Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем
Кафедра системного программирования

Корнилова Анастасия Валерьевна

Цифровая стабилизация видеопотока с использованием MEMS–датчиков

Курсовая работа

Научный руководитель:
ст. преп. Я. А. Кириленко

Санкт-Петербург
2017

Оглавление

Введение

Матрицы современных камер обеспечивают хорошее качество отдельных кадров, сравнимое с профессиональными фотографиями, в то время как качество видео оставляет желает лучшего. Повышение качества видео не только позволяет владельцам смартфонов и экшн-камер получить красивые любительские ролики, но и решает более существенные проблемы. Например, подобное улучшение увеличивает точность дистанционного управления мобильными роботами и дронами (квадрокоптерами), осуществляющими мониторинг территорий, а также снижает усталость оператора.

В большинстве случаев для решения этой проблемы достаточно избавиться от тряски камеры, на которую производится запись. Этого можно добиться двумя способами: зафиксировать камеру (или по возможности гасить движения камеры засчет специальных механизмов) или преобразовывать кадры таким образом, чтобы изображение на нем оставалось статичным. Первый подход требует наличия внешнего устройства стабилизации, например, стабилизирующего подвеса (в случае квадрокоптеров) или «плавающих» систем линз и матриц, которые присутствуют в современных фотоаппаратах.

Основной задачей при втором подходе — цифровой стабилизации — является определение смещения камеры и преобразование кадров (рис. 1). Разработки в этом направлении активно ведутся производителями программного обеспечения по обработке видео и видеомонтажу и уже нашли применение в таких продуктах как Adobe Premier, Deshaker, Movavi. Функциональность по стабилизации видео также предлагает сервис YouTube, который использует алгоритм, предложенный в работе [7]. Но используемые алгоритмы [6], [12], [10], [2] задействуют лишь информацию с кадров, в связи с чем не могут гарантировать хорошее качество работы при плохом освещении и наличии больших движущихся объектов на кадре.

Альтернативный способ определить смещение камеры при съемке — информация с MEMS-датчиков движения (MicroElectroMechanical

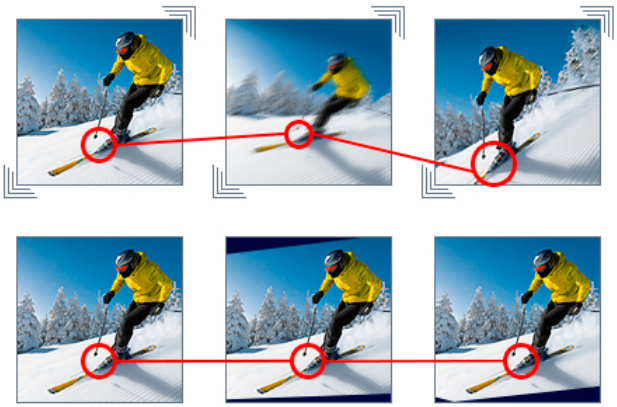

Рис. 1: Преобразование кадров для выравнивания траектории

Systems), к которым следует отнести датчик угловых скоростей (гироскоп), акселерометр, магнетометр. В таком случае затраты вычислительных мощностей и, как следствие, потребляемой электроэнергии на вычисление позиционирования камеры значительно меньше, чем при анализе кадров. Данный подход все больше находит свое применение в последние годы, когда наличие MEMS-датчиков на платформах стало стандартным, особенно в случае смартфонов и контроллеров. Например, в октябре 2016 года был представлен смартфон Google Pixel, в котором полностью отсутствует механическая стабилизация и используется алгоритм стабилизации по гироскопу. Смартфон iPhone 7 также использует стабилизацию по MEMS-датчикам в связке со стабилизацией по линзам и матрице.

Подход к стабилизации видеоизображения с использованием информации с MEMS-датчиков только начинает зарождаться, о чем свидетельствует сравнительно небольшое количество научных групп и публикаций по этой теме, а использование данного подхода ведущими компаниями говорит об актуальности обозначенной проблемы и наличии широкого пространства для ведения научных исследований.

Интерес, вызванный отсутвием общедоступных работающих решений в данной области, стал основой для проведения исследований, описанных в данной курсовой работе.

1. Постановка задачи

Целью данной курсовой работы является создание прототипа модуля для стабилизации видеоизображения с использованием информации с MEMS-датчиков. Для достижения цели были поставлены следующие задачи.

1. Сделать обзор существующих алгоритмов, выполняющих стабилизацию видео с использованием MEMS-датчиков.

2. Выработать стратегию сглаживания траектории движения камеры.

3. Разработать систему параллельной записи кадров видео и показаний MEMS-датчиков.

4. Реализовать модуль, определяющий положение камеры и траекторию ее движения по показаниям MEMS-датчиков.

5. Реализовать модуль, выполняющий преобразование кадра при заданных параметрах поворота камеры.

6. Интегрировать все модули в приложение-прототип стабилизации видео.

2. Обзор существующих решений

Данный раздел посвящен описанию существующих подходов по стабилизации видео с использованием MEMS-датчиков, также рассмотрены основные недостатки и преимущества данных подходов.

2.1. Алгоритм с использованием фильтра Гаусса

В основе алгоритма, представленного в статье [8] в 2011 году, лежит использование фильтра Гаусса[1]. 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. 2) и по новой модели движения строится поток кадров, преобразованный согласно 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.

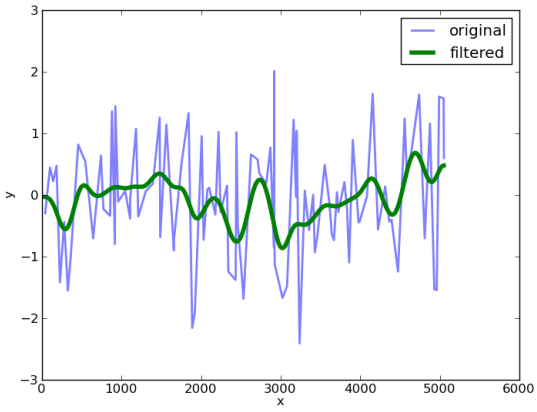

Рис. 2: Сглаживание траектории при помощи фильтра Гаусса

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

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

в горизонтальной плоскости, что не всегда соответствует поведению камеры при тряске.

2.2. 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. 3). Во внутренней области выбирается прямоугольник, в зависимости от положения которого принимается решение о положении виртуальной камеры.

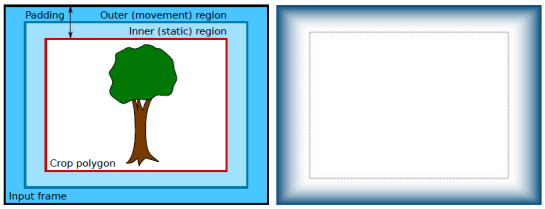

Рис. 3: Внешняя и внутренняя зоны стабилизации

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[2](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 бу-

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

фер 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. Стабилизация видео

Алгоритмы, производящие цифровую стабилизацию видео, состоят из трех основных этапов:

1) определение позиционирования камеры по MEMS-датчикам;
2) вычисление желаемого положения камеры по некоторой логике, сглаживание траектории;
3) трансформация кадра таким образом, чтобы положение камеры удовлетворяло желаемому.

Главным различием среди рассматриваемых алгоритмов является стратегия сглаживания траектории (второй этап). Так, одни решения предлагают только избавление от мелких шумов, в то время как другие используют более сложный анализ движения камеры, стараясь найти наиболее естественный вариант движения кадра для человеческого глаза.

3.1. Математическая модель движения камеры

Описывая математическую модель камеры, предполагается, что в малой временной окрестности основное движение камеры — вращательное движение. Аргументом к данному предположению является результаты исследований, описанные в работе [5], где утверждается, что модель, которая описывает только вращательные движения, работает лучше, чем та, которая дополнительно рассматривает смещения. Поставленные эксперименты подтверждают, что данного предположения достаточно.

Рассмотрим процесс преобразования кадра при повороте камеры. Обозначим за x координаты точки на проективной плоскости, а за X — координаты точки в пространстве (рис. 4). Также для каждой конкретной камеры зададим соответствующую ей матрицу K с параметрами (o_x, o_y) — оптический центр камеры, f — фокусное расстояние. Получим формулы для проективного преобразования[11]:

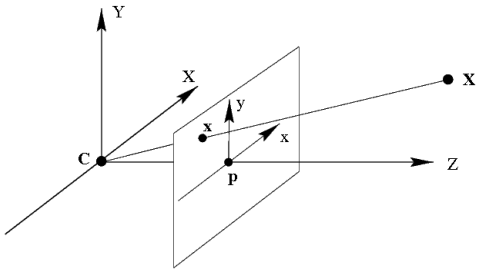

Рис. 4: Проективное преобразование

$$x = KX \tag{1}$$

$$K^{-1} = \begin{pmatrix} 1 & 0 & -o_x \\ 0 & 1 & -o_y \\ 0 & 0 & f \end{pmatrix} \tag{2}$$

Зафиксируем глобальную систему координат и предположим, что в момент времени t камера повернута относительно нее при помощи матрицы поворота $R(t)$[3] (рис. 5). Тогда проективное преобразование примет вид:

$$x = KR(t)X \tag{3}$$

Предположим, что x_i и x_j — проекции одной и той же точки X в простанстве, которая находится на кадрах i и j соответственно:

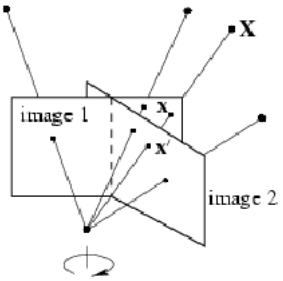

Рис. 5: Расположение точки на кадрах при повороте камеры

$$x_i = KR(t_i)X \tag{4}$$

$$x_j = KR(t_j)X \tag{5}$$

[3] https://en.wikipedia.org/wiki/Rotation_group_SO(3)

Преобразовав эти выражения, можно получить следующую связь между проекциями одной и той же точки в разные моменты времени:

$$x_j = KR(t_j)R^T(t_i)K^{-1}x_i \tag{6}$$

Таким образом, введем понятие матрицы трансформации изображения между моментами времени t_1 и t_2:

$$W(t_1, t_2) = KR(t_1)R^T(t_2)K^{-1} \tag{7}$$

$$x_j = W(t_j, t_i)x_i \tag{8}$$

3.2. Сглаживание траектории

В рамках данной работы в качества основной стратегии сглаживания траектории был применен фильтр Гаусса. Для сглаживания локальной тряски с помощью встроенных функций OpenCV экспериментально были подобраны следующие параметры фильтра: ширина окна — 256, размер ядра — 4000.

3.3. Преобразование кадров согласно движению камеры

Для проективного преобразования координат точки в OpenCV существует встроенная функция perspectiveTransform, которая позволяет узнать новое положение точки кадра при известной матрице поворота камеры. С использованием данной функции был разработан модуль, позволяющий трансформировать кадр согласно матрице поворота.

Алгоритм преобразования кадра состоит в следующем. На кадр накладывается сетка размером KxK. Узлы сетки являются опорными точками, для которых с помощью perspectiveTransform высчитывается новое положение. Остальные точки кадры преобразуются с помощью двумерной линейной интерполяции относительно особых точек.

Экспериментальным путем был подобран оптимальный размер сетки , при которых изображение остается неразмытым. Для кадров с раз-

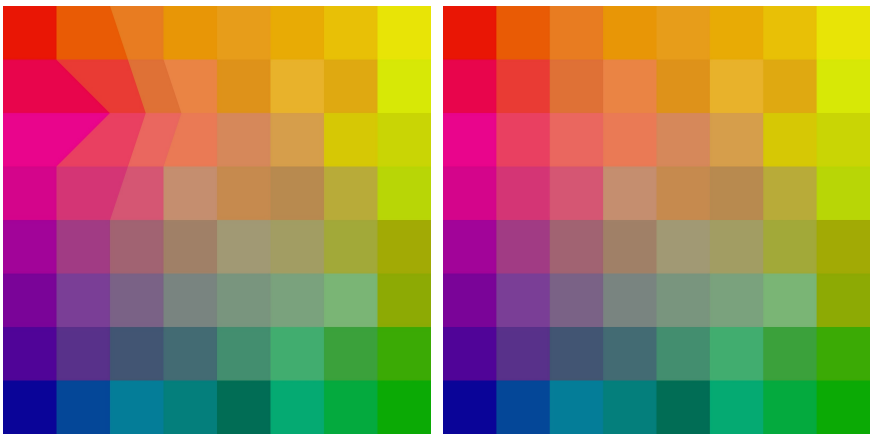

Рис. 6: Пример работы алгоритма интерполяции

решением 1920x1280 данное значение K составляет 10. На рис. 6 показаны результаты работы алгоритма интерполяции. На рис. 7 приведен пример проективного преобразования кадра при повороте камеры на 10 градусов.

Рис. 7: Проективный поворот кадра

4. Система записи кадров и показаний датчиков

В данной секции приводится описание системы параллельной записи кадров и показаний датчиков, а также рассматриваются основные аспекты ее реализации.

4.1. Выбор платформы

В качестве платформы для создания системы параллельной записи кадров и показаний датчиков было принято решение использовать операционную систему Android. Данное решение обусловлено не только сравнительной простотой программирования под эту ОС, но и ее доступностью, что позволяет производить запись тестовых данных с разных устройств. Кроме того, наличие большой базы приложений по записи видео с открытым исходным кодом позволило не разрабатывать приложение «с нуля», а лишь расширить существующую функциональность.

4.2. Событийная схема для камеры

Основной проблемой при программировании данного приложения стала реализации событийной схемы для кадров, получаемых с камеры. Основное API камеры позволяло записывать видео, не обозначая при этом конкретное время получения кадра в абсолютной системе счисления времени устройства. В случае использования данного API был доступен видеофайл с временными метками кадров относительно начала записи, а также показания гироскопа в абсолютной системе счисления. В связи с этим было необходимо использовать математические методы сопоставления временных рядов кадров и показаний датчиков (рис. 8). Особую сложность вызывала большая разница в частоте получения кадров (30 fps) и показаний датчиков (250 Hz).

Начиная с Android API level 21 (Android 4.4 и выше) дан-

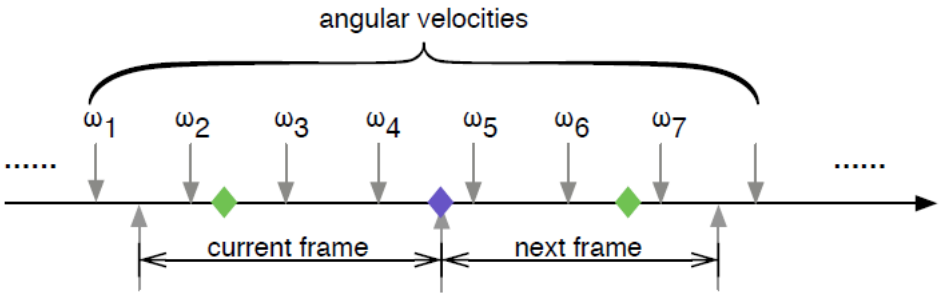

Рис. 8: Сопоставление временных рядов

ная проблема была решена в обновленном API для камеры — Camera2[3]. Это API поддерживает событийную схему, которая позволяет определить захват нового кадра посредством обработчика событий OnImageAvailableListener.

В качестве приложения по записи видео, на базе которого реализовывалась система, было выбрано приложение-прототип от Google[13], которое поддерживает вышеописанное API для камеры.

4.3. Регистрация показаний

Получение информации о новом кадре или новом показании датчика фиксируется в StringBuffer в формате $X; Y; Z; timestamp$ для показания датчика и f для кадра (рис. 9), что позволяет восстановить последовательность получения информации. В результате работы данного приложения получается два файла: видеофайл с расширением .mp4 и файл с данными с расширением .csv.

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	-0.0660404	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

Рис. 9: Формат файла синхронизации

5. Позиционирование камеры

Алгоритмы стабилизации требуют точного определения положения камеры в пространстве, в частности, ее матрицы поворота. В данном разделе рассмотрены решения данной проблемы, предложенные в рамках курсовой работы.

5.1. Гироскоп

В качестве основного датчика был выбран датчик угловых скоростей (гироскоп), так как интегрирование его показаний позволяет получить относительный поворот камеры в углах Эйлера. Помимо гироскопа также возможно использования показаний акселерометра — датчика, который показывает проекцию вектора гравитации на три оси. Но данный датчик не чувствителен к вращению в горизонтальной плоскости, поэтому не может покрыть трехмерную модель вращения камеры.

При реализации прототипа алгоритма рассматривались случаи небольших пространственных смещений камеры, поэтому достаточно линейного интегрирования угловых скоростей:

$$\theta(t+\delta) = \theta(t) + \int_{t}^{t+\delta} \omega(t)dt, \tag{9}$$

где θ - угол поворота по одной из осей, а ω - скорость поворота по этой же оси.

В случае интенсивного движения камеры необходимо использовать более сложное интегрирование, в частности, с использованием матриц повороты или кватернионов [4].

5.2. Уточнение показаний

К сожалению, не всегда на платформе установлен гироскоп с достаточной точностью. Кроме того необходимо отметить, что интегрирование показаний гироскопа влечет увеличение ошибки, в связи с чем у показаний углов растет смещение. Для избежания данной проблемы

возможно использование показаний других датчиков, например, акселерометра.

Наиболее простым решением в данном случае является применения комплементарного фильтра[9], который не позволяет показаниям гироскопа «уплывать».

Результаты

В рамках данной курсовой работы были выполенены следующие задачи.

1. Составлен обзор существующих решений по стабилизации видеоизображения с использованием MEMS-датчиков.
2. Разработана математическая модель движения камеры.
3. Реализована система сбора кадров видео, синхронизированных с показаниями датчиков на базе ОС Android.
4. Разработан модуль проективного преобразования кадра с использованием OpenCV.
5. Разработан модуль определения поворота камеры по показаниям MEMS-датчиков.
6. Реализован прототип алгоритма стабилизации с фильтром Гаусса, основанный на базе вышеописанных модулей.

Исходный код приложения по синхронизированной записи кадров видео и показаний MEMS-датчиков доступен по ссылке: `https://github.com/Korniluk13/VideoSensorApi2`. Исходный код модуля стабилизации видео доступен по ссылке: `https://github.com/Korniluk13/VideoStab`.

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. — Jul. — Vol. 32, no. 4. — P. 78:1–78:10. — Access mode: `http://doi.acm.org/10.1145/2461912.2461995`.

[3] Camera2 Android API. — Access mode: `https://developer.android.com/reference/android/hardware/camera2/package-summary.html` (online; accessed: 23.05.2017).

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

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

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

[12] 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.

[13] Приложение Camera2Video. — Access mode: `https://github.com/googlesamples/android-Camera2Video` (online; accessed: 23.05.2017).