Санкт-Петербургский Государственный Университет

Математическое обеспечение и администрирование информационных систем
Системное программирование

Дымникова Наталья Александровна

Выпускная квалификационная работа

Определение взаимного положения камер по линиям

Научный руководитель:
к.ф.-м.н., доцент кафедры системного программирования Вахитов А. Т.

Рецензент:
к.ф.-м.н. Кривоконь Д. С.

Санкт-Петербург
2017

SAINT-PETERSBURG STATE UNIVERSITY

Software and Administration of Information Systems
Software Engineering

Natalia Dymnikova

Relative camera pose using line correspondences

Graduation Project

Scientific supervisor:
PhD Docent Alexander Vakhitov

Reviewer:
PhD Dmitrii Krivokon

Saint-Petersburg
2017

Оглавление

Введение

Компьютерное зрение – это теория и технология создания инструмента, способного обнаруживать, классифицировать, распознавать некоторые объекты, то есть получать информацию из изображения. Основные задачи компьютерного зрения – это такие задачи, как распознавание объектов, детекция движения, восстановление сцены, то есть построение 3D модели, восстановление изображения, то есть удаление шума.

Также методы компьютерного зрения все чаще используются для разработки автономных машин, летательных аппаратов и дронов, которые должны перемещаться самостоятельно, без вмешательства человека. В связи с этим среди прочих технологий разрабатываются алгоритмы одновременной локализации и картографии (Simultaneous Localization and Mapping – SLAM), которые уже давно доказали свою эффективность для точечной оценки траектории во время геометрического построения неизвестной среды.

Эти алгоритмы прекрасно работают для хорошо текстурированных кадров и их последовательностей. В то время как слабо текстурированные сцены – слабое место в геометрических алгоритмах компьютерного зрения, которые опираются на соответствие точек на нескольких кадрах для определения взаимного положения нескольких камер. Слабо текстурированные сцены – это такие сцены, как городские и интерьерные кадры, на которых преобладают линии. Таким образом, на практике часто встречаются ситуации, когда известны общие линии, а не точки.

В следствии этого, существует необходимость в разработке алгоритма для решения задачи определения взаимного положения камер по линиям, а не по точкам, как это было сделано в алгоритмах одновременной локализации и картографии.

1. Постановка задачи

Конечной целью работы является реализация алгоритма определения взаимного положения трех камер при разном повороте и смещении по нескольким линиям на кадрах.

Алгоритм должен по соответствующим линиям на кадрах трех камер возвращать матрицы поворота и смещения относительно центральной камеры.

В рамках данной дипломной работы поставлены следующие задачи:

- разработать алгоритм нахождения взаимного положения камер по линиям;
- реализовать алгоритм для разных параметризаций матрицы поворота;
- протестировать на случайных данных и сравнить результаты разных параметризаций матрицы поворота;
- протестировать на реальных данных;

2. Обзор

2.1. Существующие решения

На данный момент существует решение для задачи определения относительного положения камер по точкам. Однако на практике оно не всегда работает из-за существования слабо текстурированных сцен. С точки зрения компьютерного зрения слабо текстурированные сцены отличаются от сильно текстурированных тем, что на них проще детектировать линии, а не другие объекты. Это, например, городские пейзажи и интерьерные сцены.

Этот случай, как оказывается, требует отдельного подхода к решению. Линии нельзя обрабатывать как набор точек, потому что на различных снимках могут быть различные части одной линии и установить точное соответствие между точками оказывается невозможно.

Задача определения взаимного положения камер по линиям была реализованна в нескольких работах. Во первых, в работе [4], однако в ней используется 10 и более линий, при этом алгоритм не всегда работает стабильно и очень чувствителен к наличию шума.

Также этот метод решения проблемы относительного движения используется в работе [2], где используется 3 линии, две из которых параллельны между с собой и перпендикулярны третьей, что накладывает соответствующие ограничения на каждый кадр в последовательности.

2.2. Идея решения

Идея решения задачи нахождения взаимного положения камер заключается в том, что составляется система линейных алгебраических уравнений, в которой матрицы поворота и смещение камер – неизвестные, а координаты проекций точек (в нашем случае – линий) являются известными параметрами.

В этом случае проблема сводится к составлению уравнений и к решению системы.

2.3. Метод решения систем алгебраических уравнений

Систему алгебраических уравнений от нескольких переменных можно решать с помощью базиса Грёбнера [7]. Базис Грёбнера – это множество, которое порождает идеал заданного кольца многочленов, обладающее специальными свойствами. Основное свойство этого множества заключается в том, что остаток от деления любого многочлена из данного идеала на многолчены базиса Грёбнера в любом порядке равен 0. А значит можно разложить наши многочлены по этому базису, что позволяет решить систему.

В библиотеке Automatic-Generator решена эта проблема [1]. Данная библиотека позволяет получать функцию solver, которая зависит от параметров системы, благодаря чему можно решить систему в общем виде и получать ответы для конкретных значений параметров.

2.4. Нахождение и сопоставление линий

Библиотека OpenCV [8] предоставляет возможность распознать и сопоставить линии на двух кадрах. С помощью детектора BinaryDetector находятся линии на каждом кадре, далее для них составляются дескрипторы, после чего происходит сопоставление этих дескрипторов для нахождения соответствующих линий.

3. Особенности реализации

3.1. Модель

Для того, чтобы составить систему уравнений, необходимо сначала описать модель. Будем рассматривать три камеры (или одну камеру в трех позициях в разные моменты времени). Построим систему относительно второй камеры и будем считать соответственно поворот и смещение двух камер относительно центральной. R_1, R_3 – матрицы поворота 1ой и 3ьей камер соответственно. t_1, t_3 – вектора смещения, аналогично.

Таким образом: R_2 – единичная матрица, t_2 – нулевой вектор и

$$R_1 = R1_{real} * R2_{real}^{-1};$$

$$t_1 = t1_{real} - t2_{real}$$

$$R_3 = R3_{real} * R2_{real}^{-1};$$

$$t_3 = t3_{real} - t2_{real}$$

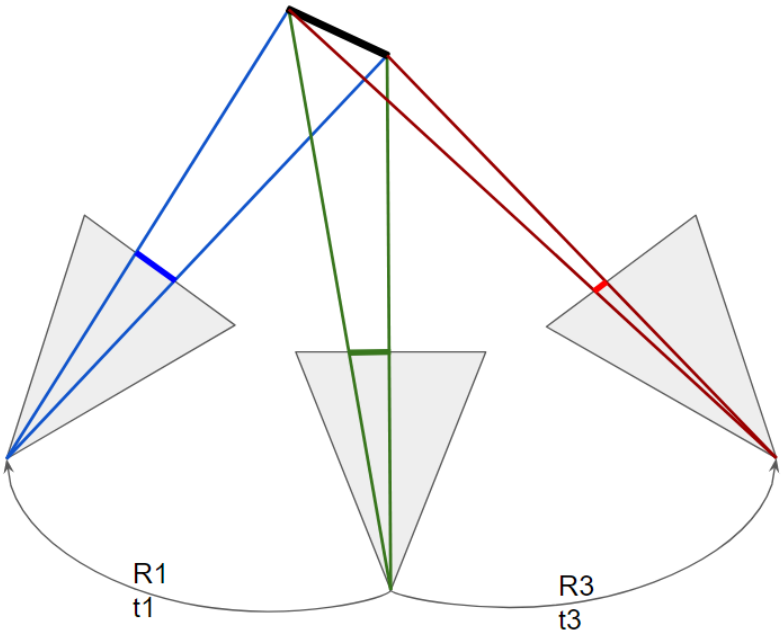

Рис. 1: Три камеры, поворот между левой и центральной – R_1, смещение – t_1, между центральной и правой – R_3 и t_1.

3.2. Проецирование и система уравнений

Для тестирования на случайных данных необходимо получить точки, спроецированные на три камеры. Для этого необходимо получить координаты точек относительно центра камеры, а затем позицию на камере.

$$point = R * real\_point + t$$

$$point = focus * point[1:2]/point[3] + noise$$

где focus – это фокусное расстояние камеры, point[3] – координата z, point[1:2] – координаты x и y. Также можно добавить шум, чтобы тестировать устойчивость системы.

Получив точки на камере, можем получить уравнения прямых по двум концам l_1, l_2, l_3 и затем составить уравнения.

$$((R_1^{-1} * l_1) \times (R_3^{-1} * l_3))^T * l_2 = 0$$

Таким образом, уравнения зависят от координат прямых, а неизвестными являются параметры матриц поворота.

3.3. Параметризации матрицы поворота

Матрицы поворота для разных случаев можно параметризовать различными способами. В данной работе было использовано три подхода.

- Параметризация матрицы поворота кватернионами:

$$R = \begin{bmatrix} a^2+b^2-c^2-d^2 & 2bc-2ad & 2bd+2ac \\ 2bc+2ad & a^2-b^2+c^2-d^2 & 2cd-2ab \\ 2bd-2ac & 2cd+2ab & a^2-b^2-c^2+d^2 \end{bmatrix}$$

  В этом случае для каждой матрицы поворота 4 неизвестных. При этом сумма их квадратов должна равняться единице.

- Параметризация матрицы поворота на небольшой угол:

$$R = \begin{bmatrix} 1 & -r_3 & r_2 \\ r_3 & 1 & -r_1 \\ -r_2 & r_1 & 1 \end{bmatrix}$$

  В этом случае для каждой матрицы поворота 3 неизвестные.

- Параметризация матрицы поворота при известном вертикальном направлении:

$$R_y = \begin{bmatrix} c & 0 & s \\ 0 & 1 & 0 \\ -s & 0 & c \end{bmatrix}$$

  В этом случае для каждой матрицы поворота всего 2 неизвестные, сумма которых равняется 1. При этом общая матрица поворота находится как произведение матриц поворота вокруг каждой оси:

$$R = R_x * R_y * R_z = R_x * R_z * R_y$$

Таким образом, наибольшее количество решений получается для первого варианта параметризации. Этот случай наиболее тяжеловесный, поэтому будем рассматривать случай, когда $R_1 = R_3$. В остальных вариантах матрицы поворота различны.

	Кол. неизв.	Макс. степень	Кол. решений
abcd	4	4	92
known	4	2	6
small	6	2	20

Таблица 1: Зависимость количества решений от количества неизвестных и максимальной степени уравнений при трех разных параметризациях: кватернионами (abcd), при известном вертикальном направлении (known) и при небольшом повороте (small).

3.4. Тестирование

Для тестирования необходимо сравнивать результат и искать ошибку. Мы вычисляем ошибку поворота аналогично работам [6] и [5], а именно: берем столбцы полученной матрицы и реальной, скалярно их перемножаем, поскольку матрицы нормированы, то получаем, что результат этого умножения – это косинус угла между векторами. И ошибкой будем считать максимальный по модулю угол в градусах.

4. Тестирование на синтетических данных

4.1. Модель

Будем рассматривать линии расположенные в кубе 4x4x4 с центром в нуле, а именно точки начала и конца линий. Для того, чтобы иметь возможность добавить небольшой шум к концам линий, точки должны быть максимально далеко друг от друга, для этого будем генерировать точки начала и конца в разных частях куба: координаты начала – от -2 до -1, а координаты конца – от 1 до 2.

Камеры разместим так, что их координаты по модулю между 4 и 6. То есть находятся на промежутке шириной 2 единицы, отстоящем от куба с линиями на 2 единицы. Камеры направлены в начало координат.

Вектор смещения получается из третьей строки матрицы поворота, так как камера смотрит в центр координат.

Также отдельно система протестирована на линиях полученных на одной плоскости. Для этого сначала генерируется плоскость, проходящая через куб, а затем точки на ней.

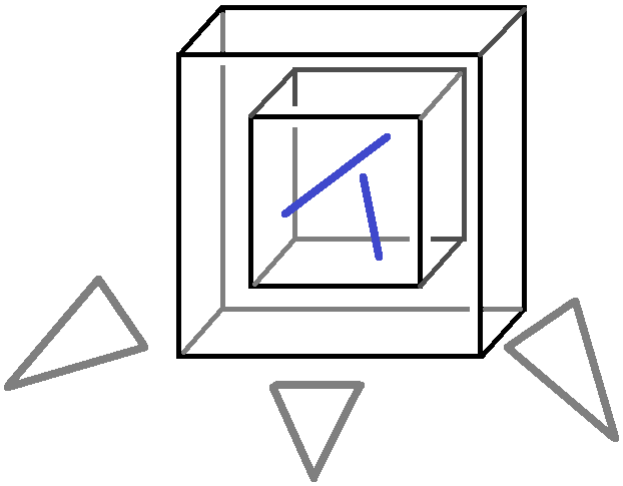

Рис. 2: Внутри большого куба располагается куб поменьше, в котором располагаются линии. Треугольниками указаны камеры, которые направлены в центр куба и находятся на небольшом расстоянии от стенок внешнего куба.

4.2. Параметризация кватернионами

В случае параметризации кватернионами камеры могут располагаться в любом месте друг от друга.

При тестировании в кубе матрица поворота точно находится в 55% случаев, погрешность 10^{-5} также в 55% и 10^{-3} в 70%. Однако метод в этом случае работает недопустимо медленно.

При тестировании на плоскости матрица поворота точно находится в 61% случаев, погрешность 10^{-5} в 61% и 10^{-3} в 67%.

4.3. Параметризация при небольшом повороте

В случае небольшого поворота камеры могут находится только недалеко друг от друга, так как они направлены в центр системы. Для этого сначала генерируется положение центральной камеры, затем матрицы поворота на небольшой угол. Из этого можно получить положение двух оставшихся камер и проверить лежат ли камеры в нужной нам части пространства. В случае неуспеха все повторяется.

При тестировании в кубе матрица поворота точно находится в 99% случаев.

При тестировании на плоскости матрица поворота также точно находится в 99% случаев.

4.4. Параметризация при известном вертикальном направлении

При известном вертикальном направлении положение камер определяется как и при параметризации кватернионами. Далее находится вектор вертикального направления, матрица вращения вокруг оси OY и матрица вращения вокруг двух других осей. Затем необходимо преобразовать координаты линий так, чтобы можно было составить уравнение с неизвестной матрицей вращения вокруг OY, для этого достаточно умножить координаты линий на обратную матрицу поворота вокруг

двух других осей.

При тестировании в кубе матрица поворота точно находится в 16% случаев, погрешность 10^{-5} также в 56% и 10^{-3} в 96%. Однако метод в этом случае работает недопустимо медленно.

При тестировании на плоскости результаты аналогичны.

4.5. Сравнение результатов

Таким образом можно построить таблицу:

R	погрешность == 0	погрешность $<10^{-5}$	погрешность $<10^{-3}$
abcd	55%	55%	70%
small	99%	99%	99%
known	16%	56%	96%

Таблица 2: Процент количества решений с заданной погрешностью при данной параметризации,а именно параметризация кварнионами (abcd), при небольшом повороте (small) и при известном вертикальном направлении (known).

Из этого можно сделать вывод, что наиболее точный случай – это случай с небольшим поворотом, однако в реальности мы не всегда можем гарантировать, что поворот действительно будет небольшим. Вариант с параметризацией кватернионами оказывается достаточно медлительным и неточным, поэтому далее он рассматриваться не будет.

5. Тестирование на реальных данных

5.1. Подготовка данных

Для тестирования на реальных данных использовалась запись – freiburg1_floor из набора данных tum rgbd benchmark. Они представляют собой набор кадров, файл с заданными матрицами поворота и векторами смещения, а также файл с вертикальными направлениями для каждого кадра.

Для начала необходимо найти прямые линии на кадрах, установить соответствие между парами кадров, а затем отфильтровать для троек кадров линии так, чтобы линии были на всех трех кадрах и соответствовали друг другу. Если линий получается меньше, чем необходимое количество для решения линейной системы уравнений, то задачу решить не удастся. Данная функциональность реализована на языке C++ с использованием библиотеки OpenCV, которая позволяет детектировать линии и находить соответствия.

Далее координаты линий сохраняются в файл, отдельный для каждой тройки кадров.

Непосредственно при тестировании эти файлы используются для получения линий, также извлекается информация о повороте и положении матрицы, а для тестирования варианта с известным вертикальным направлением необходимая информация также берется из соответствующего файла.

На изображении ниже можно видеть линии, которые распознались и соответствуют друг другу.

Рис. 3: Последовательность из трех кадров с отмеченными линиями, соответствующими друг другу.

5.2. Уточнение ответа

По скольку не все линии могут детектироваться корректно, то следует уточнить полученный ответ с помощью метода RANSAC (Random sample consensus) [3]. Для этого случайным образом выбираются линии, для них находится решение, считается количество линий, которые удовлетворяют полученному ответу в пределах некоторой погрешности. Вариант, при котором таких линий максимальное количество, и считаем ответом.

Также с помощью RANSAC мы выбираем наиболее близкий ответ.

Количество линий, удовлетворяющих ответу, можно считать несколько способами. Первый способ – просто использовать то же уравнение, что и при составлении системы алгебраических уравнений. Второй способ – с помощью уравнения трифокального тензора.

$$R_1 = [r_{11}, r_{12}, r_{13}]$$

$$R_3 = [r_{31}, r_{32}, r_{33}]$$

$$T_i = r_{1i} * t_3^T - t_1 * r_{3i}^T, \quad i = 1..3$$

$$l_2(i) = l_1^T * T_i * l_3, \quad i = 1..3$$

Таким образом, для этого варианта необходимо посчитать смещение t1 и t3. Это можно сделать используя то же уравнение трифокального тензора для некоторых линий.

Однако при использовании первого подхода средняя ошибка полу-

чается в 2 раза меньше, поэтому будем использовать его.

5.3. Параметризация при небольшом повороте

При параметризации небольшого поворота матрица не является матрицей поворота, то есть перед проверкой ее нужно преобразовать к необходимой матрице. Для этого разложим матрицу по SVD разложению $[U, S, V] = svd(R)$ и перемножим унитарные матрицы $R = U * V'$. И будем сравнивать уже полученные таким способом матрицы.

5.4. Параметризация при известном вертикальном направлении

Из вектора вертикального направления необходимо получить матрицу вращения. Далее преобразовать координаты линий аналогично тому, как это было сделано при тестировании на синтетических данных, а именно умножить координаты линий на обратные матрицы вращения вокруг осей OX и OZ. Для этого необходимо нормировать вертикальный вектор, составить матрицу, такую, что второй столбец равняется этому вектору, а все столбцы составляют правую тройку векторов.

5.5. Результат тестирования

При тестировании без RANSAC решение находится лишь в 50% случаев, однако с использованием решение находится в 100% случаев. Средняя ошибка 0.5 градусов, что допустимо в данной работе. При этом без ошибки матрицы поворота находится более чем в 30% случаев.

Заключение

В рамках данной выпускной квалификационной работы были выполнены следующие задачи:

- разработан алгоритм нахождения взаимного положения камер по линиям;
- реализован алгоритм для разных параметризаций матрицы поворота;
- протестировано на случайных данных и проведено сравнение результатов разных параметризаций матрицы поворота;
- протестировано на реальных данных;

Исходный код можно посмотреть в репозитории проекта https://github.com/NataliaDymnikova/camera-position-by-line

Список литературы

[1] Bujnak Martin. Algebraic solutions to absolute pose problems. — Center for Machine Perception, Department of Cybernetics, 2012.

[2] Elqursh Ali, Elgammal Ahmed. Line-Based Relative Pose Estimation. — 2011.

[3] Fischler M. A., Bolles R. C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. — Magazine Communications of the ACM, 1981.

[4] Kuang Yubin, Oskarsson Magnus, Astrom Kalle. Revisiting Trifocal Tensor Estimation using Lines. — 2014.

[5] Pumarola A., Vakhitov A. et al. PL-SLAM: Real-time monocular visual SLAM with points and lines. — 2017.

[6] Vakhitov Alexander, Funke Jan, Moreno-Noguer Francesc. Accurate and Linear Time Pose Estimation from Points and Lines. — ECCV2016, 2016.

[7] Б. Бухбергер. Компьютерная алгебра: символьные и алгебраические вычисления. — М.: Мир, 1985.

[8] Официальный сайт OpenCV http://opencv.org/ (дата обращения: 17.05.2017).