**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**имени М.В.ЛОМОНОСОВА**

**Факультет космических исследований**

**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

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 картографированием местности с помощью летательных аппаратов с бортовыми камерами. [https://arxiv.org/pdf/1901.00211.pdf]
* Работы, связанные с определением характеристик движения летательных аппаратов. Определение компонент движения посадочного модуля марсианских автоматических станций Spirit и Opportunity описано в [10, 11].

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. При этом изображения ландшафтов могут быть получены как с помощью наземной съемки, так и посредством съёмки с летательного аппарата. В качестве первых этапов обработки в рассматриваемых системах выполняется алгоритм, состоящий из двух логических частей:

1. детектор особых точек

2. метод прослеживания

Таким образом, цель данной работы — выбрать алгоритмы (детектор и прослеживатель), которые удовлетворяли бы требованиям скорости и надёжности для каждого из типов ландшафтных изображений: обрабатывать не менее 25 кадров в секунду и стабильно отслеживать признаки на нескольких последовательных кадрах.

Для достижения поставленной цели небходимо выполнить следующие задачи:

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

ДЕТЕТКОРЫ

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 собствевнных значений больше порога точка p рассматривается как угол (зелёная область).

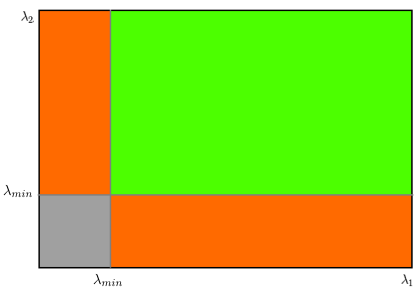


Рис. 1: диапазоны значений функции отклика детектора Shi-Tomasi.

Детектор Shi-Tomasi показывает лучшие, чем детектор Харриса, результаты при использовании алгоритма оптического потока[shi-tomasi paper 1994].

Детектор FAST

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

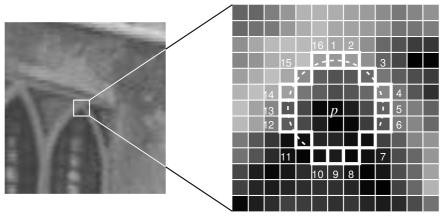


Рис. 2: детектор особых точек FAST

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

Детектор Star

Детектор особых точек Star был разработан специально для задач визуальной одометрии и SLAM. 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 интергральными изображениями, как в SURF:

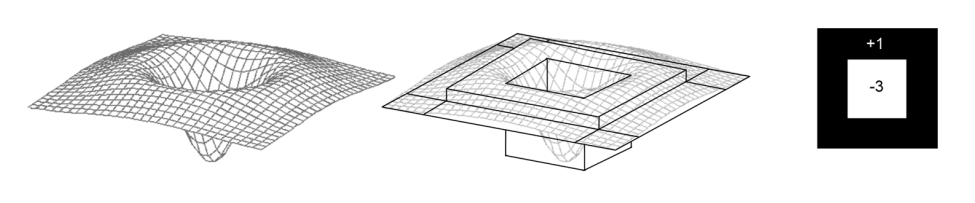


Рис. 3: аппроксимация DoG в Star: слева – функция DoG от двух аргументов; по центру – приближение DoG целыми значениями; справа – ядро фильтра, реализованное в OpenCV

ДЕСКРИПТОРЫ

Дескриптор BRIEF

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

ГОТОВЫЕ РЕШЕНИЯ ДЕТЕКТОР + ДЕСКРИПТОР

Детектор SIFT

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 использует его принципы.

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 вычситаний изображений друг из друга получается набор разностей гауссиан.

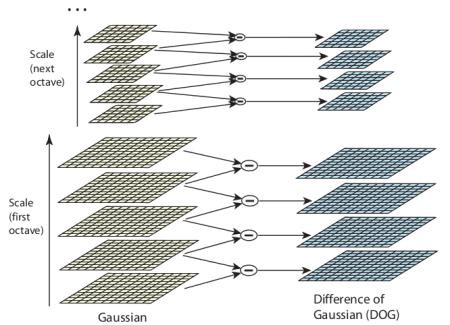


Рис. 1: пространство масштабов DoG

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

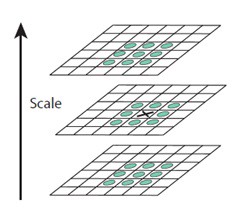


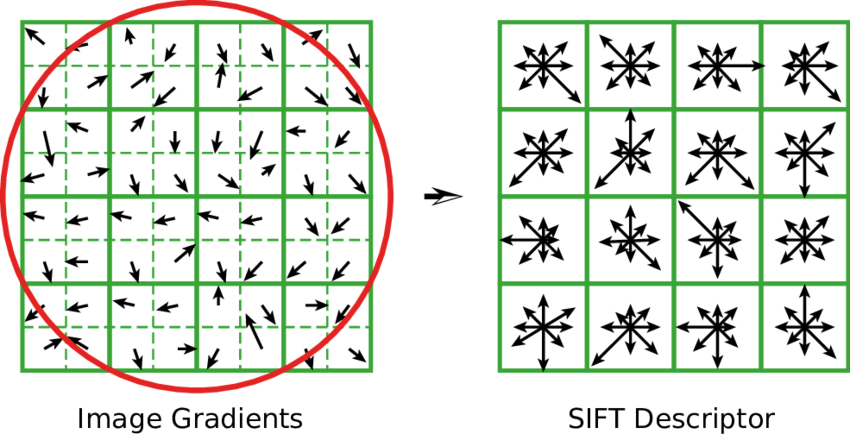
Рис. 2: поиск локального максимума

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.

Рис. 3: слева – схема градиентов на окрестности точки, разбитая на 16 квадратов, справа – SIFT-дескриптор.

Детектор SURF

SURF (Speeded-Up Robust Features) основан на тех же идеях, что и SIFT, при его разработке ставилась цель повысить эффективность работы. Так, в SIFT лаплассиан гауссианы аппроксимировался разностью гауссиан, в SURF лаплассиан гауссианы аппроксимируется бокс-фильтром:

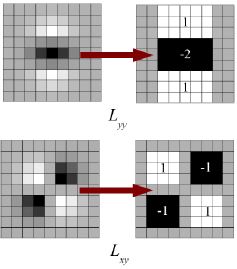


Рис. 4: пара слева – ядро свертки производной по y и ее упрощенная аппроксимация в SURF, пара справа – производная по диагонали и ее упрощенная аппроксимация в SURF. Элементы в серых ячейках принимают значение 0.

Так как свертку с бокс-фильтром можно производить с интегральными изображениями, достигается ощутимое повышение производительности по сравнению с SIFT.

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 [8, 9].

Детектор ORB

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 пикселей ярче или темнее точки p, то она выбирается как особая точка:

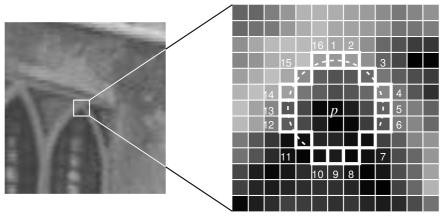


Рис. 5: детектор особых точек FAST

По умолчанию FAST не является инвариантным относительно масштаба и поворота, поэтому в детекторе ORB FAST применяется к пирамиде масштабов исходного изображения, затем применяется “мера угловатости” 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 определются моменты и главное направление:

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

Тогда выражение:

определяет LBP-код длиной n бит. В BRIEF, мдоифицированном для ORB, предлагается использовать ранее вычисленное направление для вычисления дескриптора, инвариантного к вращению, на основе локального бинарного паттерна.

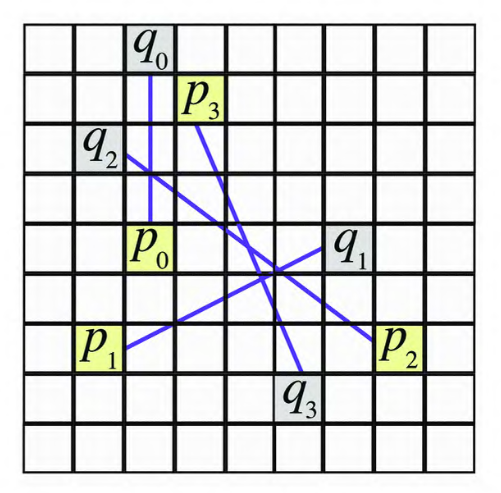


Рис. 6: локальный бинарный паттерн со случайно выбранными парами

ПРИНЦИПЫ ПРОСЛЕЖИВАНИЯ

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 [ссылка на таксономию Флинна], а именно:

* Видеокарты – устройства, преобразующие графическаий 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 тезнику 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 прораммированию 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

Рис. 1: 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

Рис. 1: 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

Ниже 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.

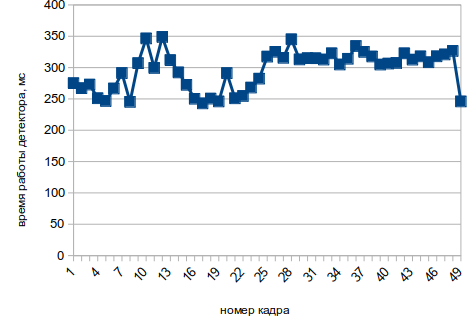


Рис. 1: график зависимости времени работы алгоритма shi-tomasi с последующим запуском алгоритма Лукаса-Канадэ от номера кадра видеопоследовательности stereo.mp4

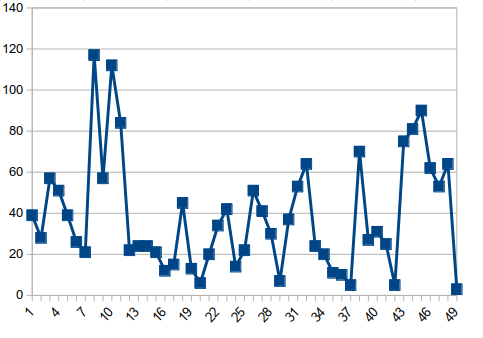
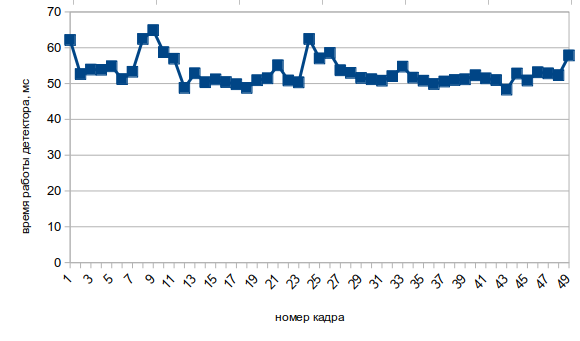
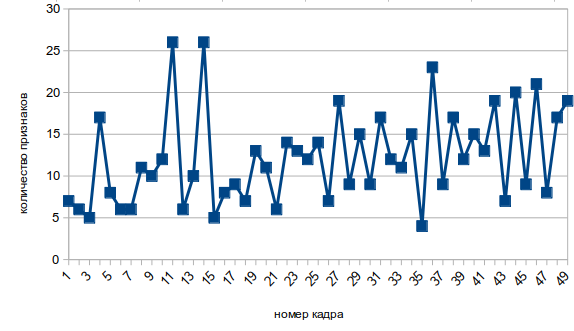


Рис. 2: график зависимости количества признаков, найденных алгоритмом shi-tomasi от номера кадра видеопоследовательности stereo.mp4

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

Рис. 3: график зависимости времени работы алгоритма shi-tomasi с последующим запуском алгоритма Лукаса-Канадэ от номера кадра видеопоследовательности icy\_scarp.mp4

Рис. 4: график зависимости количества признаков, найденных алгоритмом shi-tomasi от номера кадра видеопоследовательности icy\_scarp.mp4

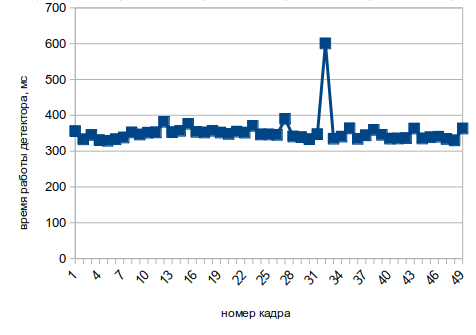


Рис. 5: график зависимости времени работы дететкора Харриса с последующим запуском алгоритма Лукаса-Канадэ от номера кадра видеопоследовательности stereo.mp4

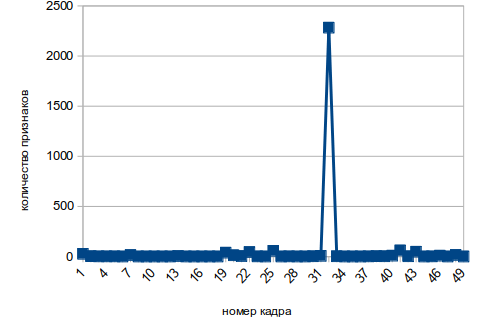


Рис. 6: график зависимости количества признаков, найденных детектором Харриса от номера кадра видеопоследовательности stereo.mp4

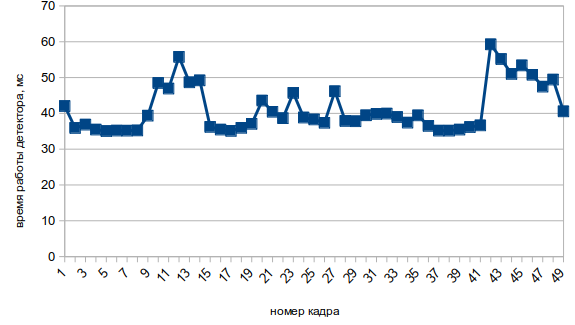


Рис. 7: график зависимости времени работы дететкора Харриса с последующим запуском алгоритма Лукаса-Канадэ от номера кадра видеопоследовательности icy\_scarp.mp4

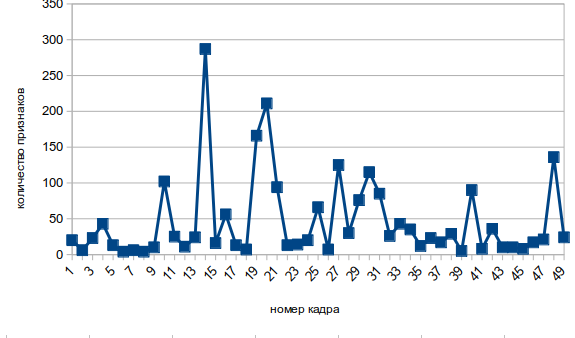


Рис. 8: график зависимости количества признаков, найденных детектором Харриса от номера кадра видеопоследовательности icy\_scarp.mp4

Из графика зависимости количества признаков от номерка кадра видно, что количество признаков, извлекаемых детектором Харриса из видеопоследовательности stereo.mp4, варьируется сильнее, чем на графике аналогичной зависимости для детектора Shi-Tomasi.

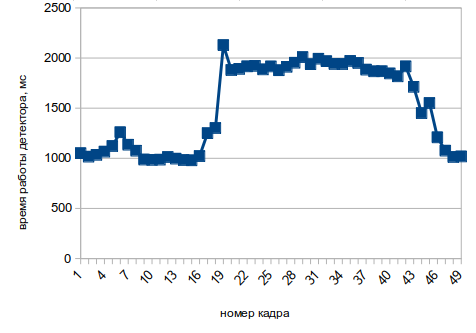


Рис. 9: график зависимости времени работы дететкора Star с последующим запуском алгоритма Лукаса-Канадэ от номера кадра видеопоследовательности stereo.mp4

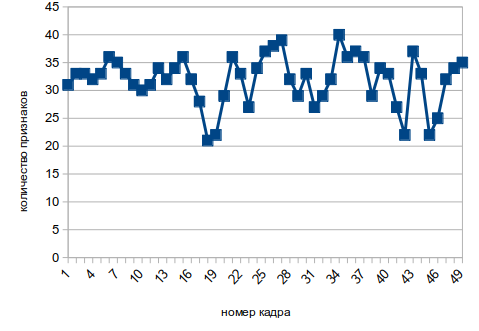


Рис. 10: график зависимости количества признаков, найденных детектором Star от номера кадра видеопоследовательности stereo.mp4

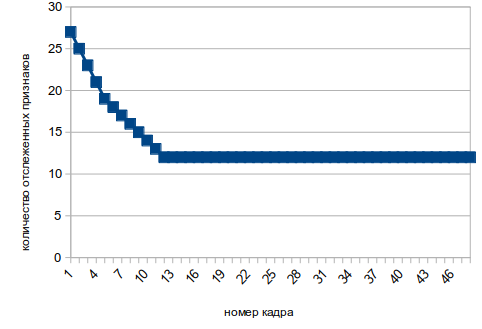


Рис. 11: график зависимости количества признаков, обнаруженных дететкором Star и прослеженных с помощью алгоритма Лукаса-Канадэ от первого кадра до текущего, от номера кадра видеопоследовательности stereo.mp4

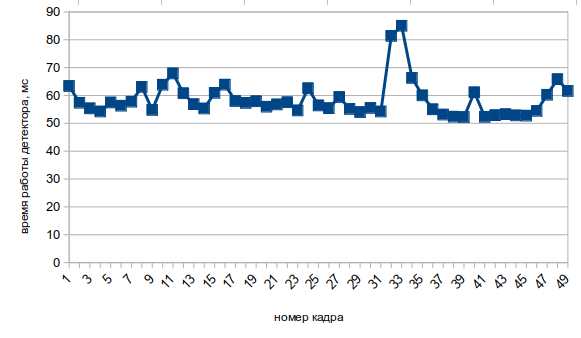


Рис. 12: график зависимости времени работы дететкора Star с последующим запуском алгоритма Лукаса-Канадэ от номера кадра видеопоследовательности icy\_scarp.mp4

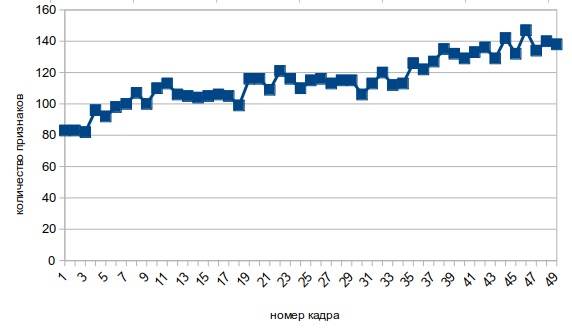


Рис.13: график зависимости количества признаков, найденных детектором Star от номера кадра видеопоследовательности icy\_scarp.mp4

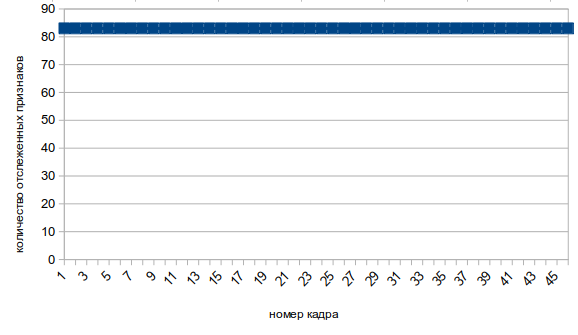


Рис.14: график зависимости количества признаков, обнаруженных дететкором Star и прослеженных с помощью алгоритма Лукаса-Канадэ от первого кадра до текущего, от номера кадра видеопоследовательности icy\_scarp.mp4

Из графиков зависимости количества прослеженных признаков от номерка кадра (Рис. 11 и 14) 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 масштабо-независимыми. Детектор Star, в свою очередь, был разработан специально для задач SLAM и является масштабо-независимым. Особенно хорошо это наблюдается на рис. 11, так как виделпоследовательность stereo.mp4 представляет собой входные данные для типичной задачи SLAM.
2. Количество признаков, отбираемых дететкором 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.

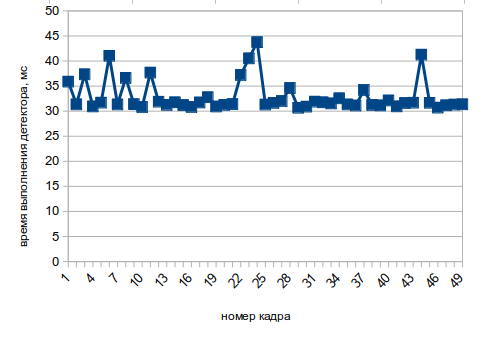


Рис. 15: график зависимости времени работы дететкора FAST с последующим запуском алгоритма Лукаса-Канадэ от номера кадра видеопоследовательности stereo.mp4

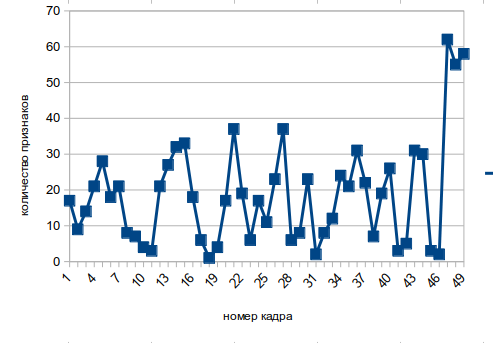


Рис. 16: график зависимости количества признаков, найденных детектором FAST от номера кадра видеопоследовательности stereo.mp4

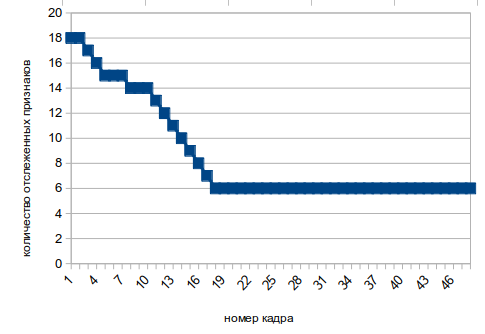


Рис. 17: график зависимости количества признаков, обнаруженных дететкором FAST и прослеженных с помощью алгоритма Лукаса-Канадэ от первого кадра до текущего, от номера кадра видеопоследовательности stereo.mp4

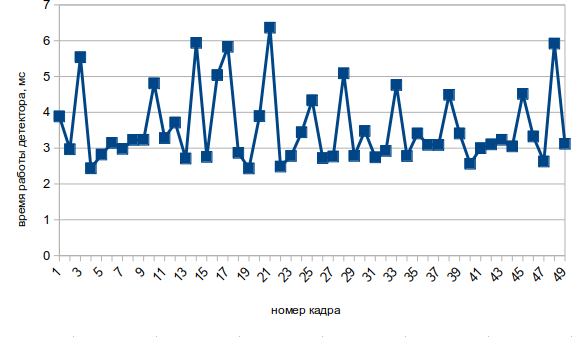


Рис. 18: график зависимости времени работы дететкора FAST с последующим запуском алгоритма Лукаса-Канадэ от номера кадра видеопоследовательности icy\_scarp.mp4

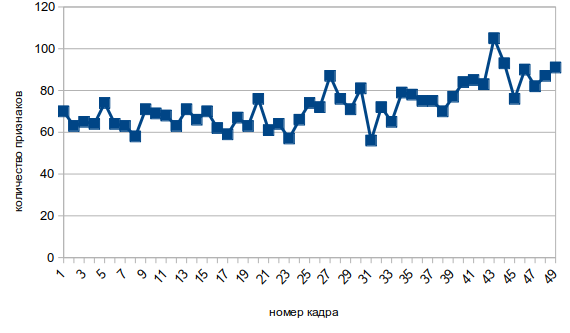


Рис.19: график зависимости количества признаков, найденных детектором FAST от номера кадра видеопоследовательности icy\_scarp.mp4

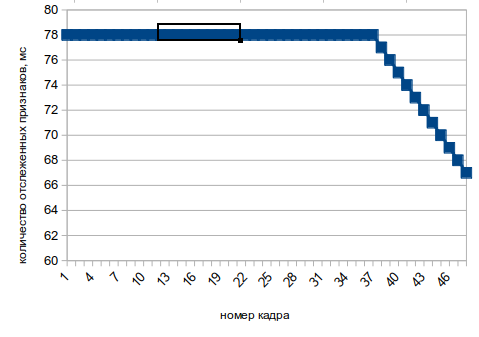


Рис.20: график зависимости количества признаков, обнаруженных дететкором FAST и прослеженных с помощью алгоритма Лукаса-Канадэ от первого кадра до текущего, от номера кадра видеопоследовательности icy\_scarp.mp4

Из графиков зависимости времени работы детектора от номера кадра (Рис. 15 и 18) видно, что детектор FAST удовлетворяет критерию работы в реальном времени. В силу принципа работы (найти n последовательных пикелей вокруг некого центра, которые будут темнее этого центра) детектор FAST хорошо извлекает признаки из границ рельефа, хребтов, и прочих элементов изображения, контрастирующих с фоном:

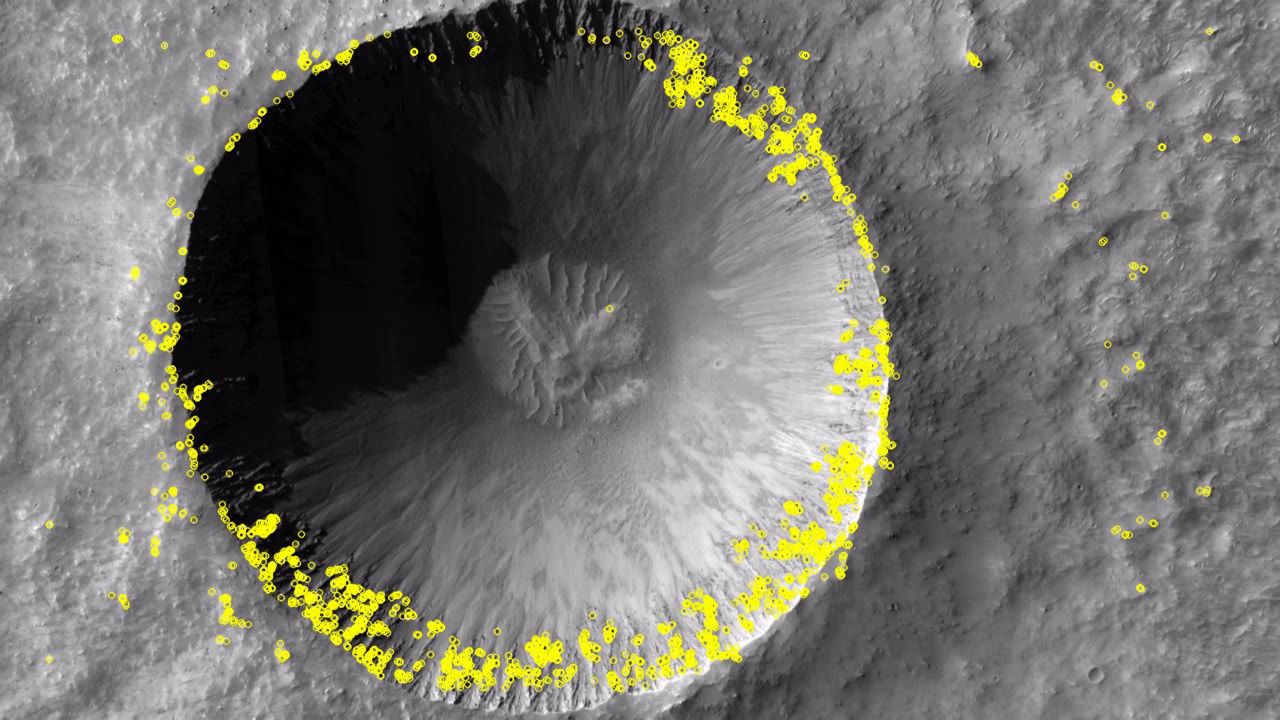


Рис.21: пример результата работы детектора FAST на изображении кратера

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. 20 не является репрезентативным.

Ранее был сделан вывод, что для видеопослеовательностей 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. 2, 6 и 8), слишком мало признаков, что не позволяет использовать их для отслеживания. Поэтому далее следующие алгоритмы:

* дететкор Харриса + BRIEF + FLANN
* дететкор Харриса + FREAK + FLANN
* дететкор Shi-Tomasi + BRIEF + FLANN
* дететкор Shi-Tomasi + FREAK + FLANN

рассмотрены не будут.

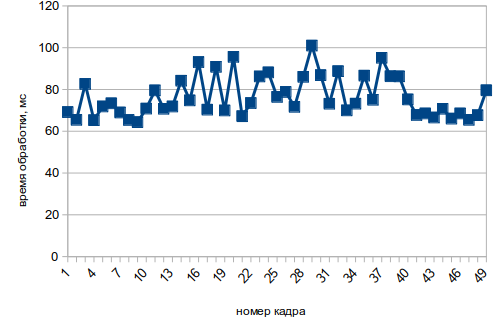


Рис. 22: график зависимости времени работы дететкора BRISK с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности stereo.mp4

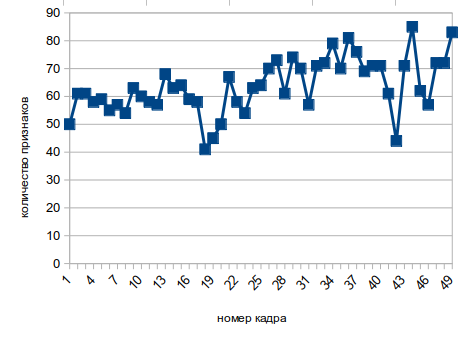


Рис. 23: график зависимости количества признаков, найденных детектором BRISK от номера кадра видеопоследовательности stereo.mp4

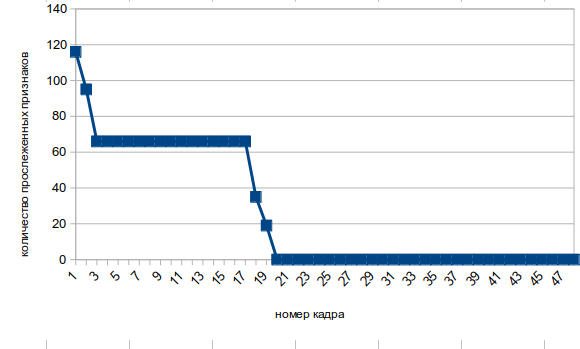
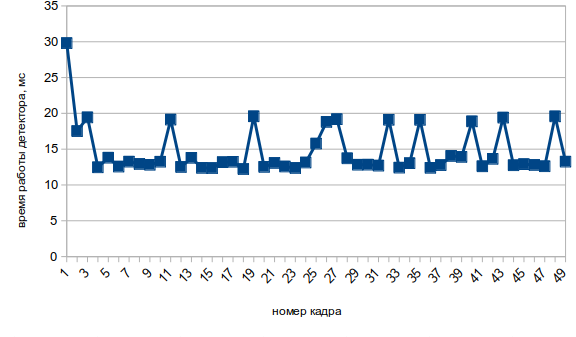


Рис. 24: график зависимости количества признаков, обнаруженных дететкором BRISK и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности stereo.mp4

Рис. 25: график зависимости времени работы дететкора BRISK с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности icy\_scarp.mp4

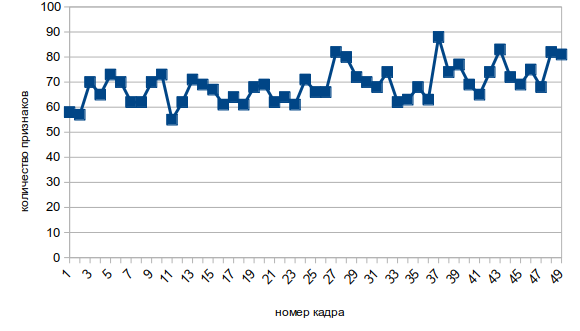


Рис. 26: график зависимости количества признаков, найденных детектором BRISK от номера кадра видеопоследовательности icy\_scarp.mp4

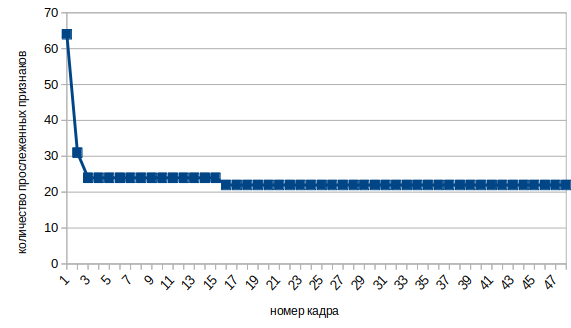


Рис. 27: график зависимости количества признаков, обнаруженных дететкором BRISK и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности icy\_scarp.mp4

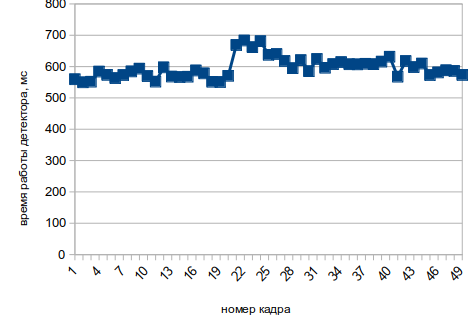


Рис. 28: график зависимости времени работы дететкора SURF на CPU с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности stereo.mp4

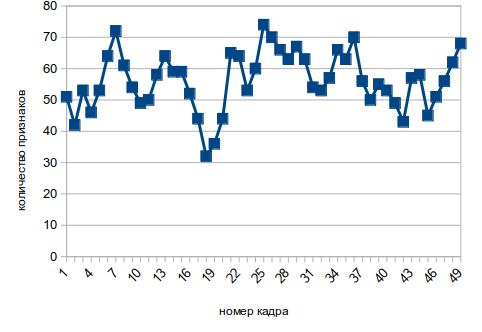


Рис. 29: график зависимости количества признаков, найденных детектором SURF на CPU от номера кадра видеопоследовательности stereo.mp4

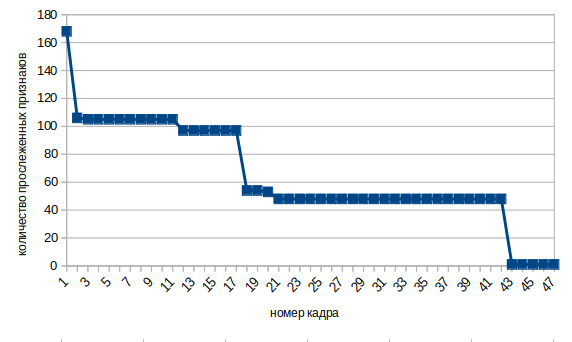


Рис. 30: график зависимости количества признаков, обнаруженных дететкором SURF на CPU и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности stereo.mp4

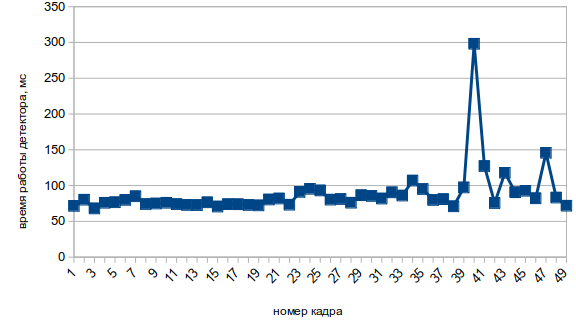


Рис. 31: график зависимости времени работы дететкора SURF на CPU с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности icy\_scarp.mp4

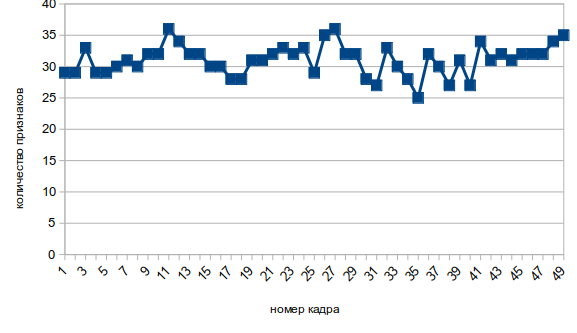


Рис. 32: график зависимости количества признаков, найденных детектором SURF на CPU от номера кадра видеопоследовательности icy\_scarp.mp4

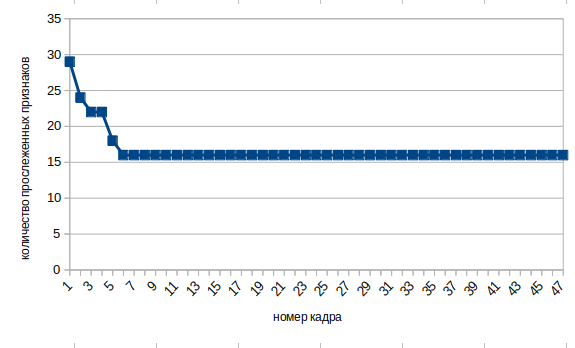


Рис. 33: график зависимости количества признаков, обнаруженных дететкором SURF на CPU и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности icy\_scarp.mp4

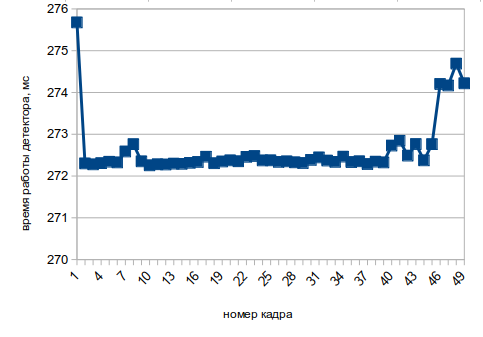


Рис. 34: график зависимости времени работы дететкора SURF на CUDA с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности stereo.mp4

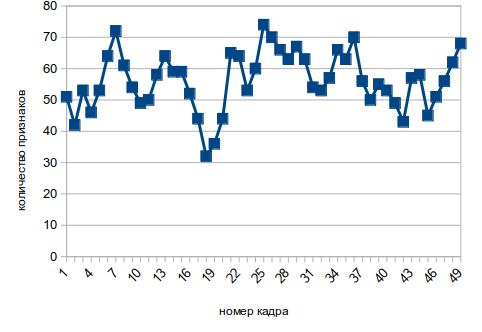


Рис. 35: график зависимости количества признаков, найденных детектором SURF на CPU от номера кадра видеопоследовательности stereo.mp4

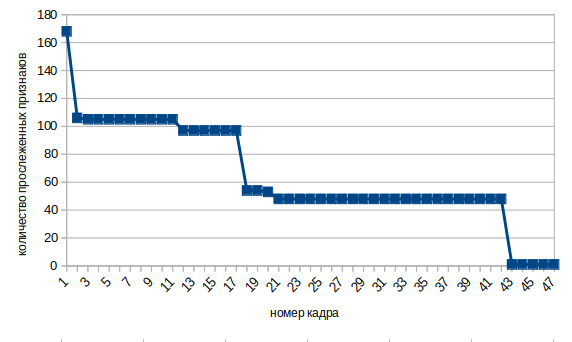


Рис. 36: график зависимости количества признаков, обнаруженных дететкором SURF на CUDA и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности stereo.mp4

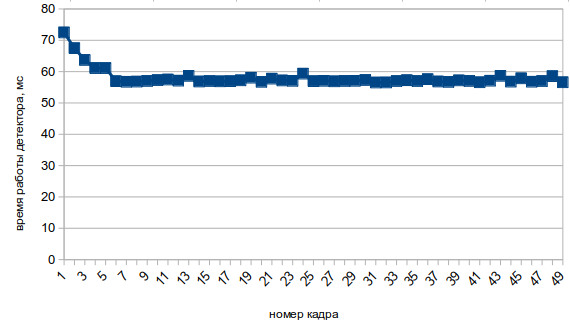


Рис. 37: график зависимости времени работы дететкора SURF на CUDA с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности icy\_scarp.mp4

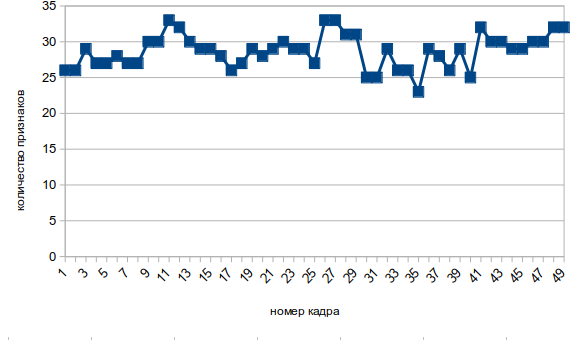


Рис. 38: график зависимости количества признаков, найденных детектором SURF на CUDA от номера кадра видеопоследовательности icy\_scarp.mp4

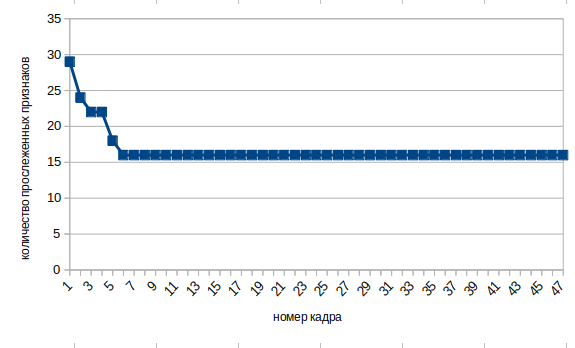


Рис. 39: график зависимости количества признаков, обнаруженных дететкором SURF на CUDA и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности icy\_scarp.mp4

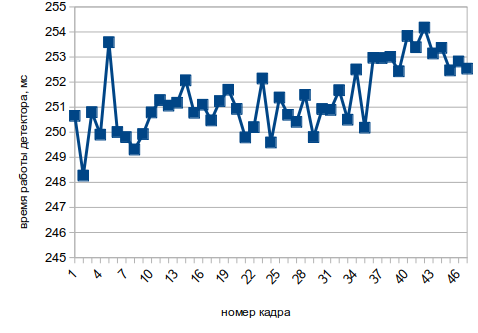


Рис. 40: график зависимости времени работы дететкора SURF на OpenCL с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности stereo.mp4

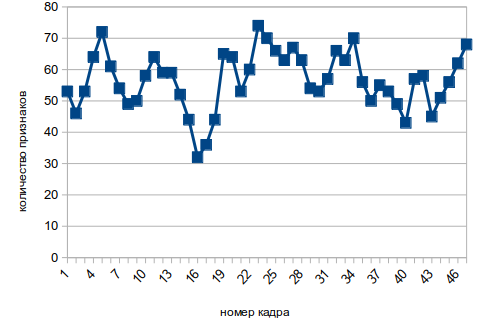


Рис. 41: график зависимости количества признаков, найденных детектором SURF на OpenCL от номера кадра видеопоследовательности stereo.mp4

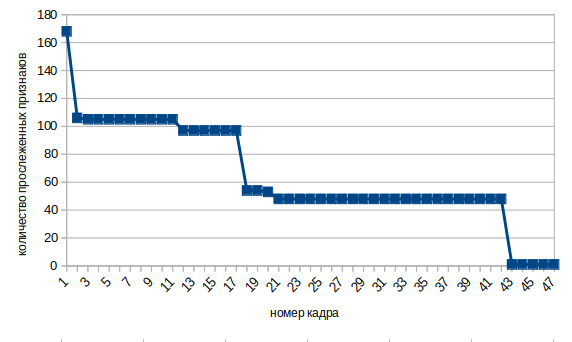


Рис. 42: график зависимости количества признаков, обнаруженных дететкором SURF на OpenCL и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности stereo.mp4

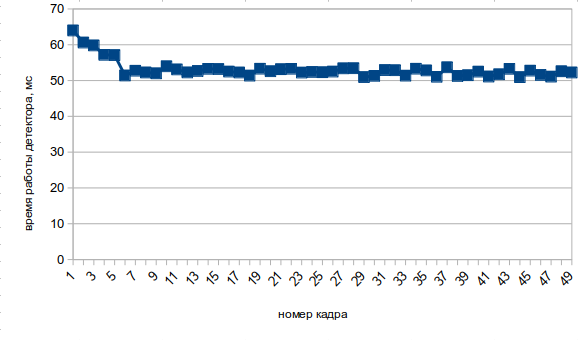


Рис. 43: график зависимости времени работы дететкора SURF на OpenCL с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности icy\_scarp.mp4

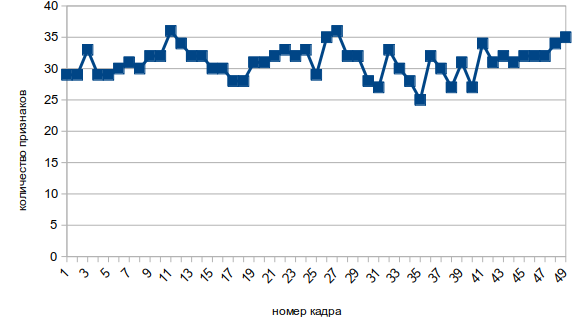


Рис. 44: график зависимости количества признаков, найденных детектором SURF на OpenCL от номера кадра видеопоследовательности icy\_scarp.mp4

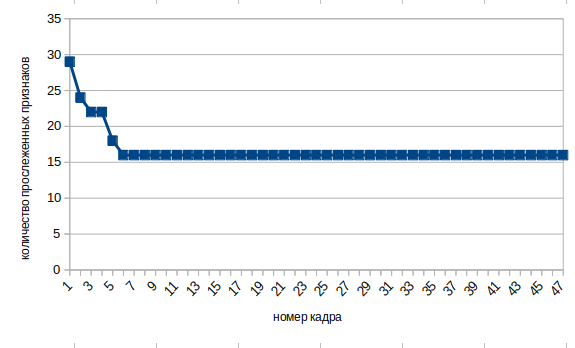


Рис. 45: график зависимости количества признаков, обнаруженных дететкором SURF на OpenCL и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности icy\_scarp.mp4

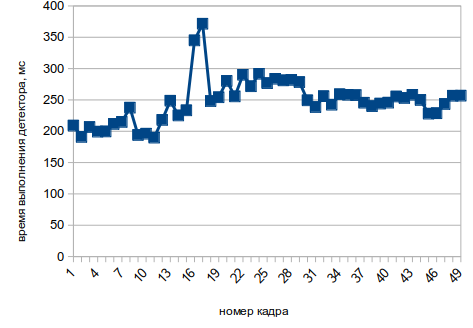


Рис. 46: график зависимости времени работы дететкора ORB на CPU с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности stereo.mp4

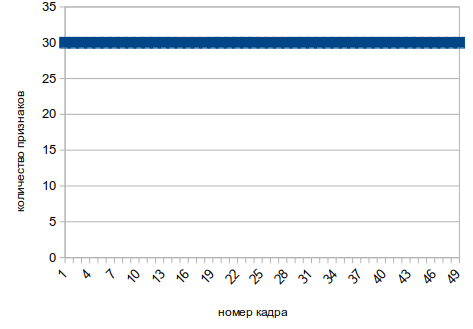


Рис. 47: график зависимости количества признаков, найденных детектором ORB на CPU от номера кадра видеопоследовательности stereo.mp4

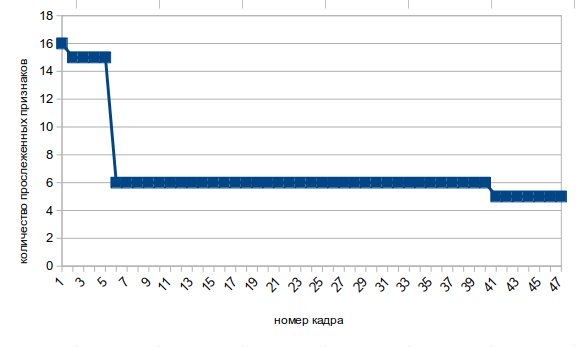


Рис. 48: график зависимости количества признаков, обнаруженных дететкором ORB на CPU и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности stereo.mp4

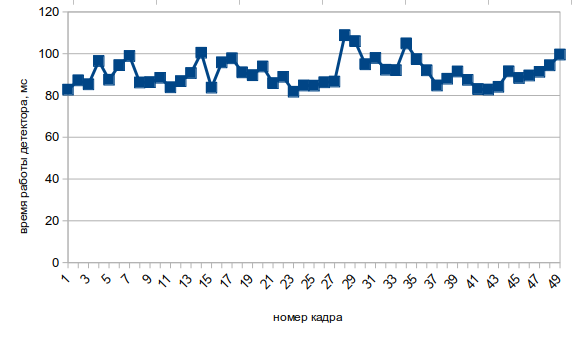


Рис. 49: график зависимости времени работы дететкора ORB на CPU с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности icy\_scarp.mp4

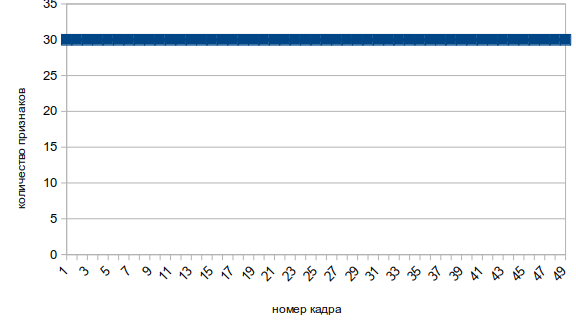


Рис. 50: график зависимости количества признаков, найденных детектором ORB на CPU от номера кадра видеопоследовательности icy\_scarp.mp4

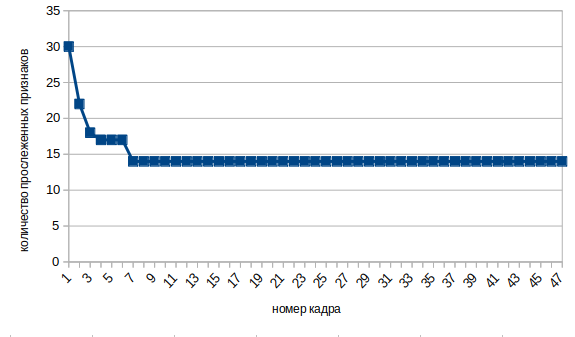


Рис. 51: график зависимости количества признаков, обнаруженных дететкором ORB на CPU и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности icy\_scarp.mp4

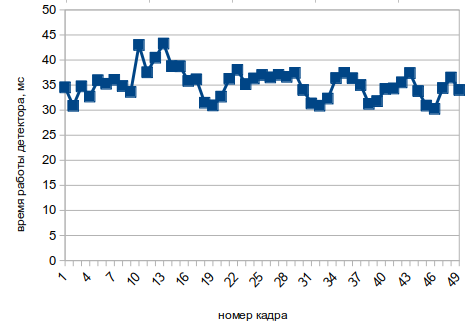


Рис. 52: график зависимости времени работы дететкора ORB на CUDA с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности stereo.mp4

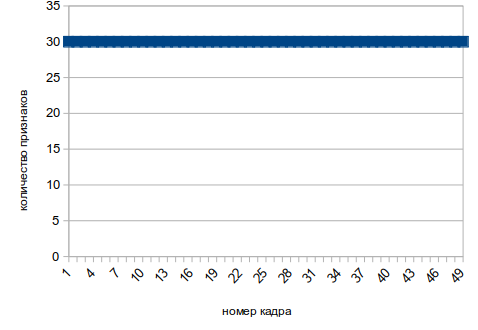


Рис. 53: график зависимости количества признаков, найденных детектором ORB на CUDA от номера кадра видеопоследовательности stereo.mp4

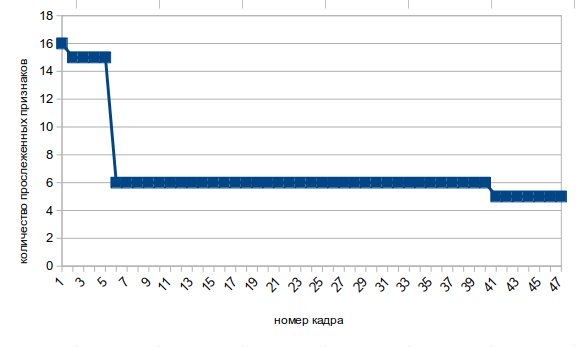


Рис. 54: график зависимости количества признаков, обнаруженных дететкором ORB на CUDA и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности stereo.mp4

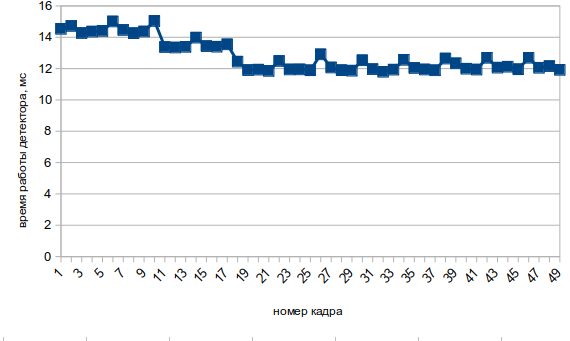


Рис. 55: график зависимости времени работы дететкора ORB на CUDA с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности icy\_scarp.mp4

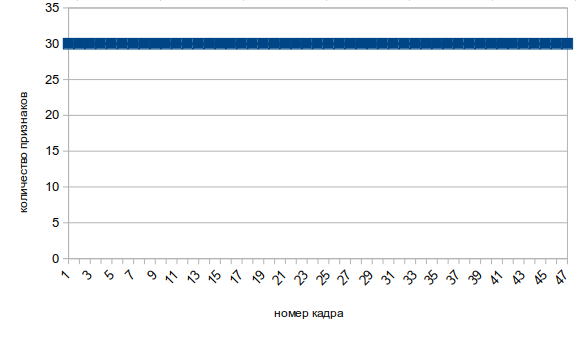


Рис. 56: график зависимости количества признаков, найденных детектором ORB на CUDA от номера кадра видеопоследовательности icy\_scarp.mp4

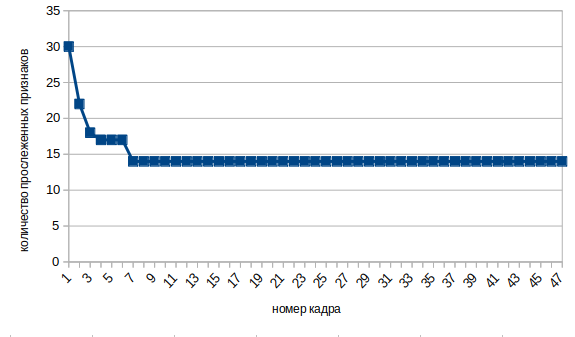


Рис. 57: график зависимости количества признаков, обнаруженных дететкором ORB на CUDA и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности icy\_scarp.mp4

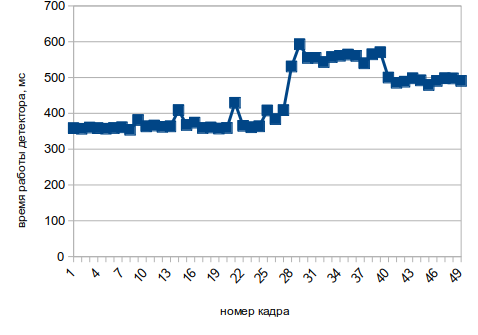


Рис. 58: график зависимости времени работы дететкора ORB на OpenCL с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности stereo.mp4

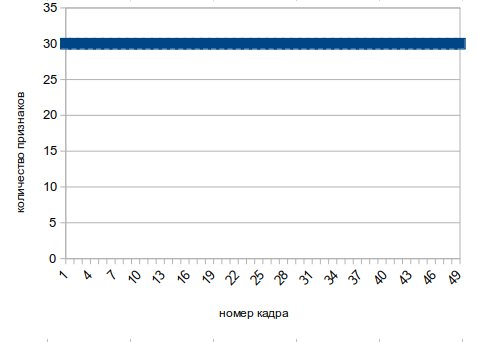


Рис. 59: график зависимости количества признаков, найденных детектором ORB на OpenCL от номера кадра видеопоследовательности stereo.mp4

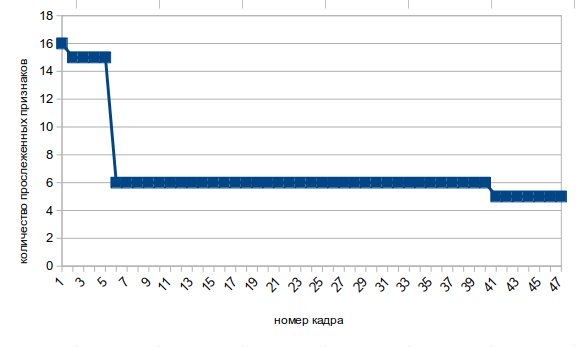


Рис. 60: график зависимости количества признаков, обнаруженных дететкором ORB на OpenCL и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности stereo.mp4

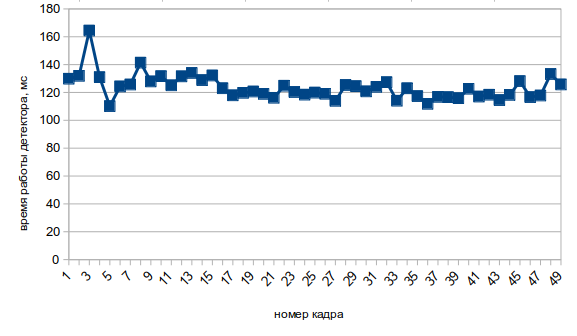


Рис. 61: график зависимости времени работы дететкора ORB на OpenCL с последующим запуском алгоритма FLANN от номера кадра видеопоследовательности icy\_scarp.mp4

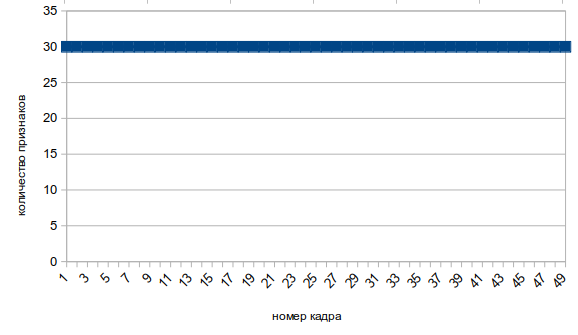


Рис. 62: график зависимости количества признаков, найденных детектором ORB на OpenCL от номера кадра видеопоследовательности icy\_scarp.mp4

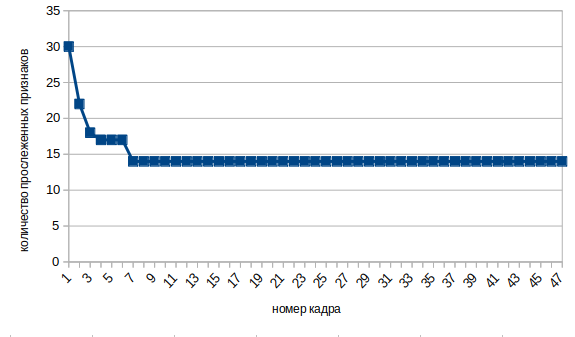


Рис. 63: график зависимости количества признаков, обнаруженных дететкором ORB на OpenCL и прослеженных с помощью алгоритма FLANN от первого кадра до текущего, от номера кадра видеопоследовательности icy\_scarp.mp4

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. 2, 4, 6, 8).
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 они будут работать стабильно для видеопоследовательностей со стабильным 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 милисекунд. 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). Необхходимо 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.

ЗАКЛЮЧЕНИЕ

СПИСОК ЛИТЕРАТУРЫ