times 3$.

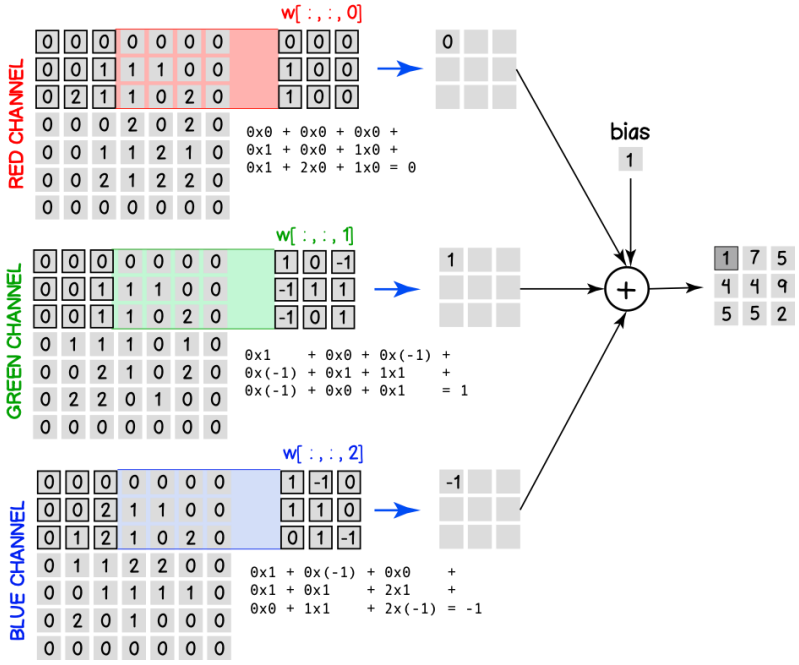

Рисунок 10. Вычисления в свёрточном слое, если входное изображение имеет три канала RGB

2.3.3 Подвыборочный слой (pooling layer)

Обычно для обработки изображения, последовательно подключается несколько свёрточных слоев (обычно это десять или даже сотня слоев), при этом каждый свёрточный слой увеличивает размерность выходного изображения, что в свою очередь приводит к увеличению времени обучения такой сети. Для того, чтобы его снизить, после свёрточных слоев применяют подвыборочный слой (*pooling layer*, *subsample layer*), которые уменьшают размерность выходного

изображения. Обычно применяется операция MaxPooling. Данный слой также снижает чувствительность к точному пространственному расположению признаков изображения, потому что, например, любая написанная цифра от руки может быть смещена от центра абсолютно в любом направлении.

В преобразования в этом слое похожи на преобразования в свёрточном слое. Этот слой также имеет ядро (*kernel*) определенного размера, который скользит вдоль изображения с задаваемым шагом (*stride*) и в отличии от сверточного слоя в по умолчанию он равен не 1×1, а равен размеру задаваемого ядра. В выходном изображении, каждая ячейка будет содержать максимальное значение пикселов в проекции ядра на определённом шаге. Также покажем наглядно, что этот слой сглаживает пространственные смещения во входном слое.

В некоторых источниках преобразование подвыборочного слоя сопоставляют с работой программ по сжатию изображений с целью снизить размер файла, но при этом не потеряв важные характеристики изображения. Также иногда предлагается не использовать этот тип слоя вовсе и для снижения разрешения выходного изображения – манипулировать значением шага перемещения фильтра в свёрточном слое (*stride*).

Также помимо MaxPooling иногда используется AveragePooling, в этом случае считается среднее взвешенное значение пикселей, попавшие в проекции фильтра. Этот фильтр ранее был популярным, но сейчас все же отдают предпочтение MaxPooling. С одной стороны, AveragePooling теряет меньше информации, так как учитывает средневзвешенное значение всех пикселов, в тоже время MaxPooling выкидывает из рассмотрения менее значимые пикселы в окне.

ГЛАВА 3. Особенности реализации алгоритма определения глубины

3.1 Описание используемого оборудования

Реализация данного проекта будет использовать следующие технологии:

1. Оборудование:
    a. Компьютер Ноутбук Xiaomi Mi Notebook на базе процессора Intel Core i5 8250U 1600 МГц
    b. Веб камера Logitech C270 с размером оптического датчика -2,1 МП 2.
2. Средства обработки изображений:
    a. Высокоуровневый язык программирования общего назначения Python
    b. Наборы данных NYU Depth V2, KITTI и DIODE.
    c. Библиотеки компьютерного зрения Open Source Computer Vision Library Angular
    d. PyTorch — фреймворк машинного обучения для языка Python с открытым исходным кодом, созданный на базе Torch.
    e. TensorFlow — открытая программная библиотека для машинного обучения.

3.2 Определение средства разработки

Python уже давно проявил себя как лучший язык программирования в сферах искусственного интеллекта, машинного обучения, машинного зрения и анализа данных.

Для данной работы также будут использоваться следующие библиотеки:

OpenCV — кроссплатформенная библиотека с открытым исходным кодом, поддерживающая фреймворки глубокого обучения такие как TensorFlow, Pytorch и разрабатывалась специально для машинного зрения. OpenCV включает и позволяет:

- 2D и 3D инструментарий
- оценку движения относительно объектов
- распознавание лиц
- распознавание жестов
- человеко-компьютерное взаимодействие
- работу с робототехникой
- распознавание движения
- идентификация объектов
- сегментацию и распознавание
- построение структуры из движения (SFM)
- отслеживание движения
- дополнительная реальность

TensorFlow — открытая программная библиотека для машинного обучения, разработанная компанией Google для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов, достигая качества человеческого восприятия.

Уникальными особенностями TensorFlow являются:

- Основная библиотека подходит для широкого семейства техник машинного обучения, а не только для глубинного обучения.
- Линейная алгебра и другая внутренняя логика хорошо видна снаружи.
- В дополнение к основной функциональности машинного обучения, TensorFlow также включает собственную систему логирования, собственный интерактивный визуализатор логов и даже мощную архитектуру по доставке данных.

Модель исполнения TensorFlow отличается от scikit-learn языка Python и от большинства инструментов в R.

Keras — открытая библиотека, написанная на языке Python и обеспечивающая взаимодействие с искусственными нейронными сетями. Она представляет собой надстройку над фреймворком TensorFlow.

Pandas — программная библиотека на языке Python для обработки и анализа данных. Работа pandas с данными строится поверх библиотеки NumPy, являющейся инструментом более низкого уровня. Предоставляет специальные структуры данных и операции для манипулирования числовыми таблицами и временными рядами. Название библиотеки происходит от эконометрического термина «панельные данные», используемого для описания многомерных структурированных наборов информации.

Matplotlib — библиотека на языке программирования Python для визуализации данных двумерной (2D) графикой (3D графика также поддерживается).

В рамках данной работы были использованы Python 3.8.12, OpenCV 4.5.4, TensorFlow 2.6.2, Keras 2.6.0, Pandas 1.3.4, Matplotlib 3.5.0.

3.3 Прототип программного обеспечения

Прототип автоматизированной обработки данных для получения карты глубины с помощью свёрточной нейронной сети.

3.3.1 Архитектура сети

Для обучения будет использоваться набор данных DIODE: Dense Indoor и Outdoor Depth Dataset.

Для кодировщика входное изображение RGB кодируется в вектор признаков с использованием сети DenseNet-169 [9], предварительно обученной на ImageNet.

Затем этот вектор подается на последовательный ряд верхних слоёв выборки [14], чтобы построить окончательную карту глубины в половине входного разрешения. Эти расширенные слои и связанные с ними скип-соединения формируют декодер.

Среди моделей, доступных в Keras API (приложения Keras), для данной задачи подходят ResNet и DenseNet [9]. DenseNet-169 был выбран потому, что, несмотря на то, что он имеет глубину в 169 слоев, он имеет относительно низкие

параметры по сравнению с другими моделями, а архитектура хорошо справляется с проблемой исчезновения градиента.

Таблица 1 показывает структуру нашего кодировщика-декодера с сетью скип-соединений. Наш кодировщик основан на сети DenseNet-169, где мы удаляем верхние слои, которые связаны с исходной задачей классификации ImageNet.

Таблица 1. Структура кодировщика-декодера

layer	**output**	**function**
INPUT	$480 \times 640 \times 3$	
CONV1	$240 \times 320 \times 64$	DenseNet CONV1
POOL1	$120 \times 160 \times 64$	DenseNet POOL1
POOL2	$60 \times 80 \times 128$	DenseNet POOL2
POOL3	$30 \times 40 \times 256$	DenseNet POOL3
…	…	…
CONV2	$15 \times 20 \times 1664$	Convolution 1×1 of DenseNet BLOCK4
UP1	$30 \times 40 \times 1664$	Upsample 2×2
CONCAT1	$30 \times 40 \times 1920$	Concatenate POOL3
UP1-CONVA	$30 \times 40 \times 832$	Convolution 3×3
UP1-CONVB	$30 \times 40 \times 832$	Convolution 3×3
UP2	$60 \times 80 \times 832$	Upsample 2×2
CONCAT2	$60 \times 80 \times 960$	Concatenate POOL2
UP2-CONVA	$60 \times 80 \times 416$	Convolution 3×3
UP2-CONVB	$60 \times 80 \times 416$	Convolution 3×3
UP3	$120 \times 160 \times 416$	Upsample 2×2
CONCAT3	$120 \times 160 \times 480$	Concatenate POOL1
UP3-CONVA	$120 \times 160 \times 208$	Convolution 3×3
UP3-CONVB	$120 \times 160 \times 208$	Convolution 3×3
UP4	$240 \times 320 \times 208$	Upsample 2×2
CONCAT3	$240 \times 320 \times 272$	Concatenate CONV1
UP2-CONVA	$240 \times 320 \times 104$	Convolution 3×3
UP2-CONVB	$240 \times 320 \times 104$	Convolution 3×3
CONV3	$240 \times 320 \times 1$	Convolution 3×3

1. Для декодера начинаем с 1×1 свёрточного слоя с тем же количеством выходных каналов, что и выход усеченного кодировщика.
2. Затем мы последовательно добавляем блоки расширенной выборки,

каждый из которых состоит из 2 билинейных расширений, за которыми следуют два 3×3 свёрточных слоя с выходными фильтрами, равными половине входных фильтров, и первый свёрточный слой из этих двух применен на соединении выхода предыдущего слоя и слоя пулинга от кодировщика с одинаковым пространственным измерением.

3. За каждым блоком, за исключением последнего, следует функция активации ReLU с параметром $\alpha = 0{,}2$.

Программная реализация алгоритма представлена в листинге:

```
def create_model(existing='', is_twohundred=False, is_halffeatures=True):
    if len(existing) == 0:
        print('Loading base model (DenseNet)..')

        # Encoder Layers
        if is_twohundred:
            base_model = applications.DenseNet201(input_shape=(None, None, 3), include_top=False)
        else:
            base_model = applications.DenseNet169(input_shape=(None, None, 3), include_top=False)
        print('Base model loaded.')

        # Starting point for decoder
        base_model_output_shape = base_model.layers[-1].output.shape

        # Layer freezing?
        for layer in base_model.layers: layer.trainable = True

        # Starting number of decoder filters
        if is_halffeatures:
            decode_filters = int(int(base_model_output_shape[-1]) / 2)
        else:
            decode_filters = int(base_model_output_shape[-1])

        # Define upsampling layer
        def upproject(tensor, filters, name, concat_with):
            up_i = BilinearUpSampling2D((2, 2), name=name + '_upsampling2d')(tensor)
            up_i = Concatenate(name=name + '_concat')(
                [up_i, base_model.get_layer(concat_with).output])  # Skip connection
            up_i = Conv2D(filters=filters, kernel_size=3, strides=1, padding='same', name=name + '_convA')(up_i)
            up_i = LeakyReLU(alpha=0.2)(up_i)
            up_i = Conv2D(filters=filters, kernel_size=3, strides=1, padding='same', name=name + '_convB')(up_i)
            up_i = LeakyReLU(alpha=0.2)(up_i)
            return up_i

        # Decoder Layers
        decoder = Conv2D(filters=decode_filters, kernel_size=1, padding='same', input_shape=base_model_output_shape,
                         name='conv2')(base_model.output)

        decoder = upproject(decoder, int(decode_filters / 2), 'up1', concat_with='pool3_pool')
        decoder = upproject(decoder, int(decode_filters / 4), 'up2', concat_with='pool2_pool')
        decoder = upproject(decoder, int(decode_filters / 8), 'up3', concat_with='pool1')
        decoder = upproject(decoder, int(decode_filters / 16), 'up4', concat_with='conv1/relu')
        if False: decoder = upproject(decoder, int(decode_filters / 32), 'up5', concat_with='input_1')
```

```
    # Extract depths (final layer)
    conv3 = Conv2D(filters=1, kernel_size=3, strides=1, padding='same', name='conv3')(decoder)

    # Create the model
    model = Model(inputs=base_model.input, outputs=conv3)
else:
    # Load model from file
    if not existing.endswith('.h5'):
        sys.exit('Please provide a correct model file when using [existing] argument.')
    custom_objects = {'BilinearUpSampling2D': BilinearUpSampling2D, 'depth_loss_function': depth_loss_function}
    model = load_model(existing, custom_objects=custom_objects)
    print('\nExisting model loaded.\n')

print('Model created.')

return model
```

3.3.2 Функция потерь

Стандартная функция потерь для задач регрессии глубины рассматривает разницу между картой глубины основания и предсказанием сети $\hat{y}$ [7, . Различные соображения, касающиеся функции потерь, могут оказать значительное влияние на скорость тренировки и общую эффективность оценки глубины. Множество вариаций функции потерь, используемых для оптимизации нейронной сети, можно найти в литературе по оценке глубины. В данной задаче необходима функция потерь, которая балансирует между восстановлением изображений глубины, минимизируя, разницу значений глубины, а также предусматривает искажения высокочастотных деталей в области изображения карты глубины. Эти детали обычно соответствуют границам объектов в сцене.

Для обучения сети определим потерю l между y и $\hat{y}$ как взвешенную сумму трех функций потерь:

$$l(y, \hat{y}) = \lambda l_{depth}(y, \hat{y}) + \lambda l_{grad}(y, \hat{y}) + l_{SSIM}(y, \hat{y})$$

Первая функция потерь l_{depth} - это значение потерь l, определяемое по значениям глубины:

$$l_{depth}(y, \hat{y}) = \frac{1}{n}\sum_{p}^{n} |y_p - \widehat{y_p}|.$$

Вторая функция потерь l_{depth} - это потеря, определяемая по градиенту изображения g изображения глубины:

$$l_{grad}(y,\hat{y}) = \frac{1}{n}\sum_p^n \left|g_x(y_p,\widehat{y_p})\right| + \left|g_y(y_p,\widehat{y_p})\right|,$$

где g_x и g_y, соответственно, вычисляют различия в компонентах x и y для градиентов глубины y и $\hat{y}$.

И наконец, индекс структурного сходства (SSIM от англ. *structure similarity*), который является широко используемой метрикой для задач реконструкции изображений. Недавно было показано, что это хороший способ анализа потерь для оценки глубины CNN. Поскольку SSIM имеет верхнюю границу единицу, определим его как потерю l_{SSIM} следующим образом:

$$l_{\text{SSIM}}(y,\hat{y}) = \frac{1-SSIM(y,\hat{y})}{2}.$$

Реализация алгоритмов нахождения ошибок показана в листинге программы:

```
def depth_loss_calculation(y_true, y_predicted, theta=0.1, maxDepthVal=1000.0 / 10.0):
    l_depth = keras_b.mean(keras_b.abs(y_predicted - y_true), axis=-1)

    dy_true, dx_true = tf.image.image_gradients(y_true)
    dy_predicted, dx_predicted = tf.image.image_gradients(y_predicted)

    l_grad = keras_b.mean(keras_b.abs(dx_predicted - dx_true) + keras_b.abs(dy_predicted - dy_true), axis=-1)

    l_ssim = keras_b.clip((1 - tf.image.ssim(y_true, y_predicted, maxDepthVal)) * 0.5, 0, 1)

    w1, w2, w3 = 1.0, 1.0, theta
    return (w1 * l_ssim) + (w2 * keras_b.mean(l_grad)) + (w3 * keras_b.mean(l_depth))
```

Пример оценки глубины с помощью данного метода приведен на рисунке 10.

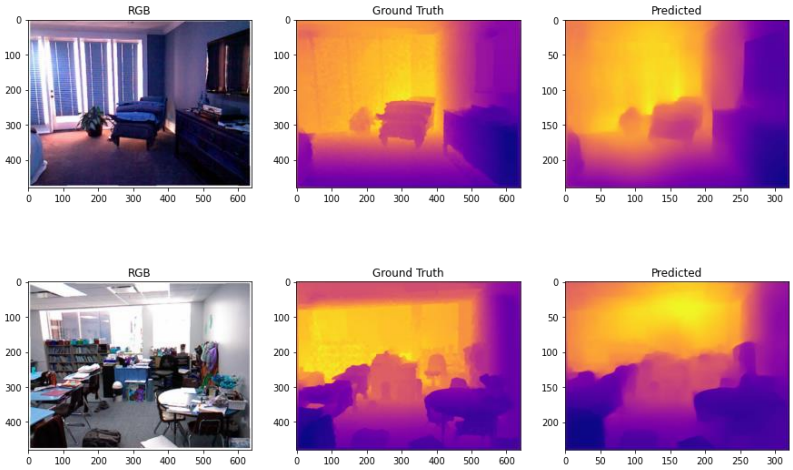

Рисунок 10. Пример построения карты глубины

ГЛАВА 4. ЗАКЛЮЧЕНИЕ

В рамках данной работы были рассмотрены монокулярные методы определения глубины точек сцены на основе глубокого обучения, а также разработан алгоритм детектирования объектов на изображении. Для реализации была использована библиотека компьютерного зрения Open Source Computer Vision Library Angular, которая позволяет разрабатывать программы анализа движения на изображении, и высокоуровневый язык программирования общего назначения Python, библиотеки машинного обучения Keras и Tensorflow, а также библиотеки Pandas и Matplotlib для предварительной обработки и визуализации данных. Результаты данной работы можно применять в сферах машинного зрения, например, для установки в видеорегистраторы или камеры видеонаблюдения, робототехнике. Реализованный алгоритм предназначен для имитации карты глубины на основе видеопотока с монокамеры. Анализ результатов обработки показывает, что контуры объектов выровнены, объекты переднего и заднего фона имеют точные очертания, что говорит о качественном восстановлении карты глубины.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. W. Lee, N. Park, and W. Woo. Depth-assisted real-time 3d object detection for augmented reality. ICAT11, 2:126–132, 2011.
2. F. Moreno-Noguer, P. N. Belhumeur, and S. K. Nayar. Active refocusing of images and videos. ACM Trans. Graph., 26(3), July 2007.
3. C. Hazirbas, L. Ma, C. Domokos, and D. Cremers. Fusenet: Incorporating depth into semantic segmentation via fusionbased cnn architecture. In ACCV, 2016.
4. Laganiere, R. OpenCV 2 Computer Vision Application Programming Cookbook / R. Laganiere. – Mumbai: Packt Publishing, 2001. – 287 p.
5. D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In NIPS, 2014.
6. B. Li, C. Shen, Y. Dai, A. van den Hengel, and M. He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs.
7. I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. 2016 Fourth International Conference on 3D Vision (3DV), pages 239–248, 2016.
8. D. Xu, E. Ricci, W. Ouyang, X. Wang, and N. Sebe. Multiscale continuous crfs as sequential deep networks for monocular depth estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5354–5362, 2017.
9. G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2261–2269, 2017.
10. A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? The KITTI vision benchmark suite, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012, 2012, pp. 3354–3361. doi:10.1109/CVPR.2012. 6248074.
11. A. Saxena, M. Sun, A. Y. Ng, Make3D: Learning 3D Scene Structure from a Single Still Image, IEEE Trans. Pattern Anal. Mach. Intell. 31 (5) (2009) 824–840. doi:10.1109/TPAMI. 2008.132.
12. N. Silberman, D. Hoiem, P. Kohli, R. Fergus, Indoor Segmentation and Support Inference from RGBD Images, in: Computer Vision - ECCV 2012 - 12th European Conference on 38 Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V, 2012, pp. 746– 760. doi:10.1007/978-3-642-33715-4_54.
13. Igor Vasiljevic, Nick Kolkin, Shanyi Zhang, Ruotian Lu,o Haochen

Wang,Falcon Z. Dai Andrea, F. Daniele, Mohammadreza Mostajabi, Steven Basart, Matthew R., Walter Gregory Shakhnarovich, DIODE: A Dense Indoor and Outdoor DEpth Dataset, TTI-Chicago, University of Chicago, Beihang University, 29 Aug 2019 – 8 p.

14. J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila. Noise2Noise: Learning image restoration without clean data. In J. Dy and A. Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 2965–2974, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

15. B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. Demon: Depth and motion network for learning monocular stereo. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5622–5631, 2017.