Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский
Академический университет Российской академии наук»
Центр высшего образования

Кафедра математических и информационных технологий

Пилюгин Кирилл Сергеевич

Построение дополненной реальности на основе визуальной и пространственной информации

Магистерская диссертация

Допущена к защите.
Зав. кафедрой:
д. ф.-м. н., профессор Омельченко А. В.

Научный руководитель:
Крыщенко А. С.

Рецензент:
Хабибуллин М. Р.

Санкт-Петербург
2017

Оглавление

Введение

Дополненная реальность - размещение виртуальных объектов в реальном окружении с целью дополнения сведений об этом окружении и улучшения восприятия информации. Данная технология становится все более популярной в последнее время и находит свое применение в различных областях: от компьютерных игр и киноэффектов до различных визуализаций в медицине и военной технике. Становится возможной разработка приложений с дополненной реальностью, работающих на мобильных устройствах - смартфонах и планшетах.

Для качественного размещения виртуального объекта в реальной среде необходимо решить задачу точной локализации текущего положения наблюдателя относительно данного места. В процессе локализации вводится локальная система координат и делается ее привязка к глобальной системе координат - определяется преобразование, связывающее их. В заданной локальной системе координат в дальнейшем осуществляется трекинг окружения - определение текущего положения и ориентации устройства в реальном времени. Система трекинга, подходящая для построения дополненной реальности, должна иметь высокую точность и быть устойчивой к быстрым перемещениям и поворотам устройства.

В большинстве существующих приложений для мобильных устройств в качестве средств, используемых для локализации, используется датчик GPS и сенсоры устройства: гироскоп, акселерометр и магнитометр. Но их применения без учета визуальной информации об окружении не позволяет добиться достаточно точной локализации и трекинга. Для получения визуальной информации о месте используются как обычные камеры, так и специальные камеры, предоставляющие возможность “3D-зрения”: камеры с сенсорами глубины и стерео-камеры. С развитием технологий и повышением производительности мобильных устройств становится возможным применение более сложных алгоритмов трекинга для качественной локализации с использованием информации с камеры устройства.

Можно рассматривать разные варианты задачи визуальной локализации в зависимости от исходной информации о сцене и используемых признаков. Для более простой и устойчивой локализации часто используются специальные маркеры - искусственные объекты, легко отличимые от остального окружения. Будем называть естественным окружение, не содержащее таких маркеров. Трекинг по видео камеры мобильного устройства в естественном окружении является одной из самых сложных задач в данной области и представляет наибольший интерес для исследования, поскольку это делает возможным более широкое распространение данной технологии.

В качестве карты естественного окружения чаще всего выступает разреженное облако точек с дескрипторами или трехмерная модель окружения. Техники построения дополненной реальности в естественном окружении можно разделить на две основ-

ные категории: использующие заранее построенную модель окружения и работающие без априорной информации о сцене, получая необходимую информацию о ней уже в процессе работы. Также возможен вариант расширяемого трекинга, когда есть карта только маленького участка окружения (например конкретный объект на сцене), а остальные части достраиваются посредством ее расширения.

Алгоритмы трекинга неизвестного окружения позволяют строить карту сцены и на ее основе производить производить трекинг в локальной системе координат. При этом данная система координат определяется в процессе инициализации системы и никак не связана с мировыми координатами. Для некоторых приложений хочется иметь возможность точного геопозиционирования, то есть привязки построенной локальной системы координат окружения к глобальным координатами. Примером такого приложения может быть проведение экскурсий по известным местам с автоматическим добавлением дополнительной информации рядом с достопримечательностями.

В процессе обзора готовых библиотек для построения дополненной реальности на мобильных устройствах выяснилось, что они предоставляют довольно ограниченный интерфейс при трекинге неизвестного окружения. Например, нет возможности получить построенную в процессе работы карту окружения или использовать дополнительные данные для ее построения. Таким образом, была поставлена задача разработки библиотеки, реализующей трекинг заранее неизвестного окружения по видео с камеры мобильного устройства с более гибкими возможностями, чем предоставляют готовые решения. В частности, хочется научиться решать задачу точной локализации на основе информации, которую можно автоматически получить из готовых данных зная приближенное положение.

1. Цель и задачи

Целью данной работы является исследование способов построения дополненной реальности в естественном окружении на мобильных устройствах на основе визуального трекинга и реализация на основе выбранных алгоритмов библиотеки для мобильных устройств на платформе Android с учетом особенностей платформы.

В качестве решаемых задач можно выделить следующие:

- Изучение готовых решений и библиотек для построения дополненной реальности и анализ возможностей, которые они предоставляют.
- Изучение алгоритмов, позволяющих осуществлять в реальном времени трекинг и построение карты окружения с достаточной производительностью и выбор подходящего
- Реализация библиотеки с учетом особенностей выбранной платформы.
- Реализация сохранения минимального количества информации о месте, достаточной для дальнейшей точной локализации в нем.
- Оценка качества трекинга и производительности полученного решения.
- Исследование способов привязки полученной карты окружения к реальным местам и возможности автоматического построения карты окружения с использованием заранее доступной информации.
- Исследование способов улучшить устойчивость локализации к изменениям в окружении.

2. Обзор готовых решений

В настоящее время существует достаточно большое количество фреймворков для разработки приложений на мобильных устройствах, связанных с дополненной и виртуальной реальностью. Их основные функции включают в себя:

- Трекинг с использованием искусственных маркеров. Наиболее распространенная функция, доступная практически во всех библиотеках.
- Распознавание и трекинг загруженных изображений. Иногда реализуется с помощью удаленного облачного сервиса, хранящего базу доступных изображений.
- Распознавание текста на изображениях.
- Распознавание стандартных геометрических объектов: коробок, цилиндров и т.д.
- Распознавание лиц людей.
- Трекинг с использованием 3D моделей, загруженных пользователем. Также иногда предоставляется утилита, позволяющая с помощью 3D реконструкции построить модель из нескольких изображений.
- Трекинг в естественном окружении.

Из наиболее универсальных и совершенных решений можно выделить Vuforia [32] и Wikitude [34], позволяющие разрабатывать приложения с дополненной реальностью для многих платформ и устройств. Vuforia включает в себя все перечисленные выше функции, кроме трекинга в неизвестном окружении. Также в виде отдельной утилиты доступен сканер объектов, позволяющий с помощью обычной камеры реконструировать трехмерную модель объекта и использовать ее в дальнейшем для трекинга.

Все рассмотренные готовые библиотеки для построения дополненной реальности, реализующие трекинг в неизвестном окружении, предоставляют пользователю только высокоуровневую информацию, позволяющую разместить виртуальные объекты на сцене. В Wikitude в последней версии была добавлена реализация трекинга в неизвестном окружении с использованием алгоритма, основанного на SLAM [35]. В процессе инициализации этой системы пользователь задает локальную систему координат, относительно которой производится трекинг. При этом интерфейс системы предоставляет только возможность добавления на сцену виртуального объекта в заданных координатах, но не позволяет получить никакой информации о текущем окружении.

Данные реализации не позволяют комбинировать трекинг окружения с использованием априорной информации о нем. Для получения такой возможности необходимо

создание более низкоуровневого фреймворка на основе одного из алгоритмов визуального трекинга, позволяющего пользователю напрямую работать с картой окружения. Тогда с использованием данных из открытых источников - например трехмерных моделей Google Earth [9] или панорамных снимков Google Street View [10] можно решать задачу привязки построенной локальной карты к глобальным координатам.

Более низкоуровневый фреймворк также позволит применять информацию, полученную об окружении в процессе работы, для ряда дополнительных задач. Примером такого применения может быть построение трехмерной модели окружения для дальнейшего использования или получение плотного облака точек для построения карты глубины сцены и более реалистичной отрисовки с использованием отсечения по Z-буферу.

3. Алгоритмы визуальной локализации и трекинга

В данной главе рассмотрим различные варианты алгоритмов, которые можно использовать для решения задачи трекинга по видео с камеры для построения дополненной реальности. Все методы, позволяющие производить трекинг естественного окружения на основе визуальной информации, разделяются на две основные категории: работающие с готовой моделью окружения и работающие в неизвестном окружении без априорной информации о нем.

3.1. Трекинг с использованием модели окружения

Рассмотрим методы, позволяющие производить трекинг естественного окружения при наличии трехмерной модели всего окружения или какого-либо объекта в нем. В статье [14] был приведен обзор разных подходов к этой задаче. Большинство методов являются рекурсивными, то есть использующими в качестве априорной информации о текущей позиции положение на предыдущем кадре.

3.1.1. Метод сэмплирования точек из ребер

Рекурсивный метод, основанный на сэмплировании точек из ребер исходного объекта. Для априорного положения текущего кадра определяется множество видимых ребер объекта и из них равномерно сэмплируется набор точек. Затем аналогично выделяются ребра из текущего кадра и строится набор точек для них. С помощью минимизации ошибки репроекции [11] выделенных точек определяется положение текущего кадра.

Этот метод хорошо подходит для трекинга полигональных объектов или объектов, имеющих четкие контуры, поскольку он основан на выделении границ. Информация о текстуре объекта не используется в процессе работы, поэтому метод сэмплирования точек быть успешно использован для однотонных объектов и является устойчивым к изменениям освещения.

3.1.2. Метод точек интереса

Кроме самой модели объекта для применения этого метода используются несколько кадров съемки объекта с разных ракурсов с известными положениями камеры. На каждом таком кадре выделяются ключевые точки на изображении и из них применением обратного проективного преобразования (репроекции) получается множество точек интереса [30] - точек на поверхности объекта. Для трекинга текущего кадра находится ближайший кадр из сохраненных, основываясь либо на гистограммах самих изображений, либо на положении камеры на предыдущем кадре. Затем, используя

априорное знание о текущем положении камеры, выбранный ближайший кадр с помощью репроекции преобразуется в промежуточное изображение объекта, близкое к текущему ракурсу. И уже на этом промежуточном изображении находятся соответствия для точек интереса, видимых на текущем кадре.

3.2. Трекинг в неизвестном окружении

В основе большинства систем, применяющихся для трекинга в неизвестном окружении по видео с камеры, лежит подход Simultaneous Localization and Mapping (SLAM). Это общее название для систем, позволяющих по видео с одной камеры одновременно строить карту заранее неизвестного окружения и производить локализацию в нем. В основном они разрабатывались в сфере робототехники для навигации роботов в неизвестной среде. В отличие от движения робота, при разработке системы трекинга для дополненной реальности нужно учитывать возможность более резких и непредсказуемых движений камеры. Рассмотрим разные варианты таких систем, их достоинства и недостатки.

Можно выделить два типа SLAM-систем: основанные на фильтрах (фильтр Калмана [15] или фильтр частиц [33]) и не использующие фильтры, а решающие задачу оптимизации для определения текущей позиции. Основной недостаток методов, основанных на фильтрах - текущая позиция и полное состояние карты сильно связаны и должны обновляться на каждом кадре. При этом в алгоритмах, не применяющих фильтры, возможно обновление текущего положения камеры с использованием только части карты без обновления состояния самой карты. Поэтому они способны показывать лучшую производительность трекинга.

Любая система SLAM включает в себя два основных модуля: трекинг и построение карты. Модуль трекинга отвечает за определение текущего положение камеры для каждого нового кадра, используя априорное положение (это может быть либо просто положение предыдущего кадра, либо приближенно смоделированное новое положение на основе модели движения камеры). Затем, сопоставляя данные нового кадра с имеющейся картой, вычисляется необходимое преобразование и определяется положение текущего кадра.

Модуль построения карты создает карту окружения в виде трехмерного облака точек с дескрипторами - разреженного или плотного. В процессе инициализации строится небольшая начальная карта, которая потом обновляется и дополняется в процессе работы системы. Положение точек карты в мировой системе координат вычисляется с помощью триангуляции [5] соответствий одинаковых ключевых точек на разных кадрах. Поскольку обновление карты не обязано осуществляться на каждом кадре, для него могут быть использованы методы, работающие достаточно долго, например bundle adjustment [4].

По способу обработки входящих кадров методы визуальной локализации разделяются на прямые и непрямые. Прямые методы учитывают яркость каждого пикселя изображения для определения текущей позиции, а непрямые сначала выделяют на изображениях особые точки и работают уже с набором ключевых точек и их дескрипторами. К плюсам прямых методов можно отнести лучшую работу в слабо текстурированных сценах, в которых невозможно выделить достаточное количество ключевых точек. При этом прямые методы хуже приспособлены к изменениям в освещении и гораздо более требовательны к производительности. Их работа в реальном времени стала возможна в последнее время только за счет параллелизации и вычислений на GPU.

В непрямых методах один из важнейших шагов любого алгоритма - выделение особых точек на изображении. Наиболее эффективным способом, используемым в большинстве систем, является детектор углов FAST [24]. Дескриптором ключевых точек называется набор значений, описывающих точку и применяющихся для их сравнения. В качестве дескрипторов возможны различные варианты: SIFT [20], SURF [26], локальные окрестности пикселей.

Из алгоритмов SLAM с открытым исходным кодом можно выделить:

- Parallel Tracking and Mapping (PTAM) [16]. Первый алгоритм, в котором трекинг и построение карты были разделены для исполнения в разных потоках. Лучше всего подходит для применения к построению дополненной реальности и допускает адаптацию для работы на мобильных устройствах с небольшой вычислительной мощностью.

- Semi-Direct Visual Odometry (SVO) [7]. Гибридный метод, сочетающий прямой и непрямой подход. Показывает очень хорошие результаты трекинга, но требует камеры с высокой частотой кадров (не менее 70 кадров в секунду), что делает его непригодным для работы на большинстве мобильных устройств.

- Large-Scale Direct monocular SLAM (LSD-SLAM) [3]. Прямой метод, использующий вероятностный подход для построение карты в виде плотного облака точек. Позволяет строить качественную модель окружения в реальном времени, но также требователен к производительности устройства.

После обзора различных алгоритмов визуальной локализации для реализации библиотеки был выбран алгоритм PTAM, исходя из его успешного применения для построения дополненной реальности на персональном компьютере и возможности адаптации для работы на мобильных устройствах.

4. PTAM - Parallel Tracking and Mapping

Исходный алгоритм PTAM разрабатывался в лаборатории компьютерного зрения в Оксфорде и его код был выложен в открытый доступ под лицензией GPL. В статье [16] описаны ключевые идеи PTAM и его применение к построению дополненной реальности на компьютере в окружениях небольшого размера - комната или офис, в которых возможно построить достаточно полную карту места, обходя его с камерой в процессе работы системы. Опираясь на данные из статьи и исходного кода оригинального алгоритма, рассмотрим принцип работы данного метода и подходы, применяемые при его реализации.

Ключевой особенностью данного алгоритма является разделение трекинга и построения карты для исполнения в разных потоках. Это позволяет использовать для построения карты более совершенные методы, ранее недоступные в алгоритмах, работающих в реальном времени. Основной поток, отвечающий за трекинг, обрабатывает текущий кадр с камеры и на основе имеющейся карты определяет текущее положение камеры. Для построения карты авторы вводят понятие ключевых кадров - кадров из видеопотока, лучше всего представляющих информацию об окружении. Поток построения карты отвечает за добавление в карту новых ключевых кадров, обновление положений точек на карте при поступлении новых измерений и фильтрацию карты - удаление лишних точек и ключевых кадров.

4.1. Определения

В данной главе будем считать мировой системой координат систему координат, введенную в процессе инициализации системы и не привязанную к реальной геопозиции. Назовем текущим положением камеры преобразование E_{CW}, представленное матрицей размера 4×4, переводящее точку из мировой системы координат в пространство камеры:

$$\mathbf{p}_{jC} = E_{CW}\mathbf{p}_{jW}.$$

Здесь $\mathbf{p}_{jC}$ - точка в локальной системе координат камеры, представленная в однородных координатах, а $\mathbf{p}_{jW}$ - соответственно координаты этой точки в мире.

Матрица E_{CW} - основной результат, получаемый системой при трекинге каждого кадра. Для удобства вычислений это же преобразование может быть представлено как элемент $SE(3)$ - группы Ли [11], определяющей множество всех возможных движений и поворотов в трехмерном пространстве.

Модель проекции камеры определяет проекцию точки из системы координат камеры на экран:

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \mathrm{CamProj}(E_{CW}\mathbf{p}_{iW})$$

Важным условием, необходимым для трекинга с использованием этой модели, является возможность продифференцировать данное уравнение для вычисления изменения положения камеры между кадрами:

$$E'_{CW} = ME_{CW} = \exp(\mu)E_{CW}.$$

В качестве модели проекции камеры здесь используется вариант модели камеры с точечной диафрагмой из [2], поддерживающий радиальное искажение от линз. Параметры модели включают в себя фокусные расстояния f_u и f_v, главную точку снимка (u_0, v_0), а также параметр искажения ω. Все параметры определяются на шаге калибровки камеры [2] и во время работы системы считаются известными. Функция проекции данной модели, преобразующая точку в локальной системе координат камеры, представленную в однородных координатах, в координаты экрана, определяется формулой

$$\mathrm{CamProj}\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} + \begin{bmatrix} f_u & 0 \\ 0 & f_v \end{bmatrix} \frac{r'}{r} \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix},$$

$$r = \sqrt{\frac{x^2 + y^2}{z^2}},$$

$$r' = \frac{1}{\omega} \arctan\left(2r \tan\frac{w}{2}\right).$$

Карта, которая строится в процессе работы системы, является набором ключевых кадров и точек с дескрипторами. Ключевой кадр S_i содержит в себе положение камеры в момент его съемки E_{CW} и набор измерений для точек карты, содержащихся в нем. Дескрипторами точек карты являются окрестности пикселей на ключевых кадрах, содержащих эти точки.

4.2. Инициализация

Шаг инициализации является обязательным для любой системы визуального трекинга, работающей с одной камерой. В этот момент строится первичная карта окружения и определяется начальное положение камеры. Для успешной инициализации системы требуются два кадра, на которых была снята одна и та же сцена с разных точек обзора, и в процессе инициализации по этим кадрам вычисляется либо фундаментальная матрица, либо матрица гомографии [11].

Инициализация PTAM требует от пользователя вручную задать два ключевых кадра, проведя достаточный параллельный сдвиг камеры между ними. При поступлении первого кадра с помощью детектора углов на нем выделяется набор ключевых точек. Затем соответствия для исходных ключевых точек на новых кадрах поддерживаются в течение всей процедуры инициализации, поэтому этот этап требует отсутствия резких и быстрых движений камеры. После добавления пользователем второго ключевого кадра из имеющихся соответствий точек с помощью алгоритма MLESAC [29] определяется 8 возможных разложений преобразования гомографии. Из них выбирается лучшая гомография, не порождающая точек с отрицательной глубиной, и определяется относительное преобразование камеры между первым и вторым кадром.

Начальная карта в виде облака точек строится на шаге инициализации с помощью алгоритма bundle adjustment. Масштаб сцены определяется в предположении о том, что сдвиг камеры в процессе инициализации был равен 10 см. За центр системы координат полученной карты принимается медиана всех точек исходной карты.

В статье [17] был предложен более простой способ инициализации PTAM, где от пользователя не требуется явное указание второго ключевого кадра. При этом делается дополнительное предположение о том, что исходная сцена является плоской.

4.3. Обработка кадра

Каждый кадр, получаемый с камеры, предварительно конвертируется в черно-белое изображение и из него строится пирамида из четырех уровней изображений меньшего размера. Нулевой уровень L_0 - это исходное изображение (например, размера 640×480) , на каждом следующем уровне ширина и высота уменьшаются в два раза.

Затем для каждого изображения из пирамиды с помощью детектора углов FAST извлекаются ключевые точки. Поскольку детектор углов использует окружность фиксированного размера, наличие четырех уровней изображения позволяет определять ключевые точки разного масштаба и сделать систему более устойчивой к изменению масштаба. На следующем шаге для каждой найденной ключевой точки вычисляется значение Shi-Tomasi [25] и все точки, значения для которых не превосходят определенного порога, отличающегося для разных уровней изображения, исключаются из рассмотрения. Это позволяет лучше контролировать количество особых точек на разных уровнях и оставлять только лучшие из них.

4.4. Трекинг

Первым шагом для определения текущего положения камеры является его аппроксимация с использованием модели движения камеры. Это приближение называется априорной позицией камеры. В качестве модели движения в PTAM использу-

ется модель угасающей скорости. Эта модель обновляет скорость на каждом кадре по следующей формуле: $V_{new} = 0.9 * (V_{frame} + V_{old})$, где V_{frame} - скорость движения, вычисленная из соответствий между положениями точек карты на текущем кадре и предыдущем, а V_{old} - предыдущее значение скорости. При равномерных движениях камеры такая модель позволяет минимизировать количество вычислений, необходимых для определения точного положения, т.к. априорное приближение оказывается близко к нему.

Для уточнения априорного положения камеры используется карта окружения, доступная на текущий момент. Точки карты проецируются на экран на основе априорного положения и из них строится множество "возможно видимых точек" - попадающих в пределы экрана после этой проекции. Нахождение соответствий между ключевыми точками выполняется в два этапа. На первом этапе выбирается небольшое число (около 50) точек из верхних уровней пирамиды изображений (крупные признаки) и их соответствия ищутся в достаточно большом радиусе относительно априорного положения. Это помогает сохранить возможность трекинга даже при резких движениях камеры, поскольку крупные признаки на изображениях сохраняются в таких ситуациях лучше. На втором шаге используются оставшиеся видимые точки, но здесь уже поиск соответствий осуществляется в маленьких окрестностях проекций точек.

Вычисление скорости движения камеры между двумя кадрами также помогает оценить возможное размытие изображения и соответственно модифицировать процесс трекинга. Если скорость движения велика, использование точных признаков из низких уровней пирамиды невозможно из-за большого размытия, поэтому второй шаг алгоритма не применяется и трекинг использует только крупные признаки, которые меньше подвержены размытию. Если же камера почти стационарна, использование крупных неточных признаков может привести к дрожанию, соответственно используется сразу второй шаг и поиск в небольших окрестностях априорных положений. Эти изменения позволяют увеличить стабильность алгоритма в ситуациях разного характера движений камеры.

Чтобы найти соответствие для точки карты p на текущем кадре, осуществляется поиск ее дескриптора - окрестности пикселей на кадре, из которого эта точка построена, в фиксированной окрестности ее предполагаемого положения на текущем кадре. Для компенсации искажений, возникающих из-за разных углов обзора данной точки, перед поиском к дескриптору применяется аффинное преобразование, задаваемое матрицей

$$A = \begin{bmatrix} \frac{\partial u_c}{\partial u_s} & \frac{\partial u_c}{\partial v_s} \\ \frac{\partial v_c}{\partial u_s} & \frac{\partial v_c}{\partial v_s} \end{bmatrix},$$

где (u_s, v_s) - это смещения, соответствующие горизонтальному и вертикальному смещению на один пиксель в кадре, из которого построен дескриптор, а (u_c, v_c) - аналогичные значения для текущего кадра.

Поиск дескриптора после такого преобразования реализуется с помощью вычисления метрики zero-mean SSD [11]. Значения пикселей сначала центрируются для уменьшения влияния освещения путем вычитания средней интенсивности изображения, а затем считается сумма квадратов расстояний между пикселями. Вычисляя значения метрики для дескриптора и окрестностей углов в фиксированной области текущего кадра, соответствие находится, если расстояние не превосходит заданного порога.

После нахождения множества соответствий для точек карты S на текущем кадре каждой точке из этого множества соответствует точка на изображении $(\widehat{u}, \widehat{v})^T$. Для определения изменения положения камеры на основе вычисленных соответствий используется метод наименьших квадратов в применении к задаче минимизации ошибки репроекции:

$$\mu' = \underset{\mu}{\operatorname{argmin}} \sum_{j \in S} \operatorname{Obj}\left(\frac{|e_j|}{\sigma_j}, \sigma_T\right)$$

Здесь $|e_j|$ - это вектор ошибки репроекции

$$e_j = \begin{pmatrix} \widehat{u_j} \\ \widehat{v_j} \end{pmatrix} - \operatorname{CamProj}\left(\exp(\mu) E_{CW} \mathbf{p}_j\right)$$

Функция $\operatorname{Obj}(\cdot, \sigma_T)$ - взвешенное среднее Тьюки [12], целевая функция для оценки среднего значения выборки, устойчивая к наличию выбросов. Параметр σ_T определяется как робастная оценка стандартного отклонения распределения, вычисленная из значений ошибок.

Качество трекинга определяется долей успешно найденных видимых ключевых точек на текущем кадре. Если оно не превышает некоторого порога (например 0.3), качество трекинга перестает считаться хорошим и новые точки не могут быть добавлены в карту. Если же эта доля меньше нижнего порогового значения (0.13), система считает, что текущее положение камеры является некорректным и пытается провести восстановление - локализацию на текущей карте без учета априорного положения. Для этого используется самое маленькое изображение из пирамиды: сравнивая среднеквадратичные отклонения изображений, среди всех ключевых кадров находится ближайший к текущему. Затем решается задача оптимизации для нахождения преобразования текущего кадра относительно найденного.

4.5. Построение карты

При построении карты система решает несколько задач:

- Добавление в карту новых ключевых кадров и точек
- Уточнение положений точек карты с использованием новых измерений

- Нахождение новых соответствий между точками карты и ключевыми кадрами
- Удаление плохих точек из карты

4.5.1. Добавление ключевых кадров

После шага инициализации карта окружения содержит только два ключевых кадра и небольшое количество точек. Впоследствии при движении камеры и удалении от начального положения она расширяется путем добавления новых ключевых кадров. Для того, чтобы текущий кадр был добавлен как ключевой, необходимо чтобы выполнялись следующие условия:

- Качество трекинга должно быть хорошим
- С момента добавления предыдущего ключевого кадра в карту должно пройти достаточное время (например 20 кадров)
- Положение камеры должно находиться на некотором удалении от положения ближайшего ключевого кадра, уже содержащегося в карте. При этом конкретное значение этого расстояния будет зависеть от глубины и масштаба карты: чем ближе камера находится к точкам карты, тем оно меньше.

При добавлении нового ключевого кадра в карту за его положение принимается текущее положение камеры, доступное из системы трекинга. Также в этот момент уже имеются ключевые точки, полученные детектором углов FAST на разных уровнях изображения. После применения метода подавления немаксимумов (non-maximal suppression) [11] и отсечения точек с маленькими значениями метрики Shi-Tomasi, формируется набор точек-кандидатов на добавление в карту.

Для вычисления глубины точки и ее положения в мире, недостаточно знать ее координаты на одном кадре. Для триангуляции выбирается соседний ключевой кадр - с ближайшим положением камеры к новому. Соответствия между ключевыми точками находятся с помощью эпиполярного поиска [11]. Сначала для точки-кандидата определяется эпиполярная прямая на втором кадре. Затем среди всех углов, лежащих на этой прямой, происходит поиск соответствия: окрестности пикселей сравниваются с помощью вычисления суммы квадратов расстояний. Если соответствие найдено, точка триангулируется и добавляется в карту.

4.5.2. Оптимизация карты

Кроме добавления новых кадров и точек, при построении карты проводится постоянное обновление положений точек карты с учетом новых измерений. Пусть текущее состояние карты содержит N ключевых кадров с положениями камер $E_{K_1W}, \ldots, E_{K_NW}$,

и M точек $p_1, \ldots, p_M$. Задачей глобальной оптимизации карты называется оптимизация всех параметров карты - позиций ключевых кадров $(\mu_2, \ldots, \mu_N)$ (так как позиция первого ключевого кадра всегда фиксирована) и положений точек карты $(p_1, \ldots, p_M)$. С помощью алгоритма Левенберга-Марквардта [19] решается задача минимизации ошибок репроекции:

$$(\mu_2, \ldots, \mu_N), (p_1, \ldots, p_M) = \underset{\{\mu\},\{p\}}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{j \in S_i} \operatorname{Obj}\left(\frac{|e_j|}{\sigma_j}, \sigma_T\right)$$

Сложность решения системы уравнений, получаемых при решении данной задачи, равна $O(N^3)$, и проведение полной оптимизации при большом размере карты ($>$100 ключевых кадров) может занимать достаточно продолжительное время, десятки секунд. Это допустимо в случае, когда система обозревает часть окружения с уже построенной картой, но может останавливать процесс построения карты.

Поэтому кроме полной оптимизации добавляется возможность локальной оптимизации, с использованием только части ключевых кадров $X \cup Y$ и части точек карты Z. А именно, в качестве подмножества ключевых кадров X выбирается последний кадр, добавленный в карту и ближайшие четыре к нему. Все точки, содержащиеся в кадрах из X, образуют множество точек Z. Также к множеству рассматриваемых кадров добавляется множество Y: это кадры, не содержащиеся в X, но включающие в себя точки из множества Z. То есть локальная оптимизация использует только окрестность последнего кадра, добавленного в карту. Это позволяет сократить асимптотику работы оптимизации до $O(NM)$ в худшем случае и успешно использовать ее в процессе построения карты.

Важным шагом для улучшения качества карты является добавление новых соответствий между точками карты и ключевыми кадрами. В момент добавления в карту точка содержит измерения только в двух кадрах, из которых она получена триангуляцией. Впоследствии для этой точки находятся соответствия и в других кадрах, если она в них содержится.

5. Реализация PTAM для платформы Android

Исходный алгоритм PTAM был адаптирован Клейном и Мюрреем в [17] для возможности работы на смартфоне iPhone 3G, но данная версия не была выложена в открытый доступ. Поскольку производительность процессора iPhone в то время была в 15-30 раз хуже, чем у среднего компьютера, авторам пришлось сильно упрощать исходный алгоритм и жертвовать точностью трекинга, чтобы достичь нужной производительности. В статье было показано, что даже на таком устройстве оказывается возможным успешный трекинг маленького окружения с использованием данного алгоритма.

Несмотря на то, что в настоящее время разница в производительности смартфонов и компьютеров заметно сокращается, простое портирование кода десктопной версии PTAM оказывается недостаточным для его работы в реальном времени на большинстве устройств. В статье [18] авторы предлагают вариант подходящей адаптации и оптимизации PTAM, в свою очередь добавляя использование ориентации устройства, полученной с помощью сенсоров IMU, для уточнения позиционирования. В данной главе рассмотрим детали реализации алгоритма для платформы Android, а также изменения и улучшения, основанные на работах [17] и [18], которые были внесены в процессе реализации в исходный алгоритм для лучшей работы на мобильных устройствах при сохранении достаточно точного и устойчивого трекинга.

5.1. Используемые языки и библиотеки

Платформа Android была выбрана в качестве целевой платформы как самая распространенная платформа среди современных мобильных устройств. Приложения, разрабатываемые для данной платформы, исполняются на многих устройствах с процессорами различной архитектуры: x86, ARM и других. Для достижения мультиплатформенности основная часть приложений и библиотек реализуется на языке Java, и полученный байткод исполняется на виртуальной машине Dalvik VM. При этом существует возможность для написания части приложения с использованием нативного кода на C/C++. С помощью Android Native Development Kit (NDK) [8] он компилируется в динамические библиотеки под нужные платформы. Это позволяет переиспользовать основную часть кода и библиотек, вызывая его через JNI-обертку из основного приложения.

Основная часть кода данной системы была написана на C++. Выбор языка был обусловлен возможностью переиспользовать большую часть исходного кода десктопной версии PTAM и наличием хороших библиотек компьютерного зрения. Обертки на Java были использованы только для передачи данных с камеры устройства в систему и передачи результата трекинга обратно в приложение. Несмотря на накладные расходы, связанные с вызовами через JNI, использование нативного кода позволяет

оптимизировать части приложения, содержащие большое количество вычислений. В данной реализации были использованы следующие библиотеки:

- CVD [37] - высокопроизводительная кросс-платформенная библиотека для компьютерного зрения, обработки изображений и видео. Использовалась для удобной работы с изображениями.
- TooN [28] - библиотека линейной алгебры с большим количеством эффективных алгоритмов. Использовалась для работы с векторами и матрицами: вычисления сингулярного разложения матрицы, разложения Холецкого и т.д.
- Agast - эффективная реализация различных детекторов углов, например FAST и AGAST.
- Tinyxml [27] - библиотека для работы с XML, использовалась для сериализации карты окружения.

5.2. Архитектура

В качестве интерфейса библиотеки, доступного приложению, выступают классы-обертки, которые загружают PTAM в виде динамической библиотеки libPTAM.so и затем обращаются к ней с использованием JNI. На рисунке 1 представлена диаграмма классов внешнего интерфейса.

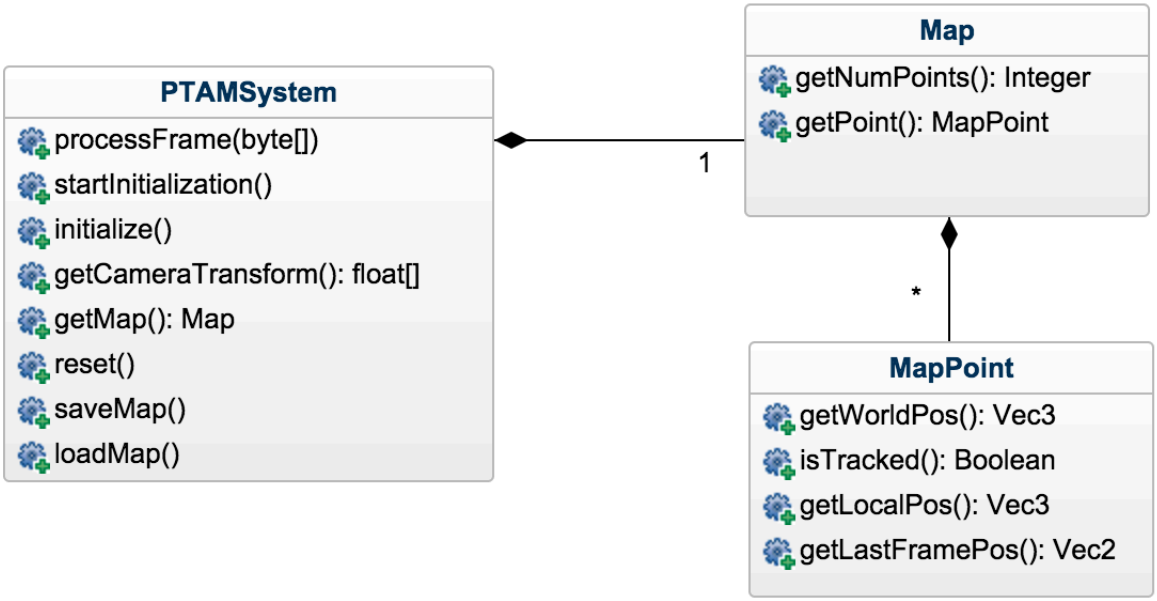

Рис. 1: Диаграмма классов внешнего интерфейса библиотеки

Основным классом системы является класс PTAMSystem. Он позволяет произвести инициализацию системы и узнавать текущее положение камеры. При начале работы указывается размер изображения, соответствующего получаемым кадрам с камеры, которые затем передаются в систему с помощью метода processFrame. Сохранение и загрузка карты по названию осуществляется с использованием методов saveMap и loadMap. Также есть возможность получать текущую карту окружения и информацию о точках карты: положение в мире и в системе координат камеры, а также дополнительную информацию о состоянии текущего трекинга: была ли эта точка найдена на последнем кадре и ее положение на нем.

В отличие от оригинального PTAM, при такой архитектуре система никак не завязана на отрисовку контента в приложении и может быть удобно использована только для осуществления трекинга. При этом возможно получение низкоуровневой информации о трекинге и карте окружения, недоступной в готовых библиотеках для построения дополненной реальности для мобильных устройств.

5.3. Адаптация алгоритма

5.3.1. Трекинг

Производительность процесса трекинга играет решающую роль в работе всей системы, поскольку он осуществляется для каждого кадра, полученного с камеры. Соответственно для достижения частоты в 30 кадров в секунду, он должен занимать не более 30 мс. Рассмотрим подробнее оригинальный алгоритм трекинга и возможности его упрощения без сильных потерь в качестве.

Алгоритм функции трекинга включает в себя следующие шаги:

1. При получении нового кадра, вычисляется априорная модель положения камеры из известного предыдущего положения и оцененной скорости движения.

2. Из текущего кадра строится пирамида изображений разного масштаба и на ней запускается поиск углов с помощью алгоритма FAST.

3. Из точек карты выбирается небольшой набор лучших ключевых точек и проецируется на текущий кадр.

4. В небольших окрестностях точек карты после проекции на изображение ищутся соответствия среди ключевых точек, выделенных детектором углов на шаге 2.

5. Из полученных соответствий уточняется априорное положение камеры.

6. Шаги 3-5 повторяются с использованием уже большего количества точек карты для более точного определения текущего положения.

Детектор углов FAST является единственной частью системы, не зависящей от размера карты и количества ключевых кадров, соответственно не допускающего ускорение посредством уменьшения карты. Адаптируя исходный алгоритм для работы на iPhone, авторы выяснили, что из-за небольшой частоты камеры и слабого процессора, размытость получаемых кадров не позволяет опираться при трекинге только на точки, полученные детектором углов FAST. Там подход уточнения положения в два этапа с использованием углов был заменен на поиск точек карты в более грубых уровнях пирамиды изображений (L1 или L2, в зависимости от степени размытости изображения). Это изменение не сильно ухудшало точность трекинга, но улучшило производительность и устойчивость к быстрым поворотом камеры.

В процессе реализации библиотеки для Android мною было проведено сравнение исходного варианта трекинга с выше описанным упрощенным. Но поскольку оно не выявило существенной разницы в производительности при ухудшении качества трекинга, так что решено было вернуться к алгоритму, включающему детектор углов FAST, с уменьшенным количеством точек карты, используемых на обоих шагах алгоритма трекинга. Количество точек на шаге грубого приближения было ограничено 30, общее количество точек для вычисления новой позиции не должно превышать 500.

5.3.2. Оптимизация ключевых кадров

Рассмотрим вычислительную сложность исходного алгоритма построения карты и его масштабируемость при увеличении количества ключевых кадров и количества точек в карте. Пусть карта окружения состоит из N точек и M ключевых кадров, причем каждая точка содержится в T ключевых кадрах. Согласно [4], различные части алгоритма bundle adjustment имеют разную асимптотику: проецирование, дифференцирование и обратная подстановка осуществляются за $O(NT)$, генерация сокращенной системы - $O(NT^2)$ и решение системы занимает $O(M^3)$. Поэтому хорошим вариантом оптимизации алгоритма является удаление из карты лишних точек и ключевых кадров.

Удаление плохих точек из карты уже было реализовано в оригинальном алгоритме PTAM. А именно, точки считались плохими, если алгоритм bundle adjustment определял их как выбросы в процессе оптимизации. Посмотрим, будут ли появляться при работе системы лишние ключевые кадры. При изначальном построении карты из-за достаточно небольшого угла обзора камеры все ключевые кадры, добавляемые в систему, важны для обеспечения достаточной плотности точек в карте. Но впоследствии возможна ситуация, когда некоторые кадры будут излишними, то есть они обозревают ту часть карты, которая уже хорошо определена другими ключевыми кадрами.

Воспользовавшись идеей из [17], добавляем в процесс построения карты дополни-

тельный шаг фильтрации готовой карты. Когда количество ключевых кадров превосходит пороговое значение (например $M = 50$), поток, строящий карту, осуществляет ранжирование текущих кадров по их полезности. Каждая точка голосует за тот ключевой кадр, который определяет ее лучше других, затем эти голоса суммируются для всех ключевых кадров. Если после такого ранжирования в карте есть ключевые кадры, полезность которых не превосходит фиксированного порогового значения, они удаляются из карты.

5.4. Сохранение карты

Реализация сохранения, загрузки и переключения между картами была основана на статье [1], описывающей расширение исходного алгоритма PTAM для работы с несколькими картами. Карта окружения состоит из двух основных частей - набор точек карты с дескрипторами и набор ключевых кадров. В качестве формата сериализации карты был выбран формат XML, позволяющий удобно сохранять необходимые атрибуты ключевых точек и кадров. Дополнительная синхронизация была добавлена для возможности заблокировать карту, чтобы другие потоки не могли ее менять во время сохранения или загрузки.

Для каждой точки карты сохраняются следующие атрибуты:

- Положение точки в мире. Основная информация о точке карты.
- Дескриптор точки - исходный ключевой кадр, уровень в пирамиде изображений, из которого пришла эта точка и ее координаты на изображении. Нужен для поиска соответствий при трекинге.
- Дополнительная информация от трекера о том, как эта точка проявляла себя при трекинге - количество раз, когда она являлась хорошим измерением (ей находилось соответствие на кадре) и выбросом. Эти данные используются при выборе маленького подмножества точек для первой фазы трекинга.

Информация о ключевом кадре включает в себя:

- Черно-белую версию изображения с камеры. Необходима для вычисления дескрипторов точек и нахождения новых ключевых точек на кадре.
- Положение камеры в данном кадре. Основная информация о ключевом кадре, использующаяся при трекинге.
- Особые точки на изображении, выделенные в момент добавления ключевого кадра в карту, и соответствующие им измерения: уровень в пирамиде изображений, положение на изображении и в пространстве камеры.

5.5. Результаты

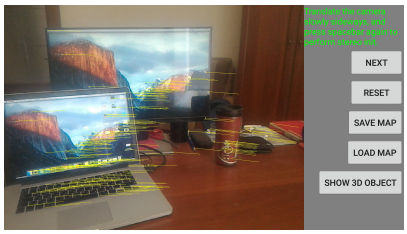

Рис. 2: Инициализация

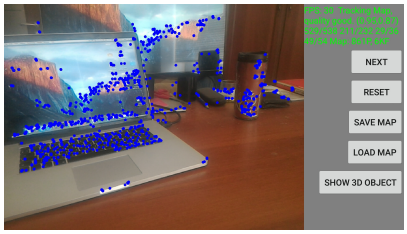

Рис. 3: Трекинг карты

Рис. 4: Добавление объекта

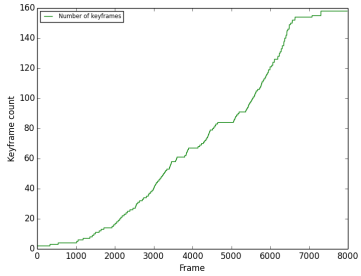

Рис. 5: Число ключевых кадров

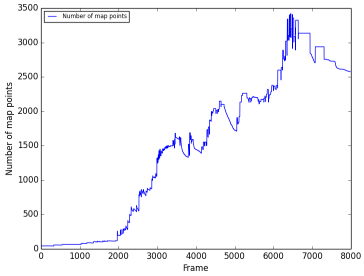

Рис. 6: Количество точек в карте

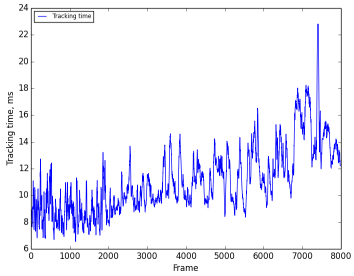

Рис. 7: Время трекинга одного кадра

Разработка и тестирование осуществлялись на смартфоне Asus Zenfone 2 с операционной системой Android 5.0 и следующими техническими характеристиками: процессор Quad-core 2.3 GHz, оперативная память 4 GB RAM. Видео с камеры устройства получается в разрешении 640x480, с частотой 30 кадров в секунду. Тестовое приложение, использующее реализованную библиотеку для трекинга, предоставляет возможность добавления трехмерного объекта на сцену в заданных локальных координатах.

В качестве окружения для тестирования системы была выбрана комната с большим количеством предметов, предоставляющих достаточное количество признаков для трекинга. В процессе инициализации (на рисунке 2) происходит медленное движение камеры параллельно столу и строится начальная карта из около 150 точек и двух ключевых кадров. Будем считать движение камеры быстрым, если она менее чем за две секунды переходит к кадру, не пересекающемуся с начальным. Если сразу после инициализации камера быстро передвигается на место, не содержащее точек начальной карты, трекинг теряется. Поэтому для успешного построения карты сначала необходимо медленно и равномерно перемещать камеру по окружению, тогда качество трекинга получается хорошим (введенная метрика качества трекинга не ниже 0.6) и новые кадры и точки успешно добавляются в карту. Дальнейший процесс трекинга (рис. 3) уже более устойчив к быстрым перемещениям камеры: даже если при наличии сильно смазанных кадров обнаруживается мало точек карты, при стабилизации камеры система быстро восстанавливается и локализуется в построенной

карте.

Рассмотрим графики, иллюстрирующие процесс построения карты окружения с использованием данной системы. Здесь камера после проведения инициализации двигалась равномерно, постоянно обозревая новые участки окружения. Итоговый размер карты составляет около 150 ключевых кадров и 3000 точек. На рисунке 5 видно, что ключевые кадры равномерно добавлялись в карту при движении камеры, соответственно поток построения карты успешно справлялся со своими задачами. При этом поскольку камера практически не возвращалась к местам с уже построенной картой, все ключевые кадры оказывались значимыми и поэтому их количество не уменьшалось. Зависимость количества точек от времени работы системы на рисунке 6 показывает, как при увеличении размера карты кроме добавления новых точек в карту происходит процесс фильтрации карты и удаление лишних либо плохих точек. Третий график подтверждает высокую производительность системы и её пригодность для работы в реальном времени: среднее время обработки одного кадра для карты размером до 3000 точек не превышает 15 мс.

5.6. Ограничения и возможные улучшения

Одним из недостатков использования алгоритма PTAM является то, что он в основном опирается на визуальную информацию об окружении, и в меньшей степени на пространственную. Поэтому простое сохранение карты для дальнейшей локализации оказывается неустойчивым даже к небольшим изменениям в окружении. Для улучшения стабильности локализации стоит рассмотреть добавление в карту новых видов признаков, более устойчивых к изменению в окружении и изменению освещения. Простым вариантом таких признаков могут являться ребра и грани, выделенные из изображений.

Выделение достаточного количества ключевых точек с помощью детектора углов является необходимым для успешной работы системы. Это накладывает определенные ограничения на окружения, в которых она может работать: в однотонных и слаботекстурированных сценах возможны проблемы. Также система является неустойчивой к сильным изменениям в освещении. Процесс двухэтапной инициализации не позволяет работать в окружениях слишком большого масштаба, поскольку там становится сложно провести инициализацию с достаточным линейным сдвигом камеры.

Наиболее интересным вариантом развития такой системы является добавление возможности построения трехмерной модели окружения и использования ее для дальнейшего трекинга. В работе [23] была разработана похожая система, способная в реальном времени производить реконструкцию трехмерного объекта одновременно с его трекингом.

6. Построение карты с привязкой к местности

Алгоритмы, рассмотренные в работе, позволяют производить локализацию в заранее неизвестном окружении. При этом при размещении виртуального объекта в реальном месте бывает недостаточно поместить его в заданную точку в системе координат камеры, а хочется получать привязку к реальным объектам. Например это может быть конкретная достопримечательность или здание. Таким образом, полученной с помощью PTAM карты местности оказывается недостаточно и появляется необходимость построения похожей карты для трекинга, но с привязкой к конкретному месту. В данной главе рассмотрим способы комбинации трекинга неизвестного окружения и априорной информации о месте для получения точной привязки к глобальным координатам. В качестве грубого приближения текущего положения наблюдателя самым удобным и естественным способом являются координаты GPS.

6.1. Выбор источника

В качестве источника визуальной информации для локализации на местности могут выступать любые изображения или видео, доступные в нужном количестве для данной местности. Хорошим вариантом источника таких изображений являются панорамы из Google Maps Street View [10], поскольку они с хорошей плотностью и равномерно покрывают местность. В статье [36] описано использование таких панорам для определения точного положения съемки фотографий.

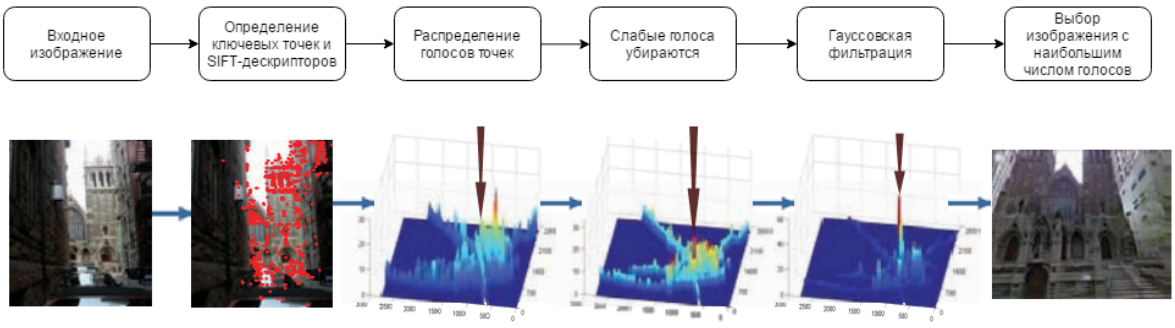

Рис. 8: Принцип работы алгоритма локализации на основе изображений

На рисунке 8 показана схема работы данной системы локализации. Взяв за основу датасет из 100000 панорам, авторы предложили следующий алгоритм для локализации изображения по запросу:

- В процессе предобработки исходного датасета строится структура данных дерево ближайших соседей, содержащая SIFT-дескрипторы всех изображений из набора.

- Для ключевых точек тестового изображения вычисляются SIFT-дескрипторы.
- После поиска лучших соответствий между дескрипторами каждая ключевая точка изображения “голосует” за свое местоположение.
- Полученное распределение фильтруется и выбирается положение с наибольшим числом голосов.

Этот метод позволяет с хорошей точностью определять реальное положение снятой фотографии.

Дополнительным плюсом использования панорам из Google Street View является возможность получения пространственной информации о местности. А именно, вместе с изображением сферической панорамы, снятой в заданных координатах GPS, для каждой точки изображения можно получить информацию о плоскости, на которой эта точка находится. Соответственно, вычисляя расстояния до таких плоскостей, можно построить карту глубин и построить трехмерное облако точек.

Альтернативным подходом является использование нетекстурированной карты, также известной как 2.5D карта: двумерные координаты зданий и их высоты. Такую карту для городов можно получать из открытых данных OpenStreetMap [22]. В работе [13] был предложен метод для локализации на улице, основанный на такой карте. В качестве признаков там используются ребра, выделяемые на изображениях. Ключевая идея метода заключается в том, что большая часть видимых ребер находятся на зданиях и являются либо горизонтальными, либо вертикальными. Это позволяет осуществлять точную локализацию наблюдателя сопоставляя видимые ребра с их положениями на карте OpenStreetMap.

6.2. Привязка карты PTAM к местности

В качестве первого шага к автоматическому построению карты местности рассмотрим следующую задачу: имея построенную карту из PTAM, будем пытаться привязать ее к реальной местности. Для простоты рассмотрим случай, когда известна точная GPS-координата текущего места и можно однозначно определить панораму, соответствующую текущему месту.

Google Street View API позволяет скачивать панорамы в хорошем качестве с помощью http-запросов. А именно, сначала получается файл с метаинформацией в формате xml, в котором содержится описание панорамы - идентификатор текущей панорамы и соседних с ней, название улицы и закодированная карта глубины. По идентификатору панорамы можно скачивать сферическое изображение в желаемом качестве по частям, задавая уровень и индексы частей. Карта глубин представляет из себя изображение размера 512x256, в котором записаны индексы плоскостей,

а также описание самих плоскостей. Сопоставляя точки на панораме с этим уменьшенным изображением, можно получить плоскость, на которой находится точка и соответственно вычислить расстояние до нее.

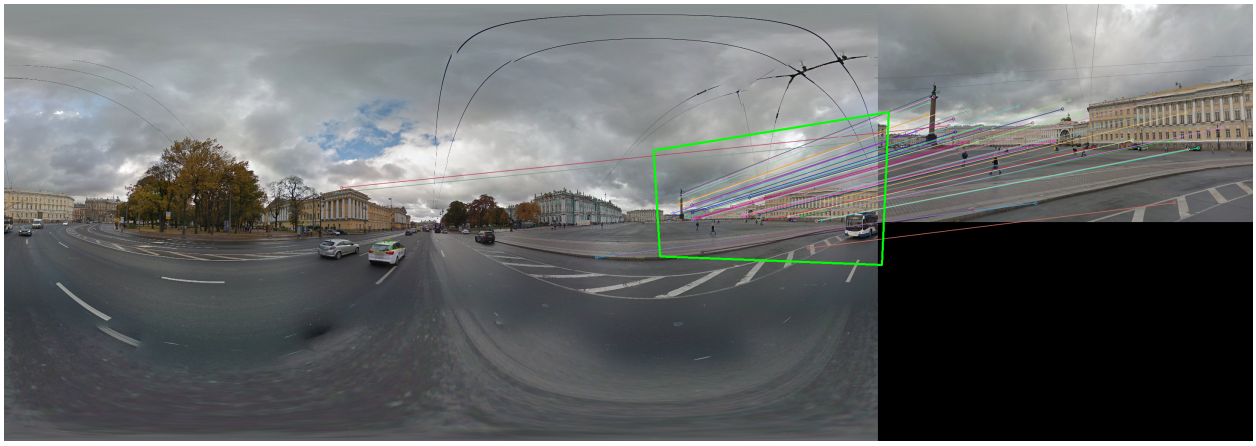

Рис. 9: Пример привязки текущего кадра к панораме

Для поиска соответствий между изображениями воспользуемся детектором ключевых точек SIFT. После нахождения соответствий между ключевыми точками текущего кадра и точками на полученной панораме с известными реальными положениями в мире, нужно найти преобразование, их связывающее. Для этого воспользуемся алгоритмом нахождения лучшей гомографии на основе метода RANSAC [6]. На рисунке 9 можно видеть пример результата такой привязки: слева - сферическая панорама из Google Street View, справа - кадр того же места с камеры. Линиями показаны соответствия между ключевыми точками карты и точками на панораме.

Применяя описанный метод в процессе работы системы, можно получать точную привязку локальной системы координат, построенной при инициализации PTAM, к мировым координатам. Основным условием для проведения успешной локализации является наличие данных, хорошо описывающих текущее окружение и позволяющих найти достаточное количество соответствий. Также можно применять для локализации более устойчивые к изменениям в окружении признаки, использующие пространственную информацию, например ребра или грани.

6.3. Инициализация с использованием панорам

Необходимость ручной инициализации делает неудобным использование системы, основанной на PTAM, в окружениях большого масштаба. В работе [31] был предложен метод подготовки карты окружения в виде облака точек для дальнейшего трекинга с использованием 3D реконструкции из видео. В качестве источника было использовано видео, снятое с всенаправленной камеры. Выделяя из видео ключевые кадры посредством сэмплирования с постоянным интервалом, полученные перекрывающи-

еся изображения затем использовались для построения трехмерного облака точек с использованием техники structure from motion [5].

Для построения облака точек из панорам Google Street View рассмотрим два подхода:

1. Выделяя ключевые точки на одной панораме, вычислять их положение в мире, используя информацию о расстояниях из карты глубин, и добавлять в облако точек, из которого получается карта окружения.

2. Находя соответствия между точками соседних панорам, производить триангуляцию и добавлять их в карту.

Первый подход является оригинальным методом, не изученным до этого. Он сочетает в себе использование визуальной и пространственной информации о месте, которую можно автоматически получать зная приближенное значение геопозиции с датчика GPS.

С применением данного подхода был разработан следующий алгоритм:

- С помощью Google Street View API скачивается нужная панорама и карта глубин для нее.

- Сферическая панорама разделяется на плоские участки и они преобразуются для исправления искажений. Участки выбираются таким образом, чтобы они примерно соответствовали кадрам, снятым обычной камерой из центра.

- Полученные изображения становятся ключевыми кадрами карты.

- Для каждого из кадров строится пирамида изображений, аналогичная PTAM, и детектором углов выделяются ключевые точки.

- Используя карту глубин, определяются мировые координаты для точек.

- Полученная структура сохраняется и используется в PTAM в качестве исходной карты.

Прототип построения такой карты был реализован в виде отдельной утилиты с использованием библиотеки компьютерного зрения OpenCV [21]. Получая на вход GPS координаты места, она находит соответствующую им панораму и строит описанную карту в том же формате, в котором сохраняется карта PTAM.

Для тестирования предложенных методов привязки карты к панорамам и автоматического построения карты был выбран набор из десяти панорам улиц Санкт-Петербурга. С помощью утилиты построения карты были построены карты соответствующих окружений и сохранены на устройство для дальнейшего трекинга с помощью PTAM. Проводилось два типа экспериментов:

- Сначала производилась инициализация PTAM, затем запускался поиск соответствий в имеющейся панораме.
- Трекинг с помощью PTAM только на основе имеющейся карты.

Основной проблемой оказались значительные несоответствия изображений из панорам и текущих кадров с камеры. Привязка к панораме работала достаточно хорошо, когда на изображениях были характерные объекты, не встречающиеся в других частях изображения: таким образом были найдены соответствия для 8 из 10 панорам. Второй подход трекинга только на основе готовой карты не дал достаточно хороших результатов: иногда системе удавалось локализоваться на какой-то небольшой части карты, содержащей хорошие признаки (например памятник характерной формы), но после отведения камеры от этого участка трекинг терялся.

Заключение

В настоящей работе были изучены методы компьютерного зрения, решающие задачу трекинга в естественном окружении для построения дополненной реальности. Используя выбранный алгоритм PTAM, была реализована библиотека для платформы Android и проведено тестирование ее производительности для успешной работы в реальном времени. Реализована возможность сохранения полученной карты окружения для дальнейшего использования.

Была рассмотрена возможность использования панорам Google Street View для решения задачи точной локализации в мировой системе координат. В качестве варианта решения предложен алгоритм привязки карты окружения, построенной с использованием PTAM, к панораме с известной геопозицией. Также реализован прототип автоматического построения карты окружения на основе панорам для дальнейшего трекинга.

Исходный код библиотеки доступен по ссылке https://github.com/kpilyugin/ARPtam

Список литературы

[1] Castle R. O., Klein G., Murray D. W. Video-rate Localization in Multiple Maps for Wearable Augmented Reality // Proc 12th IEEE Int Symp on Wearable Computers, Pittsburgh PA. — 2008. — P. 15–22.

[2] Devernay Frédréric, Faugeras Olivier. Straight Lines Have to Be Straight: Automatic Calibration and Removal of Distortion from Scenes of Structured Enviroments // Mach. Vision Appl. — 2001. — Vol. 13, no. 1. — P. 14–24.

[3] Engel J., Schöps T., Cremers D. LSD-SLAM: Large-Scale Direct Monocular SLAM. — 2014. — September.

[4] Engels Chris, Stewénius Henrik, Nistér David. Bundle adjustment rules // In Photogrammetric Computer Vision. — 2006.

[5] Faugeras Olivier. Three-dimensional Computer Vision: A Geometric Viewpoint. — MIT Press, 1993.

[6] Fischler Martin A., Bolles Robert C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography // Commun. ACM.

[7] Forster Christian, Pizzoli Matia, Scaramuzza Davide. SVO: Fast Semi-Direct Monocular Visual Odometry // IEEE International Conference on Robotics and Automation (ICRA). — 2014.

[8] Google INC, Android NDK. — URL: `https://developer.android.com/ndk/index.html` (online; accessed: 2017-05-25).

[9] Google INC, Google Earth. — URL: `https://www.google.com/earth` (online; accessed: 2017-05-25).

[10] Google INC, Google Street View. — URL: `https://www.google.com/streetview/` (online; accessed: 2017-05-25).

[11] Hartley R. I., Zisserman A. Multiple View Geometry in Computer Vision. — Second edition. — Cambridge University Press, ISBN: 0521540518, 2004.

[12] Huber Peter J. Robust statistics. — Wiley New York, 1981.

[13] Instant Outdoor Localization and SLAM Initialization from 2.5D Maps / Clemens Arth, Christian Pirchheim, Jonathan Ventura et al. // IEEE Transactions on Visualization and Computer Graphics. — 2015. — Vol. 21, no. 11. — P. 1309–1318.

[14] João Paulo Lima Francisco Simões Lucas Figueiredo Judith Kelner. Model Based Markerless 3D Tracking applied to Augmented Reality // SBC Journal on 3D Interactive Systems. — 2010. — Vol. 1.

[15] Kalman R. E. A New Approach to Linear Filtering And Prediction Problems // ASME Journal of Basic Engineering. — 1960.

[16] Klein Georg, Murray David. Parallel Tracking and Mapping for Small AR Workspaces // Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07). — Nara, Japan, 2007. — November.

[17] Klein Georg, Murray David. Parallel Tracking and Mapping on a Camera Phone // Proc. Eigth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'09). — Orlando, 2009. — October.

[18] L. Porzi E. Ricci T.A. Ciarfuglia, Zanin M. Visual-inertial tracking on Android for Augmented Reality applications // EESMS. — 2012. — September.

[19] Levenberg Kenneth. A Method for the Solution of Certain Non-Linear Problems in Least Squares // The Quarterly of Applied Mathematics. — 1944. — no. 2.

[20] Lowe David G. Object Recognition from Local Scale-Invariant Features // Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2. — ICCV '99. — 1999.

[21] OpenCV Library. — URL: `http://opencv.org/` (online; accessed: 2017-05-26).

[22] OpenStreetMap. — URL: `https://www.openstreetmap.org` (online; accessed: 2017-06-03).

[23] Real-Time 3D Tracking and Reconstruction on Mobile Phones / V.A. Prisacariu, O. Kahler, D.W. Murray, I.D. Reid // Visualization and Computer Graphics, IEEE Transactions on. — 2015. — May. — Vol. 21, no. 5. — P. 557–570.

[24] Rosten Edward, Drummond Tom. Machine Learning for High-speed Corner Detection // Proceedings of the 9th European Conference on Computer Vision - Volume Part I. — ECCV'06. — 2006. — P. 430–443.

[25] Shi Jianbo, Tomasi Carlo. Good Features to Track // 1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94). — 1994. — January. — P. 593 – 600.

[26] Speeded-Up Robust Features (SURF) / Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool // Comput. Vis. Image Underst. — 2008. — Vol. 110, no. 3. — P. 346–359.

[27] TinyXML. — URL: www.grinninglizard.com/tinyxml2/ (online; accessed: 2017-05-11).

[28] TooN: Tom's Object-oriented numerics library. — URL: https://www.edwardrosten.com/cvd/toon.html (online; accessed: 2017-05-11).

[29] Torr P. H. S., Zisserman A. MLESAC: A New Robust Estimator with Application to Estimating Image Geometry // Computer Vision and Image Understanding. — 2000. — Vol. 78. — P. 138–156.

[30] Vacchetti Luca, Lepetit Vincent, Fua Pascal. Stable Real-Time 3D Tracking Using Online and Offline Information // IEEE Trans. Pattern Anal. Mach. Intell. — 2004. — Vol. 26, no. 10. — P. 1385–1391.

[31] Ventura Jonathan, Höllerer Tobias. Wide-area scene mapping for mobile visual tracking // 11th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2012, Atlanta, GA, USA, November 5-8, 2012. — 2012. — P. 3–12.

[32] Vuforia. — URL: https://www.vuforia.com/ (online; accessed: 2017-05-11).

[33] Wikipedia, Particle filter. — URL: https://en.wikipedia.org/wiki/Particle_filter (online; accessed: 2017-06-01).

[34] Wikitude SDK. — URL: https://www.wikitude.com/ (online; accessed: 2017-05-11).

[35] Younes Georges, Asmar Daniel C., Shammas Elie A. A survey on non-filter-based monocular Visual SLAM systems // CoRR. — 2016.

[36] Zamir Amir Roshan, Shah Mubarak. Accurate Image Localization Based on Google Maps Street View // Proceedings of the 11th European Conference on Computer Vision: Part IV. — ECCV'10. — 2010. — P. 255–268.

[37] libCVD - computer vision library. — URL: https://www.edwardrosten.com/cvd/ (online; accessed: 2017-05-11).