Мой первый околонаучный проект. Инерциальный трекер

Даниил Барков

Осень 2022

Оглавление

Введение

Не буду рассказывать, когда и почему я захотел написать программу для работы с инерциальным сенсором, а начну с того момента, когда я начал изучать линейную алгебру и MATLAB.

Мне открылись новые возможности и методики, и создать прототип программы удалось очень быстро. А ещё я понял, что стремлюсь к научному познанию, и через эту работу я могу наблюдать результаты.

Текст поможет структурировать мысли и сохранить математические выкладки. Однако стили оформления и содержания будут не совпадать. Это потому, что я хочу заодно научиться техать. Вопреки желанию буду писать единицы измерения на английском, т.к. модуль записи математики не понимает кириллицу.

Описание

Сейчас я делаю такую электронную штучку с набором датчиков, которая умеет измерять ускорения, угловые скорости, магнитные поля, и давление. То есть, 10 степеней свободы. Она умеет обрабатывать данные с датчика и выдавать набор чисел, которые отправляются на компьютер по USB. Там их считывает программа, написанная на матлабе и выдаёт такую картинку. При запуске программа определяет свой наклон при помощи акселерометра, затем хитрым образом суммирует повороты по осям, компенсирует дрейф гироскопов с помощью акселерометров и получает набор векторов x, y, z. Ещё оно умеет определять азимут, но пока работает без этого.

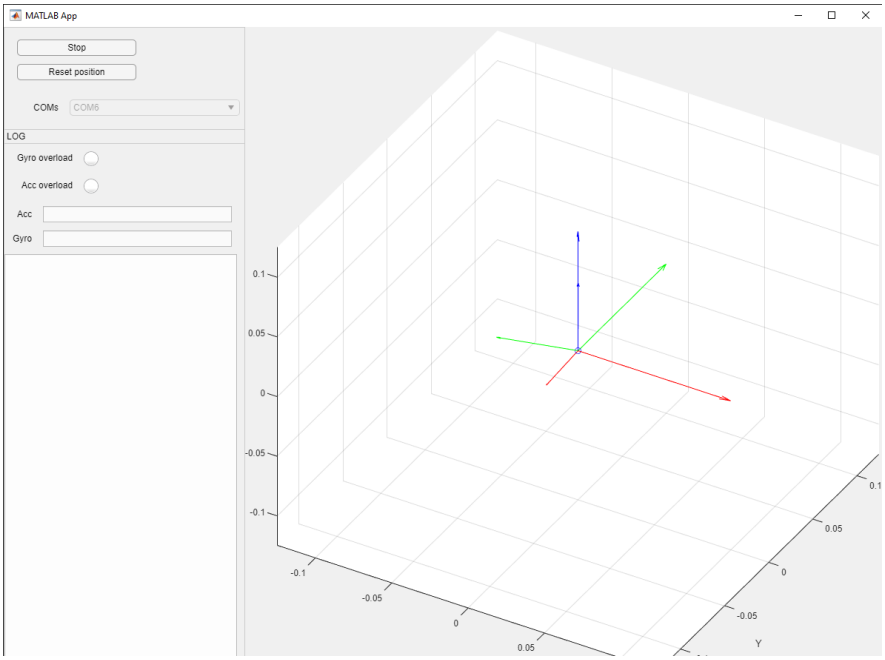

Рис. 1: Моё первое десктоп приложение

Цели и задачи

Пока что план такой - написать библиотеку, функции которой будут принимать данные с датчика, измерять время между приёмами и возвращать необходимые данные:

- трёхмерный базис, векторы которого сонаправлены с датчиком ($B_{(3\times3)}$)

- скорость относительно земли ($\overrightarrow{V}$)
- ускорение относительно земли ($\overrightarrow{A}$) и относительно датчика ($\overrightarrow{A'}$)
- азимут (θ)
- высота над уровнем моря или хотя бы над местом старта (alt)
- время между измерениями акселерометра и гироскопа (τ), магнитометра (τ_m) и барометра (τ_{alt})
- угловые скорости ($\varphi_x,\ \varphi_y,\ \varphi_z$, то есть $\overrightarrow{\varphi}$)
- крен ($\theta_r - roll$) и тангаж ($\theta_p - pitch$) для удобоваримости

$B_{3\times3}$ как бы отражает наклон датчика и ориентацию относительно севера. Он не должен дрейфовать со временем и наклоняться из-за линейного ускорения.

Затем подготовлю шаблон для настройки микроконтроллера (MK), сделать библиотеку для работы с конкретной моделью датчика.

И в конце концов выполню это всё в железе. Очев, начал я с конца, т.к. без устройства было бы тяжело проводить тесты.

Электронная начинка

Главное устройство тут - это модуль 10 DOF IMU sensor (B) от Waveshare, включающий в себя

- 3-х осевой акселерометр (до $\pm156\ m/s^2$)
- 3-х осевой гироскоп (до $\pm2000\ ^\circ/s$)
- 3-х осевой магнитометр (до $\pm4800\mu T$)
- барометр ($300 \sim 1100\ hPa$, т.е $+9000 \sim -500\ m$)
- термометр (даже)

Считыванием и обработкой данных занимается МК STM32F103C8T6, в народе Blue Pill. Работает на 72 MHz и не имеет ускорителя вычислений с плавающей точкой (FPU), поэтому считает относительно медленно. Он собирает по I2C данные с акселерометра и гироскопа каждые $\tau = 4500\mu s$, а магнитометра - $\tau_m = 15000\mu s$. С барометром я пока не работал. Результаты вычислений отправляются на комп через USB-UART конвертер, где их ловит матлаб и выводит их на экран в виде векторов и фонариков перегрузки.

Алгоритм работы и программная реализация

Термины и определения

Перед описанием алгоритма расскажу как и что я назвал. Как рассказал выше, датчик выдаёт такой набор величин:

- ускорение относительно датчика ($\overrightarrow{A'}$)
- давление (P)
- угловые скорости $\overrightarrow{\varphi'}$
- магнитное поле относительно датчика $\overrightarrow{M'}$

Также измеряются $\tau,\ \tau_m\ \tau_{alt}$.

На акселерометр всегда (если он не падает) действует сила тяжести, которую он также регистрирует своими внутренними грузиками - $\overrightarrow{G'}$. По-хорошему он должен иметь значение $\overrightarrow{G'} = (0, 0, -9.8)$, но когда датчик правильно понимает совою ориентацию. Так что оригинальную силу тяжести назовут отдельно - $\overrightarrow{G}$.

Главный термин здесь - базис $B_{3\times3}$. Под ним я подразумеваю три ортонормированных вектора датчика, как-то ориентированных в пространстве. Например, $\overrightarrow{z}$ направлен вверх, $\overrightarrow{x}$ - на север. Получится единичная матрица матрица. Если $\overrightarrow{x}$ смотрит на запад, получится так:

$$B_{(3\times3)} = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Также часто будут использоваться матрицы поворота:
вокруг $\overrightarrow{z}$:

$$rotz(\phi) = \begin{pmatrix} cos(\phi) & sin(\phi) & 0 \\ -sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

вокруг $\overrightarrow{x}$:

$$rotz(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & sin(\phi) \\ 0 & -sin(\phi) & cos(\phi) \end{pmatrix}$$

вокруг $\overrightarrow{y}$:

$$rotz(\phi) = \begin{pmatrix} cos(\phi) & 0 & -sin(\phi) \\ 0 & 1 & 0 \\ sin(\phi) & 0 & cos(\phi) \end{pmatrix}$$

вокруг произвольного $\overrightarrow{l}$:

$$rotl(\phi, \overrightarrow{l}) = \begin{pmatrix} c + (1-c)l_x^2 & (1-c)l_x l_y - sl_z & (1-c)l_z l_x + sl_y \\ (1-c)l_x l_y + sl_z & c + (1-c)l_y^2 & (1-c)l_z l_y - slx \\ (1-c)l_x l_z - sl_y & (1-c)l_y l_z + sl_x & c + (1-c)l_z^2 \end{pmatrix}$$

где

$$c = cos(\phi),\ s = sin(\phi),\ \overrightarrow{l} = (lx, ly, lz),\ |\overrightarrow{l}| = 1$$

Математика

Расскажу о алгоритме, математических формулах, физических явлениях Тут начинается самое интересное. В большинстве случаев разработчиков, например летательный устройств, интересуют углы крена θ_r и тангажа θ_p. Их можно вычислять простым способом: На акселерометр действует $\overrightarrow{G}$, который как-то проецируется на a_x, a_y, a_z, получается $\overrightarrow{A'}$. Затем вычисляется тангаж и крен:

$$\theta_p = -arcsin(\frac{a_x}{|\overrightarrow{A'}|}),\ \theta_r = -arcsin(\frac{a_y}{|\overrightarrow{A'}|}).$$

Но тут возникает проблема. При поступательном ускорении оно складывается с силой тяжести:

$$\overrightarrow{A'} = \overrightarrow{A} + \overrightarrow{G},$$

и θ_r, θ_p меняются. В таком виде формулу использовать нельзя. Отсюда появляется потребность использовать гироскоп для вычисления поворотов. Однако надо задать начальное положение, которое можно получить аналогичным верхнему способом, оставив устройство смирно полежать несколько секунд.

Тут стоит отметить, что гироскоп микроэлектромеханический и гироскоп механический - вещи разные. Второй - это маховик в нескольких рамках, который старается сохранить свою ориентацию (или как-то так, надо разобраться). МЕМС гироскоп же возвращает какую-то среднюю скорость вращения вокруг оси за период измерения. Поэтому не получится, в любой момент обратившись к гироскопу, узнать текущие наклоны. Надо регулярно вычислять повороты и как-то суммировать их.

$$\overrightarrow{\phi} = \overrightarrow{\varphi}\tau = (\phi_x,\ \phi_y,\ \phi_z)$$

Мне в принципе не нравится такое восприятие ориентации потому что оно понятно только оператору самолётика. Да и сложно представить, что происходит с азимутом, когда нос смотрит вверх, какой угол равен 180° при полёте брюхом кверху - тангаж или крен. И почему. А компьютеру это не объяснить и подавно. Надо бы найти универсальное решение: опуститься к естественному представлению тел в нашем трёхмерном пространстве и обратиться к линейной алгебре.

Так вот. Надо как-то складывать повороты, которые (по крайней мере, в моём представлении) некоммутативные. Однако для малых углов этим можно пренебречь. Поворачиваю я так:

$$B_{(3\times3)k+1} = B_{(3\times3)k} \cdot (\ rotx(\tau_k\varphi_{xk}) \cdot roty(\tau_k\varphi_{yk}) \cdot rotz(\tau_k\varphi_{zk})\ )$$

Оказалось, что если перемножать в таком порядке $rotx(\phi_x) \cdot B_{(3\times3)}$, то базис повернётся вокруг абсолютного вектора $\overrightarrow{Z}$. А если наоборот, то вокруг своего $\overrightarrow{z}$.

Все эти повороты начинаются с изначального положения, которое обязательно не вертикально и обязательно не на север. Чтобы ненароком не искривить базис (сделать $|B_{(3\times3)}| \neq 1$), я воспользуюсь испробованным методом.

При включении МК делает пару сотен опросов акселерометра, усредняет показания, получает $\overrightarrow{A'}$, который должен равняться $\overrightarrow{G}$. Но выходит

$$\overrightarrow{G} \neq B_{(3\times3)} \cdot \overrightarrow{A'}.$$

Третье - это данность, первое даже не обсуждается. Неверным остаётся второй элемент. Надо его исправить.

Датчик какой-то стороной наклонён к земле, значит надо наклонить его обратно, то есть повернуть вокруг какой-то оси на угол отклонения. К тому же ось должна проходить через 0 и быть нормальной к обоим векторам
Такая оси:

$$\overrightarrow{l} = \frac{\overrightarrow{A'} \times \overrightarrow{G}}{|\overrightarrow{A'} \times \overrightarrow{G}|}$$

Такой угол:

$$\alpha = arccos(\frac{\overrightarrow{A'} \cdot \overrightarrow{G}}{|\overrightarrow{A'}| \cdot |\overrightarrow{G}|})$$

Такой поворот:

$$B_{(3\times3)} = rotl(\phi, \overrightarrow{l}) \cdot B_{(3\times3)}$$

Дабы уменьшить погрешности в будущем, МК перезапоминает силу тяжести:

$$\overrightarrow{G} = (0,\ 0,\ -|\overrightarrow{A'}|)$$

Аналогичным образом настраивается ориентация по азимуту, только в роли $\overrightarrow{G}$ выступает $\overrightarrow{N} = (1, 0, 0)$, а $\overrightarrow{A'}$ - это $\overrightarrow{M} = B_{(3\times3)} \cdot \overrightarrow{M'}$. С обнулённой вертикальной составляющей. А вращается вокруг $\overrightarrow{Z}$.

Теперь можно вращать датчик и любоваться его отображением в матлабе. Но вот ещё одна проблема. Гироскопы сложно откалибровать, и они очень шумные. К тому же от резких поворотов, у них случаются перегрузы, и все скорости выше порога обрезаются до оного. От этого накапливаются ошибки, и вектора постепенно уплывают. Поэтому необходимо периодически корректировать ориентацию. Делается это описанным выше способом, но чтобы было плавнее, базис доворачивается не на весь угол за раз, а на какую-то подобранную долю угла (в моём случае - 0.02) а функция выполняется при $9.6 < |\overrightarrow{A'}| < 10$.
Про азимут я немного вру, т.к. магнитометр приходится калибровать, а я давно не делал это. Поэтому временно не использую компас, и если с ним возникнет путаница, не обращайте, пожалуйста, внимание.

На этом собственно математика в программе заканчивается. Сюда можно многое добавить. но об этом напишу в другом разделе.

Прототип

Кратко расскажу о том, как я написал прототип в матлабе, как он работал, и почему этого было недостаточно
Изначально все чудеса происходили в матлабе. Мобильное приложение собирало показания с датчиковна телефоне в массив, сохраняло в облачный файл, который затем открывался в матлабовском скрипте. В нём я прототипировал алгоритм, что было очень удобно. Это был прув оф консепт.

Следующим шагом стало наблюдение в реальном времени. МК общался с датчиком и передавал на ПК сырые данные, которые обрабатывались в матлабовском скрипте.

Очень удобно было проводить там калибровку (расскажу позже). Но в радиоуправляемую подводную лодочку ноут не положишь, а к микрокомпьютерам доверия нет, тк даже гоношный ноут напрягается от такой нагрузки. Поэтому стал переносить вычисления на маленький МК .

Программа для ПК

Сейчас приложение лишь принимает 10 байт, 9 из которых - это координаты трёх векторов, помноженные на 100 (векторы в длину меньше 1, а передавать single precision вчетверо дольше), а последний хранит 2 бита для индикаторов перегруза датчиков.

Хорошо бы сделать проверку целостности, перезапуск общения, отправку команд на МК. Но сходу не получилось, а необходимости в этом нет.

Программа для МК

Расскажу о том, как происходит общение с датчиком, о структуре программы и о том, как перенёс матлабовский код в СИ. О скорости работы и о том, что можно улучшить

Матричные операции

Тут я расскажу, о том, как сделал функции удобные, как в матлабе, но они не завелись на МК, и пришлось доделывать. Алгоритм можно было перенести построчно. Но вот готовых функций работы с матрицами у меня не было. Первой мыслью было воспользоваться С компайлером и получить готовый С код. Но это тоже оказалось муторно. Поэтому я решил написать свои матлаб-лайк матричные функции. Мне были необходимы:

- переменная, хранящая произвольный набор float чисел и информацию об их порядке
- умножение на константу
- сумма двух матриц
- умножение двух матриц
- транспонирование
- определитель
- векторное произведение
- скалярное произведение
- нормирование вектора
- модуль вектора
- чтение элемента m_{ij}

- запись элемента
- создание матрицы $m \times n$
- создание единичной матрицы $n \times n$
- вычисление угла между векторами
- матрицы поворота

Матлабовского должно быть вот что:

- матрица выглядит как одна переменная
- функции проверяют возможность выполнения операции
- результат возвращается в виде числа, либо записывается в переменную, указанную в аргументах
- матрицы для результатов не приходится готовить вручную, а функция удаляет и выделяет память для сохранения результата, записывает её параметры

Матрица у меня - это структура:

```
struct matrix {
    float *arr;
    uint8_t rows;
    uint8_t cols;
    uint8_t size;
};
```

Типичная функция выглядит так:
uint8_t типОперации(матрица *1, матрица *2, матрица *результат){
проверяю, что операция выполнима, иначе возвращаю 1
создаю буфер для массива
делаю вычисления, сохраняю в буфер
перевыделяю при необходимости память в *результат
переписываю из буфера в *результат
меняю параметры матрицы в *результат
возвращаю 0
};
Таким образом, можно на ходу перезаписывать матрицы, не заморачиваясь их размерами. Прототипировал я в вижуалке, и всё было хорошо.
Но почему-то на МК были проблемы с выделением памяти. Разбираться, кто виноват не стал, и просто сделал всем матрицам массив по 9 элементов. На удобстве это почти не сказалось, зато могу при компиляции прикинуть количество занятой оперативы. Когда заведу гит, выложу библиотеки туда.

Типа операционная система реального времени

Моё крутое использование micros(), прерываний и отсутствие делэев.

Проблемы

Расскажу о проблемах, связанных с работой датчика, МК и алгоритма

Корявые вычисления с плавающей точкой

Здесь я расскажу о том, то я много вычилсяю тригонометрические функции, которые по природе неидеальны.
О том, что вещественные числа в формате float подразумевают накопление ошибки.
И о том, что я хочу вычислять определитель базиса, чтобы проверять его ортонормированность. Ну и как-то чинить базис в дальнейшем.

Частота измерений датчика и интерполяция

Подумаю, какую интерполяцию использовать, и почему. (до прямых линий между точками)

Шумы и фильтрация

Построю гистограмму шумов покоящихся датчиков на разных диапазонах измерений. Подумаю, нужно ли использовать фильтр. Чего это будет стоить (скорость и память). Подберу варианты.

Калибровка

Объясню необходимость калибровки, опишу текущий метод, расскажу, почему хочу поменять.

Планы на будущее

Расскажу о том, что хочу улучшить, и куда добавить придуманный алгоритм