Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем
Кафедра системного программирования

Корнилова Анастасия Валерьевна

Цифровая стабилизация видеоизображения с использованием МЭМС-датчиков в режиме реального времени

Выпускная квалификационная работа

Научный руководитель:
ст. преп. кафедры системного программирования СПбГУ Я. А. Кириленко

Рецензент:
к.т.н., профессор факультета компьютерных наук ВШЭ Е. М. Гринкруг

Санкт-Петербург
2018

SAINT PETERSBURG STATE UNIVERSITY

Software and Administration of Information Systems
Software Engineering

Kornilova Anastasiya

Real-time digital video stabilization using MEMS-sensors

Graduation Thesis

Scientific supervisor:
senior lecturer Iakov Kirilenko

Reviewer:
PhD, professor Efim Grinkrug

Saint Petersburg
2018

Содержание

Введение

За последние годы в среднем значительно увеличилось качество кадров, получаемых с различного рода непрофессиональных камер, в то время как качество видеоизображения оставляет желать лучшего. В большинстве случаев основной причиной несовершенства видео является тряска устройства, на которое ведется съемка, поэтому видео необходимо стабилизировать. Подобная проблема актуальна не только при любительской съемке на смартфон или экшн-камеру, но и в более промышленных задачах, например, аэросъемке или при дистанционном управлении дронами по видео.

На текущий момент существует три основных вида стабилизации видео: механическая, оптическая и цифровая. Механическая стабилизация подразумевает наличие внешнего устройства, гасящего колебания камеры. К устройствам такого вида можно отнести GyroStick, SteadyCam, стабилизирующие видео с мобильных устройств и видеокамер, или стабилизирующий подвес в случае съемки с квадрокоптера. В основе оптической стабилизации лежит система «плавающих» линз и матриц, устойчивая к небольшим колебаниям камеры. Данные подходы существенно повышают качество видео, но имеют ряд ограничений и недостатков: высокая стоимость, необходимость в наличии внешнего устройства и склонность к износу.

Более универсальным аналогом является цифровая стабилизация видео, основной принцип которой состоит в программном преобразовании кадров. Но лишенная вышеуказанных недостатков цифровая стабилизация имеет другую проблему — требует больших вычислительных ресурсов, что часто приводит к долгому времени обработки. Поэтому данный подход не применим на устройствах с относительно малыми вычислительными мощностями, например, на мобильных устройствах и во встраиваемых системах, и на текущий момент встречается только в составе профессиональных видеоредакторов (Adobe Premier, Movavi), либо при наличии больших вычислительных ресурсов (YouTube).

Анализ основных существующих алгоритмов [9, 7, 17, 15] показы-

вает, что из трех составляющих этапов цифровой стабилизации видео — определение траектории движения камеры по кадрам, сглаживание данной траектории и программное преобразование кадров для их соответствия сглаженной траектории — наиболее затратным по вычислениям является первый. В большинстве случаев это связано с необходимостью выделять и сопоставлять особые точки (features) в ходе анализа последовательности кадров. При таком подходе при увеличении разрешения видео время обработки только увеличивается, а алгоритм становится чувствительным к сцене кадра и может давать искажения при меняющейся освещенности или наличии на кадре больших движущихся объектов.

Альтернативным вариантом для определения движения камеры является использование информации с МЭМС-датчиков движения (гироскоп, акселерометр), позволяющее абстрагироваться от сцены на кадре и мгновенно оценить движение камеры. Такой подход подкреплен широким распространением указанных датчиков на одной платформе с камерой (смартфоны, встраиваемые системы), что увеличивает интерес к нему. Но в силу того, что данная идея стала развиваться сравнительно недавно — в 2013 году, на текущий момент она подкреплена лишь некоторыми теоретическими наработками [11, 1] без наличия открытой реализации алгоритма и, более того, реализации метода на какой-то конкретной платформе.

Трудности прототипирования алгоритмов стабилизации с использованием МЭМС-датчиков вызваны необходимым расширением математической модели камеры разнородной информацией с датчиков и, как следствие, усложнением ее. В математической модели появляются параметры датчиков, например, частота показаний, параметр опускающегося затвора (rolling-shutter), а также параметры модели «камера-датчики», например, взаимное расположение осей камеры и датчиков. Кроме того необходимо понимать, что задача стабилизации делится на две принципиально отличающиеся подзадачи: стабилизация локальной тряски (достижение эффекта статичной камеры) и стабилизация сложного движения (сохранение основного движения, ликвидация тряски-

шума), решения для которых необходимо искать на стыке компьютерного зрения и цифровой обработки сигналов.

В свою очередь, применение алгоритмов стабилизации на конкретных устройствах связано с необходимостью подбора значений параметров математической модели, которые сильно зависят от модели камеры, датчиков и их комбинаций. Поэтому для успешного масштабирования алгоритма стабилизации, чтобы он был применим не для одного конкретного устройства, необходимо иметь алгоритм автоматической калибровки этих параметров. Помимо этого возникает проблема синхронизации камеры и датчиков, вызванная тем, что кадр проходит предварительную обработку, например, кодирование. Нахождение параметра этой задержки для конкретной платформы также должно учитываться в алгоритме автоматической калибровки.

В данной работе представлено решение задачи цифровой стабилизации видео с применением автоматической калибровки. Для оценки технической осуществимости данное решение было реализовано на мобильном телефоне с ОС Android 5.0, в работе описаны результаты экспериментов.

1. Постановка задачи

Целью данной дипломной работы является разработка алгоритма цифровой стабилизации видео с использованием МЭМС-датчиков и реализация данного алгоритма на конкретной платформе.

Были поставлены следующие задачи.

1. Предложить и реализовать прототип алгоритма стабилизации видео с использованием МЭМС-датчиков для локальной тряски и сложного движения.

2. Предложить и реализовать прототип алгоритма автоматической калибровки параметров алгоритма стабилизации.

3. Реализовать на платформе Android приложение, использующее данный алгоритм стабилизации и автоматической калибровки.

2. Обзор

В данном разделе представлены базовые математические модели и определения, использующиеся при решении поставленной задачи, даны определения метрик качества стабилизации видео, которые задействованы в алгоритмах стабилизации и калибровки, а также описан метод сбора тестовых данных и их формат.

2.1. Основные математические модели

2.1.1. Модель pinhole camera

Модель *pinhole camera* является базовой математической моделью камеры, которая описывает отображение из трехмерного пространства реального мира в двумерное пространство матрицы камеры (рис. 1).

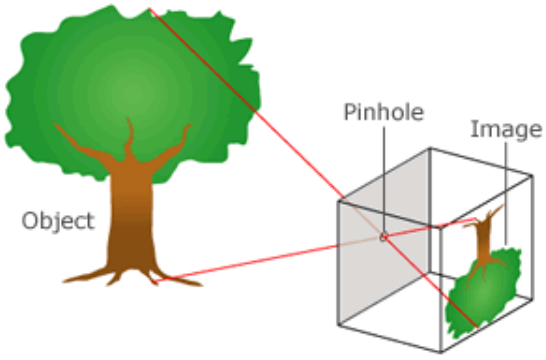

Рис. 1: Модель pinhole camera

Данная модель удовлетворяет соотношению ниже, где X — координаты точки в трехмерном пространстве, а x — координаты проекции этой точки на матрицу камеры и зависит от параметров камеры: f — фокусное расстояние камеры, (o_x, o_y) [16].

$$\begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = \begin{pmatrix} f_x & 0 & -o_x \\ 0 & f_y & -o_y \\ 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix}$$

Матрица, используемая в данном соотношении, называется *матрицей внутренних параметров камеры* и в дальнейших выкладках будет

обозначена K.

2.1.2. Вращение камеры

Рассмотрим, как изменяются координаты проекции одной и той же точки в пространстве на матрицу в случае вращения камеры. При вращении камеры с использованием оператора вращения R (матрица поворота) координаты фиксированной точки X в пространстве примут вид RX в пространстве относительно камеры. Значит, проекция этой точки будет иметь вид KRX. Если обобщить эту информацию, то для проекций x_1 и x_2 общей точки в пространстве X (рис. 2), полученных в моменты t_1 и t_2 соответственно, будут верны следующие соотношения:

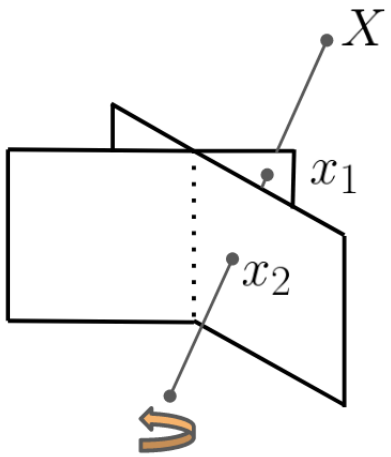

Рис. 2: Модель вращения камеры

$$x_1 = KR(t_1)X$$

$$x_2 = KR(t_2)X$$

Преобразуя эти выражения, получим следующую зависимость проекции при повороте камеры:

$$x_2 = KR(t_2)R^T(t_1)K^{-1}x_1$$

Таким образом, определим матрицу преобразования проекций между моментами времени t_1 и t_2:

$$W(t_1, t_2) = KR(t_2)R^T(t_1)K^{-1}$$

$$x_2 = W(t_1, t_2)x_1$$

2.1.3. Rolling shutter

Rolling shutter — метод съемки кадра, свойственный большинству малобюджетных камер. Основной особенностью этого метода является не мгновенный захват пикселей, описывающих сцену кадра, а последовательный захват пикселей по рядам сверху вниз. Получается, что верхние пиксели снимаются в более ранний момент времени, чем нижние. Из-за этого при движении камеры или наличии на сцене движущегося объекта возникают искажения (рис. 3) [1].

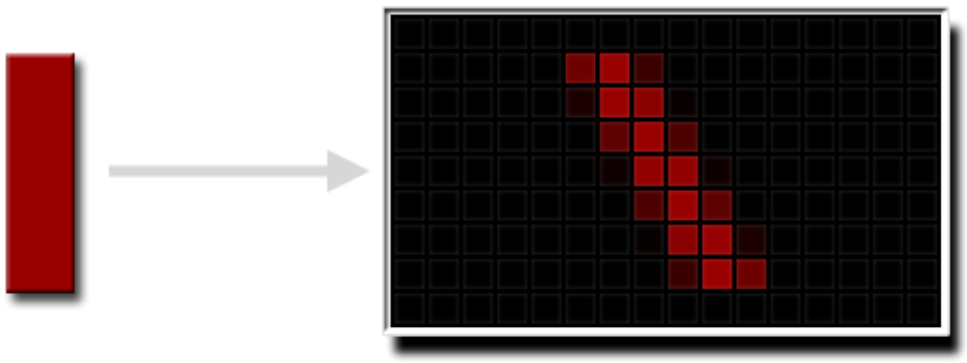

Рис. 3: Пример съемки движущегося объекта (слева) с использованием Rolling shutter

При опускании затвора время съемки пикселя линейно зависит от ряда, в котором он находится. Таким образом, если i — номер кадра, y — номер ряда на этом кадре, h – высота кадра в пикселях, t_s — время съемки одного кадра, то момент времени съемки пикселя в конкретном ряду $t(i, y)$ определяется следующим образом.

$$t(i, y) = t_i + t_s \frac{y}{h}$$

Эта информация может быть уточняющей в предыдущей модели вращения камеры:

[1] Изображения заимствованы с веб-сайта `http://www.red.com/learn/red-101/global-rolling-shutter`

$$x_2 = W(t(i_1, y_1), t(i_2, y_2))x_1$$

2.1.4. Датчик угловых скоростей

Датчик угловых скоростей или, по-другому, *гироскоп* позволяет получить информацию о текущих угловых скоростях тела, на котором этот датчик закреплен. Обычно получаемая от него информация состоит из четырех показаний: три из них соответствуют угловым скоростям вдоль трех осей датчика, четвертое соответствует временной метке (timestamp) этого показания. Наличие временной метки позволяет от угловых скоростей перейти к относительному положению тела в пространстве с помощью интегрирования.

Самым простым вариантом интегрирования является линейное интегрирование по каждой из осей без использования информации о других скоростях. Этот способ представлен в указанной формуле, где θ - угол поворота по одной из осей, а ω - скорость поворота по этой же оси.

$$\theta(t + \delta) = \theta(t) + \int_t^{t+\delta} \omega(t)dt \tag{1}$$

Более сложным, но более точным является сложное интегрирование, при котором задействуется информация с других осей. Этот метод может быть реализован как с использованием матриц поворота, так и с использованием кватернионов. Основные математические выкладки по интегрированию показаний гироскопа этими способами можно найти в работе [4].

2.2. Метрики качества стабилизации видео

На текущий момент существуют две основных метрики, оценивающие качество стабилизации видео: RMSE (root mean square error) и ITF (inter-frame transformation fidelity). Использование данных метрик для видео подразумевает, что снимаемая сцена статична и подвижной является только камера.

В основе RMSE лежит нахождение попиксельной разницы между двумя последовательными кадрами с использованием стандартной метрики L_2. После этого значения разниц складываются для всего видео. Чем выше эта сумма, тем менее стабилизировано видео.

Значение метрики ITF зависит от параметра PSNR (peak signal-to-noise ratio) между двумя последовательными кадрами $(k, k+1)$:

$$PSNR(k) = 10 \log_{10} \frac{I_{max}}{MSE(k)},$$

где I_{max} — максимальная яркость пикселя. Сама же метрика определяется следующим образом:

$$ITF = \frac{1}{N-1} \sum_{k=1}^{N-1} PSNR(k),$$

где N — количество кадров видео.

В случае метрики ITF более высокое значение говорит о лучшем качестве стабилизации.

2.3. Датасет

Для реализации алгоритмов стабилизации и калибровки необходимо иметь данные, содержащие кадры и информацию с датчиков. В рамках курсовой третьего курса было разработано мобильное приложение на базе операционной системы Android, которое собирает информацию о показаниях гироскопа и акселерометра и синхронизирует их с кадрами. Данное приложение работает на Android, начиная Android API level 21 (Android 5.0 Lollipop), в котором реализована событийная схема получения кадров с камеры.

Полученные в ходе сбора данные имеют следующий формат. В результате записи получается видеофайл в формате .mp4 и синхронизированные показания гироскопа/акселерометра в формате .csv (рис. 4). Показания датчиков имеют следующий вид: первые три значения — показания датчика по трем осям, четвертое показание — временная метка этих показаний (timestamp).

61	-0.008911133	-0.07762146	-0.067108154	13572928262166
62	-0.0025177002	-0.07229614	-0.07350159	13572933206013
63	f			
64	6.713867E-4	-0.07122803	-0.07562256	13572938180379
65	6.713867E-4	-0.07443237	-0.07562256	13572943154744
66	-0.0025177002	-0.07336426	-0.0745697	13572948098591
67	-0.007858276	-0.076553345	-0.06604004	13572953042439
68	-0.015304565	-0.07975769	-0.05857849	13572957986287
69	-0.0259552	-0.07762146	-0.051132202	13572962960652
70	-0.03555298	-0.07336426	-0.04260254	13572967904500
71	-0.04194641	-0.06590271	-0.03514099	13572972848347
72	-0.048324585	-0.058441162	-0.025558472	13572977822712
73	f			
74	-0.050460815	-0.05206299	-0.019165039	13572982766560
75	-0.053649902	-0.047790527	-0.015975952	13572987710408
76	-0.052597046	-0.049926758	-0.013839722	13572992654255
77	0.0493927	0.058441162	0.013839722	13572997628621

Рис. 4: Формат данных с гироскопа

Датасет содержит видео с разными конфигурациями: модель телефона, на который велась съемка, тип сцены — статическая или с наличием движущихся объектов, тип тряски — по одной оси или по нескольким.

2.4. Обзор решений с использованием МЭМС-датчиков

В данном разделе представлено описание существующих алгоритмов стабилизации видео с использованием МЭМС-датчиков.

2.4.1. Алгоритм с использованием фильтра Гаусса

В основе алгоритма, представленного в статье [11] в 2011 году, лежит использование фильтра Гаусса[2]. Путем интегрирования показаний MEMS-гироскопа для каждого кадра вычисляется относительное положение камеры в момент съемки, затем последовательность движений камеры сглаживается при помощи фильтра Гаусса (рис. 5) и по новой модели движения строится поток кадров, преобразованный согласно

[2] https://en.wikipedia.org/wiki/Gaussian_filter

формулам движения камеры. Для фильтра Гаусса можно задавать размер окна (на сколько дискретных точек он действует) и размер ядра (насколько сильно сглаживать). Меняя эти параметры, можно убирать либо локальную тряску, либо существенные движения.

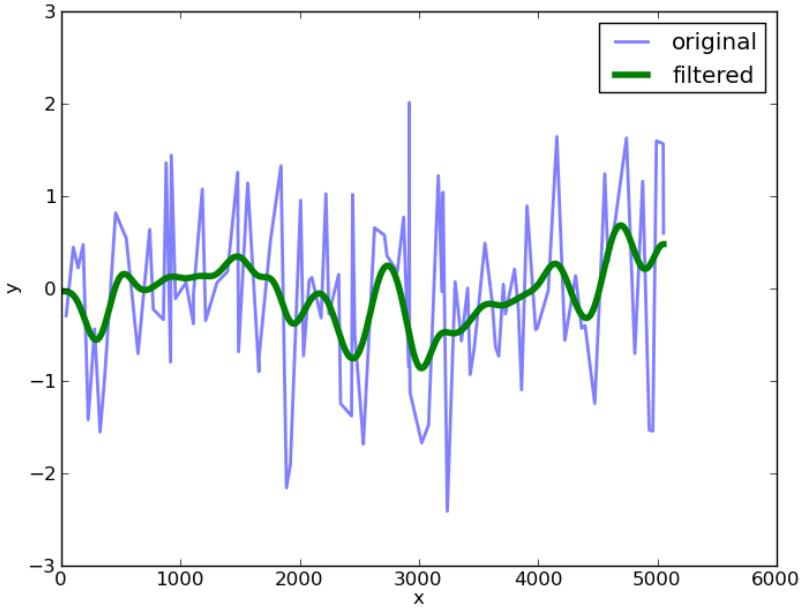

Рис. 5: Сглаживание траектории при помощи фильтра Гаусса

Исходный код прототипа представлен на Matlab, но в статье утверждается, что тестирование алгоритма проводилось на смартфоне iPhone 4. В открытой реализации представлен алгоритм с суженной областью вращения камеры — рассматривается вращение камеры только в горизонтальной плоскости, что не всегда соответствует поведению камеры при тряске.

2.4.2. Алгоритм с использованием нелинейного фильтра

В основе алгоритма, предложенного в статье [1] в 2014 году, используется более сложный нелинейный фильтр для сглаживания движений камеры.

В предлагаемом методе вводится понятие виртуальной камеры. На кадре определяются две статичные концентрические области — внутренняя и внешняя (рис. 6). Во внутренней области выбирается прямо-

угольник, в зависимости от положения которого принимается решение о положении виртуальной камеры.

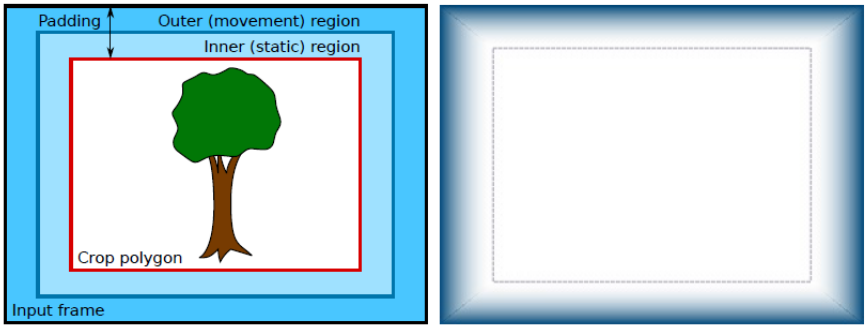

Рис. 6: Внешняя и внутренняя зоны стабилизации

При получении нового кадра высчитывается новое положение обозначенного прямоугольника на кадре. Если прямоугольник остается во внутренней области, то положение камеры остается тем же. Если хоть какая-то часть прямоугольника выходит за границу внутренней области, то скорость виртуальной камеры плавно сводится к скорости физической камеры за счет сферической линейной интерполяции — slerp[3](spherical linear interpolation). Авторы замечают, что алгоритм работает достаточно хорошо, но при достижении прямоугольником границы внутренней области происходит резкий скачок.

Также в статье предалагается адаптация данного метода для стабилизации видео в режиме реального времени. Для этого создается буфер фиксированной длины (3-5 кадров), по которому оцениваются и сглаживаются «будущие» движения камеры. Как утверждают авторы, эксперименты показывают, что использование предлагаемого метода требует меньшее количество кадров для буфера, а также дает более качественную стабилизацию по сравнению с предыдущим методом.

К сожалению, к статье не прикреплено исходного кода, по которому было бы возможно воспроизвести эксперимент.

[3]https://en.wikipedia.org/wiki/Slerp

3. Алгоритм стабилизации видео

Как уже было сказано во введении, стабилизация видео разделяется на две отдельные задачи — стабилизация локальной тряски (для достижения эффекта статичной камеры) и стабилизация видео при сложном движении (сохранение глобального движения, ликвидация локальной тряски). В данном разделе будут описанные предложенные алгоритмы и результаты их реализации.

3.1. Стабилизация локальной тряски

Под локальной тряской подразумевается вращательное движение камеры без переносов или близкое к нему. Главным результатом цифровой стабилизации в таком случае является получение видео, на котором камера статична.

3.1.1. Описание алгоритма

Предположим, что начальное положение камеры является эталонным, и алгоритм стабилизации должен преобразовать последующие кадры к такому виду, при котором достигалось бы первоначальное положение камеры. Для этого воспользуемся соотношением из 2.1.2. Зная координаты проекции точки на смещенном кадре и оператор поворота R для этого кадра, можно получить координаты точки с тем же цветовым значением, но в случае, если бы не было вращения камеры. Таким образом, для каждой точки кадра можно выполнить это преобразование и получить результирующий кадр. Для получения полной картины необходимо выполнить интерполяцию, так как данное преобразование не является инъективным. На рис. 7 продемонстрирован пример преобразования кадра при повороте камеры.

3.1.2. Методы интегрирования гироскопа

Одним из основных этапов в рамках данной задачи является нахождение оператора поворота R. Как было описано в обзоре, это мож-

Рис. 7: Пример поворота кадра вдоль вертикальной оси

но сделать двумя способами — обычным интегрированием с помощью углов Эйлера и интегрированием с использованием кватернионов.

В рамках данной дипломной работы были реализованы оба подхода. Результаты стабилизации сравнивались по определенной ранее метрике ITF — чем выше ее значение, тем лучше стабилизировано видео. На рис. 8 представлены результаты сравнения методов интегрирования на наборе видео из датасета.

На основании этих результатов можно сделать вывод, что интегрирование с использованием кватернионов качественнее стабилизирует видео. Также результаты показывают, что линейное интегрирование углов Эйлера повышает качество стабилизации видео и может быть использовано при наличии ограничений в плане вычислительных ресурсов.

3.1.3. Посекционное преобразование кадра

Частота показаний датчиков на рассматриваемых платформах обычно выше частоты получения кадров. Это можно видеть на примере файла показаний гироскопа в описании датасета. Значит, одному кадру соответствует несколько показаний гироскопа (в случае плат-

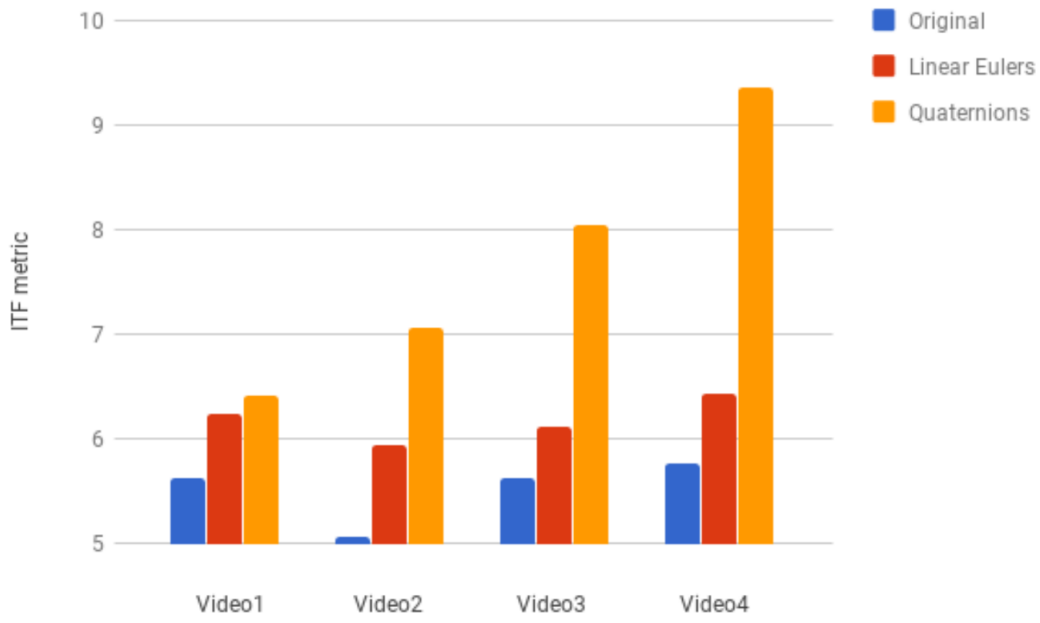

Рис. 8: Результаты сравнения алгоритмов с различными методами интегрирования гироскопа

формы Android обычно 4-8). Так как в рамках данного исследования рассматриваются камеры, снимающие по принципу rolling-shutter (см. обзор), это означает, что разным строкам соответствуют различные показания гироскопа, следовательно, оператор R может быть уточнен в зависимости от строки, в которой находится снимаемый пиксель.

В рамках данной работы была реализована поддержка модели rolling-shutter. При тестировании качества стабилизации видео с поддержкой данного параметра и без нее вариант с использованием параметра rolling-shutter показал лучшие результаты.

3.2. Стабилизация при сложном движении

Под сложным движением камеры подразумевается не только вращение в некоторой окрестности, а перемещения камеры в пространстве, полные повороты камеры. Это свойственно во время съемки при ходьбе, съемки с беспилотных летательных аппаратов. В таких случаях основную композицию съемки необходимо сохранить, а убрать только мелкую тряску камеры, из-за которой понижается качество видео.

Исследования в данном направлении выполнялись под моим руко-

водством студентом второго курса направления «Математическое обеспечение и администрирование информационных систем». В текущем разделе представлены основные результаты.

3.2.1. Фильтр Гаусса

Для того чтобы убрать локальные шумы при движении, необходимо сгладить сигнал, которым описывается движение камеры. Для сглаживания сигнала был выбран фильтр Гаусса, описанный в подходе 2.4.1. В решении были подобраны величина окна и размер ядра для оптимального сглаживания. Разработанный алгоритм поддерживает стабилизацию при трехмерном вращении камеры и стабилен при сложных движениях, например, при полном повороте вокруг одной из осей.

3.2.2. Использование показаний акселерометра для панорамной съемки

В рамках исследования был выделен отдельный случай съемки при сложном движении — панорамная съемка. При таком формате съемки после сглаживания с использованием фильтра Гаусса видео стабилизируется, но для лучшей композиции необходимо выравнивать горизонт. Для этого было предложено использовать информацию с акселерометра и расширить оператор R в формуле из 2.1.2 оператором поворота между вектором гироскопа и акселерометром.

Как результат, удалось получить выровненную панорамную съемку, задействуя показания гироскопа для ликвидации локальной тряски и показаний акселерометра для выравнивания по горизонту.

4. Алгоритм автоматической калибровки и синхронизации

В математической модели, используемой в алгоритмах стабилизации, присутствует большое количество параметров, зависящих от камеры (фокусное расстояние, оптический центр, коэффициенты искажения), датчиков и параметров системы «камера-датчики» (временная синхронизация, взаимное расположение осей камеры и датчиков). Для того чтобы алгоритмы стабилизации можно было масштабировать на широкий ряд платформ, необходимо иметь алгоритм автоматической калибровки этих параметров.

Исследования в данной области выполнялись студентом третьего курса направления «Программная инженерия» Поляковым Александром под моим руководством. Полный результат этого исследования был представлен в формате статьи на конференцию SYRCoSE'18 (Spring/Summer Young Researchers' Colloquium on Software Engineering). Статья принята к докладу на конференции и к публикации в журнале «Труды института системного программирования» [4]. В данном разделе приведены основные результаты работ по этому направлению.

4.1. Описание алгоритма

В рамках работы были выделены два основных параметра, критичных для масштабирования алгоритма: фокусное расстояние камеры и временная синхронизация камеры и датчиков. Во время исследования был осуществлен обзор и анализ существующих решений по калибровке этих параметров и их реализация в рамках фреймворка по стабилизации видео. Из рассмотренных алгоритмов ни один не показал удовлетворительных результатов, поэтому были предложены улучшения к существующим решениям, позволившие получить более точные результаты калибровки.

Предложенный в рамках исследования алгоритм работы по следую-

[4] `http://www.ispras.ru/proceedings/`

щему принципу: сначала ищется параметр временной синхронизации, после этого — фокусное расстояние. Для поиска временной синхронизации определяются две функции, каждая из которых описывает меру движения камеры — первая по показания гироскопа, вторая по смещению особых точек (features) на последовательных кадрах. Эти функции определяются следующим образом, где ω — угловые скорости гироскопа, $M(t)$ — множество сопоставленных особых точек, $(m_x,\ m_y)$ — координаты особых точек на кадре, t_{i-1} и t_i — временные метки кадров.

$$r_g(t) = \frac{\omega_x(t) + \omega_y(t) + \omega_z(t)}{3}$$

$$r_f(t) = \frac{\sum_{m \in M(t)} (m_x - m'_x) + (m_y - m'_y)}{2|M(t_i)|(t_i - t_{i-1})}$$

Полученные путем таких преобразований функции имеют следующий вид.

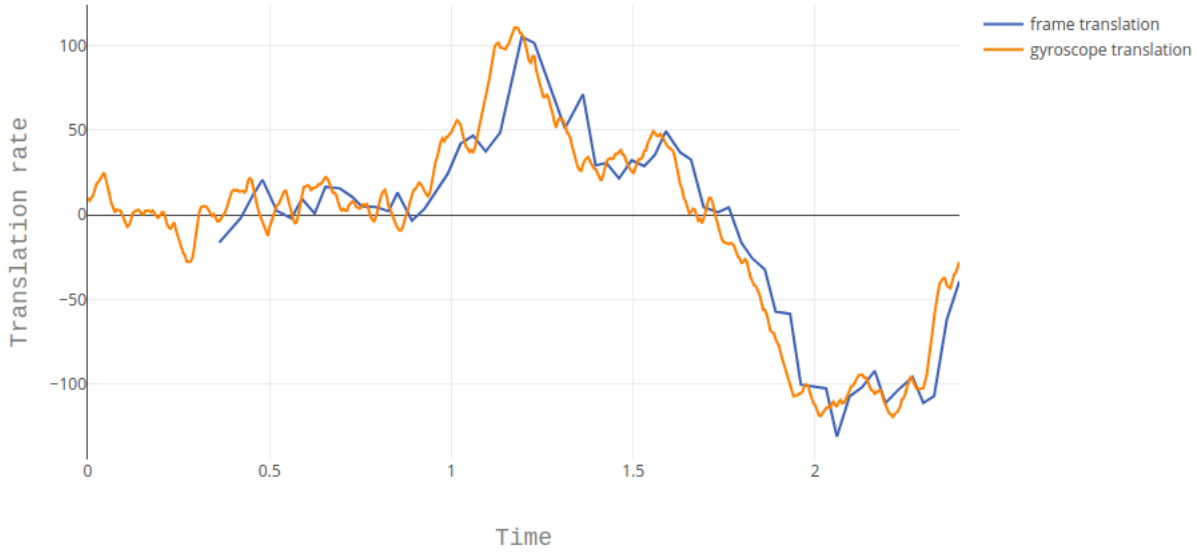

Рис. 9: Временная задержка между кадрами видео и показаниями гироскопа.

На графике (рис. 9) видно, что функции одинаково описывают движение двумя различными способами, но смещены во времени друг относительно друга на искомый параметр временной синхронизации. Для поиска этого параметра было предложено использовать функцию

кросс-корреляции для функций r_f и r_g. При постановке эксперимента для определения параметра временного сдвига была выбрана следующая функция, как наиболее точная:

$$s(a, b) = -|a - b|$$

Имея диапазон возможных значений параметра временной задержки (offset) T_d, необходимый параметр определяется следующим образом:

$$offset = \arg\max_{t_d \in T_d} \sum_{t \in T} s(r_f(t - t_d), r_g(t))$$

Следующим шагом предложенного алгоритма является поиск фокусного расстояния. Для этого определяется диапазон значений фокусного расстояния F, для каждого значения по видео вычисляется значение метрики ITF. Значение, при котором достигается максимум, принимается за текущее фокусное расстояние.

$$f = \arg\max_{f \in F} ITF(f, offset)$$

Использование данного алгоритма показало лучшие результаты в рамках математической модели, используемой для стабилизации видео. Постановка экспериментов показала, что оптимальным с точки зрения точности и производительности, является использование особых точек ORB (Oriented FAST and rotated BRIEF) [14].

5. Реализация на платформе Android

Для того чтобы доказать техническую осуществимость данного алгоритма, было принято решение реализовать разработанный алгоритм стабилизации видео с использованием МЭМС-датчиков в качестве приложения на базе операционной системы Android. Основная функциональность приложения состоит из записи видео, стабилизированного с использованием разработанного алгоритма. Главным требованием к приложению являлась обработка кадров в режиме реального времени, т.е. по окончании записи видео (при нажатии пользователем кнопки «Стоп») все кадры должны быть обработаны.

5.1. Основные технические решения

В качестве основного языка программирования был выбран Java, который является базовым для платформы Android. Android API поддерживает удобные интерфейсы для взаимодействия с камерой и датчиками, которые позволяют задавать низкоуровневые настройки данным устройствам, управлять частотой и форматами получаемых данных.

В качестве API для камеры был выбран модуль camera2, который поддерживается в Android, начиная с API level 21 (Android 5.0 и выше). Интерфейс этого модуля, в отличии от интерфейса камеры предыдущего поколения, поддерживает событийную схему получения кадров, что повышает качество алгоритма из-за точности при синхронизации кадров и показаний датчиков.

В рамках приложения было решено использовать библиотеку компьютерного зрения OpenCV C++, вызывая ее методы посредством JNI (Java Native Interface). Это решение обусловлено тем, что другие существующие реализации библиотек компьютерного зрения (OpenCV4Android SDK, OpenCV for Java и пр.) несовершенны с точки зрения использования памяти, требуют большого количества копирований изображений в процессе обработки, что критично с точки зрения предъявляемых к приложению ограничений по производительности.

5.2. Архитектура приложения

Процесс обработки видео состоит из следующих основных этапов и указан на рис. 10:

1. получение данных от камеры и гироскопа;
2. преобразование показаний гироскопа;
3. преобразование кадров для стабилизации изображения;
4. запись кадров в видео.

Для получения кадров используется обработчик событий о получении нового кадра ImageReader. ImageReader позволяет получить кадр во внутреннем формате Image Android API. Для дальнейших преобразований кадра, его необходимо скопировать из предоставленного DirectBuffer и передать в дополнительный поток обработки кадров. Для получения событий от гироскопа используется обработчик событий о получении новых данных SensorManager. Для гироскопа выставляется максимальная частота показаний, что позволяет более точно осуществлять интегрирование показаний, а также использовать посекционное преобразование кадра.

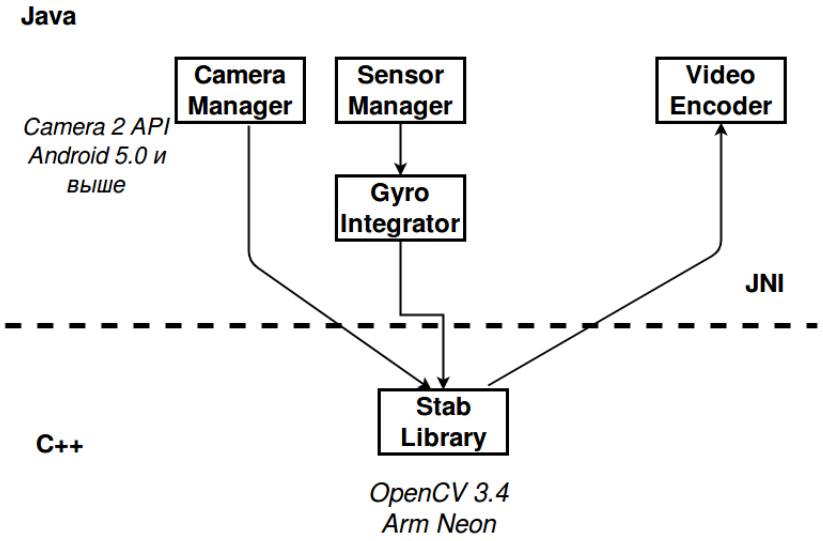

Рис. 10: Описание архитектуры приложения

Для преобразования показаний гироскопа реализован модуль, который интегрирует показания гироскопа, как указано в 2.1.2, и позволяет получить матрицу поворота для конкретной временной метки (timestamp). Данный модуль хранит информацию о всех показаниях гироскопа, что позволяет синхронизировать временные метки кадров с временным метками показаний и, как следствие, получить соответствующие матрицы повороты.

Для преобразования кадров реализована C++-библиотека с поддержкой OpenCV, в которой заложена основная логика алгоритма стабилизации.

После преобразования последовательность кадров сохраняется в Java-компоненте в видео и записывается на диск.

Заключение

В рамках дипломной работы были решены следующие задачи.

1. Предложен и реализован прототип алгоритма стабилизации видео с использованием МЭМС-датчиков для локальной тряски и сложного движения.

2. Предложен и реализован прототип алгоритма автоматической калибровки параметров алгоритма стабилизации.

3. Реализовано приложение на платформе Android, использующее данный алгоритм стабилизации и автоматической калибровки.

Полученное в ходе работ решение является доказательством технической осуществимости подхода к цифровой стабилизации с использованием МЭМС-датчиков.

Общие результаты исследования были представлены на конференции SECR'17 (Software Engineering Conference in Russia). Статья с результаты по направлению автоматической калибровке была принята к докладу на концеренции SYRCoSE'18 (Spring/Summer Young Researchers' Colloquium on Software Engineering) и к публикации в журнале «Труды института системного программирования» [5].

[5] http://www.ispras.ru/proceedings/

Список литературы

[1] Bell Steven, Troccoli Alejandro, Pulli Kari. A Non-Linear Filter for Gyroscope-Based Video Stabilization. — 2014.

[2] Bundled Camera Paths for Video Stabilization / Shuaicheng Liu, Lu Yuan, Ping Tan, Jian Sun // ACM Trans. Graph. — 2013. — . — Vol. 32, no. 4. — P. 78:1–78:10. — URL: http://doi.acm.org/10.1145/2461912.2461995.

[3] CodingFlow: Enable Video Coding for Video Stabilization / Shuaicheng Liu, Mingyu Li, Shuyuan Zhu, Bing Zeng. — 2017. — 04. — Vol. 26. — P. 1–1.

[4] Diebel James. Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors. — 2006.

[5] Feature-based real-time video stabilization for vehicle video recorder system / Wu-Chih Hu, Chao-Ho Chen, Yi-Jen Su, Tzu-Hsing Chang // Multimedia Tools and Applications. — 2018. — Mar. — Vol. 77, no. 5. — P. 5107–5127. — URL: https://doi.org/10.1007/s11042-017-4369-7.

[6] Forssén P. E., Ringaby E. Rectifying rolling shutter video from hand-held devices // 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. — 2010. — June. — P. 507–514.

[7] Full-frame video stabilization with motion inpainting / Y. Matsushita, E. Ofek, Weina Ge et al. // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2006. — July. — Vol. 28, no. 7. — P. 1150–1163.

[8] Fusion of Unsynchronized Optical Tracker and Inertial Sensor in EKF Framework for In-car Augmented Reality Delay Reduction / J. Rambach, A. Pagani, S. Lampe et al. // 2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct). — 2017. — Oct. — P. 109–114.

[9] Grundmann M., Kwatra V., Essa I. Auto-Directed Video Stabilization with Robust L1 Optimal Camera Paths // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). — 2011.

[10] Karami Ebrahim, Prasad Siva, Shehata Mohamed S. Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images // CoRR. — 2017. — Vol. abs/1710.02726. — 1710.02726.

[11] Karpenko Alexandre, Jacobs David, Baek Jongmin. Digital Video Stabilization and Rolling Shutter Correction using Gyroscopes. — 2011.

[12] Lowe David G. Distinctive Image Features from Scale-Invariant Keypoints // Int. J. Comput. Vision. — 2004. — . — Vol. 60, no. 2. — P. 91–110. — URL: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[13] Mahony R., Hamel T., Pflimlin J. M. Complementary filter design on the special orthogonal group SO(3) // Proceedings of the 44th IEEE Conference on Decision and Control. — 2005. — Dec. — P. 1477–1484.

[14] ORB: An efficient alternative to SIFT or SURF / E. Rublee, V. Rabaud, K. Konolige, G. Bradski // 2011 International Conference on Computer Vision. — 2011. — Nov. — P. 2564–2571.

[15] Spatially and Temporally Optimized Video Stabilization / Y. S. Wang, F. Liu, P. S. Hsu, T. Y. Lee // IEEE Transactions on Visualization and Computer Graphics. — 2013. — Aug. — Vol. 19, no. 8. — P. 1354–1361.

[16] Szeliski Richard. Computer Vision: Algorithms and Applications. — 2010.

[17] Zhao Z., Ma X. Subspace video stabilization based on matrix transformation and Bezier curve // 2016 Eighth International Conference on Advanced Computational Intelligence (ICACI). — 2016. — Feb. — P. 270–274.