Санкт-Петербургский Государственный Университет

Кафедра Системного Программирования

Рабочий Алексей Александрович

**Визуальная одометрия**

Курсовая работа

Научный руководитель:

Пименов Александр

Санкт-Петербург, 2016

**1.Введение**

В последние годы индустрия компьютерного зрения испытывает большой подъем, вызванный в первую очередь развитием роботостроения и созданием автомобилей управляемых без пилота. Для свободного перемещения в пространстве им необходимо получать информацию об окружающем мире и данные о собственном положении в нем, для чего применяется множество методов, в том числе визуальная одометрия.

В автомобилях без пилота используются различные методы оценки положения, такие как использование энкодеров, отслеживание с помощью GPS или ГЛОНАСС.

Перечисленные способы имеют свои недостатки. Использование энкодеров невозможно при проскальзывании колес. GPS и ГЛОНАСС недостаточно точны для локальной навигации(самые последние спутники могут определить местоположение с точностью до 60см) и неустойчивы в закрытых помещениях.

Визуальная одометрия - процесс определения позиции и ориентации объекта на основе последовательных изображений полученных с камеры. Визуальная одометрия позволяет повысить точность навигационных приборов в работах или транспортных средствах, которые используют любой способ передвижения относительно поверхности. Этот метод используется во многих приложениях робототехники, например эта техника была использована для построения марсохода NASA.

**2.Постановка задачи**

Целью данной ку курсовой работы является разработка инструментария для визуальной одометрии в библиотеке corecvs на языке C++. Для выполнения поставленной задачи необходимо решить следующие задачи:

-Вычисление оптического потока

-Вычисление существенной матрицы

-Применение RANSAC

-Вычисление угловых скоростей(углов Эйлера)

**3. Обзор**

**3.1. Оптический поток**

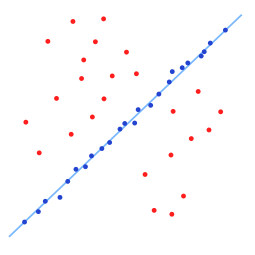
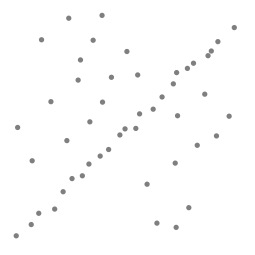
**Оптический поток** – изображение видимого движения, представляющее собой сдвиг каждой точки между двумя изображениями. По сути, он представляет собой поле скоростей(так как сдвиг эквивалентен мгновенной скорости). Суть оптического потока в том, что для каждой точки U(x, y) находится сдвиг (dx, dy) такой, чтобы исходной точке соответствовала точка на втором изображении W(x+dx, y+dy). Для определения соответствия точек надо взять какую-либо функцию точки, которая не изменяется в результате смещения. Обычно считается, что у точки сохраняется интенсивность (т. е. яркость или цвет для цветных изображений), но можно считать одинаковыми точки, у которых сохраняется величина градиента, гессиан, его величина или его определитель, лапласиан, другие характеристики. Очевидно, сохранение интенсивности дает сбои, если меняется освещенность или угол падения света. Тем не менее, если речь идет о видеопотоке, то, скорее всего, между двумя кадрами освещение сильно не изменится, хотя бы потому, что между ними проходит малый промежуток времени. Поэтому часто используют интенсивность в качестве функции, сохраняющейся у точки.

**3.2. RANSAC**

RANSAC (RANdom SAmple Consensus) – стабильный метод оценки параметров модели на основе случайных выборок. Схема RANSAC устойчива к зашумленности исходных данных.

Часто возникает задача обработки данных, в которой необходимо определить параметры модели, которая должна удовлетворять исходным данным. Все исходные данные делятся на два типа: хорошие точки, которые удовлетворяют модели, «не-выбросы»(inlier) и ложные точки, шумы – случайные включения в исходные данные, «выбросы»(outlier).

Например, если рассматривать не особые точки изображений, а просто точки на плоскости, RANSAC позволяет вписать прямую в заданное множество точек так, чтобы расстояние до как можно большего количества точек было наименьшим.



Алгоритм RANSAC часто используется в компьютерном зрении для решения задачи сопоставления изображений и оценки фундаментальной матрицы, для определения параметров расположения камеры.

**3.3. Фундаментальная матрица**

Пусть имеются две камеры, как изображено на рисунке ниже. *C* – центр первой камеры *C´* – центр второй камеры. Точка пространства *X* проецируется в *x* на плоскость изображения левой камеры и в *x´* на плоскость правой камеры. Прообразом луч точки x на изображении левой камеры является луч *xX*. Этот луч проецируется на плоскость второй камеры в прямую *l´,* называемую эпиполярной линией. Образ точки *X* на плоскости изображения второй камеры обязательно лежит на эпиполярной линии *l´*.



Таким образом, каждой точке x на изображении левой камеры соответствует эпиполярная линия l´на изображении правой камеры. При этом пара для x на изображении правой камеры может лежать только на соответствующей эпиполярной линии. Аналогично, каждой x´ на правом изображении соответствует эпиполярная линия l на левом.

Матрица F называется фундаментальной матрицей если пара точек x, x´ является стереопарой тогда и только тогда, когда x´Fx = 0.

В случае, когда матрицы камер имею вид *P = K[I | 0], P´= K´[R | t]* фундаментальная матрица может быть вычислена по формуле:

F = (K´-1)T RKT[KRT t]x.

С помощь помощью фундаментальной матрицы вычисляются уравнения эпиполярных линий. Для точки *x*, вектор, задающий эпиполярную линию, будет иметь вид *l*' = *F* *x*, а уравнение самой эпиполярной линии:

*l*'*T* *x*' = 0. Аналогично для точки *x*', вектор, задающий эпиполярную линию, будет иметь вид *l* = *FT* *x*'.

**3.4. Существенная матрица**

Существенная матрица(essential matrix) – аналог фундаментальной матрицы для откалиброванных изображений. Вычисляется по формуле:

*E* = *K*'*T* *F* *K,* где *K* и *K´* – матрицы камер. По существенной матрице можно восстановить положение и поворот второй камеры относительно первой, поэтому она используется в задачах, в которых нужно определить движение камеры. Связанная система

**3.5. Связанная система координат**

Во время полёта самолёт может описывать сложную траекторию в пространстве. Для описания этого движения вводят оси, связанные с самолётом, и описывают их движение относительно другой системы координат, например, земной. Эта система координат движется и вращается вместе с самолётом.

**Связанная система координат**(Aircraft principal axes) — это система координат, используемая для анализа движения воздушных судов в механике полета. Она состоит из продольной, поперечной и вертикальной осей, которые проходят через центр масс объекта. Продольная ось

**3.5.1. Продольная ось**

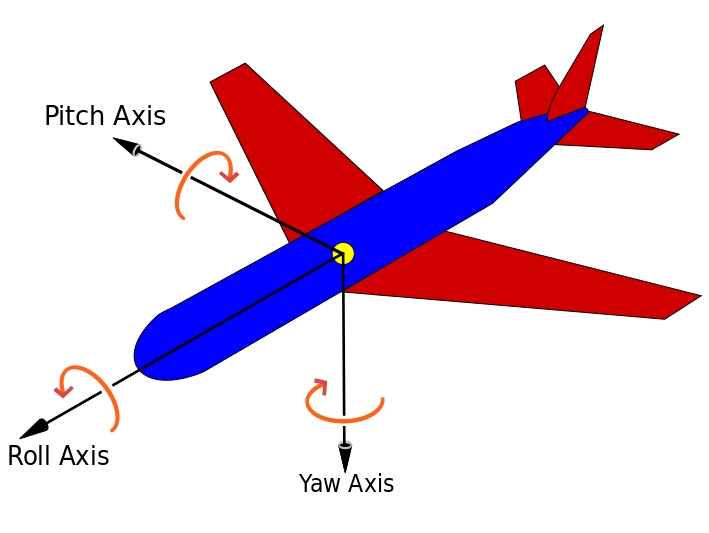
Как правило, в качестве продольной оси используют строительную ось самолёта, которая закладывается при проектировании. При вращении вокруг неё самолёт опускает одну и поднимает другую консоль крыла. Такое движение называется «крен» ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *roll*).

**3.5.2. Вертикальная ось**

Вертикальная ось — ось, лежащая в плоскости симметрии самолёта и перпендикулярная его продольной оси. Вращение вокруг неё называется «[рыскание](https://ru.wikipedia.org/wiki/%D0%A0%D1%8B%D1%81%D0%BA%D0%B0%D0%BD%D0%B8%D0%B5)» ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *yaw*). Самолёт при этом поворачивает нос влево или вправо.

### **3.5.3 Поперечная ось**

Поперечная ось — это ось, перпендикулярная плоскости симметрии самолёта, направленная в сторону правой консоли крыла, дополняющая, таким образом, связанную систему координат до правой тройки векторов. При вращении вокруг этой оси самолёт опускает и поднимает нос. Это движение (и образуемый с горизонтальной плоскостью угол) называется «тангаж» ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *pitch*).



**4. Реализация**

**4.1 Вычисление оптического потока**

**Алгоритм Лукаса-Канаде**

Для вычисления оптического потока мы используем алгоритм Лукаса-Канаде. Этот алгоритм стал стандартным подходом в вычислении оптического потока. В основе этого алгоритма лежит предположение, что значения пикселей переходят из одного кадра в следующий без изменений. Таким образом, мы делаем допущение, что пиксели, относящиеся к одному и тому же объекту, могут сместиться в какую либо сторону, но их значение останется неизменным. Конечно же это предположение имеет мало общего с реальностью, потому что от кадра к кадру могут меняться глобальные условия освещения и освещенность самого движущегося объекта. Масса проблем связана с этим допущением, но, как ни странно, вопреки всему оно достаточно хорошо работает на практике.

Алгоритм:

Рассмотрим математическую модель оптического потока, считая, что у точки в результате смещения не изменилась интенсивность. Пусть

I1 = I(x, y, t1) – интенсивность в некоторой точке (x, y) на первом изображении (т. е. в момент времени t). На втором изображении эта точка сдвинулась на (dx, dy), при этом прошло время dt, тогда

I2 = I(x+dx, y+dy, t1+dt) ≈ I1 + Ixdx + Iydy + Itdt – разложение по Тейлору функции интенсивности до первого члена, It,Iy,Ix – частные производные по координатам и времени, то есть по сути Itdt – изменение яркости в точке

(x, y) между двумя кадрами. Мы считаем, что интенсивность сохранилась, значит I1 = I => Ixdx + Iydy + Itdt = 0. Получаем уравнение с двумя неизвестными – dx и dy.

На изображении объекты размером больше 1 пикселя, значит, скорее всего, в окрестности текущей точки у других точек будут примерно такие же сдвиги. Поэтому мы возьмем окно вокруг этой точки и минимизируем (по МНК) в нем суммарную погрешность с весовыми коэффициентами, распределенными по Гауссу, то есть так, чтобы наибольший вес имели пиксели, ближе всего находящиеся к исследуемому. После простейших преобразований, получаем уже систему из 2 уравнений с 2 неизвестными:



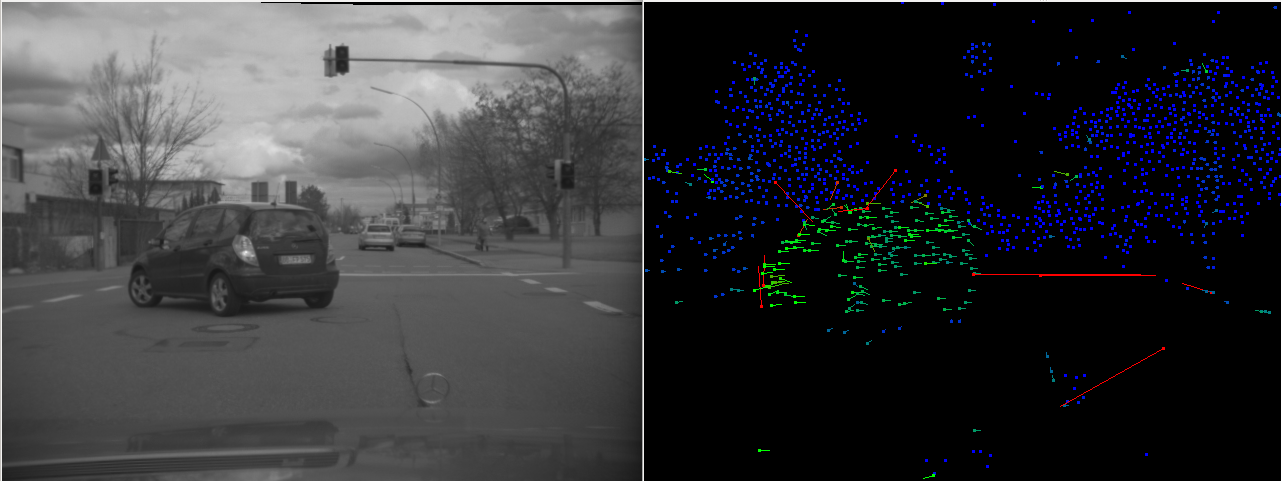
Эта система имеет единственное решение не всегда (хотя и очень часто): если детерминант системы равен нулю, то решений либо нет, либо бесконечное число. Эта проблема известна как Aperture problem – неоднозначность сдвига при ограниченном поле зрения для периодических картинок. Она соответствует случаю, когда в поле зрения попадает фрагмент изображения, в котором присутствует некоторая цикличность.

Если мы остановимся на этом этапе и реализуем этот алгоритм, то он будет успешно работать. Но только если сдвиг между соседними изображениями будет очень маленький, порядка 1 пикселя.

То есть, обычный алгоритм Лукаса-Канаде хорошо определяет маленькие сдвиги, такие, в рамках которых картинка похожа на свое линейное приближение. Чтобы с этим побороться, построим «пирамиду» изображений разного масштаба (почти всегда берется масштабирование в 2 раза по каждой оси, так проще считать) и пройдем по ним оптическим потоком от меньшего изображения к большему, тогда детектированный маленький сдвиг на маленьком изображении будет соответствовать большому сдвигу на большом изображении. На самом маленьком изображении мы обнаруживаем сдвиг не более 1-2 пикселей, а переходя от меньшего масштаба к большему, мы пользуемся результатом с предыдущего шага и уточняем значения сдвига.. Использование этого пирамидального алгоритма позволяет нам просто взять больше уровней пирамиды, а на каждом уровне брать довольно грубое приближение этой функции. Поэтому и идет расчет всего по двум соседним точкам.

Применив этот алгоритм мы получаем массив векторов потока, где начало вектора – точка на первом кадре, а конец – на втором.

Результат на рисунке. Цвет зависит от длины векторов. Например, на данном рисунке автомобиль поворачивает налево, то есть его положение относительно нас начинает сильно меняться, следовательно длины векторов увеличиваются.



**4.2. Вычисление существенной матрицы**

Для первоначальной оценки матрицы используется метод, который принимает в качестве параметра способ вычисления матрицы. Самым простым и известным алгоритмом вычисления фундаментальной матрицы является классический 8-точечный алгоритм.

Поиск фундаментальной матрицы осуществляется из уравнения

u´TFu = 0. Для каждой пары точек получаем одно уравнение относительно F. Важным свойством матрицы *F* является то, что ее ранг равен двум. То есть решение уравнения имеет одну степень свободы. Чтобы избавиться от этого F3,3 считается равным единице и решение ищется по методу наименьших квадратов. Легко показать, что u~T\*u, û'~T'\*u', то F= T*'T* \* F' \* T. Для того, чтобы задача поиска методом наименьших квадратов была хорошо обусловлена, производится процесс нормировки. Центр масс точек при этом переносится в начало координат, а среднее расстояние от начала координат до точек при этом устанавливается равным некоторой величине (оптимально - √2). Далее для нахождения ФМ достаточно применить обратное преобразование, изначально потребовав сингулярность матрицы F'. Далее, зная параметры камеры мы можем получить существенную матрицу по формуле *E* = *K*'*T* *F* *K, где K’ –* матрицы правой камеры*, а K –* матрица левой камеры.

## **4.3. Применение  RANSAC**

Для того, чтобы избавиться от статистических выбросов, в качестве эффективного решения используем метод RANSAC.

**Алгоритм:** .

Алгоритм основан на сборе статистики о входных данных. Из общего набора входных точек случайным образом выбирается некоторое подмножество фиксированного размера, которое аппроксимируется прямой. Общее количество точек входного набора вблизи прямой запоминается. Процесс повторяется несколько раз. Прямая, вблизи которой оказалось наибольшее число точек является наилучшей аппроксимацией всего множества входных точек. .

Чтобы эффективно использовать алгоритм нам необходимо знать сколько итераций выбора подмножеств надо произвести и сколько точек должно быть в подмножестве.

Минимальное количество итераций k, обеспечивающее корректность результата, когда на вход подается n точек, среди которых w точек заданы корректно, а остальные являются шумом, при минимальном возможном количестве точек в подмножестве – 2, задается формулой:  
k = w-n + sqrt(1 – wn)/wn.

Таким образом при известных:

1.Минимально необходимого количества точек в подмножестве – n

2. Необходимого количества итерация – k

3. Пороге расстояний от прямой до точки, при котором точка считается лежащей вблизи прямой – t

4. Минимальном количестве точек, при которых прямая не считается шумом – d  
Алгоритм выглядит так:  
 Пока количество итерация не достигло k:  
 - Выбрать подмножество из n точек случайным образом

-Аппроксимировать текущее множество прямой(например, методом наименьших квадратов)

-Обнулить счетчик близких к прямой точке

-Для каждой точки вне подмножества посчитать расстояние от нее до текущей прямой. Если расстояние меньше или равно порогу t, инкрементируем счетчик

-Если значение счетчика достигло d, аппроксимируем текущую прямую по всем близким точкам  
 -Добавляем прямую в набор «хороших» прямых   
 -Из набора «хороших» прямых выбираем прямую с максимальным счетчиком(количеством точек, близких к данной прямой)

4.4. Вычисление углов Эйлера

Для вычисления углов Эйлера(угловые скорости) нам необходимо получить кватернион существенной матрицы. 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.

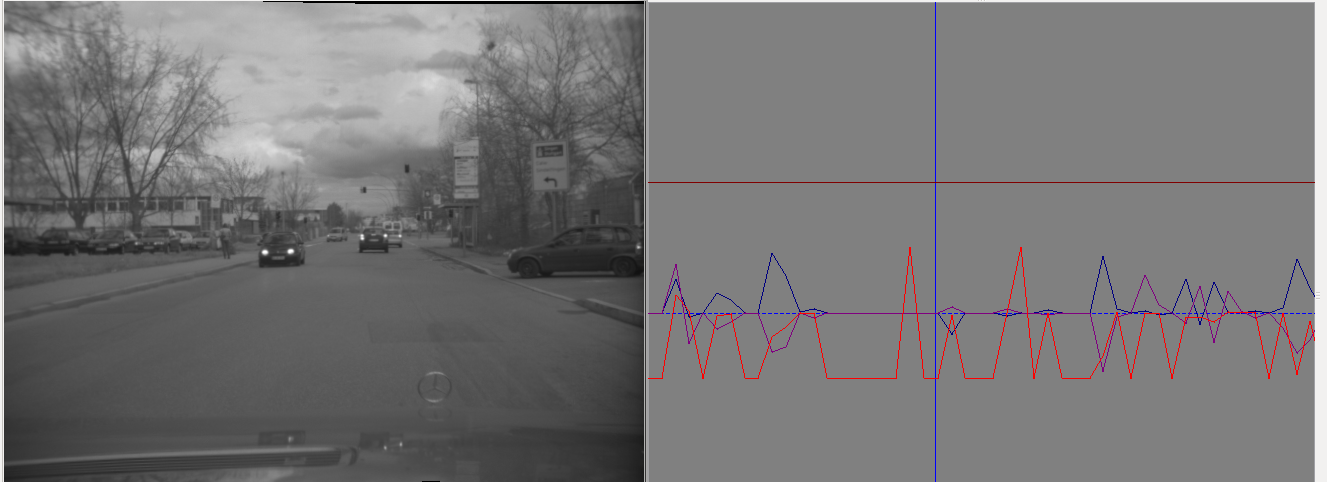
Теперь мы можем узнать углы вращения в связанной системе координат(pitch, yaw, roll) по формулам:

pitch = arctg((2\*x\*w - 2\*y\*z)/(1 - 2\*x\*x - 2\*z\*z));

yaw = arcsin(2\*x\*y + 2\*z\*w);

roll = arct((2\*y\*w - 2\*x\*z)/ (1 - 2\*y\*y - 2\*z\*z));

Результат



На данной картинке изображен график угловых скоростей. Например, в данный момент на видеопотоке автомобиль раскачивается вверх-вниз, и, следовательно, график pitch(красный) сильно колеблется.

**5. Заключение**

# В рамках данной курсовой работы был реализован необходимый инструментарий для визуальной одометрии в библиотеке corecvs. Выполнены все необходимые пункты работы: -Вычислен оптический поток -Применен RANSAC для ликвидации шумов

-Получены графики угловых скоростей, которые качественно совпадают с наблюдаемыми

Данный инструментарий был реализован на языке C++ в библиотеке corecvs и был выложен в репозиторий данной библиотеки.

**6.Литература**

1. Ж. Понс, Д. Форсайт, «Компьютерное зрение. Современный подход»
2. R. Szeliski, «Computer Vision: Algorithms and Applications»
3. Hartley R., Zisserman A. «Epipolar Geometry and the Fundamental Matrix»
4. Peter O’Donovan «Optical Flow: Techniques and Applications»
5. Marco Zuliani «RANSAC for Dummies»
6. Gregory G. Slabaugh «Computing Euler angles from a rotation matrix»