Санкт-Петербургский государственный университет

Кафедра системного программирования

Ахметьянов Азат Ришатович

Точная синхронизация записи видео на Android-смартфонах

Отчет о прохождении производственной практики

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

Консультант:
Инженер-исследователь, Сколтех А. В. Корнилова

Санкт-Петербург
2021

Оглавление

Введение

Современные смартфоны оснащены большим количеством сенсоров таких как IMU (акселерометр, гироскоп, компас), камера или набор камер, датчики температуры, GPS, обладают производительными вычислителями для обработки информации с данных сенсоров, а также способны организовываться в распределенную сеть. Это, наряду с доступностью и распространенностью смартфонов, делает их достойной альтернативой узкоспециализированным устройствам в различных областях жизнедеятельности человека. В медицине и здравоохранении активно используются IMU/GPS для анализа активности человека в течение дня [20], камера может быть применена для оценки пульса [33], а также для анализа заболеваний глаз [7] и респираторных заболеваний [14]. В области безопасности жизнедеятельности распределенная сеть смартфонов со встроенными IMU может применяться для оценки вибраций в здании и предупреждении о землетрясении [30], распределенные GPS датчики могут быть использованы для анализа столпотворений и дорожного трафика [16].

Более продвинутым классом задач, одновременно использующим фотографии и видеозаписи с камер, являются задачи одометрии [28], SfM (Structure from Motion) [27] и SLAM (Simultaneous Localization And Mapping) [26], которые могут эффективно применяться в приложениях по трехмерной реконструкции, дополненной реальности. Несмотря на существующие алгоритмы для решения данных задач с использованием монокулярного зрения (одной камеры) [10, 15, 11], качество решений может быть значительно улучшено с использованием распределенной системы камер [19] для применения алгоритмов стереозрения. Несмотря на то что современные модели смартфонов иногда имеют встроенные стереокамеры, расстояние между камерами (baseline), установленными на одном устройстве, достаточно мало для восстановления качественной карты глубины [9] по сравнению с системами камер с большим baseline-ом. По этой причине имеет смысл рассматривать сеть из смартфонов, камеры которых используются совместно.

Важно отметить, что в успешной реконструкции динамичных сцен с использованием алгоритмов стереозрения важную роль играет синхронизация камер. Разница в несколько миллисекунд между кадрами стереопары создает проблемы при построении карты глубин, как показано на Рис. 1. Таким образом, далее «точной синхронизацией» в контексте синхронизации съемки камер будем называть синхронизацию с точностью менее миллисекунды.

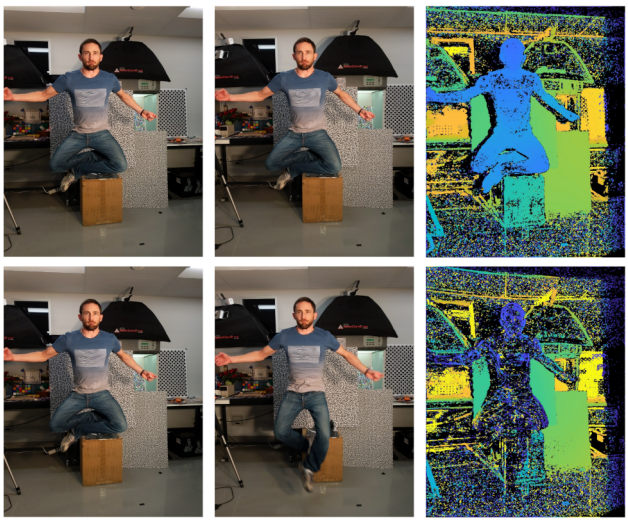

Рис. 1: Верхний ряд: стереопара из кадров с камер со смещением менее миллисекунды и полученная из них карта глубин. Нижний ряд: стереопара из кадров с камер со смещением в 66 миллисекунд и полученная карта глубин. Рис. из статьи [29].

Проблема синхронизации съемки на смартфонах состоит из двух основных частей: точная синхронизация времени смартфонов и программная синхронизация самого процесса съемки.

Синхронизация времени устройств заключается в оценке относительного смещения их локальных часов.

Точная синхронизация съемки на Android-смартфонах представляет собой отдельную проблему, так как время между запросом съемки и экспозицией сенсора непредсказуемо и может колебаться в пределах нескольких миллисекунд. Частично эта проблема решается существующими приложениями, такими как libsoftwaresync [29], SocialSync [18]. Основной подход, который применяется в данных работах, заключается в том, что периодичные потоки кадров с камер смартфонов можно

сдвигать и выравнивать между собой, добиваясь синхронизации. Однако, например, в libsoftwaresync [29] от Google Research такое выравнивание не автоматизировано и работает с конкретным устройством, требуя изменения конфигурационных значений при сборке приложения под конкретное устройство. Также в данных работах не поддерживается запись синхронизированных видео, поскольку такая возможность не была исследована. Главным образом в проблеме расширения подхода с выравниванием фаз потоков кадров на видеосъемку представляет интерес то, насколько стабильной остается фаза после смены режимов камеры с предпросмотра на видео. Поддержка синхронизированной видеозаписи важна, поскольку процессоры смартфонов применяют кодеки для сжатия видео (например, H265), благодаря которым появляется возможность записать значительно больше информации, чем при сохранении отдельных фотографий из предпросмотра камеры.

Таким образом, для постановки экспериментов с алгоритмами в рассмотренных областях исследований существует потребность в инструменте для записи синхронизированных данных с камер смартфонов, используемых совместно. Существующие решения имеют определенные недостатки, поэтому в данной работе пойдет речь о создании нового инструмента для решения задачи.

1. Постановка задачи

Целью работы является создание Android-приложения для записи синхронизированных видео с нескольких смартфонов и дальнейшего использования в перечисленных областях применения.

Для достижения цели поставлены следующие задачи.

- Разработка алгоритма для автоматизированной оценки периода временных меток кадров.
- Анализ стабильности фазы временных меток кадров при переключении режимов камеры (предпросмотр и видео).
- Реализация записи синхронизированного видео в приложении на платформе Android.
- Оценка качества синхронизации видео с использованием разработанного приложения.
- Апробация на склейке панорамного видео.

1.1. Требования к приложению для записи

К разрабатываемому приложению поставлены требования:

- коммуникация устройств, возможность начать или остановить запись по сети с одного устройства;
- использование расширенных настроек камеры, таких как время экспозиции, ISO;
- автоматизация подсчета конфигурационных значений, необходимых для выравнивания фазы потоков кадров, под конкретное устройство;
- наличие вспомогательных скриптов для проверки и постобработки полученных данных, а именно (а) разделения видео на кадры и

сопоставления им временных меток, (б) генерации списка полноценных пар кадров, т.е. таких, что в окрестности временной метки кадра с одного смартфона существует и кадр с другого (кадры могут ”проваливаться”) [23].

2. Обзор

2.1. Особенности работы с камерой в Android

2.1.1. Rolling Shutter

Камеры смартфонов применяют метод rolling shutter [25], в ходе которого изображение считывается сенсором не целостно, а построчно. Иногда это приводит к тому, что при съемке быстро движущегося объекта на сохраненном изображении заметны артефакты.

2.1.2. Camera2API

Camera2API [31] — расширенный API для управления камерой в ОС Android, который поддерживает ручное изменение различных параметров, таких как ISO, время экспозиции. Также API содержит функциональность, позволяющую получать точную временную метку того момента, когда был сделан снимок . Временная метка возвращается в наносекундах, и разработчики гарантируют, что данная метка в том же домене, что и предоставляемое ОС наносекундное время с загрузки системы.

Синхронизацию съемки делает нетривиальной задачей то, что система Android и предоставляемый разработчику API работы с камерой не позволяет запланировать съемку на достаточно точное время (с ошибкой менее миллисекунды). Даже если необходимые методы на разных смартфонах будут вызваны почти одновременно, подготовка к съемке занимает недетерминированное время и может привести к такой ошибке.

2.2. Синхронизация времени смартфонов

Сложность точной синхронизации времени нескольких смартфонов заключается а) в использовании смартфонами беспроводных сетей с нестабильной задержкой передачи сообщений, б) в разнообразии уровней поддерживаемых API операционной системы Android. Методы синхро-

низации времени можно разделить на сетевые протоколы, алгоритмы с использованием данных периферийных устройств, а также аппаратную синхронизацию. Аппаратная синхронизация с помощью триггеров требует специального оборудования и по этой причине не применима на смартфонах. Таким образом, остановимся только на первых двух видах синхронизации.

Сетевые протоколы основаны на анализе времени доставки или получения пакетов между узлами в сети. Методы, использующие периферийные устройства, зависят от данных, собранных с этих устройств. Зачастую такие методы эксплуатируют какие-либо общие события из окружающей среды, например, вспышки света в кадре при использовании камер [17].

2.2.1. Сетевые протоколы

Сетевые протоколы синхронизации времени можно разделить на следующие категории по принципу взаимодействия узлов сети.

Sender-Receiver протоколы. Отправитель посылает другому узлу временную метку, и узел пытается вычислить свое смещение относительно отправителя, оценивая задержки передачи сообщений. В основе обмена сообщениями для оценки задержки в одну сторону лежит предположение о том, что задержки в обе стороны почти одинаковы (предположение о симметрии задержек).

В данной категории рассмотрим протоколы NTP и PTP, поскольку эти протоколы имеют реализации на большом количестве платформ [1, 32, 12].

- Network Time Protocol (NTP) [8] — наиболее распространенный из сетевых протоколов, предназначен для синхронизации времени в глобальной сети. Одна из особенностей заключается в разделении источников времени на слои (так называемые «страты») по уровню их точности.

- Precision Time Protocol (PTP) [6] — одним из главных преимуществ является использование аппаратных временных меток на всех сетевых устройствах (хаб, маршрутизатор), что минимизирует ошибки, связанные с задержками сообщений. С учетом использования мобильных устройств это преимущество нивелируется, так как нет возможности реализовать физический уровень сети описанным образом.

Ошибка синхронизации в таких алгоритмах зависит от качества канала передачи данных, а именно от стабильности задержек и их симметрии.

Receiver-Receiver протоколы. Работа Receiver-Receiver протоколов основана на том факте, что широковещательные (broadcast) сообщения приходят на узлы почти одновременно. Отправитель посылает остальным узлам broadcast сообщение, при этом получатели используют только время прибытия пакета в своих локальных часах и производят синхронизацию друг с другом, не учитывая время отправителя. Такой подход хорошо подходит для синхронизации в беспроводной сети, так как он минимизирует эффект недетерминизма в задержке передачи сообщения.

Основным недостатком такого подхода является то, что в общем случае, без применения оптимизаций, он использует большое количество сообщений. Всем Receiver-узлам, получившим широковещательное сообщение, необходимо сообщить свое время либо всем остальным узлам, либо одному Receiver-узлу, который отвечает за вычисление смещений.

- Reference Broadcast Synchronization (RBS) [2] — Receiver-Receiver алгоритм синхронизации, предназначенный для использования в беспроводных сетях.

2.2.2. Методы, использующие периферийные устройства

Рассмотрим теперь методы синхронизации, основанные на анализе данных с периферийных устройств смартфона, таких как датчики IMU, камера, микрофон. Главное преимущество этих методов перед сетевыми протоколами в том, что ошибка синхронизации не зависит от качества сети. Их можно классифицировать по виду применяемых периферийных устройств.

Синхронизация с помощью данных с камер. Одним из методов точной синхронизации времени видеопотоков нескольких устройств является метод, основанный на анализе вспышек света на видео с камер, использующих Rolling Shutter [17].

Основная идея заключается в том, что, анализируя кадр камеры с эффектом Rolling Shutter и быстрым изменением освещенности, которое вызвано, к примеру, вспышкой, можно найти границу рядов, освещенных вспышкой, и не освещенных ей рядов. Этого достаточно, чтобы достаточно точно посчитать приблизительное временное смещение между кадрами.

Синхронизация с помощью датчиков IMU. Метод был предложен в работе лаборатории Мобильной Робототехники Twist-n-Sync [22]. Данный метод основывается на том, что, в случае жестко сцепленных устройств (смартфонов, роботов), их датчики гироскопа и акселерометра фиксируют общее движение. В таком случае данные датчиков можно сопоставлять при помощи кросс-корреляции и находить смещение между временем устройств.

Достоинством синхронизации с использованием IMU является то, что большинство смартфонов имеют встроенные гироскопы и акселерометры. Ограничения метода:

- нагрев устройства может негативно влиять на качество и стабильность частоты данных гироскопа или акселерометра;
- требуется жесткая сцепка устройств во время синхронизации.

2.3. Решения для синхронизированной записи данных камер и сенсоров смартфонов

Перейдем к рассмотрению существующих приложений, предназначенных для синхронизированной записи данных с камер и сенсоров смартфонов.

2.3.1. libsoftwaresync

Исследовательская группа в составе Google Research выпустила приложение с открытым исходным кодом libsoftwaresync [29].

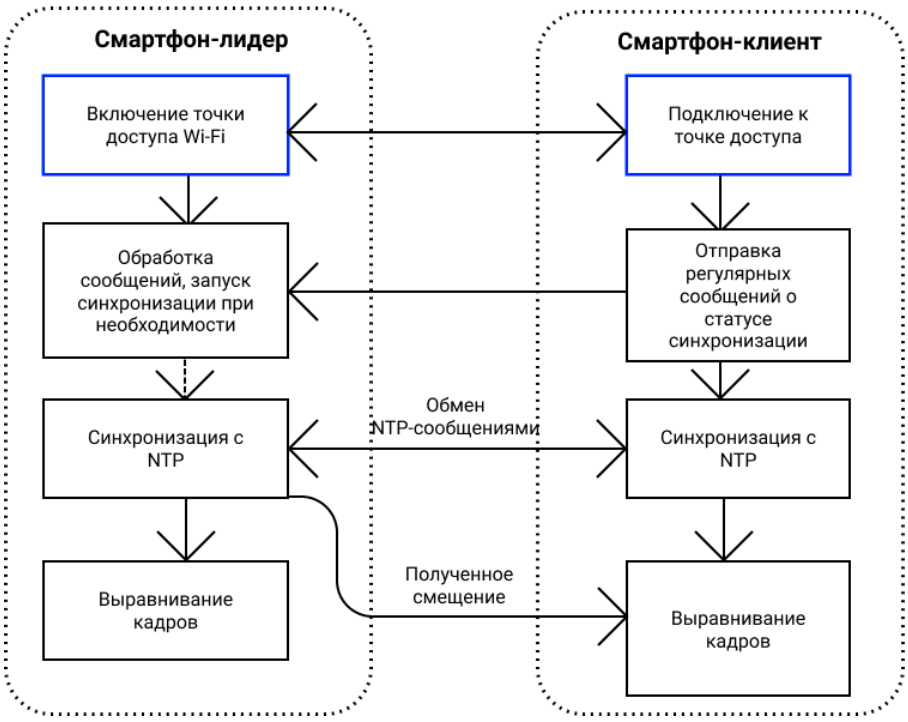

Рис. 2: Процесс съемки в приложении.

Объединив смартфоны в одну Wi-Fi сеть, с помощью этого приложения можно снимать синхронизированные фотографии на нескольких устройствах.

Принцип синхронизации заключается в том, что смартфоны запрашивают повторяющуюся съемку с фиксированной частотой, а затем выравнивают свою фазу съемки относительно синхронизированных при помощи NTP временных меток кадров. Выравнивание фазы на смартфонах осуществляется вставкой запроса на кадр с несколько отличаю-

щейся продолжительностью, что приводит к смещению фазы повторяющейся съемки и позволяет построить оптимизационный процесс для сходимости к целевой фазе. Выравнивание фазы на устройствах, не имеющих данной функциональности, происходит за счет перезапуска повторяющейся съемки до тех пор, пока не будет достигнута целевая фаза. Затем, когда пользователь нажимает на кнопку снятия фото на центральном устройстве, на всех смартфонах сохраняются кадры с совпадающими (в пределах некоторой погрешности) временными метками.

Результаты, описанные в статье [29], показали перспективность предложенного метода синхронизации съемки (в экспериментах на Google Pixel смартфонах была получена точность менее миллисекунды). Такой точности удалось добиться, поскольку NTP применялся в локальной специально созданной Wi-Fi сети из смартфонов.

Однако, данное приложение было создано по большей части для демонстрации метода на смартфонах Google Pixel, и имеет следующие ограничения:

- константы, необходимые для выравнивания фазы съемки, специфичны для каждого устройства, и в приложении не реализовано их автоматическое вычисление (конфигурационный файл для своего устройства нужно создавать вручную);
- приложение не поддерживает запись синхронизированных видео, эта возможность не проверялась с предложенным методом выравнивания фазы.

2.3.2. SocialSync

SocialSync [18] — более ранняя работа, которая применяет похожий подход с выравниванием фазы кадров предпросмотра. Основное отличие от подхода libsoftwaresync заключается в том, что разработанное приложение полностью перезапускает предпросмотр, пока не будет достигнута целевая фаза, и не использует менее ресурсозатратную и более точную возможность вставки в поток кадра со слегка отличающимся време-

нем экспозиции. Таким образом, экспериментальный анализ в работе показал точность синхронизации порядка нескольких миллисекунд.

В данной работе также не исследуется возможность применения такого подхода для записи синхронизированных видео. Еще одним недостатком является отсутствие открытого исходного кода, позволяющего лучше понять технические детали реализации работы.

3. Анализ стабильности фазы кадров в видео на смартфонах

Основной проблемой в переходе к синхронизированной записи видео на Android-устройствах было то, что стабильность фазы кадров при переходе из предпросмотра в режим записи видео не была исследована. По этой причине одной из задач данной работы было провести экспериментальный анализ этой возможности.

Чтобы получить данные, по которым можно судить о достаточно большом количестве устройств, небходимо было автоматизировать процесс их записи и сбора с множества смартфонов. Для тестирования на большом количестве разных моделей хорошо подходит сервис для облачного тестирования Firebase [4], он и был использован в качестве площадки для проведения анализа.

3.1. Сбор датасета на физических устройствах из Firebase

Было реализовано приложение, которое записывает тестовое видео длительностью в 1 минуту и создает файл, содержащий временные метки потока кадров как до начала записи, так и после. Данное приложение было запущено на 52 устройствах из firebase, удовлетворяющих следующим требованиям:

- возможность использования Camera2API;
- уровень поддержки Camera2API $\geq$ Limited.

Затем, при помощи реализованных скриптов, файлы с временными метками для каждого протестированного устройства были массово выгружены из облачного хранилища Firebase. Рассмотрение графиков, отображающих разницу i-й и $(i-1)$-й временной метки показало, что все устройства показывают поведение, которое иллюстрируют представленные графики:

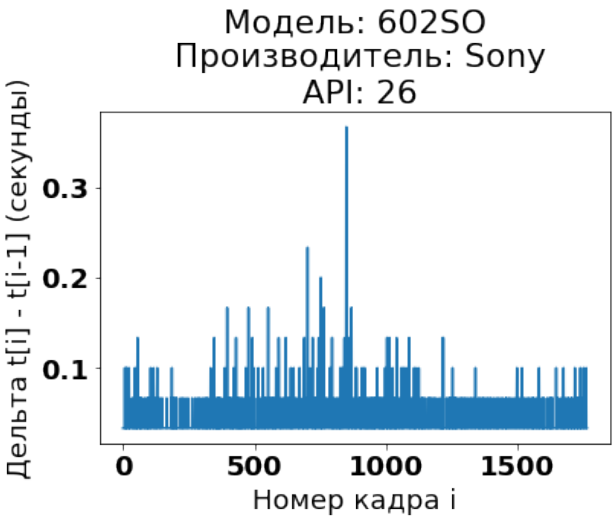

Рис. 3: Смартфон Sony.

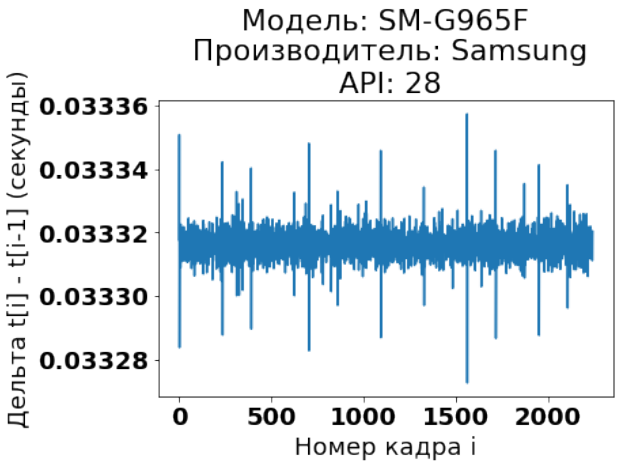

Рис. 4: Смартфон Samsung.

На первом графике заметны выбросы в разнице i-й и $(i-1)$-й временной метки, однако данные выбросы оказались кратны периоду потока кадров. Таким образом, они могут быть вызваны потерей некоторых кадров при съемке и не должны негативно влиять на общую фазу потока.

3.2. Математическая модель временных меток кадров

После анализа данных с устройств в Firebase, была предложена следующая математическая модель фазы временных меток (ϵ - погрешность, T - период временных меток). Такая модель учитывает возможность кратных периоду T выбросов, а также определяет погрешность ϵ, оценивая которую можно анализировать качество оценки периода.

$$t[i] = t[0] + N(i)T + \epsilon \tag{1}$$
$$N(i) \in \mathbb{N}$$
$$N(i) < N(i+1)$$

3.3. Алгоритм оценки периода кадров

Решаемая проблема минимизации погрешности в модели временных меток является задачей целочисленного программирования, то есть NP-трудна [24]. Поэтому, исходя из структуры выбросов (поскольку они кратны искомому периоду, их также стоит учитывать при оценке), предложено использовать метод с применением взвешенных кластеров:

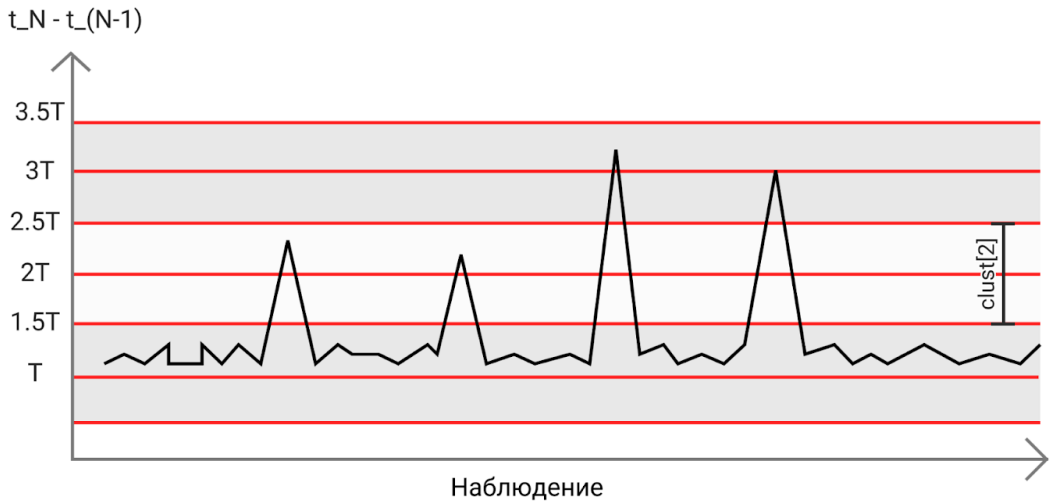

Рис. 5: Принцип работы метода со взвешенными кластерами.

$$T_{init} = \min_{i \in [2,..,N]} T_i - T_{i-1} \tag{2}$$

$$T = \frac{1}{N} \sum \frac{median(clust_i)}{i} \cdot |clust_i| \tag{3}$$

, где T_{init} необходимо для предварительного построения кластеров, а $clust_i$ представляет собой подмножество дельт временных меток из i-го кластера.

Такой метод предоставляет достаточно точную (относительно погрешности ϵ) оценку периода, потому что выбросы тоже учитываются при оценке, но при этом имеют меньший вес, зависящий от количества наблюдений в определенном кластере.

3.4. Результаты анализа

Анализ стабильности основан на предложенной математической модели временных меток кадров. Мера стабильности рассчитывалась по следующей формуле:

$std(t[i] - (t[0] + N(i)T))$ — фактически, данная формула выражает стандартное отклонение погрешности ϵ из предложенной модели.

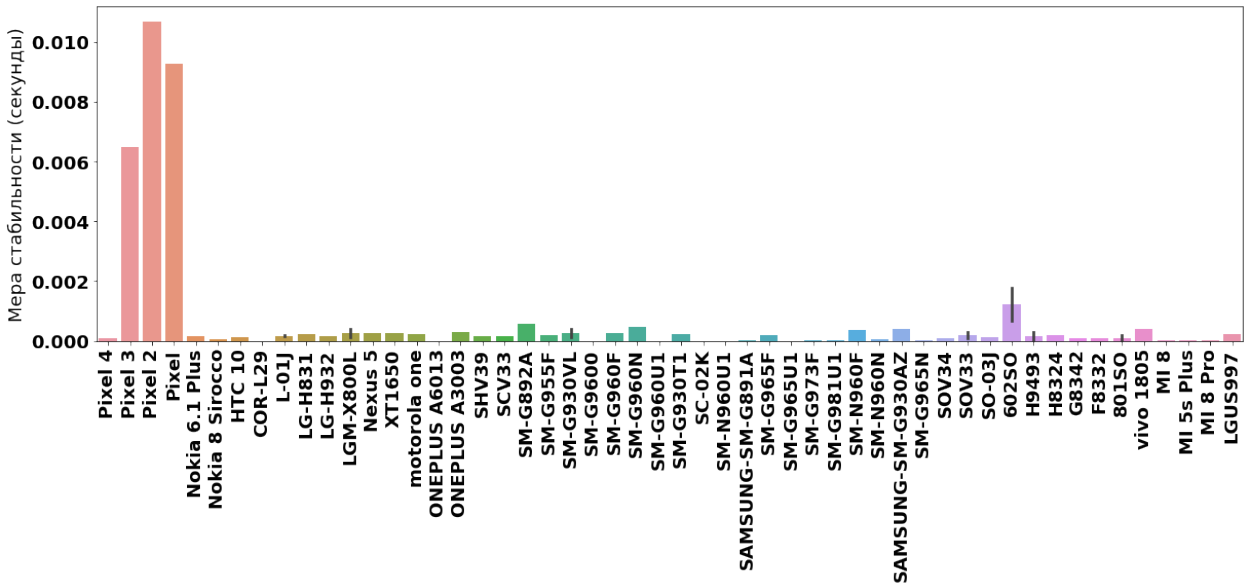

Рис. 6: График меры стабильности для набора данных из Firebase.

Анализ при помощи описанной меры показал, что на большинстве устройств за 1 минуту стандартное отклонение погрешности составляет менее 1 миллисекунды, что является приемлемым результатом в задаче синхронизации съемки.

Исключение составляет некоторое подмножество смартфонов Google Pixel, для которых, чтобы выявить возможные проблемы, необходим дополнительный анализ с физическими устройствами.

4. Реализация приложения для синхронизации записи видео с Android-смартфонов

Для реализации приложения было решено модифицировать подход Google Research, который достаточно успешно показал себя в задаче синхронизированной фотосъемки.

4.1. Общие детали реализации

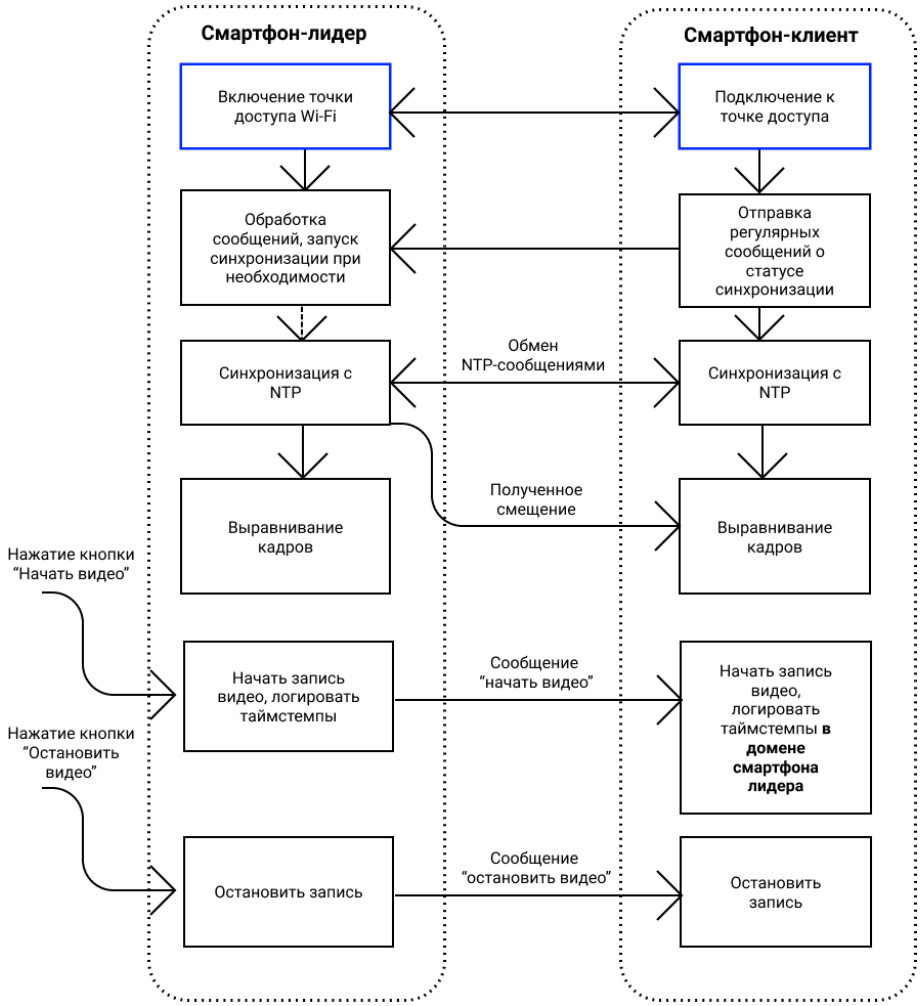

Рис. 7: Модифицированный процесс съемки в приложении.

В уже существующей системе коммуникации приложения введены новые типы сетевых сообщений: startVideo и stopVideo, которые запускают и останавливают видеосъемку на устройствах-клиентах. Также

было реализовано сохранение файлов, содержащих предоставляемые API временные метки кадров видео в наносекундах.

4.1.1. Восстановление соответствия между временными метками и кадрами видео

При наличии только файла с метками и готового видео, единственным способом восстановить соответствие между этими метками и конкретными кадрами из видео является прямое сопоставление кадров и записей с одинаковыми порядковыми номерами. Поэтому критическую важность имело то, что никакие кадры не входящие в видео (т.е. из предпросмотра камеры) не записываются в файл, а также в видео не содержится кадров, не учтенных в файле. Поскольку на платформе Android начало и завершение записи видео — асинхронные вызовы, чтобы достичь желаемого эффекта были расширены обработчики событий начала и завершения сессии съемки таким образом, что они также инициируют или останавливают запись в файл получаемых временных меток.

4.1.2. Вычисление конфигурационных значений

Одним из недостатков libsoftwaresync является ручная (на этапе сборки приложения) установка числового значения периода съемки под каждое устройство. Чтобы этого избежать, в модифицированном решении в качестве одного из предварительных шагов реализован подсчет этого значения методом с использованием взвешенных кластеров.

4.2. Экспериментальный анализ точности синхронизации при помощи вспышки

Модифицированное приложение использовалось на двух Samsung Galaxy S20, закрепленных на стенде, изображенном на Рис. 8.

Основные характеристики использованных устройств:

- Camera2API;

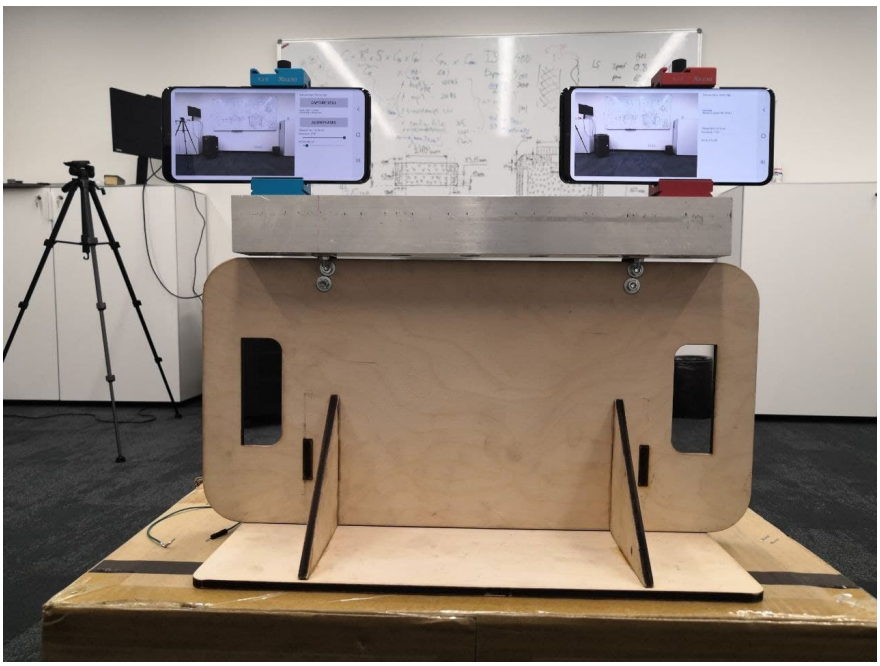

Рис. 8: Экспериментальный стенд.

- применялась стандартная задняя камера (в смартфоне есть также широкоугольная и фронтальная камеры);
- разрешение камеры: 12 МП.

Было записано 5 видео продолжительностью в 1 минуту.

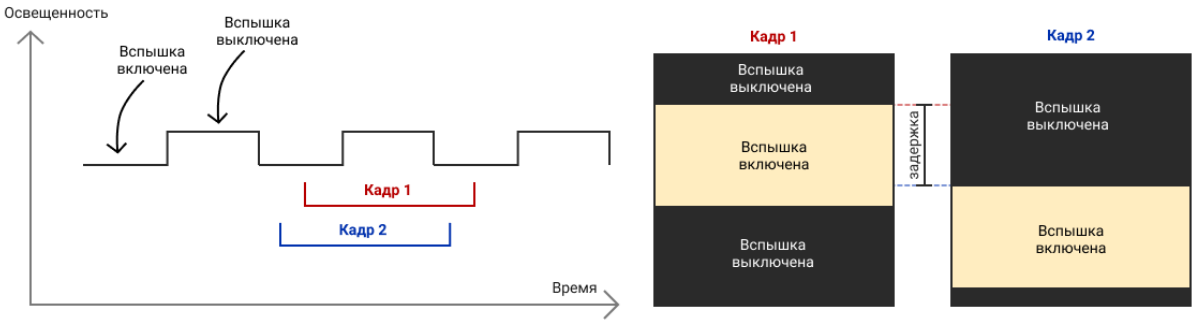

Рис. 9: Эксперимент с использованием вспышки.

Эксперимент основывался на эффекте Rolling Shutter. Быстрое изменение освещенности создавалось при помощи внешней вспышки, мигающей с частотой 5000 Гц. После этого, с помощью python-скрипта находилась граница освещенных и темных рядов. Поскольку эта граница на кадрах с разных устройств имеет различное положение, оценивалось ее смещение в рядах для каждого кадра записанных видео. Зная

приблизительное время считывания одного ряда (период вспышки поделенный на количество рядов в освещенной части), можно оценить задержку во времени между двумя кадрами, такая оценка представлена в таблице (10).

Эксперимент	Количество пар кадров	Ошибка синхронизации (мксек)
1	1098	67.07 ± 36.25
2	1183	58.55 ± 0.00
3	1167	116.99 ± 2.29
4	1156	58.55 ± 0.00
5	1054	144.92 ± 87.20

Рис. 10: Таблица временного смещения в микросекундах между кадрами для 5 пар видео. Доверительный интервал — 0.95.

Таким образом, был получен удовлетворительный результат (задержка между парами кадров во всех экспериментах не превышает значений порядка сотен микросекунд), который показывает перспективность не только синхронизации фотосъемки, но и записи синхронизированных видео.

4.3. Апробация на склейке панорамного видео

С использованием реализованного приложения производилась апробация решения на задаче склейки двух синхронизированных видео в панорамное.

Для этого применялся инструмент Hugin [5], который поддерживает интерфейс командной строки, что позволяет упростить автоматизацию склейки для большого количества кадров. Скрипт склейки действует следующим образом:

- обе видеозаписи разбиваются на кадры при помощи ffmpeg [3];
- при помощи python скрипта, сравнивающего временные метки кадров, находится соответствие между кадрами первого и второго видео, при этом лишние (такие, для которых нет пары, потому что

видеозаписи начинаются и завершаются в немного разное время) кадры выбрасываются;

- на каждой паре кадров запускается скрипт, производящий склейку двух картинок с использованием функциональности Hugin;
- склеенные кадры преобразуются обратно в видео в ffmpeg.

Рис. 11: Пример склейки синхронизированных кадров.

Рис. 12: Пример склейки кадров с задержкой в 33 миллисекунды.

В случае с панорамой точная синхронизация видеозаписей наиболее важна в сценах, где участвуют динамические объекты, поскольку быстрое движение может привести к тому, что на недостаточно точно синхронизированной паре кадров положение объекта будет заметно отличаться. Для демонстрации этого эффекта было записано видео с широким baseline, в котором человек прыгает, преодолевая границу между областями видимости камер двух смартфонов.

На изображениях (Рис. 12, Рис. 11) видно, что в случае специальной склейки несинхронизированных кадров, фигура движущегося человека склеивается с артефактами. Видео-демонстрация доступна на youtube [21].

5. Результаты

В рамках работы были выполнены следующие задачи.

- Разработан алгоритм для автоматизированной оценки периода временных меток кадров.
- Проведен анализ стабильности фазы временных меток кадров при переключении режимов камеры (предпросмотр и видео) на устройствах из Firebase, анализ показал перспективность использования метода синхронизации съемки для видеозаписи. В анализе участвовали 52 устройства, в результате большинство смартфонов продемонстрировали смещение фазы временных меток кадров за 1 минуту не более чем на 1 миллисекунду.
- Реализована запись синхронизированного видео в приложении на платформе Android на основе libsoftwaresync от Google Research. Исходный код доступен на Github [13].
- Выполнена оценка качества синхронизации видео с использованием разработанного приложения при помощи эксперимента с быстро мигающей вспышкой. Во всех 5 записанных экспериментах задержка между парами кадров не превышала значений порядка сотен микросекунд.
- Автоматизирована склейка панорамного видео, проведена апробация склейки на видео, записанных приложением.

В качестве продолжения данной работы возможно увеличение количества тестируемых устройств, записывающих синхронизированное видео, а также применение продвинутых алгоритмов 3D-реконструкции на записанных таким образом данных.

Список литературы

[1] Chronos: Time Synchronization According to the IEEE 1588 Standard. — Access mode: https://tsep.com/en/products/chronos/. Accessed 15-December-2020.

[2] Elson Jeremy, Girod Lewis, Estrin Deborah. Fine-Grained Network Time Synchronization Using Reference Broadcasts // SIGOPS Oper. Syst. Rev. — 2003. — Dec. — Vol. 36, no. SI. — P. 147–163. — Access mode: https://doi.org/10.1145/844128.844143.

[3] FFmpeg. — Access mode: https://www.ffmpeg.org/. Accessed 11-June-2021.

[4] Firebase Test Lab. — Access mode: https://firebase.google.com/docs/test-lab.

[5] Hugin - Panorama Stitcher. — Access mode: http://hugin.sourceforge.net/. Accessed 11-June-2021.

[6] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems // IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002). — 2008. — P. 1–300.

[7] Majumder Sumit, Deen M Jamal. Smartphone sensors for health monitoring and diagnosis // Sensors. — 2019. — Vol. 19, no. 9. — P. 2164.

[8] Mills D. RFC1305: Network Time Protocol (Version 3) Specification, Implementation. — 1992.

[9] Multi-view stereo. — Access mode: https://www.cs.unc.edu/~marc/tutorial/node112.html. Accessed 4-December-2020.

[10] Mur-Artal Raul, Montiel Jose Maria Martinez, Tardos Juan D. ORB-SLAM: a versatile and accurate monocular SLAM system // IEEE transactions on robotics. — 2015. — Vol. 31, no. 5. — P. 1147–1163.

[11] Mur-Artal Raul, Tardós Juan D. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras // IEEE Transactions on Robotics. — 2017. — Vol. 33, no. 5. — P. 1255–1262.

[12] NTP Implementations and Platforms. — Access mode: `http://www.ntp.org/ntpfaq/NTP-s-def-impl.htm`. Accessed 15-December-2020.

[13] RecSync-Android. — Access mode: `https://github.com/azaat/RecSync-android`. Accessed 11-June-2021.

[14] Sanyal S., Nundy K. K. Algorithms for Monitoring Heart Rate and Respiratory Rate From the Video of a User's Face // IEEE Journal of Translational Engineering in Health and Medicine. — 2018. — Vol. 6. — P. 1–11.

[15] Schonberger Johannes L, Frahm Jan-Michael. Structure-from-motion revisited // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. — 2016. — P. 4104–4113.

[16] Smartroad: Smartphone-based crowd sensing for traffic regulator detection and identification / Shaohan Hu, Lu Su, Hengchang Liu et al. // ACM Transactions on Sensor Networks (TOSN). — 2015. — Vol. 11, no. 4. — P. 1–27.

[17] Smid Matej, Matas Jiri. Rolling Shutter Camera Synchronization with Sub-millisecond Accuracy. — 2019. — 1902.11084.

[18] SocialSync: Sub-Frame Synchronization in a Smartphone Camera Network / Richard Latimer, Jason Holloway, Ashok Veeraraghavan, Ashutosh Sabharwal // Computer Vision - ECCV 2014 Workshops / Ed. by Lourdes Agapito, Michael M. Bronstein, Carsten Rother. — Cham : Springer International Publishing, 2015. — P. 561–575.

[19] Stereo vision based indoor/outdoor navigation for flying robots / Korbinian Schmid, Teodor Tomic, Felix Ruess et al. // 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems / IEEE. — 2013. — P. 3955–3962.

[20] A Survey on Activity Detection and Classification Using Wearable Sensors / M. Cornacchia, K. Ozcan, Y. Zheng, S. Velipasalar // IEEE Sensors Journal. — 2017. — Vol. 17, no. 2. — P. 386–403.

[21] Synchronized smartphone videos stitching demo. — Access mode: https://youtu.be/XWzhxAcMn1I. Accessed 11-June-2021.

[22] Twist-n-Sync: Software Clock Synchronization with Microseconds Accuracy Using MEMS-Gyroscopes / Marsel Faizullin, Anastasiia Kornilova, Azat Akhmetyanov, Gonzalo Ferrer // Sensors. — 2021. — Vol. 21, no. 1. — Access mode: https://www.mdpi.com/1424-8220/21/1/68.

[23] Understanding Android camera capture sessions and requests. — Access mode: https://medium.com/androiddevelopers/understanding-android-camera-capture-sessions-and-requests-4e54c Accessed 11-June-2021.

[24] Wikipedia. Integer programming — Wikipedia, The Free Encyclopedia. — Accessed 11-June-2021. Access mode: https://en.wikipedia.org/wiki/Integer_programming.

[25] Wikipedia. Rolling shutter — Wikipedia, The Free Encyclopedia. — Accessed 11-June-2021. Access mode: https://en.wikipedia.org/wiki/Rolling_shutter.

[26] Wikipedia. Simultaneous localization and mapping — Wikipedia, The Free Encyclopedia. — Accessed 7-December-2020. Access mode: https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping.

[27] Wikipedia. Structure from motion — Wikipedia, The Free Encyclopedia. — Accessed 4-December-2020. Access mode: https://en.wikipedia.org/wiki/Structure_from_motion.

[28] Wikipedia. Visual odometry — Wikipedia, The Free Encyclopedia. —

Accessed 5-December-2020. Access mode: https://en.wikipedia.org/wiki/Visual_odometry.

[29] Wireless Software Synchronization of Multiple Distributed Cameras / Sameer Ansari, Neal Wadhwa, Rahul Garg, Jiawen Chen // ICCP. — 2019.

[30] Zhang Dongyu, Tian Jiadong, Li Hui. Design and Validation of Android Smartphone Based Wireless Structural Vibration Monitoring System // Sensors. — 2020. — Aug. — Vol. 20, no. 17. — P. 4799. — Access mode: http://dx.doi.org/10.3390/s20174799.

[31] android.hardware.camera2. — Access mode: https://developer.android.com/reference/android/hardware/camera2/package-summary. Accessed 11-June-2021.

[32] ptpd/ptpd: PTPd official source. — Access mode: https://github.com/ptpd/ptpd. Accessed 15-December-2020.

[33] A pulse rate estimation algorithm using PPG and smartphone camera / Sarah Ali Siddiqui, Yuan Zhang, Zhiquan Feng, Anton Kos // Journal of medical systems. — 2016. — Vol. 40, no. 5. — P. 126.