

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
GRADUATION THESIS**

Разработка алгоритма синхронизации данных гироскопа и видеокамеры со сканирующим затвором для стабилизации видеоинформации, полученной с БПЛА

Обучающийся / Student Пинчук Владимир Антонович

Факультет/институт/кластер/ Faculty/Institute/Cluster факультет информационных технологий и программирования

Группа/Group М34351

Направление подготовки/ Subject area 01.03.02 Прикладная математика и информатика

Образовательная программа / Educational program Информатика и программирование 2018

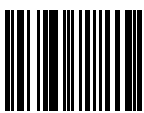
Язык реализации ОП / Language of the educational program Русский

Статус ОП / Status of educational program

Квалификация/ Degree level Бакалавр

Руководитель ВКР/ Thesis supervisor Беляев Евгений Александрович, PhD, науки, Университет ИТМО, факультет информационных технологий и программирования, программист

Обучающийся/Student

Документ подписан	
Пинчук Владимир Антонович	
27.05.2022	

(эл. подпись/ signature)

Пинчук
Владимир
Антонович

(Фамилия И.О./ name
and surname)

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Беляев Евгений Александрович	
27.05.2022	

(эл. подпись/ signature)

Беляев Евгений
Александрович

(Фамилия И.О./ name
and surname)

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ /
OBJECTIVES FOR A GRADUATION THESIS**

Обучающийся / Student Пинчук Владимир Антонович
Факультет/институт/кластер/ Faculty/Institute/Cluster факультет информационных технологий и программирования
Группа/Group М34351
Направление подготовки/ Subject area 01.03.02 Прикладная математика и информатика
Образовательная программа / Educational program Информатика и программирование 2018
Язык реализации ОП / Language of the educational program Русский
Статус ОП / Status of educational program
Квалификация/ Degree level Бакалавр
Тема ВКР/ Thesis topic Разработка алгоритма синхронизации данных гироскопа и видеокамеры со сканирующим затвором для стабилизации видеоинформации, полученной с БПЛА
Руководитель ВКР/ Thesis supervisor Беляев Евгений Александрович, PhD, науки, Университет ИТМО, факультет информационных технологий и программирования, программист

Основные вопросы, подлежащие разработке / Key issues to be analyzed

- (1) Ознакомиться с существующими решениями в области синхронизации инерциальных данных и видеоинформации.
- (2) Разработать и реализовать такой алгоритм, учитывающий наличие у камеры сканирующего затвора.
- (3) Скомбинировать различные подходы для улучшения качества синхронизации.
- (4) Оценить качество полученного решения.

Дата выдачи задания / Assignment issued on: 30.03.2022

Срок представления готовой ВКР / Deadline for final edition of the thesis 15.05.2022

Характеристика темы ВКР / Description of thesis subject (topic)

Тема в области фундаментальных исследований / Subject of fundamental research: нет / not

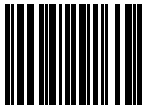
Тема в области прикладных исследований / Subject of applied research: да / yes

СОГЛАСОВАНО / AGREED:

Руководитель ВКР/

Документ	
----------	--


Thesis supervisor

подписан	
Беляев Евгений Александрович	
30.03.2022	

(эл. подпись)

Беляев Евгений
Александрович


Задание принял к
исполнению/ Objectives
assumed BY

Документ подписан	
Пинчук Владимир Антонович	
30.03.2022	

(эл. подпись)

Пинчук
Владимир
Антонович

Руководитель ОП/ Head
of educational program

Документ подписан	
Станкевич Андрей Сергеевич	
01.06.2022	

(эл. подпись)

Станкевич
Андрей
Сергеевич

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**АННОТАЦИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
SUMMARY OF A GRADUATION THESIS**

Обучающийся / Student Пинчук Владимир Антонович
Факультет/институт/кластер/ Faculty/Institute/Cluster факультет информационных технологий и программирования
Группа/Group М34351
Направление подготовки/ Subject area 01.03.02 Прикладная математика и информатика
Образовательная программа / Educational program Информатика и программирование 2018
Язык реализации ОП / Language of the educational program Русский
Статус ОП / Status of educational program
Квалификация/ Degree level Бакалавр
Тема ВКР/ Thesis topic Разработка алгоритма синхронизации данных гироскопа и видеокамеры со сканирующим затвором для стабилизации видеоинформации, полученной с БПЛА
Руководитель ВКР/ Thesis supervisor Беляев Евгений Александрович, PhD, науки, Университет ИТМО, факультет информационных технологий и программирования, программист

**ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
DESCRIPTION OF THE GRADUATION THESIS**

Цель исследования / Research goal

Разработка и реализация алгоритма, решающего задачу синхронизации данных с гироскопа и видеоряда для использования в ПО для стабилизации видео, в том числе с беспилотных летательных аппаратов.

Задачи, решаемые в ВКР / Research tasks

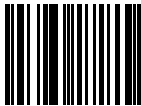
(1) Анализ существующих решений для синхронизации данных с гироскопа и видеоинформации. (2) Разработка и реализация алгоритма синхронизации данных с гироскопа и видео с учётом достоинств и недостатков проанализированных методов. (3) Оценка реализованного алгоритма и сравнение его с ранее существующими решениями по точности синхронизации.

Краткая характеристика полученных результатов / Short summary of results/findings

Разработан алгоритм синхронизации данных с гироскопа и видео, учитывающий как эффекты сканирующего затвора, так и поступательное движение камеры. Проведено сравнение с одним из алгоритмов синхронизации из ПО для стабилизации видео GyroFlow, показано превосходство разработанного алгоритма по точности на тестовых видео.

Обучающийся/Student

Документ	
----------	--

подписан	
Пинчук Владимир Антонович	
27.05.2022	

(эл. подпись/ signature)

Пинчук
Владимир
Антонович

(Фамилия И.О./ name
and surname)

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Беляев Евгений Александрович	
27.05.2022	

(эл. подпись/ signature)

Беляев Евгений
Александрович

(Фамилия И.О./ name
and surname)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	6
1. Обзор существующих подходов к синхронизации видео и данных о движении	8
1.1. Сканирующий затвор	8
1.2. Гироскоп	9
1.3. Модель камеры	9
1.4. Методы, основанные на максимизации кросс-корреляции	10
1.5. Методы, основанные на минимизации ошибки стабилизации	12
1.6. Оценка точности алгоритма синхронизации	13
1.7. Цели и задачи на ВКР	14
Выводы по главе 1	14
2. Разработка и реализация алгоритма синхронизации видеоданных и данных гироскопа	15
2.1. Интеграция угловой скорости и интерполяция кватернионов	15
2.2. Отслеживание точек	16
2.3. Переход от точек на плоскости к лучам	17
2.4. Компенсация вращательного движения	17
2.5. Необходимое условие отсутствия вращательного движения	18
2.6. Первоначальное приближение направления движения	19
2.7. Функция потерь	21
2.8. Оптимизация направления движения для пары кадров	23
2.9. Грубая оценка и оптимизация задержки гироскопа	24
2.10. Алгоритм синхронизации для короткого фрагмента видео	24
2.11. Упрощённая модификация алгоритма синхронизации	25
Выводы по главе 2	26
3. Тестирование алгоритма синхронизации	27
3.1. Описание тестового набора данных	27
3.2. Характер фактической функции задержки гироскопа	29
3.3. Проверка корректности найденных задержек	29
3.4. Оценка качества синхронизации и сравнение с упрощенной версией	30
3.5. Сравнение с одним из методов из GyroFlow	32
Выводы по главе 3	33

ЗАКЛЮЧЕНИЕ	34
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	35

ВВЕДЕНИЕ

Стабилизация изображения — технология, позволяющая компенсировать нежелательные движения изображения из-за случайных перемещений съёмочной камеры. Существует много подходов к стабилизации изображения. Одни используют следящие системы, которые при помощи управления оптической системой камеры сводят к минимуму тряску видимого изображения. Другие используют программную компенсацию движения, искажая уже отснятые изображения в соответствии с известной моделью камеры. Но для того, чтобы компенсировать нежелательные движения в видео, необходимо иметь информацию об истории этих движений.

Для получения информации об угловых перемещениях можно использовать МЭМС-гироскоп — электромеханический прибор для измерения угловой скорости. Но часто так получается, что часы гироскопа и часы, от которых тактируется камера никак не связаны. Первая из возможных причин этому — то, что система записи угловых перемещений независима от камеры и не связана с ней напрямую. Ещё одной причиной может быть то, что хотя камера и имеет встроенную возможность записывать угловые перемещения, ПО камеры построено на основе проприетарного SDK, который не имеет достаточной гибкости для реализации полноценной записи данных о движении синхронно с видео. Получается два “потока” данных с неизвестной (но ограниченной) задержкой между собой, которая, ко всему прочему, может и изменяться во времени из-за погрешностей часов и их дрейфа. Таким образом, возникает задача находить эту задержку в каждый момент времени на протяжении всего видео — иными словами синхронизировать изображение и данные о движении.

Задача синхронизации усложняется ещё тем, что большинство современных видеокамер имеет сканирующий затвор. Это значит, что изображение, получаемое с камеры строго говоря не является просто перспективной проекцией мира на плоскость в какой-то определённый момент времени. Некоторые камеры имеют настолько медленный затвор, что задержка между съёмкой верхней и нижней части кадра превышает даже половину периода между кадрами.

Данная работа посвящена разработке и реализации алгоритма, решающего задачу синхронизации данных с гироскопа и видеоряда, для использо-

вания в ПО для стабилизации видео, в том числе с беспилотных летательных аппаратов.

Актуальность данной работы объясняется тем, что существующие реализации алгоритмов, решающих задачу синхронизации видеокамеры со сканирующим затвором и гироскопа в основном являются либо коммерческой тайной, либо прекратившими своё развитие проектами.

Данная работа организована следующим образом. В первой главе представлены основные понятия, анализ существующих решений, постановка целей и задач. Во второй главе описаны подзадачи, которые возникли в ходе разработки и реализации алгоритма и варианты их решения. В третьей главе представлены результаты тестирования и сравнение двух реализованных модификаций алгоритма. В заключении подведены итоги проделанной работы, сделаны выводы.

ГЛАВА 1. ОБЗОР СУЩЕСТВУЮЩИХ ПОДХОДОВ К СИНХРОНИЗАЦИИ ВИДЕО И ДАННЫХ О ДВИЖЕНИИ

В этой главе вводятся основные понятия, необходимые для постановки задачи, проводится анализ существующих решений и выделяются их основные недостатки. Кроме того, предлагается метод оценки точности алгоритма синхронизации и ставятся задачи на ВКР.

1.1. Сканирующий затвор

Сканирующий затвор — схема считывания данных с фотоматрицы, основанная на построочном сканировании. По сути является электронным аналогом фокального затвора, широко применяемого в фототехнике. У фокального затвора имеется две шторки, сначала движение начинает первая — открывает затвор, потом — вторая, с некоторым запаздыванием, между ними образуется щель, ширина которой и задаёт выдержку. В случае сканирующего затвора, экспозиция происходит так же, но механических шторок нет.

Основные параметры, характеризующие сканирующий затвор фотоматрицы — **общая длительность считывания** (англ: total readout time) t_r — время между началом считывания первой и последней строки матрицы, его можно узнать из документации производителя матрицы, либо экспериментально, **период кадров** t_{if} — обратная величина к частоте кадров и **количество строк в кадре** — n_r . На рисунке 1 представлен процесс работы сканирующего затвора.

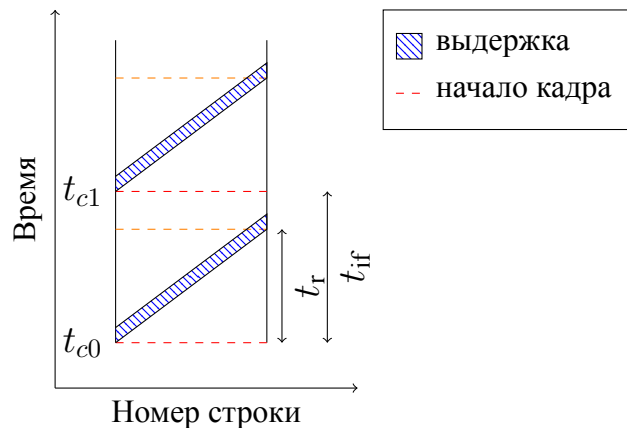


Рисунок 1 – Процесс работы сканирующего затвора

При небольших выдержках, когда не происходит значительного размытия движения, длительностью выдержки можно пренебречь. В этой работе это будет по умолчанию подразумеваться, если не сказано иначе, так как электронная стабилизация всё равно требует короткой выдержки для получения

хорошего результата. Введем обозначение $t_c(X)$ — момент времени по часам камеры, относящийся к какому-то объекту. Например, таким объектом может быть f_i — i -й кадр видео. В таком случае $t_c(f_i)$ — момент начала выдержки первой строки кадра. Таким образом, момент фиксирования камерой строки j на кадре i будет описываться следующей формулой:

$$t_{i,j} = t_c(f_i) + t_r \frac{j}{n_r}. \quad (1)$$

1.2. Гироскоп

Трехосевой гироскоп выдаёт измерения в виде векторов мгновенной угловой скорости $w_i \in R^3$, заданных в системе координат, с ним связанной с некоторой определённой **частотой дискретизации** f_s , известной с некоторой погрешностью. По аналогии с $t_c(X)$, введем обозначение $t_g(X)$ — отметка времени события X по часам гироскопа. Также введем понятие **задержки гироскопа** — функцию d , которая показывает, насколько сейчас отстают часы гироскопа от основных часов, т.е. часов камеры:

$$t_g(X) = t_c(X) + d(t_c(X)). \quad (2)$$

Поскольку кроме постоянной погрешности часов имеют место и другие эффекты, например температурная нестабильность, введем допущение, что для функции d выполняются следующие условия:

$$\begin{aligned} d_{min} &\leq d(t) \leq d_{max}, \\ d_{max} - d_{min} &\leq D, \\ |d'(t)| &\leq R, \end{aligned} \quad (3)$$

где D, R — некоторые постоянные.

1.3. Модель камеры

Видеокамера отображает некоторые точки пространства X_j в точки $p_i(X_j)$ на плоскости изображения f_i . Отображение, которое ставит в соответствие точкам пространства из поля зрения камеры точки на плоскости изображения называют **моделью камеры**.

Одна из распространённых в предметной области моделей камеры — модель Fisheye [1]. Она с достаточной точностью описывает целевые камеры, более того, для большинства камер параметры этой модели уже известны.

Пусть имеется точка $(x, y, z) \in R^3$, в системе координат, связанной с камерой (см. рисунок 2). Тогда, изображение точки описывается следующими уравнениями:

$$\begin{aligned} a &= \frac{x}{z}, \quad b = \frac{y}{z}, \\ r^2 &= a^2 + b^2, \\ \theta &= \arctan(r), \\ \theta_d &= \theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^7 + k_4\theta^9, \\ x' &= a \left(\frac{\theta_d}{r} \right) \quad y' = b \left(\frac{\theta_d}{r} \right), \\ u &= f_x x' + c_x \quad v = f_y y' + c_y. \end{aligned} \tag{4}$$

где пара (u, v) — координаты точки на изображении в пикселях, (f_x, f_y) называется фокусным расстоянием, (c_x, c_y) — главной точкой, k_1, k_2, k_3, k_4 — заранее известные параметры объектива.

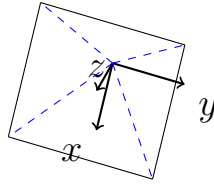


Рисунок 2 – Система координат, связанная с камерой

Координаты (u, v) проекции точки (x, y, z) на изображение не зависят от модуля вектора (x, y, z) . Таким образом, зная (u, v) , с помощью этой модели можно найти луч, входящий в камеру, проходящий через точку (x, y, z) , заданный в системе координат, связанной с камерой. Для этого, в том числе, необходимо искать корень многочлена девятой степени, это можно сделать при помощи метода Ньютона.

Стоит отметить, что реальная камера, в отличие от модели имеет сканирующий затвор, поэтому на практике эту модель нужно применять локально, по отдельности для каждой строки.

1.4. Методы, основанные на максимизации кросс-корреляции

Несмотря на наличие у большинства целевых камер сканирующего затвора, некоторые практические реализации электронной стабилизации с ис-

пользованием гироскопа просто пренебрегают этим фактом и предполагают, что весь кадр снимается мгновенно, например, это open-source проекты GyroFlow до версии 1.0.0 [2] и VirtualGimbal [3].



Рисунок 3 – Общая схема работы метода на основе кросс-корреляции

Высокоуровневая схема работы таких методов представлена на рисунке 3. Первый этап — тот или иной алгоритм отслеживания особенностей изображения (feature tracking), который применяют к парам последовательно идущих кадров. В полученных соответствиях устраняют искажения объектива по заранее известной модели камеры. Далее, для каждой пары последовательно идущих кадров находят существенную матрицу, например, как описано в [4, с. 279], из неё получают угловые перемещения между кадрами. По найденным угловым перемещениям оценивают скорости, предполагая постоянство угловой скорости на отрезках времени между кадрами. Далее, для достаточно короткого отрезка видео (единицы секунд), считается кросс-корреляция угловых скоростей, оцененных по видео и угловых скоростей, измеренных гироскопом, и проводится поиск максимума. Искомая функция задержки гироскопа полагается на рассматриваемом фрагменте видео равной задержке, при которой достигается максимум кросс-корреляции.

Поскольку задержка гироскопа изменяется относительно медленно, можно применить алгоритм на нескольких отрезках видео и проинтерполировать полученные значения в промежутках, не покрытых отрезками.

К сожалению, на камерах со сканирующим затвором, такие методы часто дают ошибки порядка t_r (общей длительности считывания). Продемонстрируем одну из причин этому на примере. Рассмотрим видео, в котором большая часть отслеживаемых точек находится ближе к верхней части кадра и другое видео, где эти точки в основном в нижней части кадра. Так как во втором случае особенности в среднем сфотографированы позже, получится так, что алгоритм будет отслеживать движение, которое происходило позже, чем в первом и выдаст другое решение.

1.5. Методы, основанные на минимизации ошибки стабилизации

К этому классу можно отнести методы [5–7]. Примерами практических реализаций может служить метод “Using visual features” в GyroFlow старше 1.0.0 версии [8] и метод использующийся в Crisp [9].

Здесь, так же как и в предыдущем классе методов, на первом этапе находят соответствия между парами точек на последовательно идущих кадрах, и устраняют искажения объектива. Затем, определяют функцию потерь как некую оценку качества стабилизации изображения при фиксированной задержке гироскопа t . Далее задача сводится к поиску минимума функции потерь. Некоторые реализации ищут этот минимум полным перебором, другие — добиваются непрерывной дифференцируемости и применяют методы оптимизации.

Независимо от метода поиска минимума, важнейшую роль играет сама функция потерь. В литературе встречаются два основных подхода к её построению. Первый подход [6], заключается в том, чтобы основываясь на теории стереозрения вывести функцию, минимизация которой будет вести к минимизации вращательного движения в стабилизированном видео. Вторым подход [5, 7], основан на идее о том, что поступательным движением камеры при построении функции потерь можно пренебречь, тогда минимизировать вращательное движение в стабилизированном видео становится значительно проще. В случае использования второго подхода, возможна деградация точности выдаваемых оценок задержки при существенном поступательном движении в видео.

Также важно заметить, что в предыдущем классе алгоритмов имелся этап нахождения существенной матрицы, на котором эффективно отсеивались выбросы с предыдущих этапов. В этом же классе, этот этап отсутствует, поэтому для фильтрации выбросов приходится применять дополнительные ухищрения. Например, в GyroFlow просто отбрасывают какую-то долю соответствий вносящих наибольший вклад в функцию потерь, а в Crisp используется такая функция потерь, которая ослабляет влияние выбросов.

1.6. Оценка точности алгоритма синхронизации

Для оценки результата работы алгоритма — функции задержки гироскопа, необходим набор данных, для которого можно сделать какие-то предположения о фактической задержке гироскопа. Предлагаемый в этой работе алгоритм будет работать на достаточно коротких отрезках видео, порядка 2 сек. Имея более длинную видеозапись, её можно разрезать на большое количество маленьких. Так как известно, что функция задержки гироскопа изменяется не быстро, хорошо работающий алгоритм должен давать близкие оценки задержки гироскопа для соседних фрагментов.

Более того, если известна дополнительная информация, что камера и гироскоп имеют достаточно стабильные часы, можно считать, что функция задержки гироскопа линейна.

В этой работе, один из используемых методов оценки будет основан именно на линейности функции задержки. На основе десятков запусков алгоритма на разных отрезках при помощи линейной регрессии будет строиться функция задержки гироскопа. Она будет считаться за фактическую, от разности результатов работы алгоритма с ней можно будет посчитать среднеквадратичное отклонение.

Второй метод, для того, чтобы убедиться в том, что найденные значения задержки действительно близки к реальным, можно воспользоваться инструментом для стабилизации видео GyroFlow [8]. Он, в том числе, позволяет вводить задержки в нескольких точках вручную смотреть результат стабилизации в реальном времени. На некоторых тестовых видео даже ошибки порядка 1-2 мс заметно портят результат стабилизации.

1.7. Цели и задачи на ВКР

Целью работы стала разработка и реализация алгоритма, решающего задачу синхронизации данных с гироскопа и видеоряда для использования в ПО для стабилизации видео, в том числе с беспилотных летательных аппаратов. Для достижения этой цели были поставлены следующие задачи:

- Составить план реализации алгоритма, реализовать необходимые вспомогательные части, в том числе загрузку исходных данных, отслеживание точек, модель камеры и т.д.
- Составить функцию потерь, не пренебрегающую поступательным движением камеры и реализовать на основе неё алгоритм синхронизации.
- Проверить работу реализованного алгоритма на разнообразных видео, в том числе снятых с БПЛА.
- Заменить функцию потерь на такую, которая бы пренебрегала поступательным движением, сравнить результаты.

Выводы по главе 1

Были введены основные понятия из предметной области, используемые в дальнейшем в работе, был проведён анализ существующих решений подобных задач, выделены их достоинства и недостатка. Также, предложены способы оценки качества выдаваемых алгоритмом синхронизации решений и поставлены задачи на ВКР.

ГЛАВА 2. РАЗРАБОТКА И РЕАЛИЗАЦИЯ АЛГОРИТМА СИНХРОНИЗАЦИИ ВИДЕОДАНЫХ И ДАННЫХ ГИРОСКОПА

По результатам анализа, проведённого в предыдущей главе, были приняты следующие решения:

- Разрабатывать алгоритм на основе подхода, минимизирующего ошибку стабилизации, поскольку подход, максимизирующий кросс-корреляцию не даёт необходимой точности синхронизации.
- Не пренебрегать поступательным движением в функции потерь, так как в предметной области встречаются видео с существенным поступательным движением и практически незаметным вращательным.
- Учитывать, что не все строки в кадре снимаются одновременно.

В этой главе описаны этапы разработки и реализации предлагаемого алгоритма.

2.1. Интеграция угловой скорости и интерполяция кватернионов

Интеграция данных гироскопа — процесс восстановления функции ориентации в пространстве от времени по известным угловым скоростям. Поскольку измерения угловой скорости имеют дискретный характер, вычислять ориентации в пространстве можно по аналогии с методом конечных сумм. Композиция кватернионов, описывающая вращение камеры на отрезке времени от $t_g(\omega_0)$ до $t_g(\omega_i)$, где ω_i — i -е измерение угловой скорости, записывается как:

$$q_i = \prod_{j=i}^0 \exp\left(\frac{\omega_j}{2f_s}\right). \quad (5)$$

Теперь можно ввести функцию ориентации камеры в системе координат, связанной с положением камеры в начальный момент времени q_{cam} , определённую на дискретном множестве, такую, что:

$$q_{cam}(t_g(\omega_i)) = q_i. \quad (6)$$

Такое представление данных о вращении чрезвычайно удобно. Например, вращение, которое происходило на протяжении времени от i -го до j -го измерения угловой скорости можно записать, как

$$q_{i...j} = q_{cam}(t_g(\omega_i))^{-1} q_{cam}(t_g(\omega_j)).$$

Функция ориентации от времени, заданная формулой (6) и определенная на дискретном множестве, описывает некоторый непрерывный процесс. Поэтому имеет смысл доопределить её до непрерывной. С учётом специфики дальнейшего её использования, необходимо, чтобы она была дважды непрерывно дифференцируемой. Есть несколько способов удовлетворить эти требования [10], в этой работе предлагается использовать интерполяцию RQBez в силу простоты её реализации.

Кватернионы рассматриваются как обычные вектора в \mathbb{R}^4 и интерполируются покомпонентно при помощи кривых Безье. Для того чтобы построить кривую Безье так, чтобы она была дважды непрерывно дифференцируемой, необходимо решить разреженную СЛАУ из n_g уравнений с n_g неизвестными, где n_g — количество интерполируемых точек [11]. Однако, за счёт особенностей её структуры, она решается за линейное время. Поскольку покомпонентная интерполяция кривыми Безье не гарантирует сохранения единичной нормы кватернионов, в RQBez после вычисления значения в точке, проводится ренормализация.

2.2. Отслеживание точек

Отслеживание точек можно реализовать различными способами. В экспериментальных версиях реализации для этого применялся KLT tracker [12]. Как оказалось впоследствии, такое решение показывает плохие результаты при наличии высококонтрастного переднего плана и низкоконтрастного фона — большинство точек для отслеживания выделяется на переднем плане. Это хорошо заметно на некоторых кадрах из тестовых видео, как на рисунке 4.

Поэтому, в работе предлагается применять для этой задачи один из самых быстрых алгоритмов для нахождения плотного оптического потока — DIS optical flow [13]. Он даёт достаточно качественный оптический поток даже в низкоконтрастных областях. Но так как плотный оптический поток не требуется для предлагаемого алгоритма, будем выбирать из него только несколько сотен векторов движения, равномерно распределённых по всей плоскости изображения.

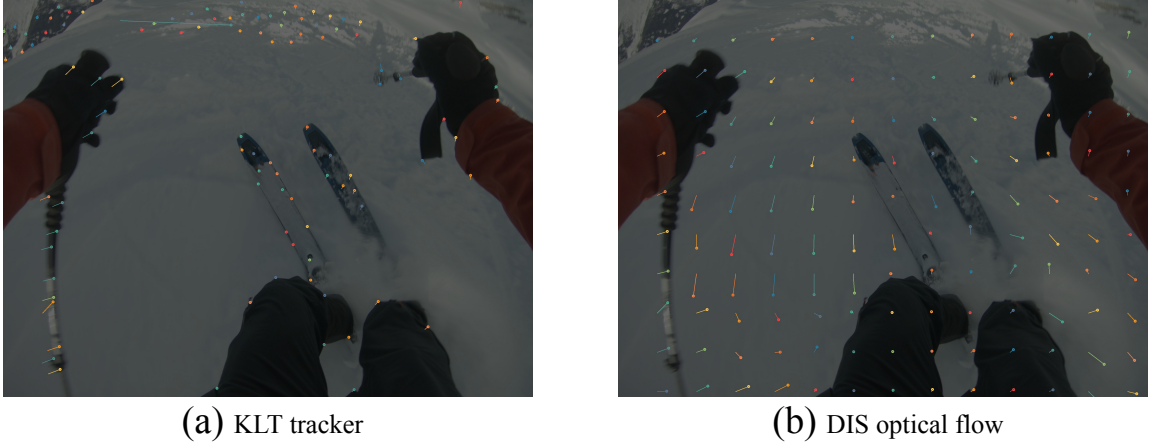


Рисунок 4 – Два способа отслеживания точек

2.3. Переход от точек на плоскости к лучам

При отслеживании точек на кадрах, точки описывались координатами пикселей на кадрах $p_i(X_j)$. Такое представление не слишком удобно и к тому же зависит от внутренних параметров камеры.

Идея в том, чтобы полностью абстрагироваться от пикселей и изображений и оперировать только с направлениями на некоторые точки в пространстве, или, иначе говоря, с лучами. Как было отмечено в главе 1.3, Fisheye модель позволяет это сделать. Обозначим вектор, задающий луч, соответствующий точке пространства X_j , запечатлённой на кадре f_i как $r_i(X_j)$. Для определённости будем считать, что $\|r_i(X_j)\| = 1$.

Поскольку при переходе к лучам теряется информация о номере строки изображения, в которую попала точка пространства X_j на кадре f_i , теряется и возможность вычислить момент съёмки этой точки на кадре f_i . Будем обозначать эти моменты времени, как $t_c(p_i(X_j))$, их имеет смысл посчитать до перехода к лучам.

2.4. Компенсация вращательного движения

Пусть имеются описание пары кадров, а именно $q_{cam}(t)$, $d(t)$ и $\{r_i(X_j)\}_{(i,j) \in S}$, где $S = \cup_{i=i_0..i_0+1} \{i, j | j \in \mathcal{F}(f_i)\}$, $\mathcal{F}(f_i)$ — множество точек пространства, видимых в кадре f_i . Необходимо преобразовать лучи так как если бы вращения камеры не было. Причём преобразовать необходимо не только лучи с кадра f_{i_0+1} , но и с кадра f_{i_0} , так как, из-за сканирующего затвора, вращение влияет и на них.

Для каждой отслеживаемой точки X_j можно проделать следующее. Можно узнать ориентации камеры в соответствующие событиям моменты вре-

мени исходя из данных гироскопа:

$$\begin{aligned} q_0 &= q_{cam}(t_g(f_i)), \\ q_{i_0} &= q_{cam}(t_g(p_{i_0}(X_j))), \\ q_{i_0+1} &= q_{cam}(t_g(p_{i_0+1}(X_j))). \end{aligned} \quad (7)$$

Идея в том, чтобы повернуть лучи так, как будто ориентация камеры не менялась с момента начала съёмки кадра f_{i_0} . Кватернионы $q_a^{-1}q_0$ и $q_b^{-1}q_0$ как раз и описывают необходимые вращения. Применим эти вращения к лучам:

$$\begin{aligned} r'_{i_0}(P_j) &= \text{rotate}(r'_{i_0}(X_j), q_{i_0}^{-1}q_0), \\ r'_{i_0+1}(P_j) &= \text{rotate}(r'_{i_0+1}(X_j), q_{i_0+1}^{-1}q_0). \end{aligned} \quad (8)$$

2.5. Необходимое условие отсутствия вращательного движения

Лемма 1. Пусть в \mathbb{R}^3 имеется две точки с координатами \mathbf{C}_1 и \mathbf{C}_2 , причём $\mathbf{C}_2 = \mathbf{C}_1 + e$ и точка \mathbf{X} . Тогда для векторов $p = \frac{\overrightarrow{C_1X}}{\|\overrightarrow{C_1X}\|}$, $p' = \frac{\overrightarrow{C_2X}}{\|\overrightarrow{C_2X}\|}$ и $v = \frac{e}{\|e\|}$ верно, что $v \cdot (p_1 \times p') = 0$

Доказательство. Векторы $e, \overrightarrow{C_1X}, \overrightarrow{C_2X}$ компланарны, значит v, p, p' также компланарны, значит $v \perp (p \times p')$, следовательно, $v \cdot (p \times p') = 0$. ■

Рассмотрим две камеры, ориентированные в одном направлении с центрами C_1 и C_2 , такими, что $\mathbf{C}_2 = \mathbf{C}_1 + e$ и точку \mathbf{X} . Известны вектора p и p' — направления на точку \mathbf{X} , заданные в системах координат, связанных с первой и второй камерой соответственно. Так как вращения между камерами нет, а p и p' — свободные вектора, то можно считать, что все векторы заданы в системе координат первой камеры. По лемме 1, существует такой вектор $v = \frac{e}{\|e\|}$, что

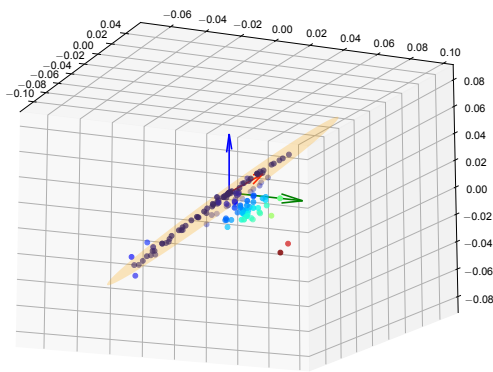
$$v \cdot (p_1 \times p') = 0. \quad (9)$$

Если же есть пара камер, вращение между которыми задаётся некоторой нетривиальной матрицей поворота R , для них также известны направления на точку \mathbf{X} в системах координат, связанных с ними: p и p' . Тогда, можно “компенсировать” вращение второй камеры — перевести вектор p' в систему координат, связанную с первой камерой: $p'_1 = R^{-1}p'$. Необходимое условие отсутствия вращения (9), в котором заменён вектор p' на p'_1 будет выполняться: $v \cdot (p \times R^{-1}p') = 0$. Так как данное условие не является достаточным, то

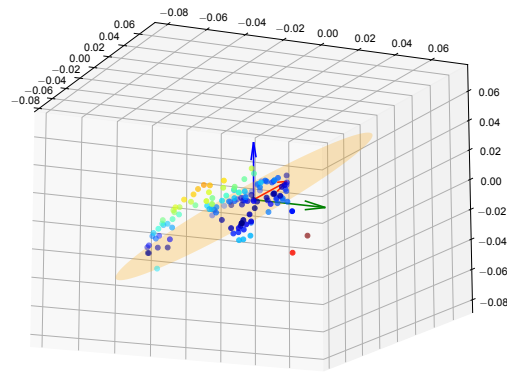
найдутся случаи, когда верно и $v \cdot (p \times p') = 0$, например, если p' окажется собственным вектором R^{-1} . Однако, в предлагаемом в работе применении, условие (9) используется многократно для одной и той же пары кадров, но различных точек пространства и случаи, когда условие выполнено для всех точек, но вращение есть маловероятны.

Если камеры имеют сканирующий затвор, то компенсацию вращения нужно делать несколько иначе, в соответствии с (8).

Отметим ещё одно важное свойство условия (9) — непрерывная дифференцируемость. На практике, это значит, что найдя некоторое приближение v можно итеративно его уточнять при помощи методов оптимизации.



(a) Вращение скомпенсировано



(b) Вращение не скомпенсировано

Рисунок 5 – Демонстрация достаточного условия отсутствия вращения на реальных данных

На рисунке 5 представлена визуализация векторов $p_i \times p'_i$, вычисленных для одного и того же набора точек \mathbf{X} , наблюдаемых из одной и той же пары камер. Для левой части рисунка была применена компенсация вращения по гироскопу, для правой — нет. Можно заметить, что на левом рисунке большинство векторов компланарны, лежат в плоскости с нормалью v , но есть и выбросы. На правом рисунке, где есть существенное вращение между изображениями, уже практически нет такой плоскости.

2.6. Первоначальное приближение направления движения

В главе 2.5 было сформулировано необходимое условие отсутствия вращения камеры. Здесь будет предложен способ найти приближение вектора v ,

имея только лучи, соответствующие некоторым видимым точкам для пары кадров, в предположении, что вращательного движения нет.

Самый простой подход — решить СЛОУ (систему линейных однородных уравнений) $Pv = \vec{0}$, где P — матрица, составленная из векторов-строк $p_i \times p'_i$, например, методом сингулярного разложения. После проверки на реальных данных, выяснилось, что такой подход недостаточно устойчив к выбросам. Пример реальных данных был представлен ранее на рисунке 5а.

Поэтому, был разработан следующий рандомизированный алгоритм. Пусть $g_i = p_i \times p'_i$ для всех $i \in S$, где S — множество индексов видимых точек на этой паре кадров. На каждой итерации строится гипотеза о направлении движения камеры: выбираются какие-то два вектора из множества векторов $g = \{g_i | i \in S\}$. В предположении, что эти два вектора не являются выбросами, вычисляется нормаль n к плоскости, в которой должны лежать все остальные векторы g_i . Гипотеза оценивается с помощью функции потерь $\mathcal{L}(n) = \text{med}\{(n \cdot g_i)^2 | i \in S\}$. После некоторого фиксированного числа итераций l , выбирается такая нормаль n , для которой значение $\mathcal{L}(n)$ будет наименьшим. Псевдокод алгоритма приведён в листинге 1.

Алгоритм 1 – Рандомизированный алгоритм для определения направления движения

```

function GuessTranslationalMotion( $g$ )
     $M_{least} \leftarrow \infty$ 
    repeat  $l$  times
         $u, v \leftarrow \text{random\_choice}(g, 2)$ 
         $n \leftarrow u \times v$ 
         $r \leftarrow \text{sorted}(\{(n \cdot h)^2 | h \in g\})$ 
         $\mathcal{L} \leftarrow \text{get\_middle\_element}(r)$ 
        if  $\mathcal{L} < M_{least}$  then
             $M_{least} \leftarrow \mathcal{L}$ 
             $v_{best} \leftarrow n$ 
        end if
    end repeat
    return  $v_{best}$ 
end function

```

Можно оценить применимость такого алгоритма на практике. Предположим, что в данных менее 50% выбросов (а это уже очень много) и пусть $l = 100$. Также предположим, что оставшиеся векторы имеют достаточно ма-

лую погрешность, чтобы ей пренебречь. Тогда можно оценить вероятность того, что алгоритм выдаст неверное решение. Такое произойдёт, если на всех итерациях хотя бы один из двух векторов будет выбросом, вероятность такого события не больше $p_{fail,max} = 1 - 0.5 \cdot 0.5 = 0.75$. Но таких гипотез проверяются l штук, поэтому вероятность ошибки можно оценить сверху числом $(p_{fail,max})^l = 0.75^{100}$, это меньше, чем 10^{-12} , что является вполне приемлемым результатом.

Визуализация типичных исходных данных для этого алгоритма, а также результата его работы показана на рисунке 5. Точками обозначаются векторы из множества g , цвет точки характеризует её удаление от найденной плоскости, полупрозрачный круг — сама найденная плоскость.

2.7. Функция потерь

Пусть известно некоторое приближение функции задержки гироскопа $d(t)$ на отрезке $[t_0 - \Delta t, t_0 + \Delta t]$. Также, для каждой пары последовательных кадров f_i и f_{i+1} принадлежащих этому отрезку, известны направляющие векторы $r_i(X_j)$ и $r_{i+1}(X_j)$ лучей, проходящих через центры соответствующих камер и точки пространства X_j , где $j \in \mathcal{F}(i, i+1)$. Устранив вращательное движение (новые лучи буду обозначать как r') и применив алгоритм 1, можно приблизительно найти направления движения от каждого предыдущего кадра f_i к последующему f_{i+1} . Обозначим направление движения от предыдущего кадра f_i к последующему f_{i+1} как \tilde{v}_i .

Далее предлагается применить выведенное ранее условие отсутствия вращательного движения (9) для оценки качества приближений t и $\{v_i\}$. В экспериментальных версиях реализации в качестве функции потерь использовалась сумма, в которой каждое слагаемое — квадрат левой части условия отсутствия вращательного движения:

$$\begin{aligned} h_{i,j} &= r'_i(X_j) \times r'_{i+1}(X_j), \\ \mathcal{L} &= \sum_{i=l}^r \sum_{j \in \mathcal{F}(i, i+1)} (\tilde{v}_i \cdot h_j)^2. \end{aligned} \tag{10}$$

Однако, в реальных данных оказалось слишком много выбросов для надёжной работы этого подхода. Пример таких данных есть на рисунке 5а. Для того чтобы снизить влияние выбросов, функция потерь была изменена, как

описано в [14, с.69] — вместо того, чтобы возводить ошибки x в квадрат, используется функция $\log(1 + (kx)^2)$. Такая функция для небольших x ведёт себя как квадратичная, при увеличении x приближается к логарифмической, снижая влияние таких x . Отдельного внимания заслуживает коэффициент k . Это гиперпараметр, который определяет, какие отклонения больше похожи на выбросы, а каким можно доверять. Сделать его одинаковым для всех кадров нельзя — есть кадры, где почти нет ни выбросов, ни движения, на них имеет смысл сделать k меньше, чем на тех, где много выбросов. Поэтому, возьмем $k = \alpha/\sigma(x)$, чтобы $\sigma(kx) = \alpha$, где σ — среднеквадратичное отклонение, α — новый гиперпараметр, который необходимо подобрать. Итак, функция потерь принимает вид:

$$\mathcal{L} = \sum_{i=l}^r \sum_{j \in \mathcal{F}(i, i+1)} \log \left(1 + (k(\tilde{v}_i \cdot h_j))^2 \right). \quad (11)$$

Для проверки и визуализации функции потерь, из одного из тестовых видео было извлечено тридцать последовательно идущих кадров. Для каждой задержки гироскопа на отрезке $[-2000, 2000]$ мс с шагом в 2 мс были оценены направления движения в каждой паре последовательных кадров при помощи описанного в 2.6 рандомизированного алгоритма. На основе этих направлений и значений задержки вычислялась функция потерь. Её значения отмечались на графике (см. рисунок 6). Из-за рандомизированного характера алгоритма, график сильно зашумлен, но поскольку в итоговом алгоритме рандомизированная часть будет обрабатывать только один раз в фазе инициализации, это не является проблемой.

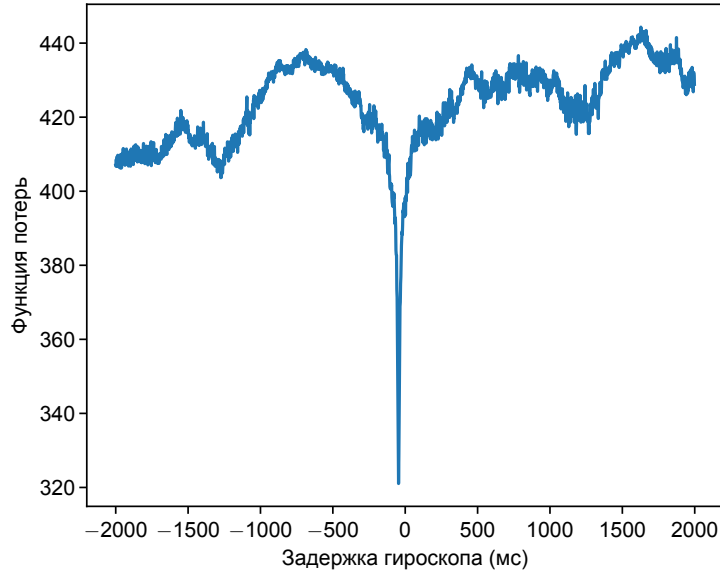


Рисунок 6 – Визуализация функции потерь

2.8. Оптимизация направления движения для пары кадров

Пусть есть пара кадров, для которой известно некоторое приближение вектора направления движения v_i . Также известно приближение для функции задержки гироскопа $d(t_c(f_i)) = d_0$ (в предположении, что задержка постоянна на коротком отрезке времени). Тогда, можно записать функцию потерь для этой пары кадров и применить итеративный алгоритм, чтобы уточнить вектор направления движения v_i , не меняя d_0 . В работе предлагается использовать для этого алгоритм BFGS в реализации из GNU Scientific Library [15]. Для его успешного применения необходимо реализовать вычисление градиента функции потерь, причём, поскольку параметр d_0 зафиксирован, дифференцировать по нему не требуется. По остальным параметрам, дифференцирование реализовано аналитически, а для того, чтобы зафиксировать условие $\|v_i\| = 1$, проведена замена переменных $v_i = \frac{v'_i}{\|v'_i\|}$.

В случае, когда необходимо уточнить векторы направления движения в функции потерь для большего количества пар кадров, можно просто проводить описанный выше процесс для каждой пары кадров независимо, что, в том числе, позволяет распараллелить процесс для лучшей производительности на многоядерных системах.

2.9. Грубая оценка и оптимизация задержки гироскопа

Для того чтобы приступить к минимизации функции потерь методами, основанными на градиентах, необходимо, в том числе, достаточно хорошее приближение $d(t_0)$.

Предлагается использовать алгоритм для приблизительной оценки v_i , описанный в главе 2.6, чтобы для каждого фиксированного $d(t_0)$ из некоторого подмножества области допустимых значений $d(t_0)$ приблизительно найти оптимум функции потерь. Взять за начальное приближение задержки гироскопа, необходимо такое значение $d(t_0)$, для которого этот оптимум будет минимальным. Псевдокод алгоритма, находящего приближение $d(t_0)$ описанным способом представлен в листинге 2.

Алгоритм 2 – Алгоритм поиска начального приближения задержки гироскопа

```
function GuessDelay( $\{r_i(X_j)\}$ ,  $\{t_c(p_i(X_j))\}$ ,  $q_{cam}$ )
     $\mathcal{L}_{min} \leftarrow \infty$ 
    for  $d \leftarrow \text{range}(d_{min}, d_{max}, step)$  do
         $r'_i(X_j) \leftarrow \text{Stabilize}(d, r_i(X_j), t_c(p_i(X_j)), q_{cam})$  // для всех  $i, j$ 
        for  $i \leftarrow$  номера выбранных кадров  $\{f_i\}$  do
             $v_i \leftarrow \text{GuessTranslationalMotion}(\{r'_i(X_j) \times r'_{i+1}(X_j)\}_j)$ 
        end for
         $\mathcal{L} \leftarrow \text{LossFunction}(d, \{v_i\}, \{r_i(X_j)\})$ 
        if  $\mathcal{L} < \mathcal{L}_{min}$  then
             $\mathcal{L}_{min} \leftarrow \mathcal{L}$ 
             $d_{best} \leftarrow d$ 
        end if
    end for
    return  $d_{best}$ 
end function
```

Имея приближения $d(t) = d_0$ и направлений движения $\{v_i\}$ для какого-то множества пар кадров, можно оптимизировать функцию потерь по переменной d_0 . В работе предлагается использовать для этого простой метод на основе градиентного спуска, использующий шаги, удовлетворяющие первому условию Вольфе.

2.10. Алгоритм синхронизации для короткого фрагмента видео

Алгоритм предлагается строить на основе методов, рассмотренных в главах 2.4 - 2.9. Чтобы получить исходные данные для этого алгоритма, можно

просто применить методы, рассмотренные в главах 3.1 - 2.1 в порядке изложения.

Общая идея заключается в том, чтобы сначала дать какое-то начальное приближение минимума функции потерь, затем с помощью методов оптимизации получить точный минимум.

Предлагаемое решение предполагает установку гиперпараметров k_i функции потерь на основе приблизительных оценок задержки гироскопа и направлений движения, а они в ходе оптимизации изменяются. Поэтому, вокруг процесса оптимизации добавлен внешний цикл с фиксированным числом итераций, который переоценивает гиперпараметры и продолжает процесс оптимизации.

Кроме того, оптимизируемая функция не является выпуклой, поэтому оптимизация может “застрывать” в локальных минимумах. Для того чтобы уменьшить вероятность таких “застрываний”, предлагается в том же внешнем цикле, который переоценивает гиперпараметры также находить новые приближения для $\{v_i\}$, тем самым возвращая их близко к глобальному минимуму.

Алгоритм 3 – Алгоритм поиска задержки гироскопа

```

function EstimateGyroDelay( $\{r_i(X_j)\}$ ,  $\{t_c(p_i(X_j))\}$ ,  $q_{cam}$ )
   $d \leftarrow \text{GuessDelay}(\{r_i(X_j)\}, \{t_c(p_i(X_j))\}, q_{cam})$ 
  repeat  $n$  times
    for  $i \leftarrow$  номера выбранных кадров  $\{f_i\}$  do
       $v_i \leftarrow \text{GuessTranslationalMotion}(\{r'_i(X_j) \times r'_{i+1}(X_j)\}_j)$ 
       $k_i \leftarrow \text{EstimateHyperparameter}(v_i, \{r'_i(X_j) \times r'_{i+1}(X_j)\}_j)$ 
    end for
    while Шаг градиентного спуска не меньше  $\epsilon$  do
      Оптимизировать функцию потерь по  $\{v_i\}$  при помощи BFGS
      Сделать шаг градиентного спуска по  $d$ 
    end while
  end repeat
  return  $d$ 
end function

```

2.11. Упрощённая модификация алгоритма синхронизации

Некоторые авторы [5, 7] в своих исследованиях при составлении функции потерь полностью пренебрегали поступательным движением камеры, мотивируя это тем, что камера движется не быстро. В данной работе было при-

нято решение отказаться от такой этой идеи, так как в предметной области — видео с БПЛА, встречается существенное поступательное движение.

Однако, например, авторы [7] получали хорошую точность синхронизации, пренебрегая поступательным движением. Постановка задачи достаточно сильно отличается от постановки в этой работе, их решение накладывает с одной стороны больше ограничений на функцию задержки гироскопа, а с другой стороны — кроме задержки гироскопа находит ещё взаимное расположение гироскопа и камеры. Таким образом, результаты этой работы с результатами, полученными в их статье напрямую не сравнить и было решено разработать упрощённую модификацию предложенного в этой работе алгоритма, которая также не учитывала бы поступательное движение и сравнить её с полноценной версией.

Для этого потребовалось изменить функцию потерь, убрав оттуда v_i :

$$\mathcal{L} = \sum_{i=l}^r \sum_{j \in \mathcal{F}(i, i+1)} \log \left(1 + (k \|h_j\|)^2 \right). \quad (12)$$

Также потребовалось убрать из цикла оптимизации весь код, связанный с первоначальной оценкой и оптимизацией v_i . Функция потерь стала по смыслу близка к функции, предложенной в [7]. Задача оптимизации стала одномерной, время работы алгоритма сократилось.

Выводы по главе 2

Были описаны этапы разработки и реализации алгоритма синхронизации видео и данных о вращательном движении камеры, учитывающего эффект сканирующего затвора и поступательное движение. Также был предложен способ существенно упростить реализацию, пренебрегая поступательным движением и, возможно, жертвуя точностью синхронизации.

ГЛАВА 3. ТЕСТИРОВАНИЕ АЛГОРИТМА СИНХРОНИЗАЦИИ

В этой главе проводится сравнение предложенного алгоритма с его упрощённой версией, пренебрегающей поступательным движением, а также методом ‘using visual features’ из GyroFlow.

3.1. Описание тестового набора данных

Был сформирован набор из следующих тестовых видео:

- а) Видео, предоставленные разработчиками GyroFlow, как сложные для автоматической синхронизации
 - 1) “skiing”, 60 к/сек, камера закреплена на голове лыжника, спускающегося с горы.
 - 2) “trail running 1”, 30 к/сек, камера в руке человека, бегущего по горной тропе.
- б) Видео, отснятые в рамках выполнения этой работы
 - 1) “table”, 60 к/сек, камера движется над плоским столом на высоте нескольких десятков сантиметров.
 - 2) “fpv flight 1”, 60 к/сек, камера закреплена на БПЛА, выполняющим полёт на высокой скорости на высоте около метра, затем резко набирающим высоту около 20 метров.
 - 3) “fpv flight 2”, 60 к/сек, камера закреплена на БПЛА, длительный полёт с разнообразными манёврами.
 - 4) “longterm drift”, 60 к/сек, выдержка зафиксирована, выполняются периодические вращательные движения, сцена неизменна.

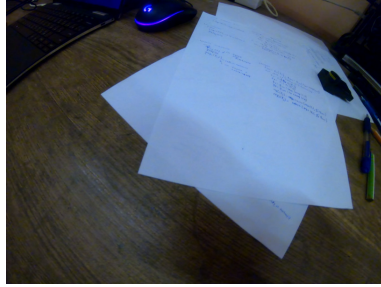
Примеры кадров из этих видео представлены на рисунке 7.



(a) skiing



(b) trail running 1



(c) table



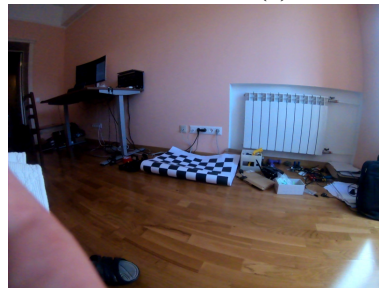
(d) fpv flight 1



(e) fpv flight 2 (первый пример)



(f) fpv flight 2 (второй пример)



(g) longterm drift

Рисунок 7 – Примеры кадров из тестовых видео

Видео отсняты на экшн-камеру GoPro Hero 6 и имеют разрешение 2704×2028 пикселей и время считывания матрицы 11.11 мс. Кроме того, в видеофайлах встроена дорожка с угловыми скоростями с частотой дискретизации $f_s = 200$ Гц. Для рассматриваемой камеры есть готовые профили в GyroFlow, это значит, что параметры модели камеры известны.

Видеопоток находится в контейнере MP4 вместе со звуком и метаданными в формате GPMF [16]. Угловые скорости — часть метаданных GPMF. Для извлечения их из видеофайла в работе предлагается использовать библиотеку *telemetry-parser* [17], которая также может разбирать инерциальные данные

из примерно десятка других форматов. Кадры видео извлекаются при помощи библиотеки OpenCV.

3.2. Характер фактической функции задержки гироскопа

Для того чтобы убедиться в применимости метода, предложенного в главе 1.6, необходимо проверить, выполняется ли предположение о линейности функции задержки гироскопа. Для этого, было отснято тестовое видео с периодическими вращательными движениями камеры, зафиксированной выдержкой и статической сценой, разбито на фрагменты длительностью по одной секунде, на каждом фрагменте запущена полная версия предложенного алгоритма синхронизации. Результаты его работы можно увидеть на рисунке 8.

Линейность в долгосрочной перспективе, очевидно, не соблюдается, однако, на видео длиной порядка 30 секунд (какими являются большинство тестовых видео), предположение о линейности функции задержки сделать можно.

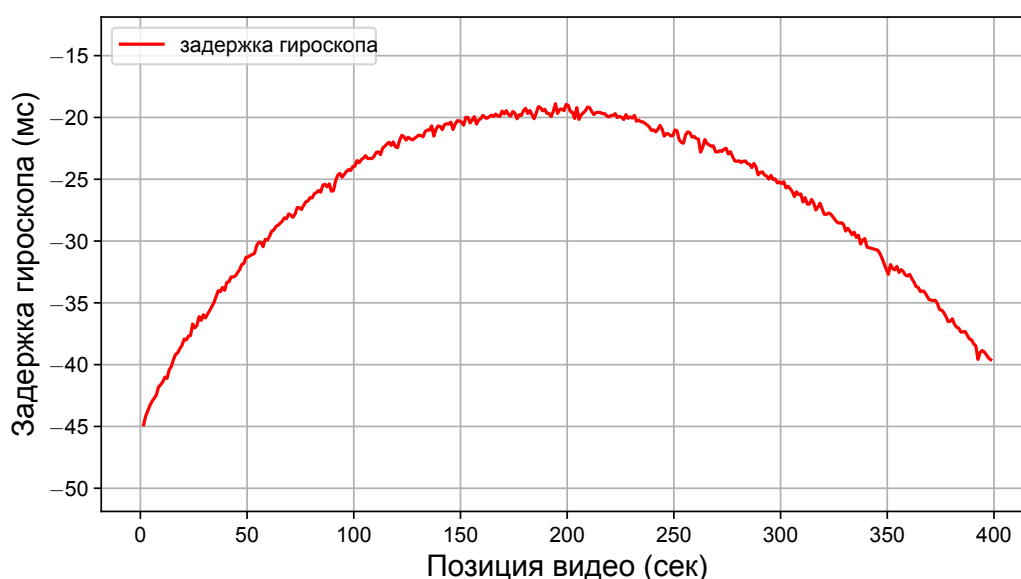


Рисунок 8 – Результат работы полной версии на наборе данных “longterm drift”

3.3. Проверка корректности найденных задержек

Для проверки соответствия полученных при помощи полноценной модификацией алгоритма синхронизации результатов действительности на всех наборах данных (в том числе из главы 3.2) использовалось ПО для стабилизации видео GyroFlow. GyroFlow позволяет вводить задержку гироскопа вручную в нескольких точках, после чего можно просмотреть стабилизированное

видео и убедиться, варьируя значение задержки гироскопа в пределах порядка 5 мс и смотря на результат стабилизации, что найденная задержка близка к реальной. Однако, нужно помнить, что GyroFlow начинает отсчитывать время не от момента съёмки первой строки первого кадра, а от середины первого кадра, поэтому к задержке гироскопа необходимо прибавлять $t_r/2$, в данном случае это $11.11/2 = 5.555$ мс. После этого, нужно взять задержку гироскопа со знаком минус и создать ручную точку синхронизации с этим значением.

3.4. Оценка качества синхронизации и сравнение с упрощенной версией

Для оценки качества выдаваемых алгоритмом решений был использован метод, предложенный в главе 1.6. Видео разбивалось на фрагменты, возможно с пересечениями, на каждом фрагменте запускался алгоритм, и выдавал приближение задержки гироскопа на этом фрагменте. Фактическую функцию задержки гироскопа для тестовых видео, не превышающих по длительности нескольких десятков секунд, можно считать линейной, как было показано в главе 3.2. По приближениям задержки гироскопа на отрезках, при помощи линейной регрессии находилась функция, которая считалась за функцию фактической задержки гироскопа. Строились графики этой функции, линейной интерполяции результатов работы алгоритма и модуля их разности. От значений разности в начальных точках отрезков видео считалось среднеквадратичное отклонение и также помещалось на рисунок. На рисунках используются две вертикальные шкалы, левая шкала для модуля разности, правая — для всех остальных графиков.

На рисунке 9 представлены результаты работы двух модификаций алгоритма, с учётом поступательного движения и без, на наборах данных “trail running 1” и “skiing”. Можно заметить, что обе модификации алгоритма отлично справляются с задачей.

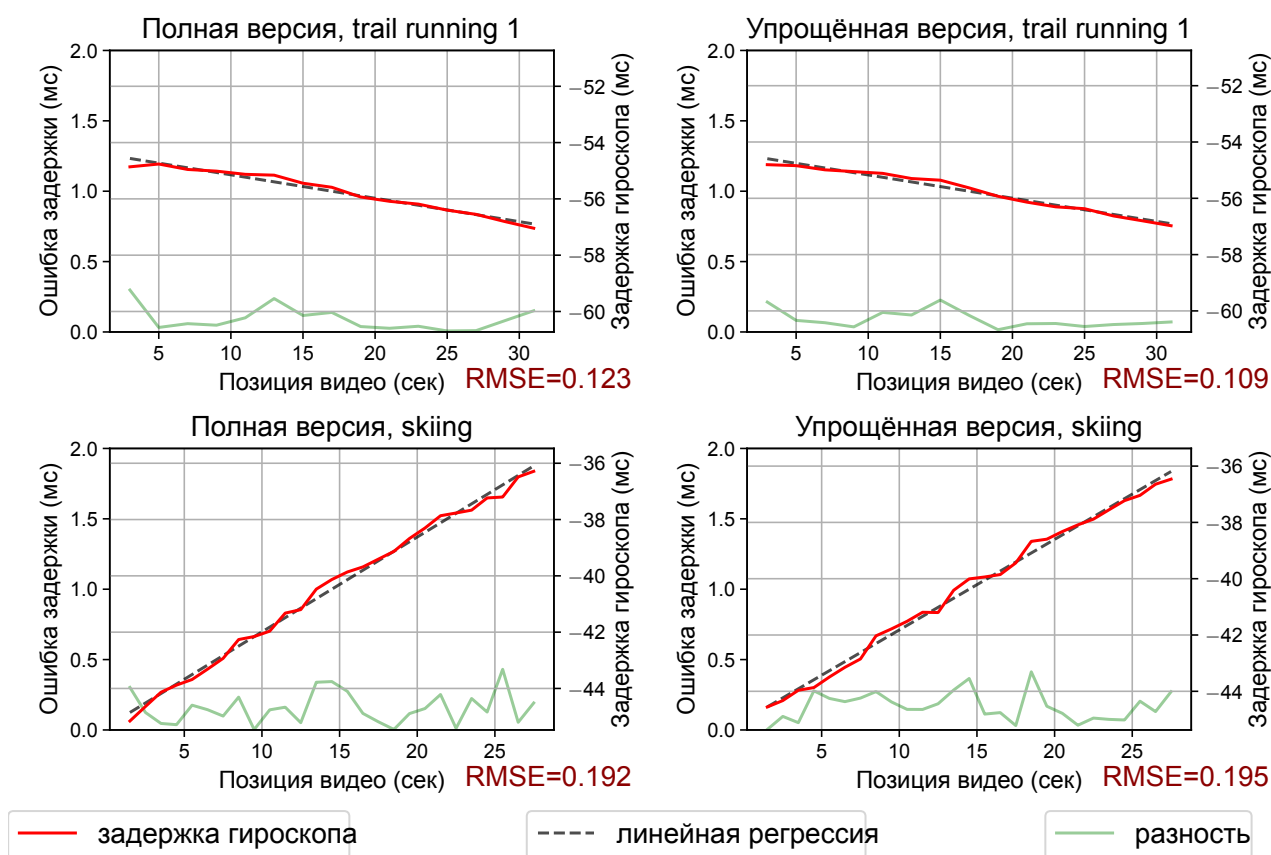


Рисунок 9 – Результаты на наборах данных “trail running 1” и “skiing”

На рисунке 10 представлены результаты работы тех же алгоритмов на наборах данных “table” и “fpv flight 1”. На этих данных модификация, пренебрегающая поступательным движением заметно проигрывает полной версии алгоритма в качестве решения. Вероятная причина такого результата заключается в следующем. Пока на видео присутствуют точки ‘на бесконечности’, то есть удалённые настолько, что поступательное движение камеры на них влияет несущественно, упрощённый алгоритм работает хорошо — влияние близких точек ослабляется аналогично выбросам, и равенство функции потерь своему минимуму фактически означает отсутствие вращательного движения. Также, упрощённый алгоритм хорошо работает на видео, где вращательное движение вызывает много большие смещения изображения в пикселях, чем поступательное. Эти же наборы данных были специально созданы для проверки изложенных выше гипотез — быстрое поступательное движение, на небольшой высоте, так, чтобы в кадр не попадали удалённые точки.

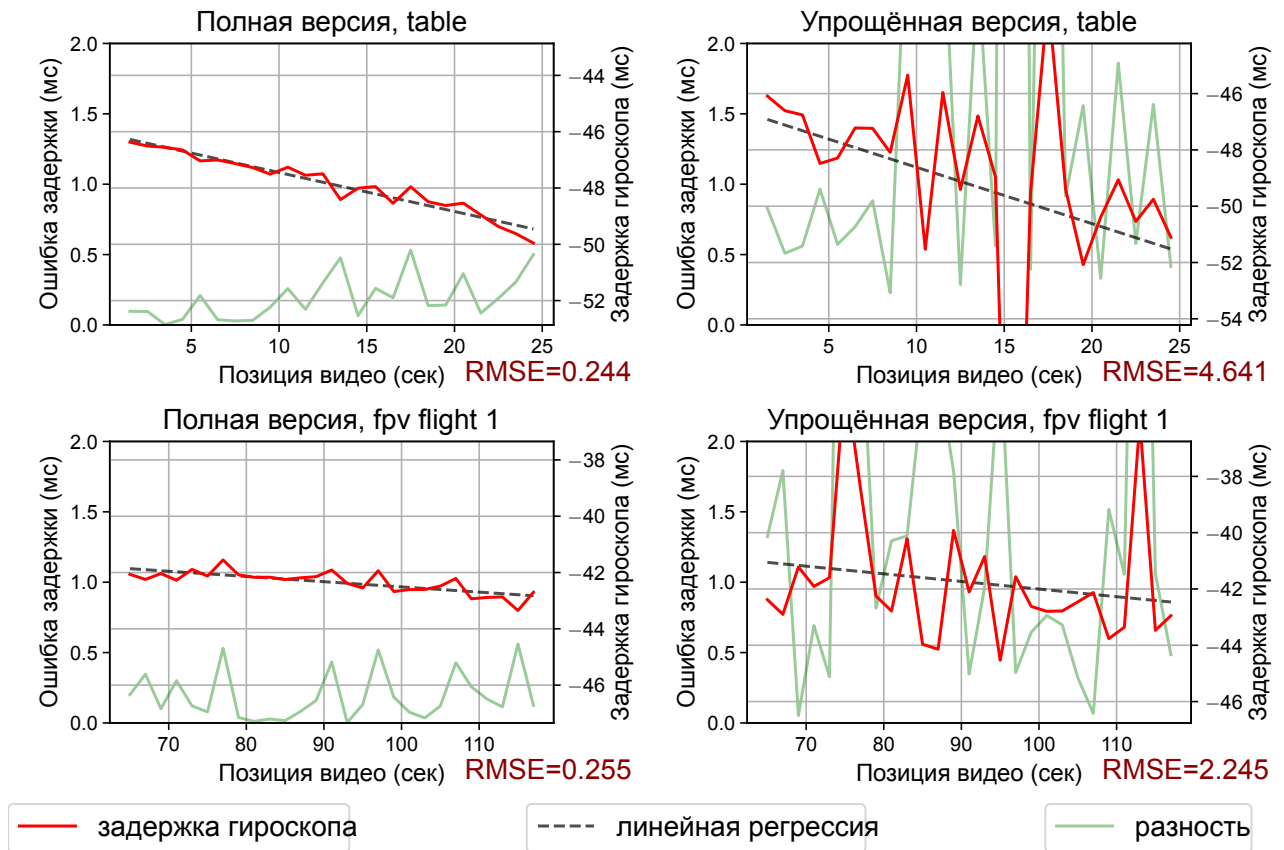


Рисунок 10 – Результаты на наборах данных “table” и “fpv flight 1”

3.5. Сравнение с одним из методов из GyroFlow

Также было проведено сравнение полной версии предлагаемого алгоритма синхронизации с методом синхронизации ‘using visual features’ из GyroFlow. Тестирование проводилось на наборе данных ‘fpv flight 2’. Для каждой секунды видео был запущен предлагаемый алгоритм синхронизации. Видео также было загружено в GyroFlow и добавлено несколько точек автоматической синхронизации на отрезке времени с 142 секунды по 153 секунду. На рисунке 11 представлены найденные задержки алгоритмом из GyroFlow и предлагаемым алгоритмом.

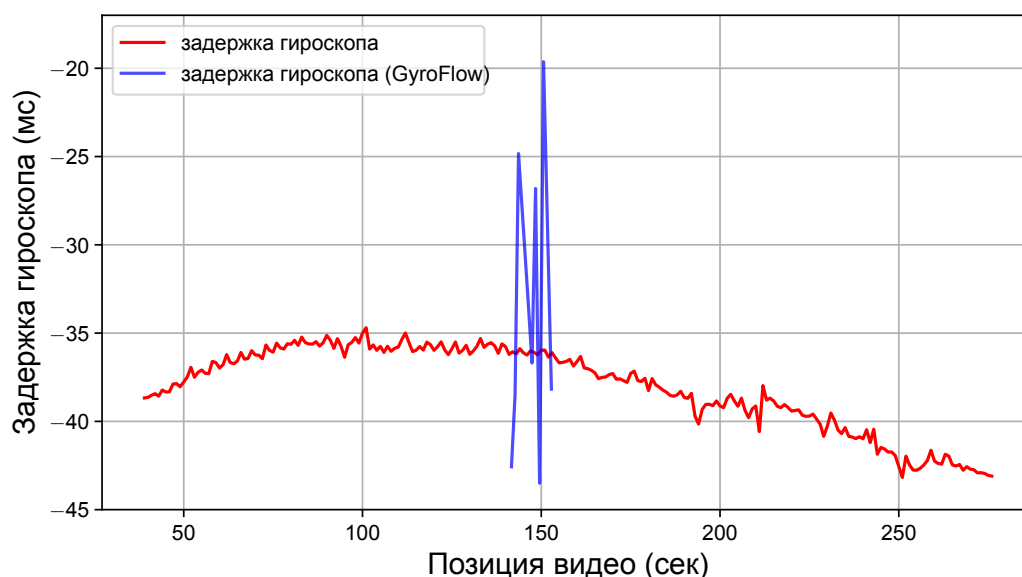


Рисунок 11 – Результаты на наборе данных “fpv flight 2”

Можно заметить, что на этом тесте алгоритм из GyroFlow проигрывает по точности предложенному алгоритму (фактическая задержка гироскопа не может меняться так быстро). Это вполне ожидаемо, поскольку, как и упрощённая версия предложенного алгоритма, алгоритм ‘using visual features’ пренебрегает поступательным движением, а этот тестовый фрагмент содержит преимущественно поступательное движение.

Выводы по главе 3

Была проверена работа и проведено сравнение двух модификаций алгоритма на нескольких тестовых видео. В результате этой проверки выяснилось, что упрощённая модификация алгоритма, пренебрегающая поступательным движением камеры, сильно уступает по точности синхронизации полной версии. Также, предложенный алгоритм на многих тестовых видео превосходит по точности синхронизации алгоритмы из GyroFlow.

ЗАКЛЮЧЕНИЕ

В рамках работы проведён анализ существующих методов синхронизации данных гироскопа и видео, разработан и реализован алгоритм, решающий эту задачу. Также, была реализована упрощённая модификация этого алгоритма, использующая функцию потерь, пренебрегающую поступательным движением.

Проведено сравнительное тестирование исходного алгоритма и этой модификации и показано, что упрощённая модификация не выдаёт удовлетворительные результаты на некоторых тестовых видео. Полная версия алгоритма, напротив, показала себя хорошо.

Ведется работа по интеграции предложенного алгоритма в ПО для стабилизации видео с открытым исходным кодом GyroFlow в качестве одного из возможных алгоритмов синхронизации.

Таким образом, все поставленные задачи были решены, цель достигнута: реализован алгоритм синхронизации данных с гироскопа и видео, учитывающий как эффекты сканирующего затвора, так и поступательное движение камеры.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 OpenCV: Fisheye camera model [Электронный ресурс]. — URL: https://docs.opencv.org/3.4/db/d58/group__calib3d__fisheye.html (дата обр. 15.05.2022).
- 2 ElvinC/gyroflow: [Legacy version] Video stabilization using IMU motion data from internal or external logs [Электронный ресурс]. — URL: <https://github.com/ElvinC/gyroflow> (дата обр. 15.05.2022).
- 3 yossato/virtualGimbal: virtualGimbal is an electronic stabilize device for videos that were taken by hand held camera. (I.e. DSLR) [Электронный ресурс]. — URL: <https://github.com/yossato/virtualGimbal> (дата обр. 15.05.2022).
- 4 *Hartley R., Zisserman A.* Multiple View Geometry in Computer Vision. — 2-е изд. — USA : Cambridge University Press, 2003. — ISBN 0521540518.
- 5 Digital Video Stabilization and Rolling Shutter Correction using Gyroscopes : тех. отч. / A. Karpenko [и др.] ; Salt Lake City Corporation. — 2011. — URL: https://graphics.stanford.edu/papers/stabilization/karpenko_gyro.pdf.
- 6 *Jia C., Evans B. L.* Online calibration and synchronization of cellphone camera and gyroscope // 2013 IEEE Global Conference on Signal and Information Processing. — 2013. — С. 731–734. — DOI: 10.1109/GlobalSIP.2013.6736995.
- 7 *Ovrén H., Forssén P.-E.* Gyroscope-based video stabilisation with auto-calibration // 2015 IEEE International Conference on Robotics and Automation (ICRA). — 2015. — С. 2090–2097. — DOI: 10.1109/ICRA.2015.7139474.
- 8 gyroflow/gyroflow: Video stabilization using gyroscope data [Электронный ресурс]. — URL: <https://github.com/gyroflow/gyroflow> (дата обр. 15.05.2022).
- 9 hovren/crisp: Camera-to-IMU calibration and synchronization toolbox [Электронный ресурс]. — URL: <https://github.com/hovren/crisp> (дата обр. 15.05.2022).

- 10 *Haarbach A., Birdal T., Ilic S.* Survey of Higher Order Rigid Body Motion Interpolation Methods for Keyframe Animation and Continuous-Time Trajectory Estimation // 3D Vision (3DV), 2018 Sixth International Conference on. — IEEE. 2018. — С. 381–389. — DOI: 10.1109/3DV.2018.00051. — URL: <http://www.adrian-haarbach.de/interpolation-methods>.
- 11 *Kluge T.* Cubic splines [Электронный ресурс]. — URL: <https://kluge.in-chemnitz.de/opensource/spline/spline.pdf> (дата обр. 15.05.2022).
- 12 Kanade–Lucas–Tomasi feature tracker - Wikipedia [Электронный ресурс]. — URL: https://en.wikipedia.org/wiki/Kanade%E2%80%93Lucas%E2%80%93Tomasi_feature_tracker (дата обр. 15.05.2022).
- 13 Fast Optical Flow Using Dense Inverse Search / T. Kroeger [и др.] // Т. 9908. — 10.2016. — С. 471–488. — ISBN 978-3-319-46492-3. — DOI: 10.1007/978-3-319-46493-0_29.
- 14 *Zhang Z.* Parameter estimation techniques: a tutorial with application to conic fitting // Image and Vision Computing. — 1997. — Т. 15, № 1. — С. 59–76. — ISSN 0262-8856. — DOI: [https://doi.org/10.1016/S0262-8856\(96\)01112-2](https://doi.org/10.1016/S0262-8856(96)01112-2). — URL: <https://www.sciencedirect.com/science/article/pii/S0262885696011122>.
- 15 GSL - GNU Scientific Library - GNU Project - Free Software Foundation [Электронный ресурс]. — 2022. — URL: <https://www.gnu.org/software/gsl/> (дата обр. 15.05.2022).
- 16 gopro/gpmf-parser: Parser for GPMF™ formatted telemetry data used within GoPro® cameras [Электронный ресурс]. — URL: <https://github.com/gopro/gpmf-parser> (дата обр. 15.05.2022).
- 17 AdrianEddy/telemetry-parser: A tool to parse real-time metadata embedded in video files or telemetry from other sources like Betaflight blackbox. Supported formats: Sony, GoPro GPMF, Insta360, Betaflight blackbox (csv and binary) [Электронный ресурс]. — URL: <https://github.com/AdrianEddy/telemetry-parser> (дата обр. 15.05.2022).