Автономная некоммерческая образовательная организация высшего образования
“Сколковский институт науки и технологий”

На правах рукописи

Егиазарян Ваге Грайрович

ВЕКТОРИЗАЦИЯ ИЗОБРАЖЕНИЙ С ПОМОЩЬЮ ГЛУБОКОГО ОБУЧЕНИЯ

РЕЗЮМЕ
диссертации на соискание ученой степени
кандидата компьютерных наук

Научный руководитель:
доктор физико-математических наук
Бурнаев Евгений Владимирович

Москва — 2024

Содержание

1. Тема диссертации

Темой данного исследования является разработка эффективных методов решения задач векторизации blue растровых изображений и изображений глубины для трехмерных объектов с использованием подходов глубокого обучения. Векторизацией объектов называют представление объекта с использованием математических примитивов и связей между ними.

Для достижения данной цели в рассматриваемой работе были предложены способы решения следующих задач: сбор данных, построение математических моделей и разработка алгоритмов векторизации. Сбор данных осуществлялся с помощью синтетической генерации данных и обработки реальных отсканированных изображений двухмерных и трехмерных объектов. Для очистки и обработки данных были разработаны автоматические алгоритмы с использованием методов компьютерного зрения, а также определены процедуры для ручной обработки данных. Предложенные алгоритмы позволяют получать размеченные данные в полуавтоматическом режиме, что открывает возможность использования методов глубокого обучения для тренировки нейронных сетей.

Для построения архитектур, хорошо решающих поставленные задачи, были исследованы различные архитектуры нейронных сетей, такие как сверточные и рекуррентные сети, а также трансформеры. Целью построения оптимальной архитектуры является создание моделей, способных точно и эффективно векторизовать различные типы изображений и трехмерных объектов.

Предложенные алгоритмы демонстрируют высокую точность и эффективность в решении задач векторизации объектов. Эти алгоритмы могут быть применены в различных областях, таких как компьютерное зрение, робототехника и визуализация данных, с целью представления объектов в виде векторной графики с высокой степенью точности и детализации.

Актуальность работы

Существует много программных продуктов для распознавания изображений и документов, однако большинство из них фокусируется на распознавании текста и либо не работают с изображениями, либо плохо справляются с ними. Как правило, эти продукты не позволяют представлять результат распознавания в виде векторной графики, одного из наиболее удобных для последующей работы форматов. В случае двухмерных и трехмерных данных существующие подходы векторизации, основанные исключительно на алгоритмах оптимизации, имеют большое количество настраиваемых гиперпараметров и плохо справляются с шумами в исходных данных. Использование нейронных сетей позволяет снизить количество гиперпараметров, и благодаря тому, что нейронные сети обучаются распознавать паттерны, удается эффективно обрабатывать даже существенно зашумленные данные.

Актуальность работы обусловлена тем, что векторизация изображений и форм является важной и сложной задачей в области компьютерного зрения, которая полноценно еще не ре-

шена. В данной диссертации рассматривается задача точной векторизации для двухмерных технических чертежей и трехмерных моделей деталей, и предлагается метод распознавания чертежей и объектов с помощью нейронных сетей в тех задачах, где требуется точное и гибкое представление двухмерных и трехмерных объектов. Результаты исследования могут быть применены в различных приложениях, таких как автоматическое распознавание и анализ технических чертежей, автоматизированное моделирование и редактирование 3D объектов, а также виртуальная и дополненная реальность.

Целью данной диссертации является исследование и разработка эффективных методов векторизации двухмерных технических чертежей и трехмерных моделей в области компьютерного зрения. Основные задачи включают в себя изучение существующих методов, сбор и обработку данных, разработку алгоритмов глубокого обучения, оценку и сравнение методов векторизации и исследование их практической применимости.

Цель достигается с помощью решения следующих **задач**:

- получение высококачественных геометрических данных для двухмерных и трехмерных объектов для последующего обучения нейросетевых моделей;
- разработка новой системы для векторизации растровых изображений технических чертежей;
- разработка алгоритма для восстановления параметрических моделей особых кривых трехмерных объектов.

Работа основана на использовании **методологии и методов** машинного обучения, дифференциальной геометрии и математического моделирования.

2. Основные результаты

Работа основана на использовании **методологии и методов** компьютерного зрения, компьютерной графики, машинного обучения и глубокого обучения.

Научная новизна работы может быть описана следующими положениями:

1. Был предложен новый метод для генерации трехмерных объектов высокого разрешения на разных масштабах. Была разработана новая глубокая многомасштабная модель для генерации облаков точек, Latent-Space Laplacian Pyramid GAN, основанная на передовых технологиях в области обучения генеративно-состязательных нейросетей для данных в виде облаков точек [1] и подходах для моделирования данных на разных масштабах [8, 26].
2. Впервые была предложена система, позволяющая точнее решить задачу векторизации отсканированных чертежей. Система включает в себя несколько компонент: специальным образом подготовленные высококачественные данные для обучения нейросетевых

моделей, новая архитектура обучаемой модели нейронной сети для векторизации, новый подход к оптимизации примитивов для построения финального представления объекта в векторном формате. Обучаемые нейросетевые модели используются на нескольких шагах данного подхода: для предварительной обработки изображения, которая позволяет снизить уровень шума; для получения начального векторного приближения изображения. Далее, с помощью решения задачи оптимизации, из полученного начального приближения строится конечный результат, а также делается его постобработка с целью минимизации количества примитивов.

3. Был представлен новый метод извлечения параметрических кривых из трехмерных облаков точек, позволяющий точно преобразовывать облака точек в аналитические модели особых кривых в трехмерном пространстве, которые описывают структурные особенности трехмерных моделей деталей и необходимы для дальнейшего построения 3D описаний форм этих объектов.

   Данные результаты были опубликованы в трудах международных конференций уровня Core A* и Core B, при этом статьи прошли двойное слепое рецензирование, а доклады - были представлены на международных конференциях.

Теоретическая и практическая значимость.

Теоретическая значимость работы обусловлена новыми методами и алгоритмами в области обработки изображений, векторизации и генерации трехмерных данных. В исследовании проанализировано текущее состояние подходов к векторизации и генерации изображений и трехмерных данных. Для построения моделей и методов применены передовые подходы из машинного обучения, современные архитектуры нейросетей, и методы оптимизации. В результате осмысления полученных практических результатов в работе расширено понимание границ применимости современных методов машинного обучения для решения задач векторизации.

Разработанные методы имеют практическую применимость в различных областях, таких как автоматическое распознавание и анализ технических чертежей, автоматизированное моделирование и редактирование трехмерных объектов.

Основные положения и результаты, выносимые на защиту:

1. *Методология* и алгоритм генерации данных,
2. *Алгоритм* преобразования растровых технических чертежей в векторную графику с сохранением информации,
3. *Алгоритм* преобразования трехмерных отсканированных объектов в векторную графику состоящую из трехмерных кривых,
4. *Методология* для оценки точности и качества векторизованных данных.

Достоверность полученных результатов обеспечивается корректным использованием протестированного в научной практике набора инструментов для исследования и анализа. Предложенные алгоритмы были экспериментально проверены на разнообразных задачах и реальных наборах данных, описывающих как двухмерные, так и трёхмерные объекты. Подробные описания проведенных экспериментов и программный код в общем доступе позволяют повторить полученные результаты. Результаты исследования были опубликованы в ведущих научных журналах и на конференциях, посвященных компьютерному зрению и распознаванию образов.

Личный вклад в результаты, выносимые на защиту. Все заявленные результаты были получены автором данной диссертации. Во всех специально упомянутых случаях как сам текст статей, так и экспериментальные результаты, представленные в них, являются результатами совместной работы с другими авторами.

В статье 1: ``Deep vectorization of technical drawings'' Егиазарян В.Г., как основной автор, разработал и реализовал систему для обработки входных изображений с последующим получением векторного представления. Кроме того, автор был ответственен за разработку нейросетевой архитектуры и алгоритма её обучения, которая преобразует растровые изображения в векторные примитивы, и совместно с соавторами разработал методы и дизайн экспериментов для оценки эффективности предложенного метода.

В статье 2: "Def: Deep estimation of sharp geometric features in 3d shapes" Егиазарян В.Г. был ответственен за разработку и улучшение второй компоненты всего пайплайна: алгоритм восстановления параметрических особых кривых из облаков точек.

В статье 3: "Latent-Space Laplacian Pyramid GAN" автор диссертации разработал и реализовал алгоритм многомасштабной генерации трехмерных облаков точек, а также методологию для оценки точности и качества полученных сгенерированных объектов.

3. Публикации и апробация работы

Публикации повышенного уровня

1. *Ваге Егиазарян**, Олег Войнов*, Алексей Артемов, Денис Волхонский, Александр Сафин, Мария Тактачева, Денис Зорин, и Евгений Бурнаев. Deep vectorization of technical drawings. Статья опубликована в тредах конференции European Conference on Computer Vision (ECCV), стр. 582–598, 2020. Проиндексирована в SCOPUS. Конференция ранга A* по версии CORE.

2. Алберт Матвеев*, Руслан Рахимов*, Алексей Артемов, Глеб Бобровских, *Ваге Егиазарян*, Эмиль Богомолов, Даниэле Паноззо, Денис Зорин, и Евгений Бурнаев. DEF: Deep estimation of sharp geometric features in 3d shapes. Статья опубликована в трудах конференции SIGGRAPH 2022, журнал ACM Transactions on Graphics (TOG), 41(4):1–22, 2022. Проиндексирована в SCOPUS. Конференция ранга A* по версии CORE.

Публикации стандартного уровня

3. *Ваге Егиазарян**, Савва Игнатьев*, Алексей Артемов, Олег Войнов, Андрей Кравченко, Йоуи Женг, Луиз Велхо и Евгений Бурнаев. Latent-space laplacian pyramids for adversarial representation learning with 3d point clouds. Представлена на конференции VISIGRAPP --- Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, стр. 421–428, 2020. Проиндексирована в SCOPUS. Конференция ранга B по версии CORE.

Автор также внес вклад в следующие работы

4. Олег Войнов, Алексей Артемов, *Ваге Егиазарян*, Александр Нотченко, Глеб Бобровских, Евгений Бурнаев, Денис Зорин. *Perceptual deep depth super-resolution.* Представлена на конференции IEEE/CVF International Conference on Computer Vision, стр. 5653-5663, 2019. Проиндексирована в SCOPUS. Конференция ранга A* по версии CORE.

5. Денис Мазур*, Станислав Морозов*, *Ваге Егиазарян**, Артем Бабенко. *Beyond vector spaces: Compact data representation as differentiable weighted graphs.* Представлена на конференции Advances in Neural Information Processing Systems (NeurIPS), Том. 32, 2019. Проиндексирована в SCOPUS. Конференция ранга A* по версии CORE.

6. Александр Коротин, *Ваге Егиазарян*, Арип Асадулев, Александр Сафин, Евгений Бурнаев. *Wasserstein-2 generative networks.* Представлена на конференции International Conference on Learning Representations (ICLR), 2021. Проиндексирована в SCOPUS. Конференция ранга A* по версии CORE.

7. Александр Коротин, *Ваге Егиазарян*, Лингхиао Ли, Евгений Бурнаев. *Wasserstein iterative networks for barycenter estimation.* Представлена на конференции Advances in Neural

Information Processing Systems (NeurIPS), 2022. Том 35. Проиндексирована в SCOPUS. Конференция ранга A* по версии CORE.

8. Арип Асадулев, Александр Коротин,*Ваге Егиазарян*, Петр Мокров, Евгений Бурнаев. *Neural optimal transport with general cost functionals.* Представлена на конференции International Conference on Learning Representations (ICLR), 2024. Проиндексирована в SCOPUS. Конференция ранга A* по версии CORE

9. Тим Деетмерс*, Руслан Свирчевски*, *Ваге Егиазарян**, Денис Кузнеделев*, Элиас Франтар, Салей Ашкбус, Александр Борзунов, Торстен Хоефлер, Дэн Алистарх. *SpQR: A Sparse-Quantized Representation for Near-Lossless LLM Weight Compression.* Представлена на конференции International Conference on Learning Representations (ICLR), 2024. Проиндексирована в SCOPUS. Конференция ранга A* по версии CORE

10. *Ваге Егиазарян**, Андрей Панферов* , Денис Кузнеделев*, Элиас Франтар, Артем Бабенко, Дэн Алистарх. *Extreme Compression of Large Language Models via Additive Quantization.* Представлена на конференции The Forty-first International Conference on Machine Learnings (ICML), 2024. Проиндексирована в SCOPUS. Конференция ранга A* по версии CORE.

Апробация работы

- “Deep vectorization of technical drawings”, постерная презентация на конференции 16th European Conference on Computer Vision (ECCV), Онлайн, 2020;
- «Векторизация с помощью глубокого обучения», доклад на семинаре Ассоциации «Искусственный интеллект в промышленности», онлайн;
- “Def: Deep estimation of sharp geometric features in 3d shapes”, постерная презентация на конференции SIGGRAPH, Канада, 2022;
- “Latent-space Laplacian pyramids for adversarial representation learning with 3d point clouds”, постерная презентация на конференции VISAPP, Мальта, 2020.

* - равный вклад

4. Структура диссертации

Основная часть диссертации состоит из четырех разделов.

В разделе 5 описывается методика получения данных для векторизации. Рассматриваются способы получения как синтетических, так и реальных данных, необходимых для обучения и оценки моделей векторизации.

В разделе 6 представлена новая система обработки данных для 2D технических рисунков с целью получения векторной графики. Описываются методы и алгоритмы, используемые для преобразования растровых изображений в векторные представления.

В разделе 7 описывается новый метод генерации трехмерных объектов высокого качества. Рассматриваются существующие методы генерации трехмерных данных и описывается общая архитектура и структура предложенного нового метода.

В разделе 8 предлагается метод получения трехмерных параметрических кривых из трехмерных объектов. Проводится сравнение эффективности разработанного метода по сравнению с существующими подходами.

5. Описание данных и постановка задач

Векторные представления часто используются для технических изображений и трехмерных моделей, таких как архитектурные планы, инженерные чертежи, трехмерные модели зданий и деталей и т.д. Востребованность векторных представлений обусловлена рядом преимуществ. Во-первых, векторная графика не зависит от масштаба, гораздо более компактна и, что наиболее важно, поддерживает простое редактирование на уровне примитивов. Другим важным преимуществом векторного представления является то, что оно обеспечивает основу для семантической структуры более высокого уровня (например, с наборами примитивов, иерархически сгруппированных в семантические объекты).

Однако во многих случаях векторное представление недоступно, а доступен лишь ``сырой'' объект. В двухмерном случае это отсканированное изображение, в трехмерном случае чаще всего изображения в RGB-D формате или облако точек. Примерами таких отсканированных изображений являются сканы выполненных вручную чертежей, для которых доступна только печатная копия исходного источника, или изображения, найденные в онлайн-коллекциях. Когда исходное векторное представление документа недоступно, его обычно реконструируют вручную на основе сканов или фотографий. Преобразование в векторное представление обычно называется *векторизацией*.

Существует широкий спектр методов, которые часто решают различные вариации проблемы векторизации и ориентированы на различные типы входных и выходных данных. Большинство из них не основаны на подходе машинного обучения и применяют алгоритмические и оптимизационные методы для решения задачи векторизации. Однако, такие подходы обычно имеют ряд недостатков, включая наличие большого количества гиперпараметров, низкую

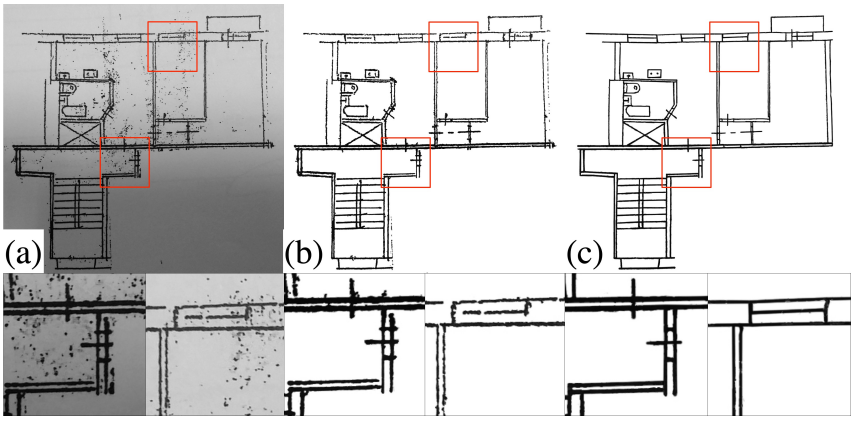


Рис. 1: Пример из набора данных DLD: (а) исходное входное изображение, (б) изображение, очищенное от фона и шума, (в) конечная целевая маска с заполненными линиями.

скорость работы, а также не эффективно учитывают особенности обработки различных типов изображений.

С развитием глубокого обучения естественным образом возникает идея использования подходов из этой области для решения задачи векторизации. Однако, такие методы также имеют свои собственные ограничения, включая потребность в большом объеме обучающих данных. В случае векторизации требуется значительный объем входных и выходных пар данных для обучения нейронных сетей.

Сбор реальных данных представляет собой трудоемкую задачу, требующую значительных ресурсов [24, 38]. Генерация синтетических данных является менее ресурсоёмкой, однако полученные данные могут существенно отличаться от реальных данных и содержать доменные сдвиги. Это означает, что нейронные сети, обученные только на синтетических данных, могут показывать плохие результаты при применении их на реальных данных. В связи с этим хорошей стратегией является использование комбинации синтетических и реальных данных.

5.1. Описание данных для двухмерных чертежей

Существует несколько способов получения пар растровых и векторных изображений. Можно сначала собрать растровые изображения и далее отслеживать линии вручную, создавая таким образом векторное представление. Такой метод является верным с точки зрения соответствия выходного векторного представления входному растровому изображению, но является ресурсоёмким. В этом случае местоположение и стиль векторного изображение может не соответствовать оригинальному рисунку. Другой способ — начать с векторного изображения и визуализировать из него растровое изображение. Этот подход более дешевый и требует значительно меньше человеческих ресурсов, но не обязательно создает реалистичные растровые изображения, поскольку известно, что ухудшение качества документов реального мира сложно моделировать [21]. В связи с указанными ограничениями было решено использовать оба способа.

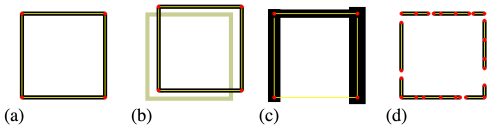


Рис. 2: (a) Векторное изображение, являющееся эталонным, и артефакты, на основе которых оценивается производительность векторизации. (b) Отклонение структуры скелета. (c) Искажение формы. (d) Чрезмерная параметризация.

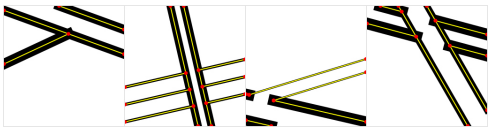


Рис. 3: Примеры синтетических тренировочных данных, использованных далее для обучения предложенной в работе нейросети для извлечения примитивов.

Задачу векторизации изображений можно разделить на две части: улучшение качества растрового изображения, чтобы избавиться от шумов и артефактов, и получение пар чистых растровых и векторных изображений.

Для решения задачи улучшения качества растрового изображения был собран набор данных реальных технических чертежей DLD и сгенерирован набор синтетических данных.

Реальные данные состоят из фотографий и сканов 81 плана этажей с разрешением $\sim 1300 \times 1000$. Чтобы подготовить растровые и соответствующие эталонные данные, была проведена ручная очистка каждого изображения с помощью удаления текста, фона и шума, а также улучшения структуры линий за счет закрашивания пробелов и повышения резкости краев (см. рис. 1).

Синтетические данные включают в себе 20000 пар изображений с разрешением 512×512. Эталонные данные были получены путем случайной генерации математических описаний примитивов. Входные изображения были получены с помощью рендеринга эталонных данных поверх одного из 40 реалистичных отсканированных бумажных фонов и ухудшения рендеринга случайным размытием, искажением, шумом и т.д. (см. рис. 2).

Для получения пар чистых растровых и векторных изображений был создан набор данных **PFP vector floor plan dataset**, который включает в себе 1554 архитектурных планов зданий из реального мира [35], и **ABC vector mechanical parts dataset** включающий в себя 10000 векторных изображений механических деталей, полученных из набора данных [23] с помощью проекции их в двухмерный канвас с использованием [34] (см. рис. 3).

5.2. Описание данных для трехмерных объектов

Трехмерные объекты имеют разные представления, среди которых чаще всего используются следующие: облака точек, RGB-D, CAD модели и модели на основе полигональных сеток (представление в виде mesh). RGB-D и облако точек можно получить путем измерения расстояния до поверхности объекта с помощью трехмерных сканеров (например, лидара, камеры RGB-

D или сканера структурированного света) или вычислить с использованием алгоритмов стереосопоставления. Mesh и CAD модели обычно создаются в процессе моделирования объектов инженерами либо дизайнерами в профессиональных программах таких как Blender, AutoCad и т.д. Как и в случае двухмерных данных, для трехмерных моделей существует несколько способов получения обучающей выборки. Один из подходов, сложный в реализации, но позволяющий относительно дешево создать множество данных — это использование генеративного моделирования. Данные, созданные таким образом, являются синтетическими и чаще всего менее реалистичны, но при этом имеется возможность контроля процесса генерации и, соответственно, свойств генерируемых данных. В качестве альтернативы можно выделить преобразование данных из CAD-моделей или других представлений (см., например, набор данных ABC [23]). Эти данные обычно более высокого качества по сравнению с результатами генеративного моделирования, и из них можно получить не только входные данные, но и эталонные. Однако их объём ограничен количеством исходных 3D моделей, и они также не могут моделировать реальные распределения отсканированных данных. Предиктивные модели, обученные на вышеуказанных данных, могут не всегда хорошо показывать себя на реальных данных из-за доменного сдвига. Наиболее точный, хотя и наиболее затратный способ получения обучающей выборки, это получение данных путем сканирования реальных объектов с использованием лидара, камеры RGB-D или сканера структурированного света [40].

Получение выборки из CAD моделей

Данные, представленные в [28], получены из CAD моделей [23] и являются коллекцией локальных участков и полных трехмерных моделей в различных разрешениях: низком, среднем и высоком, а также содержат шум разного уровня.

Для получения данных были использованы n_v виртуальных камер с равномерно распределенными местоположениями на сфере вокруг объекта (использование выборки Фибоначчи [16]), причем ось z указывает на центр масс объекта. Для каждой камеры была создана регулярная сетка (изображение) размера 64×64 пикселя (указываем r в качестве размера пикселя) и от каждого угла пикселя были направлены лучи перпендикулярно сетке. Таким образом были получены участки до 4096 точек (точки вне объекта определяются как фоновое значение).

Зная параметры камеры K, T, где $K \in \mathbb{R}^{2\times3}$ является внутренней матрицей, преобразующей координаты точек из системы координат камеры на плоскость изображения, а $T \in \mathbb{R}^{4\times4}$ является внешней матрицей преобразования камеры из системы координат камеры в глобальную систему координат [17], отобранные точки $p_{ij} = (x_{ij}, y_{ij}, z_{ij})$ (в однородных координатах) могут быть идентифицированы с изображением глубины $I = (z_{ij}^{\text{cam}})$, где $z_{ij}^{\text{cam}} = (KT^{-1}p_{ij})_3$ обозначает z-координату точки p_{ij} в системе координат камеры. Были созданы изображения со значениями расстояний до объекта путем вычисления $d = (d(p_{ij}))$ и записи пары I, d в качестве обучающего примера. В работе значение n_v равнялось 18, и набор данных дополнялся путем вращения и смещения сетки изображений в процессе генерации данных с сохранением

при этом той же ориентации оси z [28]. Таким образом была получена обучающая выборка на основе 68 полных трехмерных моделей в различных разрешениях.

6. Векторизация двухмерных изображений

Существует множество методов векторизации изображений и чертежей [30, 37, 12, 9, 3, 5, 2, 13, 15]. Один из наиболее широко используемых методов векторизации изображений - это Potrace, который находит контурные кривые [37]. Некоторые из работ используют Potrace в качестве этапа в своих алгоритмах [31, 22]. Другой набор работ основан на извлечении сети кривых и очистке топологии [12, 3, 32, 5, 4, 33, 19, 42, 20]. Работа [11] использует машинное обучение для извлечения высокоуровневых представлений из растровых чертежей, не стремясь точно воспроизвести геометрию примитивов. Работа [14] сосредоточена на повышении точности восстановления топологии. Алгоритм [13] рассматривает только простые данные и использует нейронную сеть для более точного соответствия геометрии сегментов кривых.

Каждый из методов решает различные версии проблемы векторизации и работает с разными типами входных и выходных данных. Некоторые алгоритмы умеют работать на шумных данных, а некоторые требуют на вход чистые; часть методов умеет обрабатывать цветные изображения, а часть - нет. Различные методы получают разные наборы математических примитивов в качестве выходных данных.

Хотя разные приложения предъявляют разные требования, в целом ``хорошая'' векторизация должна соответствовать следующим требованиям:

- хорошо аппроксимировать семантически или перцептивно входное изображение;
- не векторизовать шум и обрабатывать артефакты;
- минимизировать количество используемых примитивов, создавая компактное и легко редактируемое представление.

Предложенный автором данной диссертации метод [10] фокусируется на векторизации технических изображений. Разработанная система (см. рис. 4) принимает на вход растровый чертеж и создает набор графических примитивов, таких как отрезки линий и квадратичные кривые Безье, у которых есть ширина.

Метод состоит из следующих шагов:

1. Предварительная обработка расстрового изображений. На этом этапе предварительно обученная нейронная сеть удаляет шум, корректирует контраст и заполняет отсутствующие части (пропуски) на изображении.
2. Первоначальная оценка примитивов. Очищенное изображение из предыдущего шага разделяется на фрагменты, и для каждого фрагмента оцениваются начальные параметры примитивов с помощью нейронной сети.

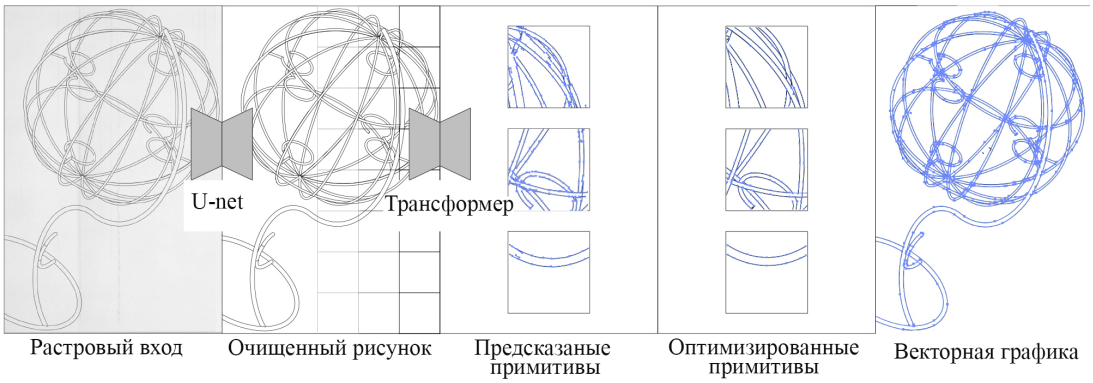


Рис. 4: Обзор представленного метода векторизации. На начальном этапе входное изображение очищается с помощью глубокой сверточной нейронной сети. Затем результат очистки разбивается на патчи (фрагменты), и для каждого патча оценивается размещение примитивов с помощью глубокой нейронной сети. Далее примитивы в каждом патче уточняются путем итеративной оптимизации. На финальном этапе патчи объединяются в одно векторное изображение.

	IoU,%	PSNR
[38]	49	15.7
Наша модель	**92**	**25.5**

Таблица 1: Сравнение результата работы метода очистки, предложенного в данной работе, с методом из работы [38].

3. Оптимизация параметров примитивов. Уточнение параметров примитивов происходит путем выравнивания их по очищенному растру.

4. Постобработка. Объединяются предсказания из всех фрагментов и удаляются лишние примитивы.

Далее рассмотрим каждый шаг более подробно.

6.1. Предварительная обработка изображений

Цель предварительной обработки состоит в очистке сырых входных данных с помощью устранения шума, заполнения пропущенных частей линий и перевода фона в белый цвет. Данную задачу можно рассматривать как задачу сегментации изображения на два класса, где один класс представляет собой фон, а другой - все остальное. Следуя идеям из работ [24, 38], в данной работе используется архитектура типа U-net [36], которая широко применяется в задачах сегментации. Модель обучается на синтетически сгенерированных данных и дообучается на реальных данных, описанных в разделе 5. В качестве функции потерь используется бинарная перекрестная энтропия. Сравнения с методом из [38] представлены в таблице 1, а пример работы методов представлен на рис. 5.

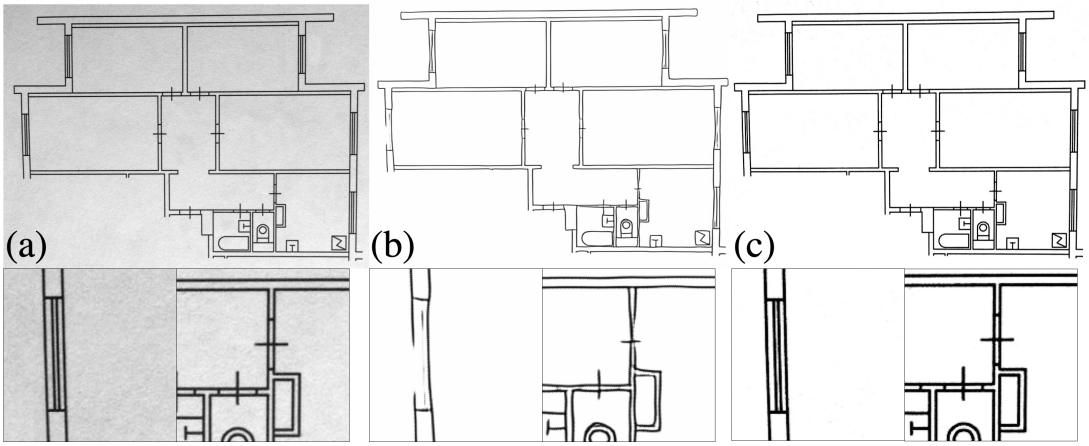


Рис. 5: Пример предварительной обработки изображения: (a) исходное сырое изображение, (b) результат метода из работы [38], (c) результат модели, предложенной автором данной работы.

6.2. Первоначальная оценка примитивов

Для начала очищенное растровое изображение, полученное из предыдущего этапа, разделяется на патчи. Далее параметры примитивов для каждого патча независимо инициализируются с помощью нейронной сети. Нейронная сеть состоит из энкодера на основе ResNet [18] и последовательности n_{dec} трансформерных блоков [39]. Каждый участок $I_p \in [0,1]^{64\times 64}$ кодируется с помощью сети энкодера $X^{\text{im}} = \text{ResNet}\,(I_p)$, а затем признаковые вложения примитивов X_i^{pr} декодируются с помощью трансформерных блоков (1)

$$X_i^{\text{pr}} = \text{Transformer}\left(X_{i-1}^{\text{pr}}, X^{\text{im}}\right) \in \mathbb{R}^{n_{\text{prim}}\times d_{\text{emb}}}, \qquad i = 1,\ldots,n_{\text{dec}}. \tag{1}$$

На выходе получается набор параметров n_{prim} размером d_{emb}. Максимальное количество примитивов задается размером (0)-го вложения $X_0^{\text{pr}} \in \mathbb{R}^{n_{\text{prim}}\times d_{\text{emb}}}$, инициализированного позиционным кодированием, как описано в [39].

Более 97% патчей в обучающей выборке содержат не более 10 примитивов. Исходя из этого было принято решение зафиксировать максимальное количество предсказываемых примитивов на уровне десяти и отфильтровывать лишние. Для получения координат контрольных точек, ширин примитивов $\Theta = \{\boldsymbol{\theta}_k = (x_{k,1}, y_{k,1}, \ldots, w_k)\}_{k=1}^{n_{\text{prim}}}$ и значений уверенности в том, что данный примитив существует $\boldsymbol{p} \in [0,1]^{n_{\text{prim}}}$, последнее полученное признаковое вложение передается в полносвязный блок. Если ширина или значение уверенности меньше, чем порог (как правило, выставляется стандартным и равным 0.5), то примитив отбрасывается.

Для предсказания параметров примитивов была обучена нейронная сеть с использованием функции потерь, оценивающей точность одновременного решения нескольких задач: бинарная энтропии для значения уверенности и взвешенной суммы отклонений от параметров (L_1)

и (L_2):

$$L\left(\boldsymbol{p}, \hat{\boldsymbol{p}}, \Theta, \hat{\Theta}\right) = \frac{1}{n_{\text{prim}}} \sum_{k=1}^{n_{\text{prim}}} \left(L_{\text{cls}}\left(p_k, \hat{p}_k\right) + L_{\text{loc}}\left(\boldsymbol{\theta}_k, \hat{\boldsymbol{\theta}}_k\right)\right), \tag{2}$$

$$L_{\text{cls}}\left(p_k, \hat{p}_k\right) = -\hat{p}_k \log p_k - (1 - \hat{p}_k) \log\left(1 - p_k\right), \tag{3}$$

$$L_{\text{loc}}\left(\boldsymbol{\theta}_k, \hat{\boldsymbol{\theta}}_k\right) = (1 - \lambda) \left\|\boldsymbol{\theta}_k - \hat{\boldsymbol{\theta}}_k\right\|_1 + \lambda \|\boldsymbol{\theta}_k - \hat{\boldsymbol{\theta}}_k\|_2^2. \tag{4}$$

Поскольку данная функция потерь не инвариантна относительно перестановок примитивов и их контрольных точек, конечные точки в примитивах и сами примитивы сортируются лексикографически.

6.3. Оптимизация параметров примитивов

Нейронная сеть для предсказания примитивов минимизирует среднеквадратическое отклонение, но даже при небольшом среднем отклонении отдельные оценки могут быть неточными. Для уточнения параметров примитивов был разработан функционал (5), который выравнивает примитивы по растровому изображению

$$\Theta^{\text{ref}} = \underset{\Theta}{\operatorname{argmin}}\, E\left(\Theta, I_p\right). \tag{5}$$

Оптимизационный функционал определяется как сумма трех членов для каждого примитива

$$E\left(\Theta^{\text{pos}}, \Theta^{\text{size}}, I_p\right) = \sum_{k=1}^{n_{\text{prim}}} E_k^{\text{size}} + E_k^{\text{pos}} + E_k^{\text{rdn}}, \tag{6}$$

где $\Theta^{\text{pos}} = \left\{\boldsymbol{\theta}_k^{\text{pos}}\right\}_{k=1}^{n_{\text{prim}}}$ - это параметры положения примитива, $\Theta^{\text{size}} = \left\{\boldsymbol{\theta}_k^{\text{size}}\right\}_{k=1}^{n_{\text{prim}}}$ - параметры размера, и $\boldsymbol{\theta}_k = \left(\boldsymbol{\theta}_k^{\text{pos}}, \boldsymbol{\theta}_k^{\text{size}}\right)$.

Положения линии заданы координатами ее средней точки и углом наклона, а размер - по ее длине и ширине. В случае дуги определяется средняя точка на пересечении кривой и биссектрисы угла между отрезками, соединяющими среднюю контрольную точку и конечные точки. Также используются длины этих сегментов и углы наклона, соединяющих “среднюю точку” с конечными точками.

Взаимодействие зарядов. Различные части функционала основываются на энергии взаимодействия единичных точечных зарядов $\boldsymbol{r}_1, \boldsymbol{r}_2$, определенных как сумма потенциалов ближнего и дальнего действия

$$\varphi\left(\boldsymbol{r}_1, \boldsymbol{r}_2\right) = e^{-\frac{\|\boldsymbol{r}_1 - \boldsymbol{r}_2\|^2}{R_{\text{c}}^2}} + \lambda_{\text{f}} e^{-\frac{\|\boldsymbol{r}_1 - \boldsymbol{r}_2\|^2}{R_{\text{f}}^2}}, \tag{7}$$

параметры R_{c}, R_{f}, λ_{f} в которых выбираются экспериментально. Энергия взаимодействия равномерно позитивно заряжённой области k-го примитива Ω_k и сетки точечных зарядов $\boldsymbol{q} = \left\{q_i\right\}_{i=1}^{n_{\text{pix}}}$ в центрах пикселей $\boldsymbol{r}_i$ определяется следующим уравнением:

$$E_k\left(\boldsymbol{q}\right) = \sum_{i=1}^{n_{\text{pix}}} q_i \iint\limits_{\Omega_k} \varphi\left(\boldsymbol{r}, \boldsymbol{r}_i\right) dr^2 \tag{8}$$

В случае кривых эта энергия приближается суммой интегралов по сегментам полилинии, выпрямляющей кривую. В предложенном функционале используется три разные сетки зарядов, закодированных как векторы длиной n_{pix}: $\hat{\boldsymbol{q}}$ представляет растровое изображение с величинами зарядов, установленными в интенсивности пикселей, $\boldsymbol{q}_k$ представляет рендеринг k-го примитива с его текущими значениями параметров, и $\boldsymbol{q}$ представляет рендеринг всех примитивов в патче. Наборы зарядов $\boldsymbol{q}_k$ и $\boldsymbol{q}$ обновляются на каждой итерации.

Рассмотрим подробнее формулу (6), которая включает в себя три компоненты.

Первая компонента отвечает за увеличение примитива для покрытия закрашенных пикселей и за его уменьшение, если покрыты незаполненные пиксели (при фиксированном положении примитива):

$$E_k^{\text{size}} = E_k\left([\boldsymbol{q} - \hat{\boldsymbol{q}}] \odot \boldsymbol{c}_k + \boldsymbol{q}_k \odot [\mathbf{1} - \boldsymbol{c}_k]\right). \tag{9}$$

Вес $c_{k,i} \in \{0, 1\}$ обеспечивает покрытие непрерывной растровой области, следуя форме и ориентации примитива.

Вторая компонента отвечает за выравнивание примитивов фиксированного размера

$$E_k^{\text{pos}} = E_k\left([\boldsymbol{q} - \boldsymbol{q}_k - \hat{\boldsymbol{q}}] \odot [\mathbf{1} + 3\boldsymbol{c}_k]\right). \tag{10}$$

Последняя компонента отвечает за схлопывание перекрывающихся *коллинеарных* примитивов; для этого члена используется $\lambda_{\text{f}} = 0$:

$$E_k^{\text{rdn}} = E_k\left(\boldsymbol{q}_k^{\text{rdn}}\right),\ q_{k,i}^{\text{rdn}} = \exp\left(-\left[|\boldsymbol{l}_{k,i} \cdot \boldsymbol{m}_{k,i}| - 1\right]^2 \beta\right) \|\boldsymbol{m}_{k,i}\|, \tag{11}$$

где $\boldsymbol{l}_{k,i}$ - направление примитива в его ближайшей точке к i-му пикселю, $\boldsymbol{m}_{k,i} = \sum_{j \neq k} \boldsymbol{l}_{j,i} q_{j,i}$ - сумма направлений всех других примитивов, взвешенных относительно их “присутствия”, и $\beta = (\cos 15^\circ - 1)^{-2}$ (выбирается экспериментально).

Можно получить приблизительное решение для формулы (5), рассматривая каждый энергетический член E_k^{pos}, E_k^{size}, E_k^{rdn} как зависящий только от параметров k-го примитива. Это позволяет эффективно вычислять градиент для предложенного функционала, поскольку необходимо дифференцировать каждый член лишь по небольшому числу параметров.

6.4. Результаты работы алгоритма

Для оценки работы алгоритма было использовано 15 тестовых изображений из набора данных DLD. Количественные результаты этой оценки представлены в таблице 2, а качественные результаты - на рис. 6. Дополнительно было проведено сравнение на 40 изображениях из набора данных PFP и 50 изображениях из ABC с разрешением $\sim 2000 \times 3000$. Количественные результаты на этих данных также представлены в таблице 2, а качественные результаты - на рис. 7 и 8.

Как было указано ранее, хорошая векторизация должна реализовываться малым числом примитивов, но при этом хорошо аппроксимировать входное изображение. По параметрам качества аппроксимации на данных PFP предложенная в данной работе система превосходит

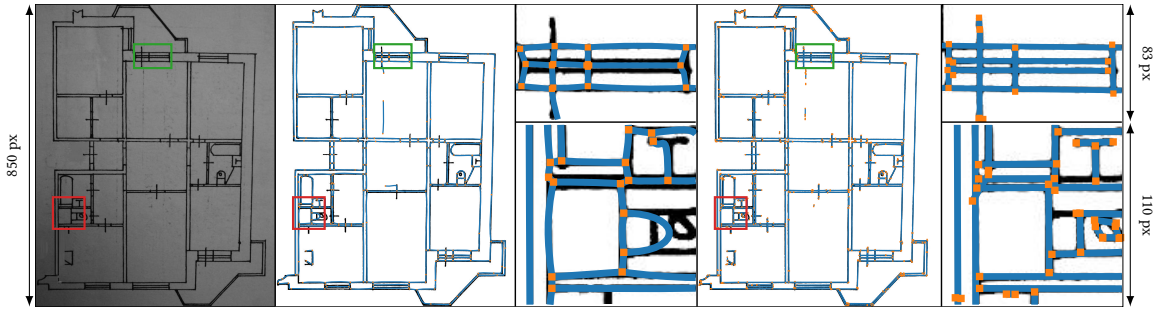


Рис. 6: Качественное сравнение на скане реального плана помещения. IoU / #P с лучшим результатом выделены жирным шрифтом. Примитивы показаны синим цветом с оранжевыми конечными точками поверх очищенного растрового изображения.

другие методы, и единственно уступает по количеству примитивов модели FvS. В случае данных ABC несмотря на то, что модель PVF превосходит предложенную в данной работе систему векторизации по метрике IoU, результат ее работы содержит гораздо больше примитивов.

	PFP				ABC				DLD	
	IoU,%	HD, px	d_M, px	#P	IoU,%	HD, px	d_M, px	#P	IoU,%	#P
FvS	31	381	2.8	696	65	38	1.7	63		
CHD	22	214	2.1	1214	60	9	1	109	47	329
PVF	60	204	1.5	38k	89	17	0.7	7818		
Наш	86/88	25	0.2	1331	77/77	19	0.6	97	79/82	452

Таблица 2: Количественные результаты векторизации на данных PFP, ABC и DLD с использованием различных методов.

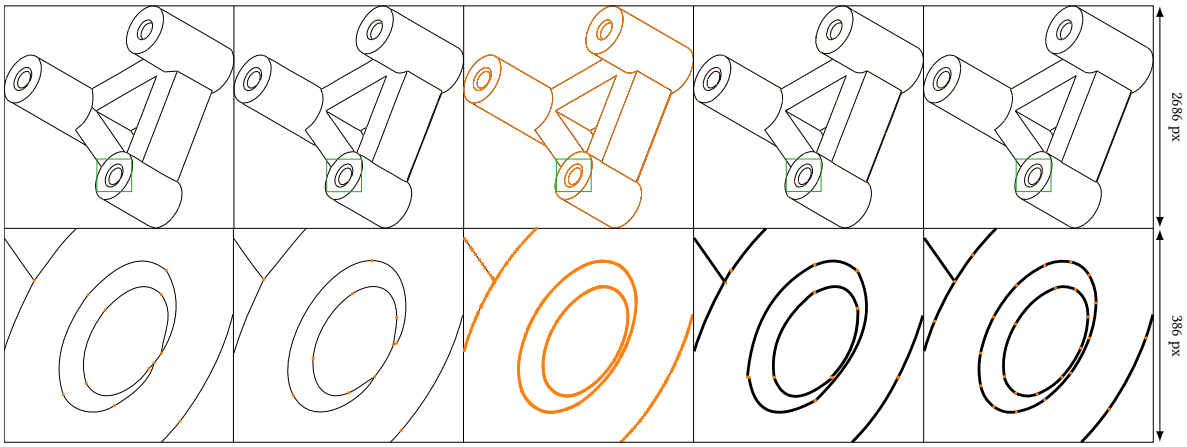
Заключение

В данном разделе был рассмотрен новый метод векторизации технических изображений с использованием нейронных сетей. Он состоит из очистки входного изображения с помощью нейронной сети на основе архитектуры U-Net, первоначальной оценки примитивов с использованием связки нейронных сетей ResNet и Transformers, оптимизации параметров примитивов и постобработки.

Предложенная работа значительно улучшает качество полученных векторных изображений технических сканов и уменьшает количество гиперпараметров благодаря использованию методов глубокого обучения и нового подхода к оптимизации.

FvS	CHD	PVF	Наш метод	Эталон,
29% / 415px	21% / 215px	64% / 140px	**89%** / **28px**	#P 1634
4.2px / **615**	1.9px / 1192	0.9px / 35k	**0.2px** / 1286	

Рис. 7: Качественное сравнение на изображении из данных PFP. Значения IoU / HD / d_M / #P с лучшим результатом выделены жирным шрифтом. Конечные точки примитивов показаны оранжевым цветом.



FvS	CHD	PVF	Наш метод	Эталон,
67%/ 32px	67%/ 7px	**95%**/ **4px**	86%/ 5px	#P 139
1.1px/ 79	1.0px/ 108	**0.2px**/ 9.5k	0.4px/ 139	

Рис. 8: Качественное сравнение на изображении из данных ABC. Значения IoU / HD / d_M / #P с лучшим результатом выделены жирным шрифтом. Конечные точки примитивов показаны оранжевым цветом.

7. Генерация синтетических данных высокого качества с описаниями трехмерных объектов

Как уже упоминалось ранее, получение реальных данных является трудоемкой задачей, а для обучения нейронной сети требуется большое количество данных (см. раздел 5). Нейронная сеть, обученная на синтетических данных, не всегда показывает хорошие результаты на реальных данных из-за доменного сдвига. Решить проблему доменного сдвига можно с помощью дообучения на реальных данных, делая синтетические данные более реалистичными или приводя реальные данные к виду, схожему с синтетическими во время инференса.

Проблема сбора реальных данных для векторизации еще более сложна в трехмерном случае. Дообучение на реальных данных все еще требует большое количество таких данных, соответствующих задаче. Альтернативный способ обучения на синтетических данных, приближенных к реальным, позволяет использовать не парный датасет синтетических и реальных данных. Собрать такой датасет можно с помощью открытых источников из интернета. В результате можно обучить модель, который будет делать синтетические данные более реальными или наоборот. В связи с этим было решено изучить вопрос перевода одного распределения в другое в трехмерном пространстве на стандартной задаче генерации трехмерных облаков точек из шума.

Генерация трехмерных данных является сложной задачей. Недавней тенденцией является применение методов генерации, основанных на данных, таких как глубокие генеративные модели [1, 25, 6]. Большинство таких методов работают только с грубой трехмерной геометрией (низкого разрешения), поскольку трехмерные формы с высоким разрешением требуют значительных вычислительных затрат для обработки и сложны для обучения. Для генерации данных высокого разрешения в работе была разработана новая глубокая каскадная (многомасштабная) модель GAN на основе лапласовой пирамиды в латентном пространстве (LSLP-GAN) [10].

Предложенная модель (схематически представлена на рис. 9) для обучения с использованием трехмерных облаков точек основана на работах по латентным GAN-ам [1] и лапласовским GAN-ам [8]. В [1] было предложено обучать GAN-ы не в пространстве облаков точек, а на латентных кодах, полученных с помощью автоэнкодера (автокодировщика). Лапласовские GAN-ы [8] решают проблему сложности обучения объектов в высоком разрешении с помощью каскадного синтеза изображений с использованием серий генеративных сетей $G_0, \ldots, G_n$, где каждая сеть G_k учится генерировать высокочастотное остаточное изображение $r_k = G_k(U(I_{k+1}), z_k)$, где I_k - восстановленное изображение, полученное от G_{k-1}. Таким образом, изображение на этапе k представляется следующим образом:

$$I_k = U(I_{k+1}) + G_k(U(I_{k+1}), z_k), \tag{12}$$

где $U(\cdot)$ - оператор увеличения разрешения, а z_k - вектор шума.

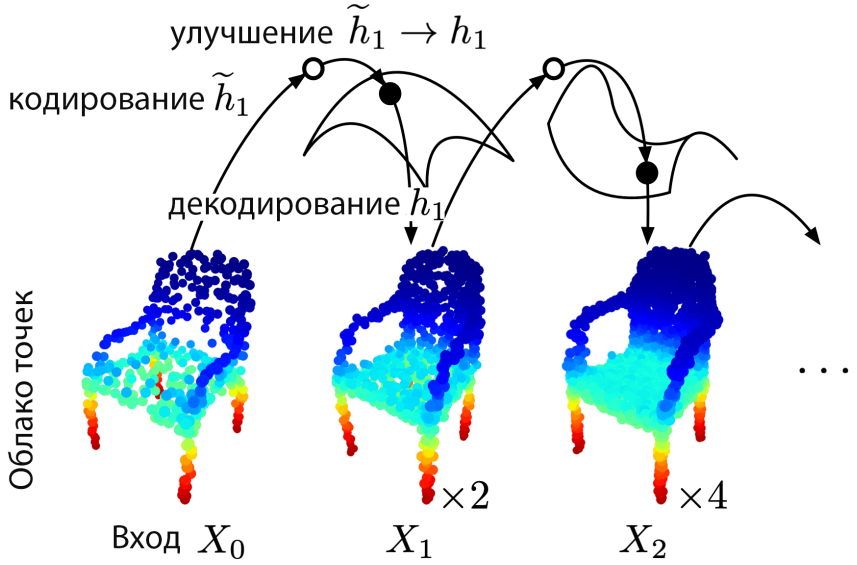


Рис. 9: Предлагаемая многомасштабная (многоэтапная) генеративно-состязательная нейросетевая модель (GAN) над латентным пространством.

Так как моделирование трехмерных облаков точек с высокой детализацией является сложной задачей из-за высокой размерности, было предложено использовать идею из работы [8] и начать с модели с низкой детализацией, которую при этом легко обучить, и разбить задачу моделирования на последовательность более простых этапов, каждый из которых направлен на постепенное увеличение детализации.

Из пространства трехмерных форм $\{\mathbb{R}^{n_k \times 3}\}_{k=1}^{K}$ с помощью обученных автоэнкодеров точек $\{(f_k, g_k)\}_{k=1}^{K}$ были сконструированы соответствующие пространства скрытых кодов $\{\mathbb{R}^{d_k}\}_{k=1}^{K}$. Следует отметить, что автоэнкодер (f_k, g_k) обучается с использованием разрешения n_k трехмерных облаков точек, которое увеличивается по мере роста k. После обучения автоэнкодеров были зафиксированы их параметры и извлечены скрытые коды для облаков точек в каждом из пространств трехмерных форм.

Лапласовская пирамида в пространстве латентных кодов

На вход в систему подается облако точек $X_{k-1} \in \mathbb{R}^{n_{k-1} \times 3}$. Целью является переход от X_{k-1} к X_k, то есть увеличение разрешения с n_{k-1} до $n_k = 2n_{k-1}$, с помощью генерации дополнительных точек на поверхности формы (см. рис. 10).

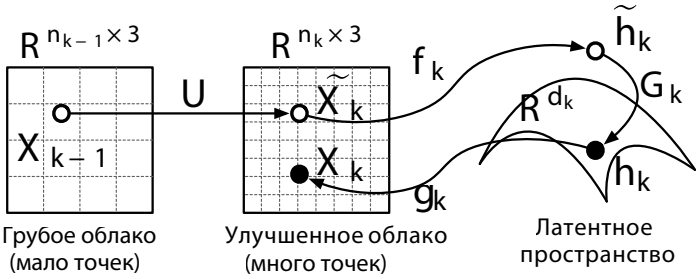


Рис. 10: Подробная схема работы предложенной лапласовской пирамиды в латентном пространстве.

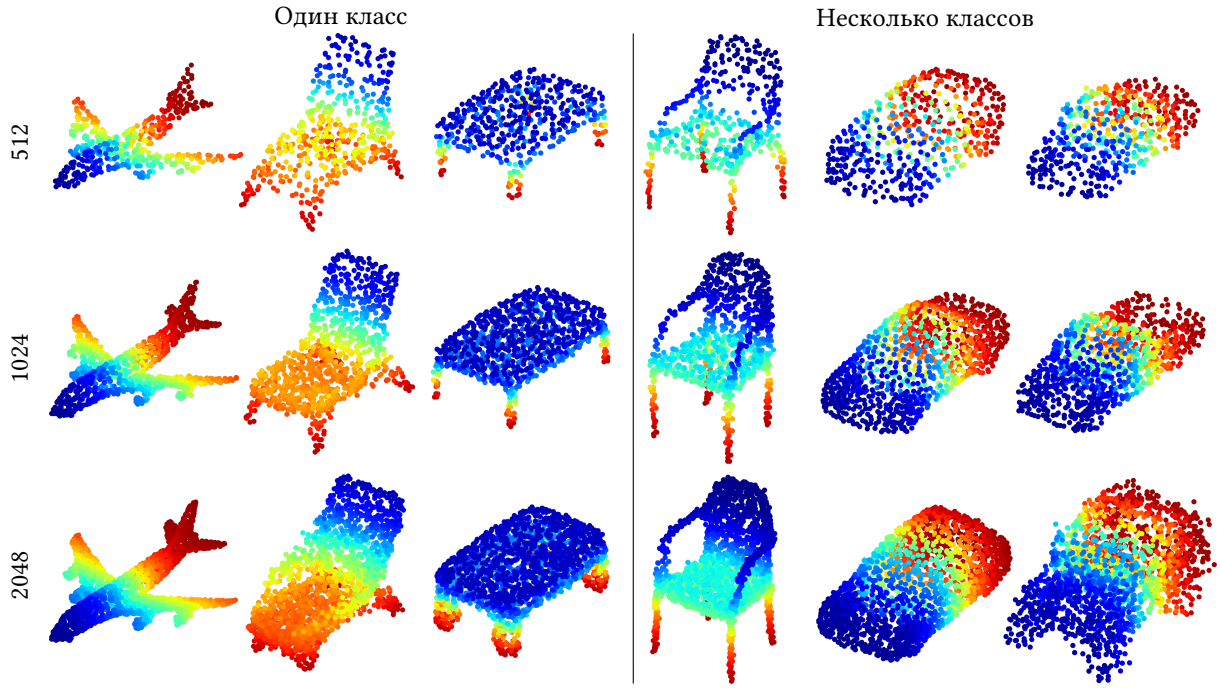


Рис. 11: Примеры форм, синтезированных с использованием нашей модели LSLP-GAN. *В левой части:* самолеты, стулья и столы, синтезированные с использованием наших моделей для одного класса. *В правой части:* образцы трехмерных форм, синтезированные с использованием нашей мультиклассовой модели.

Обработка входного облака точек начинается с помощью простого оператора интерполяции $U(\cdot)$. Процесс получения грубого облака точек $\widetilde{X}_k = U(X_{k-1})$ происходил следующим образом: для каждой точки $x \in X_{k-1}$ был создан новый экземпляр $\widetilde{x} = \frac{1}{m}\sum_{i \in \text{NN}(x)} x_i$, где $\text{NN}(x)$ - это набор из m ближайших соседей x в Евклидовом пространстве X_{k-1} (в работе было использовано $m = 7$ соседей, включая x), который был добавлен к облаку точек. Описанная процедура представляет собой простую линейную интерполяцию и формирует n_k точек, находящихся вблизи реальной поверхности.

Однако вычисленное облако точек $\widetilde{X}_k$ обычно содержит искаженные точки, и оно было рассмотрено только как приближение к нашей желаемой форме X_k. Грубое облако точек $\widetilde{X}_k$ было отображено с помощью f_k в скрытый код $\widetilde{h}_k = f_k(\widetilde{X}_k)$, который, как предполагается, отклоняется на небольшую величину от многообразия скрытых представлений из-за ошибки интерполяции в $\widetilde{X}_k$. Чтобы скомпенсировать эту погрешность в скрытом пространстве, вычисляется добавочная поправка r_k для $\widetilde{h}_k$ с помощью генеративной сети G_k, что приводит к скорректированному коду $h_k = \widetilde{h}_k + r_k = \widetilde{h}_k + G_k(\widetilde{h}_k, z_k)$. Декодирование h_k происходит с помощью g_k, и таким образом получается уточненное облако $X_k = g_k(h_k)$ с разрешением n_k.

Полная процедура в пространстве скрытых представлений представляет собой серию преобразований:

$$h_k = f_k(U(X_{k-1})) + G_k(f_k(U(X_{k-1})), z_k), \tag{13}$$

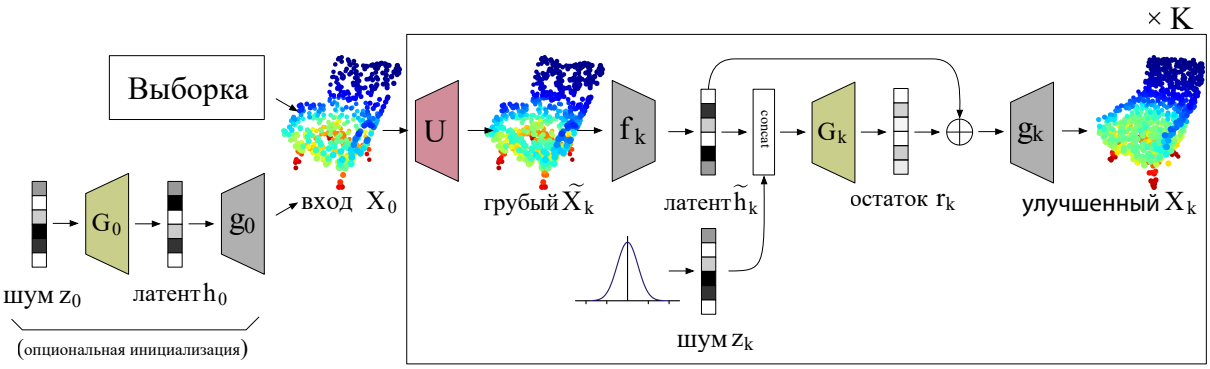


Рис. 12: Полная архитектура модели LSLP-GAN. Сеть принимает или генерирует начальное облако точек X_0 и обрабатывает его с помощью серии из K обучаемых шагов. Каждый шаг (1) увеличивает разрешение входа с помощью необучаемого оператора U, (2) кодирует увеличенную версию в скрытом пространстве с помощью f_k, (3) выполняет коррекцию скрытого кода с помощью условной GAN G_k, и (4) декодирует скорректированный скрытый код с использованием g_k.

которая является аналогом лапласовской пирамиды в скрытом пространстве (12). Данная процедура была названа Latent-Space Laplacian Pyramid GAN (LSLP-GAN).

Обучения GAN-ов на латентных кодах

На латентных кодах была обучена серия генеративных состязательных нейросетей $\{(G_k, D_k)\}_{k=1}^{K}$. Генератор G_k синтезирует ``остатки'' r_k в скрытом пространстве в зависимости от входа $\widetilde{h}_k$, а дискриминатор D_k различает реальные скрытые коды $h_k \in \mathbb{R}^{d_k}$ и синтетические $\widetilde{h}_k + G_k(\widetilde{h}_k, z_k)$. Следует отметить, что каждая (кроме первой) генеративная состязательная сеть принимает грубый скрытый код $\widetilde{h}_k$ и может рассматриваться как условная генеративная состязательная сеть (CGAN) [29].

Пример сгенерированных объектов можно увидеть на рис. 11.

Заключение

В данном разделе был представлен новый метод для многоуровневой генерации трехмерных облаков точек с помощью генеративно-состязательной нейросетевой модели. Генеративные сети работают в латентном пространстве автокодировщика, генерируя добавки, постепенно увеличивая количество и расположение точек выходной модели.

Благодаря многоуровневому подходу данный метод может быть применен не только для генерации объектов из шума, но и для улучшения входного облака точек. В частности, с помощью данного метода можно генерировать синтетические данные, близкие к реальным, для уменьшения доменного сдвига во время обучения. Также предложенную модель можно использовать как отдельный модуль, позволяющий улучшать облако точек, полученное с помощью изображения глубины отсканированных объектов, для более точной векторизации изоб-

ражений глубины. Эффективность этих подходов для улучшения векторизации является темой для будущих исследований.

Дополнительно, еще одним применением метода может служить генерация векторной графики из шума. Такая генерация происходит в два этапа: сначала генерируется облако точек, а затем применяется метод векторизации.

8. Векторизация трехмерных объектов

Векторизация трёхмерных объектов по аналогии с двухмерными подразумевает представление объектов и связей между ними с помощью математических примитивов. Наиболее распространёнными примерами являются Computer Aided Design (CAD), скелет объекта и параметрические кривые.

В данном разделе будет рассмотрен способ извлечения параметрических кривых из отсканированных трёхмерных объектов. На вход подаются изображения глубины объекта из разных точек, а на выходе получаются трёхмерные сплайны, описывающие входной объект. Метод состоит из двух частей: получение оценочного поля расстояния геометрических особенностей и извлечение параметрических кривых на основе этой информации.

Для получения поля расстояния использовалась нейронная сеть с архитектурой U-Net на основе ResNet-152, решающая задачу регрессии, а именно предсказывающая усечённое поле расстояний до особых кривых на карте глубины. Далее с помощью предложенного нового подхода эти предсказания объединялись и получались оценочные поля расстояний на полных трёхмерных моделях [28].

Извлечение параметрических кривых использует информацию оцененных полей расстояний, полученных ранее с помощью Deep Estimators of Features (DEFs) [28] и включает в себя локальный классификатор для обнаружения угловых вершин, построенную графовую структуры и аппроксимацию сплайнами.

Алгоритм основан на методе, описанном в статье [27]. Общая структура подхода сохранена, но были внесены значительные улучшения, которые автоматизируют работу алгоритма, увеличивают стабильность и улучшают качество полученных результатов. Данные улучшения были достигнуты с помощью следующих изменений:

- использование другого критерия обнаружения углов и концов кривых в (14), (17);
- более надежный этап построения полилиний на основе k ближайших соседей и другого функционала оптимизации в (19);
- добавление метода постобработки в (21).

На рис. 13 представлена визуальная иллюстрация различий между двумя алгоритмами. Как видно на данном рисунке предложенный алгоритм лучше справляется со сложными трехмерными объектами и менее склонен генерировать выбросы.

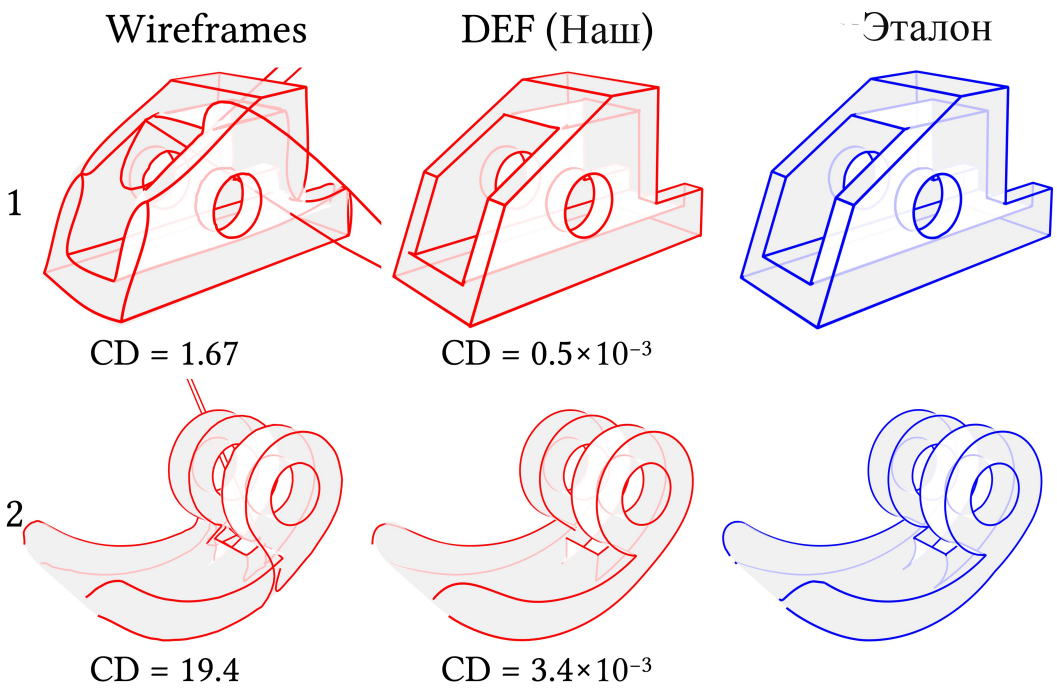


Рис. 13: Предложенный метод улучшает обнаружение углов (строка 1) и способен разрешать сложные кривые (строка 2), в то время как в результате работы *Wireframes* периодически получаются кривые, имеющие крайне большие отклонения от эталона.

8.1. Описание разработанного метода получения трехмерных параметрических кривых

Предложенный метод включает в себя несколько этапов:

- Инициализация,
- Обнаружение углов,
- Сегментация кривых и углов,
- Извлечение графа кривых,
- Аппроксимация сплайнами и оптимизация,
- Постобработка представлений, полученных на основе сплайнов.

На рис. 14 представлена визуальная схема работы нашего алгоритма. Ниже каждый из этих этапов будет рассмотрен более подробно.

8.1.1. Инициализация

На данном этапе из изначального облака точек P выбирается подмножество точек, у которых оцененное расстояние до особых кривых $\widehat{d}$ меньше порога d_{sharp}. Дальнейшее уменьшение

числа точек было получено с помощью метода семплирования диска Пуассона [7], оставляющая 10% точек. В результате было получено подмножество точек P_{sharp}.

8.1.2. Обнаружение углов

На данном этапе были выбраны опорные точки из P_{sharp} методом самой удаленной точки. В качестве опорных точек было использовано 20% точек из P_{sharp}. Затем были построены наборы B_i, содержащие точки, находящиеся в перекрывающихся евклидовых шарах радиусом R_{corner}, которые расположены в центре опорных точек и охватывают P_{sharp}.

Каждый из этих локальных наборов B_i был аппроксимирован его эллипсоидной формой с помощью вычисления главных компонент (PCA) на точках набора и получения вектора дисперсий $(\lambda_1, \lambda_2, \lambda_3)$, такого что $\lambda_1 \leqslant \lambda_2 \leqslant \lambda_3$ и $\sum_{k=1}^{3} \lambda_k = 1$, описывающего длины осей эллипсоида. Для каждого конкретного набора B_i полученные векторы были использованы для вычисления нормированной агрегации квадратичного расстояния с использованием следующего выражения:

$$\Lambda_i = \sum_{k=1}^{3} \sum_{j \in \mathcal{N}_i} \left(\frac{\lambda_k^i - \lambda_k^j}{\delta_{ij}} \right)^2, \tag{14}$$

где $\mathcal{N}_i$ - это набор индексов наборов B_j, ближайших к набору B_i, а δ_{ij} - евклидово расстояние между опорными точками наборов B_i и B_j.

Далее решается задача принадлежности локального набора B_i к кластерам точек, принадлежащим угловым точкам. Это делается с помощью сравнения Λ_i с порогом T_{variance}, относя B_i либо к угловому набору точек, либо к кривым:

$$\begin{aligned} \mathcal{B}_{\text{corner}} &= \{B_i \,|\, \Lambda_i > T_{\text{variance}}\}, \\ \mathcal{B}_{\text{curve}} &= \{B_i \,|\, \Lambda_i \leqslant T_{\text{variance}}\}. \end{aligned} \tag{15}$$

С помощью изменений $\mathcal{N}_i$, T_{variance} и R_{corner} в небольших диапазонах было получено 60 комбинаций классификаций. Затем, на основе доли угловых классификаций в конкретном наборе B_i вычислялась вероятность того, что данный набор является углом.

Для распространения классификации на все точки был применен метод k ближайших соседей с $k = 50$. Тем самым для каждой точки было получено значение $0 \leqslant w(p) \leqslant 1$.

Определение угловых точек производится с помощью применения порогового значения T_{corner}:

$$P_{\text{corner}} = \{p \in P_{\text{sharp}} : w(p) > T_{\text{corner}}\}.$$

8.1.3. Сегментация кривых и углов

Для сегментации кривых были использованы два набора точек: P_{corner}, содержащий угловые точки, и $P_{\text{curve}} = P_{\text{sharp}} \setminus P_{\text{corner}}$, содержащий точки, которые не являются угловыми. Оба этих набора были обработаны, чтобы выделить кластеры, которые определяют отдельные углы и кривые соответственно.

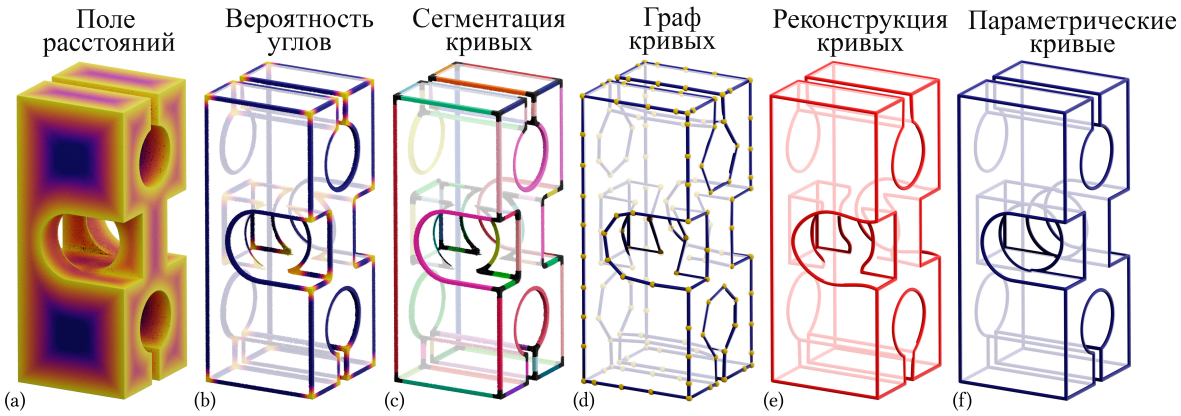


Рис. 14: На начальном этапе (a) применяется пороговая обработка расстояний для получения подмножества P_{sharp}, которое затем используется для оценки вероятностей углов (b) и построения сегментации кривых (черные кластеры соответствуют обнаруженным окрестностям углов) (c). Далее обнаруженные углы и кривые позволяют построить и оптимизировать граф кривых (d). На конечном этапе граф кривых преобразуется в набор параметрических кривых (e), которые отражают геометрию исходной формы (f).

Для сегментации точек, принадлежащих отдельным кривым, был построен плотный граф с использованием метода k-ближайших соседей (kNN). Все точки в наборе P_{curve}, которые находятся на расстоянии r (16), были соединены друг с другом . Полученный таким образом граф был разбит на связные компоненты для получения отдельных кластеров точек, соответствующих различным кривым:

$$\underbrace{r}_{\substack{\text{расстояние}\\ \text{семплирования}}} \times \underbrace{n}_{\substack{\text{кол. примеров}\\ \text{на атрибут}}} = \underbrace{l}_{\substack{\text{характеристический}\\ \text{пространственный размер}}} \times \underbrace{s}_{\substack{\text{множитель}\\ \text{масштаба}}}, \tag{16}$$

Связаная компонента определяет одну из n_{curve} кривых, и все они составляют набор кластеров точек, соответствующих каждой кривой:

$$\mathcal{P}_{\text{curve}} = \left\{ P_c \subseteq P_{\text{curve}} \,|\, \forall p \in P_c \, \exists q \in P_c, p \neq q : \|p - q\| \leqslant r \right\}_{c=1}^{n_{\text{curve}}}.$$

Для угловых точек P_{corner} процедура аналогична. Извлекаются кластеры углов $\mathcal{P}_{\text{corner}}$ путем разделения связных компонент обнаруженных угловых наборов точек.

8.1.4. Извлечение графа кривых

Из полученной на предыдущем шаге сегментации строится граф кривых, соответствующий P_{sharp}. Дальнейшая процедура состоит из следующих шагов:

- обнаружение конечных точек для каждой кривой, на основе которого кривые определяются как открытые или замкнутые;
- приближение каждой кривой кратчайшей полилинией;

- соединение подогнанных полилиний, углов и конечных точек в полный граф кривой формы;
- уточнение положения конечных точек и углов.

Рассмотрим подробнее каждый из этих пунктов.

Обнаружение конечных точек для каждой кривой. Для обнаружения конечных точек сегментированного кластера кривой P_c строится детектор конечных точек на основе окрестности, аналогичный детектору углов. Затем строятся Евклидовые окрестности E_i с радиусом R_{endpoint}, сосредоточенные в якорных точках p_{ai}, выбранных из P_c, и вычисляется их аппроксимация прямой путем вычисления PCA на точках в E_i и уменьшения размерности до одной главной компоненты. Далее каждая точка $p \in E_i$ параметризуется одной координатой $t(p)$, полученной из PCA. Для определения конечных точек кривой вычисляется доля точек $p \in E_i$, у которых параметрические координаты $t(p)$ больше или меньше параметрической координаты t_{ai} якорной точки p_{ai}:

$$V_i = \Big| \frac{1}{|E_i|} \sum_{p \in E_i} \text{sign}(t(p) - t_{ai}) \Big|, \tag{17}$$

где p_{ai} является конечной точкой в случае, если V_i больше порогового значения T_{endpoint}. $V_i = 0$ соответствует полностью симметричному случаю, в то время как $V_i = 1$ указывает на сильное преобладание точек с одной стороны якорной точки. Если для кластера кривой P_c существует только одна такая якорная точка, выбирается якорь p_{ai} со вторым наибольшим значением V_i в качестве второй конечной точки; в случае обнаружения более двух конечных точек выбираются две самые удаленные точки; если конечные точки не обнаружены, кривая считается замкнутой.

Приближение каждой кривой кратчайшей полилинией. Для открытой кривой создается граф методом k ближайших соседей путем соединения всех якорных точек кривой p_{ai} из P_c, находящихся на расстоянии, не превышающем двойное среднее расстояние выборки друг от друга. Полилиния инициализируется с помощью нахождения кратчайшего пути в таком графе между обнаруженными конечными точками с использованием алгоритма Дейкстры.

В случае замкнутой кривой выбираются три точки из кластера с наибольшим удалением и соединяются. Далее остальные точки полилинии находятся с помощью стратегии разделения. Кандидаты на разделение определяются путем вычисления p_{split}:

$$p_{\text{split}} = \arg\max_{p_i \in P_c} \left| \widehat{d_i} - \|p_i - \min_l \pi^l(p_i)\| \right| \tag{18}$$

для точек p_i из текущего кластера кривой $P_c \in \mathcal{P}_{\text{curve}}$, где $2\min_l \pi^l(p_i)$ - это проекция p_i на ближайший сегмент полилинии l. Для продолжения разделения сравнивается абсолютная разница между оценками расстояния до особых кривых $\widehat{d_i}$ и фактическими расстояниями $|p_i - \pi^l(p_i)|$ с пороговым значением T_{split}; для кандидатов p_{split}, превышающих это значение, мы разделяем полилинию, присваивая p_{split} новым вершинам полилиний и разделяя соответствующий сегмент на два сегмента.

Соединение подогнанных полилиний, углов и конечных точек в полный граф. Обнаруженные конечные точки открытых кривых заменяются соответствующими ближайшими центрами угловых кластеров. Таким образом получается конечный граф кривых $G(q, e)$, определенный позициями узлов q и связями e между ними.

Уточнение положения конечных точек и углов. Для оптимизации позиций узлов используется формула (19):

$$\min_{q} \Big(\frac{1}{|P_{\text{sharp}}|} \sum_{p \in P_{\text{sharp}}} |\widehat{d}(p) - \big|p - \pi^{G(q,e)}(p)\big|| - \sum_{\overline{q} \in l[G(q,e)]} \cos \overline{q} \Big), \tag{19}$$

где $\pi^{G}(p)$ - проекция точки p на ближайшее ребро в графе кривых G, и $\sum_{\overline{q} \in l[G(q,e)]} \cos \overline{q}$ - сумма косинусов углов между двумя последовательными ребрами, связанными с узлом $\overline{q}$, вычисленная только для набора узлов $l[G(q, e)]$, имеющих ровно два инцидентных ребра. Второй член оценивает собой жесткость полилиний и позволяет избежать острых углов между ребрами. Оптимизация помогает определить позиции узлов графа более точно, особенно на пересечениях нескольких кривых, а член жесткости делает сегменты полилиний более прямыми. По завершении этого шага конечные позиции углов определяются как координаты узлов графа с более чем двумя инцидентными сегментами.

8.1.5. Аппроксимация сплайнами и оптимизация

Для аппроксимации сплайнами необходимо получить согласованную параметризацию каждой характеристической кривой. Это делается путем разбиения графа кривой на кратчайшие пути между узлами графа со степенью, не равной 2, при этом каждый путь служит прокси для кривой, определяющей координаты параметров точек вдоль характеристической кривой. Для пути g, представленного в виде последовательности узлов графа $q_g = \{q_i\}_{i=1}^{|g|}$, получается набор ближайших точек $P_g \in P_{\text{sharp}}$, затем вычисляются проекции $\pi^g(p_i)$, $p_i \in P_g$ и получаются значения параметров $u_g = \{u_i\}_{i=1}^{|P_g|}$ как накопленная сумма норм $\pi^g(p_i)$ вдоль пути g. Одновременно вычисляются узлы t_g как равномерно распределенные параметры; количество узлов определяется как $\max\left(5, \frac{|g|}{2}\right)$.

Аппроксимация сплайном s_g пути g приводит к набору контрольных точек c_s, определяющих точную форму кривой сплайна. После аппроксимации сплайном происходит оценка точки $P_s(c_s) = \gamma(u_g, P_g, t_g, c_s)$ на кривой сплайна s_g. Эти точки, в идеале, должны быть настолько же удалены от облака точек P_g, насколько предполагает поле расстояний $\widehat{d}$. Чтобы обеспечить данное свойство, контрольные точки оптимизируются для формирования сплайна по значениям расстояния:

$$\min_{c} \sum_{i=1}^{|P_g|} \Big(\widehat{d}_i - \|p_i - \gamma(u_i, p_i, t_i, c)\| \Big)^2, \tag{20}$$

где $p_i \in P_g$, $\widehat{d}_i$ - соответствующее значение расстояния, а $\gamma(u_i, p_i, t_i, c)$ - точка, соответствующая p_i, оцененная на сплайне s_g. Кроме того, накладываются ограничения на конечные точки сплайна, чтобы они соответствовали конечным точкам полилинии.

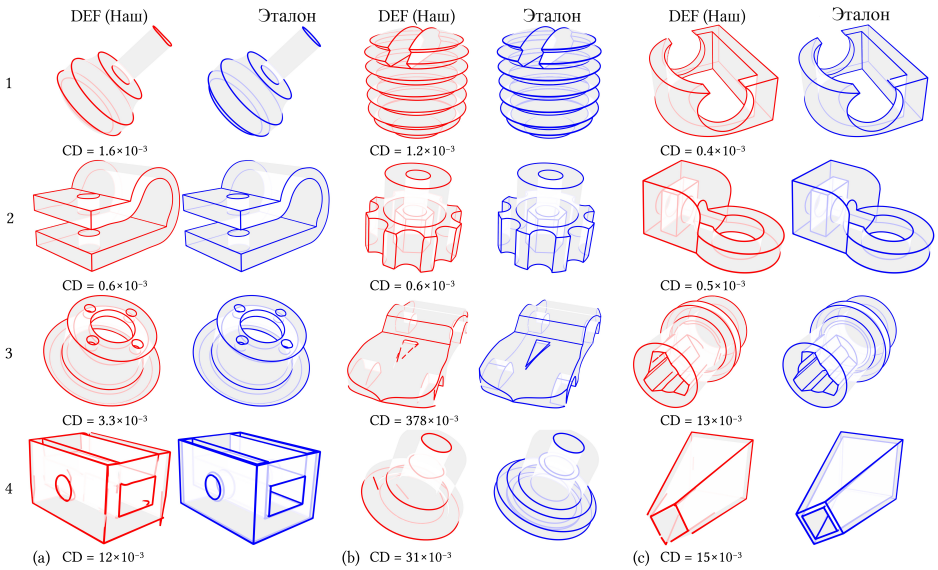


Рис. 15: Результат работы предложенного метода полной трехмерной векторизации DEF.

Описанные шаги аналогичны для замкнутых кривых: конечные точки сплайна должны встречаться в одной точке, и касательные в позициях конечных точек должны быть равны.

8.1.6. Постобработка представлений, полученных на основе сплайнов

Для улучшения итогового результата применяется процедура постобработки. После данной процедуры остаются только кривые, которые хорошо подходят изначальному облаку точек. Качество описания кривой изначального облака точек, определяется с помощью метрики качества, которая вычисляется как показатель F_1 расстояний Чамфера между выбранными кривыми и P_{sharp} и наоборот (21):

$$\begin{aligned} \text{CD}_{X\to Y} &= \frac{1}{N_X}\sum_{x\in X}\inf_{y\in Y}\|x-y\|^2, \\ F_1(T_{\text{metric}}) &= \frac{2\cdot\mathbb{1}(\text{CD}_{X\to Y}\leqslant T_{\text{metric}})\cdot\mathbb{1}(\text{CD}_{Y\to X}\leqslant T_{\text{metric}})}{\mathbb{1}(\text{CD}_{X\to Y}\leqslant T_{\text{metric}})+\mathbb{1}(\text{CD}_{Y\to X}\leqslant T_{\text{metric}})}, \end{aligned} \tag{21}$$

где $\text{CD}_{X\to Y}$ - это расстояние Чамфера от множества точек X до множества точек Y, $\mathbb{1}$ - индикаторная функция, а T_{metric} - порог, используемый для преобразования вещественных расстояний в ``жесткие'' метки 0-1. Чем меньше данный показатель тем лучше. При использовании этой метрики для постобработки P_{sharp} определяется как одно из множеств точек, в то время как другое множество представляет собой дискретизированный набор кривых.

Для начала данная метрика вычисляется с использованием всех кривых. Затем каждая кривая последовательно исключается из расчета и метрика снова вычисляется. Если после этого расстояние увеличивается или остается прежним, исключенная ранее кривая остается в итоговом наборе кривых. В противном случае кривая удаляется.

Для завершения процедуры постобработки применяется фильтрация кривых на основе их длины. Этот процесс включает в себя обнаружение связанных наборов кривых, для каждого из

которых подсчитывается количество кривых, которые его формируют, и вычисляется общая длина всех кривых в нем. Далее удаляются все кривые у которых вычисленная длина меньше порога.

Иллюстрация работы полной системы векторизации представлена на рис. 15.

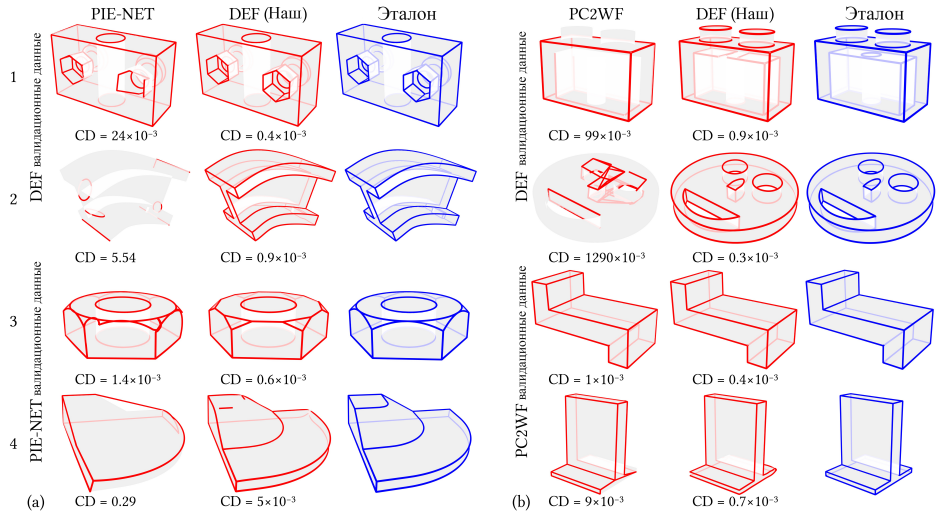


Рис. 16: Качественное сравнение работы предложенного метода DEF с результатами PIE-NET (a) и PC2WF (b).

8.2. Тестирование предложенного подхода

Для тестирования описанного подхода метод векторизации запускается на полных 3D-моделях, сэмплированных с использованием $n_v = 128$ видов. После установки параметров метод был запущен без ручного вмешательства. Выходными данными метода являются (1) параметры сплайновых кривых и (2) координаты конечных точек прямых линий.

Таблица 3: Сравнения параметрических кривых, полученных с помощью DEF и *PIE-NET*.

Метод	CD ↓	HD ↓	SD ↓
PIE-NET [41]	0.97	2.19	0.84
DEF (Наш)	**0.04**	**0.55**	**0.05**

Для оценки качества модели система была запущена на тестовом наборе 68 полных 3D-моделей (DEF-Sim). Для сравнения на этом же наборе данных были также запущены методы *PIE-NET* и *PC2WF*, и полученные результаты сравнивались с истинными параметрическими кривыми. Для вычисления метрик были получены облака точек путем сэмплирования предсказанных кривых и линий, а так же набора истинных кривых. Для полученных облаков точек были вычислены расстояния CD, HD и SD между предсказаниями и эталоном. Агрегированная статистическая оценка метрик для нашего метода и *PIE-NET* приведена в таблице 3. Качественное сравнение представлено на рис. 16.

По сравнению с *PIE-NET*, DEF обнаруживает больше экземпляров кривых, и благодаря предсказанному полю расстояний процедура подгонки зависит не только от позиций точек, а также не имеет проблем с сэмплированием. DEF может подгонять кривые различных типов, в то время как *PC2WF* предназначен только для прямых линий.

Дополнительно было продемонстрировано, что предложенная система векторизации показывает гораздо более качественные результаты по сравнению с методом на котором она основывается (метод *Wireframes*). Качественное сравнение представлено на рис. 13. Улучшенное обнаружение углов и построение полилиний на основе k ближайших соседей позволяет разработанному методу разрешать случаи близких углов и сложных кривых. Топология графа кривых направляет этап подгонки кривых и, если эта топология неточна, то это может привести к некорректным кривым, как видно на выходе метода *Wireframes*.

Заключение

В данном разделе был представлен метод векторизации изображений глубины для получения параметрического представления трехмерного объекта. Здесь векторным представлением трехмерного объекта служит каркас объекта, представленный с помощью параметрических кривых.

Для того чтобы получить параметрические кривые, на каждое из входных изображений глубины была применена нейронная сеть на основе U-Net для предсказания расстояния до особых линий. Далее полученный результат был объединен в трехмерное поле расстояний. На основе этого поля были извлечены параметрические кривые, и таким образом было получено векторное представление входного объекта.

9. Заключение

Данное исследование посвящено разработке методов для решения задач векторизации RGB изображений и изображений глубины с использованием глубокого обучения. Для решения данной задачи были разработаны и применены алгоритмы глубокого обучения и оптимизации, специально адаптированные для задач векторизации двухмерных технических чертежей и трехмерных моделей. Были собраны и предобработаны соответствующие данные, включая синтетически сгенерированные и реальные изображения и модели. Это позволило создать наборы данных, необходимые для тренировки и оценки разработанных моделей. Затем была произведена разработка нейронных сетей и методов оптимизации, которые позволяют точно и эффективно векторизировать входные объекты.

В рамках данного исследования были получены следующие результаты:

- предложен метод получения высококачественных геометрических данных для двухмерных и трехмерных объектов;
- разработана новая система для векторизации растровых изображений технических чертежей;
- разработан алгоритм для восстановления параметрических моделей, описывающих особые кривые у трехмерных форм.

Результаты данного исследования представляют собой значимый вклад в область глубокого обучения и векторизации объектов. С помощью разработанных алгоритмов была достигнута высокая точность и эффективность при решении задач векторизации объектов. Представленные модели демонстрируют способность извлекать математические примитивы и связи между ними, представляя объекты в виде точных векторных представлений. Разработанные алгоритмы имеют большой потенциал для применения в различных областях, включая распознавание и анализ технических чертежей, автоматизированное моделирование и редактирование трехмерных объектов.

В дальнейших исследованиях можно углубиться в изучение альтернативных архитектур нейронных сетей, разработку более сложных оптимизационных методов и рассмотрение других типов данных для векторизации. Также важно продолжать работу над улучшением качества и обобщающей способности моделей.

Данное исследование является важным шагом в направлении разработки эффективных методов векторизации объектов с использованием глубокого обучения. Разработанные методы открывают новые перспективы для различных приложений и дальнейших исследований, стимулируя развитие области в целом и внося ощутимый вклад в научное сообщество.

Список литературы

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *ICML*, pages 40--49, 2018.

[2] Bin Bao and Hongbo Fu. Vectorizing line drawings with near-constant line width. In *2012 19th IEEE International Conference on Image Processing*, pages 805--808. IEEE, 2012.

[3] Mikhail Bessmeltsev and Justin Solomon. Vectorization of line drawings via polyvector fields. *ACM Transactions on Graphics (TOG)*, 38(1):9, 2019.

[4] Jiazhou Chen, Mengqi Du, Xujia Qin, and Yongwei Miao. An improved topology extraction approach for vectorization of sketchy line drawings. *The Visual Computer*, 34(12):1633--1644, 2018.

[5] JiaZhou Chen, Qi Lei, YongWei Miao, and QunSheng Peng. Vectorization of line drawing image based on junction analysis. *Science China Information Sciences*, 58(7):1--14, 2015.

[6] Xuelin Chen, Baoquan Chen, and Niloy J Mitra. Unpaired point cloud completion on real scans using adversarial training. *arXiv preprint arXiv:1904.00069*, 2019.

[7] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Trans. Graph.*, 5(1):51–72, jan 1986.

[8] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*, pages 1486--1494, 2015.

[9] Luca Donati, Simone Cesano, and Andrea Prati. A complete hand-drawn sketch vectorization framework. *Multimedia Tools and Applications*, 78(14):19083--19113, 2019.

[10] Vage Egiazarian, Savva Ignatyev, Alexey Artemov, Oleg Voynov, Andrey Kravchenko, Youyi Zheng, Luiz Velho, and Evgeny Burnaev. Latent-space laplacian pyramids for adversarial representation learning with 3d point clouds. In *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications, 2020.

[11] Kevin Ellis, Daniel Ritchie, Armando Solar-Lezama, and Josh Tenenbaum. Learning to infer graphics programs from hand-drawn images. In *Advances in neural information processing systems*, pages 6059--6068, 2018.

[12] Jean-Dominique Favreau, Florent Lafarge, and Adrien Bousseau. Fidelity vs. simplicity: a global approach to line drawing vectorization. *ACM Transactions on Graphics (TOG)*, 35(4):120, 2016.

[13] Jun Gao, Chengcheng Tang, Vignesh Ganapathi-Subramanian, Jiahui Huang, Hao Su, and Leonidas J Guibas. Deepspline: Data-driven reconstruction of parametric curves and surfaces. *arXiv preprint arXiv:1901.03781*, 2019.

[14] Yi Guo, Zhuming Zhang, Chu Han, Wen-Bo Hu, Chengze Li, and Tien-Tsin Wong. Deep line drawing vectorization via line subdivision and topology reconstruction. *Comput. Graph. Forum*, 38:81--90, 2019.

[15] David Ha and Douglas Eck. A neural representation of sketch drawings. In *International Conference on Learning Representations*, 2018.

[16] JH Hannay and JF Nye. Fibonacci numerical integration on a sphere. *Journal of Physics A: Mathematical and General*, 37(48):11591, 2004.

[17] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2004.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770--778, 2016.

[19] Xavier Hilaire and Karl Tombre. Robust and accurate vectorization of line drawings. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):890--904, 2006.

[20] Ruchin Kansal and Subodh Kumar. A vectorization framework for constant and linear gradient filled regions. *The Visual Computer*, 31(5):717--732, 2015.

[21] Tapas Kanungo, Robert M. Haralick, Henry S. Baird, Werner Stuezle, and David Madigan. A statistical, nonparametric methodology for document degradation model validation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1209--1223, 2000.

[22] Byungsoo Kim, Oliver Wang, A Cengiz Öztireli, and Markus Gross. Semantic segmentation for line drawing vectorization using neural networks. In *Computer Graphics Forum*, volume 37, pages 329--338. Wiley Online Library, 2018.

[23] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9601--9611, 2019.

[24] Chengze Li, Xueting Liu, and Tien-Tsin Wong. Deep extraction of manga structural lines. *ACM Transactions on Graphics (TOG)*, 36(4):117, 2017.

[25] Chun-Liang Li, Manzil Zaheer, Yang Zhang, Barnabas Poczos, and Ruslan Salakhutdinov. Point cloud gan. *arXiv preprint arXiv:1810.05795*, 2018.

[26] Priyanka Mandikal and Venkatesh Babu Radhakrishnan. Dense 3d point cloud reconstruction using a deep pyramid network. In *WACV*, pages 1052--1060. IEEE, 2019.

[27] Albert Matveev, Alexey Artemov, Denis Zorin, and Evgeny Burnaev. 3d parametric wireframe extraction based on distance fields. In *2021 4th International Conference on Artificial Intelligence and Pattern Recognition*, AIPR 2021, page 316–322, New York, NY, USA, 2021. Association for Computing Machinery.

[28] Albert Matveev, Ruslan Rakhimov, Alexey Artemov, Gleb Bobrovskikh, Vage Egiazarian, Emil Bogomolov, Daniele Panozzo, Denis Zorin, and Evgeny Burnaev. Def: Deep estimation of sharp geometric features in 3d shapes, 2022.

[29] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[30] Haoran Mo, Edgar Simo-Serra, Chengying Gao, Changqing Zou, and Ruomei Wang. General virtual sketching framework for vector line art. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2021)*, 40(4):51:1--51:14, 2021.

[31] Vishaal Munusamy Kabilan, Brandon Morris, and Anh Nguyen. Vectordefense: Vectorization as a defense to adversarial examples. *arXiv preprint arXiv:1804.08529*, 2018.

[32] Patryk Najgebauer and Rafal Scherer. Inertia-based fast vectorization of line drawings. *Comput. Graph. Forum*, 38:203--213, 2019.

[33] Gioacchino Noris, Alexander Hornung, Robert W Sumner, Maryann Simmons, and Markus Gross. Topology-driven vectorization of clean line drawings. *ACM Transactions on Graphics (TOG)*, 32(1):4, 2013.

[34] Open CASCADE Technology OCCT. https://www.opencascade.com/, 2021. Accessed: 2021-06-01.

[35] PrecisionFloorplan. PrecisionFloorplan. http://precisionfloorplan.com. Accessed: 2020-03-05.

[36] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234--241. Springer, 2015.

[37] Peter Selinger. Potrace: a polygon-based tracing algorithm. *Potrace (online), http://potrace.sourceforge.net/potrace.pdf (2009-07-01)*, 2003.

[38] Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. Mastering sketching: adversarial augmentation for structured prediction. *ACM Transactions on Graphics (TOG)*, 37(1):11, 2018.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998--6008, 2017.

[40] Oleg Voynov, Gleb Bobrovskikh, Pavel Karpyshev, Saveliy Galochkin, Andrei-Timotei Ardelean, Arseniy Bozhenko, Ekaterina Karmanova, Pavel Kopanev, Yaroslav Labutin-Rymsho, Ruslan Rakhimov, et al. Multi-sensor large-scale dataset for multi-view 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21392--21403, 2023.

[41] Xiaogang Wang, Yuelang Xu, Kai Xu, Andrea Tagliasacchi, Bin Zhou, Ali Mahdavi-Amiri, and Hao Zhang. Pie-net: Parametric inference of point cloud edges. *Advances in Neural Information Processing Systems*, 33, 2020.

[42] Jiaojiao Zhao, Jie Feng, and Bingfeng Zhou. Image vectorization using blue-noise sampling. In *Imaging and Printing in a Web 2.0 World IV*, volume 8664, page 86640H. International Society for Optics and Photonics, 2013.