Jonathan Andrei (40051683)

Motion Capture Hand

1.0 Introduction:

The goal of this project was to build an affordable motion capture device for a hand, including fingers to experiment with motion capture techniques and have some fun. An inertial sensor unit was used to track the motion of the hand and flex sensors were used for each finger of the hand.

2.0 What was done:

The arduino code was used to read the data off of the sensor with the help of the Adafruit library and any preprocessing/filtering of the data was done at this stage. Then it would print the desired data to the serial monitor in order for the Blender application to read the values, which it would then retarget to the appropriate bones on the armature.

2.1 Roll, Pitch & Yaw:

Starting with the inertial sensor unit, I first tried to estimate roll and pitch. I estimated the roll by taking the angle in the Euclidean plane of the raw accelerometer data on the x-axis and z-axis and similarly for pitch, but for the y-axis and z-axis. Acceleration data is represented in meters per second squared, which I also converted to gravity (9.8m/s^2 = 1g). I then projected it in a processing environment on a basic rectangle object for visualization. I noticed that using the acceleration data was sensitive to vibrations.

I then tried to add a filter to the accelerometer data, where only a small percentage of the current roll being calculated would combine with a high percentage of the corrected value (the previous filtered value). I noticed that this helped reduce the noise, but it also had a slower response time.

I also estimated roll and pitch with only the gyroscope and saw that the response time was fast; however, there were drifting values when no movement of the sensor was occurring. In order to get a roll and a pitch that was not sensitive to noise and had very little drifting values, I combined both the acceleration and gyroscope to calculate the roll and pitch, while filtering at the same time.

I had difficulty calculating yaw with the acceleration data because the sensor is always perpendicular to the force of gravity. Therefore, I used the magnetometer data within the sensor in order to estimate yaw. However, the result was shaky and I am working on a way to get it smoother.

2.2 Quaternions:

For the quaternion data, I first attempted to estimate it with the previously calculated roll, pitch and yaw values that were computed. However, after further investigation through the sensor's datasheet, I found that it actually calculated quaternion data already and I found a way to extract it from the sensor. I also added a way to toggle between quaternion and euler data by modifying the rotation type to the appropriate enum in the setup function.

2.3 Flex Sensors:

The flex sensor was mainly trial and error to get it to work properly. It measures the amount of resistance applied to it. Despite being the same type of flex sensor, the values varied. I would first check the value of it when it is not bent. For example, the index finger measured approximately a value of 480 opened and a value of 250 at 90 degrees. I then mapped those values to a value between 0 to 90 degrees to make them more sensible. This inturn essentially maps the value to an appropriate angle that will be useful when retargeting that value to the finger bone.

2.4 Blender:

After the arduino code outputs the data to the appropriate com port. Using python within the blender 3D software, I used serial to read back the data, so that I can use the data to retarget it to the appropriate bone. I created a class called ImuDataManipulator, which takes in the com port, the baud rate (matches the same as the one that arduino outputs to). I also included the scene_mode to choose which scene to display the real time motion capture. For the purposes of this project, I decided to use POSE mode mainly for displaying the output. It also takes in the rot mode, which in this case it will work with quaternion mode. As well as the name of the armature that contains all the bones. It also takes in the name of the bone to apply the IMU data to and a list that contains the name of the fingers that the flex sensor data will be applied to. The readSerial method reads the serial data that the arduino C code processes. The setMode function sets the mode to POSE mode and the rotation mode to quaternion mode. The setBone method grabs the corresponding armature and the target bones to apply the data to. The retargetQuaternionData takes in four quaternions as parameters. And applies them to the w, x, y, z indices for the hand bone. Similarly, the retargetFingerBones takes in the flex data angles of the bendiness and divides it by -100, which is an arbitrary number I chose that I find works for downscaling it to fit in a unit quaternion form, which then applies that data to the y-quaternion index of the appropriate finger bone, as the fingers are aligned of this axis for going up and down. The readandTargetQuaternionData method processes the data that was read from the port and splits it separated by commas, which then takes the appropriate index of the packet and places it in the appropriate methods (retargetQuaternionData and retargetFingerBones). Note that bpy.ops.wm.redraw_timer method is a blender method that allows for the render window to update live as the sensor is moving.

The operator_hand.py file is creating an addon for the executable code. The execute function creates a class instance of the ImuDataManipulator and the bl_label titles the addon to Activate Hand, so that way the user can search for it in the view, which runs all the code to process the hand and fingers based on the sensors movements in real time. Other helper blender functions are used to make this work, such as register, which registers the addon and menu_func which makes it display in the blender menu.

3.0 What was not done:

The flex sensors were not attached to the latex glove. They did work, but were dependent on staying on the breadboard. I did try to solder the wires with a resistor to the flex sensor and then tape it on the glove. However, when I attempted to solder, the end tips broke off. I plan on re-trying after the presentation, as I wanted to make sure I had something at least working to display, instead of more broken sensors. In addition, each flex sensor only controls one bone, where I am working on a way to get it working with 3 bones for each joint of the finger for more realism.

Moreover, in the proposal I also mentioned that if I had time I would implement collisions. I did not have the chance to attempt it. To make up for it, I tried to add a 2nd inertial sensor to the elbow, as I managed to get the first one working. I had to purchase another piece of hardware called a multiplexer in order to connect the second sensor, as the arduino nano only accepts one sensor using the I2C protocol at a time, which is what the IMU uses to communicate. Unfortunately, after connecting it to the second IMU sensor, it got fried along with the arduino nano. I replaced the necessary parts and hustled to get things working again from what I had before.

4.0 Conclusion:

Despite many failures along the way, I am proud that I was able to make it this far and I plan to continue the project even when the class is finished, as it gave me a better understanding of the complexity that goes into motion capture.

References:

[1] Madgwick, S., 2010. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. [ebook] samba.org. Available at: <https://www.samba.org/tridge/UAV/madgwick_internal_report.pdf> [Accessed 5 December 2021].

[2] Youtube.com. Paul McWhorter Youtube Channel. [online] Available at: <https://www.youtube.com/user/mcwhorpj> [Accessed 5 December 2021].

[3] Adafruit BNO055 Absolute Orientation Sensor. Adafruit. [online] Available at: <https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor> [Accessed 5 December 2021].

[4] 2021. Blender 3.0.0 Python API Documentation. Blender. [online] Available at: <https://docs.blender.org/api/current/index.html> [Accessed 5 December 2021].

[5] Conversion between quaternions and Euler angles. 2021. [ebook] Wikipedia. Available at: <https://en.wikipedia.org/wiki/Conversion_between_quaternions_and_Euler_angles> [Accessed 5 December 2021].

[6] An, L., Wang, L., Liu, N., Fu, J. and Zhong, Y., 2019. A Novel Method for Estimating Pitch and Yaw of Rotating Projectiles Based on Dynamic Constraints. ncbi.nlm.nih.gov. Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6928913/> [Accessed 5 December 2021].

[7] Campbell, S., 2021. Basics of the I2C Communication Protocol. [online] Circuit Basics. Available at: <https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol> [Accessed 5 December 2021].

[8] Processing Reference. 2021. [online] Available at: <https://processing.org/reference/> [Accessed 5 December 2021].

[9] Wikipedia. 2021. Quaternion. [online] Available at: <https://en.wikipedia.org/wiki/Quaternion> [Accessed 5 December 2021].