Санкт-Петербургский государственный университет

Кафедра системного программирования

Рабочий Алексей Александрович

Визуальная одометрия в методах машинного контроля

Выпускная квалификационная работа

Научный руководитель: д. ф.-м. н., профессор Терехов А. Н.

Рецензент: ген. директор ООО «Системы компьютерного зрения» Пименов А.А.

Оглавление

Введение

Важным применением компьютерного зрения является автономная навигация роботов и транспортных средств. Для свободного передвижения в пространстве им необходимо получать данные о положении в нем и информацию об окружающем мире.

Точная локализация транспортного средства является одной из основных и наиболее важных задач. Для автономной навигации, отслеживания движения, обнаружения препятствий и предотвращения столкновения транспортное средство должно хранить информацию о своем положении с течением времени. Для этой цели существуют различные датчики, методы и системы, такие как энкодеры (датчики угла поворота), спутниковые системы навигации GPS и ГЛОНАСС, инерциальные навигационные системы, системы с лазерным или ультразвуковым дальномером.

Однако каждая из перечисленных систем имеет свои недостатки или ограничения. Энкодеры (датчики угла поя ворота) являются самой простой системой оценки положения, но их использование становится невозможным при проскальзывании колес. Спутниковые системы навигации являются наиболее распространенным решением для локализации, поскольку могут обеспечить абсолютное положение без накопления ошибок, но они эффективны только в местах с ясным видом неба и не могут использоваться в закрытых помещениях, а также недостаточно точны для локальной навигации. Оценка местоположения спутниковых систем работает с ошибками порядка метров, в то время как точные приложения требуют точности в сантиметрах, например, для автономной парковки. Дифференциальные и кинематические спутниковые системы в реальном времени могут обеспечить положение с сантиметровой точностью, но эти методы очень дороги. Инерциальные системы навигации склонны к накоплению ошибки. Системы с дальномером сложны в реализации и имеют высокую стоимость.

Производительность современных встраиваемых вычислительных систем позволяет обрабатывать видеопоток технического зрения транспортного средства в реальном времени и использовать его для решения задачи навигации ТС. Поэтому в качестве решения данной задачи предлагается метод визуальной одометрии.

Визуальная одометрия – это надежный метод, позволяющий транспортному средству устойчиво локализовать себя в пространстве, используя данные о смещения ключевых точек, информация о которых получается из анализа последовательности изображений системы технического зрения. Подобная система может работать в недетерминированной(стохастической) среде – среде, в которой не каждое последующее состояние может быть предсказано. Реальный мир – недетерминированная среда.

В данной работе мы фокусируемся на результатах, полученных с помощью камеры, установленной на наземном транспортном средстве.

Работа ведется в библиотеке coreCVS. CoreCVS – библиотека для компьютерного зрения, разработанная в компании ООО «Системы Компьютерного Зрения».

(нужно написать что-то еще)

Постановка задачи

Целью данной работы является разработка инструментария для визуальной одометрии, интегрированного в библиотеку coreCVS, функционирующего в режиме реального времени с низкой задержкой.

Для достижения этой цели были сформированы следующие задачи:

1. Проанализировать существующие подходы к решению задачи.
2. Разработать архитектуры системы.
3. Реализовать систему в рамках библиотеки coreCVS.
4. Провести апробацию и тестирование системы.

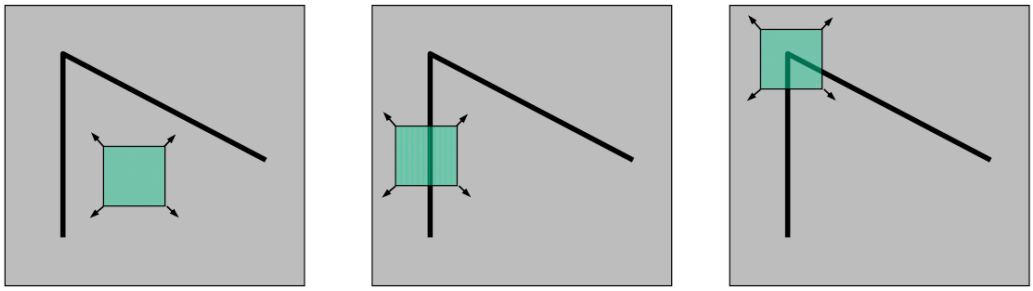
Обзор

Особые точки

Для большинства задач компьютерного зрения нам необходимо искать совпадающие точки на разных кадрах пространства. Если известно, как два изображения связаны друг с другом, можно использовать эти изображения для извлечения детальной информации о них. Говоря о совпадающих точках, имеем в виду некие отличимые фрагменты изображения, которые легко распознать. Эти фрагменты называются особыми точками.

Особая точка m – это точка изображения, окрестность которой o(m) можно отличить от окрестности любой другой точки изображения o(n) в некоторой другой окрестности особой точки o_2(m).

В качестве особых точек мы рассматриваем углы. Угол можно определить как точку, которая формируется двумя или более гранями, определяющими границу между объектами или частями одного объекта. Иначе можно сказать, что угол – это точка, у которой в окрестности интенсивность изменяется относительно центра. Мы можем легко распознать угол, если посмотреть на значения интенсивности в небольшой окрестности. Сдвиг окрестности в любом направлении должен привести к большим изменениям интенсивности, Рис.1.



a) b) c)

Рис.1: Для плоскости (a) не будет изменений при движении в любом направлении, для ребра (b) не будет изменений при движении вдоль самого ребра, для угла (c) будут значительные изменения при движении в любом направлении. [10]

Рис.2: Разные виды углов слева направо: L-связные, Y-связные (или T-связные), стреловидно-связные и X-связные

В качестве углового детектора (метода извлечения особых точек из изображения) мы используем детектор Харриса. Угловой детектор Харриса дает математический подход для определения того, какой случай с Рис.1 выполняется.

Детектор Харриса

Рассмотрим Рис 3.

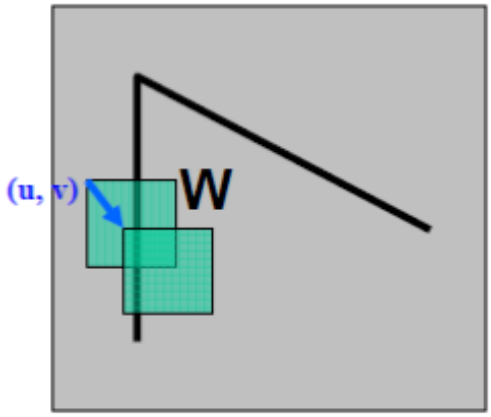


Рис.3: Изображение I, окрестность W с центром в (x, y) и его сдвиг на (u, v). [10]

Изменение интенсивности по некоторому направлению может быть определено взвешенной суммой квадрата разностей:

$$E(u,v) = \sum_{(x,y) \in W} w(x,y)(I(x+u,y+v) - I(x,y))^2 \approx \sum_{(x,y) \in W} w(x,y)(I_x(x,y)u + I_y(x,y)v)^2 \approx (x\ y)\mathbf{H}\begin{pmatrix}x\\y\end{pmatrix},$$

где w (x, y) – окрестность с центром в (x, y), функция весов, Рис.4.

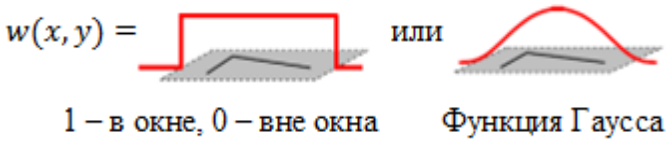


Рис.4: Для вычисления весовой функции используется бинарное окно или функция Гаусса. [10]

H – автокорреляционная матрица:

$$\mathbf{H} = \sum_{(u,v) \in W} w(u,v)\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \qquad I_x = \frac{\partial I}{\partial x}$$

Находим собственные числа матрицы H. Для матрицы 2x2 это будет выглядеть так:

$$det \begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} = 0$$

$$\lambda_{\pm} = \frac{1}{2} \left[(h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2} \right].$$

Зная собственные числа, мы можем найти собственные векторы:

$$\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = 0$$

Собственные числа и векторы матрицы определяют сдвиги с наименьшими и наибольшими изменениями:

U_+ = направление наибольшего увеличения E

λ_+ = = величина роста направления u_+

U_- = направление наименьшего увеличения E.

λ_- = величина роста направления u_-

Угол описывается сильным изменением E по всем возможным направлениям. Следовательно, чтобы точка была определена как угол, необходимо, чтобы E (u, v) было большим для малых сдвигов во всех направлениях. Минимум E (u,v) должен быть большим по всем единичным векторам [u v]. Этот минимум задается меньшим собственным значением λ_-

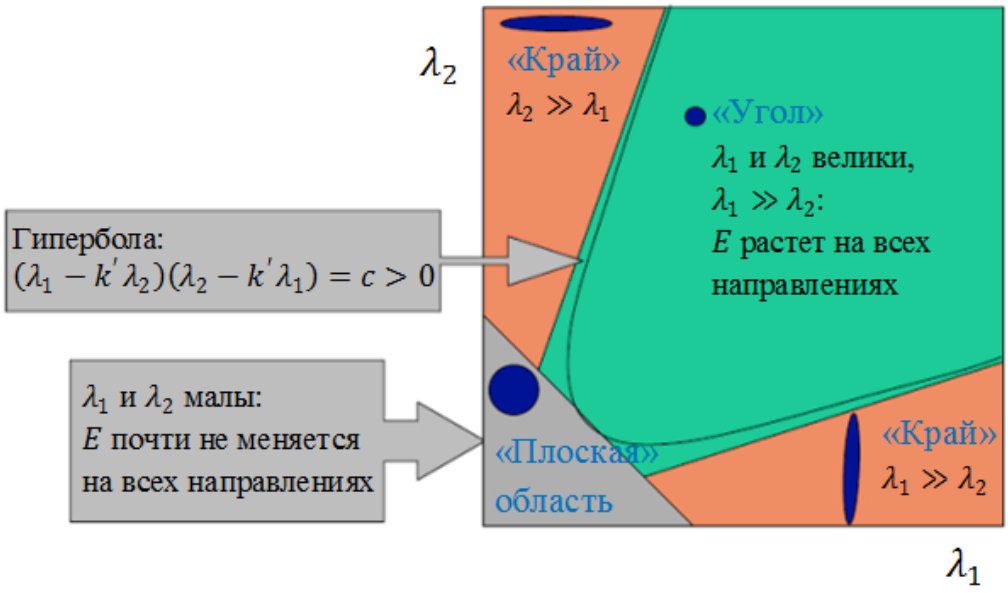


Рис 5: Классификация точек изображения с использованием собственных значений M, заменяя $\lambda\pm$ на $\lambda1$ и $\lambda2$ (в произвольном порядке). [10]

k определяется имперически в интервале [0.04, 0.06].

В итоге, критерием обнаружения особой точки является выражение:

$$(\lambda_1 - k'\lambda_2)(\lambda_2 - k'\lambda_1) = (1 + k')^2\lambda_1\lambda_2 - k'(\lambda_1 + \lambda_2)^2 = (1 + k')^2 det(H) - k' [trace(H)]^2 \geq c'.$$

На практике Харрис предложил [11] использовать меру, объединяющую 2 собственных значения в одну меру, вычисление которой является менее дорогостоящим:

$$R \equiv det(H) - k [trace(H)]^2$$

det(H) = $\lambda_1 * \lambda_2$

trace(H) – сумма главной диагонали = $\lambda_1 + \lambda_2$

После вычисления R рассматриваем только те значения, которые удовлетворяют неравенству R > 0. Затем исключаем точки, значение R которых меньше указанного порога. Находим локальные максимумы R по окрестности заданного радиуса и выбираем их в качестве угловых точек.

Детектор Харриса устойчив к вращениям, частично устойчив к аффинным изменениям интенсивности. Среди недостатков – зависимость от масштаба изображения и чувствительность к шуму.

Оптический поток

Оптический поток – изображение видимого движения краев, поверхностей и объектов изображения между двумя последовательными кадрами, вызванное перемещением объекта или камеры. Это двумерное векторное поле, где каждый вектор представляет собой вектор смещения, показывающий перемещение точек от первого кадра до второго.

Методы оптического потока вычисляют движение между двумя кадрами видеопотока, которые снимаются в момент времени t и t+Δt в каждом положении точки.

Рассмотрим пиксель (x, y, t) в первом кадре с интенсивностью I (x, y, t). Он перемещается по расстоянию (dx, dy) в следующем кадре через время Δt. Так как эти пиксели одинаковы и интенсивность не меняется, мы можем задать ограничение интенсивности:

$$I(x, y, t) = I(x + dx, y + dy, t + dt)$$

Существует несколько подходов к решению задачи вычисления оптического потока. Мы сконцентрируемся на методе Лукаса-Канаде, который стал стандартным для данной задачи.

Алгоритм Лукаса-Канаде

Метод Лукаса-Канаде предполагает, что смещение содержимого изображения между двумя ближайшими моментами (кадрами) мало и приблизительно постоянно в окрестности рассматриваемой точки p. Таким образом, можно считать, что уравнение оптического потока выполняется для всех пикселей внутри окрестности с центром в точке p. А именно, вектор локального потока (скорости) изображения (Vx, Vy) должен удовлетворять

$$\begin{aligned} I_x(q_1)V_x + I_y(q_1)V_y &= -I_t(q_1) \\ I_x(q_2)V_x + I_y(q_2)V_y &= -I_t(q_2) \\ &\vdots \\ I_x(q_n)V_x + I_y(q_n)V_y &= -I_t(q_n) \end{aligned},$$

где $q_1, q_2, \ldots, q_n$ – пиксели внутри окрестности, а $I_x(q_i)$, $I_y(q_i)$, $I_t(q_i)$ – частные производные изображения I относительно позиции x,y и времени t, оцененные в точке q_i в текущий момент времени.

Эти уравнения могут быть записаны в матричной форме Av = b, где

$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix} \quad v = \begin{bmatrix} V_x \\ V_y \end{bmatrix} \quad b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix}.$$

Полученная система имеет больше уравнений, чем неизвестных, поэтому она переопределена. Решаем по принципу наименьших квадратов, получаем систему уравнений 2x2:

$$\begin{aligned} A^T A v &= A^T b \\ \mathrm{v} &= (A^T A)^{-1} A^T b \end{aligned}.$$

В итоге имеем:

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i) I_y(q_i) \\ \sum_i I_y(q_i) I_x(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i) I_t(q_i) \\ -\sum_i I_y(q_i) I_t(q_i) \end{bmatrix}.$$

Обычное решение с наименьшими квадратами дает одинаковое значение для всех n пикселей qi в окрестности. На практике же лучше придать больший вес пикселям, которые находятся ближе к центральному пикселю p. Для этого используется взвешенная версия метода наименьших квадратов:

$$A^T W A v = A^T W b$$

или

$$\mathrm{v} = (A^T W A)^{-1} A^T W b,$$

где W – диагональная матрица размера n*n, содержащая веса $W_{ii} = w_i$, которые должны быть присвоены уравнению пикселя qi. Получаем

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i w_i I_x(q_i)^2 & \sum_i w_i I_x(q_i) I_y(q_i) \\ \sum_i w_i I_x(q_i) I_y(q_i) & \sum_i w_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i w_i I_x(q_i) I_t(q_i) \\ -\sum_i w_i I_y(q_i) I_t(q_i) \end{bmatrix}$$

В качестве весов w_i используется нормальное распределение расстояния между q_i и p.

Эпиполярная геометрия

Когда мы делаем снимок с помощью камеры-обскура (pinhole camera), мы теряем важную информацию – глубину изображения, то есть как далеко находится каждая точка изображения от камеры, потому что происходит преобразование 3D в 2D. Поэтому, чтобы знать глубину изображения, необходимо использовать более одной камеры

Основные результаты в области восстановления сцены по двум видам относятся к области эпиполярной геометрии.

Если мы используем только левую камеру, мы не можем найти трехмерную точку, соответствующую точке x на изображении, потому что каждая точка на линии OX проектируется в одну точку на плоскости изображения. При использовании стереокамеры разные точки на прямой OX проектируются в разные точки x` в правой плоскости. Таким образом, с этими двумя изображениями мы можем триангулировать правильную 3D-точку.

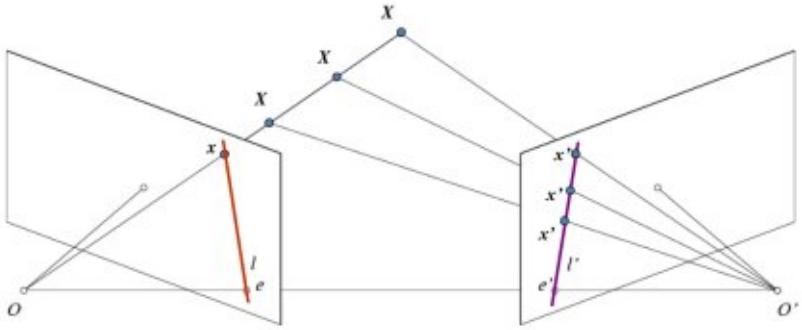


Рис.6: Имеется 2 камеры с центрами проекций в точках O и O’ , плоскости проекций П и П' этих камер, и некоторая точка X в пространстве, имеющая проекцию на каждую из проективных плоскостей, которую мы обозначаем x и x`. [9]

Проекция различных точек на OX (Рис.5) образует прямую на правой плоскости (прямая l`) – эпиполярную линию, соответствующую точке x. То есть для нахождения точки x на правом изображении, нам необходимо пройти исключительно по этой линии, а не по всему изображению. Это называется эпиполярным ограничением. Аналогично все точки будут иметь соответствующие им эпиполярные линии на другом изображении.

Эпиполярная плоскость определяется наблюдаемой точкой P и двумя центрами проекций O и O`.

На Рис.5 можно наблюдать, что проекция правой камеры O` видна на левом изображении в точке e. Данная точка называется эпиполюсом. Аналогично e` является эпиполюсом левой камеры. В некоторых случаях невозможно найти эпиполюс на изображении, он может быть вне его. Это означает, что одна камера не видит другую. Все эпиполярные линии проходят через эпиполюса, поэтому, чтобы найти координаты эпиполюса, мы можем построить много эпиполярных линий и найти точку их пересечения.

Фундаментальная матрица

Фундаментальная матрица – это алгебраическое представление эпиполярной геометрии. Учитывая пару изображений, на Рис.6 видно, что для каждой точки x в одном изображении существует соответствующая эпиполярная линия l` на другом изображении. Любая точка x` во втором изображении, соответствующем точке x, должна лежать на эпиполярной прямой l`. Эпиполярная линия – это проекция на второе изображение луча из точки x через центр камеры C первой камеры. Таким образом, имеется отображение

$$\mathbf{x} \mapsto \mathrm{l}'$$

от точки в одном изображении до соответствующей ей эпиполярной линии на другом изображении. Это отображение является сингулярной корреляцией, то есть проективным отображением от точек к линиям, которое представлено матрицей F, фундаментальной матрицей.

Отображение от точки на одном изображении до соответствующей эпиполярной линии на другом изображении может быть сведено к двум шагам. На первом этапе точка x отображается в некоторую точку x` другого изображения, лежащего на эпиполярной линии l`. Эта точка x` является потенциальным совпадением для точки x. На втором этапе эпиполярная линия l 'получается как линия, соединяющая x` с эпиполем e`.

Шаг 1: Переход через плоскость. Рассмотрим плоскость π в пространстве, не проходящую ни через один из двух центров камеры. Луч через первый центр камеры, соответствующий точке x, пересекает плоскость π в точке X. Затем эта точка X проецируется на точку x` во втором изображении. Эта процедура называется переходом через плоскость $\pi$. Так как X лежит на луче, соответствующем x, то проецируемая точка x' должна лежать на эпиполярной линии l`. Точки x и x` представляют собой изображения трехмерной точки X, лежащей на плоскости. Множество всех таких точек $x_i$ в первом изображении и соответствующие точки $x`_i$ во втором изображении проективно

эквивалентны, так как они проективно эквивалентны плоскому множеству точек x_i. Таким образом, имеется двумерная гомография H, отображающая каждое x_i в $x`_i$.

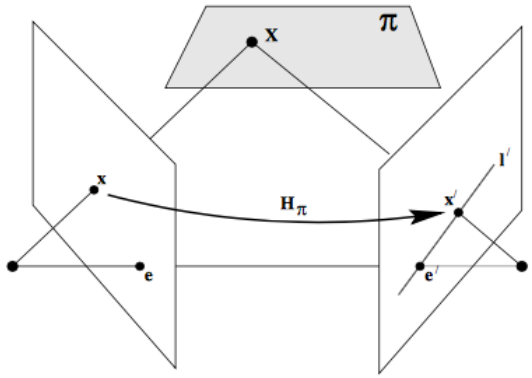


Рис.7: Точка x в одном изображении переносится через плоскость π в точку соответствия x' во втором изображении. Эпиполярная линия через x` получается присоединением x` к эпиполюсу e`. Имеем x`= Hπx и l` = [e`] × x` = [e`] × Hπx = Fx, где F = [e`] × Hπ - фундаментальная матрица. [6]

Для вектора e = (a1, a2, a3)^T обозначим кососимметричную матрицу:

$$[\mathbf{e}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}.$$

Шаг 2: Построение эпиполярной линии. Имеем точку x`, эпиполярную линию l`, проходящую через x` и эпиполюс e`. Можем составить уравнение l`= e` × x`= [e`] × x`. Поскольку x` может быть записано как x` = H x, мы имеем

$$\mathbf{l}' = [\mathbf{e}']_\times \mathrm{H}_\pi \mathbf{x} = \mathrm{F}\mathbf{x} \quad ,$$

где F = [e`] × H, фундаментальная матрица.

Это показывает, что фундаментальную матрицу F можно записать в виде F = [e`] × H, где H – отображение одного изображения на другое через любую плоскость π. Кроме того, поскольку [e`] × имеет ранг 2 и H ранг 3, F является матрицей ранга 2.

Таким образом, геометрически F представляет собой отображение из 2-мерного в одномерное проективное пространство и, следовательно, должна иметь ранг 2.

Условие соответствия

Мы рассмотрели отображение x → l`, определенное матрицей F. Теперь мы можем сформулировать основные свойства фундаментальной матрицы.

Фундаментальная матрица удовлетворяет следующему условию: для любой пары соответствующих точек x ↔ x` на двух изображениях

$$\mathbf{x}'^{\mathrm{T}} \mathrm{F}\mathbf{x} = 0.$$

Это верно, потому что, если точки x и x` соответствуют друг другу, то x` лежит на эпиполярной прямой l`= Fx, соответствующей точке x. Другими словами, $0 = x`^{T}l = x`^{T}Fx$. Обратно, если точки изображения удовлетворяют соотношению $x`^{T}Fx = 0$, то лучи, определяемые этими точками, являются компланарными. Это необходимое условие соответствия точек.

Данный способ позволяет характеризировать фундаментальную матрицу без привязки к матрицам камер, то есть учитывая только соответствующие точки изображения. Это позволяет вычислять F из одних только изображений.

Свойства фундаментальной матрицы

Предположим, что у нас есть два изображения, полученные камерами с несовпадающими центрами, тогда фундаментальная матрица F является единственной однородной матрицей 3 × 3 ранга 2, которая удовлетворяет уравнению

$$\mathbf{x}'^{\mathrm{T}}\mathbf{F}\mathbf{x} = 0$$

для всех соответствующих точек x ↔ x`. **Тогда**, в итоге, F обладает следующими свойствами:

- Если F – фундаментальная матрица пары камер (P, P`), то $F^{T}$ является фундаментальной матрицей пары в обратном порядке: (P`, P).
- Для любой точки x на первом изображении соответствующая эпиполярная линия равна l`= Fx. Аналогично $l = F^{T}x`$ представляет эпиполярную линию, соответствующую x` на втором изображении.
- Для любой точки x (отличной от e) эпиполярная линия l`= Px содержит эпиполюс e`. Таким образом, e` удовлетворяет условию $e`^{T}(Px) = (e`^{T}P)x = 0$ для всех x. Отсюда следует, что $e`^{T}F = 0$, то есть e` – левый нулевой вектор F. Аналогично Fe = 0, то есть e – нулевой вектор F.

F – однородная матрица ранга 2 с 7 степенями свободы.

Если x и x` – соответствующие точки, то $X`^{T}Fx = 0$.

l` = Fx – эпиполярная линия, соответствующая x.

l = Fx` – эпиполярная линия, соответствующая x'.

Fe = 0.

$F^{T}e`= 0$.

Существенная матрица

Существенная матрица – специализация фундаментальной матрицы для случая нормализованных координат. Существенная матрица, в отличие от фундаментальной, имеет меньше степеней свободы и обладает дополнительными свойствами.

Нормализованные координаты

Рассмотрим матрицу камеры, разложенную как P = K [R | T], и пусть x = PX – точка на изображении. Если калибровочная матрица K известна, то мы можем инвертировать ее и применить к точке x, чтобы получить точку $\dot{x} = K^{-1}x$. Тогда $\dot{x}$ = [R | T] X, где $\dot{x}$ – точка изображения, выраженная в нормированных координатах. Его можно рассматривать как изображение точки X относительно камеры [R | T], имеющую единичную матрицу I в качестве калибровочной матрицы. Матрица камеры $K^{-1}P$ = [R | T] называется нормализованной матрицей камеры.

Теперь рассмотрим пару нормализованных матриц камер P = [I | 0] и P`= [R | т]. Фундаментальная матрица, соответствующая паре нормализованных камер, обычно называется существенной матрицей и имеет вид

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R} = \mathbf{R}[\mathbf{R}^\mathsf{T}\mathbf{t}]_\times .$$

Определяющее уравнение для существенной матрицы:

$$\hat{\mathbf{x}}'^\mathsf{T}\mathbf{E}\hat{\mathbf{x}} = 0 .$$

В терминах нормализованных координат изображения для соответствующих точек x ↔ x` подстановка $\dot{x}$ и $\dot{x}$` дает $x`^{-T}K`^{T}EK^{-1}x = 0$. Сравнивая это с отношением $x`^{T}Fx = 0$ для фундаментальной матрицы, делаем вывод, что соотношение между фундаментальной и существенной матрицами равно $E = K`^{T}FK$.

RANSAC

Консенсус случайной выборки (RANSAC) – это итеративный метод оценки параметров математической модели из набора наблюдаемых данных, которые содержат выбросы. Алгоритм RANSAC был разработан в качестве метода оценки параметров определенной модели, исходя из набора данных с большим количеством выбросов. Его также можно интерпретировать как метод обнаружения выбросов. Это недетерминированный алгоритм, он дает разумный результат только с определенной вероятностью, причем эта вероятность возрастает по мере увеличения числа итераций. Процент выбросов, которые могут быть обработаны RANSAC, может составлять более 50% всего набора данных.

Простым примером работы RANSAC является вписывание прямой в набор точек на плоскости. Если предположить, что этот набор содержит выбросы, оценка параметров простым методом наименьших квадратов приведет к неверной модели. Причина в том, что он оптимально подходит ко всем точкам, включая выбросы. RANSAC берет в качестве опоры две точки, необходимые для построения прямой, и строит модель, а затем проверяет количество соответствующих точек, используя функцию оценки с заданным порогом.

Рис.8: Набор данных с выбросами, в который нужно вписать прямую. [8]

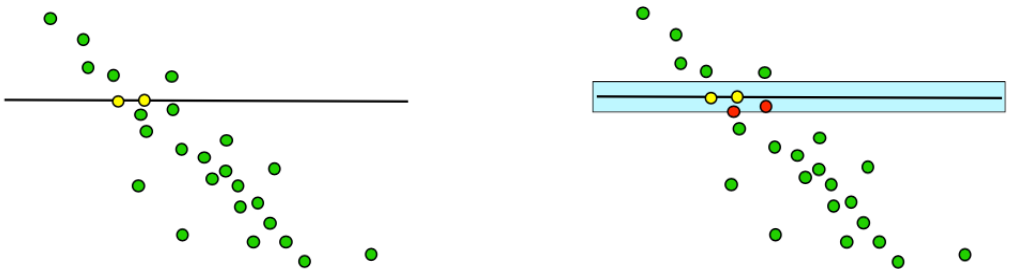


Рис.9: Для полученной модели и указанного порога в результате работы алгоритма будет получено 4 “не выброса”, на данном этапе это лучшая гипотеза. [8]

Рис.10: Для данной модели и порога будет получено 6 “не-выбросов”, эта гипотеза лучше предыдущей, поэтому она замещает ее. [8]

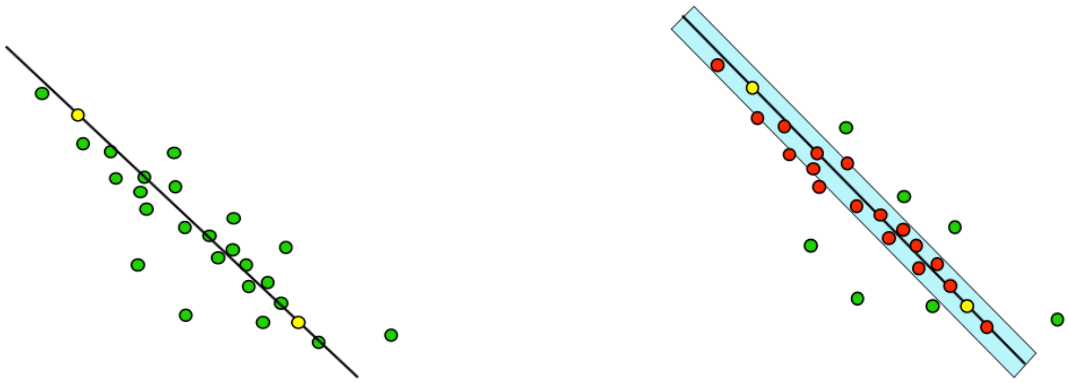


Рис.11: Для данной модели и порога будет получено 19 “не-выбросов”, данная гипотеза лучше предыдущей, замещаем. [8]

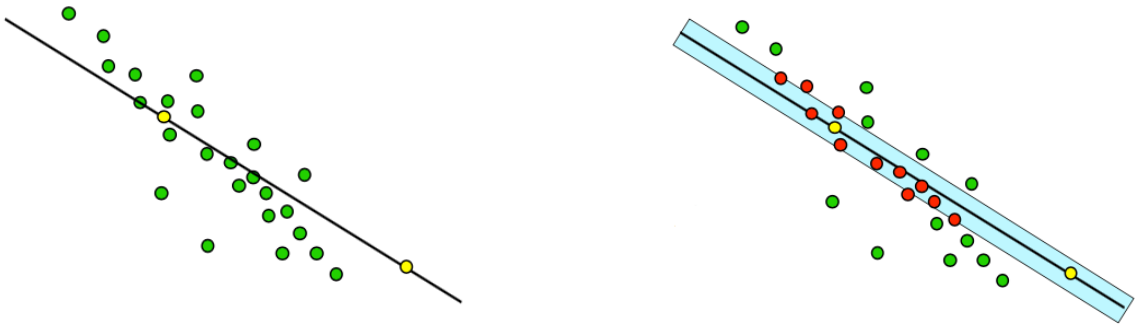


Рис. 12: Для данной модели и порога получится 13 “не-выбросов”, эта гипотеза хуже той, которая на данный момент является наилучшей, идем дальше. [8]

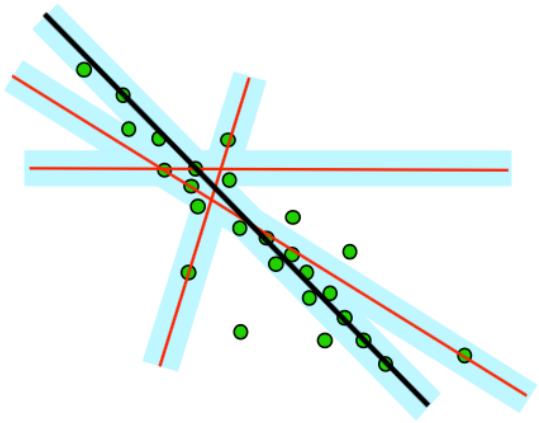


Рис.13: В результате работы алгоритма выбрана гипотеза с 19 “не-выбросами”, что является наилучшим результатом. [8]

Несмотря на множество модификаций, алгоритм RANSAC по существу состоит из двух этапов, которые повторяются итеративно:

1. Случайным образом выбираем подмножество из n различных точек. Мощность n равна наименьшей величине, достаточной для определения параметров модели. Посчитаем фундаментальную матрицу (модель), используя только элементы из выбранного подмножества. Для вычисления F используем 8-точечный алгоритм (метод вычисления описан в [3], раздел 10.1.5, стр. 314-317).
2. RANSAC проверяет, какие элементы всего набора данных удовлетворяют модели, созданной с параметрами, оцененными на первом шаге. Набор таких элементов называется консенсусным множеством.

RANSAC прекращает работу, когда вероятность обнаружения более высоких рангов консенсусного множества падает ниже определенного порогового значения. В исходной формулировке ранжирование консенсусного множества было его мощностью (то есть множества, содержащие больше элементов, ранжированы лучше, чем множества, содержащие меньшее количество элементов).

На вход алгоритма подается:
n – минимальное необходимое количество точек в подмножестве;
k – количество итераций;
t – порог расстояний от прямой до точки, при котором точка считается лежащей вблизи прямой;
d – минимальное количество точек, при котором прямая не считается выбросом.

Результатом работы метода являются параметры модели и точки исходных данных, помеченные «выбросами» или «не-выбросами».

Разложение существенной матрицы

Существенную матрицу можно вычислить из уравнения

$$\hat{\mathbf{x}}'^{\mathrm{T}}\mathrm{E}\hat{\mathbf{x}} = 0 ,$$

используя нормированные координаты изображения, либо вычислить по фундаментальной матрице, используя

$$\mathrm{E} = \mathrm{K}'^{\mathrm{T}}\mathrm{FK}.$$

После вычисления существенной матрицы, матрицы камеры могут быть извлечены из E. В отличие от фундаментальной матрицы, в которой существует проективная неоднозначность, матрица камеры может быть извлечена из существенной матрицы до масштаба и четырехкратной неоднозначности. То есть существует четыре возможных решения.

Мы можем предположить, что первая матрица камеры P = [I | 0]. Чтобы вычислить вторую матрицу камеры P`, необходимо умножить E на произведение SR кососимметричной матрицы и матрицы поворота.

$$[\mathbf{t}]_\times \mathbf{R} = \mathbf{SR}.$$

Мы будем использовать матрицы

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{Z} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Предположим, что SVD для E есть U diag(1, 1, 0) V^T. Используя матрицы, выше получим (игнорируя знаки) два возможных разложения E = SR следующим образом:

$$\mathbf{S} = \mathbf{UZU}^\mathsf{T} \quad \mathbf{R} = \mathbf{UWV}^\mathsf{T} \text{ или } \mathbf{UW}^\mathsf{T}\mathbf{V}^\mathsf{T}.$$

Покажем, что других факторизаций нет. Пусть E = SR. Форма S определяется тем, что ее левое нулевое пространство такое же, как у E. Поэтому S = UZU^T. Вращение R можно записать в виде UXV^T, где X – некоторая матрица вращения. Получаем

$$\mathbf{U}\,\mathrm{diag}(1,1,0)\mathbf{V}^\mathsf{T} = \mathbf{E} = \mathbf{SR} = (\mathbf{UZU}^\mathsf{T})(\mathbf{UXV}^\mathsf{T}) = \mathbf{U}(\mathbf{ZX})\mathbf{V}^\mathsf{T},$$

из которого выводится, что ZX = diag(1, 1, 0). Так как X – матрица вращения, то X = W или X = W^T. Полученная факторизация определяет t матрицы камеры P`, вплоть до масштаба, от S = [t]x. Норма матрицы S = UZU^T равна √2. Это означает, что если S = [t]x, то ||t|| = 1, что является удобной нормализацией для базовой линии двух матриц камеры. Так как St = 0, то t = U(0, 0, 0) ^{T} = u3, последний столбец U. Однако знак E и, следовательно, t не могут быть определены. Таким образом, в соответствии с заданной существенной матрицей существуют 4 возможных варианта матрицы камеры P` на основе двух возможных вариантов выбора R и двух возможных признаков t.

В итоге, для заданной существенной матрицы E = U diag(1,1,0) V^T и первой матрицы камеры P = [I | 0] существует четыре возможных варианта для второй матрицы P` камеры, а именно :

$$\mathbf{P}' = [\mathbf{UWV}^\mathsf{T} \mid +\mathbf{u}_3] \text{ or } [\mathbf{UWV}^\mathsf{T} \mid -\mathbf{u}_3] \text{ or } [\mathbf{UW}^\mathsf{T}\mathbf{V}^\mathsf{T} \mid +\mathbf{u}_3] \text{ or } [\mathbf{UW}^\mathsf{T}\mathbf{V}^\mathsf{T} \mid -\mathbf{u}_3].$$

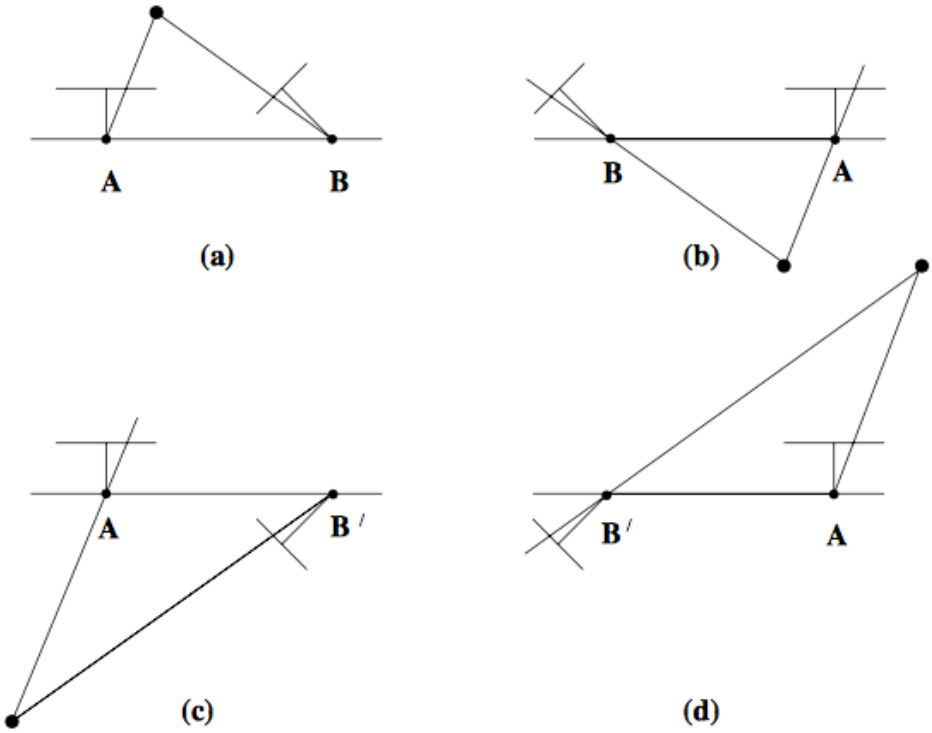


Рис.14: Четыре возможных решения для откалиброванной реконструкции E. Только в (a) – это корректно восстановленная точка перед обеими камерами. [6]

Связная система координат (углы Эйлера)

Углы Эйлера – три угла, которые могут описать ориентацию тела в зафиксированной координатной системе. Любая ориентация может быть достигнута композицией трех элементарных ротаций, то есть поворотов вокруг осей координатной системы. Эйлеровы углы определяются тремя этими поворотами. Существует всего 12 различных комбинаций поворотов.

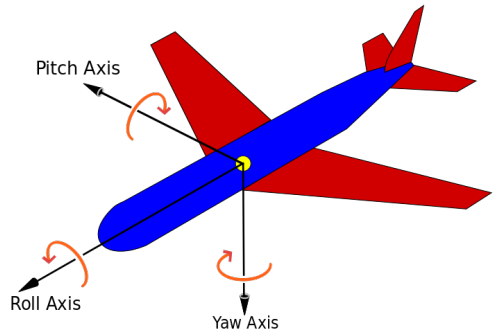


Рис.15: На данном рисунке изображены оси связной системы координат, тангаж (pitch) – поперечная ось, крен (roll) – продольная, рысканье (yaw) – вертикальная. [7]

Архитектура

Общий вид решения задачи визуальной одометрии состоит из следующих шагов:

1. Чтение последовательности изображений.
2. Обнаружение особых точек и вычисление оптического потока.
3. Вычисление существенной матрицы, применение RANSAC.
4. Декомпозиция существенной матрицы.
5. Вычисление углов Эйлера и сдвигов.

Реализация

Средства реализации

Реализация инструментария для визуальной одометрии в библиотеке coreCVS была выполнена в среде QtCreator. В качестве вспомогательных средств использовалась библиотека openCV.

Чтение данных

При чтении данных есть несколько особенностей в реализации: первая связана с тем, что мы работаем не со стереокамерой, а с одиночной, а вторая с тем, что проводить обработку черно-белых изображений намного проще, чем цветных. Поэтому нам нужно, во-первых, если на вход поступает цветной видеопоток, переводить его в черно-белый, для этого в библиотеке coreCVS существует метод toG12Buffer. Во-вторых, нам необходимо хранить два последовательных изображения, для этого мы завели буфер. И при окончании каждого шага мы присваиваем буферу актуальное изображение, а затем считываем новое.

Обнаружение особых точек и вычисление оптического потока

Для обнаружения особых точек и вычисления оптического потока был использован метод getopenCVKLT. Это метод из библиотеки coreCVS, он является оберткой трех методов библиотеки openCV: cvGoodFeaturesToTrack для нахождения особых точек, в качестве детектора используется детектор углов Харриса; cvFindCornerSubPix для уточнения координат особых точек; cvCalcOpticalFlowPyrLK для вычисления оптического потока, используется алгоритм Лукаса-Канаде. На выходе получаем набор векторов потока.

Вычисление существенной матрицы, RANSAC

Из полученных соответствий точек мы можем вычислить существенную матрицу. Мы используем метод RANSAC с 8-точечным алгоритмом. На выходе мы получим устойчивую оценку матрицы и соответствия, помеченные как «выбросы» и «не выбросы».

Декомпозиция существенной матрицы

Для нормированных камер соответствующие системы координат связаны сдвигом и вращением $x` = R(x - t)$, где R – матрица вращения 3×3, а t – трехмерный вектор

трансляции. То есть существенная матрица определяется как E = R[t]x, как было сказано в обзоре.

На выходе RANSAC мы получили существенную матрицу, и нам необходимо выделить из нее R и t. Разложим E в SVD (Сингулярное разложение), получаем U diag(1, 1, 0) V^T. Получится 4 возможных матрицы, указанных в обзоре. Далее выполним триангуляцию и получим, что только для одной из этих матриц точка сцены будет находиться перед обеими камеры. Далее будем рассматривать только этот вариант.

Вычисление углов Эйлера

Получив корректное разложение, можем вычислить углы вращения. Нам необходимо получить кватернион этого разложения. Кватернион описывает поворот вокруг оси на заданный угол (w, vx, vy, vz), где v – ось, выраженная вектором, w – компонента, описывающая поворот (косинус половины угла). Положительное значение угла разворота означает поворот вдоль вектора по часовой стрелке, если смотреть с конца вектора в его начало.

Теперь мы можем узнать углы вращения в связанной системе координат (pitch, yaw, roll) по формулам:

pitch = arcsin(2*x*z – 2*w*y);

yaw = arctg((2*x*w + 2*y*z)/(1-2*x*x-2*z*z));

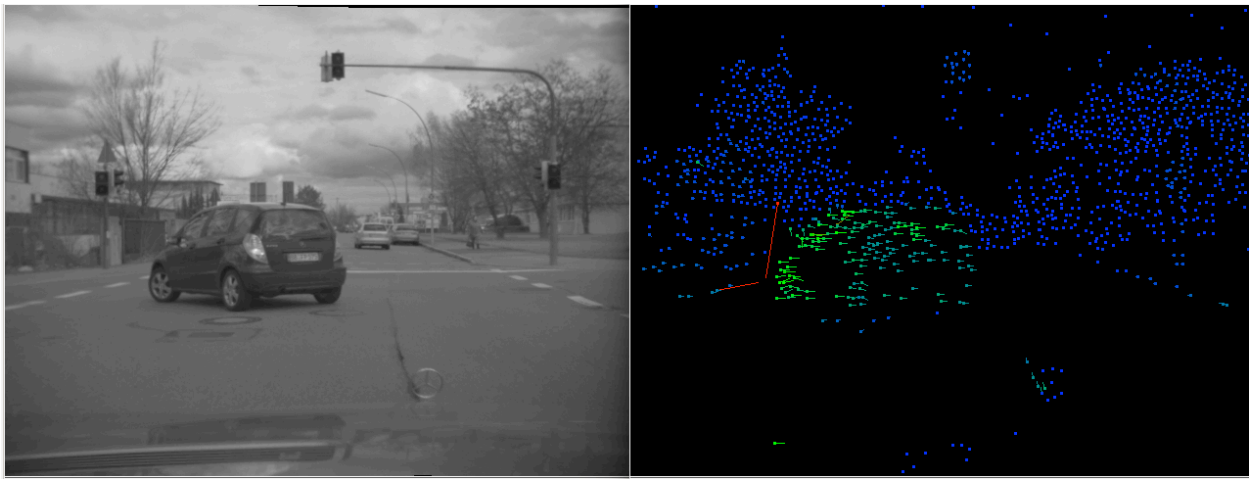
roll = arctg((2*x*y +2*z*w)/(1 – 2*y*y – 2*z*z.

Апробация и тестирование

Апробация

Оптический поток

На данном кадре автомобиль спереди начинает поворот влево (Рис.16a), на изображении оптического потока можно увидеть характерные вектора, показывающие движение данного автомобиля (Рис.16b).

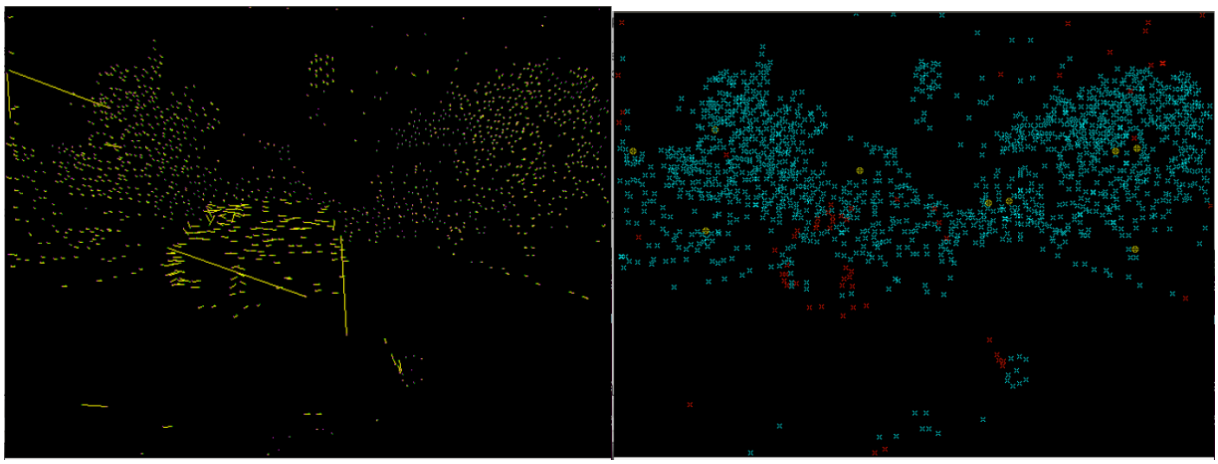


a) b)

Рис.16: a) Кадр из входного видеопотока b) Результат вычисления оптического потока для данного момента времени

RANSAC

Ситуация аналогична предыдущей, имеем оптический поток изображения в момент поворота впереди едущего автомобиля налево (Рис. 17a). На Рис. 17b наблюдаем результат работы алгоритма RANSAC. Синие точки – это начала векторов, которые алгоритм пометил как “не выброс”, красные точки – выбросы. Среди выбросов наиболее очевидными являются вектора, которые оказались шумом и вектора вращения колес автомобиля.



a) b)

Рис.17: a) Оптический поток b) Результат работы RANSAC

Угловые скорости

Для проверки алгоритма использовался видеопоток, который содержит информацию об углах поворота для каждого кадра. Наиболее значимыми являются значения угла yaw(рысканье), которые описывают непосредственно поворот автомобиля. График данных значений показан на рис.18. Сравнение показало, что погрешность алгоритма не превышает 0.03 радиана(~1.7 градуса), что является неплохим результатом.

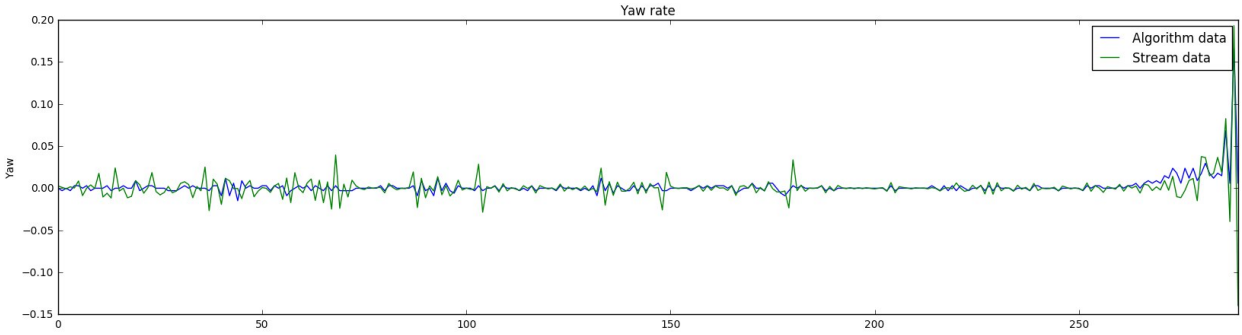


Рис.18: График отображает значения угла yaw(рысканье) в радианах для каждого кадра видеопотока. Синим отмечен результат работы алгоритма, зеленым – реальные значения.

Тестирование

На конкретном примере в алгоритм были поданы синтетические данные – сдвиг на 20 пикселей и поворот на 10 градусов вправо. После обработки этих данных алгоритм вывел такие же значения (Рис.18).

```
Rotation around: [-2.09583e-16, -1, 5.54119e-15] angle 10(10 deg)
Chosen decomposition
This matrix is rotating 10deg around axis: [-2.09583e-16, -1, 5.54119e-15]
And then shifting by vector:
[-2.8856e-13, 4.46084e-16, 1]

pitch: -1.22437e-16rad yaw: -0.174533rad roll: 9.80295e-16rad
pitch: -7.01512e-15 yaw: -10deg roll: 5.61667e-14deg
```

Рис.19: Результат работы на синтетических данных.

При добавлении шумов появляется погрешность, но незначительная (Рис.19).

```
Rotation around: [0.0157011, -0.999361, 0.0320951] angle 9.44989(9.44989 deg)
Chosen decomposition
This matrix is rotating 9.44989deg around axis: [0.0157011, -0.999361, 0.0320951
]
And then shifting by vector:
[-0.588189, 0.0122061, 0.808632]

pitch: 0.00217207rad yaw: -0.164832rad roll: 0.00512611rad
pitch: 0.12445 yaw: -9.4442deg roll: 0.293704deg
```

Рис. 20: Результат работы на зашумленных синтетических данных.

Заключение

В результате данной работы был реализован и интегрирован в библиотеку coreCVS инструментарий для визуальной одометрии.

Были выполнены все поставленные задачи:

- Проведен анализ существующих подходов к решению задачи;
- Разработана архитектура системы;
- Разработана система оценки движения и ориентации камеры в библиотеке coreCVS на языке C++;
- Проведена апробация и тестирование системы, которые показали, что результаты работы системы удовлетворяют требованиям.

Список литературы

[1] David Nister, Oleg Naroditsky, James Bergen «Visual Odometry for Ground Vehicle Applications»

[2] Marco Zuliani, 2014, «RANSAC for Dummies»

[3] Форсайт Д.А., Понс Ж., 2004, «Компьютерное зрение. Современный подход»

[4] Richard Szeliski, 2010, «Computer Vision: Algorithms and Applications»»

[5] Gary Bradski, Adrian Kaehler, 2008, «Learning OpenCV»,

[6] Richard Hartley, Andrew Zisserman, 2000, 2003, «Multiple View Geometry in Computer Vision Second Edition»

[7]Wikipedia, «Aircraft principal axes», URL:
https://en.wikipedia.org/wiki/Aircraft_principal_axes

[8] Robert Collins, «Lecture 15. Robust Estimation : RANSAC»

[9] OpenCV Tutorials, «Epipolar Geometry», URL:
http://docs.opencv.org/trunk/da/de9/tutorial_py_epipolar_geometry.html

[10] Allan Jepson, 2011, «Image Features. Part 2»

[11] Chris Harris, Mike Stephens, 1988, «A combined corner and edge detector»