Санкт-Петербургский политехнический университет Петра Великого

Курсовая работа

по дисциплине:

«Объектно-ориентированное программирование»

на тему:

«Калибровка солнечного датчика солнечной панели малого космического аппарата класса CubeSat»

Студент: Кизь Н. А.

Преподаватель: Ананьевский М.С.

Группа: 3331506/10401

Санкт-Петербург

2024

Оглавление

Введение

Задача ориентации космического аппарата по Солнцу является одной из ключевых при разработке аппарата типа CubeSat. Солнечные датчики используются для измерения углового положения Солнца относительно корпуса спутника, что позволяет корректировать его ориентацию и обеспечивать стабильность при выполнении научных и технологических экспериментов на орбите. Помимо этого, правильное направление солнечных панелей спутника на Солнце делает выработку электроэнергии более эффективной. В данной работе рассматривается одна из частей большого проекта, связанная с обработкой данных и реализацией алгоритма калибровки датчика с использованием языка Python.

Последовательность калибровки

Структура всего проекта схематично отражена на рисунке 1. Здесь нет детального описания работы всех узлов, т.к. данная работа посвящена только его части, отвечающей за обработку данных и алгоритм калибровки датчика.

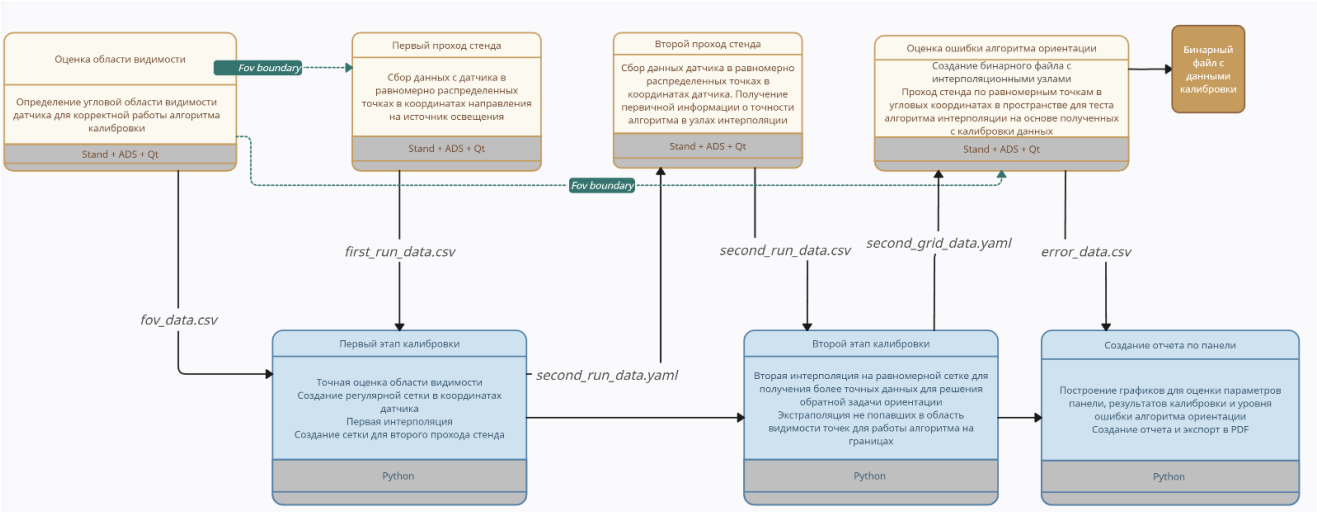

Рисунок 1 – общая структура проекта

Реализация алгоритма калибровки на языке Python

Основная часть проекта реализована на языке C++, но удобной и быстрой работы с многомерным массивом данных было принято решение использовать Python, а взаимодействие между частями проекта, написанными на разных языках, осуществляется с помощью Python API, являющаяся библиотекой, написанной на C.

Конечная задача состоит в том, чтобы определять направление вектора на источник света, имея только данные с солнечного датчика на панели. Датчик представляет из себя четыре независимых фотодиодов, расположенных симметрично относительно центра чувствительной поверхности. На выходе фототоки с фотодиодов посредством АЦП превращаются в цифровые сигналы, используя которые можно определить “центр масс” светового пятна, падающего на датчик. Введем обозначения:

$$x - \text{абсцисса центра светового пятна}$$
$$y - \text{ордината центра светового пятна}$$

Нехитрыми математическими преобразованиями можно получить эти координаты, используя четыре сигнала с фотодиодов:

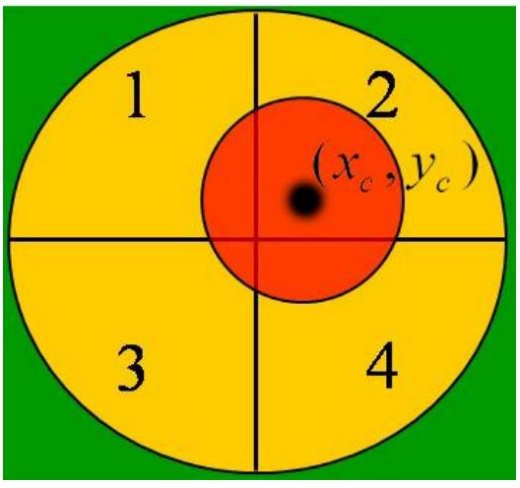

Рисунок 2 – упрощенная схема солнечного датчика

$I_1, I_2, I_3, I_4 -$ цифровые значения интенсивности фотодиодов с датчика

$$\mathrm{x} = \frac{(I_2+I_4)-(I_1+I_3)}{\sum_i I_i}$$

$$\mathrm{y} = \frac{(I_1+I_2)-(I_3+I_4)}{\sum_i I_i}$$

На основе этих значений нам нужно узнать две другие координаты - координаты вектора направления на источник света:

$X -$ абсцисса вектора направления на источник света

$Y -$ ордината вектора направления на источник света

Аппликата вектора остается свободной, исходя из соображений единичности вектора направления на источник:

$$\mathrm{X}^2 + \mathrm{Y}^2 + \mathrm{Z}^2 = 1$$

На основе измерений, снятых посредством поворота датчика по двум угловым осям (азимутальная и зенитная), можно приближенно найти отображение

$$(X, Y) \xrightarrow{A} (x, y)$$

Но задачей ориентации по солнечному датчику является поиск обратного преобразования, чтобы по координатам центра пятна на датчике определять направление на источник:

$$(x, y) \xrightarrow{A^{-1}} (X, Y)$$

Данное преобразование можно найти аналитически, но такой способ не является предпочтительным, т. к. при монтаже датчика на плату появляются начальные смещения центра датчика по координатам и углам, а после заливки панели силиконом появляются эффекты преломления, что привносит существенные ошибки в модель. К тому же, используемая в проекте дискретная модель датчика позволяет определять координаты направления на источник с хорошей точностью в гораздо большем диапазоне углов (до 60

градусов), чем аналитическая, описанная в других научных работах (до 40 градусов).

Процесс калибровки разбит на два схожих этапа, которые позволяют максимально расширить область видимости и создать благоприятные условия для создания обратного преобразования (A^{-1}).

Непосредственное создание обратного преобразования осуществляется в классе Calibration модуля Utils (см. проект). Для создания калибровочных функций используются интерполяторы из библиотеки SciPy (для билинейной интерполяции – LinearNDInterpolator, для бикубической - CloughTocher2Dinterpolator). Бикубическая интерполяция для поиска калибровочной функции показала себя более эффективно, т.к. на краях области видимости датчика появляются нелинейности (см. рисунок 3).

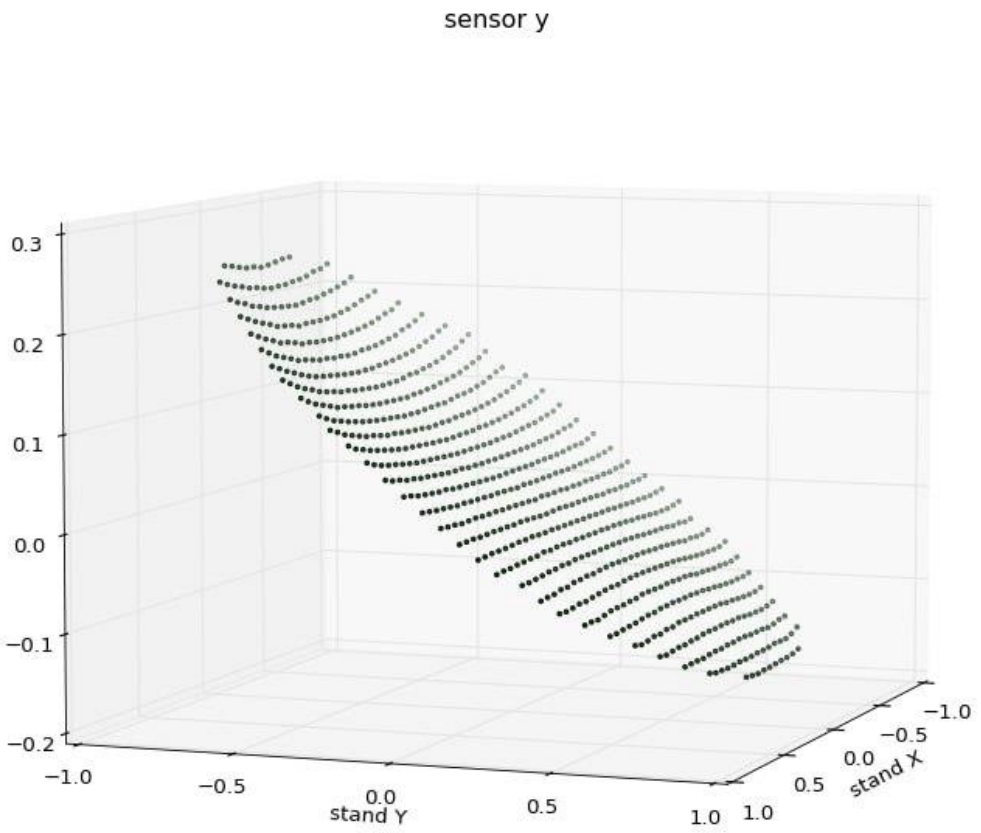

Рисунок 3 – зависимость одной из координат центра светового пятна от координат вектора направления на источник

После нахождения калибровочных функций мы фактически имеем обратное преобразование, а значит можем сформировать равномерную сетку в координатах (x, y) датчика, для того чтобы иметь более равномерное поле с узлами интерполяции во время ориентации спутника. Можно закончить процесс калибровки после первой интерполяции, но тогда мы рискуем потерять в точности алгоритма ориентации. Сравнение сетки, полученной после первого прохода стенда, с сеткой, которая будет использоваться в алгоритме ориентации, показана на рисунке 4.

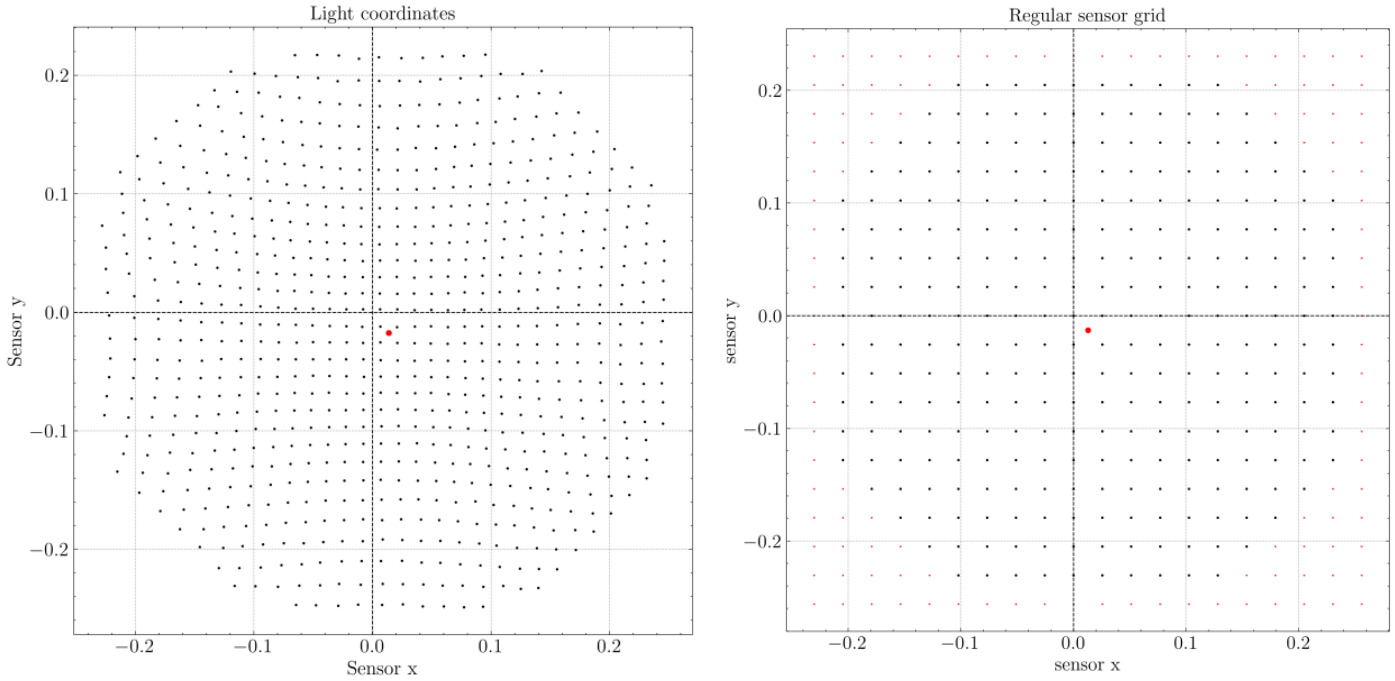

Рисунок 4 – сравнение сетки, полученной после первого прохода стенда, с сеткой, используемой для ориентации

Для каждого узла из правой сетки мы можем найти координаты (X, Y), полученные из калибровочных функций. После этого можно отправить эти данные на стенд для второго прохода, и построить первичный график ошибки алгоритма калибровки для узлов регулярной сетки, а также уточнить алгоритм калибровки посредством получения данных в непосредственной близости к узлам интерполяции. Векторное поле ошибок и его численное представление показаны на рисунках 5 и 6.

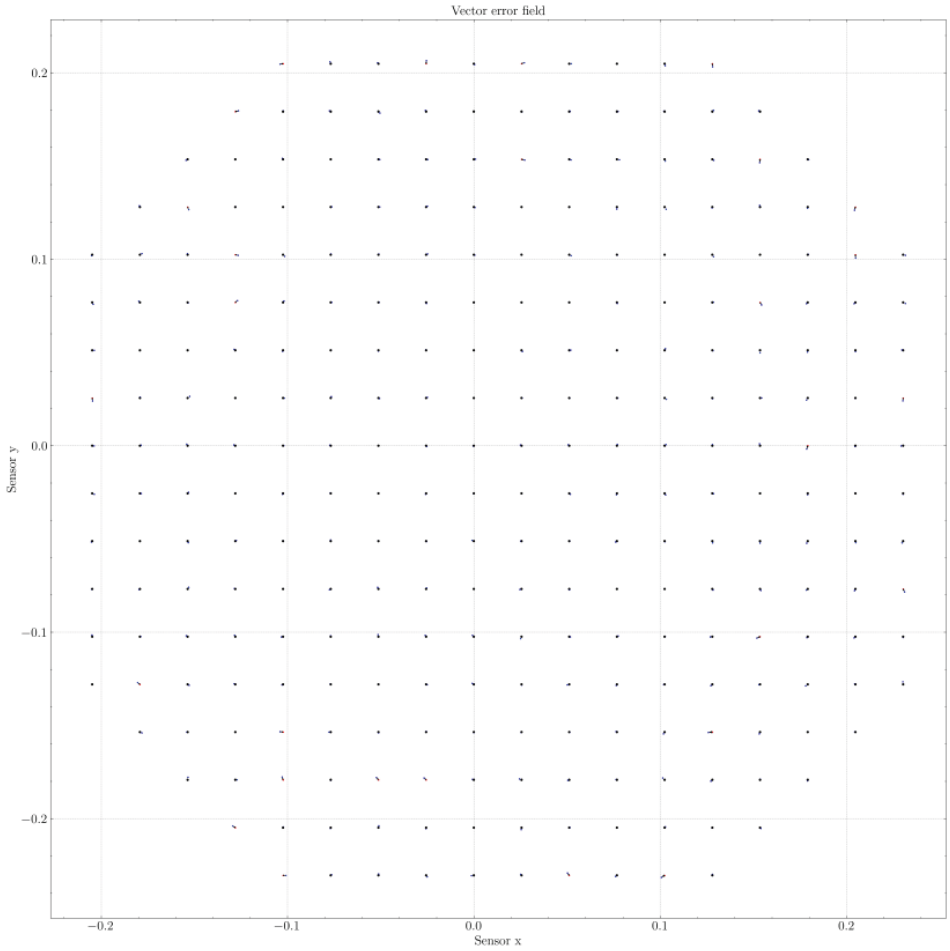

Рисунок 5 – векторное поле ошибок калибровочных функций.

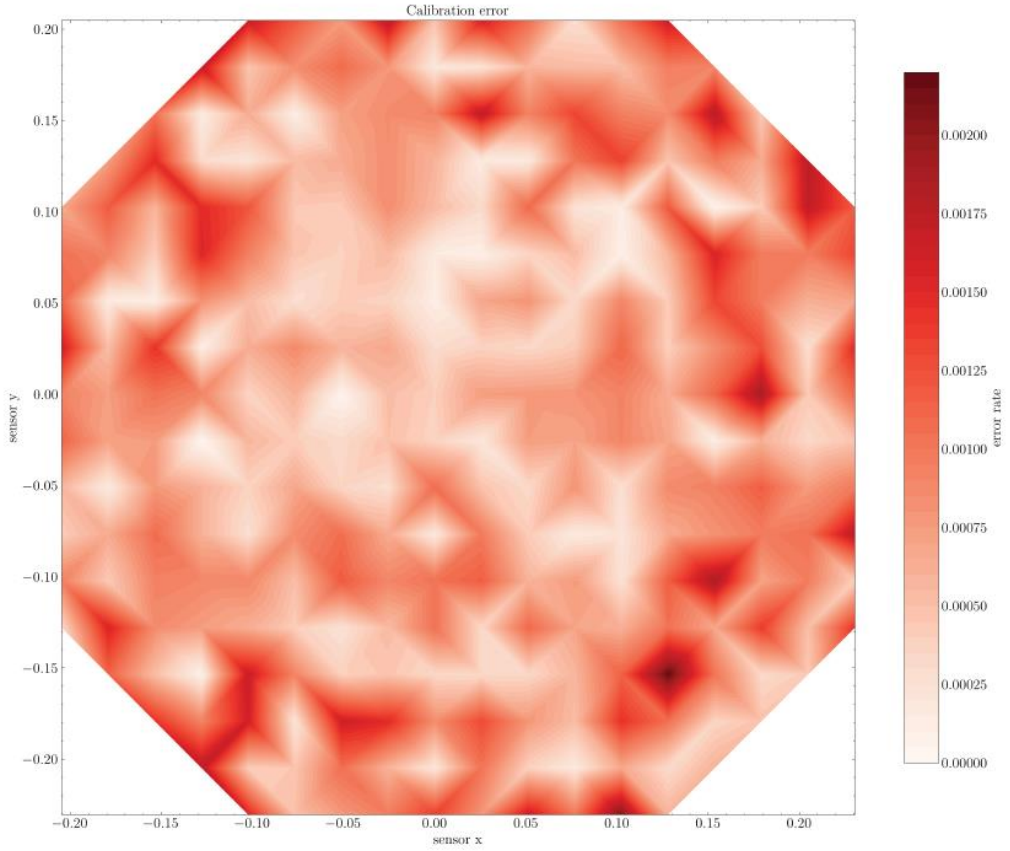

Рисунок 6 – численное представление ошибок калибровочных функций в виде модуля вектора ошибки.

Для оценки этой ошибки можно посмотреть на график шума АЦП датчика, в зависимости от накопления. Накопление – усреднение измерения по определенному количеству. В калибровке на данный момент используется значение 64.

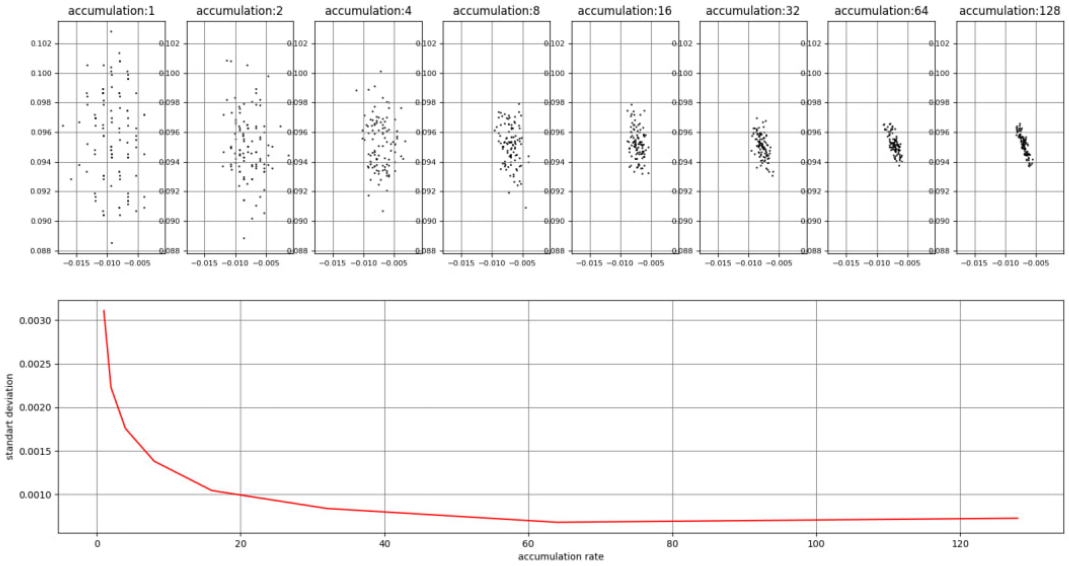

Рисунок 7 – зависимость шума АЦП датчика от уровня накопления

На графике снизу представлено стандартное отклонение от среднего по 50 измерениям с разным накоплением. Для накопления в 64 семпла можно наблюдать разброс по координатам в пределах 0,002 и 0,005 для ординаты и абсциссы, соответственно. Это значит, что ошибки, показанные на векторном поле выше, обусловлены только шумом АЦП и, возможно, другими внешними шумами, и калибровочные функции являются устойчивыми для всей области видимости датчика.

После второго прохода датчика по полученным точкам можно еще больше уточнить калибровочные функции, получив их интерполяцией на основе соединенного массива данных из первого и второго проходов. Однако следует отделить из первого массива данных все те, которые не входят в область видимости, чтобы не получить выбросов.

Также требуется решить проблему с точками, которые невозможно получить с помощью интерполяции на основе экспериментальных данных. В этих точках нет существующих данных в процессе эксперимента, но для правильной работы алгоритма на границах области видимости требуется доопределить калибровочные функции чуть дальше за область видимости датчика. Делается это посредством полиномиальной линейной регрессии со степенью 3. Подробную реализацию алгоритма экстраполяции можно посмотреть в функции __linear_extrapolation() в классе Calibration модуля Utils.

В результате экстраполяции получаются следующие калибровочные функции:

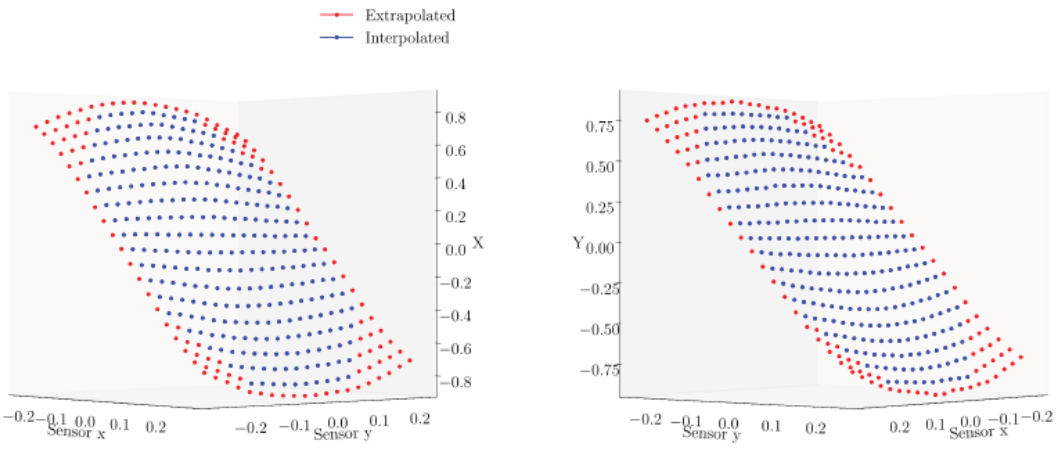

Рисунок 8 – конечные калибровочные функции датчика

После калибровки следует проверка алгоритма ориентации на стенде с полученными калибровочными данными. Алгоритм ориентации представляет из себя билинейную интерполяцию, оптимизированную под работу на микроконтроллере, и не является частью этой работы, поэтому он будет опущен. Результаты проверки алгоритма – двумерное поле абсолютной угловой ошибки, представляющей из себя зависимость угла между реальным направлением на источник света, выставленном на стенде и полученным с

помощью алгоритма ориентации, от положений датчика относительно источника света по двум сферическим углам (азимутальному и зенитному). Результаты теста представлены на рисунке 9.

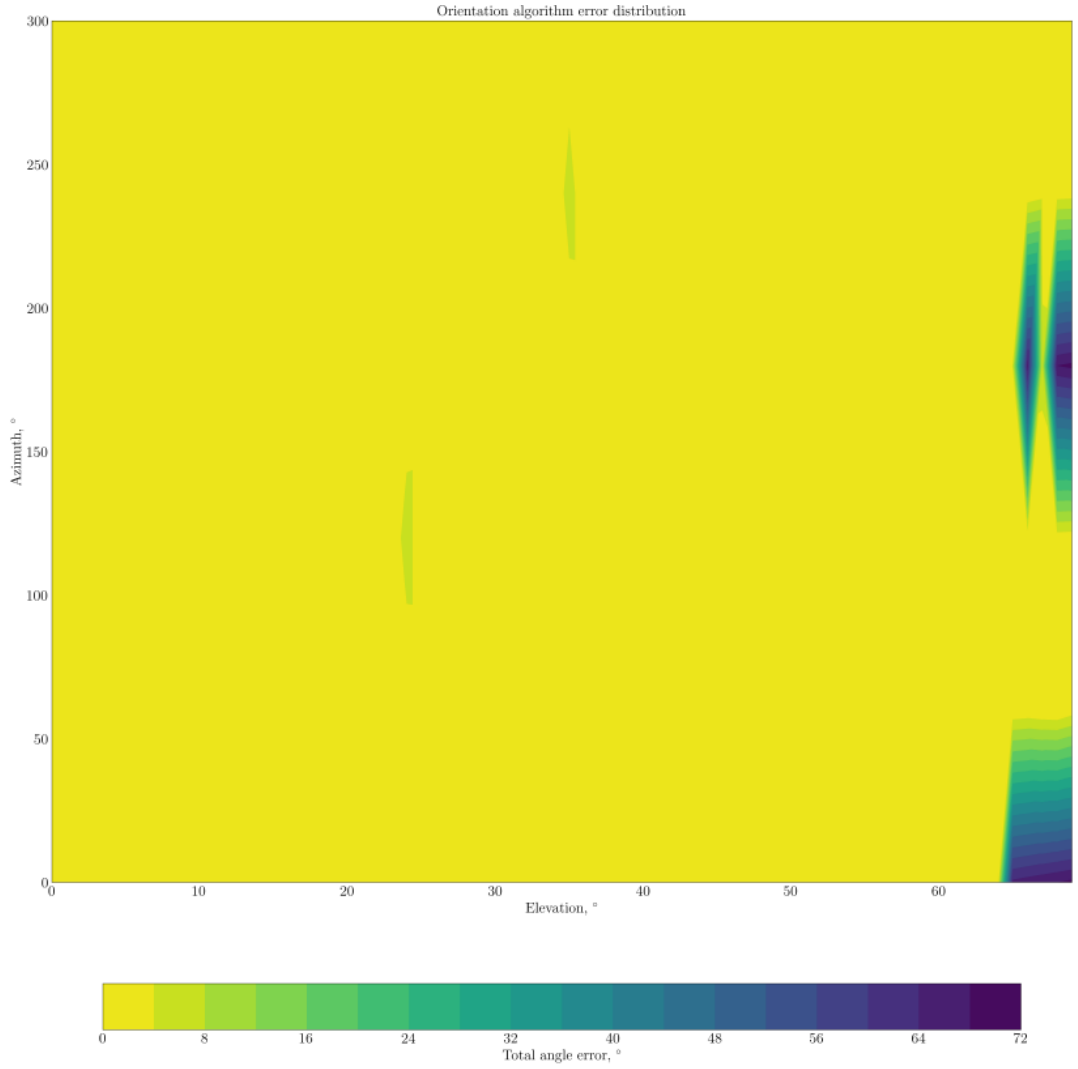

Рисунок 9 – результаты теста алгоритма ориентации на основе калибровочных данных

На основе исследования каждой панели создается отчет, в котором присутствуют все нужные графики и числовые параметры, требуемые для оценки работоспособности панели как в целом, так и в контексте ориентации по солнечному датчику. Пример отчета представлен в исходном коде проекта в директории SunSensorData.

Заключение

В ходе реализации проекта удалось наладить калибровку панелей и связать сущности, ответственные за выполнение разных задач, воедино. На данный момент процесс калибровки в достаточной степени автоматизирован и не требует каких-то серьезных временных затрат. Все части процесса калибровки выполняются последовательно и автоматически, от поиска области видимости датчика до автоматического экспорта отчета и отправки всех калибровочных данных на сервер-хранилище для последующей загрузки в память спутника.

Использование Python для работы с данными позволило существенно сократить время на разработку программного обеспечения и эффективно получить конечные калибровочные функции с использованием сторонних библиотек, таких как NumPy, SciPy, scikit-learn и scikit-image. Для обработки данных использовалась библиотека Pandas.

Список литературы

1. https://blog.satsearch.co/2020-02-12-sun-sensors-an-overview-of-systems-available-on-the-global-marketplace-for-space
2. https://elib.dlr.de/194520/1/space.0024.pdf
3. https://www.keldysh.ru/microsatellites/Bachelor_Thesis_Grigorov.pdf
4. https://nauchkor.ru/uploads/documents/60c1cfd3e4dde500012f138f.pdf
5. https://keldysh.ru/papers/2005/prep94/prep2005_94.html
6. https://pandas.pydata.org/
7. https://numpy.org/
8. https://scipy.org/
9. https://www.researchgate.net/publication/261243799_SENSOSOL_MultiFOV_4-Quadrant_high_precision_sun_sensor_for_satellite_attitude_control