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 3 курса
специальность "информатика"

Научный руководитель:
Кондратьева Ольга Михайловна
старший преподаватель кафедры МСС

Минск, 2016

РЕФЕРАТ

Курсовой проект, 19 стр., 8 рисунков, 7 источников.

РАЗРАБОТКА АЛГОРИТМОВ НАВИГАЦИИ БЕСПИЛОТНЫХ ЛЕТАТЕЛЬНЫХ АППАРАТОВ

Ключевые слова: навигация, БПЛА, feature extraction, feature matching.

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: теоретическая база, сравнительные эксперименты, алгоритм.

При написании работы использовались стандарты рекомендуемые ВАК АН РБ.

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, определения того, как с помощью геометрический преобразований и/или масштабирования 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 разрабатывает self-driving cars (машины с автопилотом) и предполагается, что в будущем человеку вообще не придётся управлять автомобилем. Это должно уменьшить число происшествий исключая “человеческий фактор” и, соответственно, сделать передвижение с помощью автомобиля безопаснее. Самый популярный сервис такси - Uber уже использует машины с автопилотом, что в будущем позволит снизить стоимость услуг сокращением траты средств на человеческие ресурсы (Компания уже уменьшила траты, используя мобильное приложение вместо диспетчеров). Американская компания Amazon 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.

Таким образом компьютерное зрение, наряду с машинным обучением, является сейчас наиболее новой и активно развивающейся областью информатики, используемой всеми лидерами отрасли. Основное применение компьютерного зрения - уменьшение человеческой работы, 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;
    - извлечение gps координат.
2. Разработка алгоритма навигации по существующей карте:
    - нахождение себя по на карте по снимку;
    - определение маршрута;
    - осуществление навигации.
3. Оптимизация алгоритма для возможности построения карты в режиме реального времени на борту БПЛА.

В рамках этого курсового проекта будет накоплена теоретическая и статистическая база для дальнейших исследований.

1.2 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 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 является беспомощным и теряет управление.

В связи с этим возникает задача нахождения и использования альтернативных источников навигации. Так как почти каждый современный беспилотник оснащён камерой возможно использование алгоритмов компьютерного зрения.

С помощью разработанного алгоритма и программного обеспечения возможна навигация дрона используя только камеру как в военных, так и в личных целях. Например: патрулирование заданной территории и выявление появления новых объектов, возвращение домой в случае потери gps сигнала, слежение за данным объектом, навигация по заданной графической точке.

1.3 Общие теоретические положения

Человек может сравнить изображения и выделять на них объекты визуально, на интуитивном уровне. Однако, для машины изображение — всего лишь ни о чем не говорящий набор данных. Одной из больших проблем в сопоставлении изображений является очень большая размерность пространства, по которому "размазана" информация. Если взять картинку размером хотя бы 100∗100, то уже получим размерность равную 10^4. Как же компьютер обретает зрение?

Основная идея состоит в том, чтобы получить какую-то характеристику, которая будет хорошо описывать изображение, легко вычисляться и к которой можно применить логическую операцию сравнения. Эта "характеристика" должна быть устойчива к различным преобразованиям (сдвиг, поворот и масштабирование изображений, 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 **feature extraction**. Ключевая точка - эта такая особая точка, которая отличается от соседних точек и будет не похожа на остальные, соответственно является, в какой-то степени, уникальным свойством этого изображения. Таким образом машина может представить изображение как модель состоящие из ключевых точек. Примером особых точек, если говорить об изображении лица человека, могут служить глаза, уголки губ, кончик носа.

После выделения особых точек компьютеру нужно уметь их сравнивать. Этот процесс называется **feature matching**. Для сравнения удобно использовать дескрипторы (**descriptor** - "описатель"). Дескриптор - своеобразный описатель или идентификатор ключевой точки, выделяющий её из остальной массы особых точек. Как мы увидим далее именно благодаря дескрипторам получается инвариантность относительно преобразований изображений. В итоге получается следующая схема решения задачи сопоставления изображений:

1. На изображениях выделяются ключевые точки и их дескрипторы;
2. По совпадению дескрипторов выделяются соответствующие друг другу ключевые точки;

3. На основе набора совпавших ключевых точек строится модель преобразования изображений, с помощью которого из одного изображения можно получить другое;

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

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 David G. Lowe в 1999 [3].

На первом этапе часто производится предварительная обработка изображения в целях улучшения качества изображения для последующего его анализа. Например, на фотографиях с камер часто появляются шумы, чтобы их устранить часто 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:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{1.1}$$

(Здесь 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:

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

Таким образом мы получаем изображения на различных масштабах с помощью (1) и получаем масштабируемое пространство - набор всевозможных, сглаженных некоторым фильтром, версий исходного изображения. Доказано, что гауссово масштабируемое пространство является линейным, инвариантным относительно сдвигов, вращений, масштаба, не смещающим локальные экстремумы, и обладает свойством полугрупп.

Инвариантность относительно масштаба достигается за счет нахождения ключевых точек для исходного изображения, взятого в разных масштабах.

Для этого строится пирамида гауссианов (Рисунок 1.1): все масштабируемое пространство 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.

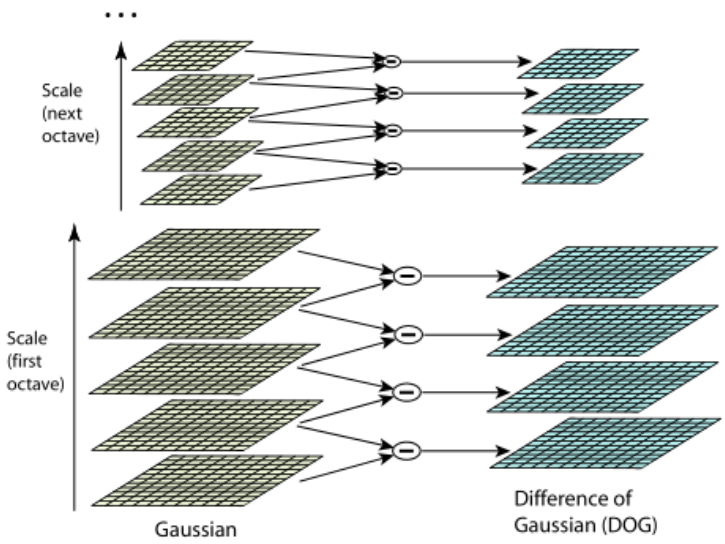


Рисунок 1.1 — Пирамида Гаусианнов

После построения пирамиды разностей гауссианов по всем точкам в пирамиде ищутся локальные экстремумы. Если точка больше (меньше) всех своих 26 соседей (Рисунок 1.2) в пирамиде разностей, то она считается ключевой.

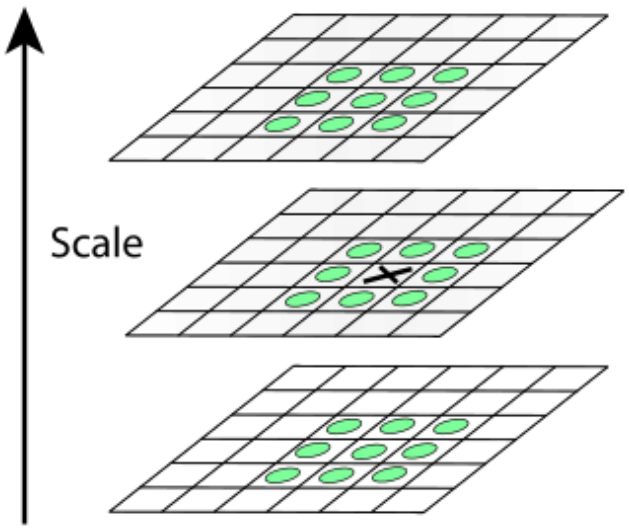


Рисунок 1.2 — Локальный экстремум в пирамиде Гауссианов

Таким образом мы для исходных изображений разных размеров мы получим ключевые точки (и небольшую область возле них) одного и того же размера - это и даёт инвариантность относительно масштабирования.

Направление ключевой точки вычисляется исходя из направлений градиентов точек, соседних с особой. Все вычисления градиентов производятся на изображении в пирамиде гауссианов, с масштабом наиболее близким к масштабу ключевой точки. Величина и направление градиента в точке (x, y) вычисляются по формулам (3) и (4) соответственно.

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

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

Направление считается в σ-окрестности ключевой точки. Каждая точка окрестности (x, y) вносит вклад и итоговое значение считается за направление ключевой точки.

1.4.2 Извлечение дескрипторов

Как уже говорилось ранее - дескриптор должен очень хорошо и уникально описывать ключевую точку. В общем случае это может быть любой объект, который будет выполнять данные функции и является инвариантным относительно преобразований исходного изображения.

В алгоритме SIFT дескриптор представляет из себя вектор, содержащий информацию об окрестности ключевой точки. Дескриптор вычисляется на том же гауссиане, на котом получен оптимальный размер особой точки. Перед вычислением, для достижения инвариантности относительно поворота изображения всю область ключевой точки поворачивают на угол направления ключевой точки.

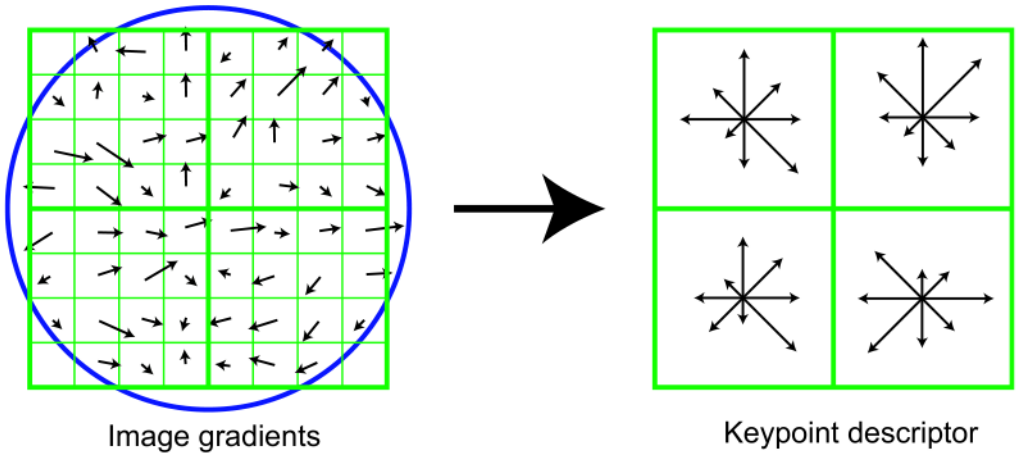


Рисунок 1.3 — Получение дескриптора

На Рисуноке 1.3 показана окрестность ключевой точки (слева) и полученный на её основе дескриптор (справа). Маленькая стрелочка, в центре

каждого пикселя в σ-окрестности обозначает градиент этого пикселя. Круг обозначает окно свертки с гауссовым ядром. Для этого ядра определяется σ, равное половине ширины окна дескриптора. В дальнейшем значение каждой точки окна дескриптора будет домножаться на значение гауссова ядра в этой точке, как на весовой коэффициент.

Как видно справа дескриптор имеет размерность 2x2x8 (количество регионов по горизонтали, количество регионов по вертикали, количество компонент гистограммы этих регионов). Гистограмма для каждого региона строится в соответствии со значениями градиентов пикселей, входящих в σ-окрестность (8 штук):

1. Каждая гистограмма так же покрывает участок в 360 градусов и делит его на 8 частей;

2. В качестве весового коэффициента берется значение гауссова ядра, общего для всего дескриптора;

3. В качестве ещё одних весовых коэффициентов берутся коэффициенты трилинейной интерполяции;

Каждому градиенту в окне дескриптора можно приписать три вещественные координаты (x, y, n), где x — расстояние до градиента по горизонтали, y — расстояние по вертикали, n — расстояние до направления градиента в гистограмме (имеется ввиду соответствующая гистограмма дескриптора, в которую вносит вклад этот градиент). Коэффициент трилинейной интерполяции определяется для каждой координаты (x, y, n) градиента как $1-d$, где d равно расстоянию от координаты градиента до середины того единичного промежутка в который эта координата попала. Каждое вхождение градиента в гистограмму умножается на все три весовых коэффициента трилинейной интерполяции.

Дескриптор ключевой точки состоит из всех полученных гистограмм. Как уже было сказано размерность дескриптора на рисунке 32 компоненты (2x2x8), но на практике используются дескрипторы размерности 128 компонент (4x4x8). Полученный дескриптор нормализуется, после чего все его компоненты, значение которых больше 0.2, урезаются до значения 0.2 и затем дескриптор нормализуется ещё раз. В таком виде дескрипторы готовы к использованию.

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 интересной точки несколько иная, поскольку в ней используются не гистограммы взвешенных градиентов, а отклики исходного изображения на вейвлеты Хаара. 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 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.

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. 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 — число позиций, в которых соответствующие символы двух слов одинаковой длины различны, вычисляется по формуле:

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

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 1.4): 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 (круге построенном с помощью алгоритма

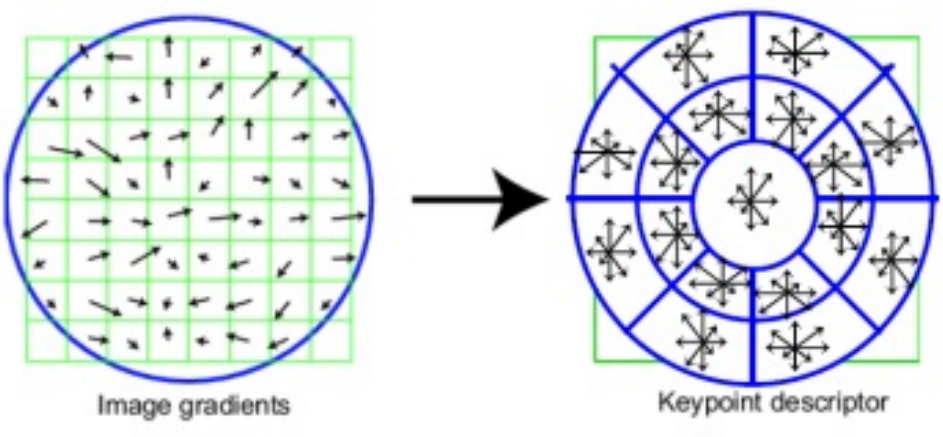


Рисунок 1.4 — Полярная сетка разбиения на бины

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, только значение $r = 3$ используется (соответствующий круг 16 пикселей окружности), и тесты показывают, что оптимальное значение $n = 9$. Это значение n 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 первоначальные алгоритмы. Был предложен Ethan Rublee в 2010 году. Также как и BRIEF, ORB имеет размер 32 байта и для сравнения использует расстояния Хэминга. После детектирования точек с помощью FAST-а ORB выделяет N топ точек используя меру Харисса. Как следует из названия, далее ORB ориентирует найденные ключевые точки. 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 доставать и сравнивать ключевые точки. Подводя итог анализа: SIFT самый первый, математически точный и медленный дескриптор, на котором основаны большинство современных эвристических алгоритмов feature extraction.

В своих дальнейших исследованиях я буду использовать SIFT - так как он является стандартом в компьютерном зрении. SURF - запатентован в США

и является закрытым, также 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.

Очень привлекательным выглядит ORB - он быстрый, устойчивый и действительно является эффективность альтернативой SIFT'у. Также он совершенно бесплатный и свободно распространяемый. Поэтому я решил остановить выбор на нём, а SIFT использовать как точное значения и для анализа результатов разными средствами. 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 **data set'ы** (наборы данных). 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 2секунды. 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.

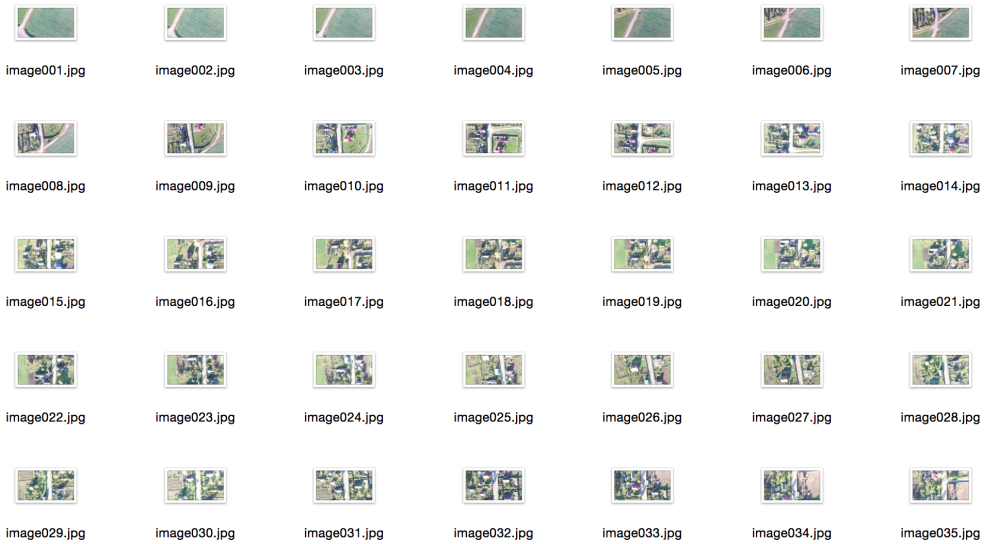


Рисунок 2.1 — Снимки полученные с БПЛА

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 нейкий **score** (результат) совпадения 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 выделятся на изображении.

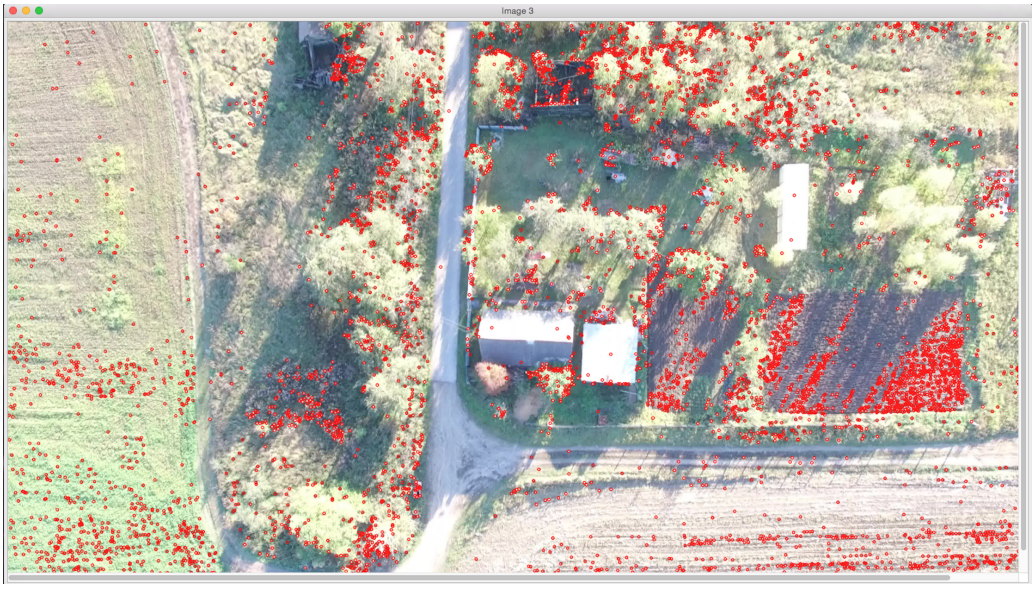


Рисунок 2.2 — Найденные ключевые точки

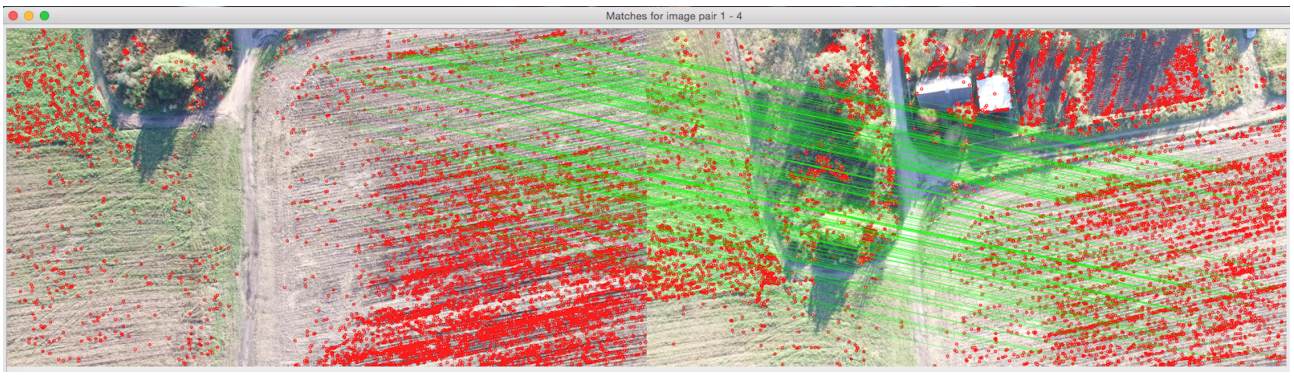
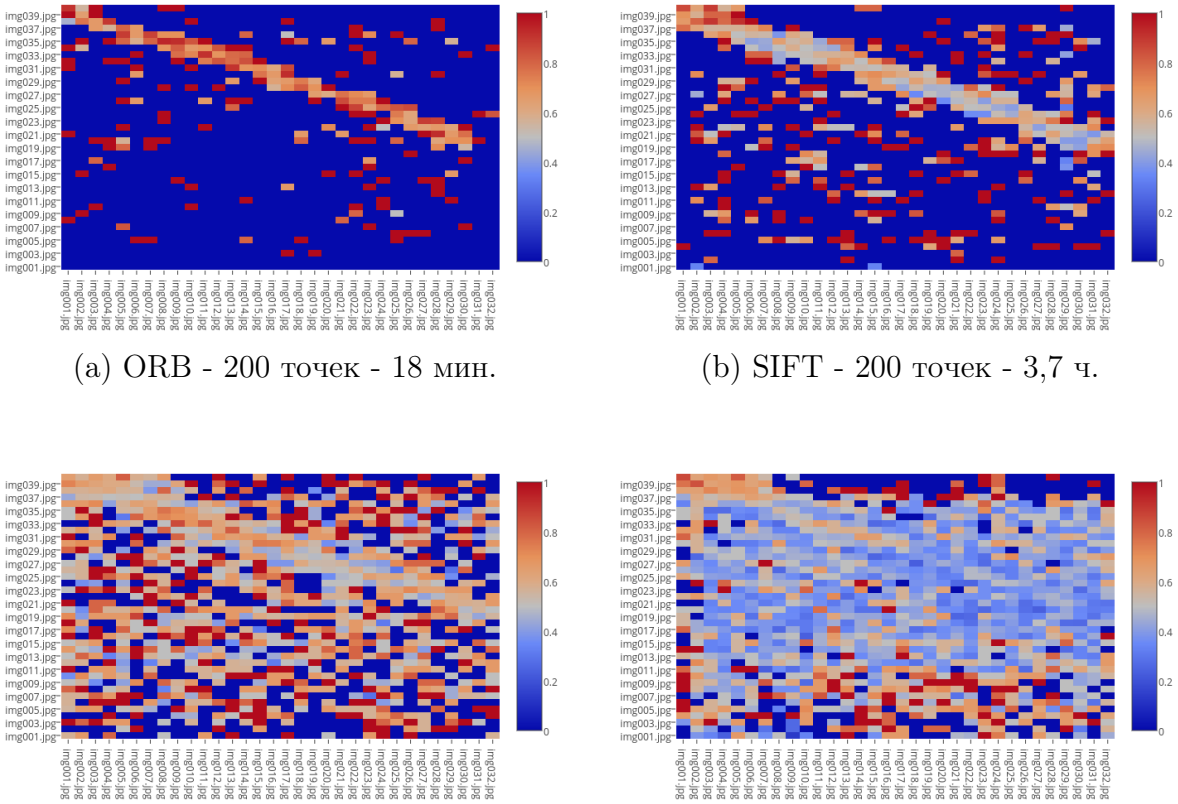


Рисунок 2.3 — Совпадения ключевых точек

Варьируя количество выделяемых ключевых точек на разных алгоритмах были поставлены эксперименты. Для визуализации качества сопоставления были построены **heatmap'ы** (тепловые карты) - диаграммы показывающие насколько хорошо совпадает изображение x_i с x_j (расположены соответственно на осях абсцисс и ординат). Сопоставление будем считать хорошим, если на диаграмме будет точно прослеживаться траектория: так как при пролётах туда-обратно мы должны получать, что x_0 и y_n, x_1 и y_{n-1} ... и т.д. изображают одну и ту же точку пространства. Также анализировалось время работы программы. Полученные результаты на наборах 40x32 изображений (1280 сравнений) представлены в виде *алгоритм - количество точек - время работы* на рисунке 2.4



(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 2.4 — Сравнительные тепловые диаграммы

2.3 Выводы

Анализируя диаграммы (Рисунок 2.4) видно, что при малом количестве выделяемых ключевых точек прослеживается траектория полёта. Причём, 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 на большом числе точек навигация с использованием этого алгоритма, конечно же, не представляется возможной. Однако мы получили первое приближение, решения "в лоб" от которого можно отталкиваться и сравнивать с ним результаты будущих исследований.

ЗАКЛЮЧЕНИЕ

В процессе выполнения данного проекта были получены следующие результаты:

- выявлены требования и поставлена задача;
- изучены и проанализированы основные алгоритмы компьютерного зрения, основанные на особых точках;
- разработан и реализован простейший алгоритм решения поставленной задачи;
- подготовлены тестовые данные, проведены эксперименты и получены результаты.

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

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.