Санкт-Петербургский Государственный Университет
Фундаментальная информатика и информационные технологии

Кафедра информатики

Шевченко Вячеслав Александрович

Разработка методики сравнения методов визуальной одометрии для монокулярных камер с использованием открытых баз данных

Бакалаврская работа

Научный руководитель:
к.ф.-м.н., ст. преп. СПбГУ Салищев С. И.

Рецензент:
к.т.н., доцент ГУАП Ронжин А. Л.

Санкт-Петербург
2018

SAINT-PETERSBURG STATE UNIVERSITY

Fundamental informatics and informational technologies
Department of informatics

Viacheslav Shevchenko

Development of methodology for comparing visual odometry approaches using open datasets

Bachelor's Thesis

Scientific supervisor:
PhD, senior lecturer Sergey Salishev

Reviewer:
PhD, assoc. prof. Alexander Ronzhin

Saint-Petersburg
2018

Оглавление

Введение

Одной из фундаментальных задач в сфере мобильных роботов и беспилотных автомобилей является *локализация объекта и построение карты окружающей территории*. Существет множество подходов к решению этой проблемы с использованием различных технических средств, например, таких, как лазерные установки типа LiDAR [5, 22], IMU [16], GPS, радар [21]. Все они, однако, имеют различные недостатки. Например, технология LiDAR очень дорогая, а GPS имеет огромную погрешность и не может использоваться сама по себе в системах, где требуется большая точность. В связи с этим, большой интерес представляют методы *визуальной одометрии*, то есть методы, которые используют информацию, захваченную с видеопотока камеры, установленной на объекте. Действительно, камеры имеют низкую стоимость по сравнению с большинством других технических средств, кроме того существуют алгоритмы, способные качественно преобразовывать фотометрическую информацию в информацию о местоположении камеры. Разумеется, и этот способ страдает от многих проблем. Например, плохое освещение может сильно испорить оценку движения, а для корректного сопоставления необходимо доминирование в окружении статичных объектов. Кроме того, существуют фундаментальные геометрические ограничения для определения точного вращения и перемещения камеры по изображениям. Для многих систем, однако, это наиболее многообещающий подход, нередки и его сочетания с использованием других дополнительных датчиков (LiDAR, IMU и т.д.). Кроме того, текущие исследования далеко продвинулись в ослаблении ограничений для применимости этого метода. Настоящая работа, следуя тенденциям, представляет обзор современных подходов визуальной одометрии для монокулярных камер.

1. Постановка задачи и терминология

1.1. VO и SLAM

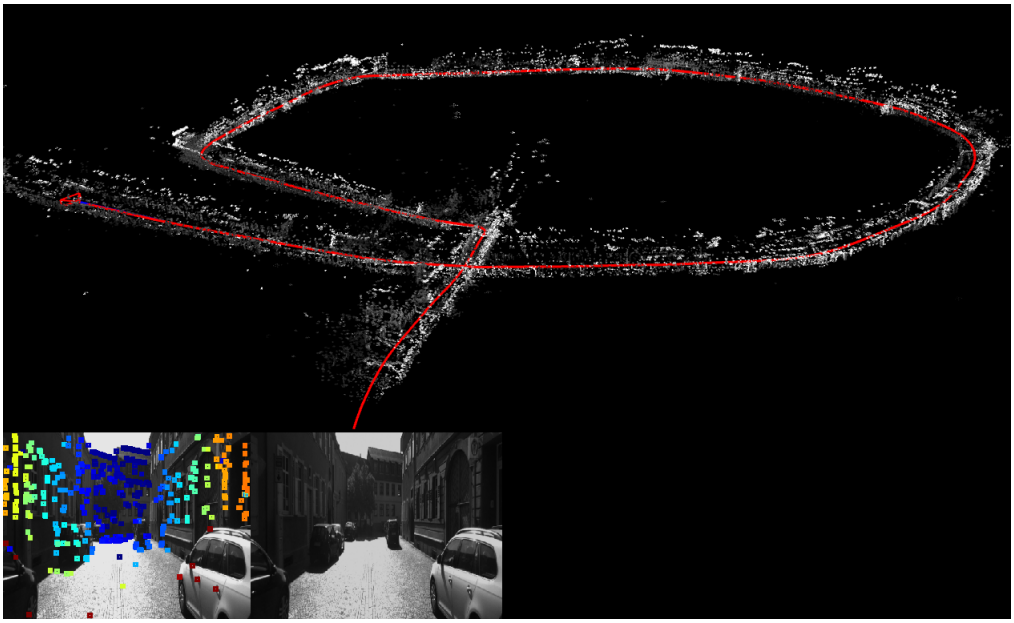


Рис. 1: Пример работы Direct Sparse Odometry на KITTI seq. 18

Визуальная одометрия — метод оценки положения и ориентации устройства с помощью анализа видеопотока с камер, установленных на устройстве (см. рис. 1). Ключевой его особенностью является то, что мы используем два последовательных кадра для оценки локального (относительного) движения объекта с итеративным построением траектории. Стоит отметить, что из-за этого VO страдает от накапливаемой ошибки оценок последовательных перемещений, и борьба с этим важный момент многих алгоритмов. Кроме того, существует фундаментальное геометрическое ограничение: по паре кадров монокулярной камеры без дополнительной информации принципиально нельзя восстановить масштаб перемещения (*scale ambiguity*) [19]. Следствием этого является невозможность точной *триангуляции пары точек изображения*, то есть определения Z-координаты точки 3D-мира. Есть технологические методы борьбы с этим (использование RGBD или стереокамер). Если же используется монокулярная камера, то инициализация и оценка карты глубин является довольно тонким местом алгоритма.

Близким понятием к визуальной одометрии является следующая задача.

Visual Simultaneous localization and mapping — определение положения и траектории движения устройства, а так же построение карты территории. Является более глобальной задачей по сравнению с VO, которая может использоваться для инициализации алгоритмов VSLAM. Одним из краеугольных камней SLAM является *замыкание циклов* (loop closing) — определение посещали ли мы уже данную область, где и проявляются тонкие моменты визуальной одометрии: неоднозначность масштаба и накопление ошибки абсолютного перемещения. Можно сказать, что VSLAM = VO + loop closing.

В данной работе рассматриваются только алгоритмы для монокулярных камер, хотя некотрые из них допускают обобщения на случай стереоизображений.

Стоит отметить, что задачи SLAM/VO *в помещении* (indoor) и *вне помещения* (outdoor) существенного отличаются друг друга (например, из-за разного фотометрического шума и средней глубины объектов). Мы постараемся сравнить алгоритмы в обоих случаях.

1.2. Предварительные сведения

Для формального описания исследуемых алгоритмов нам понадобится некоторое математическое введение.

1.2.1. Группы Ли SO(3), SE(3), Sim(3)

- Группа Ли — множество с операцией непрерывного ассоциативного умножения и выделенной еденицей (мы используем мультипликативную нотацию), обладающее структурой гладкого многообразия (т.е. локально представляющее из себя пространство $\mathbb{R}^n$).

- С такими группами тесно связаны алгебры Ли — векторные пространства, снабженные билинейным отображением $[\cdot, \cdot]$ (скобка Ли),

для которого верны тождества:

$$[x, x] = 0, \tag{1}$$

$$[x, [y, z]] + [y, [x, z]] + [z, [x, y]] = 0. \tag{2}$$

Мы не будем подробно останавливаться на соотвествующей теории. В дальнейшем лишь вскользь упомянем, как связаны эти объекты.

- $SO(3)$ — группа вращений трехмерного пространства. $R \in SO(3)$ будет обозначать представление её элемента в матричном виде.
- $SE(3)$ — группа движений твердого тела. $T \in SE(3)$ будет обозначать представление её элемента в матричном виде. Если $R \in SO(3)$, вектор перемещения $t \in \mathbb{R}^3$, то

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

- $Sim(3)$ — группа преобразований подобия. $G \in Sim(3)$ будет обозначать представление её элемента в матричном виде. Если $R \in SO(3), t \in \mathbb{R}^3, s \in \mathbb{R}$, то

$$G = \begin{bmatrix} R & t \\ 0 & s^{-1} \end{bmatrix}$$

Таким образом, G – это перемещение и растяжение.

Все вышеназванные группы являются группами Ли. Этот факт активно используется для оптимизации по элементам этих групп, так как в ходе оптимизации можно делать шаги в касательном пространстве являющемся алгеброй Ли, где операция сложения ассоциирована с групповой операцией посредством экспоненциального отображения. Важно отметить, что элементы соотвествующих алгебр Ли представляют из себя минимальную параметризацию элементов группы, а групповая

операция и операция сложения векторов связаны [1]. Будем обозначать ассоциированные с нашими группами алгебры Ли следующим образом: $\mathfrak{so}(3), \mathfrak{se}(3), \mathfrak{sim}(3)$. Экспоненциальное и обратное к нему логарифмическое отображение стандартно обозначим за exp и log без конкретного указания связанной группы.

Для примера, вращения можно параметризовать кватернионами (4 параметра), матрицей (9 параметров), трехмерным вектором (так называемое *axis-angle* представление, где вектор описывает ось вращения, а его длина угол). Последнее представление и есть элемент соотвествующей алгебры Ли[2].

1.2.2. Пространство изображения и связанные операции

- $\Omega \subset \mathbb{R}^2$ — множество нормализованных координат пикселей (пространство изображения).
- Точки изображения мы будем обозначать u, точки 3D-мира p.
- $I : \Omega \to \mathbb{R}^m$ — отображение фотоинтенсивности пикселя и его соседей (при $m > 1$). По умолчанию считаем, что $m = 1$.
- $D : \Omega \to \mathbb{R}_+$ — отображение обратной глубины пикселя. Обычно рассматривается сужение D на некоторое множество ключевых точек. Также за V обозначим дисперсию этой величины. Обычно, если мы говорим про глубину, то будем подразумевать её параметризацию через обратную глубину.
- π — оператор проекции на изображение, т.е. $\pi p = u$. Для обращения операции необходимо знать глубину, поэтому будем писать $\pi^{-1}(u, d_u) = p$.
- ξ_{ij} — элемент $\mathfrak{se}(3)$, ассоцированный с движением $T_{ij} = \exp(\xi_{ij})$ между парой кадров (i, j). При записи с одним индексом (т.е. T_i)

[1] Более подробно см. [15].

[2] О подробной связи алгебр и групп Ли, а так же их примении в задачах компьютерного зрения см. в [27].

будем понимать это как движение относительно некоторой фиксированной мировой системы координат.

- $\omega(u, d, \xi_{ij})$ — оператор проекции точки u изображения i с глубиной d на изображение j.

- $\|\cdot\|_\delta$ — функция потерь Хьюбера с параметром δ, определенная следующим образом:

$$\|x\|_\delta = \begin{cases} \frac{1}{2}[x]^2 & \text{при } |x| \le \delta, \\ \delta(|x| - \delta/2) & \text{иначе.} \end{cases} \tag{3}$$

1.2.3. Метод наименьших квадратов и оценки максимального правдоподобия

Большинство алгоритмов в своей основе используют следующую идею.

Предположим, что у нас есть шумные входные данные Y. Мы хотим найти скрытые внутренние параметры модели X, которые лучше всего объясняют Y. Можно рассмотреть следующую оценку, называемую *оценкой максимального правдоподобия*:

$$\hat{X} := \arg\max_X P(Y|X) \tag{4}$$

Известно, что такие оценки обладают большим количеством положительных свойств [29].

Можно пойти дальше и предположить, что шум во входных данных является гауссовским. В этом случае *оценки в смысле наименьших квадратов* будут являться оценками максимального правдоподобия. Однако, оценка наименьших квадратов не является *робастной*, то есть она не очень устойчива к большим шумам, а распределения ошибок для большинства реальных данных имеют тяжёлые хвосты. С этим можно бороться добавлением устойчивой функции потерь. Поэтому в основе многих методов лежит минимизация некоторого квадратичного функционала энергии, обернутого в робастную оценку (например, хьюберовскую функцию потерь).

В нашем случае Y – это какие-то свойства видеопотока, а X внутренние (фокусное расстояние, дисторсия и другие, если они не известны) и внешние (положение камеры относительно фиксированной мировой системы координат) параметры камеры.

Для численной минимизации наименьших квадратов в основном используются следующие методы:

- алгоритм Гаусса-Ньютона [3, гл. 9]
- алгоритм Левенберга-Марквадта [17, 18]
- IRLS (iteratively reweighted least squares) [3, гл. 4].

2. Алгоритмы VO и VSLAM

2.1. Классификация

Следуя общепринятной терминологии [8], модели и соответствующие им алгоритмы глобально можно разделить по следующим видам (см. рис. 1):

- Косвенные (Indirect)
- Прямые (Direct)
- Разреженные (Sparse)
- Плотные (Dense)
- Гибридные (Hybrid)

Непрямые методы преобразуют данные с камеры в некоторое промежуточное представление, например, детектируя и сопоставляя ключевые точки (как точки некоторых объектов на изображении), или используя какие-то геометрические примитивы и соотношения между ними. Можно сказать, что эти алгоритмы используют семантическую и геометрическую информацию с изображений.

Прямые методы работают с фотометрической информацией, то есть с количеством света (интенсивностью) каждого пикселя. Для прямых методов характерно использование обратной глубины (*inverse depth* [4]) как параметризации 3D-точки.

Разреженные алгоритмы используют небольшое количество точек для оценки параметров модели, и результатом их работы является разреженная реконструкция, в то время как **Плотные** стараются использовать всю информацию с изображения единовременно. Они в итоге дают более детализированный результат. Важной особенностью алгоритмов с плотной структурой является следующее: каждая точка не является независимой от остальных (есть некоторая окрестность связных точек, для которых оценка должна обладать гладкостью). Без учета этого

невозможно получить правильную плотную реконструкцию.

Гибридные методы могут в разной степени сочетать эти подходы.

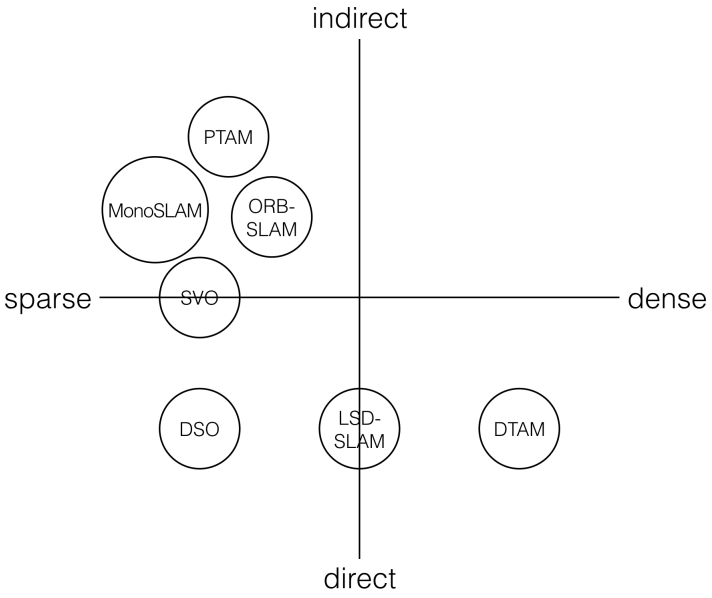


Рис. 2: Классификация алгоритмов VSLAM и VO

2.2. Методы оценки качества модели

Основные методы оценки качества представляют из себя подсчет следующих видов ошибок относительно точных данных [11, 20, 24]:

- Обозначим c_i — точное положение камеры в момент i, $\hat{c}_i$ — оценку положения. Все оценки происходят после выравнивания траекторий к общей системе отсчета в смысле $Sim(3)$ (необходимо сопоставить оцененную траекторию с точной, найдя нужный масштаб и поворот и параллельный перенос).

- Траекторная ошибка (Absolute Trajectory Error) e_{rmse}— среднеквадратичная ошибка (RMSE) отклонений перемещений:

$$e_{rmse} = \sqrt{\min_{G \in Sim(3)} \frac{1}{n} \sum (G\hat{c}_i - c_i)^2} \tag{5}$$

- Ошибка выравнивания e_{align} — оцененная траектория выравнивается относительно начального и конечного отрезка реальной тра-

ектории. Затем считается среднеквадратичное отклонение двух выровненных траекторий:

$$e_{align} = \sqrt{\frac{1}{n}\sum(G_s\hat{c}_i - G_e c_i)^2}, \tag{6}$$

где G_s минимизирует e_{rmse} для начального отрезка траектории, G_e для конечного.

- Накопленная ошибка движения $e_{drift} = \left\|G_e(G_s^{-1})\right\|$.
- Накопленная ошибка смещения e_t.
- Накопленная ошибка вращения e_r. Мы считаем эту ошибку в градусах.

Для отображения на графиках мы будем использовать следующую нотацию:

- Синий — оцененная траектория масштабно выравненная относительно конца или полной траектории.
- Желтый — оцененная траектория масштабно выравненная относительно начала.
- Зеленый — первая половина правильной траектории.
- Черный — вторая половина правильной траектории.
- Красным будем выделять примеры оценки траектории, когда алгоритм не смог корректно завершить работу.

2.3. Обзор данных для экспериментов

Для экспериментов мы используем следующие наборы данных:

- Tum Mono Dataset [25].
- KITTI Dataset [13].
- ApolloScape Dataset [1].

2.3.1. Tum Mono Dataset

Набор стостит из 50 различных последовательностей, снятых в помещении и вне помещения. Все последовательности начинаются и заканчиваются в одной точке, что позволяет считать накопленную ошибку, зная только несколько первых и последних позиций, поэтому аннотация содержит информацию только о нескольких позах. В следствие этого ошибка RMSE не слишком показательна, так как гарантированно будет маленькой.

В отличие от многих других наборов данных, авторы Tum Mono Dataset предоставляют фотометрическую калибровку (её напрямую использует алгоритм DSO).

Мы выбрали несколько последовательностей различной длины и сложности из всего набора (несколько в помещении, несколько вне).

Сложность последовательности определяется её длиной, наличием резких переходов, сложного окружения, частотой кадров, а также разбросом экспозиции (авторы предоставляют эту информацию):

	FPS	Длительность	Мин. экспозиция	Макс. экспозиция	В помещении
seq 09	50.04	46 сек.	5.03 мс.	15.03 мс.	Да
seq 13	24.99	72 сек.	0.47 мс.	32.13 мс.	Да
seq 17	40.03	124.39 сек.	0.34 мс.	2.37 мс.	Да
seq 22	20.00	316.98 сек.	0.06 мс.	0.28 мс.	Нет
seq 30	21.09	85.30 сек.	0.01 мс.	0.21 мс.	Нет
seq 50	25.13	161.12 сек.	0.27 мс.	1.65 мс.	Нет

2.3.2. KITTI Dataset

Набор представляет из себя последовательности изображений, полученных стереокамерой (мы используем изображения только с одного объектива), размещенной на автомобиле, поэтому все последовательности сняты вне помещений. Камеры откалиброваны, имеется информация о позициях, полученная с помощью LiDAR и GPS датчика. Никакая другая информация не доступна.

Все последовательности сняты со скоростью 10 FPS. Предоставленные изображения ректифицированы, для них так же проведено устранение дисторсии (впрочем, в Raw KITTI доступны изображения и без этого).

Для тестирования мы выбрали последовательности под номерами 1, 2, 3, 5, 6 и 8.

2.3.3. ApolloScape

Наиболее свежий набор данных (2018 год), предназначеный в первую очередь для задач детекции и сегментации (имеются соотвествующие аннотации). Авторы, однако, предоставляют информация о позициях и калибровку камеры, что позволяет использовать его для измерения алгоритмов одометрии.

Последовательности получены с помощью нескольких камер (мы используем только одну), размещенных на автомобиле. Вся система для получения аннотаций использует лазерный сканер, а так же IMU и GNSS. Все последовательности слишком коротки для проверки одометрии, однако, каждая следующая последовательность является продолжением предыдущей, поэтому мы совмещаем несколько подряд идущих последовательностей в одну, а затем проводим измерения. Мы взяли последовательности 1, 2, 3, 4, 5, 18, 19, 20, 21 и разделили их на четыре последовательности разной длины.

2.3.4. Эксперименты

Перед запуском на выбранных последовательностях каждый из алгоритмов был успешно запущен на демонстрационных примерах. После этого для каждой последовательности было произведено несколько попыток запуска (не всегда удачных) каждого из алгоритмов, для которого затем выбирался наилучший из полученных ответов.

Так как все представленные алгоритмы являются алгоритмами реального времени, для тестирования было достаточно мощности ноутбка с процессором Intel(R) Core(TM) i7-3537U CPU @ 2.00GHz и 8 GB опе-

ративной памяти.

2.4. LSD-SLAM

Алгоритм *Large-Scale Direct* SLAM [7] делится на три этапа (см. рис. 3):

1. Отслеживание кадров
2. Оценка глубин
3. Sim(3) ограничение и оптимизация карты территории

Разберем каждый из этапов подробно.

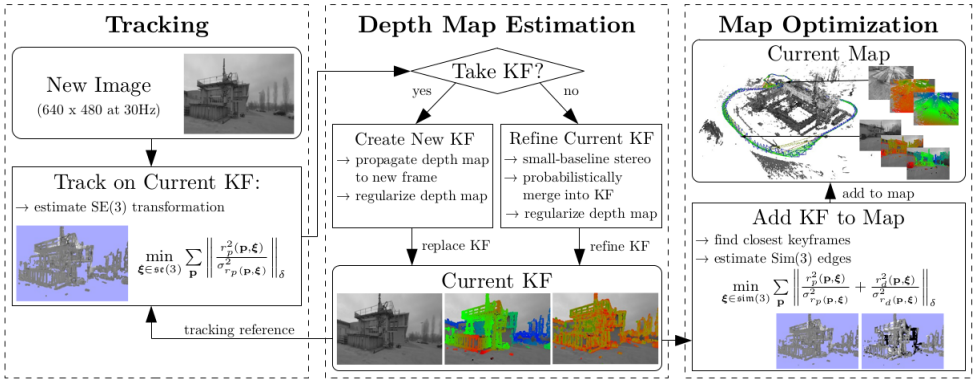


Рис. 3: Структура LSD SLAM (изображение из оригинальной статьи)

2.4.1. Отслеживание кадров

Введём фотометрическую ошибку относительно текущего ключевого кадра (KF):

$$E(\xi) = \sum_{u \in \Omega_{KF}} (I_{KF}(u) - I(\omega(u, D_{KF}(u), \xi)))^2 := \sum_{u} r^2(u, \xi). \tag{7}$$

Как выше уже было отмечено, если предполагать, что r гауссовские, то мы получим оценку максимального правдоподобия для ξ. Аналогичные рассуждения применяются и в случае других квадратичных функционалов энергии. Для самого первого кадра необходима инициализация

карты глубин, авторы предлагают делать это случайно с большой дисперсией. Если в течение первых секунд движение не будет вырожденным, то после нескольких ключевых кадров алгоритм начнёт сходиться. Далее отслеживание устроено следующим образом:

1. На текущем шаге имеется ключевой кадр $\mathcal{K} = (I_i, D_i, V_i)$. Для заданного ξ можно посчитать дисперсию $\sigma_r^2(u, \xi)$ с помощью метода *распространения неопределенности*[3], шум для интенсивности пикселя мы предполагаем гауссовским. Обратим внимание на то, что D_i и V_i определены не на всём кадре, а на некотором его подмножестве Ω_{D_i}, покрывающем, однако, всё изображение. Таким образом, алгоритм относится к классу *semi-dense*.

2. Для очередного кадра j и текущего ключевого кадра i рассмотрим следующую задачу минимизации дисперсионно-нормализованной фотометрической ошибки:

$$\min_{\xi_{ji} \in \mathfrak{se}(3)} E(\xi_{ji}) = \sum_u \left\| \frac{r^2(u, \xi_{ji})}{\sigma_r^2(u, \xi_{ji})} \right\|_\delta . \tag{8}$$

   Для увеличения робастности используется функции потерь Хьюбера с некоторым параметром δ.

3. Оптимизация проводится с помощью взвешенного метода Ньютона-Гаусса с итеративным пересчетом весов (IRLS).

Таким образом, получается оценка на ξ_{ji}. Как отмечают авторы, одна из особенностей этого метода в том, что он использует дисперсию шума карты глубин текущего ключевого кадра. Это важно, так как для монокулярной камеры (в отличие от, например, RGBD) шум глубины на каждом из пикселей сильно варьируется.

Отметим, что на данном этапе никак не используется карта глубин очередного кадра (т.к. она еще не известна). Следующий этап состоит в том, чтобы ее оценить.

[3]Подробнее см. [7, стр. 6].

2.4.2. Оценка глубин

Если камера сдвигается слишком далеко от текущего ключевого кадра, создается новый ключевой кадр из текущего изображения. Граница отсечения определяется для взвешенной комбинации относительного расстояния и угла поворота:

$$\text{dist}(\xi_{ij}) := \xi_{ij}^T W \xi_{ij}, \tag{9}$$

где W диагональная матрица весов.

Возможны два варианта:

- Мы создаем новый ключевой кадр из текущего.
- Мы оптимизируем текущий ключевой кадр из текущего.

В первом случае, текущий ключевой кадр со всей оцененной информацией (I_i, D_i, V_i) добавляется в граф ключевых кадров. Затем ключевые точки со старых и достаточно близких ключевых кадров проецируются с помощью ω на новый ключевой кадр. После этого происходит одна итерация пространственной оптимизации и отсечение аутлаеров. Затем карта глубин нормируется, чтобы средняя обратная глубина равнялась единице. Полученный множитель можно внести в оценку ξ_{ij}, так как движение определено с точностью до длины перемещения. Во втором случае, так как мы посчитали перемещение достаточно маленьким, можно считать, что некоторые регионы текущего и ключевого кадра представляют пару изображений со стереокамеры, то есть для них возможна оценка глубины. В этом случае авторы действуют аналогично способу, описанному в [10].

2.4.3. Sim(3) ограничение и оптимизация карты территории

Так как данный алгоритм решает задачу SLAM, то важным моментом является замыкание циклов и устранение плавающего масштаба. Для этого необходима оптимизация в смысле $Sim(3)$. Авторы рассматривают следующую задачу минимизации для оценки перемещения и

масштаба:

$$\min_{\xi_{ji}\in\mathfrak{sim}(3)} \sum_{u\in\Omega_{D_i}} \left\| \frac{r^2(u,\xi)}{\sigma_r^2(u,\xi)} + \frac{r_d^2(u,\xi)}{\sigma_{r_d}^2(u,\xi)} \right\|_\delta , \tag{10}$$

где

$$r_d(u,\xi) := [p']_3 - D_j([p']_{1,2}),$$
$$p' = \omega(p, D_i(p), \xi)$$

Таким образом, в дополнение к фотометрической ошибке, учитывается ошибка глубины. Это необходимое добавление, так как сама по себе фотометрическая ошибка не накладывает ограничение на масштаб. С точки зрения вычислений задача не сильно отличается от задачи (8), снова используется метод Ньютона-Гаусса с пересчётом весов.

Мы не будем подробно останавливаться на поиске циклов, однако, отметим, что множество всех ключевых кадров представленно в виде графа с весами из $Sim(3)$. В нем ищутся циклы, а его веса дополнительно оптимизируются с помощью g2o [28], осуществляя глобальную оптимизацию карты.

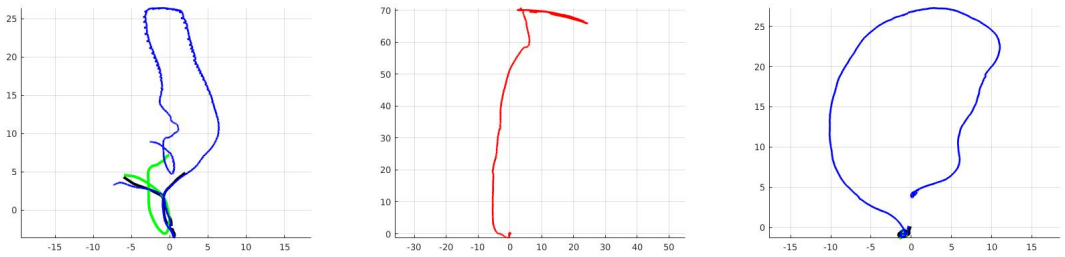
2.4.4. Полученные результаты

Открытая реализация алгоритма содержала некоторые ошибки аллокации для объектов с выравниванием [6], которые не позволяли запустить его на некоторых системах (в частности, последней версии Debian). После внесения некоторых исправлений удалось убедиться, что алгоритм работает на демо примерах.

Рассмотрим полученные результаты.

Tum Mono	e_{rmse}	e_{align}	e_r
seq 09	2.4458	8.2362	4.6625
seq 13	0.8970	42.3373	110.143
seq 17	0.1218	20.1101	30.1243
seq 22	—	—	—
seq 30	0.5228	24.5848	7.5848
seq 50	—	—	—

Рис. 4: Пример результатов



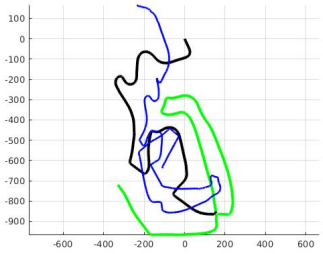
(a) LSD Mono seq. 9 (b) LSD Mono seq. 22 (c) LSD Mono seq. 30

На Tum Mono LSD SLAM показывает себя удовлетворительно. Мы, однако, имеем очень большую ошибку выравнивания, например, её хорошо видно на изображении 4a, где оцененная траектория близка к действительной, но по каким-то причинам нарушается относительный масштаб сегментов, что приводит к невозможности найти единое преобразование масштаба. В целом это характерно для алгоритмов VO на монокулярных камерах, так как масштаб смещения качественно определить невозможно. На самой сложной последовательности (22) алгоритм не смог продолжить свою работу (рис. 4b).

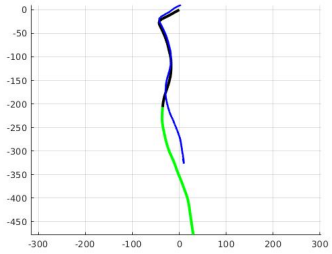
Средние результаты можно списать на то, что этот алгоритм предполагает постоянность экспозиции в то время, как для данного набора данных выдержка сильно колеблется.

Следующим набором данных, на котором тестировался алгоритм, был KITTI. Для данного алгоритма это сложные данные, так как поток изображений снят с низкой частотой, а этот алгоритм полагается на частоту хотя бы 30 FPS (KITTI предоставляет данные с частотой 10 FPS). Кроме того для KITTI калибровка, устранение дисторсии и ректификация изображений оставляют желать лучшего. Для улучшения качества результатов можно попытаться воспользоваться изображениями без устраненной дисторсии (так называемый RAW KITTI Dataset), а параметры калибровки определить другими средствами.

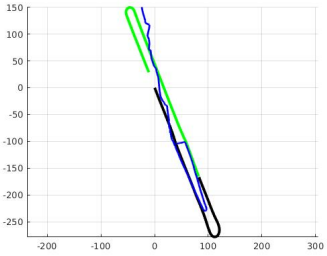
Рис. 5: Пример результатов



(a) LSD KITTI seq. 2



(b) LSD KITTI seq. 3



(c) LSD KITTI seq. 6

KITTI	e_{rmse}	e_{align}	e_r
seq 01	246.5280	604.7014	12.7058
seq 02	284.6900	520.7498	19.8548
seq 03	24.4326	113.2107	101.9944
seq 05	—	—	—
seq 06	92.5036	249.0223	48.1927
seq 08	223.0577	483.1592	111.8611

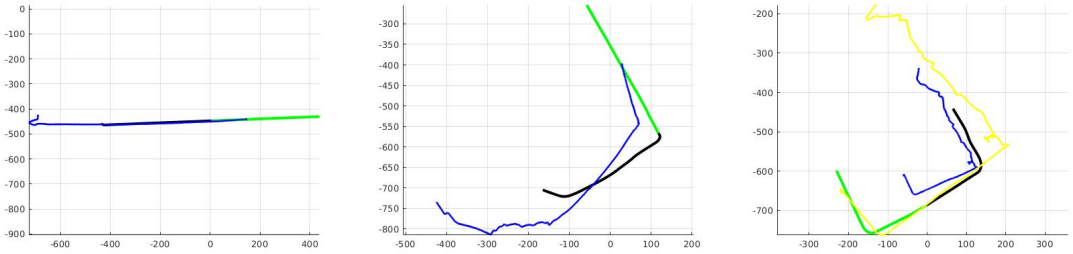
Алгоритм дает не совсем состоятельные результаты, а даже там, где последовательность простая и результат в целом удовлетворительный, видно, что происходит сильное накопление ошибки масштаба и части траектории масштабно не согласуются друг с другом (см. рисунок 7b). То есть в целом форма траектории может быть похожей на ту, которая должна быть, но выравнивания относительно начала и конца не согласуются друг с другом (см. 7a). Возможные причины мы рассмотрели выше.

Для Apollo получились схожие результаты с той же проблемой масштаба начального и конечного отрезка.

Apollo	e_{rmse}	e_{align}	e_r
seq 01	142.2548	205.53	32.43
seq 02	78.9420	480.0863	114.23
seq 03	55.2656	124.3749	33.1007
seq 04	102.773	103.23	122.957

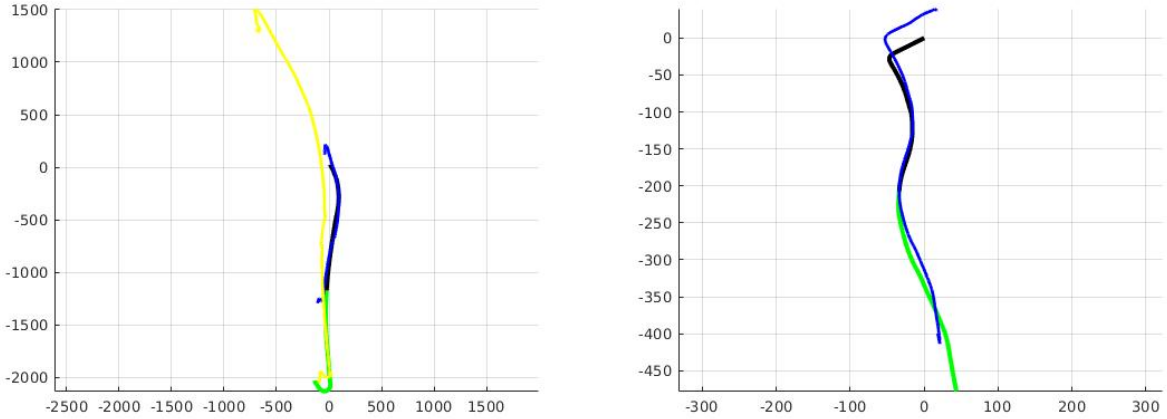
Стоит отметить, что существует расширение этого алгоритма для

Рис. 6: Пример результатов



(a) LSD Apollo seq. 1 (b) LSD Apollo seq. 2 (c) LSD Apollo seq. 3

Рис. 7: Выравнивание траектории



(a) Два выравнивания (b) Полное выравнивание

стереокамер (а также вариант для камер с большим углом обзора типа fisheye), но авторы не предоставляют открытых реализаций. Стереокамера позволяет точно оценивать масштаб смещения, поэтому расширенный алгоритм показывает гораздо лучшие результаты, в частности, имеет высокое место в score board KITTI.

2.5. Semi-direct Visual Odometry

SVO [12] в отличие от LSD-SLAM является алгоритмом только визуальной одометрии. Этот алгоритм характеризует то, что на первом этапе он минимизирует фотометрическую ошибку, а затем, используя полученные оценки на движение и 3D-2D соответствия, минимизирует геометрическую ошибку репроекции. Происходит это в несколько эта-

пов:

1. Ошибка минимизируется только по движению (*motion only Bundle Adjustment*).

2. Ошибка минимизируется только по 3D точкам (*struct only Budnle Adjustment*).

3. Ошибка минимизируется по всем неизвестным параметрам (*совместный Bundle Adjustment*).

Можно выделить два параллельных этапа (см. рис. 8):

- Оценка движения
- Оценка карты глубин

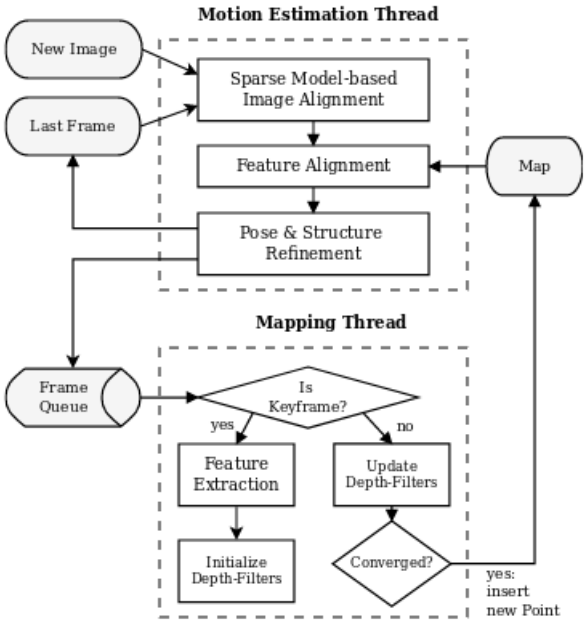


Рис. 8: Структура SVO (изображение из оригинальной статьи)

Первые два кадра используются для начальной инициализации карты глубин и движения. В каждый момент времени, используется ограниченное число ключевых кадров. Очередной кадр добавляется в список ключевых, если евклидово расстояние относительно всех ключевых кадров превышает 12% от средней глубины сцены. При превышении порога количества ключевых кадров самый дальний из них удаляется.

2.5.1. Оценка движения

Введём фотометрическую ошибку для пары соседних кадров:

$$\Delta I(T_{k,k-1}) = \sum_{u \in \mathcal{R}} \left\| I_k \left(\pi(T_{k,k-1}\pi^{-1}(u, d_u)) \right) - I_{k-1}(u) \right\|^2 = \sum_{u \in \mathcal{R}} \left\| \delta I(T_{k,k-1}, u) \right\|^2 . \tag{11}$$

где $\mathcal{R}$ – множество точек, для которых известна глубина в момент времени $k-1$, и они видны в момент времени k, а I – вектор-функция для 4×4 пиксельной области с центром в u. Рассмотрим задачу минимизации:

$$\min_{T_{k,k-1}} \frac{1}{2} \Delta I(T_{k,k-1}). \tag{12}$$

Для решения этой задачи в реализации опять таки используется метод Ньютона-Гаусса.

Из-за ошибок в предыдущих позах и неточностях в оценках 3D точек полученная оценка не является оптимальной, однако, её можно улучшить следующим образом. Необходимо рассмотреть все видимые в момент k 3D точки $\{p_i\}$ уже построенной карты. Проецируя их на текущее изображение получится набор ключевых точек $\{u'_i\}$. Далее, можно найти предыдущий ключевой кадр r, наблюдающий p_i с близким углом обзора (на нем точке u'_i соответствует некоторая точка u_i). После этого значения точек на текущем кадре $\{u'_i\}$ индивидуально оптимизируются в смысле фотометрической ошибки:

$$\min_{u'_i} \frac{1}{2} \left\| I_k(u'_i) - A_i I_r(u_i) \right\|^2 , \tag{13}$$

где A_i – соотвествующее аффинное преобразование между областями. В отличие от (11), где движение между кадрами очень маленькое и преобразование можно игнорировать, в ситуации (13) мы сделать этого не можем, т.к. кадр r скорее всего будет находится намного дальше предыдушего $k-1$. Кроме того, в этой ситуации берётся большая пиксельная область (8×8). Эту задачу можно решить с помощью обратного алгоритма Лукаса-Канаде [2]. Стоит отметить, что при такой оптими-

зации нарушается *эпиполярное ограничение*[4] между u_i и u'_i, так как мы совсем не берем его во внимание. Это приводит к появлению ошибок репроекции, но взамен мы получили большую фотометрическую согласованность между соотвествующими пиксельными областями, перенеся неопределенность в оценки перемещения. Таким образом, следующим шагом необходимо устранить обретённую ошибку репроекции:

$$\min_{T_k} \frac{1}{2} \sum_{u_i} \|u_i - \pi(T_k p_i)\|^2 . \tag{14}$$

Данная задача известна как *motion only Bundle Adjustment* и эффективно решается методом Ньютона-Гаусса. Далее, для этой же ошибки оптимизируются не только позы, но триангулированные 3D точки p_i, а затем все совместно (в целях ускорения можно отключить этот этап).

2.5.2. Оценка карты глубин

Авторы используют следующий вероятностный подход, описанный в [26]. Опишем его идею, не вдаваясь в тонкости. Возможны два варианта:

- Очередной кадр добавляется в список ключевых. Тогда с помощью FAST [23] из него извлекаются ключевые точки, а его карта глубин некоторым образом инициализируется распределением с большой дисперсией и математическим ожиданием равным средней глубине ключевого кадра. Оценка глубины моделируется смесью нормального и равномерного распределения. Первое из них отвечает за моделирование *инлаеров*[5], второй за моделирование *аутлаеров*[6]. Параметры смеси (вероятность аутлаеров, дисперсия нормального распределения и т.д.) в дальнейшем пересчитываются за счёт вновь прибывших кадров, которые мы не посчитали ключевыми.

- Кадр k, который не добавляется в список ключевых, используется для оптимизации глубины ключевых кадров r. Для каждой клю-

[4]Определение см. в [14, стр. 239].

[5]Значения, которые хорошо соотвествуют текущим наблюдениям

[6]Значения, которые плохо соотвествуют текущим наблюдениям

чевой точки u_i считается *эпиполярная прямая*[7] на кадре k, на ней находится точка u'_i, такая что область (4x4 или 8x8) с центром в u_i имеет максимальную корреляцию с соотвествующей областью вокруг u'_i (см. рис. 9). Используется при этом не вся эпиполярная прямая на кадре k, а только такие точки на ней, что при триангуляции полученная глубина будет лежать в не более чем двух стандартных отклонениях от текущей оценки. Полученная пара триангулируется, что в итоге дает новую оценку на глубину. После этого параметры смеси пересчитываются. Соотвествующую точку u'_i не всегда удается корректно найти (например, эту точку не видно на изображении k, или измерение является аутлаером). Если не удалось, то вероятность аутлаеров увеличивается и параметры смеси опять же пересчитываются. В какой-то момент, если дисперсия распределения достаточно мала, то можно выбрать моду, как хорошую оценку для глубины, и считать, что оценка для данной точки сошлась. Соотвествующую u_i 3D точку p_i, посчитанную с помощью π и оцененной глубины добавляют в глобальную карту, которая используется для оптимизации на этапе оценки передвижения.

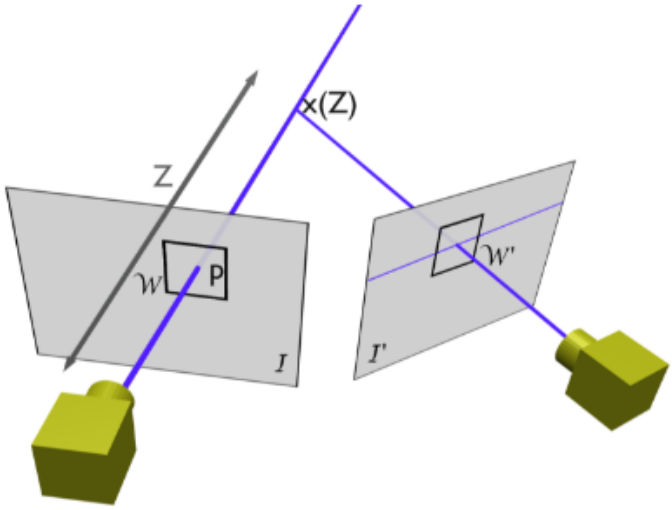


Рис. 9: Поиск точки на эпиполярной прямой (оригинальное изображение [26])

[7]Определение см. [14].

2.5.3. Полученные результаты

Изначальная версия алгоритма, а также доступная открытая реализация этого алгоритма предназначена для камер, смотрящих вниз, то есть для таких, у которых средняя глубина пикселя не очень большая. Как отметили авторы, при невыполнении этих условий алгоритм может вести себя довольно плохо.

Открытая реализация также страдала от большого количества ошибок, связанных с аллокацией и выравниванием объектов. Удалось исправить часть из них и запустить алгоритм на тестовом примере, а также на небольшом наборе последовательностей (LSD Room из демо для LSD SLAM), в которых нет точек с большой глубиной.

Попытка запустить на выбранных нами последовательностях не увенчалась успехом вследствие падения алгоритма с неясной ошибкой выделения памяти.

Авторы предоставляют исправленную версию алгоритма SVO 2.0, который можно использовать для любых камер и который не страдает от большинства ошибок, однако, его реализация закрыта, и для него доступен только бинарный файл. Мы не стали им пользоваться из-за возникших сложностей, а так же по причине, что изначальная цель была исследовать доступные реализации алгоритмов.

2.6. Direct Sparse Odometry

Наиболее современный из рассматриваемых метод [8]. Одна из его особенностей в том, что он максимально использует фотометрические параметры камеры.

Рассмотрим отображение интенсивности следующего вида $I'(u) = G(tV(u)B(u))$, где t — выдержка, G — функция гамма-коррекции, V — функция затухания объектива, B — интенсивность излучения. Для каждого кадра проведем коррекцию по G и V (если они известны, то есть имеется фотокалибровка), тогда функция интенсивности пикселя на скорректированном изображении имеет вид $I(u) = tB(u)$.

Введём следующую взвешенную фотометрическую ошибку пары изоб-

ражений (i,j) для пиксельной области $\mathcal{N}_u$ (см. рис 10):

$$E_{u,ij} = \sum_{u \in \mathcal{N}_u} w_u \left\| I_j \left(\pi(T_{i,j} \pi^{-1}(u, d_u)) \right) - b_j - \frac{t_j e^{a_j}}{t_i e^{a_i}} (I(u) - b_i) \right\|_\delta , \qquad (15)$$

где t_i, t_j — выдержка соответсвенно i и j кадра, a_i, a_j, b_i, b_j — параметры аффинной коррекции яркости (нужна, если выдержка неизвестна), а веса имеют вид:

$$w_u = \frac{c^2}{c^2 + \|\Delta I_i(u)\|^2}. \qquad (16)$$

Обратим внимание, что за счет использования аффинной коррекции данная постановка (в отличие от двух предыдущих алгоритмов, хотя в версии LSD SLAM для стереокамер это тоже учитывается [9]) отказывается от условия *постоянства яркости* (которое, на самом деле, очень сильно нарушается для среды вне помещения). Экспонециальная форма параметра a сделана для удобства оптимизации по нему (таким образом контролируется, что этот коэффициент масштаба аффинного преобразования не будет отрицательным). Полная фотометрическая ошибка тогда имеет вид:

$$E_{photo} = \sum_{i \in \mathcal{F}} \sum_{u \in \mathcal{P}_i} \sum_{j \in obs(u)} E_{u,ij}, \qquad (17)$$

где $\mathcal{F}$ — текущее окно ключевых кадров, $\mathcal{P}$ — ключевые точки, $obs(u)$ — все кадры, на которых видна u.

Если выдержка известна, то к ошибке надо добавить слагаемое:

$$E_{prior} = \sum_{i \in \mathcal{F}} (\lambda_a a_i^2 + \lambda_b b_i^2), \qquad (18)$$

которое нейтрализует a_i и b_i в E_{photo}. Обозначим за E итоговую ошибку.

При отсутствии фотометрической калибровки в параметры устанавливаются следующие значения $\lambda_a = 0$, $\lambda_b = 0$ и $t_i = 1$.

Далее, рассматривается следующая задача оптимизации:

$$\min_{\zeta \in SE(3) \times \mathbb{R}^m} E, \qquad (19)$$

где часть $\mathbb{R}^m$ содержит в себе оценки глубин, оценки внутренних параметров камеры, а так же коэффициенты аффинной коррекции. Таким образом, важной особенностью алгоритма является то, что он оптимизирует все параметры сразу совместно.

Как отмечают авторы, это единственный прямой метод, который совместно оптимизирует оценку правдоподобия сразу по всем параметрам (позы, глубина и внутренние параметры камеры).

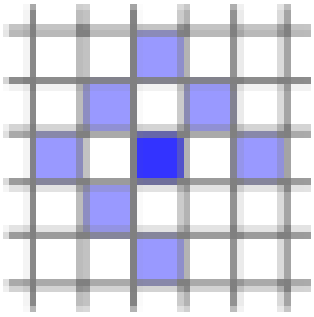


Рис. 10: Residual pattern, эксперементально выяснено, что это наиболее оптимальный паттерн

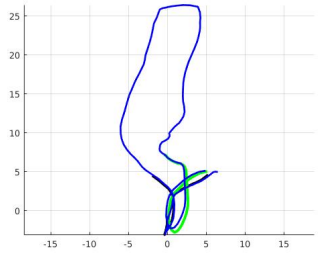
2.6.1. Полученные результаты

Рассмотрим результаты работы алгоритма на Tum Mono. Напомним, что для этих данных представлены только начало и конец точной траектории, поэтому наиболее показательна здесь ошибка выравнивания.

Tum Mono	e_{rmse}	e_{align}	e_r
seq 09	0.3125	0.7458	3.3117
seq 13	0.4910	1.4283	3.6018
seq 17	0.1200	2.6666	0.7298
seq 22	0.1195	5.2554	3.1660
seq 30	0.4028	1.0057	1.399
seq 50	0.1692	1.1151	1.5419

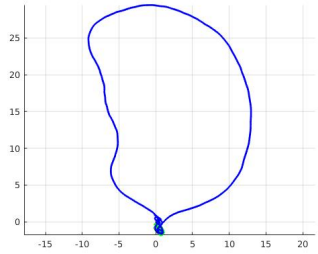
Последовательность 22 очень длинная и сложная, поэтому для неё результат ожидаемо оказался не идеальным, но видно, что в целом алгоритм показывает хорошие результаты.

Рис. 11: Пример результатов



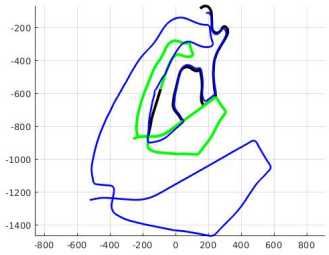
(a) DSO Mono seq. 9

(b) DSO Mono seq. 22

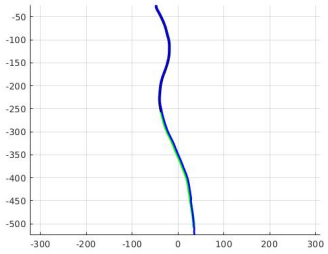


(c) DSO Mono seq. 30

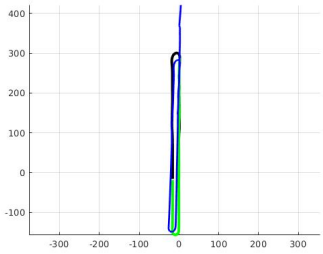
Рис. 12: Пример результатов



(a) DSO KITTI seq. 2



(b) DSO KITTI seq. 3



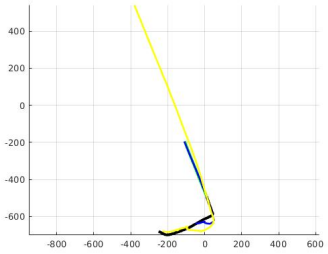
(c) DSO KITTI seq. 6

На KITTI алгоритм показал следующие неплохие результаты, в некоторых случаях сильно ошибаясь с масштабом (см. рис. 12a).

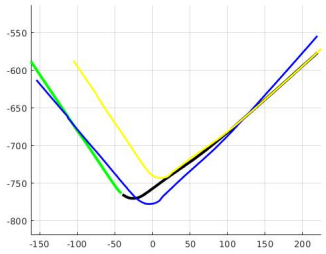
KITTI	e_{rmse}	e_{align}	e_r
seq 01	28.9490	65.9491	4.9415
seq 02	106.7573	288.6369	33.5160
seq 03	2.7147	7.8551	6.0021
seq 05	47.2962	109.5652	7.8612
seq 06	40.9388	73.9968	3.5570
seq 08	111.2177	289.8554	16.7188

С Apollo у алгоритма были трудности, в частности, при некоторых запусках алгоритм сбивался и прекращал работу. В некоторых случаях, однако, ему удавалось посчитать довольно приличный результат. Но в целом результаты нельзя назвать очень хорошими. В частности, в некоторых случаях алгоритм сильно ошибался в масштабе частей даже

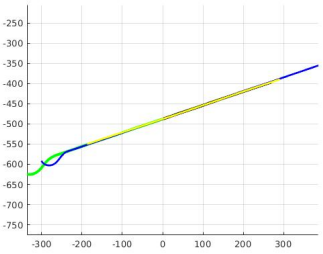
Рис. 13: Пример результатов



(a) DSO Apollo seq. 2



(b) DSO Apollo seq. 3



(c) DSO Apollo seq. 4

на короткой дистанции. С такими примерами можно ознакомиться на рис. 13a и 13c.

Apollo	e_{rmse}	e_{align}	e_r
seq 01	39.0464	169.705	89.8439
seq 02	54.893	381.9656	26.4838
seq 03	14.8912	40.354	8.9602
seq 04	20.4381	66.823	85.3456

3. Анализ результатов

Посмотрим на сводные результаты, где A, K, M — сокращения от Apollo, KITTI и Mono соответсвенно, e_1 — ошибка для DSO, e_2 — ошибка для LSD SLAM:

#	$e_{1,rmse}$	$e_{1,align}$	$e_{1,r}$	$e_{2,rmse}$	$e_{2,align}$	$e_{2,r}$
A 01	39.0464	169.705	89.8439	142.2548	205.53	32.43
A 02	54.893	381.9656	26.4838	78.9420	480.0863	114.23
A 03	14.8912	40.354	8.9602	55.2656	124.3749	33.1007
A 04	20.4381	66.823	85.3456	102.773	103.23	122.957
K 01	28.9490	65.9491	4.9415	246.5280	604.7014	12.7058
K 02	106.7573	288.6369	33.5160	284.6900	520.7498	19.8548
K 03	2.7147	7.8551	6.0021	24.4326	113.2107	101.9944
K 05	47.2962	109.5652	7.8612	—	—	—
K 06	40.9388	73.9968	3.5570	92.5036	249.0223	48.1927
K 08	111.2177	289.8554	16.7188	223.0577	483.1592	111.8611
M 09	0.3125	0.7458	3.3117	2.4458	8.2362	4.6625
M 13	0.4910	1.4283	3.6018	0.8970	42.3373	110.143
M 17	0.1200	2.6666	0.7298	0.1218	20.1101	30.1243
M 22	0.1195	5.2554	3.1660	—	—	—
M 30	0.4028	1.0057	1.399	0.5228	24.5848	7.5848
M 50	0.1692	1.1151	1.5419	—	—	—

Как видно, DSO обходит LSD SLAM на всех последовательностях и по всем метрикам. Также отметим, что в большинстве случаев LSD показывает довольно плохие результаты, связанные, в основном, с нарушением относительного масштаба частей, мы демонстрировали это на изображениях траекторий в соотвествующих частях этой работы. Хотя в целом траектории и выглядят похоже на необходимые, выбранные нами метрики (кроме RMSE) очень сильно зависят от соотношения частей (а масштаб – это все-таки слабейшая часть монокулярных алгоритмов), возможно для других метрик разница между DSO и LSD SLAM была бы не столь фатальна.

Причин почему LSD показывает себя не очень хорошо видится несколько. Во-первых, выбранные датасеты сложны для этого алгоритма, так как сильно нарушают его предположения (постоянство экспозиции, высокая частота кадров, очень точная калибровка), во вторых есть вероятность, что открытая реализация (хотя и показывающая себя хорошо на демо примерах) содержит какие-то ошибки, которые проявляют себя на последовательностях, сложных для алгоритма. Такие выводы можно сделать исходя из того, что на некоторых (особенно сложных) последовательностях алгоритм завершает работу аварийно, но не всегда по ясным причинам.

Тем не менее, это не умаляет важности алгоритма LSD, так как в дальнейшем он послужил основой для стерео версии, которая имеет доказанную эффективность для большинства данных. Кроме того, часть идей из LSD SLAM присутствует и в DSO.

Как отмечено ранее, нам не удалось запустить SVO на выбранных датасетах, поэтому мы не проводим анализ этого алгоритма.

3.1. Выводы

Алгоритм DSO ожидаемо показал лучшие результаты. Как и отмечали авторы он налагает гораздо менее сильные ограничения на данные. Хочется отметить, что при всём этом LSD показал вполне нормальные результаты, учитывая что алгоритм более старый и тестировался на непростых последовательностях. Увы, мы не можем ничего сказать про алгоритм SVO, кроме того, что имеющая реализация показалась нам не очень удовлетворительной (на данный момент она заброшена автором и не прогрессирует). Разумеется, одна из причин его неработоспособности в том, что изначальная реализация предполагает небольшую глубину кадра, но нам не удалось получить результаты даже для последовательностей в помещении. И после изучения кода было непонятно, почему алгоритм не может даже инициализировать свою работу. С другой стороны пропиретарная версия, которую удалось проверить, но для которой не были проведены полноценные тесты, оказалась до-

вольно качественной. Так что мы спишем проблемы SVO на недостаточную проработку при реализации.

Заключение

В ходе работы удалось разобраться с современными методами в задачах VO, изучить их открытые реализации и найти там ошибки, а так же познакомиться с различными наборами данных, из которых хочется отметить недавновышедший ApolloScape, так как до этого он не применялся (в литературе) для оценки качества алгоритмов одометрии.

В заключение хочется сказать, что оценка точности методов SLAM и VO является непростой задачей, так как разные алгоритмы работают при разных условиях, и небольшое их нарушение приводит к плачевным результатам. Это в основном касается монокулярных алгоритмов, в то время как алгоритмы, работающие со стереопарой, не столь подвержены сильным колебаниям в качестве.

Также я пришел к выводу, что открытые реализации требуют серьезной доработки для того, чтобы быть использованными реальных задачах.

Список литературы

[1] The ApolloScape Dataset for Autonomous Driving / Xinyu Huang, Xinjing Cheng, Qichuan Geng et al. // arXiv preprint arXiv:1803.06184. — 2018.

[2] Baker Simon, Matthews Iain. Lucas-kanade 20 years on: A unifying framework // International journal of computer vision. — 2004. — Vol. 56, no. 3. — P. 221–255.

[3] Bjorck Ake. Numerical methods for least squares problems. — Siam, 1996.

[4] Civera Javier, Davison Andrew J, Montiel JM Martinez. Inverse depth parametrization for monocular SLAM // IEEE transactions on robotics. — 2008. — Vol. 24, no. 5. — P. 932–945.

[5] Cole David M, Newman Paul M. Using laser range data for 3D SLAM in outdoor environments // Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on / IEEE. — 2006. — P. 1556–1563.

[6] Eigen alignement and allocation issues.

[7] Engel Jakob, Cremers Daniel. LSD-SLAM: Large-scale direct monocular SLAM. — 2014.

[8] Engel J., Koltun V., Cremers D. Direct Sparse Odometry // ArXiv e-prints. — 2016. — 1607.02565.

[9] Engel Jakob, Stückler Jörg, Cremers Daniel. Large-scale direct SLAM with stereo cameras // Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on / IEEE. — 2015. — P. 1935–1942.

[10] Engel Jakob, Sturm Jürgen, Cremers Daniel. Semi-dense visual odometry for a monocular camera // Computer Vision (ICCV), 2013 IEEE International Conference on / IEEE. — 2013. — P. 1449–1456.

[11] Engel Jakob, Usenko Vladyslav, Cremers Daniel. A photometrically calibrated benchmark for monocular visual odometry // arXiv preprint arXiv:1607.02555. — 2016.

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

[13] Geiger Andreas, Lenz Philip, Urtasun Raquel. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite // Conference on Computer Vision and Pattern Recognition (CVPR). — 2012.

[14] Hartley Richard, Zisserman Andrew. Multiple view geometry in computer vision. — Cambridge university press, 2003.

[15] Hertzberg Christoph. A framework for sparse, non-linear least squares problems on manifolds // UNIVERSITÄT BREMEN / Citeseer. — 2008.

[16] IMU-based localization and slip estimation for skid-steered mobile robots / Jingang Yi, Junjie Zhang, Dezhen Song, Suhada Jayasuriya // Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on / IEEE. — 2007. — P. 2845–2850.

[17] Levenberg Kenneth. A method for the solution of certain non-linear problems in least squares // Quarterly of applied mathematics. — 1944. — Vol. 2, no. 2. — P. 164–168.

[18] Marquardt Donald W. An algorithm for least-squares estimation of nonlinear parameters // Journal of the society for Industrial and Applied Mathematics. — 1963. — Vol. 11, no. 2. — P. 431–441.

[19] Monocular visual odometry using a planar road model to solve scale ambiguity / Bernd Manfred Kitt, Joern Rehder, Andrew D Chambers et al. — 2011.

[20] On measuring the accuracy of SLAM algorithms / Rainer Kümmerle, Bastian Steder, Christian Dornhege et al. // Auton. Robots. — 2009. — Vol. 27, no. 4. — P. 387–407.

[21] Radar scan matching SLAM using the Fourier-Mellin transform / Paul Checchin, Franck Gérossier, Christophe Blanc et al. // Field and Service Robotics / Springer. — 2010. — P. 151–161.

[22] Real-time loop closure in 2D LIDAR SLAM / Wolfgang Hess, Damon Kohler, Holger Rapp, Daniel Andor // Robotics and Automation (ICRA), 2016 IEEE International Conference on / IEEE. — 2016. — P. 1271–1278.

[23] Rosten Edward, Drummond Tom. Machine learning for high-speed corner detection // European conference on computer vision / Springer. — 2006. — P. 430–443.

[24] Trajectory-based comparison of slam algorithms / Wolfram Burgard, Cyrill Stachniss, Giorgio Grisetti et al. — 2009.

[25] Tum Mono dataset.

[26] Vogiatzis George, Hernández Carlos. Video-based, real-time multi-view stereo // Image and Vision Computing. — 2011. — Vol. 29, no. 7. — P. 434–441.

[27] Xu Qiang, Ma Dengwu. Applications of Lie groups and Lie algebra to computer vision: A brief survey // Systems and Informatics (ICSAI), 2012 International Conference on / IEEE. — 2012. — P. 2024–2029.

[28] g 2 o: A general framework for graph optimization / Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat et al. // Robotics and Automation (ICRA), 2011 IEEE International Conference on / IEEE. — 2011. — P. 3607–3613.

[29] Боровков Александр Алексеевич. Математическая статистика. Оценка параметров, проверка гипотез. — 1984.