Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 19.Б11-мм

Иванов Кирилл Андреевич

Оптимизация алгоритма определения относительной позы многокамерной системы

Отчёт по производственной практике

Научный руководитель:
проф. каф. СП, д.ф.-м.н., проф. А.Н. Терехов

Консультант:
М.А. Терехов

Санкт-Петербург
2022

Оглавление

1 Введение

Вычисление относительной позы двух точек обзора многокамерной системы – одна из фундаментальных проблем геометрического компьютерного зрения, алгоритмы решения которой применяются в таких задачах, как определение структуры по движению (англ. *Structure from Motion*, SfM) [12, 33], одновременная локализация и построение карты (англ. *Simultaneous Localization And Mapping*, SLAM) [8] и визуальная локализация [43]. Данная проблема может быть распространена на произвольные устройства захвата изображений путем абстрагирования от деталей реализации и введения в рассмотрение *обобщенной модели камеры* [29]. В этом случае проблема может быть сформулирована как определение позиции (вектора смещения $\mathbf{t}$) и ориентации (матрицы поворота $\mathbf{R}$) одной точки обзора обобщенной камеры относительно другой. На рисунке 1 изображено схематическое представление задачи определения относительной позы для двух точек обзора камер, закрепленных на крыше автомобиля.

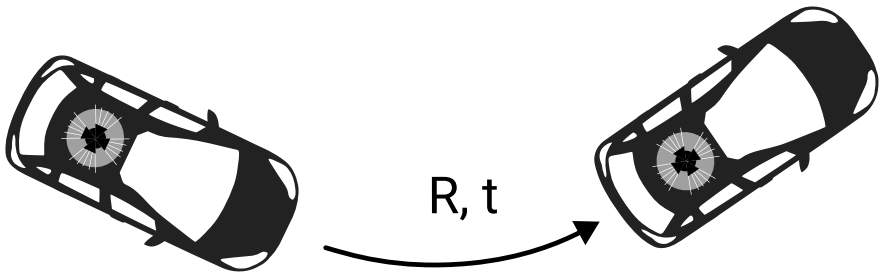


Рис. 1: Пример задачи оценки относительной позы: $\mathbf{R} \in \mathbb{R}^{3\times3}$ – искомая матрица вращение, $\mathbf{t} \in \mathbb{R}^3$ – искомый вектор смещения

Для определения обобщенной относительной позы используются специальные алгоритмы – *минимальные решатели* (англ. *minimal solvers*), которые оценивают её по минимальному набору соответствий с двух точек обзора [24, 34, 21, 17]. Поскольку вычисление относительной позы применяется в системах реального времени, например, в автономной навигации роботов, в виртуальной и дополненной реальности, важно быстро и точно получать нужные оценки.

Необходимо отметить, что зачастую набор соответствий содержит выбросы, которые не удовлетворяют действительной относительной позе. В этом случае для надежности оценки используются итеративные алгоритмы семейства RANSAC (англ. *Random Sample Consensus*) [5, 10, 31], требующие вычисления позы на каждой итерации. Однако существует проблема: оценка относительной позы решателем зачастую

занимает бóльшую часть времени итерации RANSAC, что затрудняет его применение в системах реального времени, поскольку данной схеме требуется выполнить большое количество шагов.

Таким образом, актуальной является проблема эффективного использования алгоритмов семейства RANSAC в системах реального времени. В данной работе рассматривается вопрос ускорения схемы RANSAC в задаче определения относительной позы многокамерной системы при помощи оптимизации минимального решателя по времени исполнения, что также должно сократить время каждой итерации и, как следствие, общее время, требующееся на определение позы.

2 Постановка цели и задач

Целью работы является ускорение вычисления обобщенной относительной позы многокамерной системы в схеме RANSAC путем оптимизации минимального решателя по времени исполнения. Для достижения цели были поставлены следующие задачи.

1. Отобрать реализации минимальных решателей и произвести их сравнительный анализ по времени исполнения и точности определения позы с целью выявления алгоритма для дальнейшей оптимизации.

2. Произвести профилирование и определение узких мест выбранной реализации.

3. Реализовать оптимизированную версию минимального решателя, интегрировать её в схему RANSAC и достигнуть ускорения в вычислении относительной позы.

4. Провести апробацию и анализ полученных результатов.

3 Терминология

В данной работе используются следующие термины и определения.

- **Камера** – устройство захвата изображений.
- **Мировая система координат** (англ. *world reference frame*) – система координат, относительно которой рассматривается движение в пространстве.
- **Система координат камеры** – система координат, в которой выражены измерения, полученные камерой.
- **Ориентир** (англ. *landmark*) – точка в трехмерном пространстве, выраженная в мировой системе координат.
- **Несущий вектор** (англ. *bearing vector*) – единичный вектор, представляющий полученное камерой измерение; выражен в системе координат камеры и направлен в сторону некоторого ориентира.
- **Относительная поза** – позиция и ориентация одной системы координат камеры относительно другой.
- **Соответствие** – пара несущих векторов, выраженных в разных системах координат камер и указывающих в направлении одного и того же ориентира.
- **Внутренние параметры камеры** – параметры, определяющие отображение 3D точки на плоскость изображения камеры.

4 Обзор предметной области

В настоящей секции производится обзор необходимой теории для реализации алгоритма вычисления обобщенной относительной позы, а также рассматриваются существующие подходы к ее оценке.

4.1 Соответствия точек

Установление соответствий между измерениями с разных точек обзора – важная часть задачи оценки относительной позы. Для выявления соответствий между двумя камерами необходимо определить *ключевые точки* (например, углы или грани) на изображениях и представить их в виде *дескрипторов*, схожесть которых позволяет сопоставить точки на двух изображениях. Классическими алгоритмами детекции являются, например, SIFT (Scale-Invariant Feature Transform) [25] или ORB [28]. Зачастую набор выявленных соответствий содержит выбросы: эта ситуация проиллюстрирована на рисунке 2. Стоит отметить, что в настоящее время активно разрабатываются алгоритмы детекции и сопоставления точек, основанные на нейронных сетях [7, 6, 36], которые позволяют уменьшить долю выбросов.

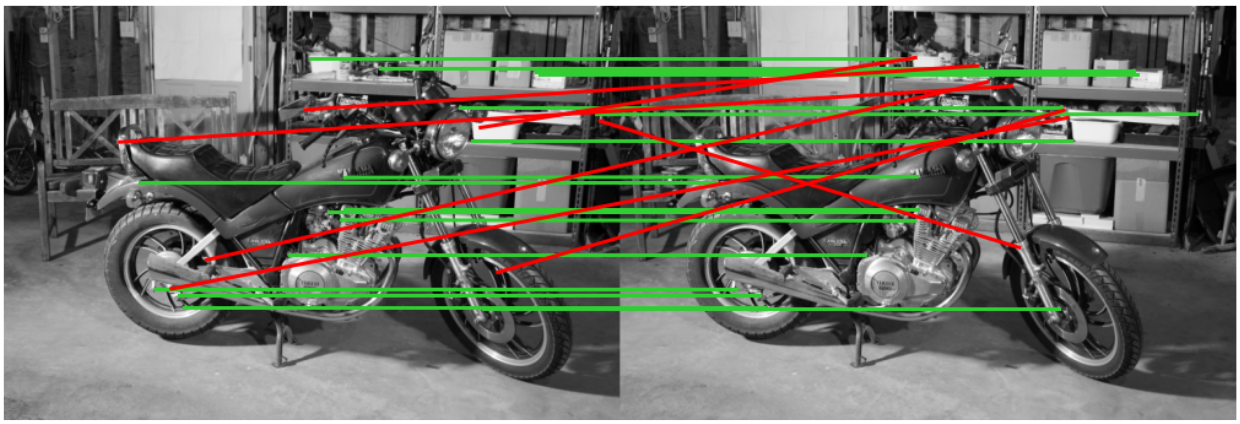


Рис. 2: Набор соответствий точек на двух изображениях: красным цветом выделены выбросы, зеленым – действительные соответствия

4.2 Обобщенная камера

В случае, если камера внутренне откалибрована, то есть внутренние параметры камеры известны, ее измерения можно выражать в виде *несущих векторов*: это представление показано на рисунке 3.

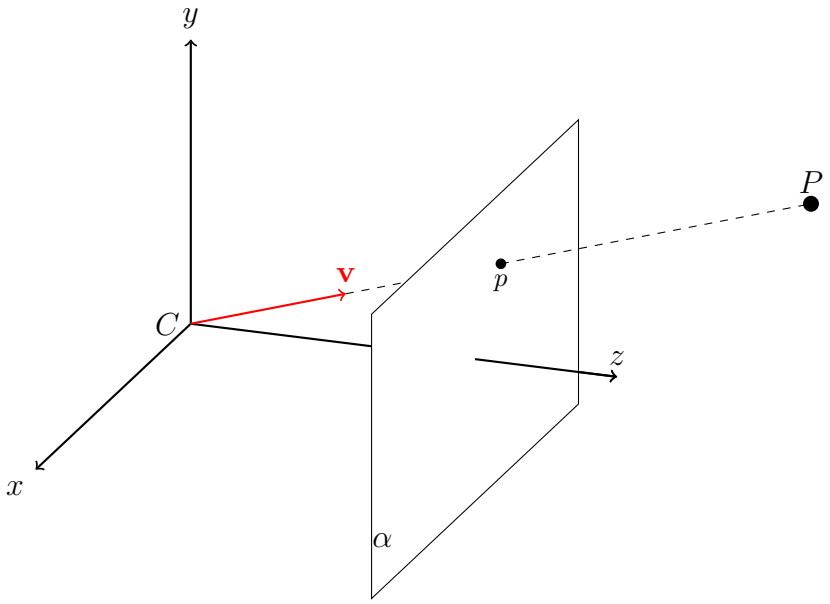


Рис. 3: Представление измерения внутренне откалиброванной камеры. C – центр проекции камеры, α – плоскость изображения, P – ориентир, p – проекция P на плоскость изображения (измерение камеры), $\mathbf{v}$ – несущий вектор, соответствующий P

Под *обобщенной камерой* понимается устройство захвата изображений, имеющее произвольную форму [29]. Такое абстрагирование от деталей реализации устройства позволяет разрабатывать алгоритмы определения относительной позы унифицированно для большинства моделей камер. Так, измерения, захваченные многокамерной системой или панорамной камерой, могут быть выражены в виде единой обобщенной камеры.

Измерения обобщенной камеры представляются при помощи векторов Плюккера (*Plücker vectors*) [29, 34]. Пусть $v \in \mathbb{R}^3$ – несущий вектор, выраженный в системе координат камеры, тогда соответствующее измерение l, выраженное в системе координат обобщенной камеры, определяется следующим соотношением:

$$l = \begin{pmatrix} v \\ t_c \times v \end{pmatrix} \in \mathbb{P}^5(\mathbb{R}), \tag{1}$$

где t_c – смещение системы координат камеры относительно центра обобщенной камеры, $\mathbb{P}^5(\mathbb{R})$ – пятимерное проективное пространство над $\mathbb{R}$. На рисунке 4 проиллюстрировано соотношение 1.

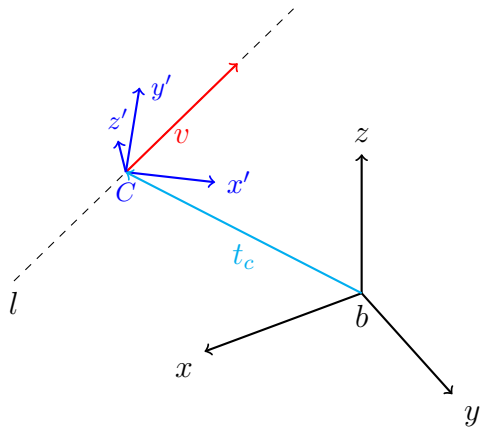


Рис. 4: Представление измерения обобщенной камеры: b – центр обобщенной камеры, C – центр камеры, которая захватила измерение, t_c – смещение системы координат камеры относительно системы координат обобщенной камеры, v – несущий вектор, выраженный в системе координат камеры, l – измерение камеры, выраженное в системе координат обобщенной камеры при помощи вектора Плюккера

4.3 Обобщенная относительная поза

Относительная поза между двумя камерами определяется как позиция и ориентация одной камеры относительно другой. Пусть P – ориентир, тогда задача определения относительной позы может быть сформулирована как поиск матрицы вращения $\mathbf{R} \in \mathbb{R}^{3\times3}$ и вектора смещения $\mathbf{t} \in \mathbb{R}^3$ таких, что выполняется соотношение 2:

$$[P]_1 = \mathbf{R}[P]_2 + \mathbf{t}, \tag{2}$$

здесь $[P]_1, [P]_2$ – выражения ориентира P в системах координат 1 и 2 соответственно. Пусть теперь $\mathbf{v}, \mathbf{v}'$ – несущие векторы, соответствующие одному ориентиру P и выраженные в системах координат первой и второй камеры соответственно, тогда они должны удовлетворять *эпиполярному ограничению* [21]:

$$\langle \mathbf{v}, (\mathbf{t} \times \mathbf{R}\mathbf{v}') \rangle = \mathbf{v}^T [\mathbf{t}]_\times \mathbf{R}\mathbf{v}' = \mathbf{v}^T \mathbf{E}\mathbf{v}' = 0, \tag{3}$$

где $\mathbf{t} \in \mathbb{R}^3$ – смещение второй камеры относительно первой, $\mathbf{R} \in \mathbb{R}^{3\times3}$ – вращение из ориентации второй камеры обратно в ориентацию первой, $[\mathbf{t}]_\times \in \mathbb{R}^{3\times3}$ – кососимметричная матрица, представляющая векторное произведение. Матрица $\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$ называется *существенной*.

Под *обобщенной относительной позой* понимается позиция и ориентация системы координат одной обобщенной камеры относительно другой: эта конфигурация проиллюстрирована на рисунке 5. Если l, l' – измерения, соответствующие ориентиру P, полученные первой и второй камерой соответственно, то они связаны следующим со-

отношением, называемым *обобщенным эпиполярным ограничением* [29, 21]:

$$l^T \begin{pmatrix} \mathbf{E} & \mathbf{R} \\ \mathbf{R} & \mathbf{0} \end{pmatrix} l' = 0, \tag{4}$$

где $\mathbf{E}$ – существенная матрица, $\mathbf{R}$ – вращение из точки обзора 2 обратно в точку обзора 1, $\mathbf{0} \in \mathbb{R}^{3\times3}$.

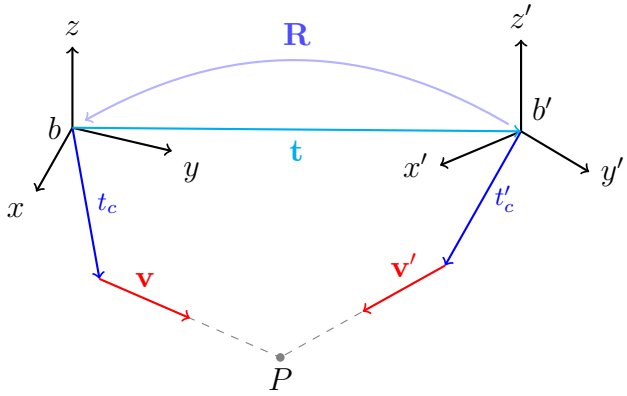


Рис. 5: Обобщенная относительная поза: неизвестными являются $\mathbf{t}$ и $\mathbf{R}$, $\mathbf{t}$ – смещение второй обобщенной камеры относительно первой, $\mathbf{R}$ – поворот, переводящий ориентацию второй системы координат обобщенной камеры обратно в первую, t_c, t'_c – смещения камеры относительно обобщенной камеры, P – ориентир, $\mathbf{v}, \mathbf{v}'$ – несущие векторы

4.4 Подходы к вычислению относительной позы

В наши дни наиболее распространены подходы оценки относительной позы на основе итеративных схем семейства RANSAC, а также нейросетевые подходы. В данной подсекции рассматриваются особенности обоих подходов, их достоинства и ограничения.

4.4.1 Нейросетевые подходы

В последнее десятилетие начал активно исследоваться вопрос применения нейронных сетей в задаче оценки относительной позы. Начиная с модели PoseNet, предложенной в 2015 году [19], стали появляться различные нейросетевые архитектуры [18, 15, 9, 2, 4], в которых оценка относительной позы рассматривалась как задача регрессии. Опираясь на результаты работы [15] можно заключить, что подход на основе RANSAC значительно превосходит нейронные сети в точности вычислений относительной позы, важно отметить, однако, что последние показывают превосходящие результаты в условиях однотипности окружения, например при однотонности текстур или отсутствии большого количества ключевых точек.

На основе недавнего исследования [1] можно выделить следующие ограничения нейросетевого подхода:

1. переобучение под конкретные точки обзора;
2. низкая точность в условиях нового или изменяющегося окружения.

4.4.2 Подходы на основе схемы RANSAC

Классическим является подход оценки относительной позы с использованием схемы RANSAC [10, 5]. Идея алгоритма заключается в итеративном извлечении m элементов из набора набора соответствий X (m – количество соответствий, необходимых для минимального решателя), вычислении обобщенной относительной позы при помощи минимального решателя и определении *множества консенсуса*, то есть набора соответствий $C \subset X$, согласованных с вычисленной позой. Алгоритм заканчивает свою работу, когда достигнут лимит итераций или получен набор консенсуса, размер которого больше некоторого порога, и возвращает оценку для лучшего набора консенсуса.

Максимальное количество итераций схемы N выбирается так, чтобы с вероятностью p хотя бы одна выборка из m соответствий не содержала выбросов:

$$N = \frac{\log(1-p)}{\log(1-\omega^m)}, \tag{5}$$

здесь ω – вероятность того, что выбранное соответствие не является выбросом. Обычно берут $p = 0.99$.

Основным достоинством схемы RANSAC является надежность оценки даже в случае большой доли выбросов в наборе соответствий. На основании работ [31, 38] можно выделить следующие ключевые ограничения алгоритмов семейства RANSAC.

1. Рост количества итераций при росте количества соответствий, необходимых минимальному решателю, поэтому в таких схемах важно, чтобы набор необходимых соответствий был действительно минимальным.
2. Увеличение количества требуемых итераций при наличии выбросов в наборе данных.

4.4.3 Минимальные решатели

Для определения относительной позы по минимальному набору соответствий используются специальные алгоритмы – *минимальные решатели*. Поскольку такие алгоритмы встраиваются в схему RANSAC, они должны обеспечивать быструю и точную оценку. Для вычисления относительной позы минимальные решатели опираются на геометрические ограничения, накладываемые на соответствия точек, например,

эпиполярное или аффинное. Известно, что для оценки позы в обобщенном случае необходимы шесть соответствий, которые позволяют определить неизвестные параметры позы. Наиболее часто используются следующие алгоритмы оценки обобщенной позы по минимальному набору соответствий.

- Шеститочечный решатель, предложенный Х. Стюениусом и др. в 2005 году [34]. Основная идея алгоритма – использование базисов Грёбнера для получения системы полиномиальных уравнений относительно неизвестных параметров позы. Основной недостаток – полиномиальная система имеет 64 решения, что требует устранение неоднозначности и, как следствие, дополнительных временных затрат.

- Семнадцатиточечный линейный решатель, предложенный Х. Ли и др. в 2008 году [24]. Основная идея – представление неизвестных в виде системы линейных уравнений. Основной недостаток – большое количество требуемых соответствий.

- Обобщенный восьмиточечный решатель на основе собственных значений, предложенный Л. Кнайпом и Х. Ли в 2014 году [21]. Основная идея – определение переменных вращения независимо от переменных смещения в закрытой форме при помощи итеративной минимизации функции стоимости описывающей вращение. Основной недостаток – оптимальное количество требуемых соответствий больше, чем шесть, а также риск сходимости к локальному минимуму.

- Линейный шеститочечный решатель, разработанный Д. Вентурой и др. в 2015 году [41]. Основная идея – использование аппроксимации первого порядка для оценки параметров позы. Основной недостаток – предположение об использовании решателя для оценки небольших движений.

- Шеститочечный решатель, предложенный В. Ларссоном и др. в 2017 году [23]. Основная идея – решение системы полиномиальных уравнений относительно параметров позы и использование автоматического создания шаблона исключения (elimination template) и базисов Грёбнера, позволяющих её решить. Основной недостаток – полиномиальная система имеет 64 решения, как и в алгоритме Х. Стюениуса.

- 3+3-точечный решатель, предложенный Д. Чжао и Б. Гуань в 2021 году [42]. Основная идея – решение системы полиномиальных уравнений при помощи метода скрытой переменной для исключения смещения. Основной недостаток – требуется, чтобы по три соответствия были получены двумя камерами с разных точек обзора.

Также заслуживает внимания исследование Ю. Кастена и др. [17], в котором предложен алгоритм определения относительной позы в полуобобщенном случаем: здесь

оценка позы новой камеры производится относительно семейства камер, для которых позы известны. Данный алгоритм потенциально имеет неплохие характеристики и может быть обобщен.

Важно отметить, что количество необходимых соответствий может быть уменьшено путем введения дополнительных ограничений (например, известный угол поворота или планарность движения) [13, 14, 37, 26] или рассмотрения задачи центральной относительной позы [27, 35]. Поскольку в данной работе рассматривается обобщенная постановка задачи, детальный обзор задач с ограничениями и центральных задач не проводится.

4.4.4 Вывод

Исходя из необходимости точной оценки позы и критичности ограничений нейросетевого подхода можно сделать вывод о целесообразности дальнейшего рассмотрения схемы RANSAC и минимальных решателей.

5 Сравнительный анализ минимальных решателей

В данной секции проводится экспериментальное сравнение существующих реализаций минимальных решателей по времени исполнения и точности определения позы.

5.1 Условия эксперимента

Для проведения эксперимента отбирались реализации на языке C++, что обусловлено его широким применением в задачах компьютерного зрения, как следствие, наличием большого количества библиотек, таких как OpenCV, Eigen, Ceres Solver, а также эффективностью во встраиваемых программах.

5.1.1 Исследуемые решения

Реализации минимальных решателей для определения обобщенной относительной позы были найдены в библиотеках OpenGV [20], COLMAP [32], PoseLib [22] и multi-camera-motion [40]. Для сравнения были отобраны решатели, представленные в таблице 1.

Таблица 1: Исследуемые минимальные решатели

Название	Кол-во соответствий	Исследуемые реализации	Замечание
6 точ.	6	PoseLib	Larsson et al. (CVPR 2017)
5+1 точ.	6	PoseLib	Центральная поза (5 соотв.) и масштаб (1 соотв.)
17 точ.	17	OpenGV	Li et al. (IEEE 2008)
8 точ.	8	OpenGV, COLMAP	Kneip et al. (CVPR 2014)
6 точ. (лин.)	6	multi-camera-motion	Ventura et al. (ICCJ 2015)
3+3 точ.	6	Предоставлена авторами	Zhao et al. (2021)

5.1.2 Характеристики системы

Эксперименты проводились на ЭВМ со следующими характеристиками.

- Операционная система: Ubuntu 21.10.
- Процессор: Intel(R) Core(TM) i5-11300H @ 3.10GHz (4 ядра, отключенный Turbo Boost).
- Оперативная память: 16 Гб DDR4 2400 МГц.
- Язык программирования: C++17.
- Компилятор: g++ 11.2.0.
- Флаги компилятора: `-O3 -march=native -ffast-math -fno-unsafe-math-optimizations -funroll-loops`

5.2 Вопросы и цель эксперимента

Целью настоящего эксперимента является выявление реализации минимального решателя, демонстрирующей сравнительно превосходящие результаты как по точности определения позы, так и по времени исполнения. Для достижения цели были поставлены следующие исследовательские вопросы.

1. Какова зависимость точности оценки позы от уровня зашумленности данных? Какие решатели менее остальных восприимчивы к уровню зашумленности?
2. Какие решатели лидируют в скорости вычисления относительной позы?
3. Какова производительность интегрированных в схему RANSAC решателей?

5.2.1 Набор данных

Для проведения экспериментов использовался синтетической набор данных [39], представляющий измерения, полученные многокамерной системой, закрепленной на крыше автомобиля. Модель системы, включает в себя четыре камеры, захватывающие изображения слева, справа, спереди и сзади автомобиля. Вдобавок для исследования производительности алгоритмов, интегрированных в схему RANSAC, использовались наборы данных AutoVision [30], HoloLens 2 [11] и MultiFoV [3]. Так, набор AutoVision содержит последовательности изображений, полученных при помощи закрепленной на крыше автомобиля системы из пяти камер. Набор данных HoloLens состоит из изображений, захваченных устройством HoloLens 2, а Multi-FoV предоставляет созданную в Blender последовательность кадров, захваченных четырьмя камерами с широким углом обзора. Примеры изображений, а также конфигурация синтетического набора представлены на рисунке 6.

(a) Синтетический (b) AutoVision (c) HoloLens 2 (d) MultiFoV

Рис. 6: Наборы данных, используемые для проведения экспериментов

Для исследование устойчивости решателей в соответствия вносился гауссовский шум $\epsilon \sim \mathcal{N}(0, \sigma^2)$ в направления несущих векторов, выраженный в градусах. Проводилось по 100 замеров для всех возможных комбинаций следующих величин:

- длина движения: от 3 м до 20 м с шагом 2.55 м;
- ширина сцены: от 15 м до 80 м с шагом 7.2 м;

- угол поворота: от 0 рад до $\frac{\pi}{6}$ рад с шагом 0.06 рад;
- уровни шума, соответствующие стандартному отклонению σ: от 0° до 0.24° с шагом 0.012°.

Стоит отметить, что ширина сцены играет важную роль в эксперименте, поскольку чем шире сцена, тем дальше точки, и тем ближе ситуация к центральной обобщённой позе, в которой такие алгоритмы могут выдавать нестабильный результат.

5.3 Метрики

Для сравнения алгоритмов в точности вычисления обобщенной относительной позы были использованы следующие геометрические метрики.

Абсолютная ошибка смещения (*absolute translation error*) – это $l2$ норма вектора ошибки $\epsilon = \hat{\mathbf{t}} - \mathbf{t}$, где $\hat{\mathbf{t}}, \mathbf{t}$ – оценка вектора смещения и его действительное значение соответственно, выраженные в мировой системе координат:

$$AbsoluteTranslationError(\mathbf{t}, \hat{\mathbf{t}}) = \|\epsilon\|_2 = \|\hat{\mathbf{t}} - \mathbf{t}\|_2.$$

Абсолютная ошибка вращения. Пусть $\mathbf{R}, \hat{\mathbf{R}} \in SO(3)$ – действительное вращение и его оценка. Тогда абсолютная ошибка вращения определяется следующим образом:

$$AbsoluteRotationError(\mathbf{R}, \hat{\mathbf{R}}) = \|\log(\mathbf{R}\ \hat{\mathbf{R}}^{-1})\|.$$

Угловая ошибка смещения (*angular translation error*) – угол между $\hat{\mathbf{t}}$ и $\mathbf{t}$, показывающий отклонение оценки вектора смещения от его действительного значения:

$$AngularTranslationError(\mathbf{t}, \hat{\mathbf{t}}) = \arccos \frac{\langle \hat{\mathbf{t}}, \mathbf{t} \rangle}{\|\hat{\mathbf{t}}\|_2 \|\mathbf{t}\|_2}.$$

Измерение и угловой, и абсолютной ошибки смещения вычисленной позы обусловлено существованием конфигураций, в которых определить масштаб вектора движения затруднительно (например, случай чистого движения без поворота), тем не менее направление движения должно оставаться близким к истинному.

5.4 Результаты

В результате экспериментального исследования были выявлены три реализации, превосходящие остальные по времени исполнения, – 17-точечный, линейный шеститочечный и 5+1-точечный решатели. В то же время восьмиточечные решатели из COLMAP и OpenGV демонстрируют удовлетворительное время работы, при этом реализация из второй библиотеки работает несколько быстрее. Остальные же решатели,

в свою очередь, демонстрируют существенно более высокие показатели времени. Полученные оценки представлены на рисунке 7 и в таблице 2.

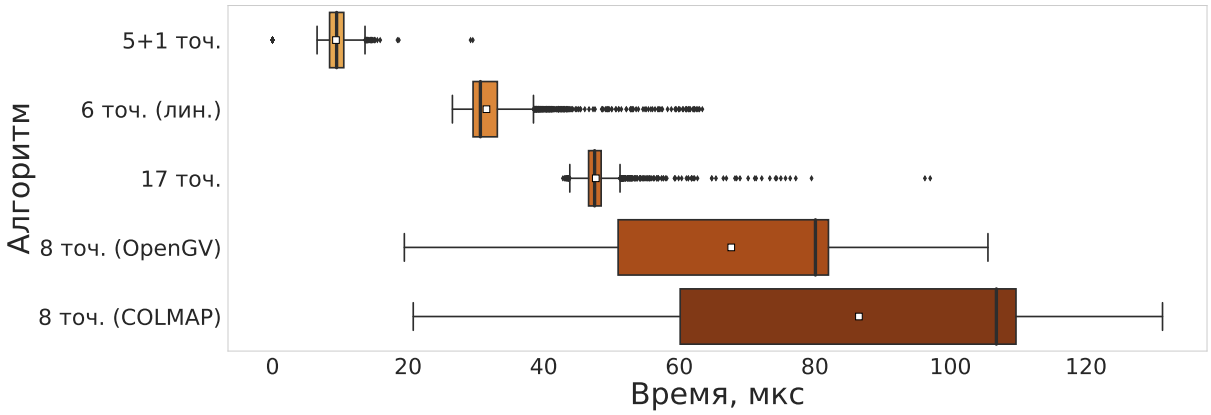
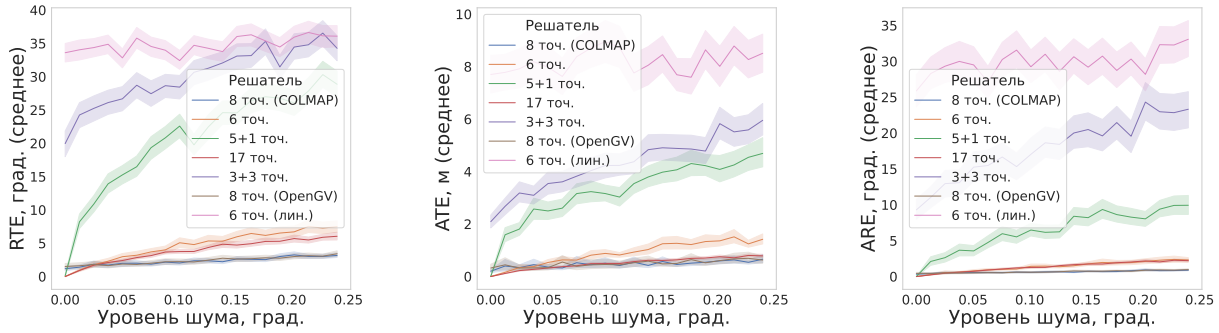


Рис. 7: Время исполнения решателей: белому квадрату соответствует среднее время; 6 точ. и 3+3 точ. не отображены

Таблица 2: Оценки времени работы минимальных решателей

Решатель	17 точ.	3+3 точ.	5+1 точ.	6 точ.	6 точ. (лин.)	8 точ. (COLMAP)	8 точ. (OpenGV)
Среднее, мкс	49	988	9	860	32	86	69
Медиана, мкс	48	989	9	859	31	106	81
Ст. отклонение, мкс	5	51	2	47	4	29	19

На рисунке 8 приведена зависимость ошибок решателей от уровня зашумленности соответствий. На основании полученных результатов, можно выделить алгоритмы 6 точ., 17 точ., 8 точ. (OpenGV), 8 точ. (OpenGV) как наиболее устойчивые в представленном наборе, при этом восьмиточечные решатели, хотя и несколько уступают первым двум алгоритмам при нулевой зашумленности, с её увеличением отклоняются от истинных значений позы не так сильно. Усредненные по всем запускам значения ошибок, представленные в таблице 3, позволяют установить, что восьмиточечные решатели являются наиболее стабильными среди представленных.



(a) Угл. ошибка смещения (b) Абс. ошибка смещения (c) Абс. ошибка вращения

Рис. 8: Зависимость ошибок решателей от уровня зашумленности данных

Таблица 3: Оценки ошибок минимальных решателей

	method_name	17 точ.	3+3 точ.	6 точ.	8 точ. (OpenGV)	8 точ. (COLMAP)	5+1 точ.	6 точ. (лин.)
ARE,°	Медиана	0.8	4.3	0.5	0.3	0.3	1.8	11.3
	Среднее	1.2	12.2	1.3	0.6	0.6	5.3	22.4
	Ст. отклонение	1.5	22.6	3.4	1.4	1.3	12.8	29.1
ATE, м	Медиана	0.3	2.2	0.3	0.1	0.1	1.3	3.6
	Среднее	0.5	2.7	0.7	0.4	0.4	2.1	4.1
	Ст. отклонение	0.5	2.3	1.2	0.8	0.8	2.2	2.2
RTE,°	Медиана	1.9	17.8	1.7	0.9	0.9	8.1	29.8
	Среднее	3.5	28.9	4.4	2.3	2.2	18.1	33.0
	Ст. отклонение	4.7	28.1	9.6	5.0	4.9	23.6	19.7

Эксперименты со схемой RANSAC на наборах данных выявили три решателя, которые превосходят остальные в соотношении точности оценки и времени исполнения, а именно шеститочечный, 5+1-точечный и реализации восьмиточечного решателя. Так, шеститочечный решатель демонстрирует относительное превосходство в точности определения позы, тем не менее работает значительно медленее, чем 5+1 и 8-точечные. В то же время 5+1-точечный работает быстро и точно на всех трех датасетах, превосходит реализации 8-точечного на датасетах HoloLens и Multi-FoV, однако уступает им на датасете AutoVision. На рисунках 9 и 10 представлены указанные соотношении, при этом отсутствие отметки решателя означает, что он показал неудовлетворительные результаты либо по времени, либо по точности.

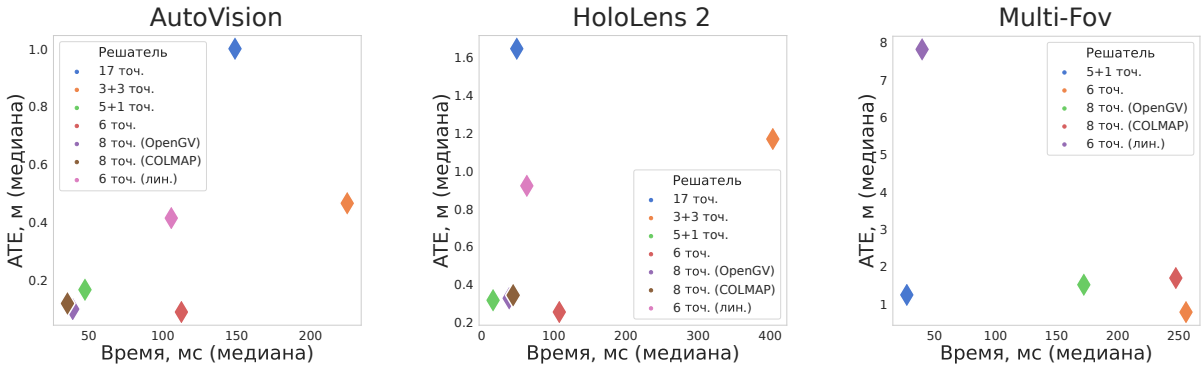


Рис. 9: Сравнение минимальных решетелей в схеме RANSAC: соотношения времени и абсолютной ошибки смещения

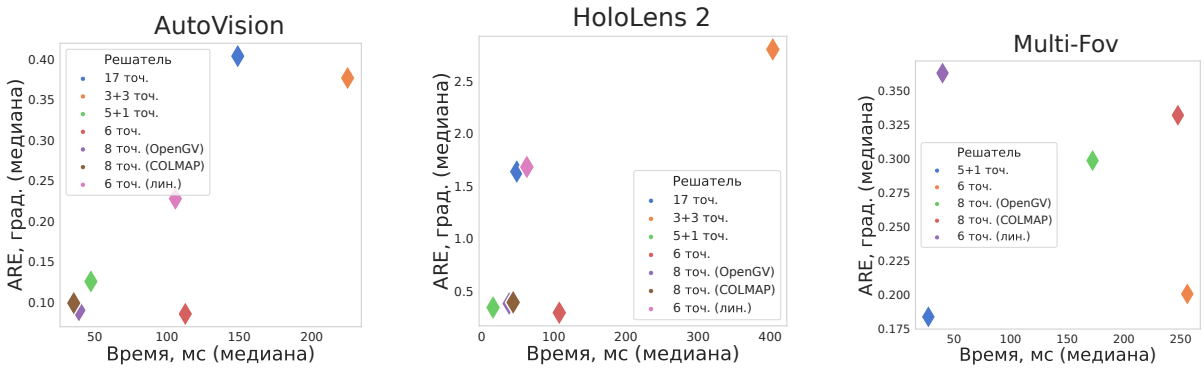


Рис. 10: Сравнение минимальных решетелей в схеме RANSAC: соотношения времени и абсолютной ошибки вращения

Таким образом, на основании результатов исследования устойчивости и времени исполнения можно сделать вывод об оптимальности использования восьмитотчечного решателя из библиотеки OpenGV для дальнейшей оптимизации, поскольку он демонстрирует устойчивость к зашумленности данных, а также имеет относительно высокую скорость работы. Решатель 6 точ. (PoseLib), показавший хорошую устойчивость к шуму, требует в среднем примерно в десять раз больше времени для оценки позы, чем 8 точ. (OpenGV). Важно также отметить, что на момент начала работы ещё не существовало эффективной интеграции 5+1-точечного решателя в схему RANSAC.

6 Оптимизация алгоритма

В настоящей секции приводятся анализ восьмиточечного решателя из библиотеки OpenGV, его профилирования с целью выявления узких мест, а также описание проведенной оптимизации.

6.1 Анализ базовой версии алгоритма

Детальное описание работы 8-точечного решателя можно найти в работе Л. Кнайпа и др. [21], однако остановимся на ключевых для оптимизации моментах его работы.

Пусть есть набор соответствий $\{(v_i, v_i')\}_{i=1}^8$, где v_i, v_i' – несущие векторы, направленные на один и тот же ориентир с разных точек обзора многокамерной системы. Тогда, представив их в виде векторов Плюккера, получим набор $\{(l_i, l_i')\}_{i=1}^8$, где

$$l_i = \begin{pmatrix} v_i \\ t_{c,i} \times v_i \end{pmatrix},$$

$t_{c,i}$ – смещение системы координат i-й камеры относительно центра обобщенной камеры; l_i' выражается аналогичным образом.

В этом случае обобщенное эпиполярное ограничение для i-й пары соответствий может быть преобразовано к следующему виду:

$$g_i^T \tilde{t} = 0,$$

$$g_i = \begin{pmatrix} v_i \times \mathbf{R} v_i' \\ v_i^T([t_{c,i}]_\times \mathbf{R} - \mathbf{R}[t_{c,i}']_\times) v_i' \end{pmatrix} \text{ и } \tilde{t} = \begin{pmatrix} \omega \mathbf{t} \\ \omega \end{pmatrix}, \omega \in \mathbb{R} \setminus \{0\}.$$

Учитывая все соответствия, получаем следующее ограничение:

$$G^T t = \begin{pmatrix} g_1 \dots g_8 \end{pmatrix}^T \tilde{t} = 0. \tag{6}$$

Поиск вращение $\mathbf{R}$ может быть выражен как следующая задача минимизации:

$$\mathbf{R} = \arg\min_{\mathbf{R}} \lambda_{H,min},$$

здесь $H = GG^T = \Sigma_{i=1}^8 g_i g_i^t$, $\lambda_{H,min}$ – минимальное собственное значение матрицы H.

Таким образом, исходный алгоритм сначала вычисляет начальное приближение для вращения, затем с его помощью выполняет градиентный спуск в пространстве вращений и находит искомое вращение, после чего вычисляет смещение. Псевдокод алгоритма представлен в листинге 1 (детали реализации $CalculateInitialRotation$, $Minimize$ и $FindTranslation$ опущены).

Для дальнейшей оптимизации важно выделить следующие ключевые особенности работы алгоритма:

- матрица H зависит от параметров вращения и пересчитывается на каждом шаге градиентного спуска;
- из определения матрицы $G = \begin{pmatrix} g_1 \dots g_8 \end{pmatrix}$, где $g_i \in \mathbb{R}^4$, видно, что каждый её столбец не зависит по данным от остальных, что открывает возможности для применения параллелизма.

Listing 1 Обобщенный восьмиточечный решатель

```
1: function Solver8pt(\{(v_i, v_i')\}_{i=1}^8)
2:     R_init ← CalculateInitialRotation(\{(v_i, v_i')\}_{i=1}^8)
3:     (R, λ_{H,min}) = Minimize(R_init)                ▷ Градиентный спуск
4:     t = FindTranslation(λ_{H,min})
5:     return (R, t)
6: end function
```

6.2 Профилирование минимального решателя

Для реализации оптимизированной версии 8 точечного решателя необходимо произвести профилирование исходного алгоритма с целью определения узких мест и фрагментов кода, исполнение которых можно ускорить.

Для профилирования был использован Google Performance Tools (gperftools). Выбор данного инструмента обусловлен наличием наглядного представления результата исполнения программы в виде графа вызовов с указанием затраченного процессорного времени, а также небольшими накладными расходами, связанными с профилированием.

На основе анализа результатов профилирования, можно заключить, что узким горлышком 8 точ. (OpenGV) является функция `composeG`, исполнение которой занимает в среднем 55% времени вычисления относительной позы. Было установлено, что эта функция вызывается в среднем 188 ± 92 раз за один вызов минимального решателя, что объясняется использованием её результата на каждом шаге итеративной минимизации. На рисунке 11 приведен полученный в результате профилирования граф вызовов.

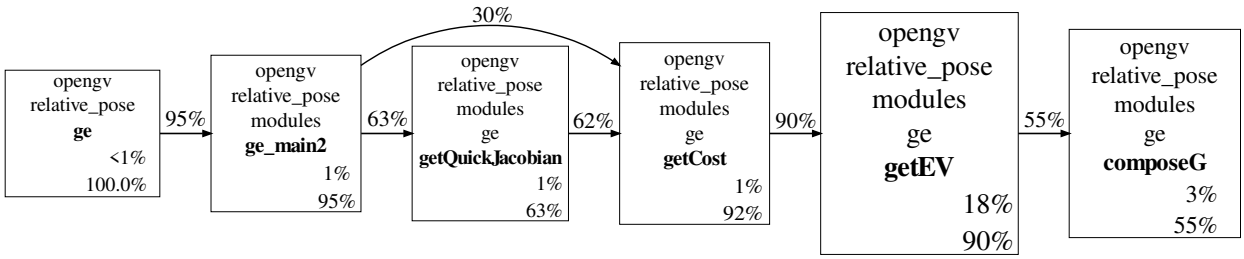


Рис. 11: Граф вызовов 8pt (OpenGV): отображаются узлы с долей семплов большей, чем 0.1

На основе полученных результатов и анализа кода решателя можно сделать следующие выводы.

- На данный момент распределение памяти в функции `composeG` неоптимально: трафик запрашиваемых данных на каждом вызове больше, чем действительно нужно в случае определения позы для восьми соответствий.
- Функция `composeG` содержит последовательные фрагменты кода, которые не были оптимизированы компилятором и которые могут быть ускорены при помощи векторизации и использования параллелизма по данным.
- В качестве схемы итеративной минимизации используется градиентный спуск, сходимость которого может быть ускорена путем подбора оптимальных параметров, также можно рассмотреть вариант его замены на другой алгоритм минимизации.

6.3 Реализация

Описанные ранее способы оптимизации нашли свое применение в предлагаемой реализации восьмиточечного решателя. Новая версия была написана на языке C++ и опубликована в качестве библиотеки на GitHub [16]. Диаграмма компонентов представлена на рисунке 12. Данная функциональность имеет в качестве внешней зависимости лишь библиотеку линейной алгебры Eigen. Для сборки библиотеки используются следующие технологии: компилятор g++ или clang, система автоматизации сборки CMake. В настоящее время библиотека распространяется только для операционных систем на базе GNU/Linux.

Рассмотрим подробнее ответственности каждого компонента:

- **Solver** отвечает за вычисление обобщенной относительной позы, принимая в качестве входных данных восемь пар несущих векторов, а также соответствующие им смещения системы координат камеры относительно центра обобщенной камеры;

- **Math** отвечает за математические преобразования, необходимые для работы алгоритма;
- компонент **Util** содержит служебную функциональность;
- компонент **Types** включает описание типов, участвующих в работе алгоритма.

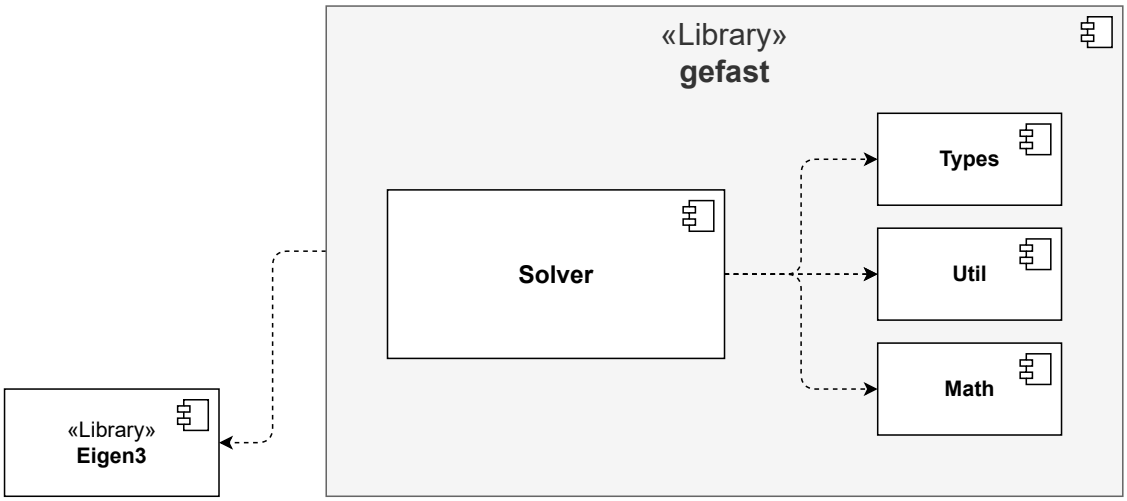


Рис. 12: Диаграмма компонентов библиотеки

6.3.1 Детали реализации

Рассмотрим ключевые оптимизации, примененные в предлагаемой реализации восьмиточечного решателя.

Функция `composeG` была переписаны с использованием принципа SIMD, для этого используемые данные были преобразованы к векторизованному виду. Фрагменты кода, использующие параллелизм, были написаны при помощи интринзиков компилятора. В настоящее время поддержка параллелизма по данным обеспечивается лишь для процессоров с векторным расширением AVX2, в отсутствии последнего соответствующие фрагменты кода возвращаются к последовательному исполнению.

Для ускорения сходимости схемы итеративной минимизации градиентный спуск был заменен на алгоритм Бройдена – Флетчера – Гольдфарба – Шанно (BFGS), который, несколько увеличив время одной итерации из-за вычисления аппроксимации гессиана, обеспечил существенное сокращение их количества. Как следствие, количество вызовов функции `composeG` было сокращено примерно в два раза (с 188 ± 92 до 83 ± 27).

7 Экспериментальное исследование полученной реализации

В данной секции проводится экспериментальное сравнение полученной реализации с базовой версией и с существующими минимальными решателями по времени исполнения и точности определения позы. Эксперименты проводились при тех же условиях, что и в секции 5.

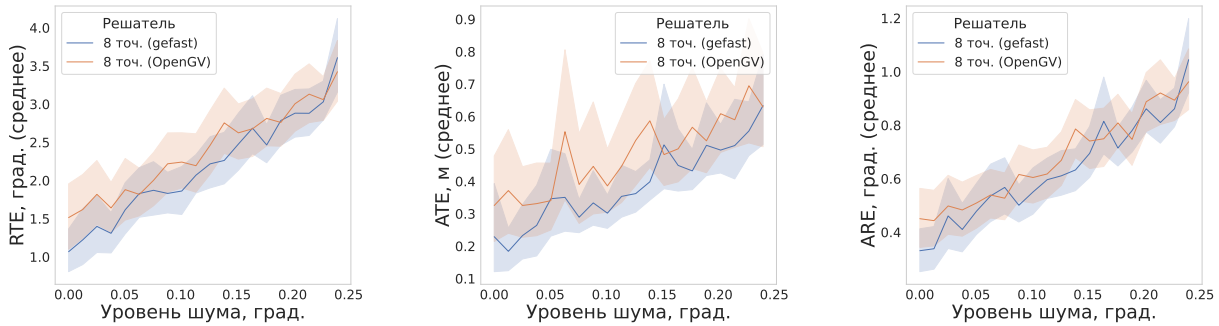
7.1 Вопросы и цель эксперимента

Целью настоящего эксперимента является установление качества полученной оптимизированной версии восьмиточечного минимального решателя с точки зрения стабильности и времени исполнения. Для достижения цели были поставлены следующие исследовательские вопросы.

1. Какова устойчивость оптимизированного алгоритма к зашумленности данных?
2. Каково ускорение предложенной реализации по отношению к базовой версии?
3. Какова производительность полученной реализации в схеме RANSAC?

7.2 Результаты

В ходе экспериментального исследования было установлено отсутствие существенных различий в стабильности между базовым и предложенным алгоритмами. Зависимость качества оценки поза от уровня шума в соответствиях обеих версий показана на рисунке 13, а в таблице 4 представлено сравнение ошибок полученной реализации и других алгоритмов.



(a) Угл. ошибка смещения (b) Абс. ошибка смещения (c) Абс. ошибка вращения

Рис. 13: Зависимости ошибок базовой и оптимизированной версий восьмиточечного решателя от уровня зашумленности данных

Таблица 4: Сравнение оценок ошибок полученной реализации с другими алгоритмами

	method_name	17 точ.	3+3 точ.	6 точ.	8 точ. (OpenGV)	8 точ. (COLMAP)	8 точ. (gefast)	5+1 точ.	6 точ. (лин.)
ARE,°	Медиана	0.8	4.3	0.5	0.3	0.3	0.3	1.8	11.3
	Среднее	1.2	12.2	1.3	0.6	0.6	0.6	5.3	22.4
	Ст. отклонение	1.5	22.6	3.4	1.4	1.3	1.3	12.8	29.1
ATE, м	Медиана	0.3	2.2	0.3	0.1	0.1	0.1	1.3	3.6
	Среднее	0.5	2.7	0.7	0.4	0.4	0.3	2.1	4.1
	Ст. отклонение	0.5	2.3	1.2	0.8	0.8	0.7	2.2	2.2
RTE,°	Медиана	1.9	17.8	1.7	0.9	0.9	0.9	8.1	29.8
	Среднее	3.5	28.9	4.4	2.3	2.2	2.1	18.1	33.0
	Ст. отклонение	4.7	28.1	9.6	5.0	4.9	4.5	23.6	19.7

Что касается времени исполнения, предложенная реализация превосходит все рассматриваемые алгоритмы, за исключением 5+1-решателя, на синтетических данных. Сравнение алгоритмов по времени исполнения представлено на рисунке 14. Важно также отметить, что предложенная реализация имеет существенно меньший разброс времени исполнения – об этом свидетельствует узкий межквартильный размах.

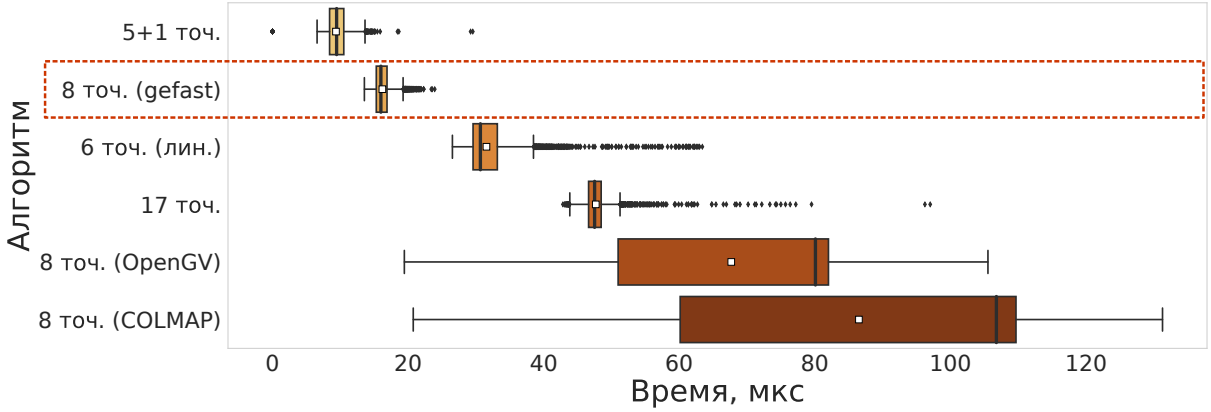


Рис. 14: Время исполнения решателей: белому квадрату соответствует среднее время; 6 точ. и 3+3 точ. не отображены

Исходя из оценок времени исполнения, представленных в таблице 5, можно установить, что оптимизированная версия работает в среднем примерно в четыре раза быстрее, чем базовая, на данных без выбросов.

Таблица 5: Оценки времени работы минимальных решателей

Решатель	17 точ.	3+3 точ.	5+1 точ.	6 точ.	6 точ. (лин.)	8 точ. (COLMAP)	8 точ. (OpenGV)	8 точ. (gefast)
Среднее	49	988	9	860	32	86	69	16
Медиана	48	989	9	859	31	106	81	16
Ст. отклонение	5	51	2	47	4	29	19	1

Оценки работы решателей в схеме RANSAC представленны в таблице 6. Здесь $\epsilon_R = AreaUnderCurve$ для вращения с порогом 1°, $\epsilon_t = AreaUnderCurve$ для абсолютного смещения с порогами 0.3 м (AutoVision и HoloLens 2) и 1 м (Multi-FoV),

t_{med}, t_{mean} – оценки медианного и среднего времени исполнения соответственно (мс), $\#inl$ – медианная доля найденных действительных соответствий, ограничение на количество итераций RANSAC – 3000, лучший результат выделен жирным, а следующий за ним подчеркнут. На основании полученных оценок можно заключить, что восьмиточечные решатели имеют точность, сравнимую с 6-точечным решетелем, на наборах данных с большим количеством действительных соответствий (AutoVision, HoloLens 2), работая при этом существенно быстрее. В то же время предложенная реализация имеет наименьшее время исполнения на вышеупомянутых наборах данных. Важно, однако, отметить, что все восьмиточесные решатели, включая оптимизированную версию, демонстрируют ухудшение производительности в случае, если набор соответствий содержит существенную долю выбросов (> 0.5) – это отражается на результатах, полученных на наборе данных Multi-FoV. Связано это с тем, что решателям, работающим с неминимальным количеством соответствий (> 6), требуется большее количество итераций в схеме RANSAC для удовлетворения критериям его остановки (см. выражение количества итераций 5).

Таблица 6: Оценки работы интегрированных в схему RANSAC минимальных решателей

Решатель	**AutoVision**					**HoloLens 2**					**Multi-FoV**				
	ϵ_R	ϵ_t	$\#inl$	t_{med}	t_{mean}	ϵ_R	ϵ_t	$\#inl$	t_{med}	t_{mean}	ϵ_R	ϵ_t	$\#inl$	t_{med}	t_{mean}
6 точ.	**0.90**	**0.54**	**0.80**	109.95	132.23	**0.57**	**0.31**	**0.63**	110.19	413.49	0.73	**0.97**	**0.49**	314.25	545.28
6 точ. (лин.)	0.53	0.26	0.67	103.67	96.50	0.20	0.14	0.44	64.22	63.67	0.45	0.92	0.43	42.06	52.20
8 точ. (gefast)	0.89	**0.54**	**0.80**	**28.98**	**47.17**	0.52	0.29	**0.63**	**14.75**	**23.39**	0.57	0.95	0.46	50.23	41.12
8 точ. (COLMAP)	0.89	0.52	**0.80**	35.46	50.02	0.49	0.28	0.61	43.55	103.98	0.58	0.95	0.46	247.50	203.75
8 точ. (OpenGV)	0.89	0.52	**0.80**	36.12	49.35	0.49	0.28	0.62	39.11	86.92	0.57	0.95	0.46	193.31	158.07
17 точ.	0.49	0.24	0.49	148.81	123.81	0.27	0.19	0.34	48.59	68.07	0.16	0.91	0.07	143.02	125.22
5+1 точ.	0.85	0.41	0.77	45.41	76.18	0.53	0.28	0.60	15.64	25.65	**0.76**	0.96	**0.49**	**26.89**	**35.06**
3+3 точ.	0.77	0.35	0.71	137.66	158.62	0.39	0.25	0.55	152.68	735.64	0.68	0.91	0.44	677.46	1096.85

Дополнительно был проведен анализ зависимости точности позы, выдаваемой восьмиточечным решателем, от ограничения на максимально количество итераций в схеме минимизации. Полученные оценки приведены в таблице 7. Так, можно заключить, что с увеличением количества итераций наблюдается незначительное улучшение качества оценки позы при отсутствии существенного изменения времени работы на наборах данных AutoVision и Hololens 2. На датасете Multi-FoV наблюдается обратная ситуация: здесь с увеличением количества итераций улучшение качества более заметно, тем время исполнения также возрастает.

Таблица 7: Анализ влияния ограничения количества итераций 8-точечного решателя на точность позы и время работы

Конфигурация	**AutoVision**					**HoloLens 2**					**Multi-FoV**				
	ϵ_R	ϵ_t	$\#inl$	t_{med}	t_{mean}	ϵ_R	ϵ_t	$\#inl$	t_{med}	t_{mean}	ϵ_R	ϵ_t	$\#inl$	t_{med}	t_{mean}
11 итераций	0.89	0.54	0.80	28.98	47.17	0.52	0.29	0.63	14.75	23.39	0.57	0.95	0.46	50.23	41.12
12 итераций	0.89	0.55	0.80	26.74	40.13	0.52	0.29	0.63	13.85	23.23	0.59	0.96	0.46	51.22	42.26
14 итераций	0.89	0.55	0.81	27.21	41.29	0.53	0.30	0.63	14.33	24.71	0.60	0.96	0.47	56.32	47.00
16 итераций	0.90	0.56	0.81	27.15	40.79	0.54	0.31	0.63	14.89	26.66	0.60	0.97	0.47	59.78	51.32
18 итераций	0.90	0.55	0.81	27.90	41.43	0.54	0.31	0.64	15.66	28.58	0.61	0.97	0.47	63.43	55.61
20 итераций	0.90	0.56	0.81	27.80	41.68	0.54	0.31	0.64	16.13	30.42	0.61	0.97	0.47	66.94	59.25

Вдобавок была рассмотрена возможность сокращения количества итераций в обобщенном восьмиточечном решателе до 6 и 7. Результаты работы алгоритма для шести, семи и восьми соответствий при оптимальном ограничении на количество итераций с точки зрения соотношения точности позы и времени работы представлены в таблице 8. Можно заметить отсутствие критичных различий между 8- и 7-точечными версиями решателя, в то же время шеститочечная версия заметно проседает по качеству.

Таблица 8: Анализ поведения обобщенного решателя на основе собственных чисел для 6, 7 и 8 требуемых соответствий

Конфигурация	**AutoVision**					**HoloLens 2**					**Multi-FoV**				
	ϵ_R	ϵ_t	$\#inl$	t_{med}	t_{mean}	ϵ_R	ϵ_t	$\#inl$	t_{med}	t_{mean}	ϵ_R	ϵ_t	$\#inl$	t_{med}	t_{mean}
8 точ., 16 ит.	**0.90**	**0.56**	**0.81**	27.15	40.79	**0.54**	**0.31**	**0.63**	14.89	**26.66**	0.60	**0.96**	**0.47**	59.78	51.32
7 точ., 20 ит.	0.89	0.54	0.80	27.48	40.95	0.50	0.28	0.61	15.13	29.33	**0.61**	0.95	0.46	46.05	51.20
6 точ., 20 ит.	0.87	0.47	0.79	**25.08**	**39.04**	0.42	0.25	0.58	**14.16**	28.10	0.55	0.94	0.43	**37.54**	**45.01**

7.3 Вывод

На основании полученных результатов можно заключить, что предложенная реализация превосходит остальные в отношении точности оценки к скорости работы на наборах данных с долей действительных соответстви большей, чем 0.5. В то же время уступая шеститоченым алгоритмам в противоположном случае.

Также интересным наблюдением является относительная стабильность 7-точечной версии, которая может быть применена в условиях большого количества выбросов в наборе, поскольку в этом случае для сходимости схемы RANSAC потребуется меньшее количество итераций. Однако оценки, выдаваемые шеститочечными решателями при таких условиях, всё еще лучше.

8 Заключение

В рамках данной работы были выполнены следующие задачи.

- Отобраны реализации минимальных решателей и произведен их сравнительный анализ по времени исполнения и точности определения позы, а также выявлен алгоритм для оптимизации.
- Произведено профилирование и определение узких мест выбранной реализации минимального решателя.
- Реализована оптимизированная версия восьмиточечного минимальный решателя и достигнуто ускорение в вычислении относительной позы.
- Проведена апробация и анализ полученных результатов.

Реализация оптимизированной версии может быть найдена по следующему адресу: https://github.com/kirill-ivanov-a/gefast.

Вдобавок были выполнены следующие задачи.

- Проанализирована зависимость скорости работы RANSAC от ограничения на количество итераций минимизации в восьмиточечном решателе.
- Проанализирована возможность сокращения количества требуемых соответствий для работы восьмиточечного решателя до шести и семи.

Список литературы

[1] Back to the Feature: Learning Robust Camera Localization from Pixels to Pose / Paul-Edouard Sarlin, Ajaykumar Unagar, Mans Larsson et al. // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. — 2021. — P. 3247–3257.

[2] Balntas Vassileios, Li Shuda, Prisacariu Victor. Relocnet: Continuous metric learning relocalisation using neural nets // Proceedings of the European Conference on Computer Vision (ECCV). — 2018. — P. 751–767.

[3] 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.

[4] CamNet: Coarse-to-fine retrieval for camera re-localization / Mingyu Ding, Zhe Wang, Jiankai Sun et al. // Proceedings of the IEEE/CVF International Conference on Computer Vision. — 2019. — P. 2871–2880.

[5] Chum Ondřej, Matas Jiří, Kittler Josef. Locally optimized RANSAC // Joint Pattern Recognition Symposium / Springer. — 2003. — P. 236–243.

[6] D2-net: A trainable cnn for joint detection and description of local features / Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla et al. // arXiv preprint arXiv:1905.03561. — 2019.

[7] DeTone Daniel, Malisiewicz Tomasz, Rabinovich Andrew. Superpoint: Self-supervised interest point detection and description // Proceedings of the IEEE conference on computer vision and pattern recognition workshops. — 2018. — P. 224–236.

[8] Durrant-Whyte Hugh, Bailey Tim. Simultaneous localization and mapping: part I // IEEE robotics & automation magazine. — 2006. — Vol. 13, no. 2. — P. 99–110.

[9] En Sovann, Lechervy Alexis, Jurie Frédéric. Rpnet: An end-to-end network for relative camera pose estimation // Proceedings of the European Conference on Computer Vision (ECCV) Workshops. — 2018. — P. 0–0.

[10] Fischler Martin A, Bolles Robert C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography // Communications of the ACM. — 1981. — Vol. 24, no. 6. — P. 381–395.

[11] Group ETH Zurich Computer Vision, Zurich Microsoft Mixed Reality & AI Lab. The ETH-Microsoft Localization Dataset. — URL: https://github.com/cvg/visloc-iccv2021/ (online; accessed: 2021-11-25).

[12] Hartley Richard, Zisserman Andrew. 3D Reconstruction of Cameras and Structure // Multiple View Geometry in Computer Vision. — 2 edition. — Cambridge University Press, 2004. — P. 262–278.

[13] Hee Lee Gim, Faundorfer Friedrich, Pollefeys Marc. Motion estimation for self-driving cars with a generalized camera // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. — 2013. — P. 2746–2753.

[14] Hee Lee Gim, Pollefeys Marc, Fraundorfer Friedrich. Relative pose estimation for a multi-camera system with known vertical direction // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. — 2014. — P. 540–547.

[15] Image-based localization using lstms for structured feature correlation / Florian Walch, Caner Hazirbas, Laura Leal-Taixe et al. // Proceedings of the IEEE International Conference on Computer Vision. — 2017. — P. 627–637.

[16] Ivanov Kirill. A fast implementation of generalized eigensolver for calibrated camera relative pose estimation. — URL: https://github.com/kirill-ivanov-a/gefast (online; accessed: 2022-05-11).

[17] Kasten Yoni, Galun Meirav, Basri Ronen. Resultant based incremental recovery of camera pose from pairwise matches // 2019 IEEE Winter Conference on Applications of Computer Vision (WACV) / IEEE. — 2019. — P. 1080–1088.

[18] Kendall Alex, Cipolla Roberto. Modelling uncertainty in deep learning for camera relocalization // 2016 IEEE international conference on Robotics and Automation (ICRA) / IEEE. — 2016. — P. 4762–4769.

[19] Kendall Alex, Grimes Matthew, Cipolla Roberto. Posenet: A convolutional network for real-time 6-dof camera relocalization // Proceedings of the IEEE international conference on computer vision. — 2015. — P. 2938–2946.

[20] Kneip Laurent. OpenGV: A library for solving calibrated central and non-central geometric vision problems. — URL: https://laurentkneip.github.io/opengv/ (online; accessed: 2021-11-17).

[21] Kneip Laurent, Li Hongdong. Efficient computation of relative pose for multi-camera systems // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. — 2014. — P. 446–453.

[22] Larsson Viktor. PoseLib - Minimal Solvers for Camera Pose Estimation. — URL: https://github.com/vlarsson/PoseLib (online; accessed: 2021-11-17).

[23] Larsson Viktor, Astrom Kalle, Oskarsson Magnus. Efficient solvers for minimal problems by syzygy-based reduction // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. — 2017. — P. 820–829.

[24] Li Hongdong, Hartley Richard, Kim Jae-hak. A linear approach to motion estimation using generalized camera models // 2008 IEEE Conference on Computer Vision and Pattern Recognition / IEEE. — 2008. — P. 1–8.

[25] Lowe David G. Distinctive image features from scale-invariant keypoints // International journal of computer vision. — 2004. — Vol. 60, no. 2. — P. 91–110.

[26] Minimal Cases for Computing the Generalized Relative Pose Using Affine Correspondences / Banglei Guan, Ji Zhao, Daniel Barath, Friedrich Fraundorfer // Proceedings of the IEEE/CVF International Conference on Computer Vision. — 2021. — P. 6068–6077.

[27] Nistér David. An efficient solution to the five-point relative pose problem // IEEE transactions on pattern analysis and machine intelligence. — 2004. — Vol. 26, no. 6. — P. 756–770.

[28] ORB: An efficient alternative to SIFT or SURF / Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary Bradski // 2011 International conference on computer vision / Ieee. — 2011. — P. 2564–2571.

[29] Pless Robert. Using many cameras as one // 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings. / IEEE. — Vol. 2. — 2003. — P. II–587.

[30] 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.

[31] Raguram Rahul, Frahm Jan-Michael, Pollefeys Marc. A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus // European Conference on Computer Vision / Springer. — 2008. — P. 500–513.

[32] Schönberger Johannes Lutz. COLMAP. — URL: https://colmap.github.io/ (online; accessed: 2021-11-18).

[33] Schonberger Johannes L, Frahm Jan-Michael. Structure-from-motion revisited // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2016. — P. 4104–4113.

[34] Stewénius Henrik, Oskarsson Magnus, Aström Kalle, Nistér David. Solutions to minimal generalized relative pose problems. — 2005.

[35] Stewenius Henrik, Engels Christopher, Nistér David. Recent developments on direct relative orientation // ISPRS Journal of Photogrammetry and Remote Sensing. — 2006. — Vol. 60, no. 4. — P. 284–294.

[36] Superglue: Learning feature matching with graph neural networks / Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, Andrew Rabinovich // Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. — 2020. — P. 4938–4947.

[37] Sweeney Chris, Flynn John, Turk Matthew. Solving for relative pose with a partially known rotation is a quadratic eigenvalue problem // 2014 2nd International Conference on 3D Vision / IEEE. — Vol. 1. — 2014. — P. 483–490.

[38] Szeliski Richard. Computer Vision: Algorithms and Applications // Computer Vision: Algorithms and Applications. — 2 edition. — Springer, 2021. — P. 510–513.

[39] Terekhov Mikhail. Generalized Relative Pose Estimation. — URL: https://github.com/MikhailTerekhov/grpose (online; accessed: 2021-11-21).

[40] Ventura Jonathan. Multi-camera Motion: Efficient minimal solvers for multi-camera motion. — URL: https://github.com/jonathanventura/multi-camera-motion (online; accessed: 2021-11-17).

[41] Ventura Jonathan, Arth Clemens, Lepetit Vincent. An efficient minimal solution for multi-camera motion // Proceedings of the IEEE International Conference on Computer Vision. — 2015. — P. 747–755.

[42] Zhao Ji, Guan Banglei. On Relative Pose Recovery for Multi-Camera Systems // arXiv preprint arXiv:2102.11996. — 2021.

[43] A survey on visual-based localization: On the benefit of heterogeneous data / Nathan Piasco, Désiré Sidibé, Cédric Demonceaux, Valérie Gouet-Brunet // Pattern Recognition. — 2018. — Vol. 74. — P. 90–109.