МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ М.В. ЛОМОНОСОВА

ФИЛИАЛ МГУ ИМЕНИ М.В.ЛОМОНОСОВА В Г.ЕРЕВАНЕ

Направление 01.03.02 «Прикладная математика и информатика»

Выпускная квалификационная работа

Тема: Программные средства для реконструкции 3D-модели на основе изображений

Исполнитель:
Студент (ка) 4 курса
Закарян Андрей Мелконович

Научный руководитель:
Смирнов Илья Николаевич,
к.ф.-м.н., старший преподаватель
кафедры общей математики факультета
ВМК МГУ имени М. В. Ломоносова

Допустить к защите:
Зам. исполнительного директора
по учебной работе
к.ф.н., доцент
Багиян Жан Григорьевич
_____
«02» 06 2019

Ереван-2019

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ имени М.В.ЛОМОНОСОВА

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ

Отзыв на выпускную квалификационную работу

Закаряна Андрея Мелконовича

«Программные средства для реконструкции 3D модели на основе изображении»

Выпускная квалификационной работа Закаряна А.М посвящена разработке программы для реконструкции 3D модели на основе изображений.

В рамках данной работы студент разработал приложение для платформы Android. Студентом были изучены язык Java и среда разработки Android приложений. Были протестированы библиотеки, предназначенные для обработки изображений. Были протестированы методы для поиска «ключевых точек» на паре изображений (SIFT, SURF и другие), а также методы для поиска общих точек. На основе выбранных методов была реализована реконструкция облака точек с помощью библиотеки OpenCV. Студентом был разработан авторский алгоритм определения плоских структур на изображении по полученному облаку точек. Алгоритм реализован на языке C++.

Следует отметить, что в результате работы Закаряном А.М. разработан алгоритм, существенно превосходящий (до 10 раз) по скорости обработки фотографий объектов, содержащих плоскости, другие коммерческие продукты.

Считаю, что Закарян А.М проделал большую работу, и выпускная квалификационная работа заслуживает оценки «ОТЛИЧНО»

Научный руководитель:

к.ф-м.н., ст. препод.  И.Н. Смирнов

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ имени М.В.ЛОМОНОСОВА

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ

Рецензия на выпускную квалификационную работу

Закаряна Андрея Мелконовича

«Программные средства для реконструкции 3D модели на основе изображений»

В работе Закаряна А.М. рассмотрена задача распознавания и представления объекта набором плоскостей методами ортофотограмметрии. В работе оптимизированы алгоритмы реконструкции для работы с объектами, которые преимущественно состоят из плоскостей и разработана программа для платформы Android, которая дает возможность делать снимки и создавать по ним 3D модели объектов.

К замечаниям следует отнести, что обработка облака точек производится на удаленном сервисе по запросу, а не на мобильном устройстве, что требует наличия доступа сервису и не вмегда удобно конечному пользователю. Однако анализ плоскостей в облаке точек реализован уже в мобильном приложении, что является преимуществом перед имеющимися коммерческими продуктами.

Выпускная квалификационная работа Закаряна Андрея Мелконовича на тему «Программные средства для реконструкции 3D модели на основе изображений» выполнена на высоком научном уровне, автор показал себя квалифицированным специалистом, владеющим аппаратом математической статистики, методами машинного обучения и ортофотограмметрии. Считаю, что она заслуживает оценки “отлично”.

Рецензент:
к.ф-м.н., м.н.с.
кафедра НДСи ПУ

Е.И. Атамась

Российская Федерация * Московский государственный университет имени М.В. Ломоносова * Факультет вычислительной математики и кибернетики * г. Москва *
ОТДЕЛ КАДРОВ

Подпись удостоверяю
Ведущий специалист по кадрам
Т.Г. Коваленко

Содержание

1. Введение

Трехмерная реконструкция - процесс получения формы и облика реальных объектов [1]. Исследование 3D реконструкции всегда было сложной задачей. Трехмерная реконструкция объектов является общенаучной проблемой и основной технологией широкого спектра областей, таких как системы автоматизированного проектирования, компьютерная графика, анимация, компьютерное зрение, медицинская визуализация, виртуальная реальность и т.д. Например, информация о болезни пациентов может быть представлена в 3D на компьютере, что предлагает новый и точный подход в диагностике и, таким образом, имеет жизненно важное значение в области медицины. Другим примером применения 3D реконструкции является создание цифровой модели рельефа с помощью лазерных снимков.

Было разработано множество методов 3D реконструкции, которые делятся на две основные группы: активные и пассивные. Активные методы предполагают прямого взаимодействия с объектом. Примером может служить радиометрический метод, при котором на объект испускается излучение. Пассивные методы производят восстановление модели объекта без взаимодействия с ним. Одним из таких методов является фотограмметрия, где информация об объекте извлекается из цифровых изображений, сделанных с разных ракурсов.

С каждым годом появляются новые программы и алгоритмы для генерации пространственных данных. Обновляются существующие программы, принося новые возможности своим пользователям и более высокую производительность. Каждая из них обрабатывает данные по своему, использует различные комбинации алгоритмов и подходит для определенной цели.

Другим различием программ является конечный результат их работы. Программа может представить объект в виде облака точек, поверхностной модели либо полигональной сетки с текстурой. В данной работе рассматривается только класс тех программ и инструментов, которые основаны на фотограмметрической обработке цифровых изображений и воссоздании облака точек.

Одни приложения могут иметь преимущества в некоторых областях, так как они специализируются на конкретных видах объектов. Например, Agisoft Metashape в основном специализируется на картографии и археологии.

Для данной работы тоже была выбрана специфическая проблема - разработка алгоритма для реконструкция 3D модели объектов, которые в основном состоят из плоскостей. Такой алгоритм может помочь при создании модели экстерьера и интерьера помещения.

2. Постановка задачи

Задача дипломной работы - создание программы для реконструкции 3D моделей помещений для платформы Android. В рамках данной работы были поставлены следующие цели:

- Исследовать существующие программы для реконструкции 3D моделей и анализировать их работу
- Изучить и подобрать алгоритмы, которые можно применить для 3D реконструкции объектов
- Разработать алгоритм для поиска плоскостей в облаке точек и устранения дефектов реконструкции
- Разработать приложение для платформы Android

3. Обзор существующих инструментов

Существует множество программ для обработки изображений и создания модели с помощью фотограмметрии. Было протестировано много программ и проведен анализ их работы. Ниже описаны предназначения и алгоритмы работы некоторых программ.

3.1. Описание инструментов

Agisoft Metashape

Agisoft Metashape [2] (в прошлом Agisoft Photoscan) — это программный продукт, который выполняет фотограмметрическую обработку изображений. Он применяется в картографии, горном деле, разработке карьеров, точном земледелии, археологии, дизайне игр. Данный продукт имеет удобный пользовательский интерфейс с множеством возможностей - редактирование и классификация плотных облак точек, генерация и текстурирование 3D моделей, воссоздание сетки по карте глубины и т.д.

При большом объеме входных данных Agisoft создает достаточно точную модель объекта, но требует большой мощности компьютера, а обработка может занять много времени. Например для создания детальной модели здания потребуется 730 изображений, которые будут обрабатываться в течение 9 часов. Для пользователей, не имеющих мощного оборудования, предоставляется возможность облачной обработки данных.

Meshroom

Meshroom [3] **—** бесплатное программное обеспечение с открытым исходным кодом, построенное на фреймворке AliceVision. Рабочий процесс программы построен на основе узлов, каждый из которых соответствует отдельному этапу фотограмметрического конвейера. На каждом этапе у пользователя есть возможность редактировать полученный промежуточный результат. Благодаря такому устройству Meshroom является простым в использовании. Программа также имеет интерфейс командной строки.

Meshroom не специализируется на каких-либо областях применения, поэтому при одном и том же наборе данных можно получить более высокое качество, если использовать Agisoft Metashape.

RealityCapture

RealityCapture [4] **—** программное решение для фотограмметрии, которое автоматически извлекает 3D модели из набора обычных изображений, лазерных сканов или изображений, снятых с помощью синхронизированных камер. Программа автоматически разбивает задачу на меньшие части, которые лучше всего подходят для имеющихся аппаратных средств. Как и Agisoft Metashape, данная программа применяется преимущественно в картографии, археологии, дизайне игр, а также в производстве и научных исследованиях.

3DF Zephyr

3DF Zephyr [5] позволяет абсолютно автоматически, без ручного редактирования реконструировать 3D модели по изображениям. Программа имеет простой в употреблении интерфейс, и в случае необходимости пользователь может вмешаться в процесс.

Autodesk ReCap

Autodesk ReCap [6] разработан для интеграции в рабочие процессы, использующие несколько инструментов. Программное обеспечение имеет два основных режима - один для работы с аэрофотосъемками, а другой для ближней фотограмметрии. Выдающейся особенностью данной программы является автоматическая очистка, которая автоматически идентифицирует и удаляет лишние объекты, которые случайно попали в кадр. ReCap используется в археологии для цифрового сохранения культурного наследия.
У Autodesk также есть приложение 123D Catch, которое предназначено для мобильных устройств. Приложение требует доступа в сеть, так как все данные обрабатываются в облаке.

SCANN3D

SCANN3D [7] **—** приложение для реконструкции трехмерных моделей на телефонах и планшетах. Приложение дает возможность фотографировать объект с нескольких сторон и дает подсказки о том, откуда стоит сделать следующий снимок. В настройках приложения можно выбрать качество конечной 3D модели. Программа требует минимум 20 фотографий для продолжения работы, но на слабых устройствах обработка длится очень долго и часто завершается неудачей. Для получения качественных моделей создатели рекомендуют использовать устройства с сильной аппаратурой.

3.2. Анализ инструментов

Выше перечисленные приложения являются самыми известными фотограмметрическими инструментами. Эти программы, как и многие другие, требуют большого объема входных данных. Каждая из них направлена на работу с изображениями определенной области. В основном это изображения, сделанные с помощью дронов, для картографии. Другим примером является археология - посредством фотограмметрических инструментов можно сохранять цифровые копии памятников культурного наследия.

Такие программы хорошо подходят для сложных задач. Для того, чтобы получить детальный результат часто требуется профессиональное оборудование, изображения высокого качества, иногда и лазерные снимки. Для обработки данных требуется мощная аппаратура и много времени.

Для получения точной 3D модели при реконструкции помещений требуется большое количество высококачественных снимков. Но даже при большом объеме данных, точки, которые на самом деле лежат в одной плоскости, будут отклонятся от нее в модели. Программа, разработанная в рамках данной работы, предполагает, что модель состоит из большого количества плоскостей и проектирует отклоненные точки на данные плоскости. Такой алгоритм позволяет при меньшем наборе входных данных получать более точную модель, при этом используя меньше ресурсов.

4. Реконструкция облака точек

Процесс реконструкции 3D модели на основе цифровых изображений состоит из нескольких этапов [8]:

- **Приобретение данных —** съемка изображений объекта с разных ракурсов и выбор наилучших изображений
- **Предварительная обработка —** оценка параметров камеры и определение ключевых точек
- **Оценка относительного расположения —** сравнение признаков изображений и определение расположения камер по отношению друг к другу
- **Кластеризация —** итеративная реконструкция облака точек при помощи последовательного добавления изображений в кластер

4.1. Приобретение данных

Существуют различные методов приобретения данных, например:

- Лидар (*Light Identification Detection and Ranging*) — технология получения и обработки информации об удалённых объектах с помощью активных оптических систем, использующих явления поглощения и рассеяния света в оптически прозрачных средах.
- Microsoft Kinect — 3D камера, основанная на технологии Structured light, которая позволяет получить карту глубины поверхностей объектов, попавших в поле зрения камеры.
- Координатно-измерительная машина (КИМ) — устройство для измерения геометрических характеристик объекта. Машина может управляться вручную оператором или автоматизировано компьютером.

Данные методы требуют различное оборудование. Для данной работы был выбран метод, который доступен большинству людей — приобретение фотографий с помощью камеры.

Желательно использовать камеру с высоким разрешением и четкими снимками. Для получение более четких снимков можно использовать штатив. Также можно использовать камеру телефона, но в этом случае понадобится большее количество фотографий.

При съемке изображений желательно следовать нескольким правилам. На рисунке 4.1 изображены советы для съемки плоских и изолированных объектов, а также интерьера.

Основная идея данных правил в том, чтобы каждая точка поверхности попадала как минимум на два изображения и, желательно, под разными углами. При таких условиях получится восстановить наибольшее количество точек.

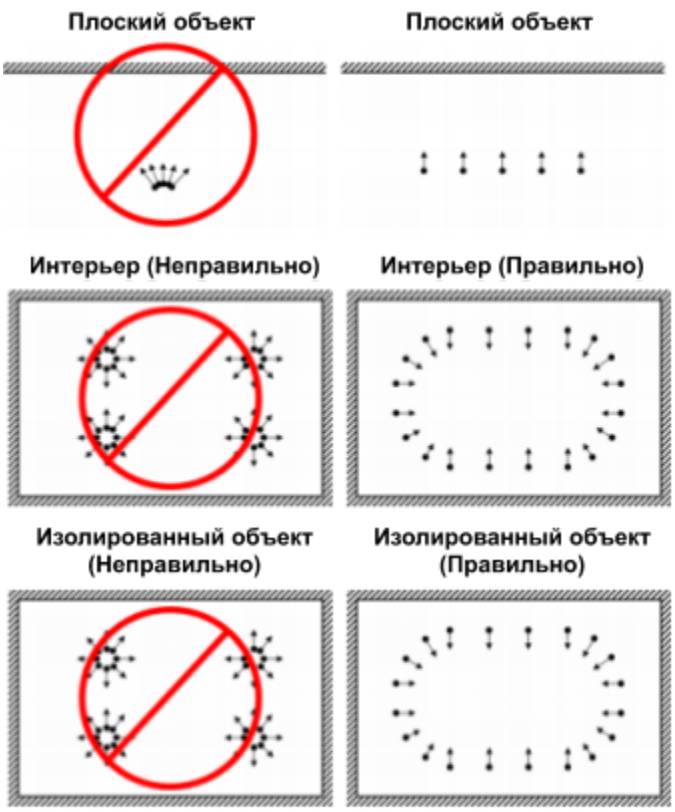

Рис 4.1 Правила съемки

4.2. Предварительная обработка

4.2.1. Калибровка камеры

Матрица внутренней калибровки камеры — это матрица, которая переводит трехмерные координаты в однородные 2D координаты изображения [9]. Она имеет следующий вид:

$$K = \begin{bmatrix} f_x & 0 & x_0 \\ & f_y & y_0 \\ & & 1 \end{bmatrix} \quad (2.1)$$

Каждый из параметров матрицы описывает геометрическое свойство камеры.

- f_x , f_y — фокусные расстояния камеры в пикселях. В основном они равны $f_x = f_y = f$
- x_0, y_0 — координаты главной точки, т. е. координаты пересечения оптической оси объектива с плоскостью сенсора.

Для получения корректных результатов 3D реконструкции необходимо выполнить автоматическую калибровку камеры. Современные камеры добавляют к изображению дополнительную информацию в виде заголовка EXIF [10] (*Exchangeable Image File Format*). Информацию из EXIF файла можно использовать для расчета фокусного расстояния и размера пикселя сенсора фотоаппарата. С помощью этих данных можно вычислить tматрицу калибровки, которая пригодится на следующих этапах реконструкции.

4.2.2. Выделение ключевых точек

Вторым этапом предварительной обработки является выделение ключевых точек изображения, т.е. тех точек, окрестность которых хранит в себе много информации. Наиболее известными алгоритмами выявления ключевых точек являются SIFT(*Scale Invariant Feature Transform*) и SURF(Speeded-Up Robust Features) [11].

Оба алгоритма состоят из двух основных шагов:

1. обнаружение точек интереса
2. извлечение дескрипторов для окрестности каждой точки

После проведения тестов на различных изображениях было выявлено, что точки, полученные с помощью алгоритма SIFT, лучше подходят для реконструкции. Поэтому, для данной работы был выбран алгоритм SIFT.

Преимущество SIFT-а состоит в том, что он находит точки независимо от масштаба и поворота изображения. Ключевые точки ищутся как экстремумы функции DoG(*Difference of Gaussians*) по всем масштабам изображения. А дескрипторам присваивается значение локальных градиентов в каждой точке.

Рис 4.2 Ключевые точки (алгоритм SIFT)

На рисунке 4.2 изображены ключевые точки, полученные с помощью алгоритма SIFT. На том же самом изображении SURF распознал 21777 ключевых точек, в то время как SIFT обнаружил 58945 точек.

4.3. Оценка относительного расположения

Первым шагом данного этапа является сопоставление ключевых точек между двумя изображениями, т.е. нахождение пар ключевых точек, которые соответствуют друг другу. Данная задача сводится к задаче поиска ближайшего соседа. Для сравнения точек используются дескрипторы, которые были получены с помощью SIFT-а [12].

Рис 4.3 Сопоставление изображений

На рисунке 4.3 изображены два снимка, снятые с разных ракурсов, а линиями показаны совпадения некоторых ключевых точек. Всего между изображениями было найдено 10112 совпадений. Для поиска совпадений использовалась библиотека FLANN (*Fast Library for Approximate Nearest Neighbors*) [13].

Можно заметить, что на изображении есть линии, значительно отклоняющиеся от остальных. Такие линии соответствуют неправильно найденым соответствиям и фильтруются после работы алгоритма.

После того, как сопоставлены точки на двух изображениях, можно найти фундаментальную матрицу. Данная матрица показывает относительное расположение точек на изображениях, не учитывая внутренних параметров камеры. Для поиска фундаментальной матрицы требуется как минимум 7 точек. Существует алгоритм который находит фундаментальную матрицу используя 7 точек, но данный алгоритм включает в себя решение нелинейных уравнений. Более эффективным методом является

алгоритм использующий 8 точек (*normalized 8-point algorithm*) [14].

Имея фундаментальную матрицу и матрицу калибровки, которая была получена на первом этапе, можно вычислить существенную матрицу:

$$E = K'^{T} \cdot F \cdot K \quad (2.2)$$

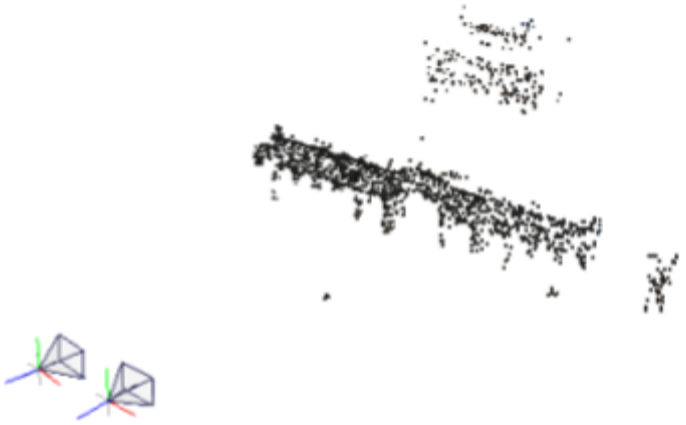

Рис 4.4 Определение расположения камер

Где K - это матрица калибровки, а F - фундаментальная матрица.

Существенная матрица показывает относительное расположение двух камер.

Далее, используя существенную матрицу и координаты ключевых точек на двух изображениях, можно восстановить трехмерные координаты этих точек. Данный процесс называется триангуляцией.

На рисунке 4.4 изображено облако точек, полученное с помощью двух изображений, и позиции камер, с которых были сняты данные изображения.

4.4. Кластеризация

Целью данного этапа является восстановление позиций всех камер. Под кластером понимается набор камер с их внутренними параметрами и облако точек, сконструированное из ключевых точек. Процесс кластеризации начинается с тех двух изображений, которые имеют больше всего совпадений среди ключевых точек. Для данных изображений вычисляется относительное расположение камер и облако точек. На каждом следующем шаге последовательно добавляется по одному изображению, до тех пор, пока в кластер не будут включены все изображения. Каждый раз, после добавления очередного изображения в кластер, производится оптимизация позиций полученных точек с целью минимизации ошибок в проекциях.

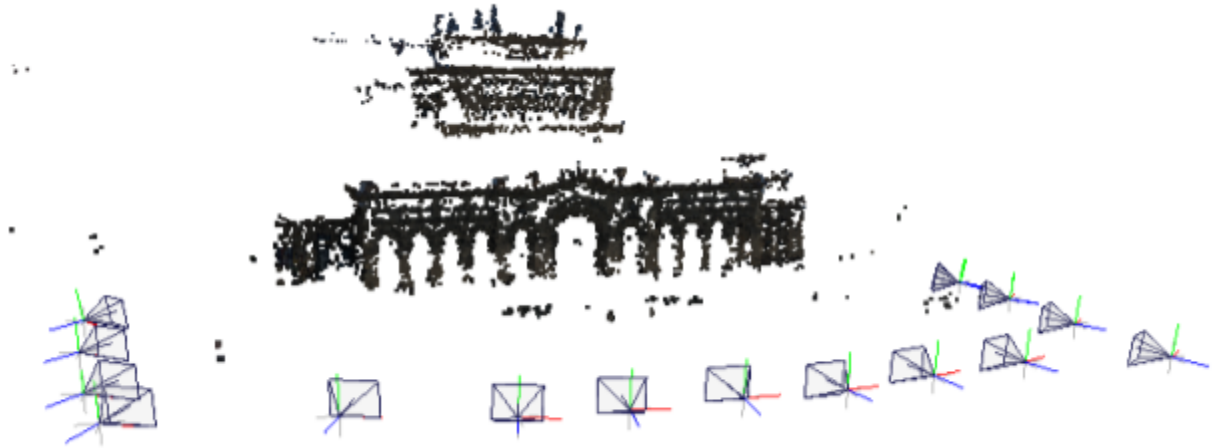

Рис 4.5 Определение расположения всех камер

На рисунке 4.5 изображен результат кластеризации. Определены положения всех камер и составлено редкое облако точек.

4.5. Реализация

Для поиска относительного расположения камер, в данной работе была использована система Buldler SfM (*Structure from Motion*) [15] с открытым исходным кодом, написанная на языках C/C++. На вход Bundler получает набор изображений и их ключевые точки, которые предварительно были вычислены с помощью алгоритма SIFT. На выходе производит редкое облако точек и 3D позиции камер.

Рис 4.6 Густое облако точек

Для получения густого облака точек используется программа PMVS(*Patch-based Multi-view Stereo Software*) [16]. PMVS получает на вход изображения и параметры камеры и на их основе производит густое облако точек.

Для работы с данными программами нужно скачать их исходный код и скомпилировать с помощью системы автоматизации сборки программного обеспечения CMake, либо скачать заранее скомпилированные исполняемые файлы этих программ.

С целью полной автоматизации реконструкции густого облака точек был написан Bash скрипт **Reconstruct.sh**. В качестве параметра скрипт принимает путь к папке с изображениями. Скрипт запускает Bundler с изображениями из данной папки. Bundler вычисляет позиции камер и сохраняет в файле. После чего скрипт приводит результат к формату, понятному для PMVS и запускает реконструкцию облака точек. Конечный результат сохраняется в файле формата PLY(*Polygon File Format*) [17], для дальнейшей обработки.

На изображении 4.6 показан результат реконструкции густого облака точек. Заметно, что в полученном облаке точек присутствует большое количество шума, Например фонтан перед зданием или кусочки неба. В следующей главе описан метод, который был разработан для устранения подобного шума.

5. Реконструкция плоскостей

Главной задачей данной работы является реконструкция объектов, преимущественно состоящих из плоскостей. Облако точек, полученное в результате реконструкции, содержит в себе шум и точки, которые на самом деле лежащие в одной плоскости, в полученной модели отклоняются от нее. Метод, представленный ниже, помогает существенно уменьшить шум и получить более точную модель.

5.1. Поиск плоскостей

Для поиска плоскости, которая лучше всего описывает облако точек, был выбран алгоритм RANSAC(*Random sample consensus*) [18]. Данный алгоритм пытается уменьшить количество итераций даже при большом объеме входных данных.

В основном RANSAC используется для поиска линий на плоскости, так как он менее чувствителен к выбросам, чем обычная линейная регрессия. Но его можно использовать и для поиска плоскостей в пространстве.

На вход алгоритма поступают разные данные:

- X — набор исходных данных
- M — функция, позволяющая вычислить параметры модели по набору данных из n точек
- E — функция оценки соответствия точек полученной модели
- t — порог для функции оценки
- k — количество итераций метода

Для решения задачи поиска плоскостей в облаке точек с помощью алгоритма RANSAC X, M и E были выбраны следующим образом:

- X — облако точек, полученное в результате 3D реконструкции
- M — функция, которая вычисляет параметры плоскости, проходящей через заданные 3 точки
- E — фуннкция, определяющая расстояние между точкой и плоскостью

Каждая итерация алгоритма состоит из двух основных этапов:

1. Построение гипотезы:
    - Из исходного облака точек случайным образом выбираются 3 различные точки.
    - На основе выбранных точек подсчитываются параметры плоскости, которая проходит через данные точки. Построенную модель принято называть гипотезой.
2. Проверка гипотезы:
    - Для каждой точки из исходного облака вычисляется расстояние от построенной плоскости. Полученные значения сравниваются с порогом t.
    - Если расстояние от точки до плоскости не превышает заданного порога t, точка помечается как инлаер. В противном случае, она помечается выбросом.
    - Гипотеза сравнивается с наилучшей из предыдущих. Если в плоскости, построенной на текущей итерации, лежит больше точек, то гипотеза принимается и замещается предыдущую лучшую гипотезу.

Результатами данного алгоритма являются:

- параметры плоскости
- точки, помеченные выбросами либо инлаерами.

Основным недостатком метода RANSAC является то, что нет определенного количества итераций алгоритма, которое может гарантировать оптимальный результат. С увеличением количества итераций результат улучшается. Но все равно существует малая вероятность того, что алгоритм не определит ни одну модель, соответствующую входным данным.

5.2. Оптимизация алгоритма

Алгоритм RANSAC оптимален для нахождения одной плоскости, хорошо аппроксимирующей облако точек. Однако в случае большого количества плоскостей в одном и том же облаке использовать RANSAC не удасться. Алгоритм будет считать оптимальными большие плоскости, а плоскости с маленьким количеством точек не будут рассматриваться. Кроме того, вычисления на большом количестве точек стоят очень дорого.

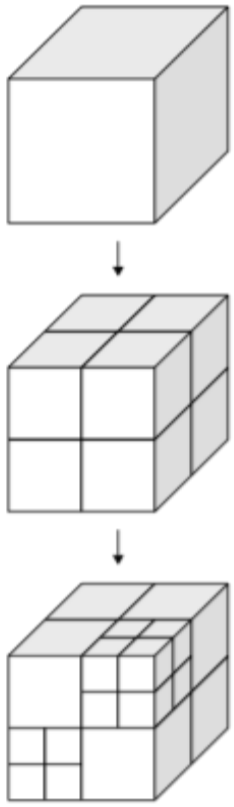

Рис 5.1 Разбиение пространства

Для решения этих проблем была выбрана следующая схема:

- Разбиение пространства, содержащего облако точек, на маленькие части
- Поиск плоскостей в каждой из частей методом RANSAC
- Соединение плоскостей из соседних частей пространства

Разбиение пространства производится с помощью октодерева [19]. Октодерево — тип древовидной структуры данных, в которой у каждого внутреннего узла ровно восемь потомков. Каждый потомок соответствует одной из ячеек трехмерного пространства, разбитого на восемь частей. Пример разбиения показан на рисунке 5.1. Те октанты, в которых находится малое количество точек, либо их нет вообще, остаются не разбитыми.

После рекурсивного разбиения пространства на октанты, в самых маленьких октантах производится поиск плоскостей методом RANSAC. При таком подходе есть возможность найти не только доминирующие плоскости, но и локальные плоскости. Такая возможность может помочь, например, при нахождении лестниц помещения.

После того, как в каждом октанте найдены плоскости, остается соединить те части плоскостей, которые относятся к одной, большой плоскости. Для этого сравниваются части плоскостей, лежащие в соседних октантах.

Сравнение происходит по трем факторам:

1. **Параллельность нормалей**. Если нормали параллельны, либо отклонены друг от друга на небольшой угол, есть вероятность, что данные части лежат в одной и той же плоскости. В противном случае не нужно проверять остальные факторы и сразу опровергнуть предположение.
2. **Выравненность плоскостей**. Возможен случай, когда нормали параллельны и сами плоскости тоже параллельны. Для устранения такого случая нужно убедится, что прямая, проходящая через концы нормалей, перпендикулярна им.
3. **Близость плоскостей**. Считаем, что части относятся к одной плоскости только в том случае, когда они находятся довольно близко друг к другу и имеют общую грань.

Объединение плоскостей происходит на каждом уровне октодерева. Сначала сравниваются октанты, являющиеся листьями деревьев. Процесс продолжается до тех пор, пока не будут объединены восемь самых крупных октантов.

После определения параметров всех плоскостей, фильтруются точки, которые не принадлежат ни одной из них. Все точки, которые находятся рядом с плоскостями проектируются на них. В результате получается модель с маленьким количеством шума и ровными плоскостями.

Рис 5.2 Модель после восстановления плоскостей

6. Программная реализация

Программа была написана на языке программирования C++ на платформе Linux, при помощи среды разработки Visual Studio Code. Ниже описаны основные классы программы.

Point

Структура для хранения вершины из облака точек. В данной структуре хранятся трехмерные координаты точки и ее цвет в формате RGB.

Plane

Класс для хранения и расчета параметров плоскостей. Полями данного класса являются центр плоскости, нормаль, радиус, толщина и массив точек. Класс имеет следующие методы:

- **distance** — считает расстояние от точки до плоскости
- **accept** — решает, принадлежит ли точка данной плоскости
- **addPoint** — добавляет точку к плоскости
- **computeEquation** — вычисляет формулу и параметры плоскости
- **merge** — объединяет две плоскости
- **flatten** — проектирует все точки на плоскость

PointCloud

Класс для хранения облака точек. Для работы с облаками точек реализованы методы:

- **merge** — объединяет два облака точек
- **center** — вычисляет расположение центра облака точек
- **addPoint** — добавляет точку к облаку
- **boundingBox** — вычисляет границу облака точек

Ransac

Класс реализующий метод RANSAC. В данном классе реализована одна публичная функция, которая производит вычисления по алгоритму RANSAC и принимает на вход облако точек и предельные значения. В конце работы функция возвращает плоскость, если та была найдена.

Octree

Класс реализующий структуру октодерева. Класс содержит в себе подкласс Node, представляющий один узел октодерева, имеющий 8 потомков. Класс имеет конструктор, для преобразования объекта класса облака точек (*PointCloud*) в октодерево. Функция **detectPlanes** предназначена для обнаружения плоскостей в октодереве.

PLY

Класс для работы с PLY файлами. PLY файл состоит из заголовка и тела. Пример PLY файла приведен в приложении А.

В заголовке указывается кодировка файла, количество вершин и список свойств вершины. Класс работает с файлами кодировки ***ascii 1.0*** и свойствами ***x, y, z, red, green, blue.*** Где ***x, y, z*** — трехмерные координаты вершины, а ***red, green, blue*** — ее цвет.

Тело файла состоит из списка вершин. Каждая строка соответствует одной вершине. В теле должно быть столько строк, сколько указано в заголовке.

Кроме вершин в файле формата PLY можно хранить грани и ребра, но для решения задачи данной работы достаточно только вершин.

В данном классе реализованы функции чтения и записи данных в файл формата PLY:

- **read —** считывает облако точек из файла и возвращает объект класса **PointCloud**
- **write —** записывает облако точек в PLY файл

7. Пользовательский интерфейс

Для работы с программой было реализовано два интерфейса:

- интерфейс командной строки
- Android интерфейс

7.1. Интерфейс командной строки

С помощью данного интерфейса пользователю дается возможность взаимодействовать с программой из командной строки. В качестве параметров командной строки нужно указать папку, в которой находятся исходные изображения. По умолчанию будет использована текущая папка. Также можно передать желаемый путь и название выходного файла с ключом **--output**.

Интерфейс рассчитан и на тот случай, когда у пользователя уже есть готовый ply файл с облаком точек, в котором нужно найти плоскости. В таком случае пользователь может указать путь к файлу и программа не будет вызывать весь конвейер реконструкции, а только его последний шаг — поиск плоскостей.

7.2. Интерфейс Android

Интерфейс для платформы Android был разработан с помощью среды разработки Android Studio [20]. Алгоритм поиска плоскостей, реализованный на языке C++, был скомпилирован под Android с помощью набора инструментов Android NDK (*Native Development Kit*) [21].

В интерфейсе пользователь может сделать снимки объекта и запустить конвейер реконструкции. Изображения отправляются на сервер и сама 3D реконструкция происходит там. При окончании процесса реконструкции, результат отправляется на устройство и поиск плоскостей происходит локально, на устройстве Android. После чего пользователь может посмотреть на результирующее облако точек. Снимок экрана Android интерфейса в приложении Б.

8. Результаты работы

Основные результаты работы:

- Исследованы существующие программы для реконструкции 3D моделей и анализирована их работа
- Изучены основные алгоритмы машинного зрения
- Подобраны алгоритмы, для построения конвейера 3D реконструкции объектов
- Разработан алгоритм для поиска плоскостей в облаке точек и устранения дефектов реконструкции
- Изучена система автоматической сборки CMake
- Разработана программа для командной строки
- Разработан пользовательский интерфейс для платформы Android
- Проведено тестирование программы на различных зданиях

9. Список литературы

[1] Трехмерная реконструкция [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Трёхмерная_реконструкция

[2] Agisoft Metashape [Электронный ресурс]. URL: https://www.agisoft.com/

[3] Meshroom [Электронный ресурс]. URL: https://alicevision.github.io/#meshroom

[4] RealityCapture [Электронный ресурс]. URL: https://www.capturingreality.com/Product

[5] 3DF Zephyr [Электронный ресурс]. URL: https://www.3dflow.net/

[6] Autodesk ReCap [Электронный ресурс]. URL: https://www.autodesk.com/products/recap/overview

[7] SCANN3D [Электронный ресурс]. URL: https://play.google.com/store/apps/details?id=com.smartmobilevision.scann3d&hl=en

[8] Ричарда Селиски. *Компьютерное зрение: алгоритмы и приложения*

[9] Ричард Хартли. *Проективная геометрия в компьютерном зрении*

[10] EXIF [Электронный ресурс]. URL: https://en.wikipedia.org/wiki/Exif

[11] Адриан Келер, Гэри Брэдски. *Изучаем OpenCV 3*

[12] Гонсалес Р., Вудс Р. *Цифровая обработка изображений*

[13] FLANN [Электронный ресурс]. URL: https://www.cs.ubc.ca/research/flann/

[14] Normalized 8-point algorithm [Электронный ресурс]. URL: https://en.wikipedia.org/wiki/Eight-point_algorithm

[15] Bundler [Электронный ресурс]. URL: https://www.cs.cornell.edu/~snavely/bundler/

[16] PMVS [Электронный ресурс]. URL: https://www.di.ens.fr/pmvs/

[17] PLY [Электронный ресурс]. URL: http://paulbourke.net/dataformats/ply/

[18] Марко Зулиани. *RANSAC for Dummies*

[19] Октодерево [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Октодерево

[20] Android Studio [Электронный ресурс]. URL: https://developer.android.com/studio

[21] Android NDK [Электронный ресурс]. URL: https://developer.android.com/ndk

Приложение А. PLY файл

ply
format ascii 1.0
element vertex 10
property float x
property float y
property float z
property uchar red
property uchar green
property uchar blue
end_header
-0.102458 0.939643 -5.50122 175 188 202
-0.0827546 0.932971 -5.50492 179 192 205
-0.0839842 0.931626 -5.50332 167 179 192
-0.0839842 0.931626 -5.50332 167 179 192
-0.0773634 0.929642 -5.50515 174 185 197
-0.0791405 0.928223 -5.50489 119 123 128
-0.0775466 0.928612 -5.50782 138 143 150
-0.0799393 0.926393 -5.5063 141 148 155
-0.0808382 0.926214 -5.50556 144 152 160
-0.0793742 0.927402 -5.50602 118 123 127

Приложение Б. Android интерфейс