Санкт-Петербургский Государственный Университет
Математическое обеспечение и администрирование информационных систем
Системное Программирование

Терехов Михаил Андреевич

Визуальная одометрия для систем широкоугольных камер с (не-)пересекающимися областями зрения

Выпускная квалификационная работа бакалавра

Научный руководитель:
доц. каф. СП, к.т.н. Ю.В. Литвинов

Консультант:
программист Д.А. Корчёмкин

Рецензент:
ст.преп. каф. инф., к.ф.-м.н., С.И. Салищев

Санкт-Петербург
2020

SAINT PETERSBURG STATE UNIVERSITY

Software and Administration of Information Systems

Software Engineering

Terekhov Mikhail

Visual odometry for wide-angle camera systems with (non-)intersecting fields of view

Bachelor's thesis

Scientific supervisor:

Candidate of Engineering Sciences,

docent Yurii Litvinov

Consultant:

software engineer Dmitry Korchemkin

Reviewer:

Candidate of Physics and Mathematics Sciences,

senior lecturer Sergey Salishev

Saint Petersburg

2020

Оглавление

1. Введение 4
2. Цель и задачи 6
3. Обзор существующих решений 7
   3.1. Алгоритмы 7
   3.2. Данные 9
   3.3. Direct Sparse Odometry для одной широкоугольной камеры 10
4. Переработка архитектуры одометрии 13
5. Работа с датасетом Oxford RobotCar 14
6. Работа с датасетом Multi-FoV 16
7. Совместная оптимизация 17
   7.1. Оптимизационная функция 17
   7.2. Символьное дифференцирование функций остатков . . . 18
   7.3. Вычисление якобиана функции остатков 19
8. Замеры времени работы 19
9. Сравнение однокамерной и многокамерной одометрии 20
   9.1. Качество траектории 21
   9.2. Качество облаков точек 23
10. Результаты 27

Приложения 28
A. Метод наименьших кваратов 28
   A.1. Подбор модели 28
   A.2. Алгоритм Левенберга-Марквардта 29
   A.3. Параметризация движений 30
   A.4. Функции потерь 31

Список литературы 32

1. Введение

Наука и индустрия уверенно двигаются к системам автономного управления автомобилем. Несмотря на связанные риски, компания Tesla уже предоставляет покупателям бета-версию автопилота [1]. Такой автопилот может решать всевозможные задачи по управлению автомобилем, от экстренного торможения до полностью автономного вождения в определённых условиях. Тем не менее, стоимость необходимых датчиков, оборудования и разработки программного обеспечения ещё долго будет оставаться лимитирующим фактором в повсеместном внедрении подобных систем.

Существенно дешевле было бы разработать технологию, опирающуюся на ограниченный набор датчиков и предоставляющую специфическую для этого набора функциональность. Например, уже сейчас на многие автомобили в рамках технологии *surround view* устанавливаются несколько камер, обеспечивающих круговой обзор окрестностей автомобиля (см. рис. 1). Сегодня эти камеры используются для демонстрации окружения водителю, иногда даже без предобработки.

Современные научные достижения в области 3D-реконструкции могут позволить расширить область применимости технологии surround view. Системы *визуальной одометрии* восстанавливают движение камеры и наблюдаемую мест-

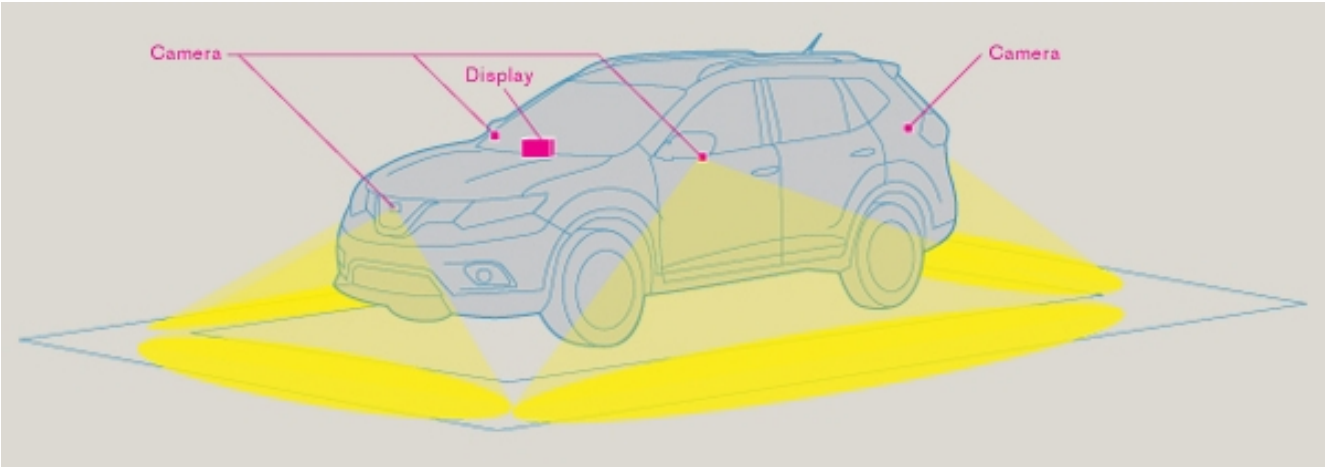


Рис. 1. Расположение камер в типичной системе surround view

ность (иногда в виде облака точек, а иногда — в виде непрерывной поверхности) по видео в реальном времени. Визуальная одометрия на камерах с автомобиля может использоваться для навигации в подземных паркингах или для планирования движения при парковке. Кроме того, подобная система может работать как дополнение к спутниковым системам навигации для предоставления гладкой траектории автомобиля.

В таких применениях, как автоматическая парковка, особенно важным фактором ставится полнота информации об окрестностях автомоблия. Если какое-то препятствие не попадёт в сгенерированное облако точек, то велика вероятность задеть это препятствие при движении. Различные методы визуальной одометрии дают различную плотность облака точек, и поэтому не все могут быть использованы для обнаружения препятствий.

Несмотря на описанные выше применения, сегодня системы визуальной одометрии слабо приспособлены к работе с группами камер, особенно если эти камеры направлены в разные стороны и требуется достаточно плотное облако. Данная работа призвана приспособить подходы, достигающие нужной плотности облака точек, к наиболее общей конфигурации системы камер.

За основу была взята курсовая работа прошлого года [2], результатом которой стало создание прототипа системы визуальной одометрии, работающего не в реальном времени с одной широкоугольной камерой. Прототип работал медленно, но доказал применимость использованных подходов к одометрии, и поэтому было принято решение развивать его далее до полноценной системы.

Результатом данной работы стала система визуальной одометрии, работающая с группами широкоугольных камер. Система была существенно ускорена по сравнению с прототипом. Было произведено тестирование на синтетическиих данных, показавшее превосходство многокамерной одометрии по сравнению с однокамерной с аналогичным набором параметров. В дальнейшем полученная система визуальной одометрии может стать основой для разработки различных приложений в области систем помощи водителю *ADAS (Advanced Driver*

Assistance Systems).

Основным вкладом данной работы является совмещение методов одометрии, достигающих высокой плотности облака точек, с системами широкоугольных камер в произвольной конфигурации. Существующие решения, в отличие от данной работы, не выложены в открытый доступ, не достигают требуемой плотности облака точек или накладывают ограничения на конфигурацию системы камер.

Оставшаяся часть работы организована следующим образом. В секции 2 описывается цель работы и основные задачи. Далее, в секции 3 анализируются существующие системы многокамерной одометрии (3.1) и датасеты, подходящие для тестирования (3.2). Затем описывается архитектура системы. При этом в секции 3.3 представлена архитектура однокамерного решения, послужившего основой для данной работы, а в секции 4 — изменения, которые потребовалось произвести для поддержки многокамерных систем. В секциях 5 и 6 описывается проделанная работа с данными. В секции 7 описывается математическая модель совместной оптимизации, используемая в данной работе. Затем в секции 8 описывается иземерние времени работы реализованного алгоритма совместной оптимизации. Наконец, в секции 9 производится качественный анализ созданной системы одометрии, а в секции 10 представлены основные результаты данной работы.

2. Цель и задачи

Данная работа нацелена на создание системы визуальной одометрии для работы с группами широкоугольных камер с потенциально непересекающимися углами обзора. При этом для обеспечения применений в сфере ADAS требуется плотность восстановленного облака в несколько тысяч видимых точек на каждом кадре, а также присутствие в облаке основных статических объектов сцены.

Для доработки прототипа одометрии до полноценной системы и обобщения до многокамерного случая потребовалось решить следующие задачи:

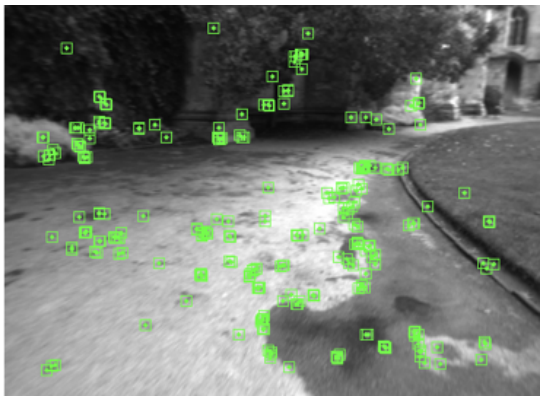
- подготовить данные для тестирования многокамерной одометрии;
- переработать архитектуру одометии для многокамерного случая;
- заменить библиотечную реализацию алгоритма оптимизации на самодельную, специфичную для одометрии. Для этого потребуется реализовать:
  - символьное дифференцирование функции остатков,
  - блочное вычисление гессиана и градиента;
- оптимизировать алгоритм одометрии по времени работы.

3. Обзор существующих решений

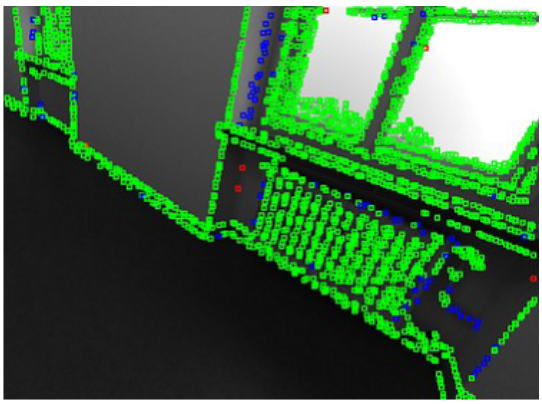
3.1. Алгоритмы

Основные современные методы визуальной одометрии принято разделять на *непрямые* и *прямые*. Первые характеризуются тем, что на этапе препроцессинга выделяют на каждом кадре набор *ключевых точек*, таких как [3], и далее работают только с ними. Информация о соответствиях точек между кадрами позволяет непрямым подходам оптимизировать *геометрическую ошибку репроекции*, т. е. расстояние в пикселях. Использование ключевых точек накладывает ограничения на плотность облака точек: необходимо устойчиво сопоставлять регионы различных кадров, а достаточно выделяющихся регионов на кадре обычно немного (см. рис. 2).

Прямые подходы оптимизируют *фотометрическую ошибку*, т. е. разницу в интенсивностях пикселей. Это позволяет использовать все области изображения с достаточно большим градиентом. Использование прямых методов делает возможными такие применения одометрии, как, например, обнаружение



(*а*) Ключевые точки системы [5]



(*б*) Контрастные точки системы [4]

Рис. 2. Сравнение плотности облаков точек, получающихся в результате работы непрямых и прямых подходов

статических препятствий. Из-за этого прототипом реализуемой системы была выбрана использующая прямой подход система DSO (Direct Sparse Odometry) [4].

На сегодняшний день представлено не так много алгоритмов визуальной одометрии, приспособленных к работе с группой широкоугольных камер. Существующие решения [6, 7] являются непрямыми, и потому не достигают высокой плотности облака точек. В то же время, основные системы [4, 8], оптимизирующие разницу в интенсивностях пикселей, были обобщены для работы с широкоугольными камерами [9, 10], но не для работы с произвольной группой камер. Более того, версии для fisheye-линз имеют закрытый исходный код.

Примером многокамерной одометрии с прямым подходом служит система [11] из проекта AutoVision [12]. Этот проект нацеливается на реализацию автономного управления автомобилем с использованием камер как основных источников информации. Реализованная в рамках этого проекта визуальная одометрия, как и данная работа, основывается на системе DSO. Однако в проекте используется условие, что у каждой камеры есть стерео-пара. Из-за этого подобная одометрия не применима, например, к системе surround view. Кроме того, описанная в проекте одометрия не выложена в открытый доступ.

Система SVO [13] (Semi-direct Visual Odometry), в свою очередь, была приспособлена к группам камер. Эта система использует комбинацию прямого и

непрямого подходов. Авторы SVO ограничились лишь кратким описанием необходимых изменений в алгоритме и потенциальными преимуществами от использования нескольких камер. На наш взгляд, многокамерные системы требуют более тщательного рассмотрения и сравнения с монокулярными и стерео-аналогами.

3.2. Данные

Для классической монокулярной и стерео-одометрии существует множество тестовых датасетов [14–16]. В то же время, существенно меньше вариантов подходят для работы в многокамерном случае. Одним из немногих датасетов, предоставляющих такую возможность, является Oxford RobotCar [17]. В нём представлены данные с нескольких сенсоров, установленных на автомобиль, проезжающий по одной и той же дороге в различных условиях в течение года. К датасету прилагается SDK с внутренней калибровкой камер и взаимным расположением сенсоров. Мы используем RobotCar как основу для тестирования на реальных данных.

Одним из недостатков RobotCar является то, что SDK предоставляет калибровку камер, пригодную для предварительной ректификации[1] видео, но не полноценные модели камер. Наш подход заключается в непосредственном использовании изображений с широкоугольной камеры, так что параметров ректификации оказывается недостаточно. Эта идея поддерживается авторами другого, более нового датасета WoodScape [18]. WoodScape гораздо лучше подходит для наших целей, но его релиз так и не состоялся, и от тестирования на WoodScape пришлось отказаться.

Для тестирования алгоритмов одометрии также оказываются полезны синтетические видео, так как для них доступны точные траектории и глубины. Кроме того, при работе со сгенерированным видео возникает меньше проблем

[1] Ректификация — преобразование изображений с широкоугольной камеры в изображения с перспективной. Обычно сопровождается потерей информации, поскольку края кадра обрезаются.

с оптикой, что помогает на ранних этапах тестирования. Мы хотели воспользоватся симулятором CARLA [19] для генерации собственных видео, однако выяснили, что CARLA использует *Level Of Detail (LOD)* при рендеринге. Из-за LOD детали изображения могут резко измениться при переходе от кадра к кадру, что является особенно неприятным эффектом для прямых методов одометрии.

3.3. Direct Sparse Odometry для одной широкоугольной камеры

Данная работа основывается на курсовой работе [2], результатом которой была система визуальной одометрии, работающая для широкоугольной камеры. Эта система, в свою очередь, опирается на описание [10] расширения системы *Direct Sparse Odometry (DSO)* [4]. DSO — это система визуальной одометии, показавшая конкурентоспособность прямого подхода как на синтетических, так и на реальных данных. Её расширение [10] не выложено в открытый доступ, подобную систему пришлось реализовывать самостоятельно.

Система DSO работает инкрементально, опираясь на уже полученные данные: на основании существующего облака точек производится *трекинг* нового кадра для определения его положения. Далее, на этом кадре отслеживаются новые и уточняются старые точки, и за счёт этого обновляется облако. Оставшееся процессорное время используется для *совместной оптимизации* — уточнения облака и расположений кадров путём оптимизации функции ошибок методом наименьших квадратов. Математическая основа метода наименьших квадратов более подробно описана в приложении A.

Процесс работы системы схематично изображён на рис. 3.

Для работы совместной оптимизации в реальном времени необходимо, чтобы набор параметров оптимизации всегда был ограничен небольшим количеством положений кадров. В то же время для повышения стабильности нужно, чтобы параметры оптимизации покрывали существенный промежуток времени. Чтобы удовлетворить оба требования, принято использовать скользящее окно из *ключевых кадров*. Ключевые кадры выбираются разреженно из входного по-

тока кадров так, чтобы покрывать возможно бóльшую часть сцены, но при этом оставлять количество параметров ограниченным. В каждый момент времени в одометрии поддерживается окно из не более, чем N активных кадров, которые участвуют в совместной оптимизации. В курсовой работе, как и в данной, вслед за [4] используется $N = 7$.

Архитектура ядра однокамерной одометрии представлена на рис. 4. Ключевые кадры, помимо расположения в пространстве, хранят преобразование яркости и два типа точек. Незрелые точки (`ImmaturePoint`) отслеживаются на приходящих кадрах, при этом значение их глубины постепенно уточняется. Когда точность становится высокой, точка становится пригодной для использования в совместной оптимизации и конвертируется в `OptimizedPoint`. Преобразование яркости же используется для моделирования изменений времени затвора и размера диафрагмы.

Любой приходящий кадр сначала не является ключевым. Информация о таком кадре хранится в классе `PreKeyFrame`. При этом положение кадра хранится относительно ближайшего ключевого, так как трекинг кадров (класс `FrameTracker`) возвращает положение именно в таком формате. Затем, в случае, если произошло достаточное смещение с прошлого ключевого кадра, новый кадр тоже становится ключевым.

Важно также отметить, что прототип одометрии использовал библиотеку

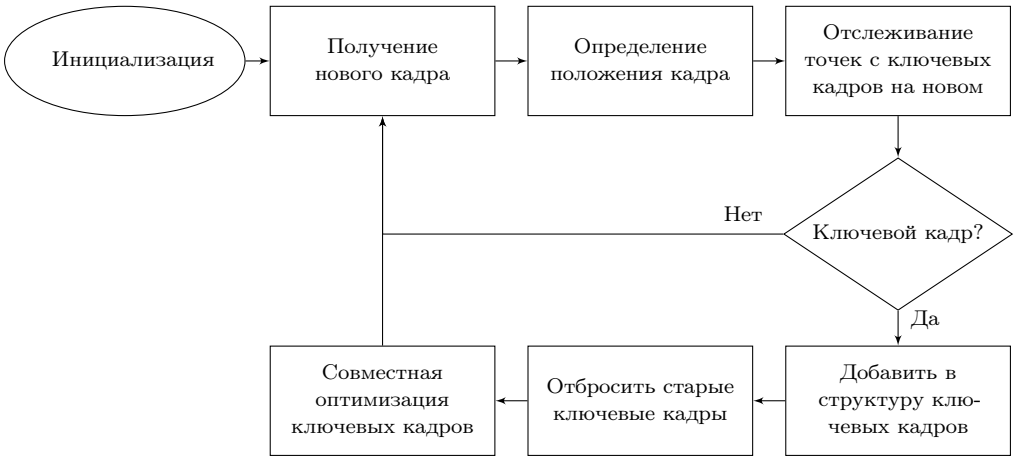


Рис. 3. Алгоритм одометрии из курсовой [2]

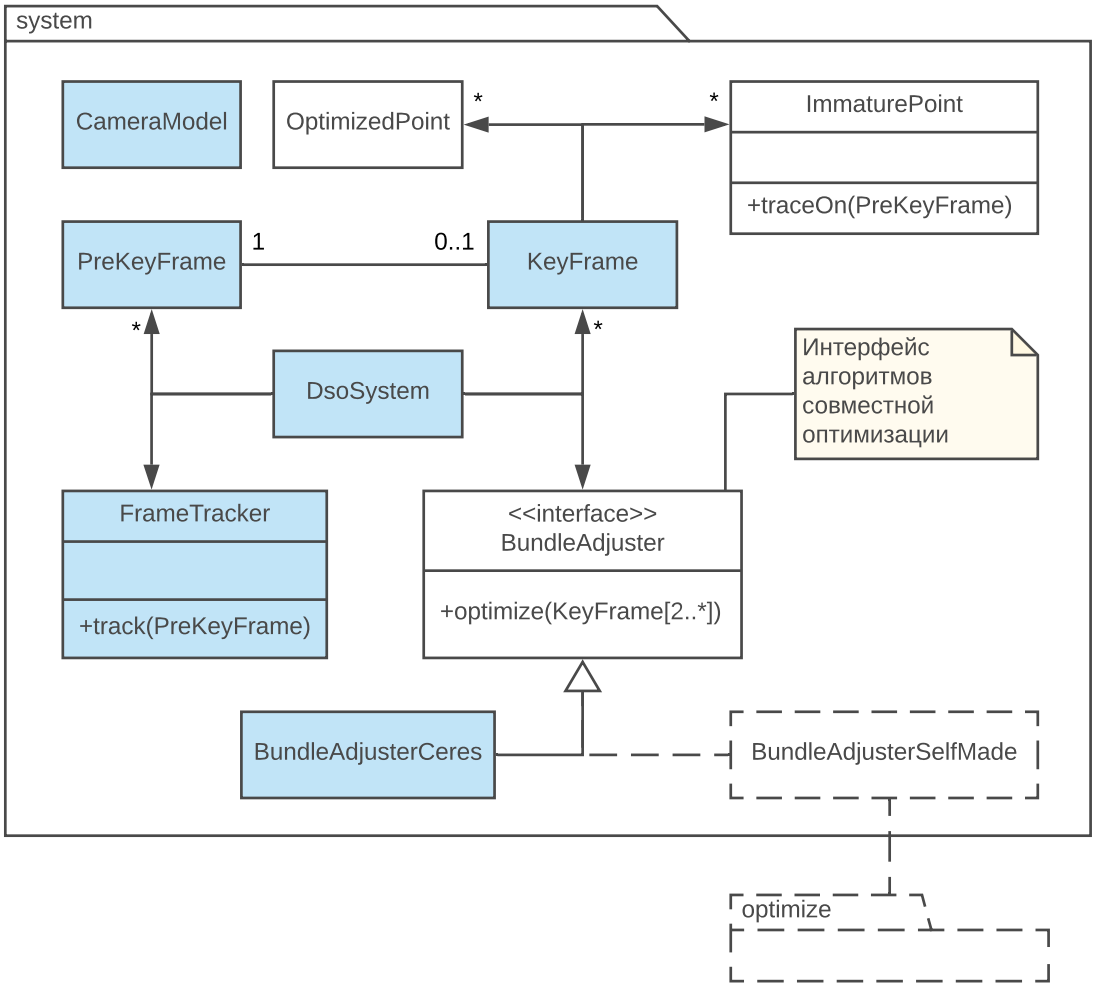


Рис. 4. Архитектура ядра однокамерной одометрии. Синим цветом отмечены классы, потребовавшие существенных изменений при переходе к многокамерным системам. Пунктиром отмечено место добавления переписанной совместной оптимизации.

ceres-solver [20] для совместной оптимизации. Эта библиотека позволяет использовать метод автоматического дифференцирования, при котором необходимые производные функции остатков находятся автоматически по коду, который вычисляет значение функции. Этот метод делает использование ceres-solver особенно удобным для апробации идей: минимизацию функции можно получить, написав лишь код для её вычисления. Автоматическое дифференцирование использовалось и в прототипе одометрии. Совместная оптимизация была реализована в классе `BundleAdjusterCeres`.

4. Переработка архитектуры одометрии

Поддержка многокамерных систем требовала изменения множества деталей в архитектуре одометрии. При этом, несмотря на то, что принципы работы на верхнем уровне остались прежними, были затронуты многие классы, составляющие ядро архитектуры системы (см. рис. 4).

Например, при наличии многокамерной системы ключевой кадр должен хранить данные, соответствующие всем изображениям, заснятым системой в фиксированный момент времени. Поскольку расположение камер в системе друг относительно друга полагается жёстко зафиксированным, положение мультикадра остаётся общим для всех изображений, но преобразования яркости и наборы точек являются свойствами каждого изображения по отдельности.

Ещё одним важным примером затронутой сущности является *модель камеры* (`CameraModel`). В многокамерном случае каждой камере соответствует отдельная модель, и появляется сущность `CameraBundle`, которая хранит эти модели и их расположение относительно центра системы.

Описанные выше преобразования схематично изображены в таблице 1.

Следующим изменением стал отказ от использования библиотеки ceres-solver. Вообще говоря, использование этой библиотеки не влечёт значительное

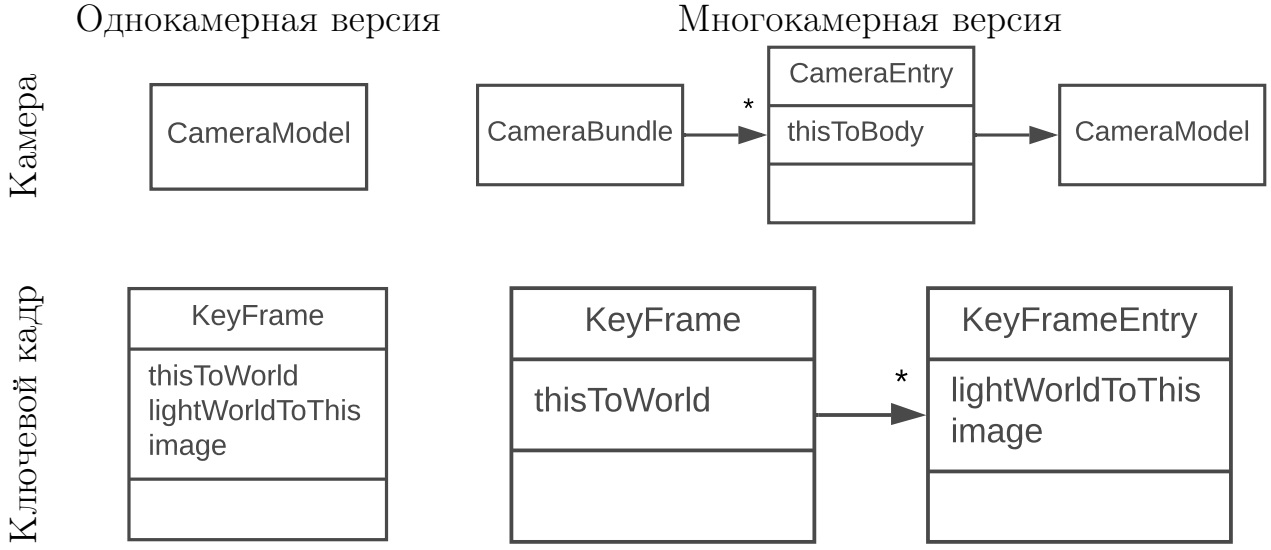


Таблица 1. Примеры преобразований, необходимых для перехода к многокамерному случаю

время работы. Основной причиной медлительности прототипа из курсовой было использование автоматического дифференцирования, при котором многие вычисления дублируются. ceres-solver предоставляет интерфейс для ручного расчёта якобиана, при котором можно добиться того же ускорения, которое было получено в данной работе. Но расчёт якобиана составляет значительную часть реализации алгоритма Левенберга-Марквардта, если не нацеливаться на универсальность, которой облаждает ceres-solver. В то же время использование ceres-solver мешало реализации дальнейших планов.

В данной работе в дальнейшем предполагается использование метода *маргинализации* старых кадров, описанного в [4]. Этот метод позволяет эффективно учитывать старые, уже зафиксированные положения кадров и точки во время совместной оптимизации и за счёт этого уменьшить ошибку одометрии. Маргинализация работает путём изменения гессиана непосредственно перед вычислением очередного шага алгоритма Левенберга-Марквардта. Но ceres-solver не формирует гессиан в явном виде. Вместо этого, он получает очередной шаг путём QR-разложения якобиана функции остатков. Учитывая это и тот факт, что вычисление якобиана функции остатков в любом случае придётся реализовывть самостоятельно, от использования ceres-solver было решено отказаться.

5. Работа с датасетом Oxford RobotCar

При сборе данных Oxford Robotcar среди датчиков присутствовало шесть камер двух видов — по три камеры каждого вида. При этом камеры разных видов имеют разную частоту съёмки. Для работы нашей многокамерной одометрии требуется синхронизация кадров, полученных с разных камер. Первая тройка камер направлена вперёд и имеет существенное пересечение областей видимости между камерами. Поэтому, мы можем воспользоватся только второй тройкой камер: левой, правой и расположенной на багажнике. Основная проблема, с которой пришлось столкнуться при использовании RobotCar, за-

ключалась в необходимости производить собственную калибровку камер. Дело в том, что SDK датасета предоставляет лишь таблицы для ректификации, а не полноценные модели широкоугольных камер. В связи с этим часть информации о линзах теряется (см. рис. 5).

Калибровка камер осуществлялась с помощью системы, разработанной в компании “Системы Компьютерного Зрения” для модели [21]. Для оценки качества результатов использовалось сравнение таблицы ректификации, полученной из модели камеры, с таблицей из SDK:

$$\mathrm{err}_\pi = \frac{1}{|Q|} \sum_{p \in Q} \pi \left(K^{-1} p_H \right) - \mathrm{LUT}\,(p) \tag{1}$$

где $Q \subset \mathbb{N}^2$ — множество пикселей изображения, K — матрица камеры после ректификации, $p_H = (p^\intercal, 1)^\intercal$, π — полученная функция проекции, $\mathrm{LUT} : Q \to \mathbb{R}^2$ — look-up table, таблица ректификации из SDK.

Перебор параметров модели и алгоритма калибровки позволил добиться ошибок, представленных в таблице 2. Ошибка в один пиксель означает, что репроекция пикселя с одного кадра на другой будет в среднем ошибаться на два пикселя. Для используемых методов это довольно крупная погрешность, предположительно вызванная тем, что паттерн для калибровки быстро двигался при съёмке, создавая эффект размытия. Тем не менее, подобную калибровку

(*а*) Ректифицированный кадр (*б*) Исходный кадр

Рис. 5. Ректифицированный кадр как часть исходного

можно использовать в дальнейшем, поскольку известно, что одометрия способна работать и на значительно пониженном разрешении изображений.

6. Работа с датасетом Multi-FoV

В курсовой, на которой основывается данная работа, для тестирования был использован синтетический датасет Multi-FoV [22]. Этот датасет предоставляет симуляцию проезда автомобиля по городу, при этом доступно видео с одной камеры, глубины пикселей[2] и точная траектория камеры.

Авторы датасета производили рендеринг в системе Blender [23] и предоставили `.blend`-файлы вместе с датасетом. Кроме того, авторы датасета предоставили исправление [24] к исходному коду Blender, добавляющий возможность рендеринга изображений с широкоугольных камер.

Для рендеринга изображений с широкоугольной камеры был собран мо-

[2] Глубина пикселя — расстояние от камеры до объекта, соответствующего этому пикселю.

Камера	левая	правая	на багажнике
err_π (пикс.)	0.6	0.7	1.0

Таблица 2. Ошибки сравнения с LUT, полученные после калибровки

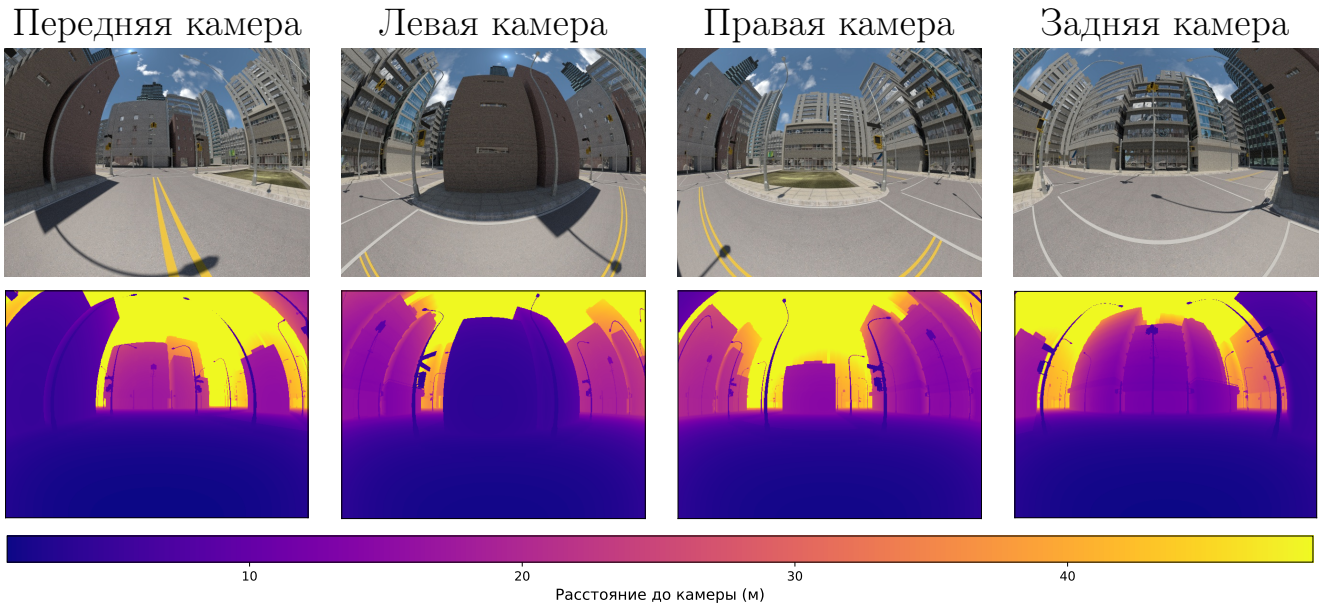


Рис. 6. Пример сгенерированного мульти-кадра и точных глубин для него

дифицированный Blender, после чего в `.blend`-файл были добавлены новые камеры. Взаимное расположение камер было зафиксировано таким образом, чтобы напоминать систему surround view. В результате удалось сгенерировать датасет Multi-FoV, модифицированный для работы с системами камер. Пример мульти-кадра[3] из полученного датасета приведен на рис. 6.

7. Совместная оптимизация

Совместная оптимизация используется в DSO для уточнения взаимного расположения кадров и глубин точек на них. Она представляет из себя минимизацию функции ошибок методом Левенберга-Марквардта. Читатель может найти подробную информацию о методе минимизации в приложении A. Далее будут разбираться лишь специфические для данного случая аспекты минимизации.

7.1. Оптимизационная функция

В нашем случае функция ошибок выглядит следующим образом:

$$
\begin{gathered}
E = \sum_{I_h \in F} \sum_{p \in P(I_h)} \sum_{I_t \in obs(p)} E_{p,t} \\
E_{p,t} = \sum_{\Delta p \in R} w_p \rho \left(I_t \left[\tilde{p} + \Delta \tilde{p} \right] - \left(\mathrm{e}^{a_t - a_h} \left(I_h \left[p + \Delta p \right] - b_h \right) + b_t \right) \right) \\
\tilde{p} = \pi_\alpha \left(\left(\omega_\alpha^{-1} \omega_t^{-1} \omega_h \omega_\beta \right) \cdot \left(\mathrm{e}^d \pi_\beta^{-1} \left(p \right) \right) \right)
\end{gathered}
\tag{2}
$$

где F — множество ключевых кадров; P — множество точек на кадре I_h, $obs\,(p)$ — кадры, на которые может репроецироваться точка p, R — паттерн репроекции, w_p — вес остатка, зависящий от градиента (рассчитывается аналогично DSO), ρ — функция потерь Хьюбера, $\tilde{p}$ — точка p, репроецированная на кадр

[3] Мульти-кадр — совокупность кадров со всех камер, наиболее близких к одному моменту времени.

t. Параметрами, по которым производится оптимизация, являются положения кадров ω_h, ω_t, логарифм глубины точки d, а так же параметры аффинных преобразований яркости a_h, b_h, a_t, b_t.

При этом паттерн репроекции — окрестность из нескольких пикселей вокруг точки, которая репроецируется. Все пиксели из этой окрестности сравниваются с репроецированными версиями на другом кадре, что повышает устойчивость минимизации к случайным совпаениям интенсивности между кадрами.

Структура и мотивация использования именно такой функции оптимизации подробно расписаны в статьях [4, 10], поэтому мы не будем здесь на этом останавливаться.

В нашей функции E есть два отличия от [10]. Во-первых, добавлен центр системы камер и преобразования $\omega_\alpha, \omega_\beta$ центров соответствующих камер в b. Во-вторых, мы решили параметризовывать глубины точек через e^d, чтобы естественным образом избавиться от отрицательных глубин.

7.2. Символьное дифференцирование функций остатков

Предподсчёт производных по ω

Как известно, имеет место представление движений $\omega \in \mathbb{M}_{3\times4}$ как матриц, отображающих векторы в однородных координатах в векторы в неоднородных. Предлагается рассматривать отображение

$$\begin{gathered} L : \mathbb{R}^7 \times \mathbb{R}^7 \to \mathbb{M}_{3\times4} \\ L\left(\omega_h, \omega_t\right) = \omega_\alpha^{-1}\omega_t^{-1}\omega_h\omega_\beta \end{gathered} \tag{3}$$

Вычислим теперь частные производные этого отображения

$$\begin{aligned} L'_{h,1} &= \frac{\partial L}{\partial x_1}, \ldots, \quad L'_{h,7} = \frac{\partial L}{\partial x_7} \quad L_{h,i} \in \mathbb{M}_{3\times4} \\ L'_{t,1} &= \frac{\partial L}{\partial x_8}, \ldots, \quad L'_{t,7} = \frac{\partial L}{\partial x_{14}} \quad L_{t,i} \in \mathbb{M}_{3\times4} \end{aligned} \tag{4}$$

Такие производные можно предподсчитать для каждой пары кадров перед очередной итерацией алгоритма оптимизации. Положим $v := \mathrm{e}^d \pi^{-1}(p)$. Тогда можно вычислить $\frac{\partial \tilde{p}}{\partial \omega_h}$ так:

$$\begin{gathered} \frac{\partial \tilde{p}}{\partial \omega_h} = J_\pi D_h \\ D_h = \begin{bmatrix} L'_{h,1} v_H & \cdots & L'_{h,7} v_H \end{bmatrix} \end{gathered} \tag{5}$$

где $J_\pi \in \mathbb{M}_{2\times 3}$ — якобиан отображения π.

7.3. Вычисление якобиана функции остатков

Далее, вычисление якобиана производится прямолинейно по правилу дифференцирования композиции. Изображение представляется как дифференцируемая функция с помощью билинейной интерполяции.

Предподсчёт производных по ω позволяет вычислять якобиан в 2.5 раза быстрее, чем при использовании автоматического дифференцирования, использовавшегося в курсовой работе. Поскольку вычисление блоков якобиана может производиться независимо, дальнейшее ускорение может быть достигнуто с помощью тривиального распараллеливания.

8. Замеры времени работы

Для демонстрации ускорения совместной оптимизации по сравнению с прототипом из курсовой, были произведены замеры процессорного времени на процессоре `Intel Core i7-8650U CPU @ 1.90GHz`.

Среднее время работы совместной оптимизации измерялось при запусках одометрии на многокамерной версии датасета Multi-FoV. Для этого совместная оптимизация из курсовой, основанная на библиотеке ceres-solver, была модифицирована для поддержки многокамерных систем. Эта модификация также использовалась для тестирования новой версии оптимизации путём сравнения ре-

зультатов оптимизации. Многокамерная одометрия запускалась с одинаковым набором параметров: 7 ключевых мульти-кадров, 2000 точек, участвующих в оптимизации, 10 итераций оптимизции. Поскольку целью была демонстрация ускорения путём реорганизации вычислений, многопоточность, которую может использовать ceres-solver, была отключена.

В результате измерений получилось, что обновлённая совместная оптимизация может выполняться в среднем за 1.05 с, в то время, как для старой версии это время составляет 50 с. Такая значительная разница вызвана тем, что в старой версии вычисления остатков из одного паттерна репроекции повторяют друг друга, что было исправлено в обновлённой реализации. Такое исправление несложно было бы сделать и в старой версии, и оно ускорило бы вычисления в 9 раз, по количеству пикселей в паттерне репроекции. Даже в таком случае старая версия проигрывала бы в 5 раз.

Важно отменить, что ускорение было получено вовсе не за счёт отказа от использования ceres-solver. Как уже говорилось, эта библиотека предоставляет возможность рассчитать якобиан функции остатков самостоятельно, и затем передать его в алгоритм оптимизации. Переиспользовав вычисление якобиана из обновлённой версии, можно было бы добиться аналогичных результатов по времени работы и с использованием ceres-solver. Отказаться от этой библиотеки было решено по другим причинам, которые были описаны выше.

9. Сравнение однокамерной и многокамерной одометрии

Многокамерная и однокамерная одометрия были протестированы на многокамерной версии датасета Multi-FoV. Для того, чтобы выровнять параметры запуска, многокамерная версия запускалась с 2000-ми точек на мульти-кадр (500 на кадр), а однокамерная — с 2000-ми точек на кадр. Обе версии запускались с ground-truth инициализацией двух ключевых кадров. Чтобы точнее просимулировать инициализацию системы на реальных данных, точные глуби-

ны предоставлялись для ключевых точек ORB на каждом кадре, после чего глубины в контрастных точках определялись с помощью интерполяции.

Основными результатами работы одометрии являются траектория и облако точек. При этом однокамерная одометрия может определить и то, и другое с точностью до общего положения и масштаба. Если увеличить все расстояния на сцене в два раза, видеопоток не изменится. Более того, из-за невозможности зафиксировать масштаб сцены, он зачастую меняется со временем. Многокамерная одометрия способна определить масштаб, поскольку ей доступны метрические расстояния между камерами, и за счёт этого она обладает фундаментальным преимуществом.

Далее приводится качественное сравнение однокамерной и многокамерной версий одометрии.

9.1. Качество траектории

Для оценки качества траектории использовались три метрики: *относительная ошибка положения, ориентации и масштаба.* Каждая метрика может быть рассчитана для произвольного фрагмента траектории $t_{i,j} = \left(t_i \dots t_j\right)$. Для этого фрагмент выравнивается с аналогичным фрагментом $t^{gt}_{i,j} = \left(t^{gt}_i \dots t^{gt}_j\right)$ из точной траектории и, в случае однокамерной одометрии, масштаб фрагмента выравнивается по расстоянию между первым и вторым кадрами. По выровненному фрагменту $\tilde{f}_{i,j} = \left(\tilde{t}_i \dots \tilde{t}_j\right)$ можно получить *абсолютные ошибки положения, ориентации и масштаба*:

- ошибка положения — расстояние между $\tilde{t}_j$ и t^{gt}_j;
- ошибка ориентации — угол между $\tilde{t}_j$ и t^{gt}_j;
- ошибка масштаба — отношение длины $\tilde{t}_{i,j}$ и $t^{gt}_{i,j}$.

Эти ошибки накапливаются с течением времени, поэтому качественными характеристиками одометрии является скорость их приращения за единицу

пройденного расстояния. Отношение ошибки положения, ориентации или масштаба к длине правильного фрагмента траектории $t_{i,j}^{gt}$ называется относительной ошибкой положения, ориентации или масштаба соответствующего фрагмента. Относительную ошибку можно вычислить для любого фрагмента $t_i \dots t_j$ полученной траектории.

Из-за ограниченной длины траектории из датасета Multi-FoV, фрагментов большой длины в этой траектории меньше, чем фрагментов маленькой. Однако нам бы хотелось учитывать вклад фрагментов разной длины с одинаковым весом, как если бы траектория была бесконечной. Для этого при вычислении общей относительной ошибки использовалась следующая методика усреднения:

1. Разбить набор длин фрагментов на 5 промежутков.

2. Для каждого промежутка посчитать среднюю ошибку фрагментов с длиной из этого промежутка.

3. Среднее из значений ошибок для каждого промежутка будет являться общей ошибкой.

Вычисленные значения ошибки для однокамерной и многокамерной одометрии приведены в таблице 3. Видно, что многокамерная одометрия, как и ожидалось, существенно меньше страдает от дрейфа масштаба.

При анализе полученных результатов нужно также учесть, что, несмотря на одинаковое количество точек на кадр, в оптимизационной задаче многокамерной одометрии участвует больше остатков, чем в однокамерной. Это связано с тем, что при наличии многокамерной системы одна и та же точка может быть репроецирована на большее количество кадров. На практике в многокамерной одометрии оказывается примерно в 2.5 раза больше остатков. Это увеличивает необходимое для вычисления время, но также и уменьшает ошибку в траектории. Используя центральную предельную теорему, можно грубо оценить

	положение (%)	ориентация (°/м)	масштаб (%/м)
многокамерная одометрия	0.91	0.016	0.0073
одокамерная одометрия	1.21	0.005	0.02

Таблица 3. Значения ошибки в траектории для однокамерной и многокамерной одометрии

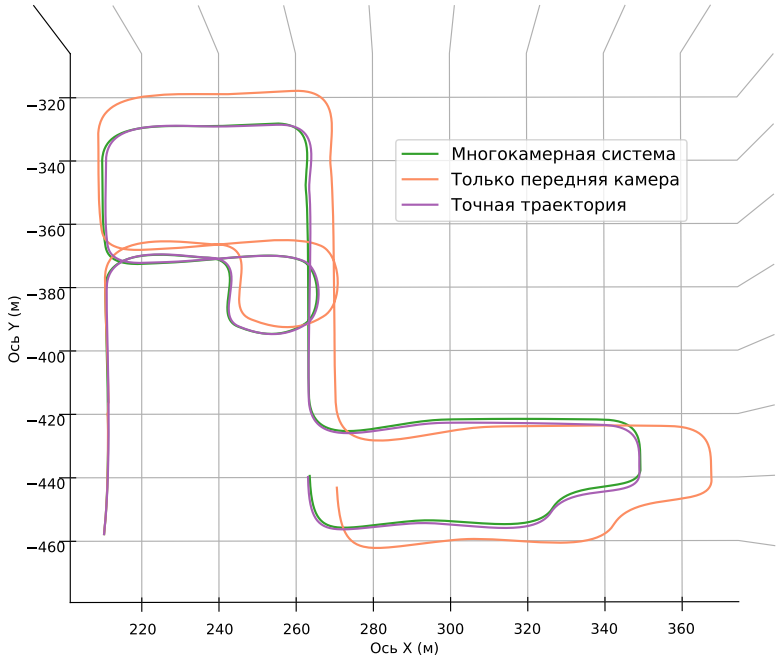
ошибку в траектории как величину, обратно пропорциональную корню из числа остатков. Это позволяет проинтерпретировать результаты в таблице 3:

- ошибка в траектории меньше в многокамерном случае лишь за счёт увеличения вычислительных ресурсов;
- многокамерная одометрия хуже справляется с оценкой ориентации;
- многокамерная одометрия позволяет уменьшить ошибку в масштабе не только за счёт дополнительных вычислительных ресурсов, но и за счёт превосходства подхода.

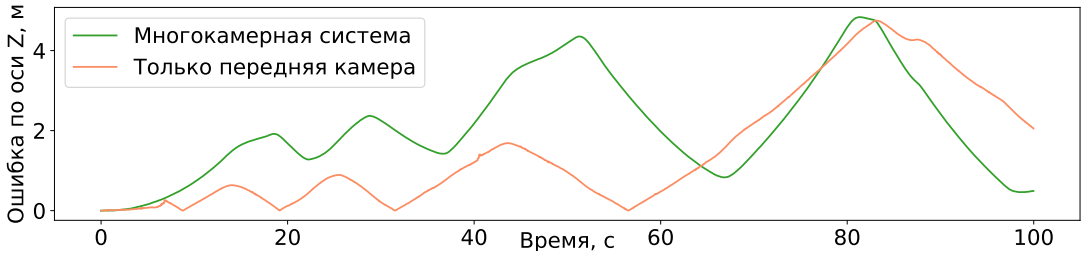
Траектории, по которым были получены данные результаты, изображены на рис. 7, *а*. Поскольку по виду сверху нельзя оценить смещение по вертикальной оси, абсолютная ошибка в смещении вдоль оси Z вынесена на отдельный график 7, *б*. Распределения относительных ошибок для различных длин фрагментов траектории, а так же средние значения ошибки для каждого промежутка длин приведены на рис. 7, *в*.

9.2. Качество облаков точек

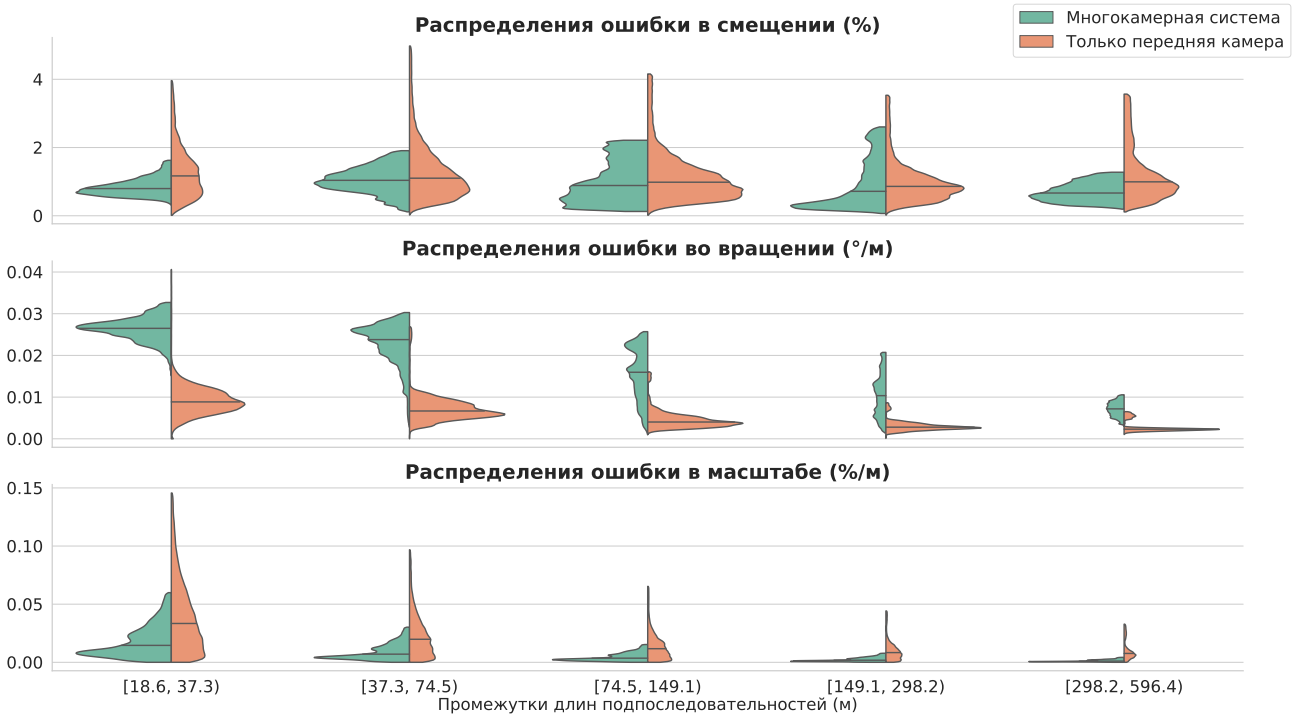
Для проверки качества облака точек, генерируемого одометрией, нужно прежде всего точное облако, с которым можно было бы сравниться. Поскольку сгенерированный датасет Multi-FoV предоставляет глубины каждого пикселя на любом кадре, можно распроецировать каждый пиксель в систему координат камеры, а затем, используя известное точное положение камеры в пространстве, перенести распроецированную точку в мировую систему координат. Так можно получить плотное облако точек, покрывающее все видимые объекты сцены.



($а$) Траектории однокамерной и многокамерной одометрии



($б$) Ошибка по оси Z



($в$) Относительная ошибка траекторий однокамерной и многокамерной одометрии

Рис. 7. Качественое сравнение однокамерной и многокамерной одометрии

Использование полученного облака напрямую, однако, будет давать преимущество многокамерной системе, поскольку она рассчитывает глубины для пикселей со всех тех же камер и в тех же направлениях, которые были использованы для построения облака. Чтобы уменьшить это преимущество, пространственное распределение точек в точном облаке делается более равномерным. Для этого пространство, содержащее облако, разбивается на кубы со строной 5 см, после чего в каждом кубе все точки заменяются средним из этих точек. Полученное точное облако обозначим C_{gt}

Следующая проблема при оценке облака заключается в том, что траектория одометрии со временем отдаляется от правильной, и генерируемое облако смещается вместе с ней. Чтобы этого избежать, вся траектория из 3000 кадров была разбита на отрезки по 100 кадров, каждый из которых выравнивался с точной траекторией по отдельности. По такой кусочно-выровненной траектории строилось облако C, которое затем сравнивалось с точным облаком.

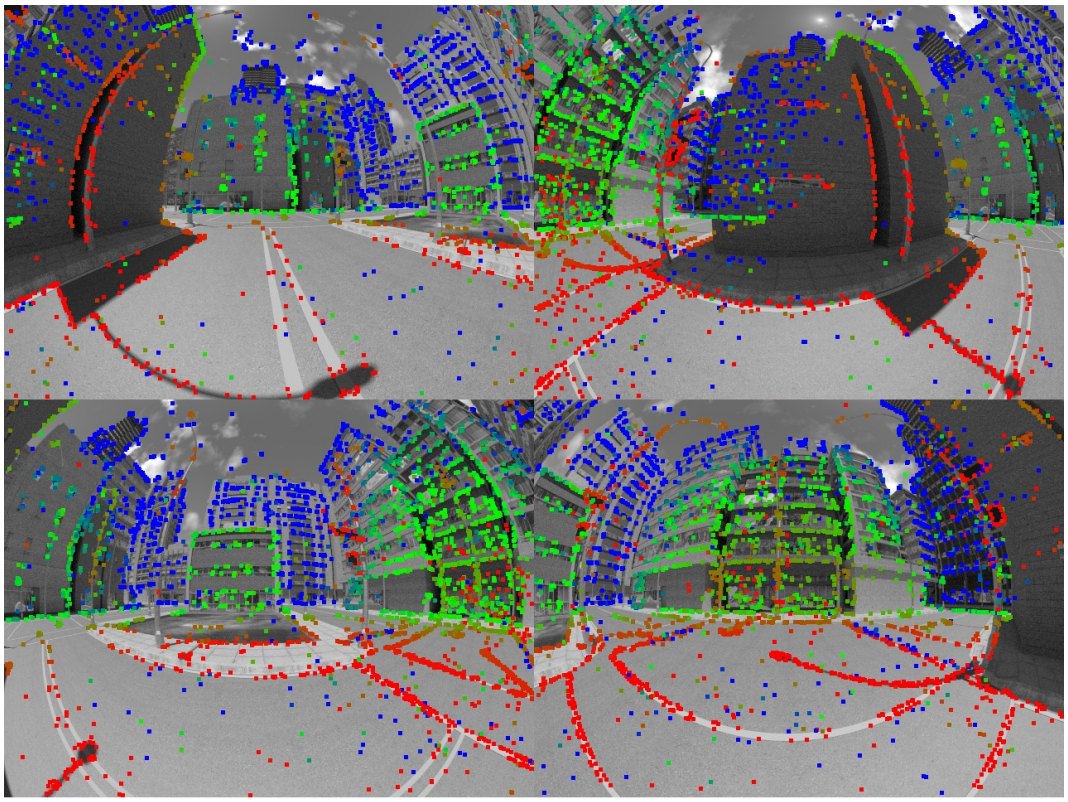
Для оценки качества построенного облака использовались две метрики: *точность* $a(C)$ и *полнота* $f(C)$ облака. Неформально говоря, точность можно определить как процент точек полученного облака, попавших близко к точному облаку, а полноту — как процент точного облака, покрытый полученным. Более строго:

$$\begin{aligned} a(C) &= \frac{1}{|C|}\left|\left\{p \in C|\ \exists p_{gt} \in C_{gt}\ \ \left\|p - p_{gt}\right\| < d\right\}\right| * 100\% \\ f(C) &= \frac{1}{|C_{gt}|}\left|\left\{p_{gt} \in C_{gt}|\ \exists p \in C\ \ \left\|p - p_{gt}\right\| < d\right\}\right| * 100\% \end{aligned} \tag{6}$$

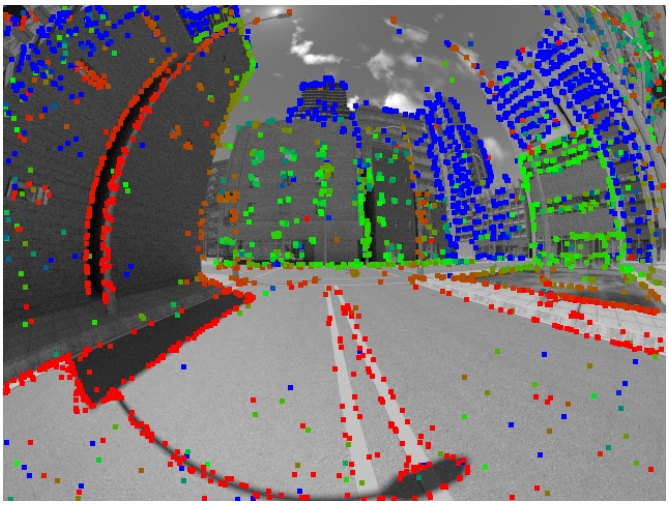
При использовании реалистичного масштаба было выбрано максимально дозволенное отклонение $d = 20$ см. Вычисленные метрики для однокамерной и многокамерной одометрии приведены в таблице 4.

	Однокамерная	Многокамерная
Точность (%)	16	39
Полнота (%)	9	23

Таблица 4. Метрики качества облаков точек для однокамерной и многокамерной одометрии



($а$) Спроецированные на мульти-кадр глубины, полученные многокамерной одометрией



($б$) Спроецированные на кадр глубины, полученные однокамерной одометрией

Рис. 8. Сравнение облаков, полученных в результате работы однокамерной и многокамерной одометрии. Красным обозначаются близкие точки, затем с увеличением расстояния цвет переходит в зелёный, а затем — в синий.

По результатам видно, что многокамерная одометрия ожидаемо превосходит однокамерную как по точности, так и по полноте облака точек. Для визуального сравнения генерируемых облаков, они также были спроецированы на входные кадры (рис. 8). По проекции можно заметить, что многокамерная одометрия, располагая тем же количеством точек, что и однокамерная, может более разреженно распределять его по окружающим объектам, за счёт чего и происходит выигрыш в качестве, демонстрируемый таблицей 4.

10. Результаты

Таким образом, на данный момент достигнуты следующие результаты:

- налажено взаимодействие с датасетом Oxford RobotCar;
- датасет Multi-FoV переделан для поддержки многокамерных систем;
- добавена поддержка многокамерных систем в архитектуру одометрии;
- реализована ускоренная совместная оптимизация, для работы которой:
  - написано символьное дифференцирование функции остатков,
  - написано блочное вычисление гессиана и градиента,
  - получено ускорение в 50 раз по сравнению со старой версией;
- создана система визуальной одометрии, работающая с группами широкоугольных камер в произвольной конфигурации;
- произведён качественный анализ результатов работы многокамерной одометрии по сравнению с однокамерной.

А. Метод наименьших кваратов

А.1. Подбор модели

Алгоритм Левенберга-Марквардта применяется для минимизации функций специального вида, обычно возникающих в задачах *подбора модели*. Вообще говоря, подбор модели — оценка неизвестных параметров модели по известным данным с сенсоров с использованием моделей сенсоров. Более строго, пусть $\mathbf{x} \in \mathbb{R}^n$ — неизвестные параметры, а $\left\{s_i(\mathbf{x})\right\}_{i=1}^m$ — модели сенсоров. Тогда ожидается, что сенсоры, получив на вход набор параметров $\tilde{\mathbf{x}}$, выдадут значения $\left\{s(\tilde{\mathbf{x}})\right\}$, которые можно будет использовать. Пусть теперь иизвестно, что значения сенсоров оказались равны $\{\tilde{s}_i\}$. Тогда можно оценить значения параметров как

$$\tilde{\mathbf{x}} = \arg\min_{\mathbf{x}} \sum_{i=1}^{m} \left(s(\mathbf{x}) - \tilde{s}\right)^2 \tag{7}$$

Сенсором, используемым в данной работе, является камера. Её моделью можно назвать отображение проекции $\pi(\mathbf{v})$, отображающее точку в трёхмерном пространстве $\mathbf{v}$ в точку в двухмерном пространстве изображения. Непрямые подходы к визуальной одометрии действуют именно так — с помощью соотнесения ключевых точек можно определить ожидаемое положение $\tilde{p}$ точки на кадре. Используя $\mathbf{v}$ как незвестную величину, получаем функцию ошибок, аналогичную 7.

Прямые подходы идут дальше, и полагают возвращаемым значением сенсора *интенсивность* изображения в точке $\pi(\mathbf{v})$. Интенсивности точках с целыми координатами полагаются равными значениям соответстущих байтов в изображении, а в нецелых производится интерполяция. Так можно представить изображение как непрерывную функцию $I : \Omega \to \mathbb{R}_+$, где $\Omega \subset \mathbb{R}^2$ — прямоугольная область изображения. В итоге неизвестными параметрами, опять же, полагается расположение точки $\mathbf{v}$[4], а моделью сенсора считается отображение $I(\pi(\mathbf{v}))$.

[4] В данной работе используется информация о том, что точка $\mathbf{v}$ лежит на известном луче, поэтому вместо трёх параметров-координат точки $\mathbf{v}$ используется один — глубина.

A.2. Алгоритм Левенберга-Марквардта

Для дальнейших выкладок удобно представить минимизацию 7 в более общем виде:

$$\tilde{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2} \left\|\mathbf{r}(\mathbf{x})\right\|^2 \qquad \mathbf{r} : \mathbb{R}^n \to \mathbb{R}^m \tag{8}$$

Функция $\mathbf{r}$ называется *функцией остатков*, а минимизируемая величина $E(\mathbf{x}) = \frac{1}{2} \left\|\mathbf{r}(\mathbf{x})\right\|^2$ — *функцией энергии*.

Для эффективной минимизации функции энергии предлагается рассмотреть разложение функции остатков в ряд Тейлора в окрестности точки $\mathbf{x}_0$:

$$\mathbf{r}(\mathbf{x}_0 + \Delta\mathbf{x}) \approx \mathbf{r}(\mathbf{x}_0) + \mathbf{J}\Delta\mathbf{x} \tag{9}$$

где $\mathbf{J}$ — матрица частных производных $\mathbf{r}$, или *якобиан*. Такое разложение позволяет приблизить функцию энергии в окрестости точки $\mathbf{x}_0$:

$$E(\mathbf{x}_0 + \Delta\mathbf{x}) \approx E(\mathbf{x}_0) + \left(\mathbf{J}^\intercal \mathbf{r}(\mathbf{x}_0)\right)^\intercal \Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^\intercal \mathbf{J}^\intercal \mathbf{J}\Delta\mathbf{x} \tag{10}$$

Это разложение — приближение использования первых трёх членов ряда Тейлора функции E. При этом $\mathbf{g} := \mathbf{J}^\intercal \mathbf{r}(\mathbf{x}_0)$ совпадает с *градиентом* функции E, а $\mathbf{H} := \mathbf{J}^\intercal \mathbf{J}$ — приближение *гессиана*, то есть матрицы вторых производных функции E. С учётом введённых определений можем переписать 10:

$$E(\mathbf{x}_0 + \Delta\mathbf{x}) \approx E(\mathbf{x}_0) + \mathbf{g}^\intercal \Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^\intercal \mathbf{H}\Delta\mathbf{x} \tag{11}$$

Минимизация функции E производится итеративно. На каждой итерации у алгоритма имеется текущее значение параметров $\mathbf{x}_k$, в его окрестности вычисляется разложение 11, и по разложению вычисляется приращение параметров:

$$\Delta\mathbf{x} := -\left(\mathbf{H} + \lambda\mathbf{I}\right)^{-1} \mathbf{g} \tag{12}$$

где $\mathbf{I}$ — единичная матрица, $\lambda > 0$ — параметр, изменяющийся во время оптимизации и контролирующий размер шага.

При маленьком λ шаг $\Delta\mathbf{x} \approx -\mathbf{H}^{-1}\mathbf{g}$ похож на шаг алгоритма Гаусса-Ньютона, который точно минимизирует правую часть разложения 11. Использование маленького λ оправдано, если разложение 11 точно приближает функцию E. Если же λ большое, то шаг алгоритма $\Delta\mathbf{x} \approx -\lambda\mathbf{g}$ приближается к шагу градиентного спуска, и поэтому увеличение λ рано или поздно приведёт к успешному шагу алгоритма. Более подробный разбор контроля шага с помощью λ читатель может найти, например, в статье [25].

A.3. Параметризация движений

Представление движений является хорошо известной проблемой в сообществах компьютерной графики и зрения. Особенный интерес представляет параметризация вращений. Минимально для представления вращения необходимо три параметра. К примеру, углы Эйлера или представление с помощью оси и угла позволяют задать вращение с помощью трёх чисел. К сожалению, и то, и другое представление страдают от вырожденных положений, в которых невозможно малым приращением параметров выразить определённые малые изменения вращения, или приращение в определённом направлении не изменяет положение.

Удачным компромиссом компактности и удобства использования оказывается представление вращений с помощью *единичных кватернионов*[5]. Полностью движение при этом представляется с помощью 7 параметров (4 на вращение и 3 на смещение), но небольшое *приращение* движения можно представить в виде минимально необходимых 6 параметров с помощью арсенала *групп Ли*. Для данной работы основным примером группы Ли является группа движений в трёхмерного пространства, не меняющих ориентацию SE(3). Этой группе со-

[5] http://graphics.stanford.edu/courses/cs348a-17-winter/Papers/quaternion.pdf (доступна на 30.05.2020)

ответствует алгебра Ли $\mathfrak{se}(3)$ и экспонента $\exp : \mathfrak{se}(3) \to \mathrm{SE}(3)$. Использование экспоненты позволяет элегантно выразить приращение к движению ω_0, задействовав 6 параметров:

$$\omega = \omega_0 \boxplus \Delta\omega := \exp(\Delta\omega)\,\omega_0 \tag{13}$$

причём отображение будет дифференцируемо по $\Delta\omega$. Это позволяет на каждом шаге оптимизации искать приращение $\Delta\omega$ и избегать понижения ранга якобиана, которое происходило бы при использовании всех 7 параметров.

A.4. Функции потерь

На практике в данных, приходящих с сенсоров, регулярно оказываются выбросы. Например, при репроекции точки с кадра на кадр её может перекрыть более близкий объект, или поиск соответствия точек между кадрами может вернуть ложное соответствие. Наличие ложного измерения может существенно повлиять на минимум 7, поскольку отличие от предсказанного значения возводится в квадрат. Чтобы избежать подобных проблем, можно использовать т.н. *функции потерь*, которые слабо влияют на малые значения остатков, но существенно уменьшают влияние больших остатков на функцию энергии. С использованием функции остатков $\rho : \mathbb{R} \to \mathbb{R}_+$ функция энергии принимает вид:

$$E(\mathbf{x}) = \sum_{i=1}^{m} \rho\left(\mathbf{r}_i(\mathbf{x})\right) \tag{14}$$

Такое изменение оставляет возможность использовать разложение 11 при модификации выражений гессиана и градиента. Известной функцией потерь, использующейся и в данной работе, является *функция потерь Хьюбера*:

$$\rho_{huber}(r) := \begin{cases} r^2 & |r| < r_{th} \\ r_{th}\left(2\,|r| - r_{th}\right) & |r| \geq r_{th} \end{cases} \tag{15}$$

Список литературы

1. Tesla autopilot capabilities. — 2019. — Access mode: https://www.tesla.com/en_EU/autopilot (online; accessed: 30.05.2020).
2. Терехов М. Визуальная одометрия для широкоугольных камер: прямой подход. — http://se.math.spbu.ru/SE/YearlyProjects/spring-2019/344/344-Mikhail-Terekhov-report.pdf (online, accessed: 30.05.2020). — 2019.
3. Orb: An efficient alternative to sift or surf / E. Rublee, V. Rabaud, K. Konolige, G. Bradski // European Conference on Computer Vision. — 2011. — P. 2564–2571.
4. Engel J., Koltun V., Cremers D. Direct sparse odometry // IEEE transactions on pattern analysis and machine intelligence. — 2018. — Vol. 40, no. 3. — P. 611–625.
5. Mur-Artal R., Montiel J., Tardos J. Orb-slam: A versatile and accurate monocular slam system // Transactions on Robotics. — 2015. — Vol. 31, no. 5. — P. 1147–1163.
6. Harmat A., Trentini M., Sharf I. Multi-camera tracking and mapping for unmanned aerial vehicles in unstructured environments // Journal of Intelligent and Robotic Systems. — 2015. — Vol. 78, no. 2. — P. 291–317.
7. Urban S., Hinz S. Multicol-slam - a modular real-time multi-camera slam system. — 1610.07336.
8. Engel J., Schöps T., Cremers D. Lsd-slam: Large-scale direct monocular slam // European Conference on Computer Vision. — Springer, 2014. — P. 834–849.
9. Caruso D., Engel J., Cremers D. Large-scale direct slam for omnidirectional cameras // IROS. — 2015.
10. Omnidirectional dso: Direct sparse odometry with fisheye cameras / H. Matsuki, L. von Stumberg, V. Usenko et al. // IEEE Robotics and Automation

Letters and Int. Conference on Intelligent Robots and Systems (IROS). — IEEE, 2018.

11. Towards robust visual odometry with a multi-camera system / Peidong Liu, Marcel Geppert, Lionel Heng et al. // 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) / IEEE. — 2018. — P. 1154–1161.
12. Project autovision: Localization and 3d scene perception for an autonomous vehicle with a multi-camera system / Lionel Heng, Benjamin Choi, Zhaopeng Cui et al. // 2019 International Conference on Robotics and Automation (ICRA) / IEEE. — 2019. — P. 4695–4702.
13. Svo: Semidirect visual odometry for monocular and multicamera systems / Christian Forster, Zichao Zhang, Michael Gassner et al. // IEEE Transactions on Robotics. — 2016. — Vol. 33, no. 2. — P. 249–265.
14. Geiger A., Lenz P., Urtasun R. Are we ready for autonomous driving? the kitti vision benchmark suite // Conference on Computer Vision and Pattern Recognition (CVPR). — 2012.
15. The euroc micro aerial vehicle datasets / Michael Burri, Janosch Nikolic, Pascal Gohl et al. // The International Journal of Robotics Research. — 2016. — Vol. 35, no. 10. — P. 1157–1163.
16. Benefit of large field-of-view cameras for visual odometry / Zichao Zhang, Henri Rebecq, Christian Forster, Davide Scaramuzza // 2016 IEEE International Conference on Robotics and Automation (ICRA) / IEEE. — 2016. — P. 801–808.
17. Maddern W., Pascoe G., Linegar C., Newman P. 1 Year, 1000km: The Oxford RobotCar Dataset. — 2017. — http://ijr.sagepub.com/content/early/2016/11/28/0278364916679498.full.pdf+ht
18. Woodscape: A multi-task, multi-camera fisheye dataset for autonomous driving / Senthil Yogamani, Ciarán Hughes, Jonathan Horgan et al. // arXiv preprint arXiv:1905.01489. — 2019.

19. CARLA: An open urban driving simulator / Alexey Dosovitskiy, German Ros, Felipe Codevilla et al. // Proceedings of the 1st Annual Conference on Robot Learning. — 2017. — P. 1–16.
20. Agarwal S., Mierle K., Others. Ceres solver. — http://ceres-solver.org (online, accessed: 30.05.2020).
21. Scaramuzza D., Martinelli A., Siegwart R. A flexible technique for accurate omnidirectional camera calibration and structure from motion // Fourth IEEE International Conference on Computer Vision Systems (ICVS'06). — 2006. — P. 45.
22. Benefit of large field-of-view cameras for visual odometry / Zichao Zhang, Henri Rebecq, Christian Forster, Davide Scaramuzza // 2016 IEEE International Conference on Robotics and Automation (ICRA) / IEEE. — 2016. — P. 801–808.
23. Community B. O. — Blender - a 3D modelling and rendering package. — Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. — Access mode: http://www.blender.org (online; accessed: 30.05.2020).
24. Zhang Z., Rebecq H., Forster C., Scaramuzza D. Patch for adding an omnidirectional camera model into blender (cycles). — https://github.com/uzh-rpg/rpg_blender_omni_camera (online, accessed: 30.05.2020). — 2016.
25. Madsen K., Nielsen H. B., Tingleff O. Methods for non-linear least squares problems. — 2004.