МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Кафедра многопроцессорных систем и сетей

РАЗРАБОТКА АЛГОРИТМОВ НАВИГАЦИИ БЕСПИЛОТНЫХ ЛЕТАТЕЛЬНЫХ АППАРАТОВ

Курсовой проект

Пажитных Ивана Павловича
студента 4 курса
специальность “информатика”

Научный руководитель:
Кондратьева Ольга Михайловна
старший преподаватель кафедры МСС

Минск, 2017

РЕФЕРАТ

Курсовой проект, 22 стр., 6 рисунков, 12 источников.

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

Объекты исследования: системы навигации беспилотных летательных аппаратов, алгоритмы компьютерного зрения.

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

Цели работы: изучение возможности применения методов компьютерного зрения в системах навигации беспилотных летательных аппаратах.

Области применения: модели и алгоритмы, работающие на борту беспилотных летательных аппаратов.

Результаты: сравнительные эксперименты, алгоритм решения.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время интенсивно развивается такая область информатики как компьютерное зрение (computer vision). Существует множество алгоритмов по распознаванию и поиску объектов на картинке, сравнения изображений, нахождения геометрического преобразования при помощи которого можно из одного изображения получить другое. Самая популярная библиотека, которая предоставляет реализации основных алгоритмов - решение с открытым исходным кодом OpenCV [1].

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.

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. И это только несколько проектов, а общее количество стартапов в этой области которых увеличивается с каждым днём.

Таким образом компьютерное зрение сейчас - это новая, очень популярная и активно развивающаяся область информатики, используемая всеми лидерами отрасли.

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.

ГЛАВА 1. ОБЛАСТЬ КОМПЬЮТЕРНОГО ЗРЕНИЯ

1.1 Актуальность и практическая значимость

Как следует из названия БПЛА не имеют пилота, но это не значит, что они не пилотируемы. Управление беспилотником требует специального обучения, сосредоточенности и является очень утомительным для оператора. Основополагающим необходимым 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.

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.

С помощью разработанного алгоритма и программного обеспечения будет возможна навигация дрона, используя только камеру. Дополнительные возможности применения обширны: патрулирование заданной территории и обнаружение новых объектов, не присутствовавших ранее, возвращение в заданную точку в случае потери gps сигнала, слежение за данным объектом, построение 3D карт местности, навигация по заданной цифровой карте.

1.2 Постановка задачи

Необходимо изучить и проанализировать различные существующие алгоритмы компьютерного зрения, провести практические эксперименты и впоследствии применить накопленные знания для решения задачи навигации беспилотного летательного аппарата в условиях отсутствия GPS сигнала, с использованием имеющихся с оптических приборов. Ниже будут рассмотрены такие глобальные подходы как Structure from Motion (SFM) и Simultaneous Localization and Mapping (SLAM), с использованием таких алгоритмов как Bundle Adjustment, SIFT, SURF, ORB и их различных реализаций.

1.3 Основные теоретические понятия

Люди обладают зрением, что позволяет нам распознавать изображения и объекты на них, сравнивать их между собой и всё это мы делаем бессознательно, автоматически. Однако, для машины изображение — всего лишь закодированные данные, набор нулей и единиц. Одной из больших проблем в сопоставлении изображений является очень большая размерность пространства, которое несёт информацию. Если взять картинку размером хотя бы

$100 * 100$ пикселей, то уже получим размерность равную 10^4 пикселей. Поэтому методы анализа изображения должны быть быстрыми и эффективными.

Как же компьютер 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, изменения яркости, изменения положения камеры). Чтобы определять 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 - это такая особая точка, которая сильно отличается от близлежащих точек по какой-то обусловленной характеристике. Она должна быть не похожа на остальные точки вокруг, соответственно является, в какой-то степени, 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.

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 виде. Как мы увидим далее, именно благодаря дескрипторам получается инвариантность относительно преобразований изображений.

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 описывает феномен, позволяющий человеку восстанавливать трёхмерную структуру окружающего мира по двухмерным проекциям на сетчатку глаза.

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

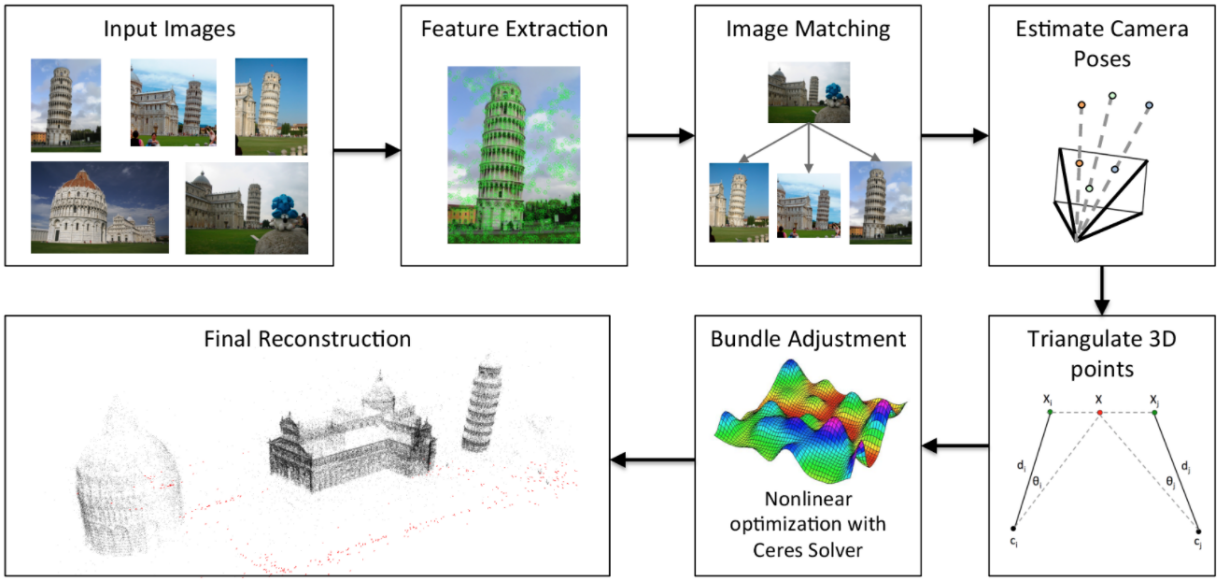

Рисунок 2.1 — **SFM pipeline**

Таким образом 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. Позиционированние камер и расположение их в трехмерном пространстве.

Далее будут подробнее рассмотрены описанные выше этапы и применяемые на них алгоритмы. Для решения первых двух пунктов используются *алгоритмы основанные на особых точках* (**feature-based algorithms**).

2.1 Алгоритм SIFT

Scale-invariant feature transform (SIFT) - алгоритм компьютерного зрения для выделения ключевых точек и их дескрипторов. Алгоритм был разработан в Университете Британской Колумбии и опубликован Дэвидом Лоу (*David G. Lowe*) в 1999 году [3].

На первом этапе 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.

2.1.1 Извлечение ключевых точек

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:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.1}$$

В уравнении (2.1): L — значение гауссиана в точке с координатами (x, y), а σ — радиус размытия. G — гауссово ядро, I — значение исходного изображения, $*$ — операция свертки.

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

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{2.2}$$

Таким образом, с помощью (2.1), мы получаем набор изображений, являющихся исходным изображением взятым в разных масштабах. Извлечение ключевых точке на определённом изображении из этого набора будет гарантировать инвариантность относительно сдвига, поворота и изменения размера изображения.

Как же выбирается нужный масштаб исходного снимка?. Для этого строится пирамида гауссианов (Рисунок 2.2): весь набор масштабированных изображений разбивается на некоторые участки - октавы и при переходе от одной октавы к другой размеры изображения уменьшаются вдвое. После этого

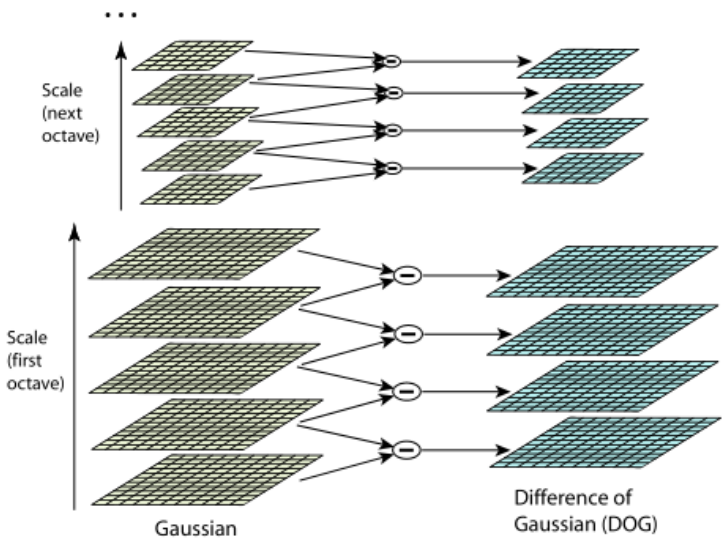

Рисунок 2.2 — Пирамида гаусианнов

строится пирамида разностей гауссианов, состоящая из разностей соседних изображений в пирамиде гауссианов.

После построения пирамиды разностей гауссианов по всем точкам в пирамиде ищутся локальные экстремумы. Если точка больше (меньше) всех своих 26 соседей (Рисунок 2.3) в пирамиде разностей, то она считается ключевой.

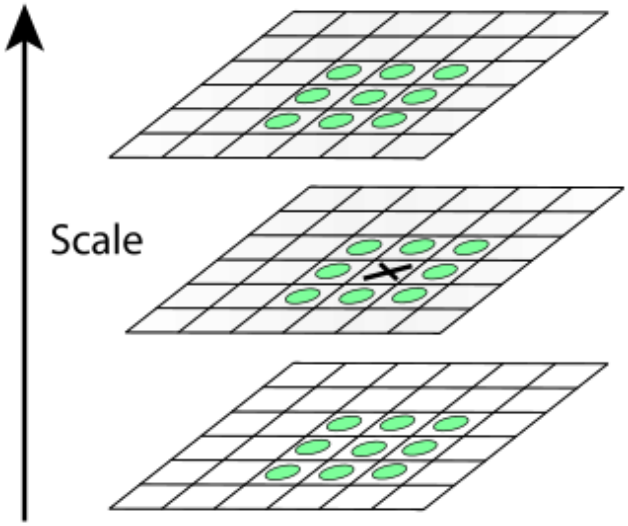

Рисунок 2.3 — Локальный экстремум в пирамиде Гауссианов

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. Величина и направление градиента в точке (x, y) вычисляются

по формулам (2.3) и (2.4) соответственно.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (2.3)$$

$$\theta(x,y) = \arctan\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right) \quad (2.4)$$

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

2.1.2 Извлечение дескрипторов

Как уже говорилось ранее, дескриптор должен уникально описывать ключевую точку. В общем случае это может быть любой объект, который будет выполнять данные 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.

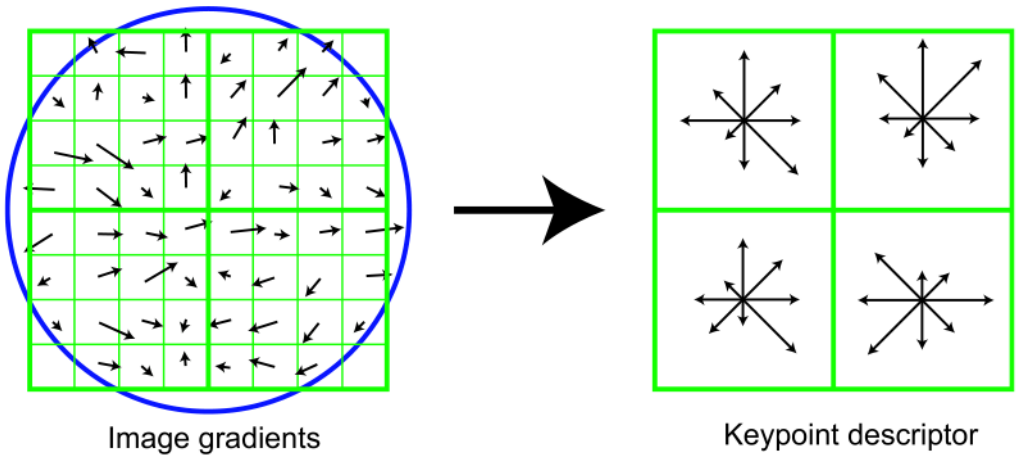

Рисунок 2.4 — Получение дескриптора

На Рисунке 2.4 показана окрестность ключевой точки (слева) и построенный для неё дескриптор, состоящий из гистограмм (справа). Стрелочками в центре каждого пикселя в σ-окрестности обозначен градиент этого пикселя. Как видно на правой стороне изображения, дескриптор имеет размерность

2x2x8 (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 штук).

Все полученные гистограммы и составляют итоговый дескриптор ключевой точки. В конце 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.

Как говорилось ранее, размер дескриптора равен 2x2x8 = 32 компоненты, но на практике больше распространены и активнее используются дескрипторы размерности 128 компонент (4x4x8).

2.2 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.

2.2.1 Дескриптор SURF

В 2008 был представлен ближайший конкурент SIFT дескриптора, SURF [4] дескриптор. 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 каж-

дом 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.

2.2.2 Дескриптор BRIEF

Чем меньше длина дескриптора, тем меньше памяти требуется для его хранения, и меньше времени на сравнение его с другими. Эта черта очень важна при обработке большого числа изображений большой размерности. К наиболее компактным относится дескриптор BRIEF [5]. 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 pi_1 больше, чем яркость в точке pi_2, 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 элементарного теста — 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 64 байта. Это очень мало, учитывая, что SIFT-дескриптор состоит из 128 действительных чисел, то есть занимает как минимум 512 байтов. Наконец, сравнение BRIEF-дескрипторов занимает очень мало времени, 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:

$$d_{ij} = \sum_{k=1}^{p} |x_{ik} - x_{jk}| \tag{2.5}$$

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.

2.2.3 Дескриптор GLOH

Дескриптор GLOH (Gradient location-orientation histogram) [6] является модификацией SIFT-дескриптора, который построен с целью повышения надежности. По факту вычисляется SIFT дескриптор, но используется полярная сетка разбиения окрестности на бины (Рисунок 2.5): 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.

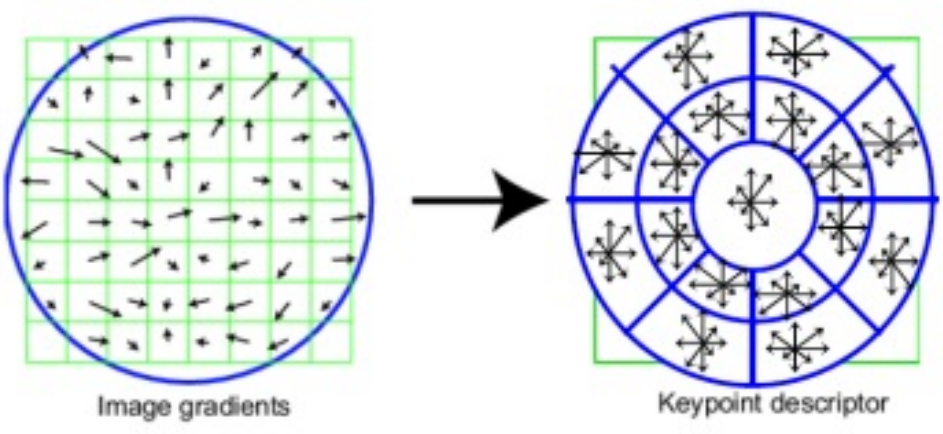

Рисунок 2.5 — Полярная сетка разбиения на бины

2.2.4 FAST детектор

FAST (Features from accelerated segment test - Особенности ускоренных испытаний сегмента) - 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 под центром считается особенностью. Хотя r в принципе, может принимать любое значение, чаще всего используется значение $r = 3$ (которому соответствует круг 16 пикселей окружности), и тесты показывают, что оптимальное значение $n = 9$. Это значение n наименьшее, при котором края не обнаруживаются.

2.2.5 Дескриптор ORB

ORB (Oriented FAST and rotated BRIEF) [7] - ещё один алгоритм основанный на детекторе ключевых точек FAST и бинарных дескрипторах BRIEF. 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 топ 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, полученной на предыдущем шаге.

2.3 Алгоритм "Bundle adjustment"

Bundle adjustment (дословно "*регулировка пучков*") - алгоритм проективной геометрии который, решая системы нелинейных уравнений (путем минимизации ошибки) 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 изображено 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 2.6 проиллюстрировано, как в процессе работы алгоритма меняется расположения исходных камер для нахождения истинных координат трёхмерной точки.

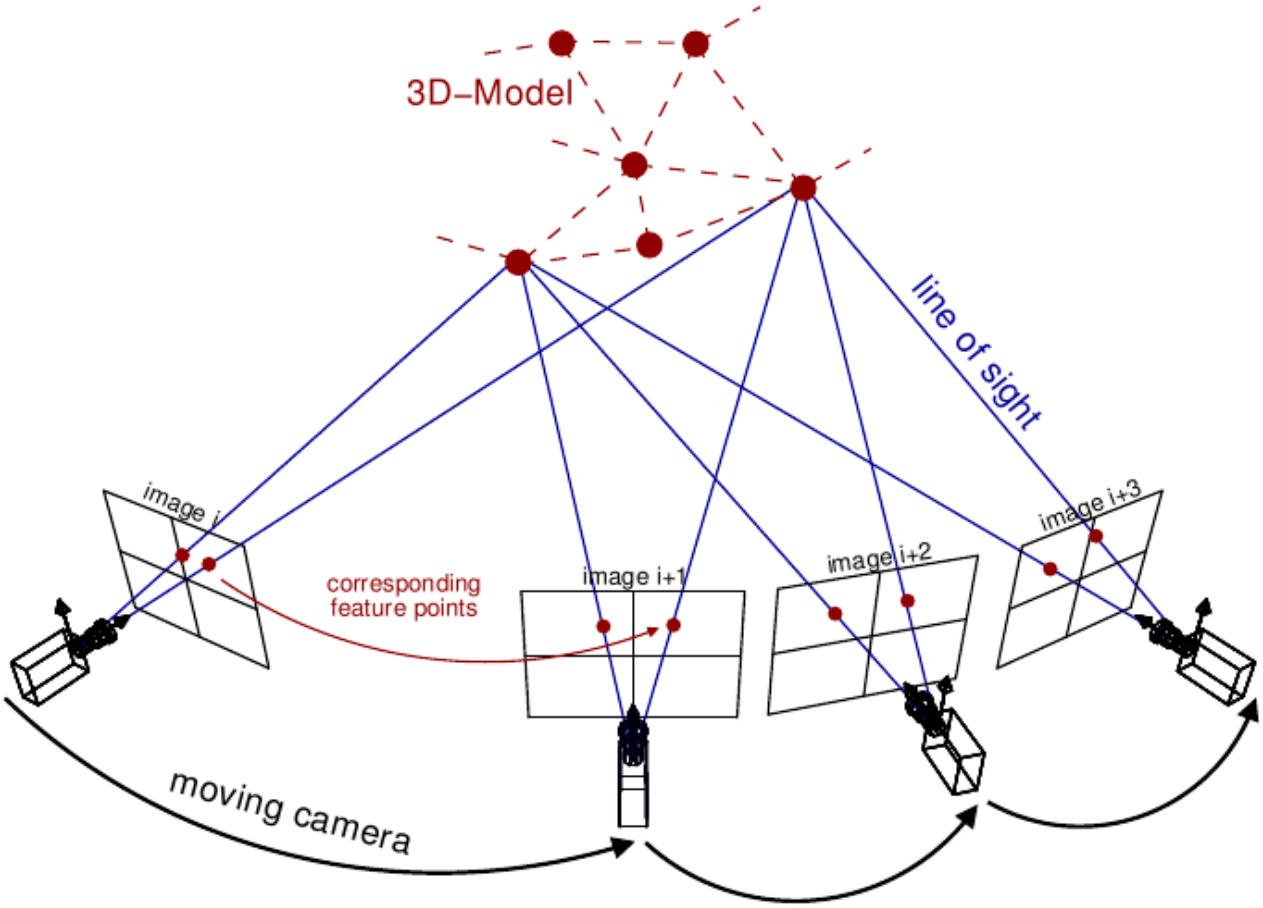

Рисунок 2.6 — **Пример работы Bundle adjustment**

2.4 Выводы

В этой главе был рассмотрен метод построения 3d реконструкции из набора снимков, называемый Structure from Motion. Были подробно 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 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 [8].

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 решений данной проблемы, 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 в себе “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.

SLAM можно разбить на части:

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 состовляющих 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 и 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 "думает" они должны находится по расчётам. Это означает что положение полученное через одеметрию не точное;

4. Через данные о действительном расположении ориентиров обновляется позиция робота - таким образом 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 обрабатываются офлайн за большое количество времени.

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 Parallel Tracking And Mapping

Parallel Tracking And Mapping (PTAM [9]) - визуальный метод адаптирующий SLAM для для области дополненной реальности.

Впервые в 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. Подробнее работа алгоритма была рассмотрена в предыдущей секции.

3.2.2 Large Scale Direct SLAM

Large Scale Direct SLAM (LSD-SLAM [10]) - плотный прямой монокулярный SLAM метод, который вместо использования ключевых точек оперирует непосредственно интенсивностью (числовыми значениями пикселей) изображения.

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 (*augmented reality*).

3.2.3 ORB SLAM

ORB SLAM - опубликованный в 2015 [11] монокулярный метод, основанный на особых точках. Как следует из названия алгоритм использует для извлечения ключевых точек рассмотренный ранее детектор ORB (Oriented

FAST and Rotated BRIEF)[7]. Благодаря высокой скорости ORB (извлечение особых точек за милисекунды) достигается работа в реальном времени, а инвариантность ORB к поворотам камеры и смене освещения обеспечивает надёжность метода.

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

3.2.4 Semi-direct Visual Odometry

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

Semidirect Visual Odometry (SVO [12]) - полупрямой SLAM метод. Карта в этом подходе представляет собой координаты FAST (Features from Accelerated Segment Test) особых точек. 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 микро летательных 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 новый 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. Активно используется как для навигации, так и в новых сферах информатики и программного обеспечения, таких как виртуальная и дополненная реальность.

LSD-SLAM и ORB-SLAM разными подходами добиваются 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.

ЗАКЛЮЧЕНИЕ

В процессе выполнения предыдущей работы было выявлено, что метод Structure from Motion не подходит для работы в реальном времени на борту беспилотного летательного аппарата.

В процессе выполнения данной работы были получены следующие результаты:

- выявлены требования и поставлена задача;
- рассмотрен новый метод Simultaneous Localization and Mapping;
- изучены и проанализированы различные реализации SLAM.

Полученные теоретические знания и практические навыки, а также результаты экспериментов являются хорошей основой для будущей работы. В результате исследования выявлено, что последние SLAM алгоритмы являются идеальными кандидатами, для решения задачи навигации в режиме реального времени на борту беспилотного летательного аппарата.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. GitHub репозиторий библиотеки с открытым исходным кодом OpenCV [Электронный ресурс] / OpenCV Community - 2016. - Режим доступа: https://github.com/opencv/opencv. - Дата доступа: 26.09.2016.

2. 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: 10.11.2016.

3. Lowe, D. Distinctive Image Features from Scale-Invariant Keypoints / D. Lowe // International Journal of Computer Vision — 2004. — no. 60 (2). — pp. 91-101.

4. Bay H., Ess A., Tuytelaars T., Van Gool L. SURF: Speeded up robust features // Computer Vision and Image Understanding – 2008. – no. 110. – pp. 346–359.

5. Calonder M., Lepetit V., Strecha C., Fua P. BRIEF: Binary Robust Independent Elementary Features // Proc. European Conference on Computer Vision – 2010. – pp. 778–792.

6. Kalal Z., Matas J., Mikolajczyk K. Forward-backward error: automatic detection of tracking failures ICPR'10 - 2010. – pp. 2756-2759.

7. 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.

8. Smith R.C., Cheeseman P. On the Representation and Estimation of Spatial Uncertainty // The International Journal of Robotics Research - 1986. - no. 5 (4). - pp. 56–68.

9. Georg Klein, David Murray. Parallel Tracking and Mapping for Small AR Workspaces // 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.

10. J. Engel, T. Schöps, D. Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM // 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.

11. Raúl Mur-Artal, J. M. M. Montiel, Juan D. Tardós ORB-SLAM: A Versatile and Accurate Monocular SLAM System. // IEEE Transactions on Robotics - 2015. - vol. 31, no. 5 - pp. 1147-1163.

12. Christian Forster, Matia Pizzoli, Davide Scaramuzza. SVO: Fast Semi-Direct Monocular Visual Odometry // 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.