МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ М. В. ЛОМОНОСОВА

Факультет космических исследований

Магистерская программа

**«Робототехника и Интеллектуальные технологии»**

Выпускная квалификационная работа

**«Применение точечных детекторов в алгоритмах обработки изображений ландшафтных сцен»**

Выполнил:

Шелепанов Даниил Андреевич, М-208 РИ

Научный руководитель:

д.ф.-м.н. Богуславский Андрей Александрович

Москва, 2021

Оглавление

[Введение 3](#_Toc73057695)

[Детекторы 4](#_Toc73057696)

[Детектор углов Харриса 4](#_Toc73057697)

[Детектор Shi-Tomasi 5](#_Toc73057698)

[Детектор FAST 6](#_Toc73057699)

[Детектор Star 7](#_Toc73057700)

[Дескрипторы 8](#_Toc73057701)

[BRIEF 8](#_Toc73057702)

[FREAK 9](#_Toc73057703)

[Алгоритмы, содержащие в себе детектор и дескриптор 11](#_Toc73057704)

[SIFT 11](#_Toc73057705)

[SURF 14](#_Toc73057706)

[ORB 15](#_Toc73057707)

[BRISK 17](#_Toc73057708)

[Методы прослеживания 20](#_Toc73057709)

[Обзор ускорителей и стандартов 21](#_Toc73057710)

[Тестовые видеопоследовательности 22](#_Toc73057711)

[Методика выполнения работы 24](#_Toc73057712)

[Результаты выполнения программы 25](#_Toc73057713)

[Интерпретация результатов и вывод 37](#_Toc73057714)

[Заключение 39](#_Toc73057715)

[Приложение 42](#_Toc73057716)

# Введение

В последние 20 лет применение систем технического зрения (СТЗ) в анализе изображений ландшафтов для решения разнообразных задач достигло уровня успешного применения. Среди различных работ в данной области можно выделить работы, связанные с обработкой изображений ландшафтов, полученных как в результате съёмки с воздуха (картографирование местности с помощью летательных аппаратов с бортовыми камерами [2], Определение компонент движения посадочного модуля марсианских автоматических станций Spirit и Opportunity [11, 12]), так и с бортовой камеры наземного РТК в естественной среде (Nistér, Naroditsky, Bergen [4]).

В работах по сопоставлению изображений друг с другом и с картами существенным допущением является то, что на изображениях ландшафтов как правило отсутствуют объекты с заранее известной моделью, то есть в основу работы таких систем необходимо поставить точечные признаки. В качестве первых этапов обработки в рассматриваемых системах выполняется алгоритм, состоящий из двух логических частей:

1. детектор точечных признаков

2. метод прослеживания

Важной характеристикой такого алгоритма является количество кадров, на которых можно проследить один признак. В частности, для определения компонент движения посадочных модулей станций Spirit и Opportunity [11, 12] было проанализировано три последовательных кадра. Таким образом, цель данной работы — для каждого из двух типов ландшафтных изображений (наземная и орбитальная съёмки) выбрать алгоритмы, которые удовлетворяли бы требованиям скорости и надёжности: обрабатывать не менее 25 кадров в секунду и стабильно отслеживать признаки не менее чем на трёх последовательных кадрах.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. Выбрать детекторы точечных признаков, соответствующие поставленной цели;
2. Выбрать методы прослеживания;
3. Выбрать вычислители, для которых есть реализации различных детекторов точечных признаков;
4. Оценить работу алгоритмов на видеопоследовательностях, представленных в исходных данных, и вычислителях по количеству обрабатываемых в секунду кадров и надёжности отслеживания признаков;
5. На основании результатов оценки выбрать алгоритмы, удовлетворяющие цели работы.

# Детекторы

Первой частью искомого алгоритма является детектор точечных признаков. Точечный признак — точка, отличающаяся от соседних точек в некоторой окрестности по определенному критерию. Детектор точечных признаков должен обеспечивать инвариантность нахождения одних и тех же особых точек относительно преобразований изображений. Различные детекторы используют различные техники поиска и имеют свои особенности, достоинства и недостатки.

## Детектор углов Харриса

Детектор Харриса является детектором не признаков, а особых точек, то есть не включает в себя метод построения дескриптора. Предназначен для обнаружения углов. Для этого в нем используются собственные значения и матрицы:

где — первая производная локальной свертки окрестности изображения в точке с ядром гауссовой функции со стандартным отклонением . Cобственные значения и матрицы представляют изменения интенсивностей изображения в ортогональных направлениях. Рассмотрим функцию отклика (cornerness measure):

при небольшом значении параметра , например, . В силу общих свойств собственных значений имеем:

.

Детектор Харриса широко используется в задачах получения визуальной одометрии, например, в [5, 7].

## Детектор Shi-Tomasi

Детектор Shi-Tomasi является модификацией детектора Харриса. В детекторе Shi-Tomasi функция отклика определяется как:

Если её значение выше некоего заранее заданного порога, точка считается углом (особой точкой). Ниже на рис. 1 можно видеть, что только когда минимальное из собственных значений больше порога точка p рассматривается как угол (зелёная область).

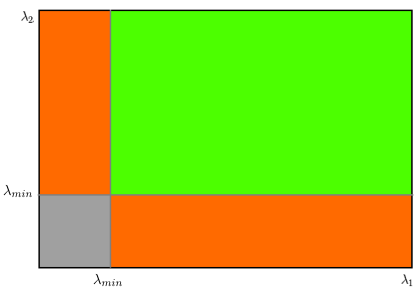


Рисунок 1. Диапазоны значений функции отклика детектора Shi-Tomasi

Детектор Shi-Tomasi показывает лучшие, чем детектор Харриса, результаты при использовании алгоритма оптического потока.

## Детектор FAST

Детектор особых точек FAST (Features from Accelerated Segments Test) сравнивает значение яркости точки p со значением яркости окружающих его 16 пикселей: если большинство пикселей, принадлежащих этой окружности, не похожи на точку p (значительно ярче или темнее), то она выбирается как особая точка:

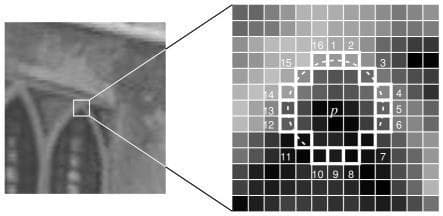


Рисунок 2. Детектор особых точек FAST

По умолчанию FAST не является инвариантным относительно масштаба и поворота, но известны его модификации, решающие эти проблемы.

## Детектор Star

Детектор особых точек Star (название отражает форму шаблона дескриптора) был разработан специально для задач визуальной одометрии и SLAM (simultaneous localization and mapping – одновременная локализация и навигация – класс задач, связанных с автономным перемещением РТК). Детекторы Харриса, Shi-Tomasi и FAST, с одной стороны, хорошо подходят для данных задач, так как точки, которые ими находятся, хорошо локализуемы. В то же время они не являются инвариантными относительно масштаба, так как детекторы Харриса, Shi-Tomasi и FAST не производят поиск в пространстве масштабов. Детектор SURF, наоборот, инвариантный относительно масштаба, но признаки, найденные им, плохо локализуются с увеличением масштаба. Детектор Star создан таким образом, чтобы предоставить как высокую локализуемость признаков, так и их инвариантность относительно масштаба. Для этого в Star происходит расчёт функции отклика DoG (difference of Gaussians, разность Гауссиан), поиск локальных экстремумов и подавление немаксимумов с использованием функции отклика детектора Харриса, учитывающей масштаб.

Так как Star разрабатывался специально для работы в режиме реального времени, работа с пространством масштабов должна происходить очень быстро. Это достигается путём работы с интегральными изображениями, как в SURF:

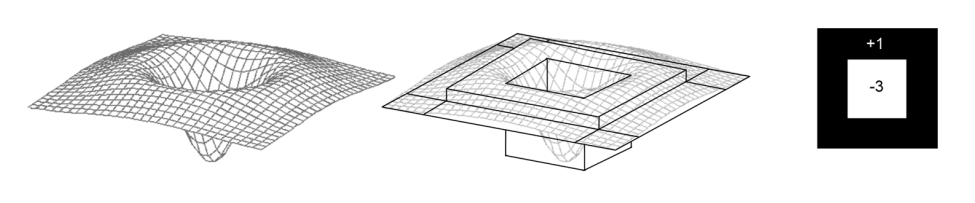


Рисунок 3. Аппроксимация DoG в Star: слева — функция DoG от двух аргументов; по центру – приближение DoG целыми значениями; справа – ядро фильтра, реализованное в OpenCV.

# Дескрипторы

Дескриптор – вектор, описывающий окрестность особой точки и позволяющий однозначно выделить её среди других точек, используется для сопоставления точечных признаков, принадлежащих разным изображениям. Как и детекторы точек, генераторы дескрипторов реализуют различные подходы и имеют свои сильные и слабые стороны.

## BRIEF

Генератор дескрипторов BRIEF (Binary Robust Independent Elementary Features) составляет дескриптор из серии тестов: в окрестности особой точки случайным образом выбирается несколько (обычно 128, 256 или 512) пар пикселей, между каждой из которых производится сравнение интенсивности, результат сравнения i-ой пары (1 или 0) записывается в дескриптор на место i-ого бита. Таким образом, дескриптор может занимать 16, 32 или 64 байт памяти. Для уменьшения шумов и увеличения надёжности дескриптора к изображению предварительно применяется фильтр Гаусса.

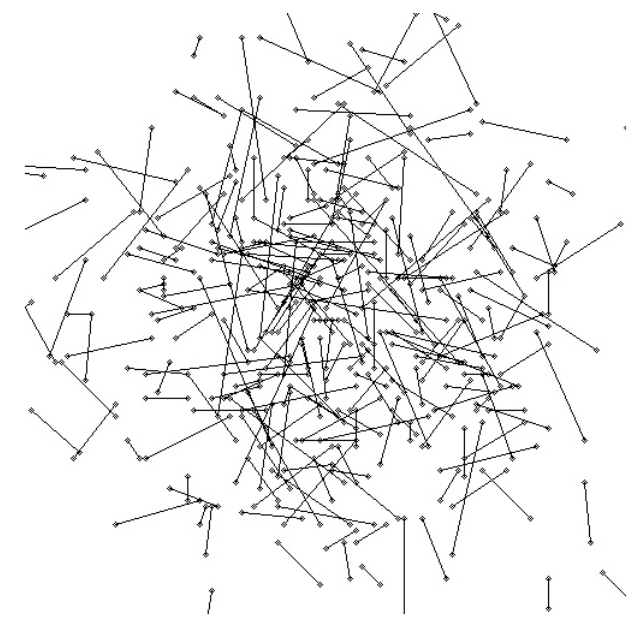


Рисунок 4. Пример распределения пар пикселей в дескрипторе BRIEF.

## FREAK

Генератор дескрипторов FREAK (Fast Retianl Keypoint) основан на том же принципе, что и BRIEF – серия тестов и запись результатов в вектор, для уменьшения чувствительности к шуму используется фильтр Гаусса. Главное различие состоит в том, что в алгоритме FREAK сравниваются не интенсивности пикселей, а взвешенные по Гауссу суммы неких подобластей. На рис. 5 изображена схема сравниваемых подобластей дескриптора.

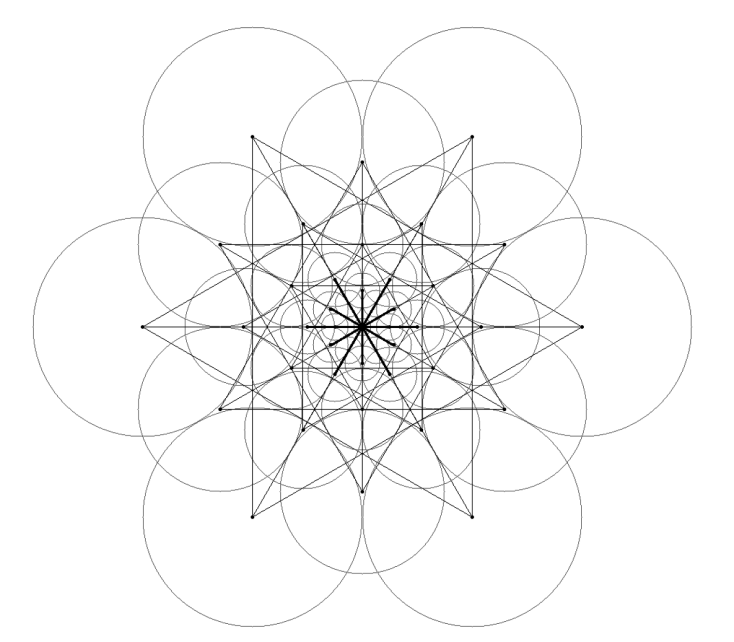


Рисунок 5. Сравниваемые подобласти дескриптора FREAK. Размер окружности отражает размер подобласти, линиями соединены сравниваемые подобласти.

На рис. 5 можно видеть, что чем дальше подобласти от центра, тем больше радиусы окружностей. Эта идея отражает идею рецептивного поля зрительной системы человека: чем ближе точка к центру поля зрения, тем больший отклик она создаёт.

# Алгоритмы, содержащие в себе детектор и дескриптор

Существует класс алгоритмов, объединяющих в себе как детектор точечных признаков, так и генератор дескрипторов. Преимущество таких методов в том, что, объединив два алгоритма в один, возможно избежать большого количества накладных коммуникационных расходов. Также такой подход позволяет объединить ускоспециализированные детектор точек и генератор дескрипторов, которые по отдельности не представляли бы ценности.

## SIFT

Детектор SIFT (Scale Invariant Feature Transform) – детектор, инвариантный к масштабированию и поворотам, то есть с достаточной степенью точности с его помощью можно найти одни и те же признаки на масштабированных и повернутых друг относительно друга изображениях. Непосредственно данный детектор в работе не используется, но его стоит упомянуть, так как следующий детектор SURF является его модификацией.

В качестве метода поиска особых точек используется поиск локальных максимумов в пространстве масштабов разности гауссиан (DoG). Пространство масштабов состоит из сверток гауссова ядра с исходным изображением в разных масштабах (коэффициент масштабирования ). Набор сверток изображения одного масштаба называется октава. В каждой октаве, состоящей из изображений, путем последовательных вычитаний изображений друг из друга получается набор разностей гауссиан.

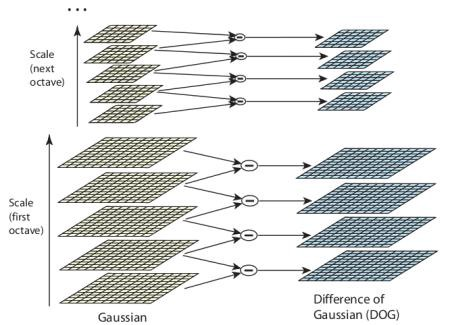


Рисунок 6. Пространство масштабов DoG.

Каждая точка сравнивается со своей окрестностью, которая состоит из 8 соседних точек на текущем масштабе, 9 точек на следующем и 9 на предыдущем масштабах:

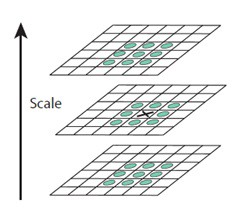


Рисунок 7. Поиск локального максимума.

Если точка является локальным максимумом, она становится точкой-кандидатом. Так как точек-кандидатов получается слишком много, точки, лежащие на границах и в слабоконтрастных областях, отбрасываются.

В качестве дескриптора точки SIFT использует гистограмму направлений. В окрестности радиуса , где - номер слоя в пространстве масштабов, для каждой точки вычисляются ее модуль и направление:

Направление отображается в 1 из 36 счетчиков, каждый из которых представляет интервал 10 градусов, начальные значения счетчиков равны нулю. При попадании направления в какой-либо интервал к его значению прибавляется соответствующий модуль. Получается гистограмма градиентов. Локальные максимумы счетчиков величиной не менее 80% глобального максимума определяют доминантные направления, которые обеспечивают инвариант относительно вращения.

Для обеспечения инварианта относительно масштабирования окрестность точки разбивается на 16 квадратов, в каждом из которых строится гистограмма с интервалом 45 градусов из значений градиентов, попавших в данный квадрат. В результате получаются 16 векторов длиной 8, из которых можно составить один вектор длиной 128, это и есть SIFT-дескриптор.

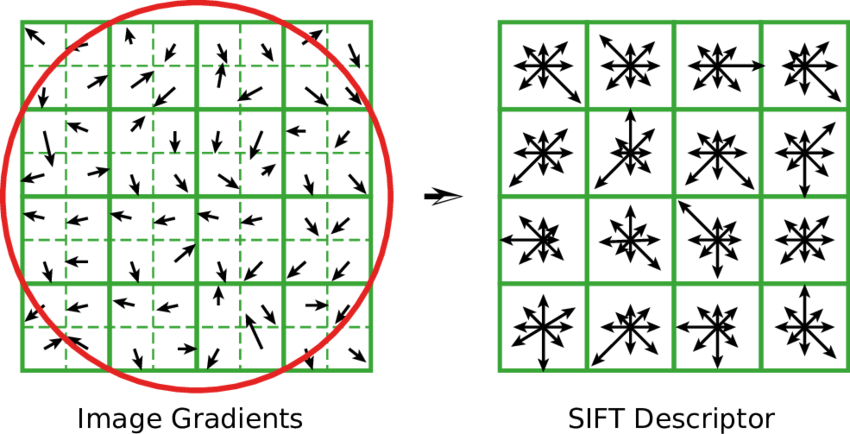


Рисунок 8. Слева – схема градиентов на окрестности точки, разбитая на 16 квадратов, справа – SIFT-дескриптор.

## SURF

Детектор SURF (Speeded-Up Robust Features) является модификацией детектора SIFT, при его разработке ставилась цель повысить эффективность работы. Так, в SIFT лаплассиан гауссианы аппроксимировался разностью гауссиан, в SURF лаплассиан гауссианы аппроксимируется бокс-фильтром:

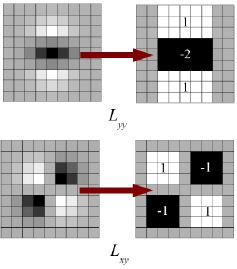


Рисунок 9. Пара слева – ядро свертки производной по y и ее упрощенная аппроксимация в SURF, пара справа – производная по диагонали и ее упрощенная аппроксимация в SURF. Элементы в серых ячейках принимают значение 0.

Так как свертку с бокс-фильтром можно производить с интегральными изображениями, достигается ощутимое повышение производительности по сравнению с SIFT. При этом во многих случаях в силу простоты функции отклика значительно возрастает локализуемость признаков[1].

В качестве метода построения дескриптора используется гистограмма градиентов, как и в SIFT, но в нем используются взвешенные суммы по прямоугольным подокнам (хаароподобные признаки), окружающим особую точку, чтобы упростить и ускорить аппроксимацию значений градиентов.

Детектор SURF широко используется в составе систем автоматической посадки [11, 12].

## ORB

Детектор ORB (Oriented FAST and Rotated BRIEF) состоит из модифицированных детектора особых точек FAST и дескриптора BRIEF.

Детектор особых точек FAST сравнивает значение яркости точки p со значением яркости окружающих его 16 пикселей: если более 8 последовательных пикселей ярче или темнее точки p, то она выбирается как особая точка (См. Рис. 2).

По умолчанию FAST не является инвариантным относительно масштаба и поворота, поэтому в детекторе ORB FAST применяется к пирамиде масштабов исходного изображения, затем применяется функция отклика детектора Харриса, чтобы выделить N наиболее характерных точек на разных масштабах. Так обеспечивается инвариантность относительно масштаба.

Окружность радиуса , используемая в детекторе особых точек, интерпретируется как окрестность точки p, для которой определяются моменты и главное направление:

BRIEF В качестве дескриптора использует битовый вектор, определяемый локальным бинарным паттерном (Local Binary Pattern). Пусть:

.

Тогда, задав i-ому биту в векторе результат сравнения i-ой пары, можно определить LBP-код длиной n бит. В BRIEF, модифицированном для ORB, предлагается использовать ранее вычисленное направление для вычисления дескриптора, инвариантного к вращению, на основе локального бинарного паттерна.



Рисунок 10. Локальный бинарный паттерн со случайно выбранными парами.

## BRISK

Детектор BRISK (название не является аббревиатурой) использует идею BRIEF — дескриптор как серия тестов. Детектор особых точек в BRISK — модификация FAST, определяющая не только ориентацию, но и масштаб особой точки. Масштаб определяется путём построения пространства масштабов и последующего построения интра-октав — промежуточных уровней пространства масштабов. Затем вычисляются отклики детектора особых точек и производится подавление немаксимумов. После этого для каждой точки вычисляется отклик детектора на предыдущем и последующем слоях, и полученные точки интерполируются квадратичной функцией. Максимум этой функции принимается за масштаб:

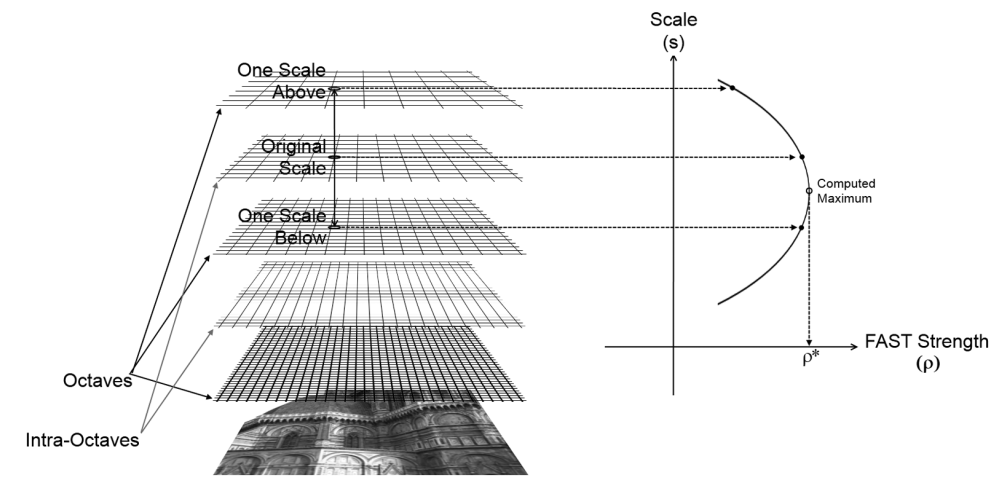


Рисунок 11. Вычисление масштаба особой точки детектором точек BRISK.

Признаки BRISK имеют не только масштаб, но и ориентацию. Она вычисляется вместе с дескриптором. Тесты дескриптора производятся между подобластями с заданными позициями и радиусами. Вокруг центра дескриптора расположены несколько концентрических окружностей, на каждой из которых расположены подобласти, обозначенные окружностями разного радиуса. При тестировании для каждой подобласти вычисляется взвешенная по Гауссу сумма, затем происходит сравнение всех подобластей со всеми. При этом для каждой подобласти существует значение d такое, что результаты сравнения с подобластями внутри окружности радиусом d используются для построения дескриптора, а результаты сравнения с подобластями вне окружности радиусом d используются для построения гистограммы направлений и вычисления ориентации.

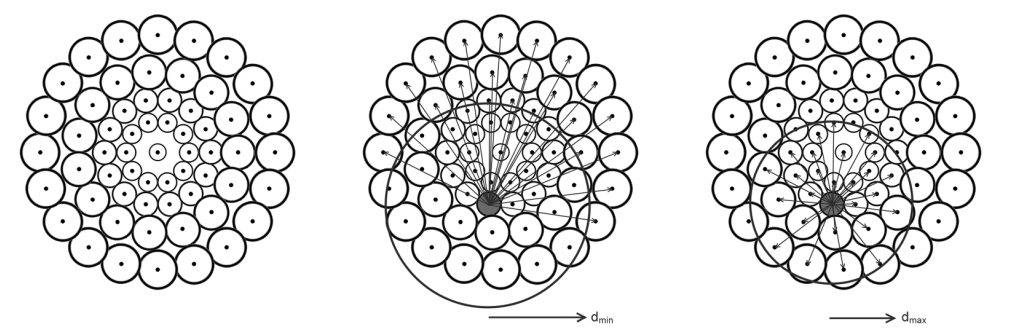


Рисунок 12. Дескриптор BRISK: слева – шаблон расположения подобластей; в центре: прямыми линиями обозначены пары, результат сравнения которых будет участвовать в вычислении ориентации; справа – прямыми линиями обозначены пары, входящие в дескриптор.

# Методы прослеживания

Можно выделить два метода прослеживания особых точек:

1. Вычисление оптического потока

2. Сопоставление дескрипторов

Оптический поток — метод, решающий задачу определения перемещения искомой точки между кадрами. В общем случае результат представляет собой поле перемещений, где каждому пикселю исходного изображения ставится в соответствие вектор перемещения. Методы вычисления оптического потока не требует ничего кроме самих точек. Его недостатки:

* из-за допущения о небольшом перемещении точек он не подходит для прослеживания значительных изменений между кадрами;
* из-за допущения об одновременном параллельном перемещении всех точек области весьма вероятны выбросы.

В данной работе будет решаться задача разреженного оптического потока (допущение о том, что искомых точек сильно меньше, чем пикселей в изображении) с помощью алгоритма Лукаса-Канадэ.

Главным недостатком второго принципа является необходимость дополнительно вычислять дескрипторы особых точек с целью прослеживать дескрипторы. Полный перебор всех дескрипторов исключает возможность работы в реальном времени, поэтому в данной работе для сопоставления дескрипторов будет использоваться библиотека FLANN, реализующая метод Approximate Nearest Neighbour — приближенный поиск ближайшего соседа.

# Обзор ускорителей и стандартов

Для алгоритмов обработки изображений характерно большое количество одинаковых операций, например, операция свёртки, выполненная для каждой подобласти изображения заданного размера. Среди архитектур современных вычислителей, подходящих для параллельного выполнения таких операций и относительно доступных, можно выделить SIMD – архитектуры, а именно:

* Видеокарты – устройства, преобразующие графический образ, хранящийся в памяти компьютера, в изображение, пригодное для вывода на монитор. Как правило, видеокарты реализуют арифметические операции с векторами чисел, что позволяет выполнить одну инструкцию одновременно для нескольких наборов аргументов. В последнее время видеокарты реализуют технику GPGPU – технику использования видеокарты для неграфических вычислений.
* CPU с поддержкой SSE и AVX – расширений набора команд векторными операциями

В данной работе использованы CPU Intel Core i5 и GPU Nvidia GTX 1050.

Среди стандартов, реализующих технику GPGPU, наиболее популярны следующие:

* CUDA – это программно-аппаратная платформа, реализованная на GPU Nvidia и позволяющая использовать ресурсы GPU для произвольных вычислений.
* OpenCL – это стандарт, описывающий интерфейс к программированию приложений, использующих параллельные вычисления на различных CPU и GPU. Главное отличие от CUDA – разработка ведется несколькими компаниями, следовательно, стандарт открыт и реализован многими производителями для своих вычислителей.

Таким образом, в данной работе используются следующие пары вычислитель – стандарт параллельности:

* Intel Core i5 + SSE/AVX;
* Nvidia GeForce GTX 1050 + CUDA;
* Nvidia GeForce GTX 1050 + OpenCL.

# Тестовые видеопоследовательности

В качестве тестовой видеопоследовательности, изображающей съемку наземного ландшафта, выбрана видеопоследовательность stereo.mp4, снятая бортовой камерой передвижного РТК. Её параметры:

* Продолжительность: 00:00:05.12
* Размер кадра: 4928x2056
* Частота кадров: 25 кадров в секунду



Рисунок 13. Кадр из видеопоследовательности stereo.mp4.

В качестве тестовой видеопоследовательности, изображающей съемку ландшафта со спутника, выбрана видеопоследовательность icy\_scarp.mp4, созданная из одного снимка подстилающей поверхности Марса, снятого орбитальным телескопом HiRise. Её параметры:

* Продолжительность: 00:00:05.06
* Размер кадра: 1920x1080
* Частота кадров: 25 кадров в секунду



Рисунок 14. Кадр из видеопоследовательности icy\_scarp.mp4.

# Методика выполнения работы

Для оценки различных детекторов и архитектур необходимо провести пробные запуски алгоритмов для каждой видеопоследовательности. Пусть D – набор детекторов, V – набор видео, A – набор архитектур. Тогда оценка будет производиться следующим образом:

Для каждого d из D:

Для каждого a из A:

Для каждого v из V:

если d не поддерживает a:

прервать итерацию, начать следующую

вычислить fps(d, a, v)

вычислить featuresFound(d, a, v)

вычислить featuresTracked(d, a, v)

Где:

* fps(d, a, v)- время работы в миллисекундах, потраченное детектором d на обработку видео v на архитектуре a;
* featuresFound(d, a, v)- количество признаков, найденных на каждом кадре видеопоследовательности;
* featuresTracked(d, a, v)- для каждого кадра: количество признаков, отслеженных от начала видеопоследовательности до текущего кадра.

# Результаты выполнения программы

В приложении представлены результаты выполнения программы для каждого алгоритма: график зависимости времени работы алгоритма от номера кадра, график зависимости количества найденных признаков от номера кадра и график зависимости количества признаков, прослеженных от первого до текущего кадра от номера кадра. Все алгоритмы были запущены с такими настройками, чтобы для каждого кадра возвращать не более нескольких сотен признаков, иначе алгоритм будет выполняться слишком долго для того, чтобы претендовать на обработку данных в реальном времени.

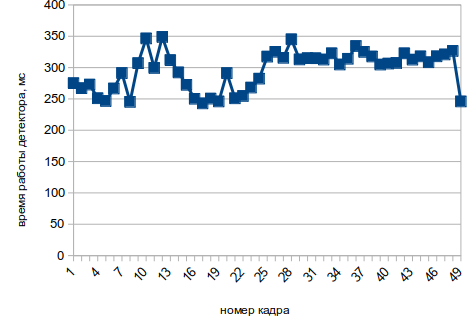


Рисунок 15. График зависимости времени работы алгоритма shi-tomasi с последующим запуском алгоритма Лукаса-Канадэ от номера кадра видеопоследовательности stereo.mp4.

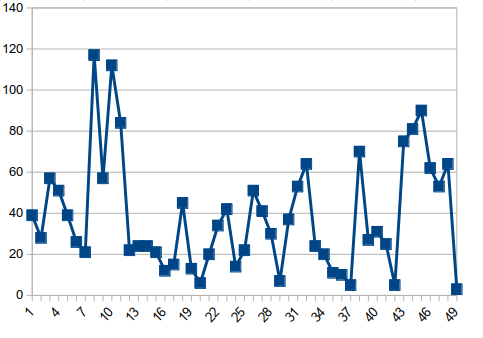


Рисунок 16. График зависимости количества признаков, найденных алгоритмом shi-tomasi от номера кадра видеопоследовательности stereo.mp4.

Из графика зависимости количества признаков от номерка кадра (рис. 17) видно, что количество признаков, извлекаемых алгоритмом Shi-Tomasi из видеопоследовательности stereo.mp4, сильно варьируется от 0 до более чем сотни, причём часто число признаков на кадре меньше пяти. Кадры с таким маленьким количеством признаков появляются довольно часто, в том числе и сразу после начала работы алгоритма, что в принципе не позволяет прослеживать признаки стабильно. Поэтому для детекторов Харриса и Shi-Tomasi графики зависимости количества прослеженных признаков от номера кадра опущены.

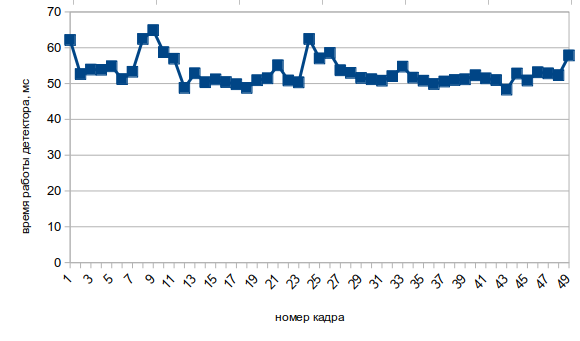


Рисунок 17. График зависимости времени работы алгоритма shi-tomasi с последующим запуском алгоритма Лукаса-Канадэ от номера кадра видеопоследовательности icy\_scarp.mp4.

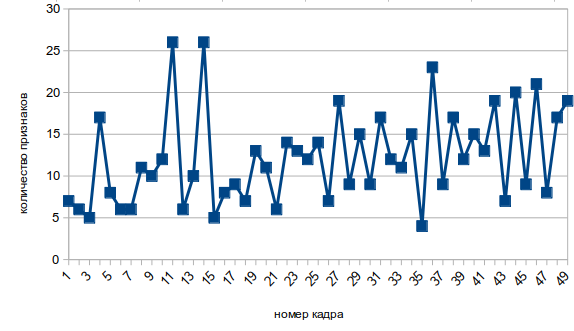


Рисунок 18. График зависимости количества признаков, найденных алгоритмом shi-tomasi от номера кадра видеопоследовательности icy\_scarp.mp4.

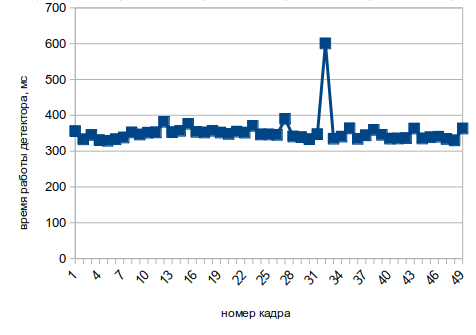


Рисунок 19. График зависимости времени работы детектора Харриса с последующим запуском алгоритма Лукаса-Канадэ от номера кадра видеопоследовательности stereo.mp4.

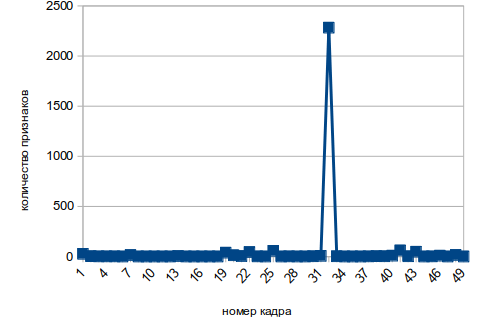


Рисунок 20. График зависимости количества признаков, найденных детектором Харриса от номера кадра видеопоследовательности stereo.mp4.

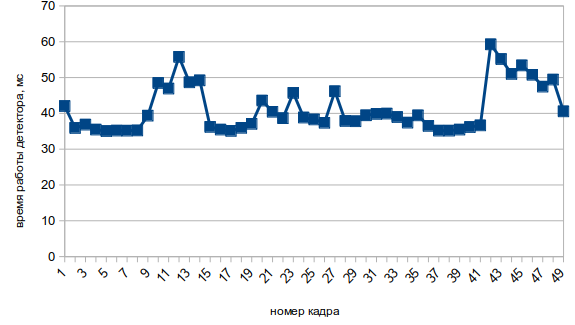


Рисунок 21. График зависимости времени работы детектора Харриса с последующим запуском алгоритма Лукаса-Канадэ от номера кадра видеопоследовательности icy\_scarp.mp4.

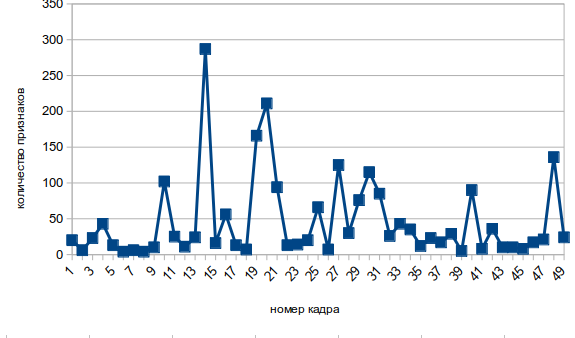


Рисунок 22. График зависимости количества признаков, найденных детектором Харриса от номера кадра видеопоследовательности icy\_scarp.mp4.

Из графика зависимости количества признаков от номерка кадра (рис. 17) видно, что количество признаков, извлекаемых алгоритмом Shi-Tomasi из видеопоследовательности stereo.mp4, сильно варьируется от 0 до более чем сотни, причём часто число признаков на кадре меньше пяти. Кадры с таким маленьким количеством признаков появляются довольно часто, в том числе и сразу после начала работы алгоритма, что в принципе не позволяет прослеживать признаки стабильно. Поэтому для детекторов Харриса и Shi-Tomasi графики зависимости количества прослеженных признаков от номера кадра опущены.

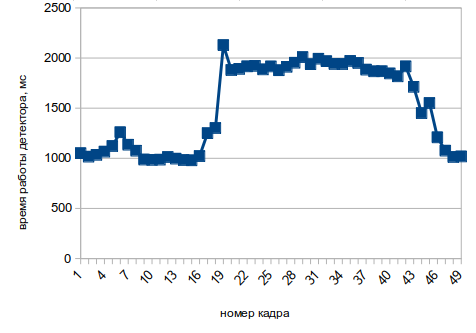


Рисунок 23. График зависимости времени работы детектора Star с последующим запуском алгоритма Лукаса-Канадэ от номера кадра видеопоследовательности stereo.mp4.

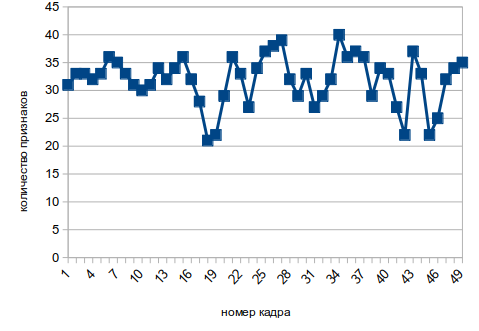


Рисунок 24. График зависимости количества признаков, найденных детектором Star с последующим запуском алгоритма Лукаса-Канадэ от номера кадра видеопоследовательности stereo.mp4.

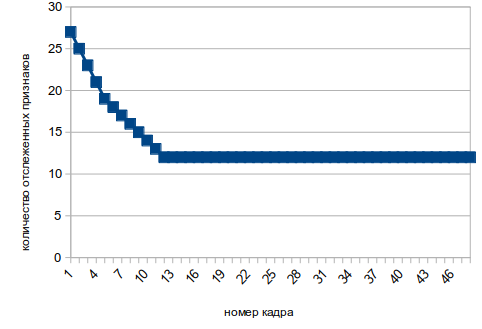


Рисунок 25. График зависимости количества признаков, обнаруженных детектором Star и прослеженных с помощью алгоритма Лукаса-Канадэ от первого кадра до текущего, от номера кадра видеопоследовательности stereo.mp4.

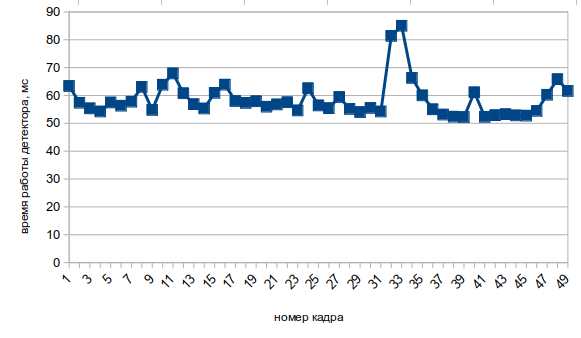


Рисунок 26. График зависимости времени работы детектора Star с последующим запуском алгоритма Лукаса-Канадэ от номера кадра видеопоследовательности icy\_scarp.mp4.

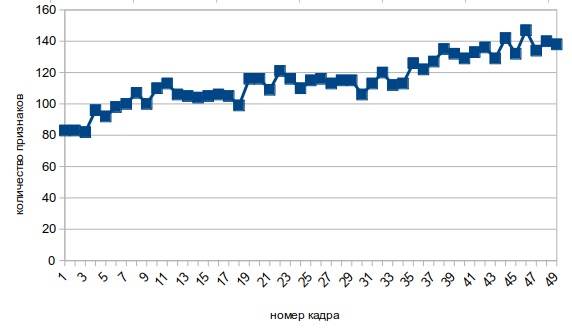


Рисунок 27. График зависимости количества признаков, найденных детектором Star от номера кадра видеопоследовательности icy\_scarp.mp4.

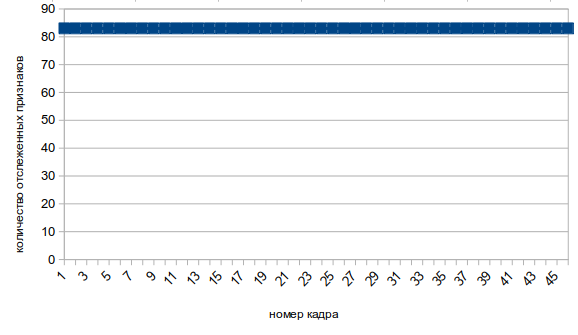


Рисунок 28. График зависимости количества признаков, обнаруженных детектором Star и прослеженных с помощью алгоритма Лукаса-Канадэ от первого кадра до текущего, от номера кадра видеопоследовательности icy\_scarp.mp4.

Из графиков зависимости количества прослеженных признаков от номерка кадра (Рис. 26 и 29) видно, что признаки, найденные детектором Star, отслеживаются очень хорошо (не более 5% ошибок). Это объясняется двумя факторами:

1. Детекторы Харриса и Shi-Tomasi, хотя и извлекают хорошо локализуемые признаки, не являются инвариантными относительно масштаба. Детектор Star, в свою очередь, был разработан специально для задач SLAM и является масштабно-независимым. Особенно хорошо это наблюдается на рис. 26, так как видеопоследовательность stereo.mp4 представляет собой входные данные для типичной задачи SLAM.
2. Количество признаков, отбираемых детектором Star, находится в достаточно маленьком диапазоне. В библиотеке OpenCV, используемой в данной работе, для детекторов Харриса и Shi-Tomasi не реализован на уровне библиотеки отбор признаков с наиболее мощным откликом. Реализация его в пространстве пользователя будет содержать различные накладные расходы.

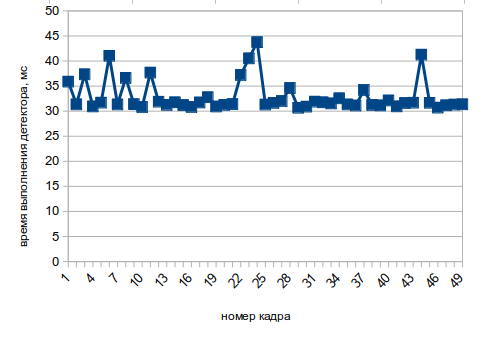


Рисунок 29. График зависимости времени работы детектора FAST с последующим запуском алгоритма Лукаса-Канадэ от номера кадра видеопоследовательности stereo.mp4.

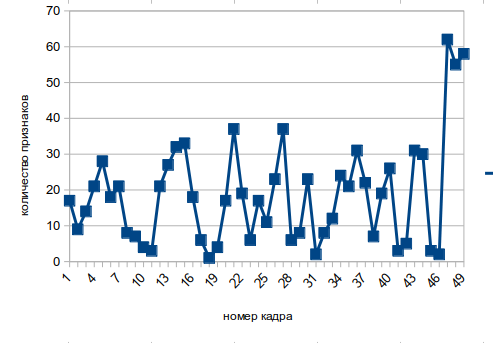


Рисунок 30. График зависимости количества признаков, найденных детектором FAST от номера кадра видеопоследовательности stereo.mp4.

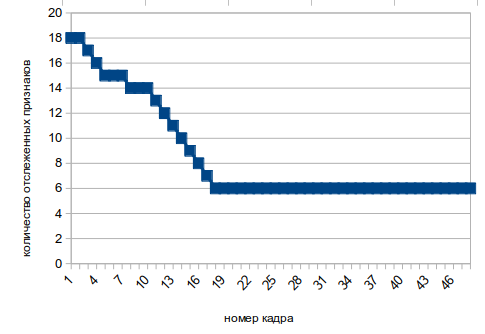


Рисунок 31. График зависимости количества признаков, обнаруженных детектором FAST и прослеженных с помощью алгоритма Лукаса-Канадэ от первого кадра до текущего, от номера кадра видеопоследовательности stereo.mp4.

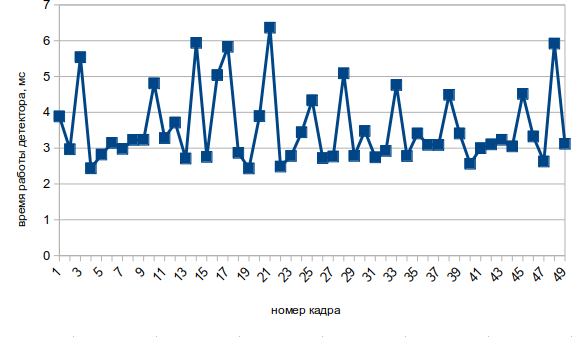


Рисунок 32. График зависимости времени работы детектора FAST с последующим запуском алгоритма Лукаса-Канадэ от номера кадра видеопоследовательности icy\_scarp.mp4.

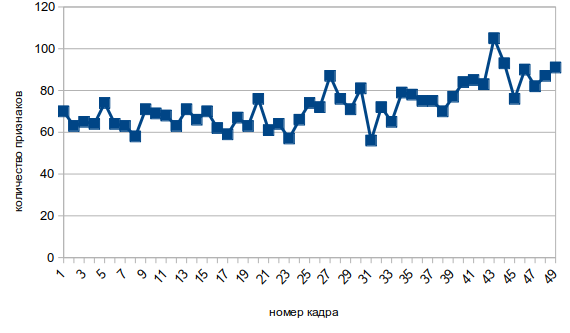


Рисунок 33. График зависимости количества признаков, найденных детектором FAST от номера кадра видеопоследовательности icy\_scarp.mp4.

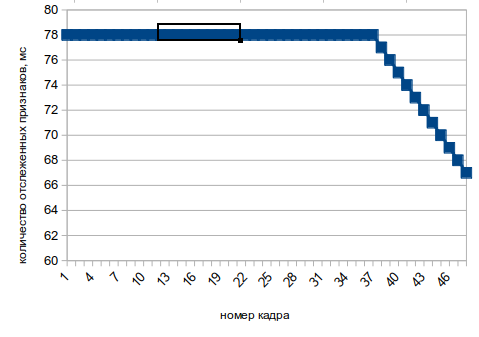


Рисунок 34. График зависимости количества признаков, обнаруженных детектором FAST и прослеженных с помощью алгоритма Лукаса-Канадэ от первого кадра до текущего, от номера кадра видеопоследовательности icy\_scarp.mp4.

Из графиков зависимости времени работы детектора от номера кадра (Рис. 30 и 33) видно, что детектор FAST удовлетворяет критерию работы в реальном времени. В силу принципа работы (найти n последовательных пикселей вокруг некого центра, которые будут темнее или светлее этого центра) детектор FAST хорошо извлекает признаки из границ рельефа, хребтов, и прочих элементов изображения, контрастирующих с фоном:

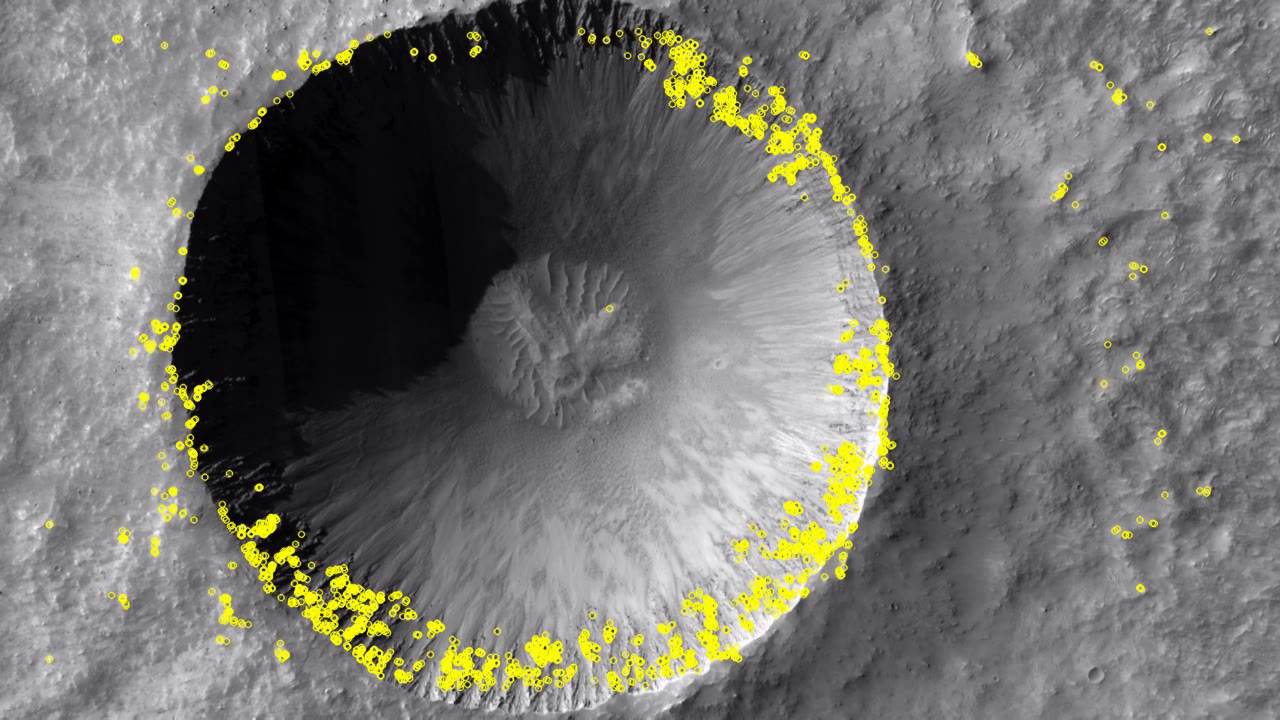


Рисунок 35. Пример результата работы детектора FAST на изображении кратера.

Главный недостаток детектора FAST – кучность признаков, которая сохраняется даже с учётом подавления немаксимумов. Из-за этого большая часть признаков с помощью алгоритма Лукаса-Канадэ отслеживается неверно, и визуально в больших скоплениях признаков количество ошибок может превышать 50% от общего числа прослеженных признаков, поэтому график на рис. 35 не является репрезентативным.

Ранее был сделан вывод, что для видеопоследовательностей из исходных данных с учётом ограничений на время работы детекторы Харриса и Shi-Tomasi без соответствующей постобработки могут вернуть либо слишком много признаков, либо, что происходит чаще (судя по графикам на рис.17, 21 и 23), слишком мало признаков, что не позволяет использовать их для отслеживания. Поэтому далее следующие алгоритмы:

* детектор Харриса + BRIEF + FLANN;
* детектор Харриса + FREAK + FLANN;
* детектор Shi-Tomasi + BRIEF + FLANN;
* детектор Shi-Tomasi + FREAK + FLANN;
* рассмотрены не будут.

# Интерпретация результатов и вывод

На основании представленных выше графиков можно сделать три наблюдения:

* 1. Количество признаков, найденных алгоритмами с сопоставлением дескрипторов в качестве принципа прослеживания, находится в меньшем интервале, чем количество признаков, найденных алгоритмами, использующими оптический поток (рис. 17, 19, 21, 23).
  2. Минимального количества признаков, найденных алгоритмами с сопоставлением дескрипторов, всегда достаточно для прослеживания.
  3. Графики зависимости количества прослеженных признаков от номера кадра для детекторов BRISK, SURF и ORB похожи, то есть имеют близкие точки экстремумов и перегибов.

Из этих наблюдений можно сделать вывод о том, что для исходных видеопоследовательностей, используемых в данной работе, алгоритмы на основе сопоставления дескрипторов признаков работают более стабильно, чем алгоритмы на основе оптического потока. При этом, учитывая их поведение, можно сделать вывод о том, что алгоритмы на основе оптического потока будут работать стабильно для видеопоследовательностей с неподвижным фоном и отдельными перемещающимися объектами.

Далее, для видеопоследовательности stereo.mp4 среди всех алгоритмов можно выделить BRISK на CPU, который вместе с запуском FLANN в среднем отрабатывает за 80 миллисекунд. Хотя данный результат и не удовлетворяет критерию работы в реальном времени, он наиболее близок к нему. Другой кандидат – ORB на CUDA, который отрабатывает в среднем за 37 миллисекунд. Однако, для его корректной работы в состав РТК необходимо включить видеокарту от NVIDIA, что повлечёт за собой рост себестоимости РТК и уменьшает стойкость к внешним воздействиям (видеокарту легко повредить механически). Необходимо также добавить, что размер кадра в stereo.mp4 слишком большой, и включение в состав РТК стереопары с меньшим разрешением кадра снизит требования к вычислителю.

Для видеопоследовательности icy\_scarp.mp4 можно также выделить BRISK на CPU, так как он в среднем обрабатывает один кадр со скоростью 20 миллисекунд, что удовлетворяет требование работы в реальном времени. Также можно выделить SURF и ORB на платформе CUDA, но они, как было указано ранее, требуют более дорогое и менее надёжное оборудование в составе РТК.

В целом, на основании графиков зависимости времени работы от номера кадра детекторов SURF и ORB на платформах CPU и CUDA можно сделать вывод о том, что параллельные вычисления значительно ускоряют алгоритмы обнаружения и прослеживания особых точек.

# Заключение

В данной работе были проанализированы различные подходы к вычислению и прослеживанию особых точек: были выбраны вычислители, принципы прослеживания, проанализированы результаты, полученные из исходных данных. При решении задачи прослеживания признаков было показано, что как для видеопоследовательностей, изображающих наземную съёмку ландшафта, так и орбитальную, выдвинутым в начале работы критериям удовлетворяют детекторы BRISK на CPU и ORB на CUDA. Также практические результаты помогли выяснить, что технологии параллельных вычислений значительно ускоряют вычисления, связанные с поблочной обработкой изображений.

**Список литературы**

1. Adrian Kaehler, Gary Bradski: Learning OpenCV 3; Release date: December 2016; Publisher: O'Reilly Media, Inc.; ISBN: 9781491937990.
2. Alhafni, Fernando, Lays, Ribeiro, Park, Lee, Mapping Areas using Computer Vision Algorithms and Drones; University of Bridgeport. Bridgeport, CT, 06606.
3. Computer Vision: A Modern Approach (2nd Edition). Pearson; 2 edition (November 5, 2011); ISBN-13: 978-0136085928.
4. David Nistér, Oleg Naroditsky, and James Bergen Sarnoff Corporation. Visual odometry for ground vehicle applications. CN5300 Princeton, New Jersey 08530.
5. Johnson Andrew, Aaron Seth, Chang Johnny, Cheng Yang, Montgomery James, Mohan Swati, Schroeder Steven, Tweddle Brent, Trawny Nikolas, Zheng Jason. The Lander Vision System for Mars 2020 Entry Descent and Landing. AAS Guidance Navigation and Control Conference, Breckenridge, Colorado, February 2-7, 2017.
6. Johnson, A., Willson, R., Cheng, Y. et al. Design Through Operation of an Image-Based Velocity Estimation System for Mars Landing. Int J Comput Vision 74, 319–341 (2007).
7. Labbé, M, Michaud, F. RTAB‐Map as an open‐source lidar and visual simultaneous localization and mapping library for large‐scale and long‐term online operation. J Field Robotics. 2019; 35: 416– 446.
8. Lanza P., Noceti N., Maddaleno C., Toma A., Zini L., Odone F. (2012) A Vision-Based Navigation Facility for Planetary Entry Descent Landing. In: Fusiello A., Murino V., Cucchiara R. (eds) Computer Vision – ECCV 2012. Workshops and Demonstrations. ECCV 2012. Lecture Notes in Computer Science, vol 7584. Springer, Berlin, Heidelberg.
9. Matthies, L., Maimone, M., Johnson, A. et al. Computer Vision on Mars. Int J Comput Vis 75, 67–92 (2007).
10. Reinhard Klette. 2014. Concise Computer Vision: An Introduction into Theory and Algorithms. Springer Publishing Company, Incorporated.
11. Willson, R.G. & Johnson, Andrew & Goguen, J.D.. (2005). MOC2DIMES: A Camera Simulator for the Mars Exploration Rover Descent Image Motion Estimation System. European Space Agency, (Special Publication) ESA SP. 603. 102.
12. Zini L., Odone F., Verri A., Lanza P., Marcer A. (2011) Relative Pose Estimation for Planetary Entry Descent Landing. In: Koch R., Huang F. (eds) Computer Vision – ACCV 2010 Workshops. ACCV 2010. Lecture Notes in Computer Science, vol 6469. Springer, Berlin, Heidelberg.

# Приложение

Ниже представлены результаты выполнения алгоритмов на тестовых видеопоследовательностях.

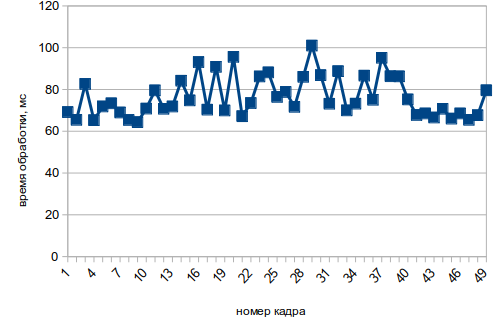


Рисунок 36. График зависимости времени работы детектора BRISK с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности stereo.mp4.

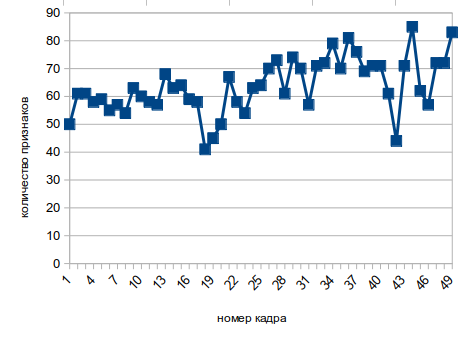


Рисунок 37. График зависимости количества признаков, найденных детектором BRISK от номера кадра видеопоследовательности stereo.mp4.

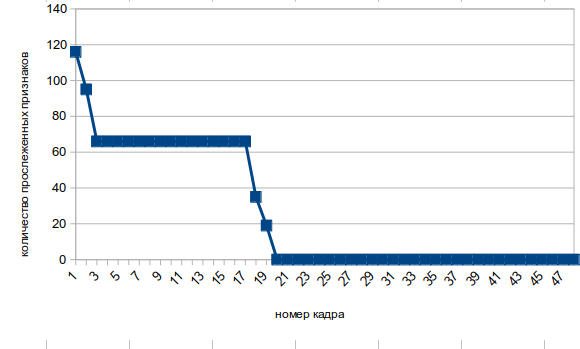


Рисунок 38. График зависимости количества признаков, обнаруженных детектором BRISK и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности stereo.mp4.

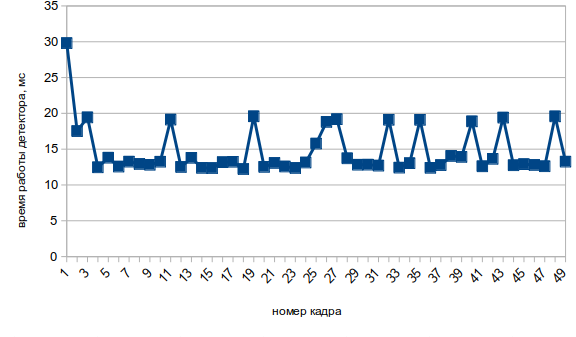


Рисунок 39. График зависимости времени работы детектора BRISK с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности icy\_scarp.mp4.

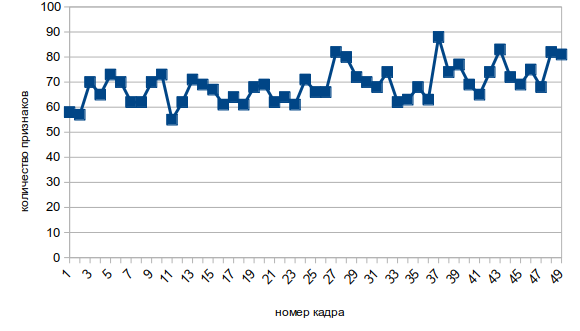


Рисунок 40. График зависимости количества признаков, найденных детектором BRISK от номера кадра видеопоследовательности icy\_scarp.mp4.

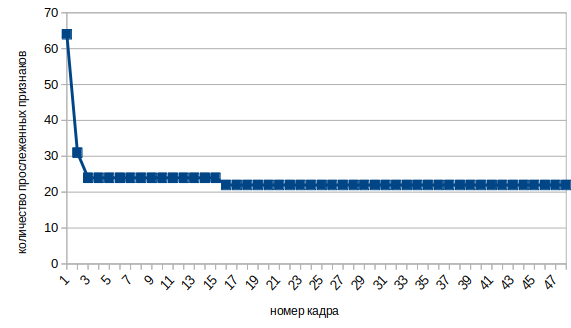


Рисунок 41. График зависимости количества признаков, обнаруженных детектором BRISK и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности icy\_scarp.mp4.

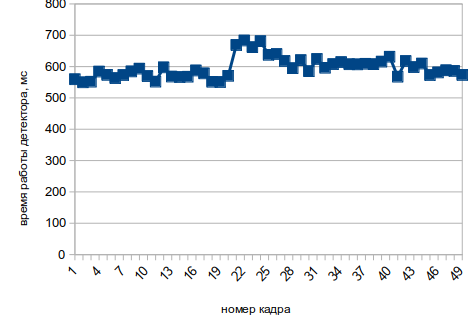


Рисунок 42. График зависимости времени работы детектора SURF на CPU с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности stereo.mp4.

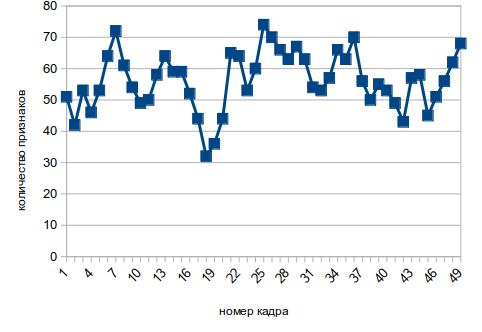


Рисунок 43 .График зависимости количества признаков, найденных детектором SURF на CPU от номера кадра видеопоследовательности stereo.mp4.

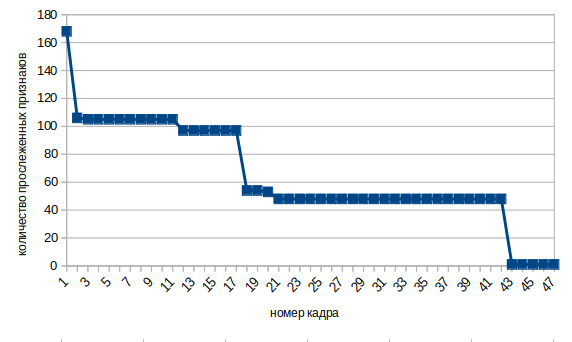


Рисунок 44. График зависимости количества признаков, обнаруженных детектором SURF на CPU и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности stereo.mp4.

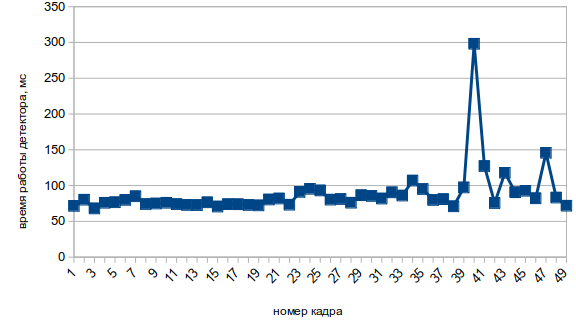


Рисунок 45. График зависимости времени работы детектора SURF на CPU с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности icy\_scarp.mp4.

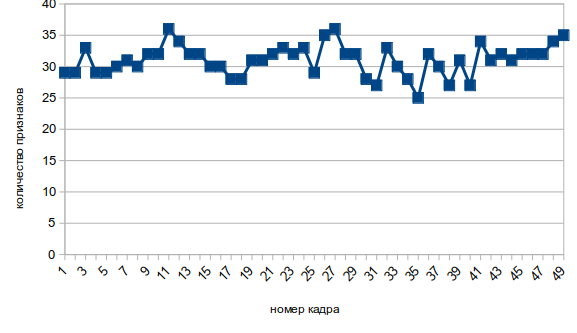


Рисунок 46. График зависимости количества признаков, найденных детектором SURF на CPU от номера кадра видеопоследовательности icy\_scarp.mp4.

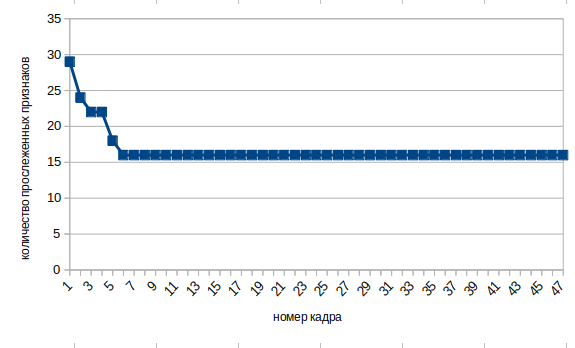


Рисунок 47. График зависимости количества признаков, обнаруженных детектором SURF на CPU и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности icy\_scarp.mp4.

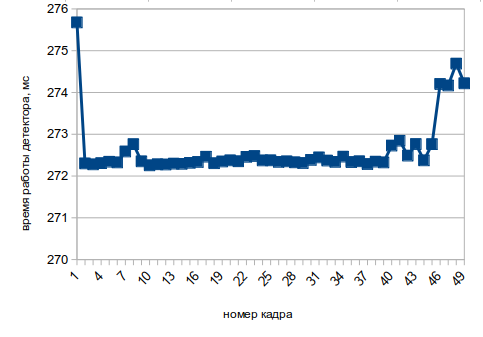


Рисунок 48. График зависимости времени работы детектора SURF на CUDA с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности stereo.mp4.

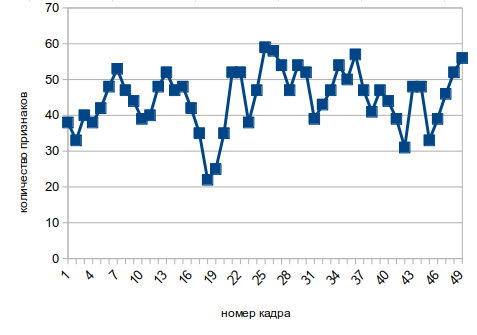


Рисунок 49. График зависимости количества признаков, найденных детектором SURF на CUDA от номера кадра видеопоследовательности stereo.mp4.

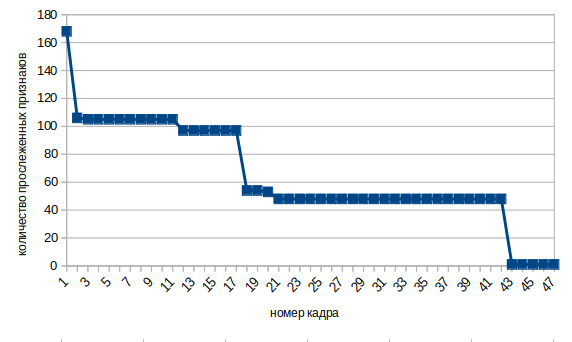


Рисунок 50. График зависимости количества признаков, обнаруженных детектором SURF на CUDA и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности stereo.mp4.

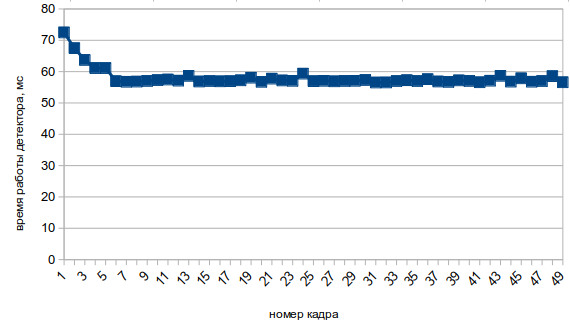


Рисунок 51. График зависимости времени работы детектора SURF на CUDA с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности icy\_scarp.mp4.

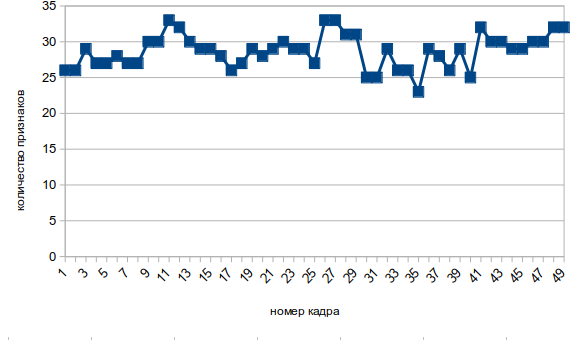


Рисунок 52. График зависимости количества признаков, найденных детектором SURF на CUDA от номера кадра видеопоследовательности icy\_scarp.mp4.

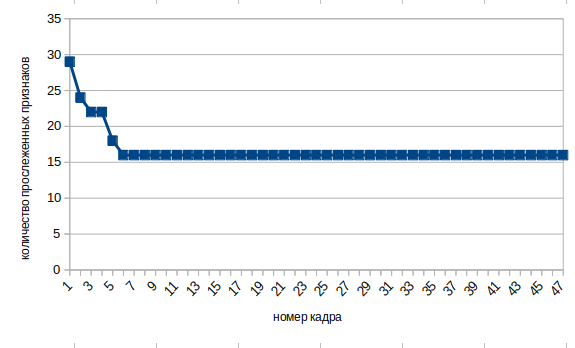


Рисунок 53. График зависимости количества признаков, обнаруженных детектором SURF на CUDA и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности icy\_scarp.mp4.

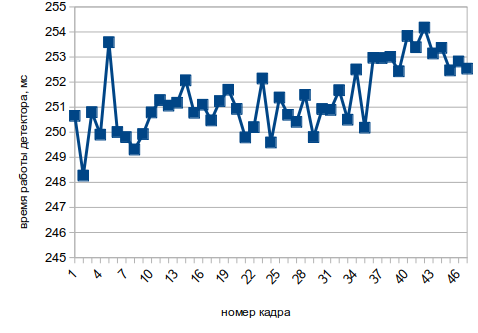


Рисунок 54. График зависимости времени работы детектора SURF на OpenCL с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности stereo.mp4.

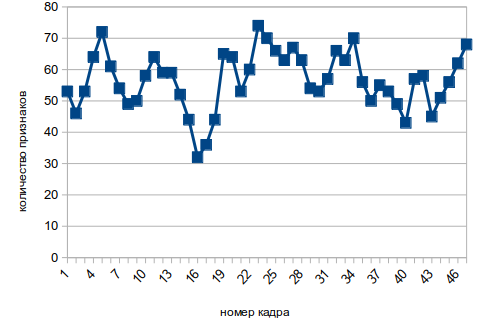


Рисунок 55. График зависимости количества признаков, найденных детектором SURF на OpenCL от номера кадра видеопоследовательности stereo.mp4.

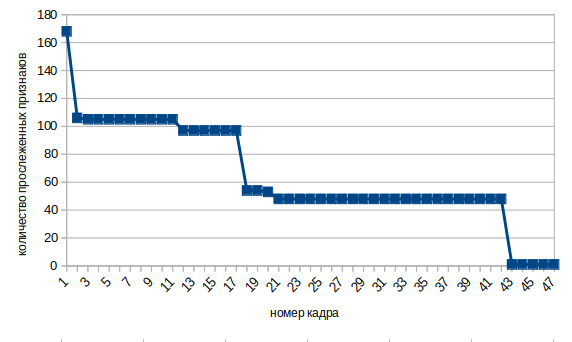


Рисунок 56. График зависимости количества признаков, обнаруженных детектором SURF на OpenCL и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности stereo.mp4.

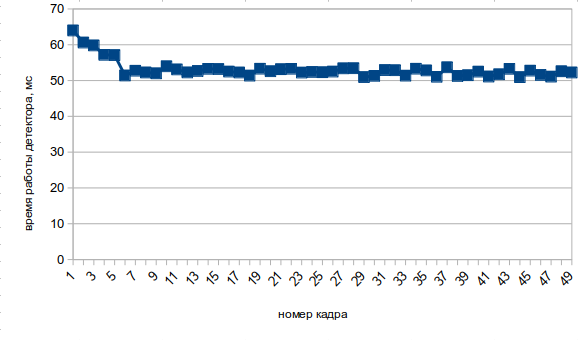


Рисунок 57. График зависимости времени работы детектора SURF на OpenCL с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности icy\_scarp.mp4.

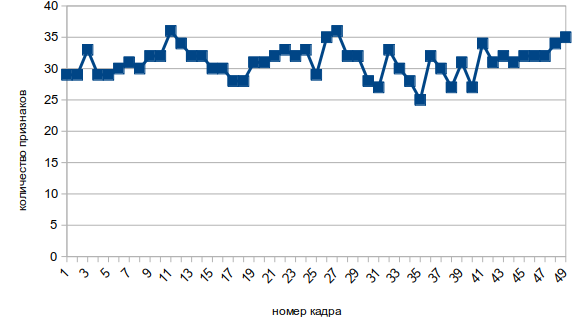


Рисунок 58. График зависимости количества признаков, найденных детектором SURF на OpenCL от номера кадра видеопоследовательности icy\_scarp.mp4.

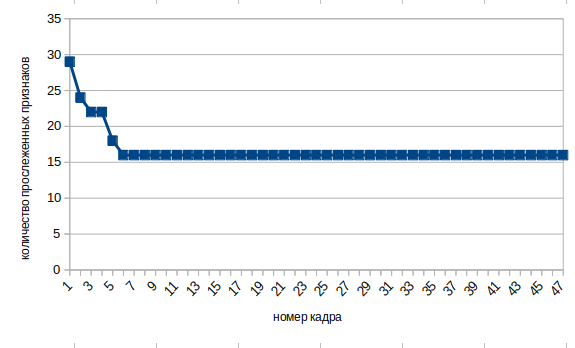


Рисунок 59. График зависимости количества признаков, обнаруженных детектором SURF на OpenCL и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности icy\_scarp.mp4.

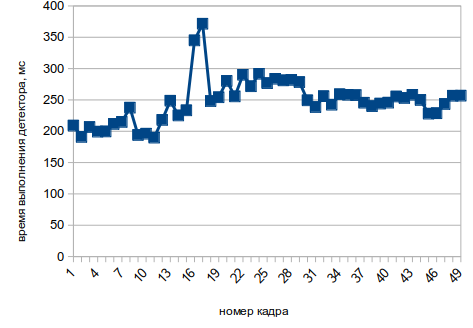


Рисунок 60. График зависимости времени работы детектора ORB на CPU с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности stereo.mp4.



Рисунок 61. График зависимости количества признаков, найденных детектором ORB на CPU от номера кадра видеопоследовательности stereo.mp4.

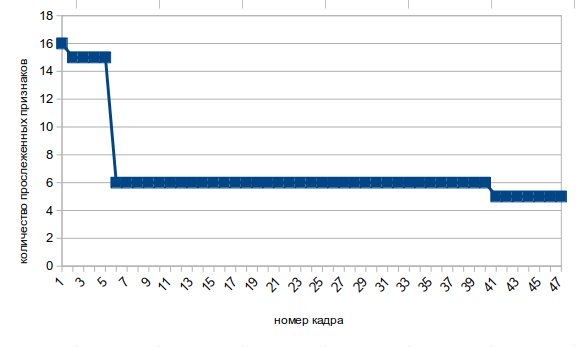


Рисунок 62. График зависимости количества признаков, обнаруженных детектором ORB на CPU и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности stereo.mp4.

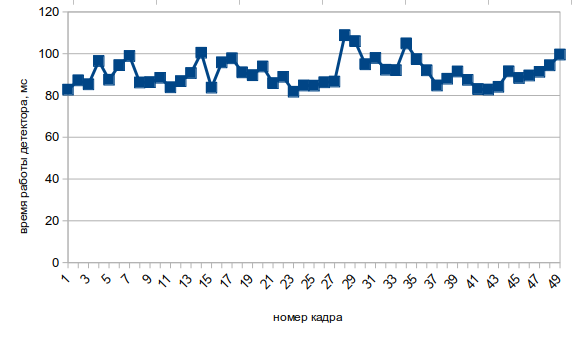


Рисунок 63. График зависимости времени работы детектора ORB на CPU с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности icy\_scarp.mp4.

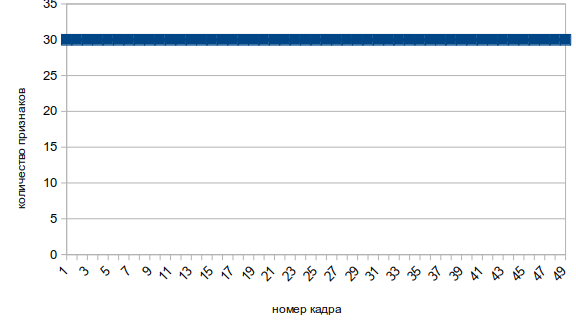


Рисунок 64. График зависимости количества признаков, найденных детектором ORB на CPU от номера кадра видеопоследовательности icy\_scarp.mp4.



Рисунок 65. График зависимости количества признаков, обнаруженных детектором ORB на CPU и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности icy\_scarp.mp4.

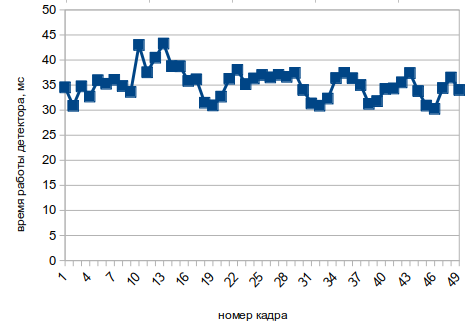


Рисунок 66. График зависимости времени работы детектора ORB на CUDA с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности stereo.mp4.

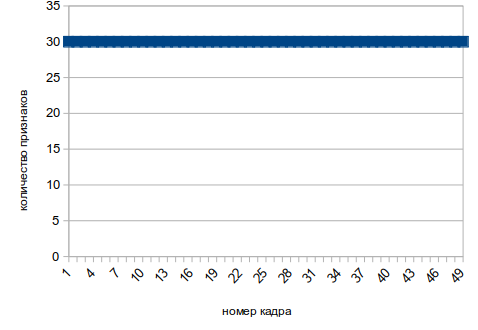


Рисунок 67. График зависимости количества признаков, найденных детектором ORB на CUDA от номера кадра видеопоследовательности stereo.mp4.

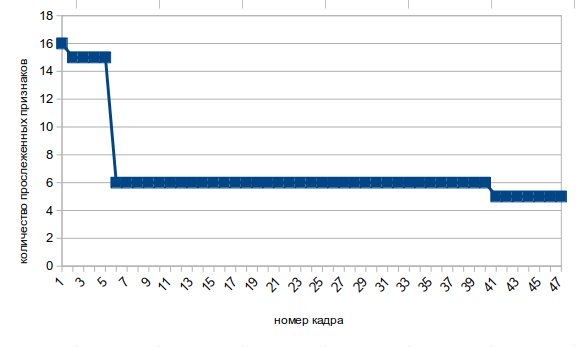


Рисунок 68. График зависимости количества признаков, обнаруженных детектором ORB на CUDA и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности stereo.mp4.

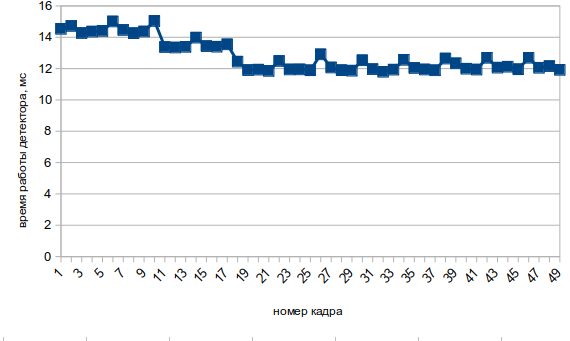


Рисунок 69. График зависимости времени работы детектора ORB на CUDA с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности icy\_scarp.mp4.

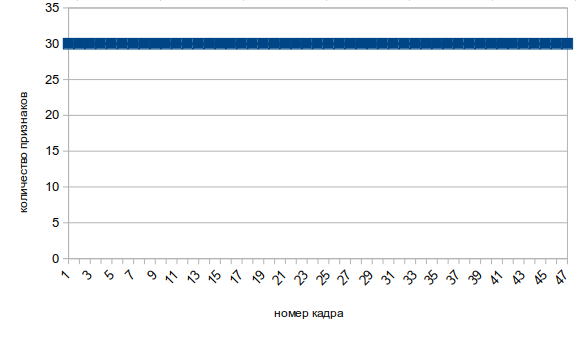


Рисунок 70. График зависимости количества признаков, найденных детектором ORB на CUDA от номера кадра видеопоследовательности icy\_scarp.mp4.



Рисунок 71. График зависимости количества признаков, обнаруженных детектором ORB на CUDA и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности icy\_scarp.mp4.

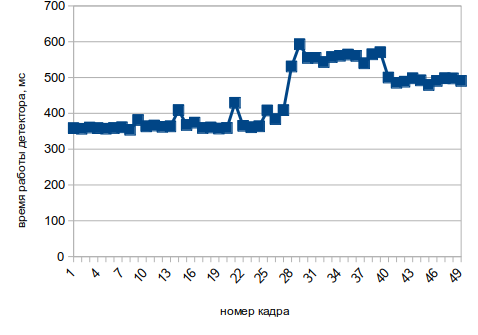


Рисунок 72. График зависимости времени работы детектора ORB на OpenCL с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности stereo.mp4.

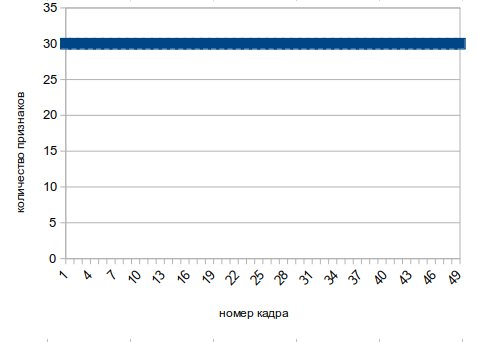


Рисунок 73. График зависимости количества признаков, найденных детектором ORB на OpenCL от номера кадра видеопоследовательности stereo.mp4.

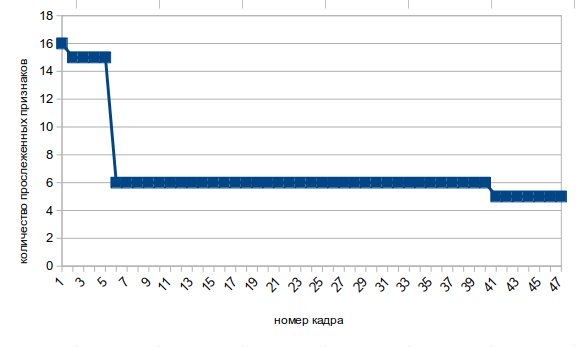


Рисунок 74. График зависимости количества признаков, обнаруженных детектором ORB на OpenCL и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности stereo.mp4.

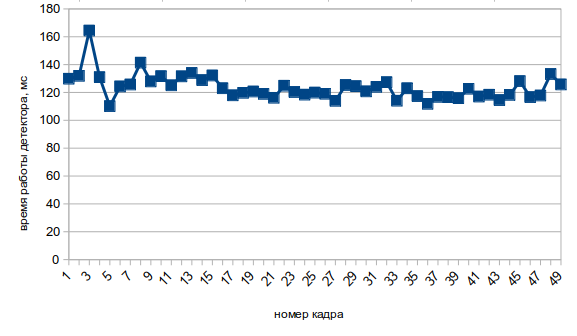


Рисунок 75. График зависимости времени работы детектора ORB на OpenCL с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности icy\_scarp.mp4.

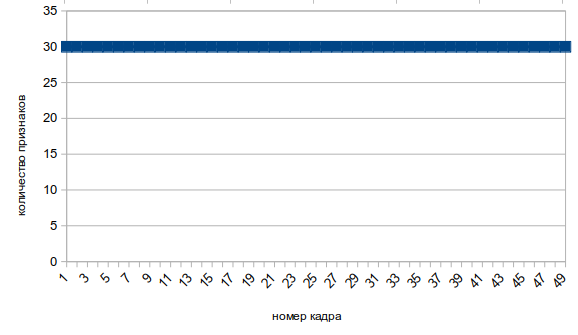


Рисунок 76. График зависимости количества признаков, найденных детектором ORB на OpenCL от номера кадра видеопоследовательности icy\_scarp.mp4.



Рисунок 77. График зависимости количества признаков, обнаруженных детектором ORB на OpenCL и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности icy\_scarp.mp4.