Visualizing Position and Orientation in 3D Space Using Common External Controllers

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Nicholas Miller

Fall, 2022

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Briana Morrison, Department of Computer Science

Visualizing Position and Orientation in 3D Space Using Common External Controllers

CS4991 Capstone Report, 2022

Nicholas Miller
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
ncm2kjm@virginia.edu

ABSTRACT

Currently, there are few easily accessible ways to interface with common motion control capable hardware, such as the Nintendo JoyCon, hampering the use of motion control and inertial sensing technology in small, independent projects. To make this technology available to more users, I built an openly-accessible library to interface with common motion control capable devices. The library reads the device's inertial sensors, interprets the data, and visualizes it as a position and orientation in 3D space. With this library, the user is able to receive and visualize accurate position and orientation for short time periods without recalibration. Additional work on this project could add functionality to interface with other devices and improve length of effective use-time with automatic error correction and calibration.

1 INTRODUCTION

Inertial motion-sensing technology, and the systems that use it, have been around for years with a multitude of varied applications. Many modern devices, such as GPS systems, vehicles, smartphones, and game controllers, come with and use inertial motion-sensing hardware. When you pick up your smartphone and it automatically wakes up, that is an example of motion sensors at work. Yet, despite the relatively easy access to motion-capable devices, the use of motion controls in independent projects is remarkably inaccessible.

The hardware that enables this motion-sensing technology is a sensor called an inertial measurement unit (IMU). Obtaining and using an IMU by itself, while possible, requires a prohibitive amount of knowledge about hardware for the vast majority of people. The obvious choice then becomes to use one of the many motion-capable devices we have access to in our daily lives. However, the systems these devices use to interpret the readings from their sensors effectively are usually proprietary, impeding their accessibility.

To make motion control technology more commonly accessible, I designed an open system to interpret and display IMU sensor data. The device of choice for this particular project is the Nintendo JoyCon gaming controller.

2 RELATED WORKS

The work done in my project was informed by, and in some cases directly utilized, multiple other sources and previous work relating to inertial measurements. A book by Kok, et al. (2017) provides foundational knowledge about interpreting inertial data as well as a report on newer techniques [1]. I was able to implement some of the signal processing and filtering techniques described in the text, such as an extended Kalman filter.

Next, a code library created by another open-source developer, Smart (2022), served as a basis for my project. This library, called the JoyShockLibrary, is used to interface with both the Nintendo JoyCon and Playstation Dualshock controllers. It reads their raw IMU data in real time and was designed for the purpose of game development [2]. My project employs this library to serve the connection and reading functions for the external JoyCon. To this functionality, my project adds processing to clean the raw data and convert it to position and orientation as well as the visualization component. I also seek to expand the target audience beyond game development to other areas.

Last, Skog and Handel (2006) discuss various calibration techniques using a statistical approach [3]. I implement some of the basics of the calibration techniques they describe. Additionally, some of the more advanced techniques can be considered for potential improvements on the project.

3 PROJECT DESIGN

The project was designed to be modular to support future growth and can be divided into three major components: reading the raw data, cleaning and interpreting that data, and visualization.

3.1 Reading Raw IMU Data

This component's purpose was to interface with the external device, in this case a Nintendo JoyCon, and read raw data from the onboard IMU. The primary functionality for this is done by the JoyShockLibrary library, which connects to the joycon over Bluetooth and continuously polls the IMU for data every 15ms [2]. Notably, the onboard IMU for a joycon reports a new measurement every 5ms, so the slower polling rate of the library results in slightly decreased accuracy [2]. However, I decided that avoiding having to implement my own system to poll the external device was worth the slight loss in accuracy.

3.2 Cleaning and Interpretation

The raw data received from the previous component (3.1) is noisy and prone to error, as is true for any sensor measuring the real world. Additionally, the raw measurements are initially represented as acceleration and angular velocity, whereas we are looking to determine position and orientation. Thus, this component is tasked with cleaning the noisy data and calculating the position and orientation from the cleaned measurements.

The first portion of my data cleaning algorithm is a low-pass filter smoothing step. This step filters out higher frequency signals in the data and smooths variation, reducing the effect of noise which is typically higher frequency than the true data. To implement the smoothing step, I used a 3-wide moving average filter which averages the last three recorded datapoints to obtain the reported value at each poll. I opted for a moving average filter, as opposed to a more advanced low-pass filter, because it is very simple and computationally cheap. The IMU data is not a heavily frequency-based signal so more advanced methods are not necessary, given their cost; and a moving average filter serves well to counteract noise.

The next stage of cleaning the data is an extended Kalman filter as described by Kok, et al. (2017). At its core, this filter involves a nonlinear state space model characterized by a time update and a measurement update in a cycle. The time update is defined by a differentiable model that predicts what the next state of the system will be after a span of time. Then, when a measurement from the sensor is taken at the measurement update, the prediction is corrected and the model updated to be more accurate. The specific prediction models used for IMU readings involve a great deal of math and are

detailed in Kok, et al's book [1]. This predict-update cycle results in a model that quickly converges to accurately represent the true representation and gives clean data. A graph of some of the cleaned acceleration data by each axis can be seen in Figure 1, demonstrating a quick jerk of the controller followed by smaller shaking movement.

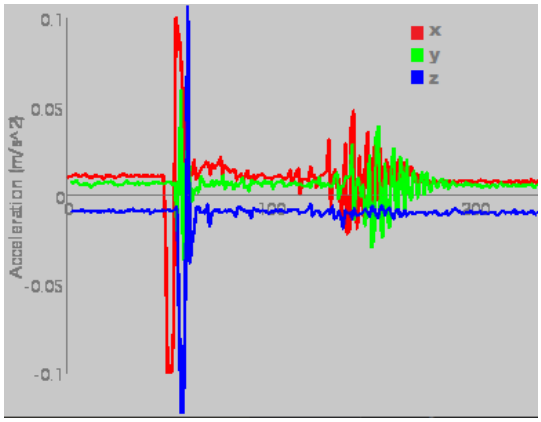

Figure 1: Cleaned Acceleration Data Graph

The choice to use the extended Kalman filter instead of one of the other, more advanced options described by Kok, et al. was again motivated by simplicity and computational complexity. The Kalman filter works well and is considerably cheaper to compute than the other proposed methods.

With the cleaned data, the last step of this component is to convert the acceleration and angular velocity readings into position and orientation. For the orientation, this was accomplished by converting the gyroscopic readings to a unit quaternion format. The unit quaternion represents a rotation in 3D space, which is applied to the current orientation state using quaternion multiplication. For the position, this was slightly more complicated as I had to account for the effect of gravity that causes unwanted readings of the accelerometer. To separate a gravity vector, I used the computed orientation to determine which direction the device is oriented relative to the downward gravity vector. Then, I removed the equivalent gravity vector with known constant magnitude from the acceleration readings. Using these adjusted acceleration readings, performing a numeric double integral calculation granted a representation of the position.

3.3 Visualization

The final component is the visualization step. This component takes the position and orientation state computed in the previous component (3.2) and visualizes it in a useful manner. The key challenge with this portion of the project was effectively conveying position and orientation in 3D space on a 2D screen. To aid in this task, I utilized the processing animation library which provides a window and the ability to draw shapes [4]. Similarly to my use of the JoyShockLibrary, I chose to use the processing library to avoid having to build my own graphics backbone. I then built a framework around this library to convert position and orientation coordinates to 3D shapes, which were then projected onto the 2D processing window. The results of this can be seen in Figure 2.

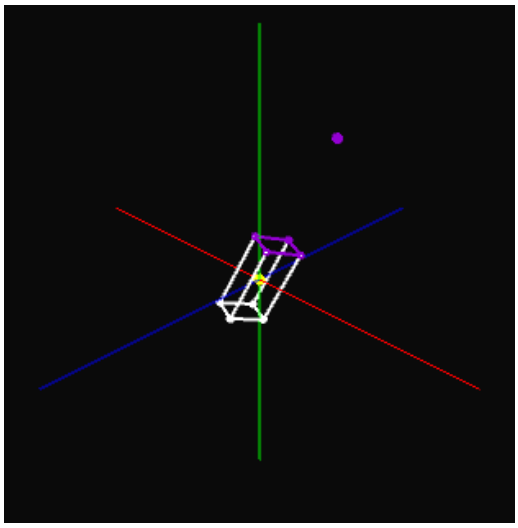
Figure 2: Visualized Device in 3D

To make the depiction clearer, I used axis lines to help orient the user's perception. In addition, the colored end of the device is used to indicate the front and

the floating purple dot helps to combat competing perspectives from certain angles. There is also a yellow dot showing the center of the device and the wireframe structure, rather than a solid shape, was chosen to make the edge lines more contrasted. These features were all intended to provide a clear and unambiguous representation of the device in 3D space.

4 RESULTS

The result of this project was a system that can read inertial data off a Nintendo JoyCon controller and accurately display that data as position and orientation in 3D space. However, due to a lack of automatic recalibration, systematic bias did develop over time if the system is used for too long without manual recalibration. Additionally, the error differed between the position and orientation. For the orientation measure, empirical estimates show that the error was maintained under 5% for the first 30 seconds of continuous operation and 15% for the first 90 seconds. The position demonstrated a higher rate of decay of accuracy, estimating that the error reached 10% in the first 30 seconds and 25% by 90 seconds. These statistics indicate a need for continuous recalibration methods to increase effective time of use.

Referring to the projects original goal of providing greater independent access to this technology, the complete system is publicly available on GitHub. Unfortunately, no statistics are available for the use of the system by others. However, I have reused the system I built in this project for other independent projects of my own. Thus, the system is completely publicly available and potentially useful.

5 CONCLUSION

The library described throughout this paper was designed to use commonly available inertial motion-sensing devices to visualize position and orientation in 3D space. The library provides functionality across the whole pipeline of converting raw inertial data to practical visualizations to maximize its ease of use. Importantly, however, the library is also modular in its construction allowing for easy improvement and customization to the user's needs and expanding its applicability in various situations. Through these vectors, this project seeks to increase the accessibility of inertial motion-sensing technology and enhance its use in independent projects, which often serve as the starting point for new and revolutionary ideas.

6 FUTURE WORK

To further improve on this project, there are three primary avenues of work to be considered. The first would be to enhance the cleaning and interpretation component to deal with systematic and time-increasing bias. Notably, the system currently struggles with a tendency to drift significantly from the expected output after some time in use at which point it requires recalibration. Potential remedies to this include implementing automatic recalibration techniques and sensor fusion by utilizing other sensors on the device.

The second future effort that could be made would be to extend the list of usable devices. Currently, the library has only been developed to interface with the Nintendo JoyCon controller. Augmenting the data reading component to add support for more common devices, such as other game controllers or mobile phones, would make the library more accessible and align with the project's goals.

Last, the third objective that might be accomplished by future work would be an expansion to the visualization component. Original plans for the project included a more robust visualization library that could display the device's location in a simulated

3D world. More ambitious designs included the ability to map physical objects in the real world into this simulated 3D world. Additional work on the project could seek to implement these designs in some manner.

REFERENCES

[1] Kok, M., Hol, J. D. and Schön, T. B. 2017. Using Inertial Sensors for Position and Orientation Estimation. *Foundations and Trends in Signal Processing* 11, 1-2 (2017), 1–153. DOI:http://dx.doi.org/10.1561/2000000094

[2] Smart, J. 2022. JoyShockLibrary. (April 2002). Retrieved from https://github.com/JibbSmart/JoyShockLibrary

[3] Skog, I. and Händel, P. 2006. Calibration of A MEMS inertial measurement unit. *XVII IMEKO World Congress Metrology for a Sustainable Development* (January 2006).