

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ
БЕЛАРУСЬ**

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Кафедра многопроцессорных систем и сетей

**РАЗРАБОТКА АЛГОРИТМОВ НАВИГАЦИИ БЕСПИЛОТНЫХ
ЛЕТАТЕЛЬНЫХ АППАРАТОВ**

Курсовая работа

Пажитных Ивана Павловича
студента 3 курса
специальность "информатика"

Научный руководитель:
Кондратьева Ольга Михайловна
старший преподаватель кафедры МСС

Минск, 2017

АННОТАЦІЯ

Пажитных І.П. Розробка алгоритмів навігації беспілотних летательних апаратів (БПЛА). Курсова робота / Мінськ: БГУ, 2017 - 24 с.

В цій роботі розглянуті основні алгоритми комп'ютерного зору, їх застосування для побудови 3D карт місцевості по знімках з БПЛА та можливість реалізації навігації по цій карті. Створено та реалізовано додаток, який дозволяє будувати 3D моделі та проводити пошуки за ними.

АННАТАЦЫЯ

Пажытных І.П. Разработка алгоритмов навигации беспилотных летательных аппаратов (БПЛА). Курсовая првект / Мінск: БДУ, 2016 - 24 с.

У гэтай працы разгледжаны асноўныя алгоритмы камп'ютарнага зроку, іх выкарыстоўванне для будавання 3D мапы мясцовасці па здымках з БПЛА і магчымасць ажыццяўлення навігацыі па гэтай мапе. Спраектавана і разпрацавана праграмнае забеспячэнне, з дапамогай якога магчыма будаваць 3D мадэлі і ажыццяўляць пошук па ім.

ANNOTATION

Pazhitnykh I.P. Development of algorithms for navigation of unmanned aerial vehicles (UAV). Course project / Minsk: BSU, 2016 - 24 p.

In this work, the basic algorithms of computer vision and their application to build 3D maps of the area from UAV images were considered. Designed and implemented an application that allows build 3D models and realize search.

РЕФЕРАТ

Курсовая работа, 24 стр., 11 рисунков, 9 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, приложение.

СОДЕРЖАНИЕ

| | |
|---|-----------|
| ВВЕДЕНИЕ | 5 |
| 1 СРАВНИТЕЛЬНЫЙ АНАЛИЗ АЛГОРИТМОВ | 6 |
| 1.1 Постановка задачи | 6 |
| 1.2 Актуальность и практическая значимость | 6 |
| 1.3 Общие теоретические положения | 7 |
| 1.4 Алгоритм SIFT | 8 |
| 1.4.1 Извлечение ключевых точек | 8 |
| 1.4.2 Извлечение дескрипторов | 10 |
| 1.5 Анализ других алгоритмов | 11 |
| 1.5.1 Дескриптор SURF | 12 |
| 1.5.2 Дескриптор BRIEF | 12 |
| 1.5.3 Дескриптор GLOH | 13 |
| 1.5.4 FAST детектор | 13 |
| 1.5.5 Дескриптор ORB | 14 |
| 1.6 Выводы | 14 |
| 2 ЭКСПЕРИМЕНТАЛЬНЫЕ РЕЗУЛЬТАТЫ | 16 |
| 2.1 Подготовка данных | 16 |
| 2.2 Matching эксперименты | 16 |
| 2.3 Выводы | 18 |
| 3 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ | 19 |
| 3.1 Выбор технологий | 19 |
| 3.2 Разработка алгоритма поиска | 19 |
| 3.3 Приложение для построения реконструкции | 20 |
| 3.4 Выводы | 21 |
| ЗАКЛЮЧЕНИЕ | 23 |
| СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ | 24 |

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 человека-машинного взаимодействия), системы дополненной реальности.

Крупнейшая мировая ИТ 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.

СРАВНИТЕЛЬНЫЙ АНАЛИЗ АЛГОРИТМОВ

1.1 Постановка задачи

Необходимо изучить и проанализировать существующие алгоритмы компьютерного зрения, провести практические эксперименты и впоследствии применить накопленные знания для решения задачи навигации беспилотного летательного аппарата в условиях отсутствия GPS.

Задачу навигации можно разбить на этапы:

1. Построение 3D карты местности:

- сбор и подготовка данных;
- восстановление модели местности;

2. Разработка алгоритма навигации по существующей карте:

- нахождение себя на 3D карте по снимку;
- извлечение gps координат.
- осуществление навигации.

3. Оптимизация алгоритма для возможности построения карты в режиме реального времени на борту БПЛА.

1.2 Актуальность и практическая значимость

Как следует из названия БПЛА не имеют пилота, но это не значит, что они не пилотируемые. Управление беспилотником требует специального обучения, сосредоточенности и является очень утомительным для оператора. Основополагающим необходимым условиям 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 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, навигация по заданной графической точке.

1.3 Общие теоретические положения

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 - эта 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. Этот процесс называется *сопоставление признаков* (**feature matching**). Для сравнения удобно использовать *дескрипторы* (**descriptor** “описатель”). Дескриптор - своеобразный описатель или идентификатор ключевой точки, представляющий точку в удобном для сравнения виде. Как мы увидим далее, именно благодаря дескрипторам получается инвариантность относительно преобразований изображений. В итоге получается следующая схема решения задачи сопоставления изображений:

1. Выделение ключевых точек и дескрипторов;
2. Сравнение дескрипторов и нахождение паросочетаний соответствующий друг другу особых точек;
3. Нахождение геометрического преобразования, которое переводит ключевые точки одного изображения в соответствующие им точки другого

изображения.

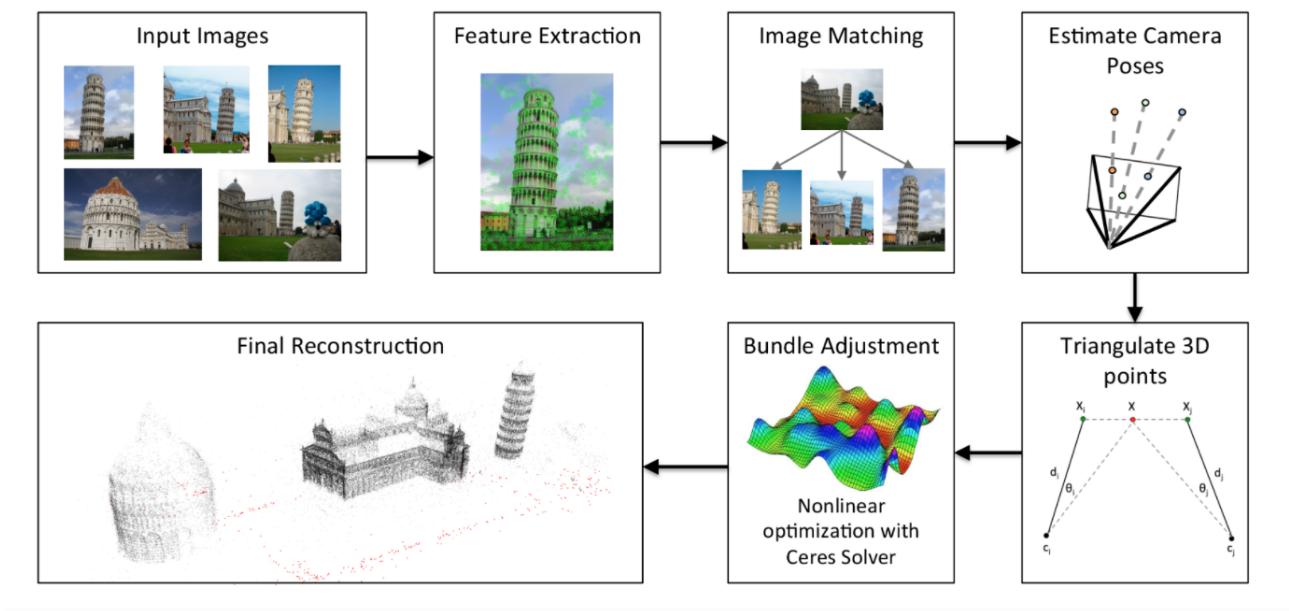


Рисунок 1.1 — **Structure From Motion**

На рисунке 1.1 представлена схема, демонстрирующая процесс извлечения *структурь из движение* (**structure from motion**). Таким образом восстанавливается 3d модель поверхности.

Далее будут подробнее рассмотрены *алгоритмы основанные на особых точках* (**feature-based algorithms**).

1.4 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 [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.

1.4.1 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:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (1.1)$$

В уравнении (1.1): 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 гауссiana с другим радиусом размытия:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma) \quad (1.2)$$

Таким образом, с помощью (1.1), мы получаем набор изображений, являющихся исходным изображением взятым в разных масштабах. Извлечение ключевых точек 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 1.2): 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.

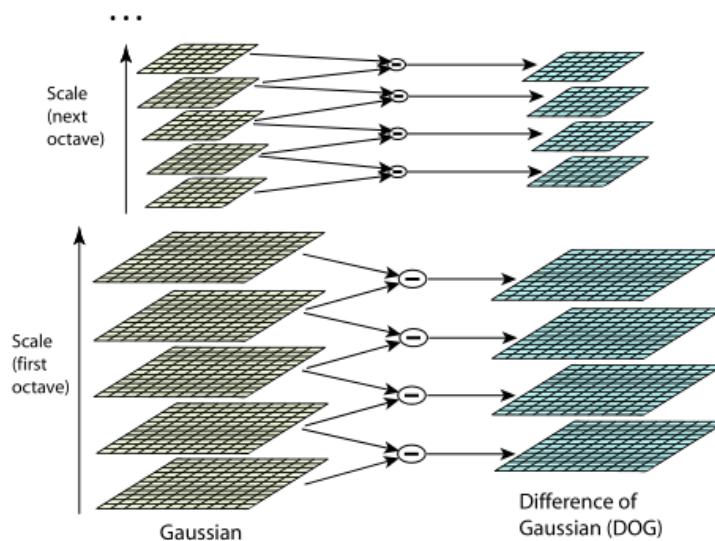


Рисунок 1.2 — Пирамида гауссианов

После построения пирамиды разностей гауссианов по всем точкам в пирамиде ищутся локальные экстремумы. Если точка больше (меньше) всех своих 26 соседей (Рисунок 1.3) в пирамиде разностей, то она считается ключевой.

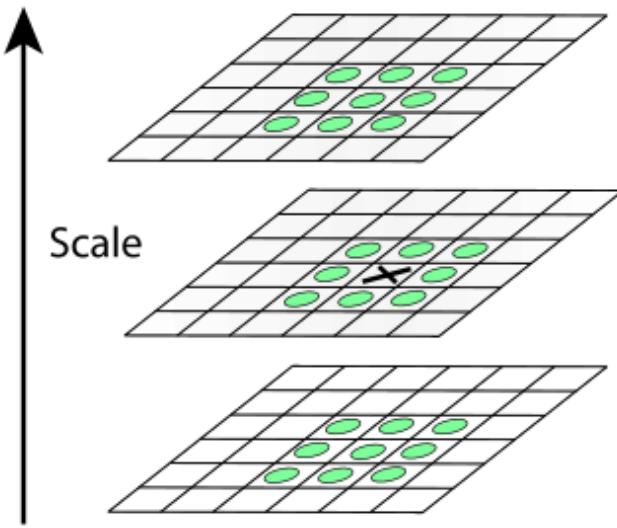


Рисунок 1.3 — Локальный экстремум в пирамиде Гауссианов

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 (1.3) и (1.4) соответственно.

$$m(x, y) = \sqrt{(L(x + 1, y) - L(x - 1, y))^2 + (L(x, y + 1) - L(x, y - 1))^2} \quad (1.3)$$

$$\theta(x, y) = \arctan \left(\frac{L(x, y + 1) - L(x, y - 1)}{L(x + 1, y) - L(x - 1, y)} \right) \quad (1.4)$$

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.

1.4.2 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 котором получен оптимальный размер особой точки. Для

достижения инвариантности относительно поворота изображения перед вычислением всю область ключевой точки поворачивают на угол направления ключевой точки.

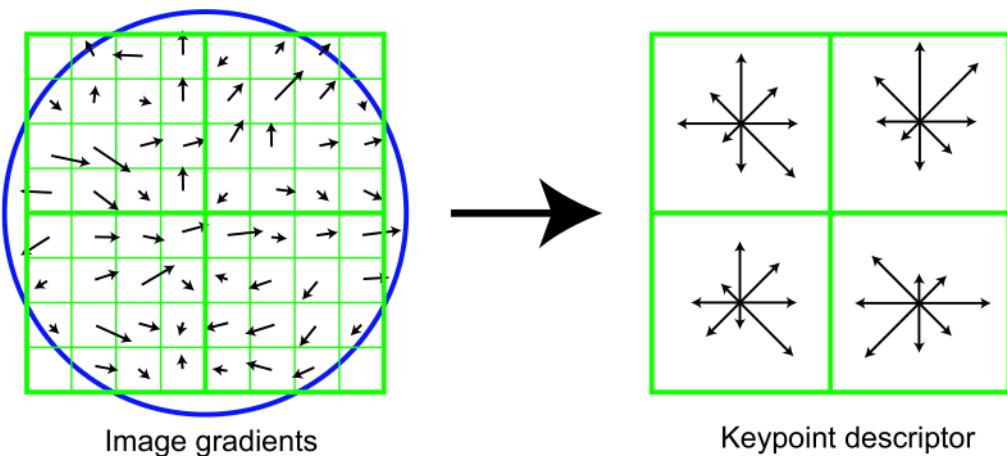


Рисунок 1.4 – Получение дескриптора

На Рисунке 1.4 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 $2 \times 2 \times 8$ (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 $2 \times 2 \times 8 = 32$ компоненты, но на практике больше распространены и активнее используются дескрипторы размерности 128 компонент ($4 \times 4 \times 8$).

1.5 Анализ других алгоритмов

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.

1.5.1 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 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 мультиплексному изменению яркости достигается путем нормировки дескриптора. SURF является эвристическим, но и более быстрым, чем SIFT.

1.5.2 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 p_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:

$$d_{ij} = \sum_{k=1}^p |x_{ik} - x_{jk}| \quad (1.5)$$

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.

1.5.3 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 1.5): 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.

1.5.4 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

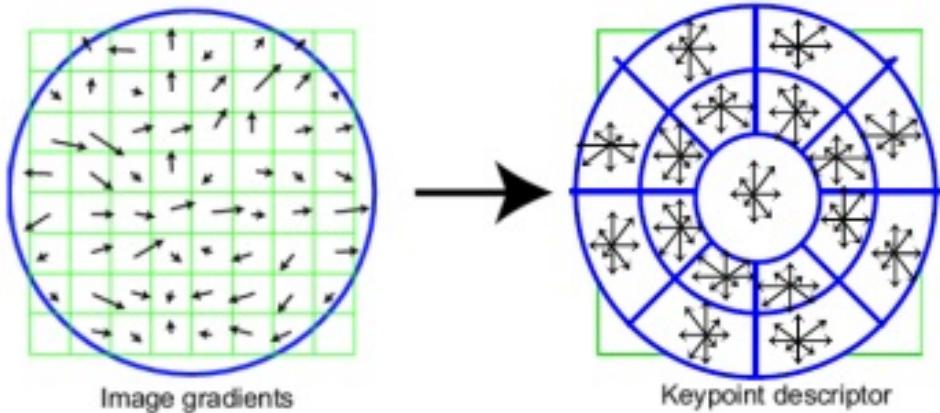


Рисунок 1.5 — Полярная сетка разбиения на бины

п 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 и наименьшее, при котором края не обнаруживаются.

1.5.5 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 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.

1.6 Выводы

В этой главе мной были рассмотрены основные алгоритмы компьютерного зрения, с помощью которых можно находить и сравнивать ключевые точки. Подводя итог анализа: SIFT самый первый, математически точный и медленный дескриптор, на котором основаны большинство современных эвристических алгоритмов feature extraction. SURF - запатентованный в США алгоритм и поэтому является закрытым. Авторы BRIEF приводят результаты экспериментов в которых при одинаковых условиях на некоторых тестовых изображениях точность детектирования с помощью BRIEF почти в

1.5 раза выше, чем с использованием SURF-дескрипторов. Дескриптор ORB быстрый, устойчивый ко многим искажениям и действительно является эффективной альтернативой алгоритму SIFT. Также ORB находится свободном доступе.

В своих дальнейших исследованиях я решил провести эксперименты с использованием SIFT - так как он является стандартом в компьютерном зрении, а ORB использовать как альтернативу и сравнить результаты по времени работы и точности сопоставления. OpenCV предоставляет удобный интерфейс создания дескрипторов и детекторов, что даёт возможность динамически сменять features methods. Практические эксперименты и их результаты будут рассмотрены в следующей главе.

ЭКСПЕРИМЕНТАЛЬНЫЕ РЕЗУЛЬТАТЫ

2.1 Подготовка данных

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

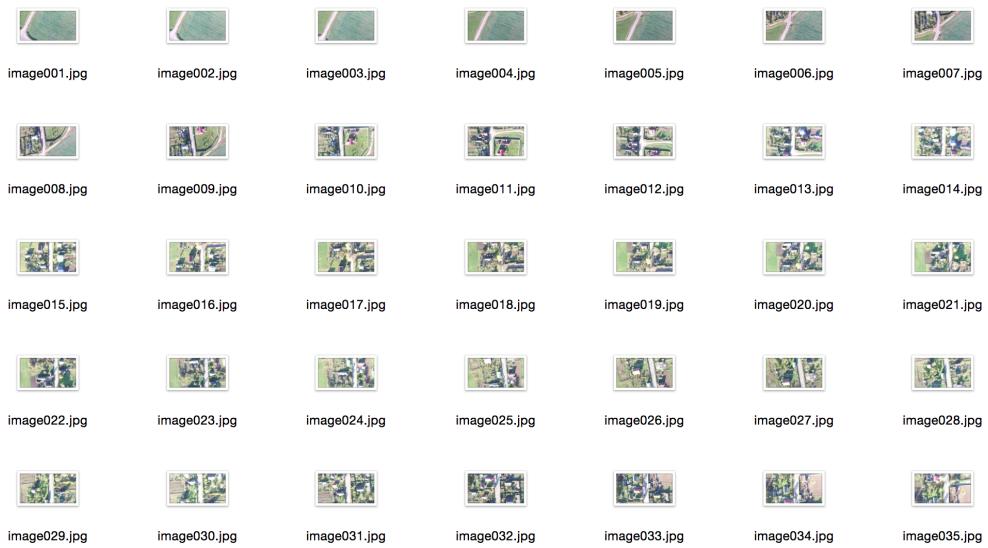


Рисунок 2.1 — Снимки полученные с БПЛА

2.2 Matching эксперименты

Как первое приближение (начальные данные, от которых впоследствии можно будет отталкиваться) был взят следующий простой алгоритм: каждый снимок сделанный на прямом пролете (кривая AB) сопоставляется с каждым снимком из обратного (кривая BA). Для n прямых снимков и m обратных получаем $n * m$ сравнений. В итоге для \forall пары снимков $\{i, j | i = \overline{1, n}, j = \overline{1, m}\}$ получаем *результат сравнения (score)* их ключевых точек, на основе которого можно судить соответствуют ли эти снимки одной и той же точке в пространстве.

Мной была написана реализация этого алгоритма на языке программирования *Python*, используя SIFT [2] и ORB [7] в реализации OpenCV [1]. В

качестве параметров скрипт принимает название алгоритма и максимальное количество ключевых точек, которые будут выделяться на изображении.

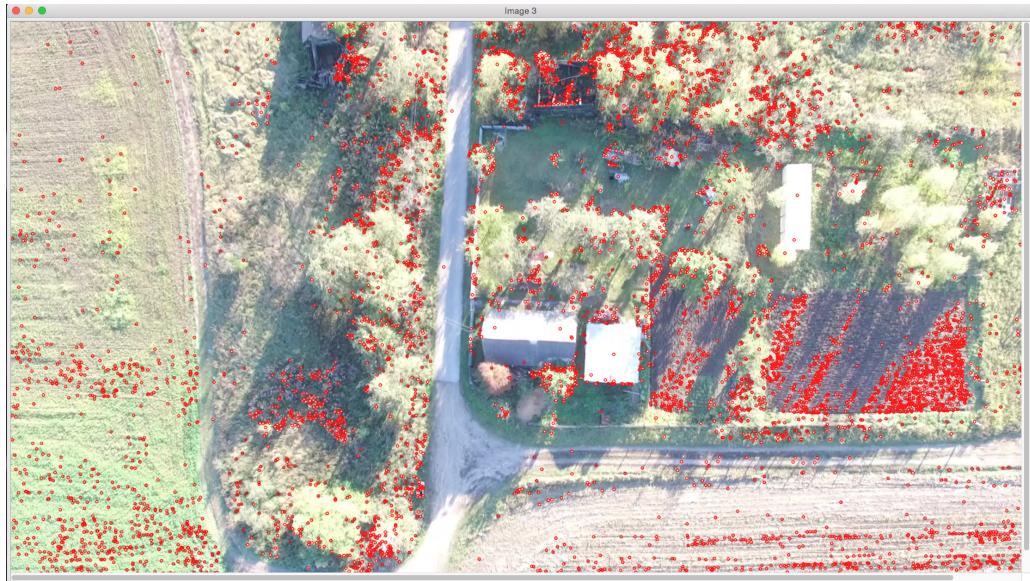


Рисунок 2.2 — Найденные ключевые точки

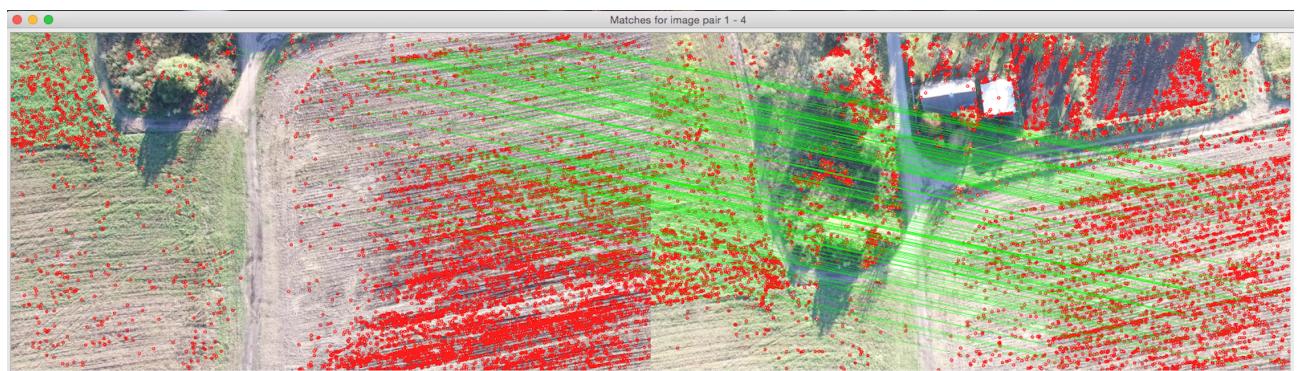


Рисунок 2.3 — Совпадения ключевых точек

Варьируя количество выделяемых ключевых точек и используя разные алгоритмы, были поставлены 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” этот пиксель 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 2.4 представлены полученные результаты на наборах из 40x32 изображений (1280 сравнений). Легенда изображений: *алгоритм* - количество точек - время работы.

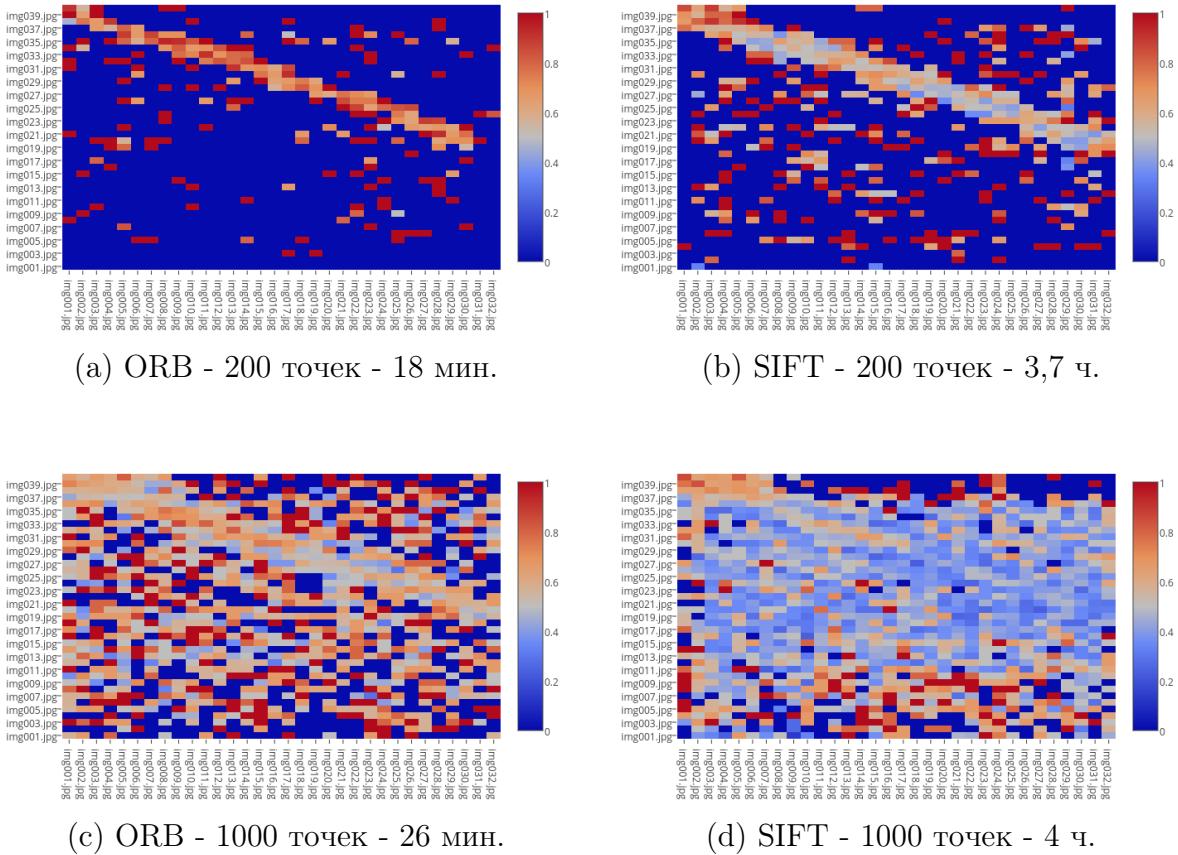


Рисунок 2.4 — Сравнительные тепловые диаграммы

2.3 Выводы

Проанализировав диаграммы на рисунке 2.4, можно сделать вывод, что при малом количестве выделяемых ключевых точек прослеживается траектория полёта. Также заметно, что *SIFT* даёт много ложных сопоставлений при очень большом времени работы - 3,5 часа против 20 минут у *ORB*. Однако 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 при большом числе извлекаемых точек, навигация с использованием этого алгоритма, конечно же, не представляется возможной. Однако мы получили первое приближение, решение “в лоб”, от которого можно отталкиваться в дальнейшем и сравнивать с ним результаты будущей работы.

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

3.1 Выбор технологий

К разрабатываемому приложению были поставлены 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 [8]. 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++, очень хорошая и подробная документация.

Для написания графического пользовательского интерфейса отлично подходил *Qt* [9]. *Qt* - это кроссплатформенный инструментарий разработки приложений на языке программирования C++. *Qt* 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. Также предоставлены обширные инструменты по быстрому и удобному созданию интерфейсов.

3.2 Разработка алгоритма поиска

Итак, после выполнения всех этапов Structure From Motion мы имеем 3d модель - реконструкцию поверхности. Модель представляет из себя набор точек пространства, также мы можем привязать к ним gps-данные. Цель алгоритма - найти на построенной 3d карте расположение нового снимка. Предполагается, что снимок взят не, из исходного датасета, но на нём присутствует та же область пространства, иначе ничего не будет найдено. 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.

3.3 Приложение для построения реконструкции

На рисунке 3.1 представлен интерфейс разработанного приложения. Модель - швейцарский карьер построенный на датасете, взятом из открытых источников.

В приложении реализован следующий функционал:

- создание / открытие 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 содержащий всю информацию и с ним будет ассоциирован проект. При визуализации модели красным 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 (см. рисунок 3.1).

При построении модели можно настроить такие параметры как: количество потоков, в которых будет выполняться каждая часть процесса Structure

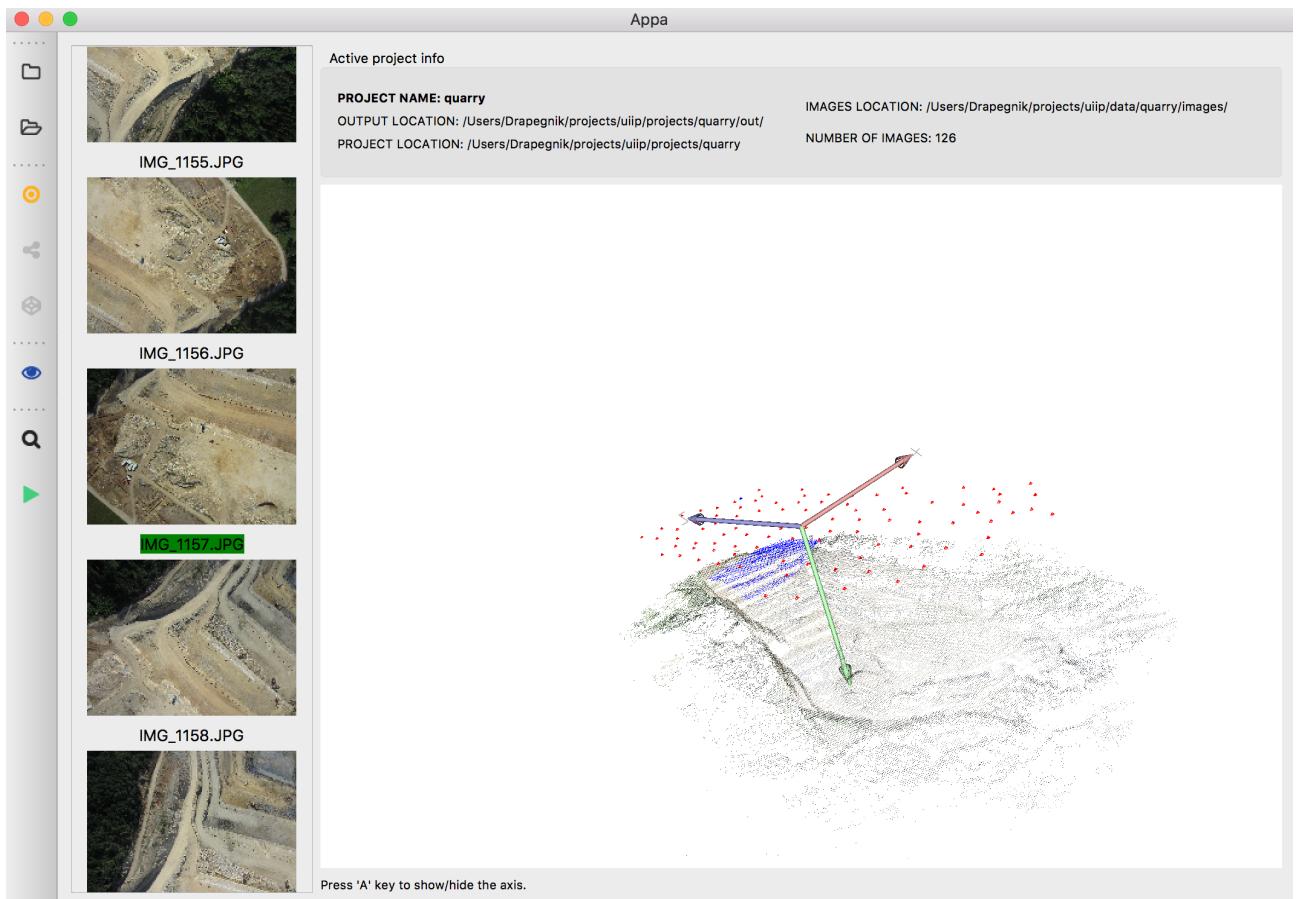


Рисунок 3.1 — Appa - приложение для построения и визуализации 3d моделей, осуществления поиска по ним

From Motion, тип дескриптора и детектора (в данный момент поддерживаются рассмотренный ранее SIFT [2], а также AKAZE), стратегия сопоставления снимков (Brute Force или Cascade Hashing). Остальные настройки касаются внутренних и внешних параметров камеры. (см. рисунок 3.2)

После выполнения поиска ключевые точки модели сопоставленные с искомым снимком подсвечиваются красным. Рассматривая производительность: поиск на датасете из 127 снимков, при извлечении порядка 5000 ключевых точек на каждом изображении осуществляется, в среднем, за 40 – 50 секунд.

3.4 Выводы

В этой главе была представлена проделанная практическая работа. Протестированы новые технологии и решения. Получен результат работы - рабочее приложение, которое можно дорабатывать и развивать. В будущем планируется доработка приложения до стабильной версии и распространение его в свободном доступе.

Анализируя алгоритм и результаты поиска: итоговое время в разы 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.

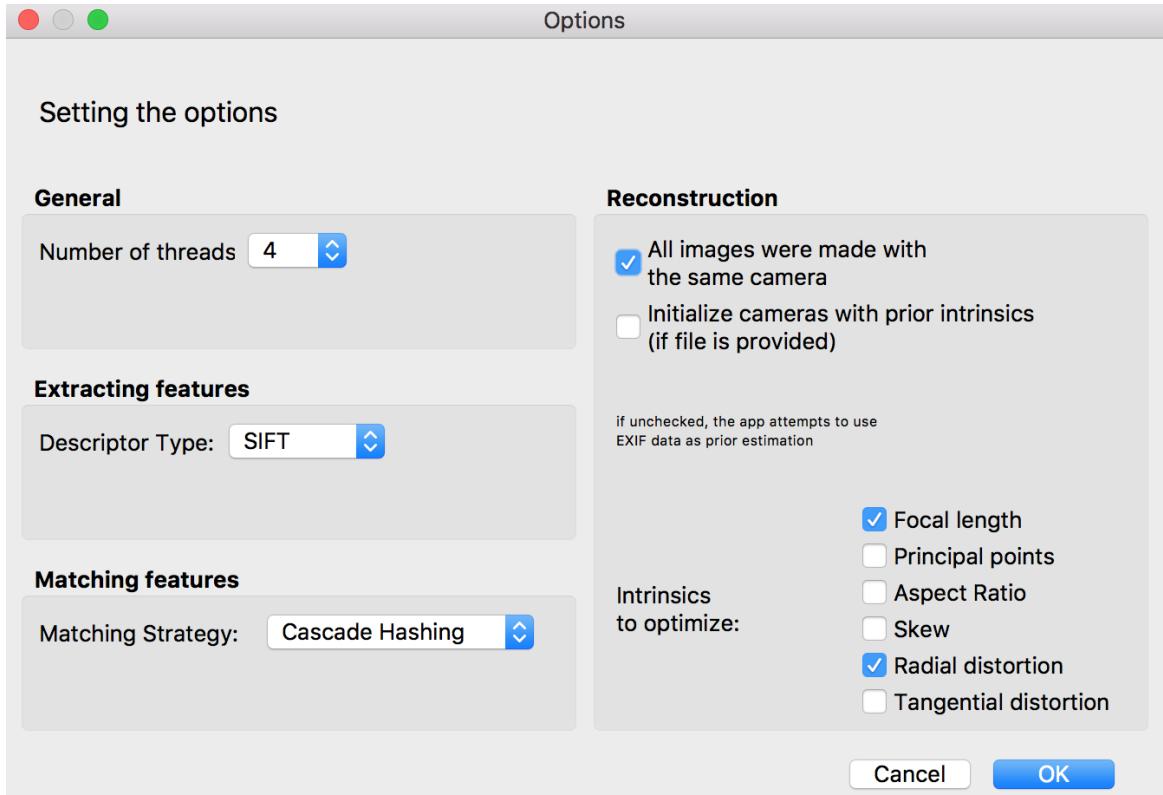


Рисунок 3.2 — Различные параметры построения модели

ЗАКЛЮЧЕНИЕ

В процессе выполнения данной работы были получены следующие результаты:

- выявлены требования и поставлена задача;
- изучены и проанализированы основные алгоритмы компьютерного зрения, основанные на особых точках;
- разработан простейший алгоритм решения задачи поиска;
- подготовлены тестовые данные, проведены эксперименты;
- спроектировано и реализовано приложение, позволяющее строить 3D модели и осуществлять поиск по ним.

Полученные теоретические знания и практические навыки, а также результаты экспериментов являются хорошей основой для будущих исследований.

Выбранная область и проблема позволяют не останавливаться на достигнутом и, в дальнейшем, продолжить работу над задачей.

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. Официальная документация библиотеки с открытым исходным кодом TheiaSfm [Электронный ресурс] / Chris Sweeney - 2016. - Режим доступа: <http://www.theia-sfm.org/>. - Дата доступа: 17.04.2017.
9. Официальный сайт C++ фрэймворка QT [Электронный ресурс] / The Qt Company - 2017. - Режим доступа: <https://www.qt.io/>. - Дата доступа: 20.04.2017.