МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Кафедра многопроцессорных систем и сетей

ПАЖИТНЫХ
Иван Павлович

РАЗРАБОТКА АЛГОРИТМОВ НАВИГАЦИИ БЕСПИЛОТНЫХ ЛЕТАТЕЛЬНЫХ АППАРАТОВ

Дипломная работа

Научный руководитель:
старший преподаватель
О. М. Кондратьева

Допущена к защите

« » 2018 г.

Зав. кафедрой многопроцессорных систем и сетей
кандидат физико-математических наук, доцент С.В. Марков

Минск, 2018

РЕФЕРАТ

Дипломная работа, 43 страницы, 15 рисунков, 16 источников.

НАВИГАЦИЯ, БПЛА, РЕКОНСТРУКЦИЯ, КОМПЬЮТЕРНОЕ ЗРЕНИЕ, ОСОБЫЕ ТОЧКИ, ДЕСКРИПТОРЫ, ДЕТЕКТОРЫ, СОПОСТАВЛЕНИЕ ИЗОБРАЖЕНИЙ

Объектом исследования являются способы навигации беспилотных летательных аппаратов и алгоритмы компьютерного зрения.

Цель работы – изучение возможности применения методов компьютерного зрения в системах навигации беспилотных летательных аппаратах.

Методы исследования: изучение соответствующей литературы и публикаций, проведение экспериментов, разработка программного обеспечения.

В результате работы были рассмотрены и проанализированы различные существующие алгоритмы компьютерного зрения, предложен алгоритм восстановления местоположения по цифровой карте местности, разработано приложение для построения цифровой карты местности и осуществления поиска по ней. Проведены сравнительные эксперименты.

Области применения: системы навигации, модели и алгоритмы, работающие на борту беспилотных летательных аппаратов, реконструкция 3D карты местности.

РЭФЕРАТ

Дыпломная праца, 43 старонкі, 15 малюнкаў, 16 крыніц.

НАВІГАЦЫЯ, БПЛА, РЭКАНСТРУКЦЫЯ, КАМПУТАРНЫ ЗРОК, КЛЮЧАВЫЯ КРОПКІ, ДЭСКРЫПТАРЫ, ДЭТЭКТАРЫ, СУПАСТАЎЛЕННЕ ВЫЯВАЎ

Аб'ектам даследвання з'яўляюцца метады навігацыі беспілотных лятаючых апаратаў і алгарытмы кампутарнага зроку.

Мэта працы – даследванне магчымасці выкарыстоўвання метадаў кампутарнага зроку ў сістэмах навігацыі беспілотных лятаючых апаратах

Метады даследвання: аналіз адпаведнай літаратуры і публікацый, правядзенне эксперыментаў, распрацоўка праграмнага забеспячэння.

У выніку працы былі разгледжаны і прааналізаваны разнастайныя існуючыя алгарытмы кампутарнага зроку, прапанаваны алгарытм аднаўлення месцазнаходжання па 3D мапе мясцовасці, распрацавана прыкладанне для пабудовы 3D мапы мясцовасці і ажыццяўлення пошука па ёй. Праведзены параўнаўчыя эксперыменты.

Галіны прымянення: сістэмы навігацыі, мадэлі і алгарытмы, якія працуюць на барту беспілотных лятаючых апаратаў, рэканструкцыя 3D мапы мясцовасці.

ABSTRACT

Graduate work, 43 pages, 15 pictures, 16 sources.

NAVIGATION, UAV, RECONSTRUCTION, COMPUTER VISION, KEYPOINTS, DESCRIPTORS, DETECTORS, FEATURE MATCHING

The object of the research are methods of navigation of unmanned aerial vehicles and computer vision algorithms.

Purpose – to research the possibility of applying computer vision algorithms in the navigation systems of unmanned aerial vehicles.

Methods of the research are: to analyze publications, to perform experiments, to develop the software.

As a result of the work, various existing computer vision algorithms were examined and analyzed, an algorithm for restoring the location on a digital terrain map was proposed, an application for constructing a digital terrain map and searching for it was developed. Comparative experiments were carried out.

The scopes are: navigation systems, models and algorithms working on board unmanned aerial vehicles, reconstruction of a 3D map.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время интенсивно развивается такая область информатики как компьютерное зрение (computer vision), в которой учёные и инженеры решают задачу достижения высокого уровня распознавания и понимания видео и изображений для компьютера (машины). С технической стороны конечная цель – автоматизация задач, требующих наличия человеческой зрительной системы. Задачи включают в себя методы для получения, обработки, анализа и интерпретации изображений из реального мира в строго структурированном и понятном для программ виде.

Также существуют такие подзадачи как распознавание и поиск объектов на изображении, сравнение и сопоставление изображений, видео трекинг (слежение), поиск геометрического преобразования, переводящего одно изображение в другое, восстановление (руконструкция) 3D моделей и многие другие.

Алгоритмы компьютерного зрения активно используются в системах управления процессами (промышленные роботы, автономные транспортные средства), системах видеонаблюдения, системах организации информации (индексация баз данных изображений), системах моделирования объектов или окружающей среды (анализ медицинских изображений, топографическое моделирование), системах взаимодействия (устройства ввода для системы человеко-машинного взаимодействия), системы дополненной реальности.

Приложения алгоритмов обширны: от подводных управляемых аппаратов до беспилотных летательных. Маленькие колесные роботы и марсоходы НАСА, применение как в военных целях, так и в повседневных задачах.

Крупнейшая мировая IT корпорация Google уже сейчас разрабатывает self-driving cars (машины с автопилотом) которые, как предполагается, в будущем изменят существующее представление об автомобилях: человеку вообще не придётся управлять машиной. Это должно снизить число аварий, исключая такую причину дорожно-транспортных происшествий как “человеческий фактор” и, соответственно, сделать передвижение с помощью автомобиля безопаснее.

Самый популярный сервис такси Uber уже использует машины с автопилотом, что в будущем позволит ещё больше снизить стоимость услуг сокращением траты средств на человеческие ресурсы (компания уже автоматизировала процесс заказа такси, и диспетчеры были заменены мобильным приложением).

Американская компания Amazon – крупнейший в мире сервис продаж через интернет – открыла магазин без кассиров, в котором с помощью алгоритмов компьютерного зрения определяется какие товары клиент положил себе в корзину и их стоимость автоматически списывается с карты при выходе из магазина. Команда Oculus – подразделение Facebook – создаёт очки вирту-

альной реальности Oculus Rift, которые создатель Facebook Марк Цукенберг считает следующим поколением цифровых устройств, которые должны прийти на смену смартфонам.

Многие проекты, использующие компьютерное зрение, направлены на автоматизацию рутинной работы, уменьшение человеческого труда. Одна из основных задач – улучшение качества жизни путём высвобождения одного из самых дорогих ресурсов – человеческого времени.

И это только несколько проектов, а общее количество стартапов в этой области увеличивается с каждым днём. Это ведёт к увеличению решений и библиотек, многие из который публикуются в открытом доступе. Например, одним из используемых мной решений было OpenCV – самая популярная библиотека с открытым исходным кодом, которая предоставляет реализации основных алгоритмов компьютерного зрения.

Таким образом, компьютерное зрение сейчас – это новая, очень популярная и активно развивающаяся область информатики, с огромным количеством прорывов и открытий которые происходят прямо сейчас.

ГЛАВА 1 АКТУАЛЬНОСТЬ ЗАДАЧИ

1.1 Навигация с помощью GPS

GPS (Global Positioning System) – спутниковая система радио-навигации, которая позволяет, с помощью GPS приёмника, определять координаты и время в любой точке Земли, из которой беспрепятственно видны четыре или более GPS спутника. Система разработана в Соединённых Штатах Америки для нужд Военно-воздушных сил, в 1980х годах доступ к системе был предоставлен для гражданских лиц. Так как система принадлежит правительству США и полностью подконтрольна ему, существует возможность для выборочного ограничивания доступа к системе или ограничение качества сигнала, которая и была использована в 1999 году против индийской армии в ходе Каргильской войны. Также такие препятствия как здания или горы ослабляют сигнал.

Основная идея, которая лежит в основе работы системы: зная точные координаты спутников в определённый момент времени, благодаря эффекту Доплера (изменения частоты принимаемого сигнала при изменении положения передатчика), можно вычислить текущее положение относительно передатчиков (спутников). На GPS спутниках установлены очень точные атомные часы которые синхронизированы между всеми спутниками и земным временем. Также с большой точностью известны положения каждого спутника. Спутники непрерывно транслируют данные о своём текущем положении и времени на борту. GPS приёмник принимает эти сигналы от четырёх спутников и решает систему уравнений с четырьмя неизвестными: тремя координатами и временем. Точность такого метода составляет до пяти метров.

1.2 Актуальность и практическая значимость

Как следует из названия, беспилотные летательные аппараты (дроны) не имеют пилота, но это не значит, что они не пилотируемы. Управление беспилотником требует специального обучения и сосредоточенности, является очень утомительным для оператора. Основополагающим условием для работы дрона является наличие GPS сигнала, что делает его очень уязвимым и зависимым от внешних обстоятельств. В отсутствие сигнала системы глобального позиционирования дрон теряет управление.

GPS не очень надёжная система, в работе которой могут быть сбои и помехи. В связи с этим возникает задача нахождения и использования альтернативных источников навигации. Так как почти каждый современный беспилотник оснащён камерой, то возможно применение алгоритмов компьютерного зрения для решения данной задачи. С помощью таких алгоритмов и про-

граммного обеспечения должна быть возможна навигация дрона, используя только камеру. Зная начальное местоположение и отслеживая с помощью алгоритмов компьютерного зрения свое перемещение, робот может вычислять текущее местоположение относительно начального.

Дополнительные возможности применения обширны: патрулирование заданной территории и обнаружение новых объектов, не присутствовавших ранее (например обнаружение лесных пожаров), возвращение в заданную точку в случае потери GPS сигнала, слежение за данным объектом, построение 3D карт местности, навигация по заданной цифровой карте.

1.3 Постановка задачи

Цель – исследовать возможность применения камеры дрона для определения текущего местоположения и решения задачи навигации беспилотного летательного аппарата в условиях отсутствия GPS сигнала.

Для достижения цели необходимо:

1. Изучить и проанализировать существующие алгоритмы компьютерного зрения:
   - дескрипторы SIFT, SURF, ORB;
   - алгоритм проективной геометрии Bundle Adjustment;
   - метод Structure from Motion (SFM);
   - метод Simultaneous Localization and Mapping (SLAM).
2. Провести практические эксперименты и сравнения разных подходов;
3. Разобраться в принципах работы операционной системы Robot Operating System (ROS);
4. Реализовать приложение для построения 3D карты местности и осуществления поиска по ней.

ГЛАВА 2 МЕТОД STRUCTURE FROM MOTION

Structure from Motion (дословно "*структура из движения*", SFM) – техника построения трёхмерных структур из последовательности двухмерных изображений (фотографий, кадров видео), используемая в области компьютерного зрения. В биологии этот термин описывает феномен, позволяющий человеку восстанавливать трёхмерную структуру окружающего мира по двухмерным проекциям на сетчатку глаза.

2.1 Основные тереоретические понятия

Люди с рождения обладают зрением, что позволяет нам распознавать изображения и объекты на них, сравнивать их между собой, оценивать расстояния и размеры. Всё это человеческий мозг делает бессознательно, автоматически. Однако, для машины изображение — всего лишь закодированные данные, набор нулей и единиц. Одной из основных проблем в сопоставлении изображений является очень большая размерность пространства, которое обладает информацией. Если взять маленькую картинку размером хотя бы $100 * 100$ пикселей, то уже получим количество информации равное 10^4 пикселей. Поэтому методы анализа изображения должны быть быстрыми и эффективными.

Существует ряд других проблем и сложностей при попытке интерпретации изображений. Одни и те же объекты на разных изображениях могут очень сильно отличаться. На это влияют такие параметры как точка зрения, освещение (один и тот же объект может быть белым, серым или чёрным в зависимости от освещения), масштаб, деформация и перекрытие объектов, маскировка (слияние с фоном). Поэтому для полноты картины требуется как можно больше различных изображений одного объекта.

Так как же машина обретает зрение? Основная идея состоит в том, чтобы получить какую-то характеристику, которая будет хорошо описывать изображение, легко вычисляться и для которой можно ввести оператор сравнения. Эта "характеристика" должна быть устойчива к различным преобразованиям (сдвиг, поворот и масштабирование изображения, изменение яркости, изменение положения камеры). Это необходимо для того, чтобы было возможно определить один и тот же объект на изображениях, сделанных с разных углов, расстояний и при разном освещении.

Все эти условия приводят к необходимости выделения на изображении особых, *ключевых точек* (**key points**). Этот процесс называется *извлечение признаков* (**feature extraction**). Ключевая точка (локальная особенность) – это такая особая точка, которая достаточно хорошо отличается от близлежащих точек по какой-то определённой характеристике. Она должна быть сильно "непохожа" на остальные точки и являться уникальным свойством

изображения в своей локальной области. Таким образом, машина может представить изображение как модель, состоящую из особых точек. Например, на изображении человеческого лица функции ключевых точек могут выполнять глаза, уголки губ, кончик носа.

К особым точкам предъявляются следующие требования:

1. Повторяемость (Repeatability). Особенность не должна менять свое положение при изменении точки зрения или освещения;
2. Значимость (Saliency). Каждая особенность должна иметь уникальное описание;
3. Компактность (Efficiency). Количество особенностей должно быть существенно меньше числа пикселей изображения;
4. Локальность (Locality). Особенность должна занимать небольшую область изображения, чтобы снизить вероянтность перекрытия другими объектами.

После выделения особых точек компьютеру нужно уметь их сравнивать (отличать друг от друга). Этот процесс называется *сопоставление признаков* (**feature matching**). Для сравнения удобно использовать *дескрипторы* (**descriptor** – "описатель"). Дескриптор – своеобразный идентификатор ключевой точки, представляющий её в удобном для сравнения и понятном для машины виде. Дескриптор является вектором, содержащим признаки особой точки. Именно благодаря дескрипторам получается инвариантность относительно преобразований изображений.

2.2 Последовательность шагов процесса SFM

На рисунке 2.1 представлена схема, демонстрирующая процесс восстановления 3D модели поверхности.

Можно выделить следующие составляющие процесса реконструкции:

1. Выделение ключевых точек и дескрипторов;
2. Сравнение дескрипторов и нахождение паросочетаний соответствующих друг другу особых точек;
3. Нахождение геометрического преобразования, которое переводит ключевые точки одного изображения в соответствующие им точки другого изображения;
4. Позиционирование камер (изображений) и расположение их в трехмерном пространстве.

Далее будут подробнее рассмотрены описанные выше этапы и используемые алгоритмы. Алгоритмы, применяемые на первых двух этапах, называют *алгоритмами, основанными на особых точках* (**feature-based algorithms**).

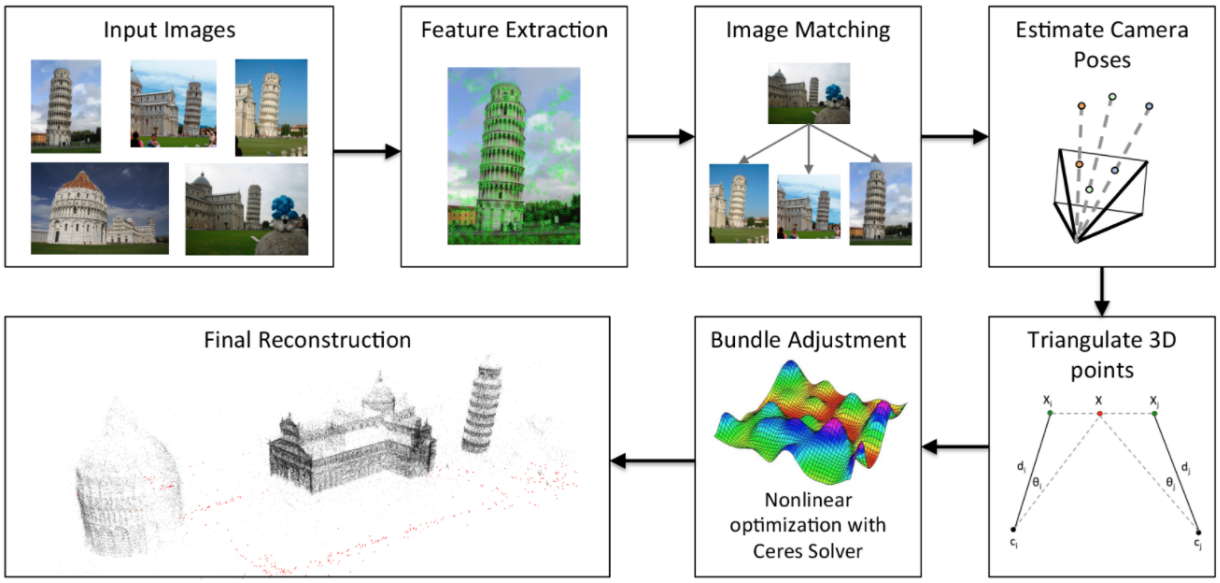


Рисунок 2.1 — **Последовательность шагов процесса SFM**

2.3 Детектор Харриса

Детектор Харриса (*Harris Corner Detector*, 1988 год) – один из первых и самых популярных методов извлечения особенностей. Является детектором углов [1].

Определение углов – метод используемый в компьютерном зрении для извлечения особенностей. Угол, в общем случае, может быть определен как пересечение двух ребер. Также угол может быть определен как точка, для которой есть два разных доминирующих направления градиента в локальной окрестности точки. Таким образом, угол подходит под многие определения особенности и может быть использован в качестве особой точки.

Метод основан на основном свойстве угла – выявлении доминирующих градиентов, что следует из определения угла. Визуально его можно описать как сканирование изображения путём движения окна в локальной области какой-то точки и распознавание изменения яркости при различных сдвигах сразу в нескольких направлениях – характерная особенность угла.

Пусть I – это изображение, (x, y) – точка на нём, являющаяся центром окрестности. Для того чтобы измерить изменение окрестности, локальная область передвигается на $(\delta x, \delta y)$ и разница вычисляется по формуле:

$$f(\delta x, \delta y) = \sum_{x,y} w(x, y)[I(x + \delta x, y + \delta y) - -I(x, y)]^2, \tag{2.1}$$

где $w(x, y)$ – функция веса для элемента суммы.

Чаще всего функция веса полагается равным нормальному распределению: чем ближе точка к центру окрестности, тем больший вклад она вносит,

таким образом отбрасываются граничные значения. Получается что приближённое значение изменения яркости равно:

$$f(\delta x, \delta y) \approx \begin{bmatrix} \delta x & \delta y \end{bmatrix} M \begin{bmatrix} \delta x \\ \delta y \end{bmatrix}, \tag{2.2}$$

где матрица M – интенсивность изображения вдоль осей x и y соответственно.

Матрица M вычисляется по формуле:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_x^2 \end{bmatrix} \tag{2.3}$$

Далее вычисляются собственные значения результирующей матрицы. Потом в зависимости от их значений определяется является ли рассматриваемая точка особой. Если оба значения маленькие – значит что окрестность не меняется и точка не принадлежит углу. Если одно значение существенно больше другого, значит при движении в одном направлении окрестность сильно меняется, а при движении в другом – остаётся неизменной. Это значит что точка лежит на границе. В случае, когда оба собственных значения матрицы достаточно велики и примерно одинаковы, можно говорить что точка образует угол.

Алгоритм детектора следующий:

1. Вычислить градиент изображения в каждом пикселе;
2. Вычислить матрицу M для каждого пикселя в локальной окрестности;
3. Вычислить собственные значения матрицы M;
4. Отбросить точки, которые не проходят пороговые значения;
5. Найти локальные максимумы функции отклика в заданной окрестности;
6. Выбрать N самых сильных локальных максимумов.

Детектор Харриса является инвариантным к изменению угла поворота, но не инвариантен к изменению масштаба. Это проблему решают детекторы, которые будут рассмотрены далее.

2.4 Алгоритм SIFT

Scale-invariant feature transform (SIFT), метод Лоу – алгоритм компьютерного зрения для выделения ключевых точек и их дескрипторов. Алгоритм был разработан в Университете Британской Колумбии (Ванкувер, Канада) и опубликован Дэвидом Лоу (*David G. Lowe*) в 1999 году [2]. Применяется в следующих задачах: распознавание объектов, позиционирование и

навигация роботов, склеивание изображений (построение панорамных снимков), 3D моделирование, распознавание жестов, трекинг видео (сопоставление перемещения).

2.4.1 Удаление шумов

Перед алгоритмом часто производится предварительная обработка изображения в целях улучшения его качества для последующего анализа. Например, на фотографиях с камер возможно появление шумов. Чтобы их устранить используют гауссовское размытие с маленьким радиусом или медианные фильтры, которые, работая с цифровым сигналом изображения подавляют шумовые частоты.

Фильтрация происходит следующим образом: каждый пиксель заменяется средним взвешенным значением в своей локальной окрестности. Веса, с которыми берутся значения каждого пикселя из локальной окресности, составляют *ядро фильтра*. Операция применения фильтра с ядром g к изображению f называется *свёрткой*, обозначается $f * g$ и вычисляется по формуле:

$$(f * g)[m, n] = \sum_{k,l} f[m - -k, n - -l] * g[k, l] \tag{2.4}$$

В фильтре Гаусса коэффициенты ядра распределены по нормальному двумерному распределению. Параметрами такого фильтра является коэффициент дисперсии σ и размер ядра фильтра r. Обычно размер полагается равным 3σ. Фильр Гаусса является низкочастотным фильтром, то есть пропускает только низкие частоты и подавляет высокие. С помощью фильтра Гаусса достигается эффект расфокусировки изображения.

Существует вид шумов, называемых “соль и перец” – случайные чёрные и белые пиксели на изображении. Для этого вида шума Гауссов фильтр будет работать плохо, так как он просто заменит белые пиксели на средние серые. В таких случаях используют *медианный фильтр*. Он работает следующим образом: все значения яркости пикселей из локальной окрестности раскладываются в ряд, сортируются и в качестве результирующего значения выбирается медиана. За счёт этого достигается устойчивость к одиночным выбросам, которыми и являются шумы “соль и перец”.

2.4.2 Извлечение ключевых точек

SIFT трансформирует исходное изображение в большую коллекцию особых векторов, каждый из которых инвариантен к основным изменениям изображения. Представим, что у нас есть два изображения одного и того же объекта, но с большой разницей в масштабе (рисунок 2.2, где красным цветом обведены области одного масштаба). Вместо того, чтобы извлекать особенности на различных масштабах и потом сопоставлять их между собой, метод

SIFT предлагает гораздо более эффективный способ: найти функцию от области изображения, такую, что для областей, изображающих один и тот же объект даже в разных масштабах, будет возвращать одинаковые значения.

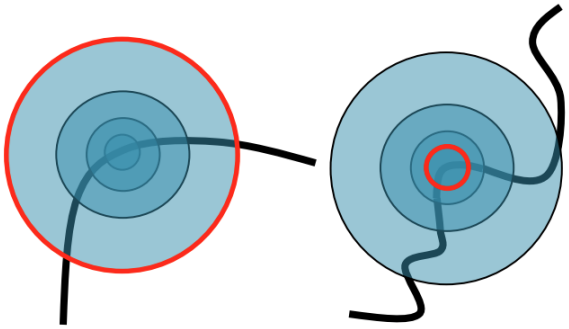


Рисунок 2.2 — Пример области в разных масштабах

Основополагающим моментом в нахождении особых точек является построение пирамиды гауссианов (**Gaussian**) и разностей гауссианов (**Difference of Gaussian, DoG**).

Гауссианом (или изображением, размытым гауссовым фильтром) является изображение:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.5}$$

В уравнении (2.5):

- L - значение гауссиана в точке с координатами (x, y);
- σ - радиус размытия;
- G - гауссово ядро;
- I - значение исходного изображения;
- $*$ - операция свертки.

Разностью гауссианов называют изображение, полученное путем попиксельного вычитания одного гауссиана исходного изображения из гауссиана с другим радиусом размытия:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{2.6}$$

Таким образом, с помощью (2.6), мы получаем набор изображений, являющихся исходным изображением, взятым в разных масштабах. Извлечение ключевых точек, таких что они являются неизменными на всех изображениях из этого набора, будет гарантировать инвариантность относительно сдвига, поворота и изменения размера изображения. Для этого строится пирамида гауссианов (рисунок 2.3 [2]): весь набор масштабированных изображений разбивается на некоторые участки – октавы и при переходе от одной октавы к другой размеры изображения уменьшаются вдвое. После этого строится пирамида разностей гауссианов, состоящая из разностей соседних изображений в пирамиде гауссианов.

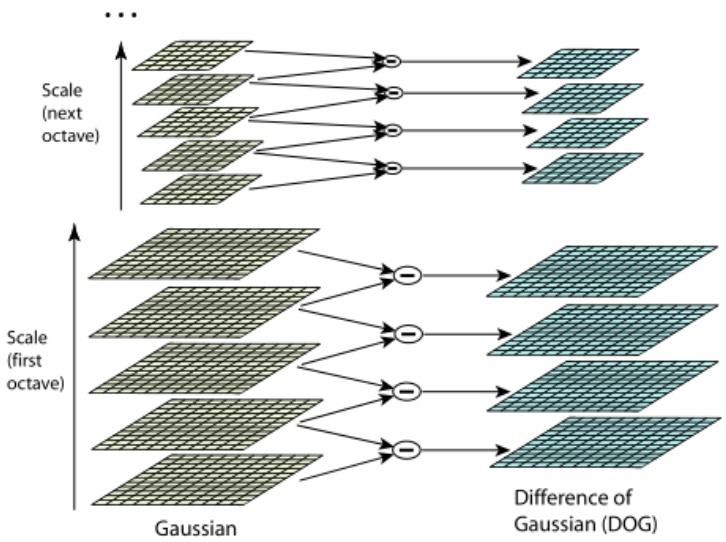


Рисунок 2.3 — Пирамида гаусианнов

После построения пирамиды разностей гауссианов по всем точкам в пирамиде ищутся локальные экстремумы. Если точка больше (меньше) всех своих 26 соседей (рисунок 2.4 [2]) в пирамиде разностей, то она считается ключевой, а масштаб изображения на котором она вычислена берётся как основной.

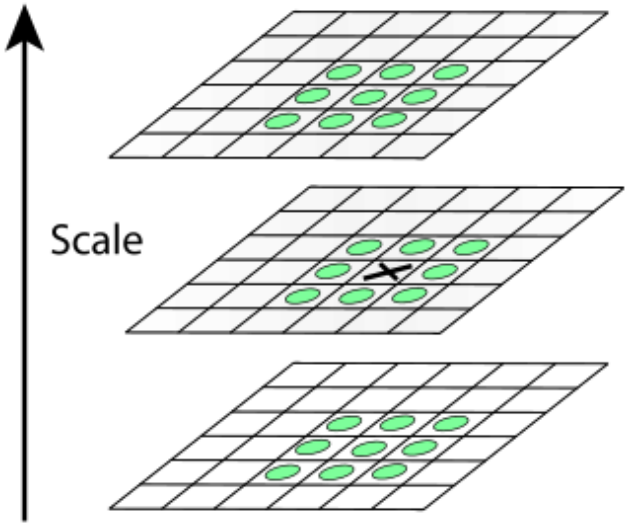


Рисунок 2.4 — Локальный экстремум в пирамиде Гауссианов

Направление особой точки вычисляется на изображении из пирамиды гауссианов в масштабе, полученном на предыдущем шаге. Ориентация ключевой точки – суммарное направление градиентов точек, находящихся в σ-окрестности особой точки. Каждая точка окрестности влияет на итоговое направление. Вся особая область поворачивается так, чтобы доминантное направление градиента было направлено в одном направлении для всех точек, например вверх. Величина и направление градиента в точке (x, y) вычисля-

ются по формулам (2.7) и (2.8) соответственно.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (2.7)$$

$$\theta(x,y) = \arctan\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right) \quad (2.8)$$

Таким образом для исходных изображений разных размеров мы получили ключевые точки (и небольшую область возле них) одного и того же размера, что даёт инвариантность относительно масштабирования. А с помощью ориентации особой точки достигается инвариантность относительно поворота.

2.4.3 Извлечение дескрипторов

Дескриптор должен уникально описывать ключевую точку. В общем случае это может быть любой объект, который будет выполнять следующие функции: быть удобным для сравнения и являться инвариантным относительно преобразований исходного изображения.

В алгоритме SIFT дескриптор представляется как вектор, содержащий информацию об окрестности ключевой точки. Дескриптор вычисляется на том же гауссиане, на котом получен оптимальный размер особой точки. Для достижения инвариантности относительно поворота изображения перед вычислением всю область ключевой точки поворачивают на угол направления ключевой точки, который был найден на предыдущем шаге алгоритма.

На рисунке 2.5 показана окрестность ключевой точки (слева) и построенный для неё дескриптор, состоящий из гистограмм направлений градиентов (справа). Стрелочками в центре каждого пикселя в σ-окрестности обозначен градиент этого пикселя. Как видно на правой стороне изображения, дескриптор имеет размерность 2x2x8 = 32 (количество регионов по горизонтали, количество регионов по вертикали, количество компонент гистограммы этих регионов). Гистограмма для каждого региона является суммарным значением градиентов пикселей, входящих в σ-окрестность (8 значений).

Все полученные гистограммы и составляют итоговый дескриптор ключевой точки. Затем дескриптор нормализуется – все компоненты делятся на максимальное значение – в итоге каждая компонента находится в диапазоне $[0, 1]$. Далее всем компонентам, значение которых больше 0.2, присваивается значение 0.2, а после этого дескриптор нормализуется ещё раз.

Классический размер дескриптора равен 32 компоненты, но на практике больше распространены и активнее используются дескрипторы размерности 128 компонент (4x4x8).

Так как в качестве значений вектора дескриптора используются градиенты, изменение яркости будет отражено в гистограмме, а это значит что дескриптор будет инвариантным к изменению освещения.

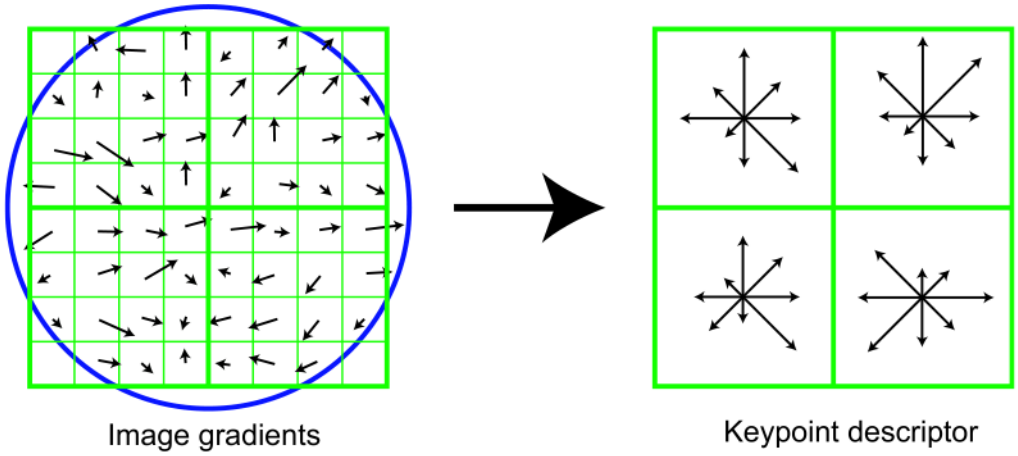


Рисунок 2.5 — Получение дескриптора

2.5 Анализ других алгоритмов, основанных на особых точках

SIFT дескрипторы не лишены недостатков. Не все полученные точки и их дескрипторы будут отвечать предъявляемым требованиям. Естественно, это будет сказываться на дальнейшем решении задачи сопоставления изображений. В некоторых случаях решение может быть не найдено, даже если оно существует. Например, при поиске аффинных преобразований (или фундаментальной матрицы) по двум изображениям кирпичной стены может быть не найдено решения из-за того, что стена состоит из повторяющихся объектов (кирпичей), которые делают похожими между собой дескрипторы разных ключевых точек. Несмотря на это обстоятельство, данные дескрипторы хорошо работают во многих практически важных случаях. SIFT является наиболее математически обоснованным, но относительно медленным алгоритмом.

2.5.1 Дескриптор SURF

В 2008 был представлен ближайший конкурент SIFT дескриптора, SURF (Speeded up robust features) дескриптор [3]. В идейном смысле он похож на своего предшественника, но процедура описания окрестности особой точки несколько иная, поскольку в ней используются не гистограммы взвешенных градиентов, а отклики исходного изображения на *вейвлеты Хаара* (**Haar wavelets**). Вейвлет — математическая функция, позволяющая анализировать различные частотные компоненты данных. Вейвлет Хаара — один из первых и наиболее простых вейвлетов, обладает компактным носителем, хорошо локализован в пространстве, но не является гладким.

На первом шаге получения дескриптора вокруг ключевой точки строится квадратная область, которую ориентируют по некоторому предпочтительному направлению. Затем область разделяется на квадратные сектора. В каждом из секторов в точках, принадлежащих регулярной сетке, вычисляются

отклики на два вида вейвлетов — горизонтально и вертикально направленные. Отклики взвешиваются Гауссианом, суммируются по каждому сектору, и образуют первую часть дескриптора.

Вторая часть дескриптора SURF состоит из сумм модулей откликов. Это сделано для того, чтобы учитывать не только факт изменения яркости от точки к точке, но и сохранить информацию о направлении изменения. SURF-дескриптор имеет длину 64. Как и SIFT, SURF-дескриптор инвариантен к аддитивному изменению яркости. Инвариантность к мультипликативному изменению яркости достигается путем нормировки дескриптора. SURF является эвристическим, но и более быстрым, чем SIFT.

2.5.2 Дескриптор BRIEF

Чем меньше длина дескриптора, тем меньше памяти требуется для его хранения, и, соответственно, тем меньше времени тратится на сравнение их друг с другом. Эта характеристика играет очень значимую роль при обработке большого числа изображений большой размерности. К наиболее компактным относится дескриптор BRIEF (Binary Robust Independent Elementary Features) [4]. Для вычисления дескриптора в точке p сравниваются значения яркости точек, расположенных в ее окрестности. При этом сравниваются значения яркости не всех точек со всеми, а анализируется лишь небольшое подмножество соседних пар точек, координаты которых распределены случайно (но одинаковым образом для каждой анализируемой точки p). Если яркость в точке p_{i1} больше, чем яркость в точке p_{i2}, то i-я компонента дескриптора принимает значение 1, в противном случае она становится равной нулю. Фрагмент, по которому вычисляются дескрипторы, предварительно сглаживается. BRIEF-дескрипторы чрезвычайно просты в вычислении, поскольку их значения равны результату сравнения двух чисел. Они также очень компактны, поскольку результат сравнения —- это число 0 или 1, то есть один бит.

В стандартной реализации для построения одного BRIEF-дескриптора требуется выполнить 256 сравнений, что дает итоговую длину 32 байта. Это очень мало, учитывая, что SIFT-дескриптор состоит из 128 действительных чисел, то есть занимает как минимум 512 байтов. Наконец, сравнение BRIEF-дескрипторов производится очень быстро, поскольку сводится к вычислению *расстояния Хэмминга* между двумя последовательностями битов. Расстояние Хэмминга (**Hamming distance**) —- число позиций, в которых соответствующие символы двух слов одинаковой длины различны, – вычисляется по формуле:

$$d_{ij} = \sum_{k=1}^{p} |x_{ik} - x_{jk}| \tag{2.9}$$

Эта элементарная операция выполняется чрезвычайно быстро на любом современном процессоре. Сами по себе дескрипторы BRIEF не инвариантны к повороту. Однако такой инвариантности можно добиться, если предварительно повернуть фрагмент вокруг ключевой точки на угол, соответствующий, например, доминирующему направлению градиента яркости, как это делается для дескрипторов SIFT и SURF. Точно так же можно достичь инвариантности к другим ракурсным искажениям.

2.5.3 Дескриптор GLOH

Дескриптор GLOH (Gradient location-orientation histogram) является модификацией SIFT-дескриптора и построен с целью повышения надежности [5]. По факту вычисляется SIFT дескриптор, но используется полярная сетка разбиения окрестности на бины (рисунок 2.6): 3 радиальных блока с радиусами 6, 11 и 15 пикселей и 8 секторов. В результате получается вектор, содержащий 272 компоненты, который проецируется в пространство размерности 128 посредством использования анализа главных компонент. Дескриптор GLOH наряду с SIFT является одним из самых точных методов.

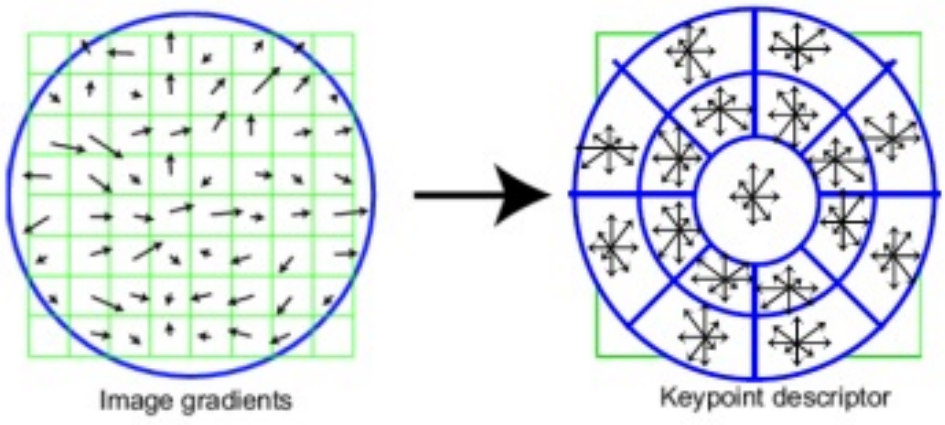


Рисунок 2.6 — Полярная сетка разбиения на бины

2.5.4 Детектор FAST

FAST (Features from accelerated segment test) – алгоритм детекции ключевых точек.

Детектор считает пиксели в круге Брезенгема (находится с помощью алгоритма Брезенгема построения кривых 2-го порядка) радиуса r вокруг точки кандидата. Если n смежных пикселей ярче чем центр, по крайней мере, в t раз или темнее центра, то пиксель в центре считается особенностью. Хотя r в принципе, может принимать любое значение, чаще всего используется значение $r = 3$, которому соответствует круг 16 пикселей окружности. Тесты

показывают, что оптимальное значение $n = 9$. Это значение n наименьшее, при котором края не обнаруживаются.

2.5.5 Дескриптор ORB

ORB (Oriented FAST and rotated BRIEF) – ещё один алгоритм основанный на детекторе ключевых точек FAST и бинарных дескрипторах BRIEF [6]. Как следует из названия, ORB дополняет и улучшает алгоритмы, на которых был основан.

Алгоритм был предложен Ethan Rublee в 2010 году. Также как и BRIEF, ORB имеет размер 32 байта и для сравнения использует расстояния Хэмминга. После детектирования точек с помощью FAST-а ORB выделяет N лучших точек, используя меру Харрисса. Далее ORB ориентирует найденные ключевые точки, что также отражено в его названии. Так как BRIEF плохо работает с поворотом, ORB исправляет это с помощью ориентации.

2.6 Алгоритм “Bundle adjustment”

Bundle adjustment (дословно “*регулировка пучков*”) – алгоритм проективной геометрии который, решая системы нелинейных уравнений, находит 3D координаты ключевых точек в пространстве. Используется на последних этапах процесса Structure from Motion.

Одна трёхмерная точка в реконструируемой модели соответствует нескольким двухмерным точкам на исходных изображениях, потому что на снимках изображена одна и та же местность или объект с разных ракурсов. Если спроецировать 3D точку на изображения – лучи должны попасть в соответствующие ей 2D точки. И, в свою очередь, все лучи должны собраться в один “пучок” в точке на трёхмерной модели.

Суть метода Bundle adjustment:

1. Представление всех лучей, которые должны сойтись в одной точке как систему алгебраических уравнений;
2. Нахождение ошибки между проекцией и реальной точкой на изображении;
3. Минимизация этой ошибки путем решения систем нелинейных уравнений и передвижения камер (изображений);
4. В конечном итоге получение такого расположения исходных изображений в 3D пространстве, что лучи соответствующие одной точке собираются в один пучок с минимально возможной ошибкой.

На рисунке 2.7 проиллюстрировано, как в процессе работы алгоритма меняются расположения исходных камер для нахождения истинных координат трёхмерной точки.

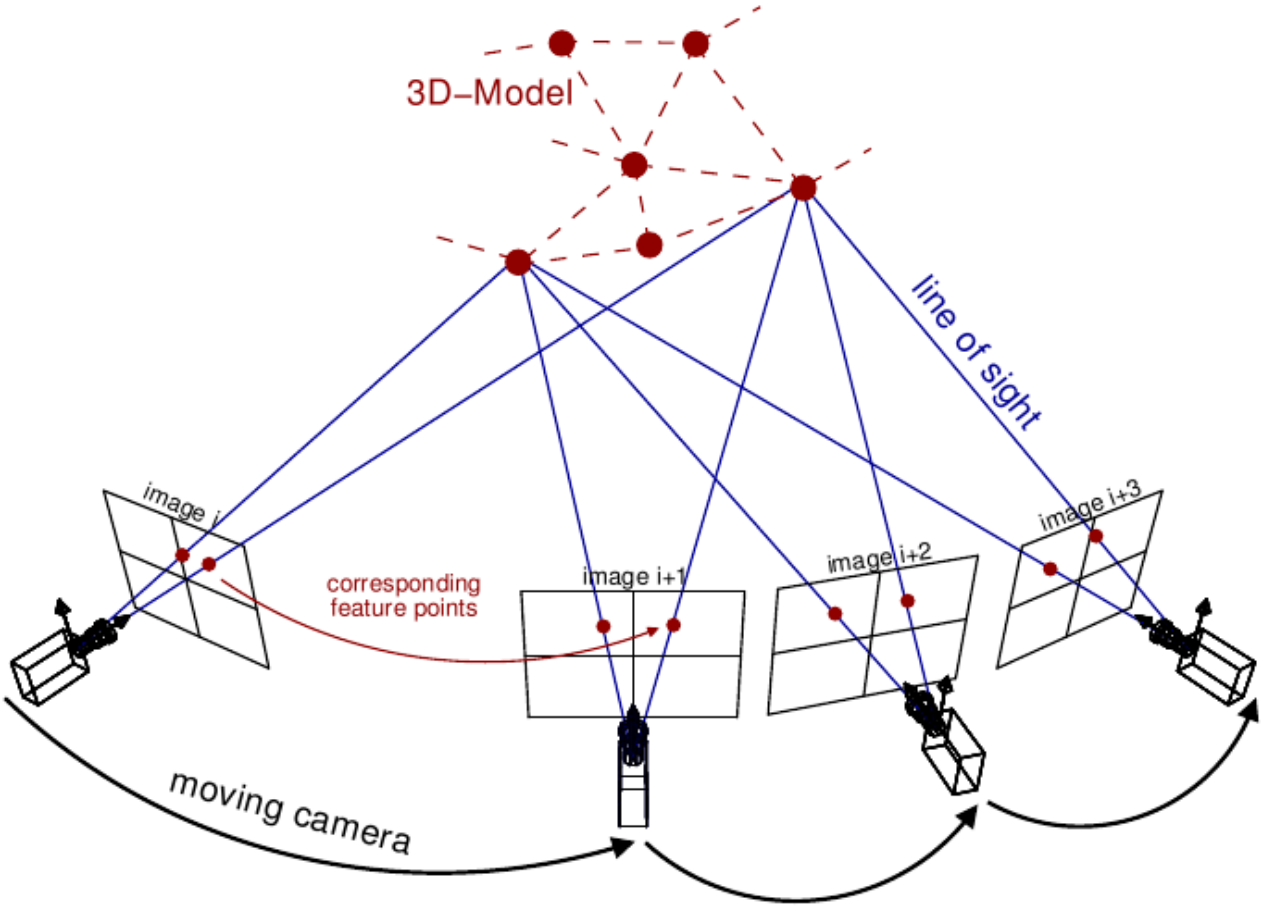


Рисунок 2.7 — Пример работы алгоритма Bundle adjustment

2.7 Выводы

В этой главе был рассмотрен метод построения 3D реконструкции из набора снимков, называемый Structure from Motion. Были разобраны составляющие его этапы и проанализированы различные алгоритмы, которые могут использоваться на каждом шаге метода. Среди рассмотренных дескрипторов и детекторов можно выделить SIFT, как самый точный и надёжный, и современный ORB, как быстрый, но в то же время достаточно устойчивый.

Как показали проведённые практические эксперименты, Structure from Motion больше применим для офлайн построения больших карт местности и 3D реконструкций, чем для работы в реальном времени на борту. Он точный, но медленный. Так как для успешной навигации требуется онлайн работа, то это приводит к необходимости проведения дальнейших исследований и поиска подходящего метода. Метод SLAM будет рассмотрен в следующей главе.

ГЛАВА 3 МЕТОД SLAM

3.1 Описание метода

Simultaneous Localization and Mapping (SLAM) – метод одновременной локализации и построения карты, первоначально предложенный Питером Чесманом (*Peter Cheeseman*) и Рэндалом Смитом (*Randall C. Smith*) для задачи ориентации колёсных роботов с лазерными сенсорами в плоских пространствах и опубликованный в 1986 году [7].

В общем случае SLAM решает проблему построения или обновления карты неизвестной окружающей среды и одновременной навигации по ней. Используется в различных средах и с различными роботизированными устройствами, например, self-driving автомобили, домашние роботы-пылесосы, планетоходы, воздушные и даже подводные беспилотные аппараты.

Существует много различных реализаций и вариаций данного метода. Это связано с тем фактором, что алгоритм сильно зависит от окружающей среды, в которой предполагается его использование. Так, например, выделяют среды с многочисленными ярко выраженными ориентирами (ландшафты с отдельно стоящими деревьями) и с их отсутствием (коридоры, комнаты). Так же алгоритмы разделяют по типу строящейся карты – плоская или трёхмерная. В первом случае может строиться карта препятствий – массив, где элементы, отражающие положение препятствий, имеют значение 1, а все остальные — 0, а во втором – облако точек, описывающее окружающий мир.

SLAM представляет “проблему курицы и яйца”: карта нужна для навигации по ней, но в то же время оценка текущего положения нужна для построения карты. Несмотря на это существуют алгоритмы, решающие эту проблему в реальном времени.

Метод SLAM можно разбить на этапы:

1. Извлечение ориентиров (*Landmarks extraction*);
2. Объединение (сопостовление) ориентиров с разных позиций (*Data association*);
3. Оценка положения (*State estimation*);
4. Обновление положения и ориентиров (*State and landmarks update*).

Одной из важнейших составляющих SLAM процесса, является получение сведений о текущей позиции робота через одометрию. Одометрия – использование данных о движении приводов, углах поворота колёс/лопастей, текущей скорости, показаний гироскопа для оценки перемещения. Одометрия даёт лишь приблизительное положение, которое SLAM уточняет, основываясь на ориентирах.

Следующие требования предъявляются к ориентирам:

1. Они должны легко выделяться с разных позиций используемым средством восприятия окружающей среды;

2. Они должны быть уникальны и легко отличимы друг от друга;
3. Их должно быть достаточное количество в окружающей среде;
4. Они должны быть стационарными.

При использовании для навигации БПЛА камеры под все описанные требования подходят ключевые точки, которые мы рассматривали ранее. Они могут быть выделены и сопоставлены с помощью уже рассмотренных алгоритмов SIFT или ORB. В таком случае SLAM называют основанным на особых точках (*feature based*).

Рассмотрим работу алгоритма на примере:

1. Робот выделяет ориентиры и их позиции;
2. Робот двигается. Через одометрию он вычисляет позицию, в которой он "думает", что находится;
3. Робот опять оценивает положения ориентиров, но получает, что они находятся не там, где он "думал" на предыдущим шаге. Это означает, что положение, полученное через одометрию, не является точным;
4. Через данные о действительном расположении ориентиров обновляется позиция робота, то есть происходит уточнение;
5. Положения ориентиров сохраняются для дальнейшей навигации по ним, алгоритм переходит на первый шаг.

Для возможности работы алгоритма в реальном времени обработка каждой следующей порции данных (ориентиров, кадров с камеры) должна выполнятся до того, как будут получены новые данные. Это и отличает SLAM от рассмотренного ранее SFM, где на вход поступает сразу большое количество изображений и долго обрабатываются в режиме офлайн.

Для триангуляции и расположения камер используется рассмотренный ранее Bundle Adjustment. Проблема в том что время его работы быстро растёт с увеличением размера построенной карты. Решается данная проблема с помощью разнесения задач локализации и построения карты по разным процессам. Ориентация происходит в реальном времени, а Bundle adjustment перестраивает карту в фоновом режиме. По завершению работы фоновый процесс обновляет карту для процесса ориентации, а тот, в свою очередь, добавляет новые ориентиры для уточнения карты и это повторяется бесконечно.

3.2 Сравнение различных SLAM-алгоритмов

Все рассмотренные в данном разделе алгоритмы и их реализации находятся в открытом доступе и их исходный код можно найти на GitHub.

3.2.1 Метод Parallel Tracking And Mapping

Parallel Tracking And Mapping (PTAM) – визуальный метод, адаптирующий SLAM для области дополненной реальности [8].

Впервые в методе PTAM в 2007 году было предложено распараллелить процессы локализации и построения карты. Является прародителем современных быстрых SLAM решений. Как и большинство визуальных методов использует Bundle Adjustment и хранит карту как облако точек.

3.2.2 Метод Large Scale Direct SLAM

Large Scale Direct SLAM (LSD-SLAM) – плотный прямой монокулярный SLAM метод, который вместо использования ключевых точек оперирует непосредственно интенсивностью изображения (числовыми значениями пикселей) [9].

LSD-SLAM параллельно запускает три функции: трекинг (локализация), построение карты и оптимизация карты. В отличие от методов, основанных на ключевых точках, прямой прямой метод не представляет изображения в виде абстракции (набора ключевых точек), а сохраняет полные изображения.

Вместо минимизации ошибки проекций точек в этом прямом методе минимизируется попиксельная разность (фотометрическая разность) на ключевых кадрах. Для построения и уточнения карты оценивается попиксельная глубина вместо оценки особенностей изображений, таких как 3D точки и векторы нормали.

Также в прямых методах за счёт сохранения большего количества точек получается гораздо более плотная и полная 3D реконструкция.

LSD-SLAM работает в реальном времени даже на CPU, что позволяет использовать его даже на современных смартфонах. Также он является монокулярным: использует только одну камеру. Поэтому он получил широкое распространение в сфере дополненной реальности.

3.2.3 Метод ORB SLAM

ORB SLAM – монокулярный метод, опубликованный в 2015 и основанный на особых точках [10]. Как следует из названия алгоритм использует для извлечения ключевых точек рассмотренный ранее детектор ORB (Oriented FAST and Rotated BRIEF). Благодаря высокой скорости ORB (извлечение особых точек за милисекунды) возможна работа в реальном времени, а инвариантность ORB к поворотам камеры и смене освещения обеспечивает надёжность метода.

В настоящее время данный метод называют лучшим из всех SLAM, основанных на особых точках.

3.2.4 Метод Semi-direct Visual Odometry

Выбирая между прямыми и основанными на особых точках методами, некоторые разработчики решили извлечь все лучшие стороны из обоих подходов.

Semi-direct Visual Odometry (SVO) – полупрямой SLAM метод [11]. Карта в этом подходе представляет собой координаты особых точек, вычисленых с помощью алгоритма FAST. С другой стороны, как и в прямых методах, для оценки используется минимизация разницы глубин, поэтому FAST особенности помещаются в глубинный фильтр и позднее используются для такой оценки. SVO разрабатывался специально для работы на борту микролетательных аппаратов. Как утверждают разработчики алгоритм работает в реальном времени на скорости 55 кадров в секунду на встроенном бортовом компьютере и 300 кадров в секунду на обычном ноутбуке.

3.3 Выводы

В этой главе был рассмотрен более современный метод построения 3D карты местности. Было проведено сравнение с рассмотренным ранее методом SFM.

SLAM – очень активно развивающаяся в последнее время область с большим количеством работ и достойных реализаций. Используется как для навигации, так и в новых сферах информатики и программного обеспечения, таких как виртуальная и дополненная реальности.

Методы LSD-SLAM и ORB-SLAM разными подходами добиваются возможности работы в реальном времени и могут быть использованы для навигации БПЛА. SVO-SLAM разработан и оптимизирован специально для применения на маломощных устройствах, таких как дроны.

ГЛАВА 4 ЭКСПЕРИМЕНТАЛЬНЫЕ РЕЗУЛЬТАТЫ

4.1 Подготовка данных

Для проведения экспериментов были получены данные видеосъёмки дроном Phantom DJI 4. Из этих видеороликов мной были подготовлены *наборы данных* (**datasets**). Мной было экспериментально установлено, что для того чтобы получить 70% перекрытие на соседних изображениях (необходимое условие для хорошего сопоставления) требуется нарезать видео с частотой хотя бы 1 кадр в 2 секунды. Отдельно рассматривались прямые и обратные (в противоположную сторону) пролеты БПЛА над одной и той же местностью, для возможности имитации задачи возвращения дрона домой по построенной карте. Пример полученной раскадровки представлен на рисунке 4.1.

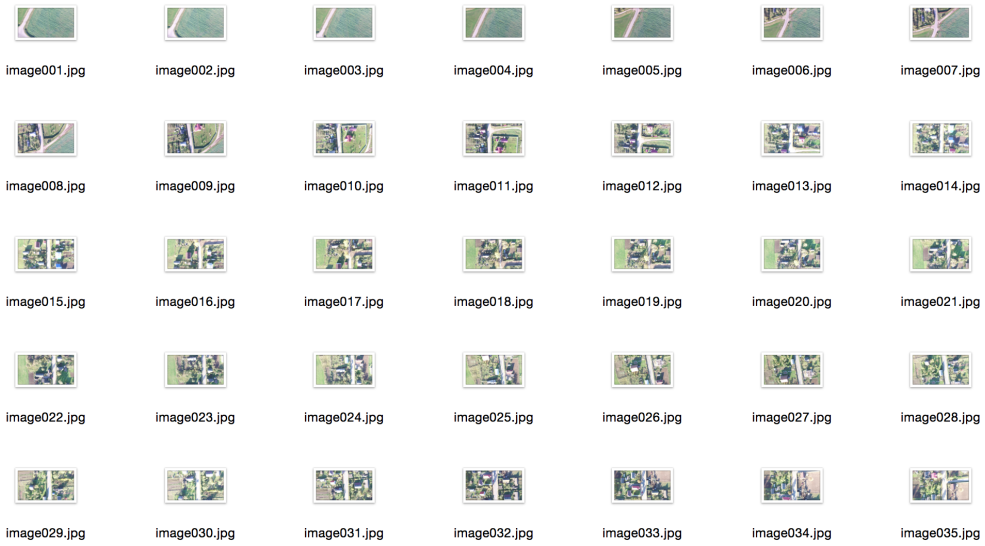


Рисунок 4.1 — Снимки, полученные с БПЛА

4.2 Эксперименты по сопоставлению изображений

Как первое приближение используется решение “в лоб”: каждый снимок, сделанный на прямом пролете (кривая AB), сопоставляется с каждым снимком из обратного пролёта (кривая BA). Для n прямых снимков и m обратных получаем $n * m$ сравнений. В итоге для каждый пары снимков получаем *результат сравнения* (**score**) их ключевых точек, на основе которого можно судить, соответствуют ли эти снимки одной и той же точке в пространстве.

Этот алгоритм для экспериментов был реализован мной на языке программирования *Python*, с использованием библиотеки OpenCV [15]. В экспериментах я использовал дескрипторы SIFT и ORB. В качестве параметров

программа принимает название алгоритма и максимальное количество ключевых точек, которые будут выделятся на изображении. Далле на рисунках представлены результаты работы программы. На рисунке 4.2 красным цветом показаны выявленные ключевые точки, а на рисунке 4.3 эти точки сопоставлены на двух изображениях и совпадения соединены зелёной линией.

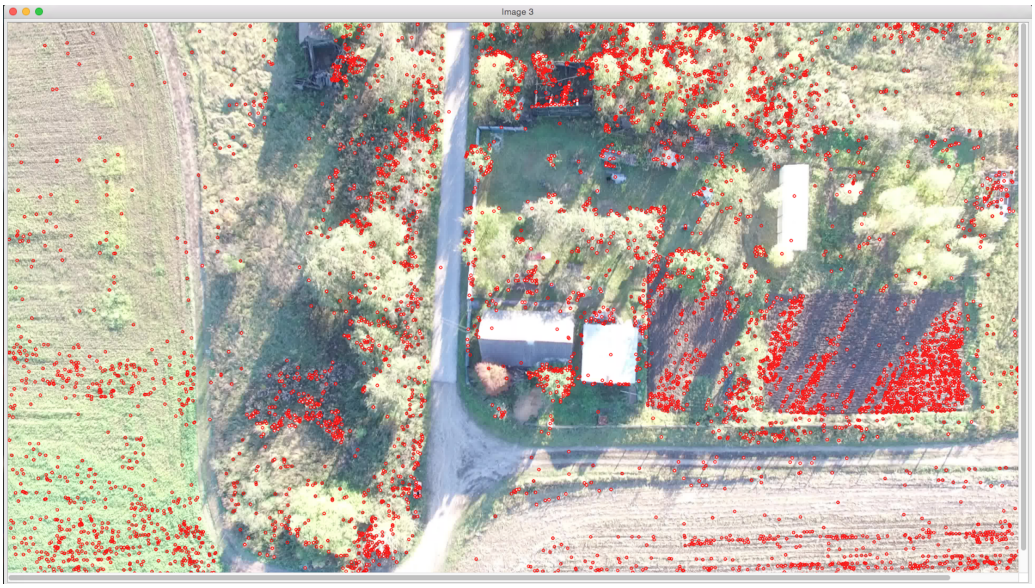


Рисунок 4.2 — Найденные ключевые точки

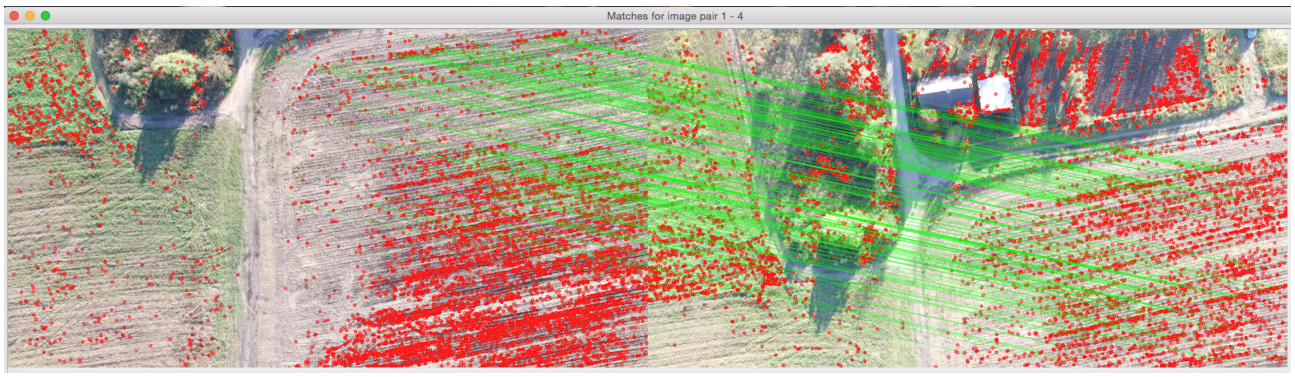
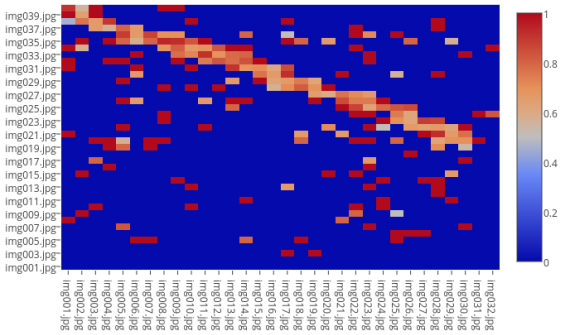


Рисунок 4.3 — Совпадения ключевых точек

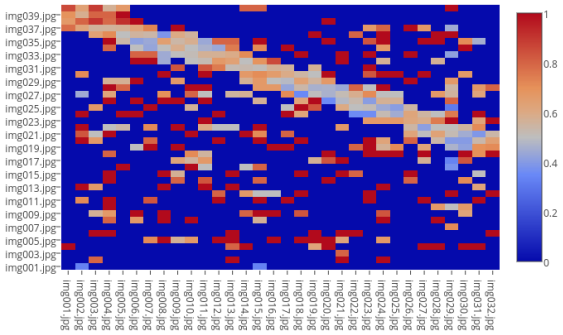
Варьируя количество выделяемых ключевых точек и используя разные алгоритмы, были проведены эксперименты. Для визуализации качества сопоставления были построены *тепловые карты* (**heatmaps**) – диаграммы показывающие насколько хорошо совпадает изображение x_i с y_j (расположены соответственно на осях абсцисс и ординат). Чем выше score сопоставления x_i с y_j изображения, тем "теплее" цвет пикселя на диаграмме.

Сопоставление можно считать успешным, если на диаграмме хорошо прослеживается траектория: так как при пролётах туда-обратно мы должны получать, что на x_0 и y_n, x_1 и y_{n-1} ... и т.д. видны одни и те же точки пространства. Также анализировалось время работы программы. На рисунке 4.4

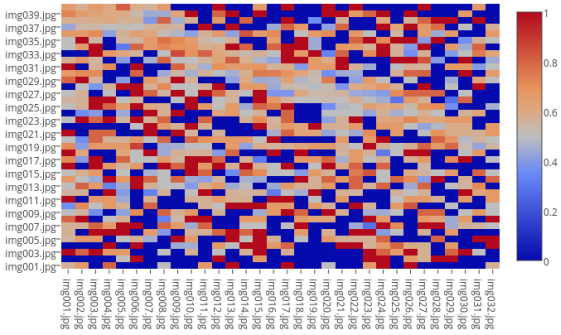
представлены полученные результаты на наборах из 40x32 изображений (1280 сравнений). Подписи к изображеням: *алгоритм – количество точек – время работы*.



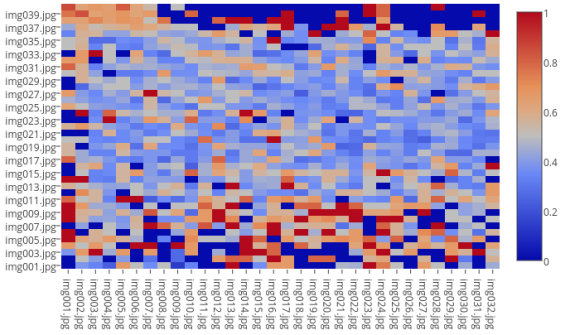
(a) ORB – 200 точек – 18 мин.



(b) SIFT – 200 точек – 3,7 ч.



(c) ORB – 1000 точек – 26 мин.



(d) SIFT – 1000 точек – 4 ч.

Рисунок 4.4 — Сравнительные тепловые диаграммы

4.3 Выводы

Анализ полученных результатов позволяет сделать следующие выводы:

- при малом количестве выделяемых ключевых точек прослеживается траектория полёта;
- *SIFT* даёт много ложных сопоставлений при очень большом времени работы – 3,5 часа против 20 минут у *ORB*;
- при увеличении точек совпадения "размазываются" по всей диаграмме – это значит, что ошибка растёт и нужно улучшать методы feature extraction для получения лучших ключевых точек.

Учитывая время работы и полученные результаты, навигация БПЛА с использованием решения "в лоб", не представляется возможной. Однако это решение может быть использовано в качестве первого приближения, от которого можно отталкиваться в дальнейшем и сравнивать с ним результаты последующей работы.

ГЛАВА 5 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

5.1 Выбор технологий

5.1.1 Библиотека проективной геометрии

К разрабатываемому приложению предъявляются следующие требования: высокая производительность, удобный и кроссплатформенный пользовательский интерфейс, минимум зависимостей.

Для выполнения процесса **Structure From Motion** была выбрана реализация от Криса Суини (*Chris Sweeney*) – библиотека проективной геометрии с открытым исходным кодом Theia [12]. Автор библиотеки, исследователь Вашингтонского университета, занимается разработками в области компьютерного зрения и виртуальной реальности, имеет степень Ph.D., а также множество научных публикаций. Выбор именно этой библиотеки обусловлен несколькими причинами: легковесность (не имеет зависимостей от больших библиотек, таких как OpenCV или Boost), узкая специализация и направленность на решение конкретной задачи, реализация на C++, а также очень хорошая и подробная документация.

5.1.2 Пользовательский графический интерфейс

Для написания графического пользовательского интерфейса отлично подходит библиотека *Qt* [13].

Qt – это кроссплатформенный инструментарий разработки приложений на языке программирования C++, который:

- позволяет запускать написанное с его помощью программное обеспечение в большинстве современных операционных систем (*Windows, macOS, Linux*) путём простой компиляции программы для каждой операционной системы без изменения исходного кода;
- включает в себя множество классов для работы с сетью, базами данных, также предоставлены обширные инструменты по быстрому и удобному созданию интерфейсов;
- использует Metaobject-компилятор, который компилирует программы, написанные на *Qt*, в C++ код, таким образом расширяя функционал и возможности языка C++;
- добавляет очень удобные обёртки для STL контейнеров, такие как: QString, QArray, QList, QLinkedList, QStack, QQueue, QMap и QSet;
- состоит из таких основных модулей как: Qt Core, Qt GUI, Qt QML, Qt Network, Qt SQL и Qt WebEngine;

- включает в себя виджеты (компоненты), такие как: кнопки, прогресс бары, переключатели, чекбоксы – исчерпывающий набор для создания GUI.

В комплекте с *Qt* идёт собственная интегрированная среда разработки QtCreator, а также Qt Designer, который позволяет создавать интерфейс, просто перетягивая и компонуя виджеты и компоненты на форме, а после генерирует каркасы контроллеров.

5.1.3 Система сборки проекта

В качестве системы сборки проекта мной был выбран инструмент Cmake. Cmake – это кроссплатформеная, бесплатная и свободно распространяемая система управления процессами сборки программного обеспечения. Поддерживает иерархическую структуру и вложенные структуры. Позволяет описать все зависимости проекта и правила для их поиска и сборки. Основным достоинством является кэширование файлов, включённое по умолчанию, что позволяет значительно увеличить скорость сборки, путем перекомпилирования только изменённых файлов, а оставшиеся брать из кэша. Это является очень критичным для приложений написанных на C++, для которых время компиляции может занимать до пяти минут.

Для описания сборки в директории требуется создать файл CMakeLists.txt с перечислением списка файлов для компиляции, а также указать ссылки на библиотеки и имя цели, которое в дальнейшем будет использоваться для запуска собранного исполняемого файла.

После завершения своей работы Cmake генерирует специальный Makefile, который используется стандартный командой make для сборки C++ приложений.

5.1.4 Стиль кода

При разработке приложения было решено придерживаться стиля кода, принятого в компании Google. Google C++ Style Guide – самый популярный справочник по стилю, набор правил и советов по написанию чистого, читаемого и расширяемого кода на C++ от компании Google. Для того чтобы точно придерживаться этих правил я использовал cpplint – скрипт, который с помощью набора регулярных выражений сканирует все файлы проекта и проверяет выполнение правил, выводит ошибки в удобном и понятном виде.

5.2 Разработка алгоритма поиска

Итак, после выполнения всех этапов метода Structure From Motion построена реконструкция поверхности в виде 3D модели. Модель представляет

из себя набор точек пространства с известными для них GPS координатами. Цель алгоритма – найти на построенной 3D карте расположение нового снимка. Предполагается, что на снимке присутствует та же область пространства, что и в модели, иначе ничего найдено не будет. Также снимок должен быть не из исходного набора данных (датасета), то есть оличный от тех, на основе которых строилась модель. Иначе поиск не будет иметь смысла. Следующая задача алгоритма – найти геометрическое преобразование и с его помощью определить точные координаты точки пространства, из которой был сделан искомый снимок.

Для осуществления поиска по модели вместе с каждой 3D точкой сохраняется набор дескрипторов всех особых точек, соответствующих этой реальной точке.

В итоге получается следующий алгоритм:

1. На вход поступает очередной снимок;
2. Находим ключевые точки и извлекаем соответствующие им дескрипторы;
3. Сравниваем полученные дескрипторы с сохранёнными в модели;
4. Находим камеру из исходного датасета, для которой получили наилучшее сопоставление;
5. Находим геометрическое преобразование, с помощью которого искомый снимок проецируется на “лучшую” камеру;
6. По известным GPS-координатам исходной камеры и геометрическому преобразованию находим местоположение искомой камеры.

Также, кроме одной камеры, возможно получение всей области, на которую накладывается искомый снимок.

5.3 Приложение для построения реконструкции

На рисунке 5.1 представлен интерфейс разработанного приложения. Модель – швейцарский карьер, построенный на датасете, снятом с помощью беспилотного летательного аппарата. Датасет был взят из открытого источника.

В приложении реализован следующий функционал:

- создание нового/открытие существующего проекта;
- просмотр датасета текущего проекта;
- извлечение ключевых точек;
- построение модели;
- визуализация модели;
- поиск по построенной модели.

Рассмотрим функционал приложения подробнее. При создании проекта надо ввести имя проекта, путь к директории с изображениями и директорию для проекта. В этой директории будет создан конфигурационный файл, со-

Рисунок 5.1 — Приложение для построения и визуализации 3D моделей, осуществления поиска по ним

держащий всю информацию о проекте: пути, настройки и параметры. С этим файлом и будет в будущем ассоциирован проект. Соответственно для открытия проекта требуется открыть папку, содержащую этот конфигурационный файл.

При визуализации модели, красным цветом отрисовываются положения исходных камер, с которых видны ключевые точки. При выборе изображений на боковой панели слева точки выбранного изображения, которые попали в конечную модель, подсвечиваются синим цветом.

При построении модели можно настроить такие параметры как количество потоков, в которых будет выполнятся каждая часть процесса Structure From Motion, тип дескриптора и детектора (поддерживаются SIFT и AKAZE), стратегия сопоставления снимков (Brute Force или Cascade Hashing). Остальные настройки касаются внутренних и внешних параметров камеры. Окно выбора параметров представлено на рисунке 5.2.

После построения модели её можно сохранить в текстовый файл специального формата, по умолчанию – в папку текущего проекта. Благодаря этому уже построенные модели можно открывать и визуализировать очень быстро. В одном проекте может быть много моделей, выбор текущей активной модели

осуществляется при запуске процесса визуализации.

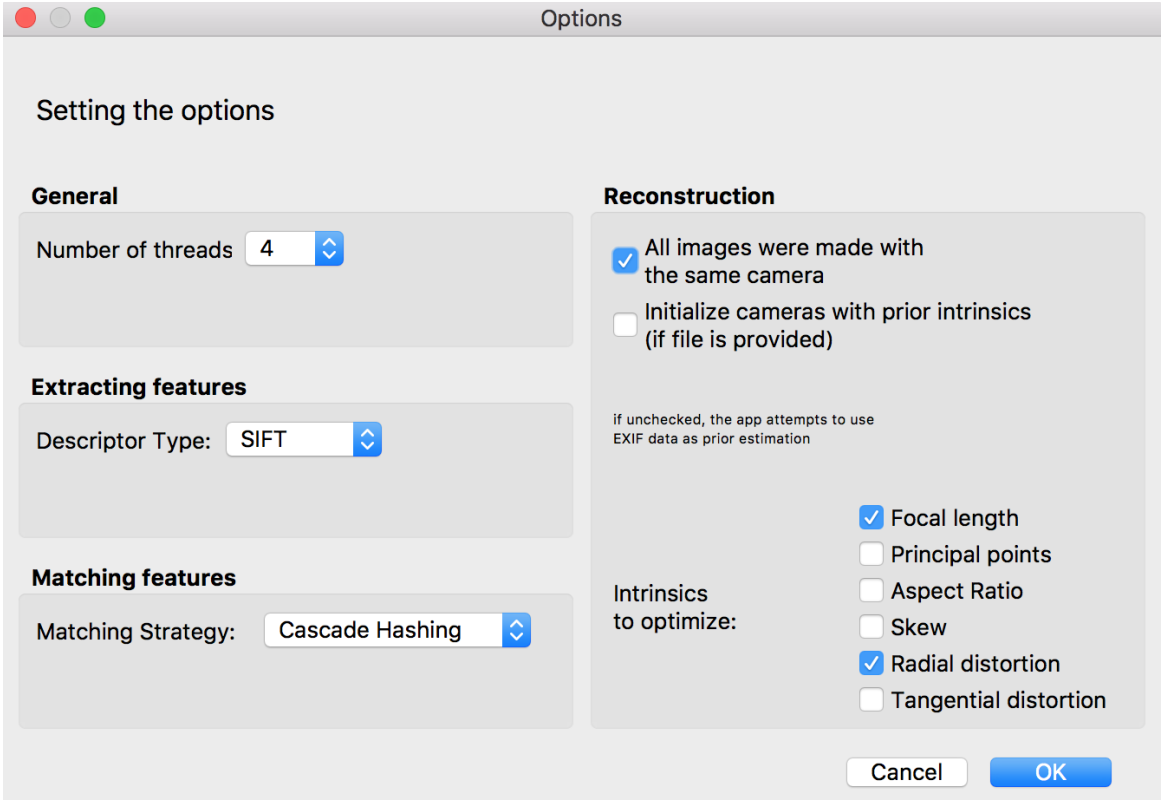


Рисунок 5.2 — Различные параметры построения модели

При запуске процесса поиска открывается диалоговое окно для выбора изображения, которое мы собираемся найти на построенной 3D карте. После выполнения поиска ключевые точки модели, сопоставленные с искомым снимком, подсвечиваются красным цветом. Таким образом можно определить координаты искомого снимка, так как координаты для изображений составляющих карту известны. Тестирование приложения на производительность показало, что поиск на датасете из 127 снимков, при извлечении порядка 5000 ключевых точек на каждом изображении осуществляется, в среднем, за $40 - 50$ секунд.

5.4 Выводы

В этой главе была представлена проделанная практическая работа. Проанализированы новые технологии и решения. Получен результат работы – приложение для построения 3D модели и поиска по ней.

По результатам анализа алгоритма поиска можно сделать вывод: итоговое время лучше полученного экспериментально в начале исследований. Но этого всё ещё недостаточно для стабильной работы в реальном времени на борту беспилотного летательного аппарата. Требуется оптимизация и доработка алгоритма поиска.

ГЛАВА 6 ФРЭЙМВОРК ROBOT OPERATING SYSTEM

6.1 Определение

Операционная система для роботов **ROS** – распределенный фрэймворк для программирования роботов [16]. ROS включает в себя основные функции операционной системы: аппаратную абстракцию, низкоуровневый доступ к устройствам, предоставление стандартных функций для простых задач, управление пакетами. Несмотря на это ROS нельзя назвать операционной системой в традиционном понимании этого определения. Можно сказать, что ROS – это удобная архитектура, а также набор правил и пакетов для её построения. Включает огромное количество свободно распространяемых пакетов, комбинируя которые, можно строить сложные системы (навигация, планирование, восприятие, моделирование и др.). Поддерживается только на операционной системе Linux (Ubuntu).

Основные достоинства ROS:

1. Возможность абстрагироваться от конкретного робота и оборудования для него;
2. Возможность повторного использования кода следующим способом: вынесение его в пакет и использование как зависимость;
3. Удобная отладка и тестирование как на реальном роботе, так и с помощью симулятора;
4. Распределённость, с помощью которой достигается высокий коэффициент масштабирования;
5. Независимость от языка разработки программного обеспечения, поддерживаются такие языки как: C++, Python, Matlab и другие.

6.2 Архитектура системы

В основе ROS лежит система графа. Обработка данных осуществляется в узлах (*nodes*), которые связываются “рёбрами” – сообщениями. Для осуществления коммуникаций в системе существует понятие каналов (*topics*). Каждый узел может как подписаться на получение сообщений из какого-то канала, так и осуществлять публикацию новых каналов и сообщений. Все узлы имеют уникальные имена и могут находить друг друга через специальный сервис – *master*. Каждый узел может предоставлять свой сервис или пользоваться сервисом другого узла. В отличие от сообщений, которые реализуют связь один ко многим, сервис позволяет устанавливать соединение один к одному. Поэтому сервис ожидает подтверждения о получении сообщения от своего подписчика. Сообщения поддерживают вложенные структуры,

массивы, имеют строгую типизацию. Всё это предоставляет дополнительные возможности для удобной передачи данных. Узлы связываются с внешним миром через стандартные потоки ввода и вывода (**stdin**/**stdout**), представленные в виде пакета **rosout**. Все системные узлы объединены в подграф, который называется **roscore**.

6.3 Выводы

Таким образом ROS обеспечивает очень удобную и расширяемую систему для программирования роботов. Например, за работу обычного робота может отвечать сразу несколько условно независимых узлов: один контролирует лазер, с помощью которого робот ищет препятствие, другой – управляет колесами и приводом, ещё один отвечает за позиционирование, а следующий за планирование.

Такое чёткое следование принципу разделения ответственности (*single responsibility*) ведёт к дополнительной отказоустойчивости: так как узлы изолированы, то выход из строя одного не должен касаться остальных. Также это упрощает структуру кода, так как сразу, на уровне архитектуры, декомпозирует задачи и позволяет узлам экспортировать минимальный набор программных интерфейсов.

ГЛАВА 7 ЭКСПЕРИМЕНТЫ С МЕТОДОМ SLAM

Для проверки возможности использования метода SLAM совместно с методами построения и восстановления карты местности было решено имитировать поток данный с БПЛА с помощью внешней веб-камеры, подключённой к порту usb.

7.1 Настройка и калибровка камеры

Для работы с веб-камерой через ROS требуется использовать пакет **cv-camera**, который предоставляет узел, взаимодействующий с камерой и принимающий от неё поток видео.

Как и во всех задачах компьютерного зрения, перед началом работы требуется откалибровать камеру, то есть получить её внутренние и внешние параметры. С помощью калибровки получаются такие параметры, как фокусное расстояние, угол наклона, и принципиальная точка (точка соответствующая центру фотографии), которые образуют так называемую матрицу камеры и коэффициенты дисторсии (искажения).

На рисунке 7.1 представлен процесс калибровки камеры.

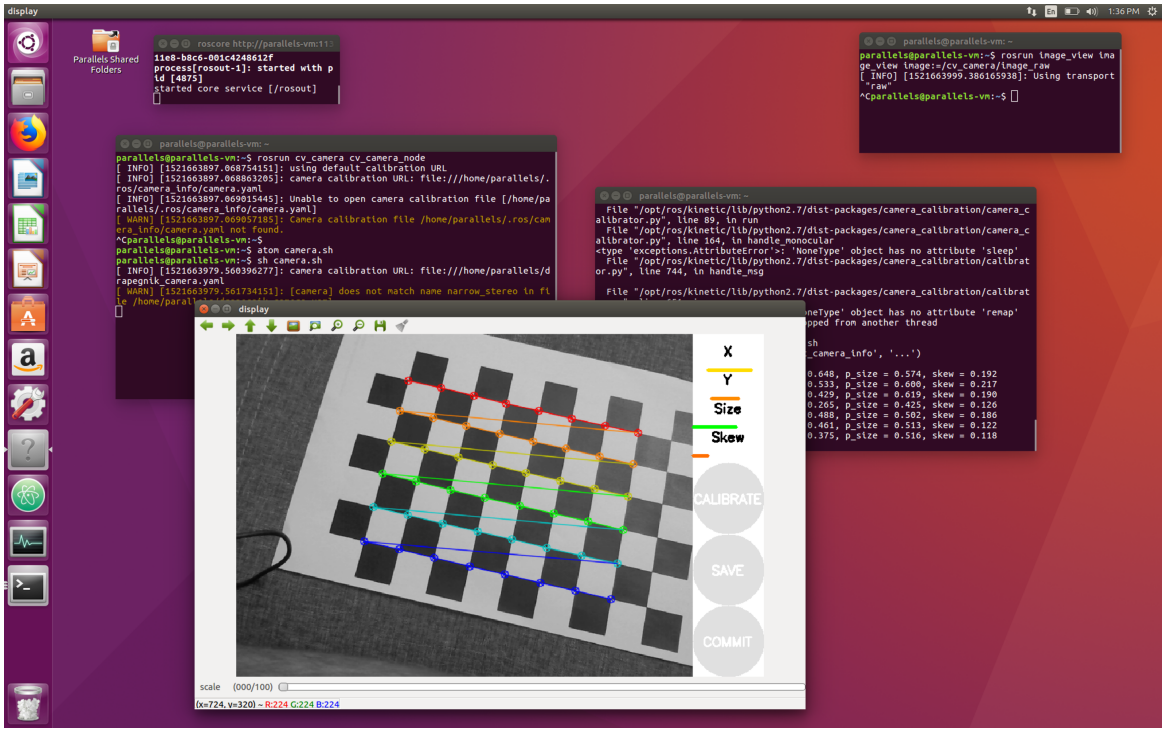


Рисунок 7.1 — Калибровка камеры с помощью изображения шахматной доски

Процедура калибровки представляет из себя следующее:

1. Выбор предмета с известной геометрией, обычно используется изображение шахматной доски;
2. Подготовка 30 или более изображений выбранного предмета с разных ракурсов и расстояний;
3. Определение ключевых точек на полученных фотографиях;
4. Определение коэффициентов дисторсии через минимизацию ошибки;
5. Нахождение остальных параметров – через решение уравнений, полученных путем сопоставления изображений.

Полученные параметры камеры экспортируются в файл и используются для конфигурации SLAM алгоритмов.

7.2 Связывание через ROS

Для связывания компонент приложения вместе и запуска их через ROS будем регистрировать и создавать новые узлы.

Для начала создадим новый узел, который будет отвечать за веб-камеру. Он будет принимать от неё видео поток и публиковать его в специальный канал **/cv_camera/image_raw**. В качестве альтернативы веб-камеры можно использовать снимки из датасетов, которые рассматривались ранее. Для этого напишем простой узел, в котором можно задать частоту кадров в секунду и директорию, в которой находятся снимки. Также можно указать имя канала, в который узел будет публиковать снимки с заданной частотой. Последний вариант предпочтительнее для отладки, так как не требует подключения веб-камеры и её перемещения в пространстве, что является утомительным занятием. Пример работы представлен на рисунке 7.2.

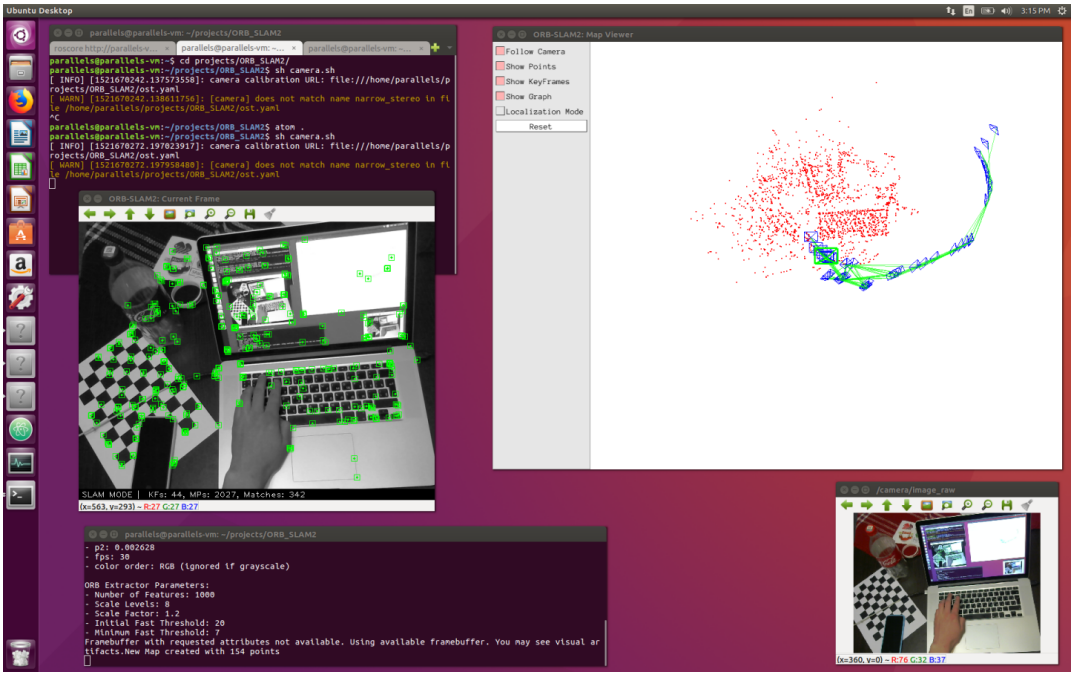


Рисунок 7.2 — Пример работы SLAM с веб-камерой

Далее SLAM запускается как отдельный узел и подписывается на канал с изображениями. После получения данных он отправляет их процессам позиционирования и построения карты ориентиров. При определении положения камеры SLAM публикует эти данные в следующий канал. На этот канал, в свою очередь, подписано приложение, которое при получении данных с камеры визуализирует их и наносит на карту траекторию движения. После этого полученные от SLAM данные используются для инициализации процесса Structure From Motion, что ускоряет процесс построения плотной карты местности.

ЗАКЛЮЧЕНИЕ

Навигация беспилотных летательных аппаратов – сложная и актуальная задача, требующая применения различных подходов. Компьютерное зрение – очень прогрессивная область информатики, которая получает применение в разнообразных сферах, в особенности автоматизации человеческого труда.

В процессе выполнения данной работы мной:

- рассмотрены основные алгоритмы компьютерного зрения;
- проанализированы дескрипторы SIFT, SURF, ORB и проведены сравнительные эксперименты;
- детально разобрана и настроена операционная система для программирования роботов ROS;
- предложен алгоритм восстановления местоположения по 3D карте местности;
- спроектировано и разработано приложение для построения и визуализации 3D карты местности, а также поиска по ней с использованием таких методов как Structure From Motion и Bundle Adjustment;
- с помощью ROS продемонстрирован пример совместной работы алгоритма Simultaneous localization and mapping и разработанного приложения в реальном времени с использованием веб-камеры.

В результате исследований и экспериментов выявлено, что последние реализации алгоритма SLAM, адаптированные для работы с камерами и на небольших мобильных устройствах, являются идеальными кандидатами для решения задачи навигации по 3D карте местности в режиме реального времени. Следовательно они могут быть использованы на борту беспилотного летательного аппарата в условиях отсутствия GPS сигнала. Использование результатов работы SLAM алгоритма в качестве начальных данных для процесса SFM позволяет быстрее получить детальную карту местности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Harris, C. A Combined Corner and Edge Detector // Chris Harris, Mike Stephens // Alvey Vision Conference. – 1988. – pp. 15-65.
2. Lowe, D. Distinctive Image Features from Scale-Invariant Keypoints / D. Lowe // International Journal of Computer Vision – 2004. – no. 60 (2). – pp. 91-101.
3. Bay H., SURF: Speeded up robust features / Bay H., Ess A., Tuytelaars T., Van Gool L. // Computer Vision and Image Understanding – 2008. – no. 110. – pp. 346–359.
4. Calonder M., BRIEF: Binary Robust Independent Elementary Features / Lepetit V., Strecha C., Fua P. // Proc. European Conference on Computer Vision – 2010. – pp. 778–792.
5. Kalal Z., Forward-backward error: automatic detection of tracking failures / Kalal Z., Matas J., Mikolajczyk K. // ICPR'10 – 2010. — pp. 2756-2759.
6. Rublee E., ORB: an efficient alternative to SIFT or SURF / Rublee E., Rabaud V., Konolige K., Bradski G. // IEEE International Conference on Computer Vision (ICCV) [Electronic resource] – 2011. – Mode of access: `http://www.vision.cs.chubu.ac.jp/CV-R/pdf/Rublee_iccv2011.pdf`. – Date of access: 17.12.2016.
7. Smith R.C., On the Representation and Estimation of Spatial Uncertainty / Smith R.C., Cheeseman P. // The International Journal of Robotics Research – 1986. – no. 5 (4). – pp. 56–68.
8. Klein, G. Parallel Tracking and Mapping for Small AR Workspaces / Georg Klein, David Murray. // Proc. International Symposium on Mixed and Augmented Reality (ISMAR) [Electronic resource] – 2007. – Mode of access: `http://www.robots.ox.ac.uk/~gk/publications/KleinMurray2007ISMAR.pdf`. – Date of access: 17.06.2017.
9. Engel, J. LSD-SLAM: Large-Scale Direct Monocular SLAM / J. Engel, T. Schöps, D. Cremers. // European Conference on Computer Vision (ECCV) [Electronic resource] – 2014. – Mode of access: `https://vision.in.tum.de/_media/spezial/bib/engel14eccv.pdf`. – Date of access: 15.10.2017.
10. Mur-Artal, R. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. / Raúl Mur-Artal, J. M. M. Montiel, Juan D. Tardós // IEEE Transactions on Robotics – 2015. – vol. 31, no. 5 – pp. 1147-1163.
11. Forster, C. SVO: Fast Semi-Direct Monocular Visual Odometry / Christian Forster, Matia Pizzoli, Davide Scaramuzza. // IEEE International Conference on Robotics and Automation (ICRA) [Electronic resource] – 2014. – Mode of access: `http://rpg.ifi.uzh.ch/docs/ICRA14_Forster.pdf`. – Date of access: 27.09.2017.

12. Sweeney, C. TheiaSfm Source Code and Documentation // Github [Electronic resource] – 2018. – Mode of access: http://www.theia-sfm.org/. – Date of access: 17.04.2018.
13. The Qt Company, Qt Documentation // The Qt Company Website [Electronic resource] – 2018. – Mode of access: https://www.qt.io/. – Date of access: 20.04.2018.
14. Mordvintsev A., Introduction to SIFT (Scale-Invariant Feature Transform) / A. Mordvintsev, K. Abid // OpenCV Python Documentation [Electronic resource] – 2013. – Mode of access: https://goo.gl/5DqpcN. – Date of access: 10.11.2017. OpenCV Community.
15. OpenCV Community, OpenCV Source Code // GitHub [Electronic resource] / OpenCV Community – 2018. – Mode of access: https://github.com/opencv/opencv. – Date of access: 26.09.2017.
16. Open Source Robotics Foundation, Inc., Robot Operation System Documentation [Electronic resource] – 2018. – Mode of access: http://wiki.ros.org/. – Date of access: 03.05.2018.