# Motion Capture Using MPU6050

Principles and Implementation of Inertion Motion Capture

## Abstract

Inertial sensors can measure acceleration and angular velocity and represent them equivalently in a 3D coordinate system. In this article, we measure data actually using a small computer, a Raspberry Pi, and a 6DoF sensor, the MPU6050, and implement real-time motion capture using OpenGL.

**Author** Lee Hansu (lhs1438@gmail.com)

## 1. Introduction

The era of individual media creation has arrived. YouTube, the world's largest video platform, has more than 2 billion monthly visitors. Twitch, the world's largest internet broadcasting platform, has more than 17.5 million daily visitors.(YouTube Sep 5, 2020)(Twitch Sep 5, 2020) These platforms have impacted the emergence of a VTuber market, initiated by Kizuna Ai. The size of gaming and e-sports market is also increasing.(IT Chosun, Sep 5, 2020)(Aju Business Daily, Sep 5, 2020) Due to the impact of COVID-19, it is predicted that the importance of "Untact(contactless)" will increase further.(MK Sep 5, 2020) For example, recently, Travis Scott's virtual concert held in an online game 'Fortnite' has made in the headlines.(HYPEBEAST Sep 6, 2020)

There is a common technology used from VTubers, games, virtual concerts, movies, animation, sports, and more - Motion capture. Motion capture uses sensors to record human movement

to bring life to digital character modeling. This article introduces the principles of inertial system, measure data using Raspberry Pi and MPU6050 - a 6DoF(degrees of freedom) sensor, and implement real-time motion capture using OpenGL.

## 2. Background Theory

Motion capture (commonly known as mocap) is a method of creating 3D animation by measuring the actual movement of a person or other object using sensors. Compared to key frame animation and simulation, it is used in a variety of fields because it requires less effort and can achieve more realistic movements, sometimes even interacting in real time, such as in VR.(Wikipedia Sep 7, 2020)

The concepts of skeleton and joint are necessary to dataize human movement using motion capture. Each joint of the body is represented by a joint node, and the connection between them is called a skeleton.(Chulhee Jung, Myungwon Lee, 2008, 37) For each joint you want to capture, you can attach sensors or markers to obtain data, or you can capture only the core part and interpolate the rest. (Jongmin Kim, Jaehee Lee 2011, 150).

Inertial motion capture is a motion capture method that uses inertial sensors (gyroscopes, geomagnetic sensors, accelerometers). When one moves, his acceleration, angular velocity, and geomagnetic data are measured by the sensors, yielding the distance traveled and the angle of rotation (roll, pitch, and yaw). These data enable one's movement to be animated.

Roll, pitch, and yaw are the angles of rotation of a rigid body about the x, y, and z axes, respectively. They are represented using Euler angles or quaternions.

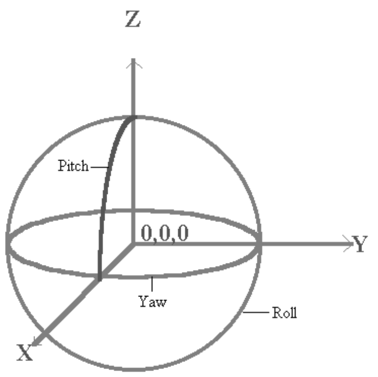


**Figure 1 – Roll, Pitch, and Yaw (ResearchGate, Sep 14, 2020)**

To calculate the Euler angles, we first measure the angle of rotation around the z-axis, and then measure the angle of rotation back along the axis we rotated.

The measured angles are called ψ, θ, and ϕ, where ψ and ϕ are up to 2π(rad) and θ is between -π/2 and π/2. The range of θ decreases as the angle of rotation is measured along the rotated axis, a phenomenon known as gimbal lock.(Wikipedia, Sep 6, 2020) Euler angles are an instrinsic way to rotate a rotated coordinate system about an axis, while roll-pitch-yaw is an extrinsic way to rotate about a fixed x, y, and z axis.(Tistory blog, Sep 14, 2020)

With an accelerometer, it's easy to find the angle of rotation using trigonometry. Theoretically, when the components of the acceleration vector of the three axes x, y and z are decomposed, respective angles can be found using inverse tangent.

$$pitch = \arctan\left(\frac{-G_x}{\sqrt{G_y^2 + G_z^2}}\right) \qquad roll = \arctan\left(\frac{G_y}{G_z}\right)$$

**Figure 2 – Calculating roll and pitch from accelerometer (ProteShea, Sep 14, 2020)**

Realistically, the yaw value cannot be found because gravitational acceleration always acts in the direction of the Earth's center. Using a gyroscope is a simpler way to do this. You can simply integrate the angular velocity in each direction over a unit of time.

$$angular\ velocity: \dot{\theta} = \frac{d\theta}{dt}$$

Take the integral of both sides to get angular position:

$$\int \dot{\theta} = \int \frac{d\theta}{dt}$$

Approximated integration

$$\theta(t) = \int_0^t \dot{\theta}(t)dt \approx \boxed{\sum_0^t \dot{\theta}(t)T_S}$$

**Figure 3 - Calculating roll, pitch, and yaw from gyroscope (ProteShea, Sep 14, 2020)**

Accelerometers tend to minimise error over time, but they run into problems when they start moving from a standstill. They also have the disadvantage of not being able to get a yaw value. Gyroscopes track the actual value fairly well, but because they use an integral to get the angle, there is a cumulative error that causes the value to drift.

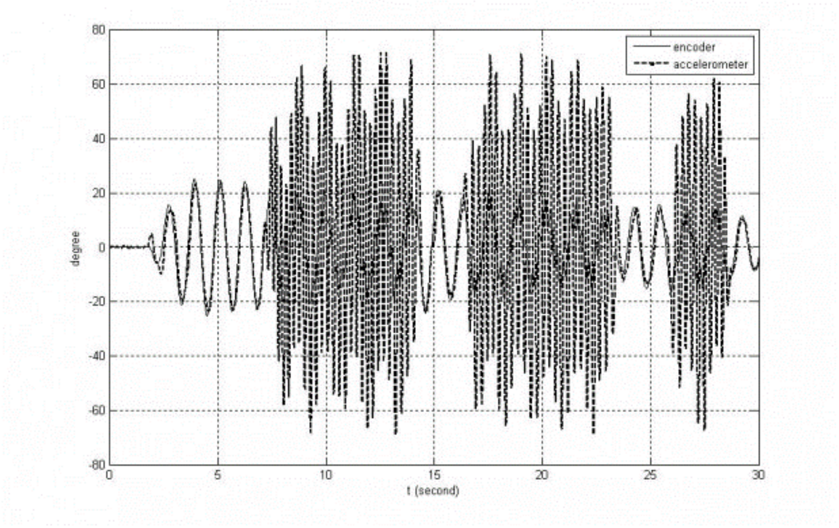


**Figure 4 – Comparison of results of measuring the swing of the pendulum by an encoder and a gyroscpoe (Media Contents, Sep 14, 2020)**

When measuring the swing of the pendulum by an encoder and a gyroscpe, We can see that integrating the results from the gyroscope gives us a fairly good representation of the change, albeit some drift in the overall angle.

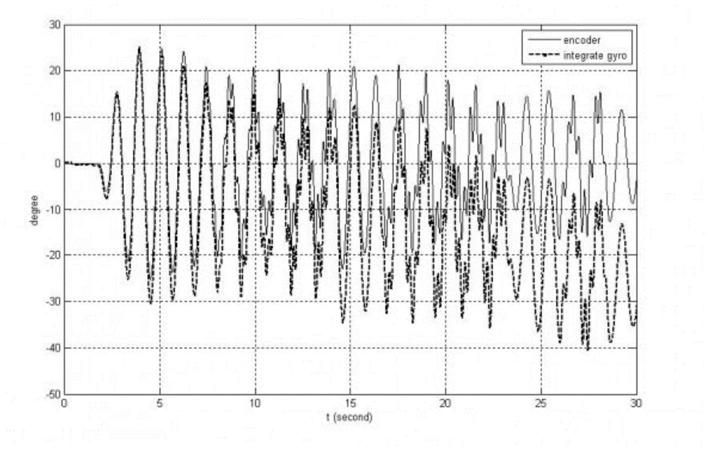


**Figure 5 – Comparison of results of measuring the swing of the pendulum by an encoder and an accelerometer (Media Contents, Sep 14, 2020)**

When measuring the swing of the pendulum by an encoder and an accelerometer, Contrary to the results using a gyroscope, the accelerometer shows no drift in angle. However, the results are unreliable when the arm moves a lot. Another way of saying this is that the pendulum itself is unreliable in places where there are large changes in angle.

Therefore, it is recommended to calibrate the measurements from both sensors to get more accurate results. The two most common methods for calibrating measurements are the complementary filter and the Kalman filter. Here we'll look at the simpler method, the complementary filter.

A complementary filter is a filter that removes the negative parts of each measurement so that they can be combined to get a more accurate value.

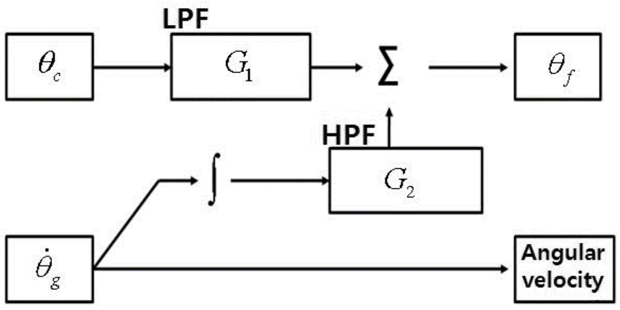


**Figure 6 – Complementary filter (Media Contents, Sep 14, 2020)**

The formula yielded by incorporating the filter is as follows:

$$angle = 0.98 * (angle + gyrData * dt) + 0.02 * (accData)$$

**Figure 7 – An example of complementary filter (MechaSolution, Sep 14, 2020)**

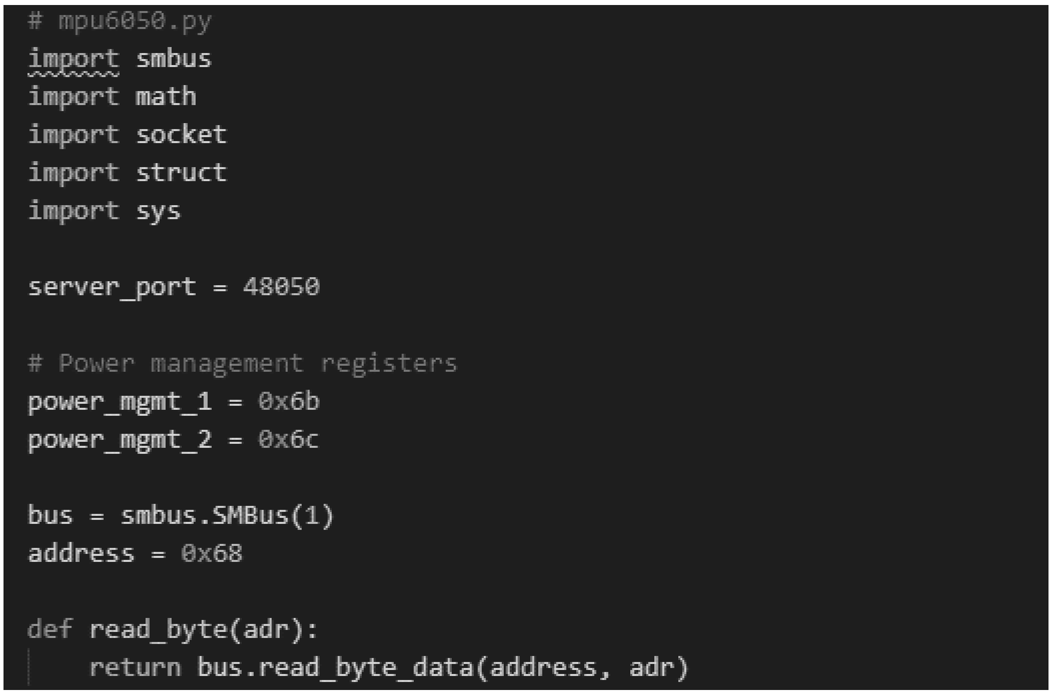
The gain values of 0.98 and 0.02 are arbitrary and can be adjusted depending on the measurements.

## 3. Implementations

The experimental setup is as follows. Connect the MPU6050 GY-521 sensor to a Raspberry Pi 3.

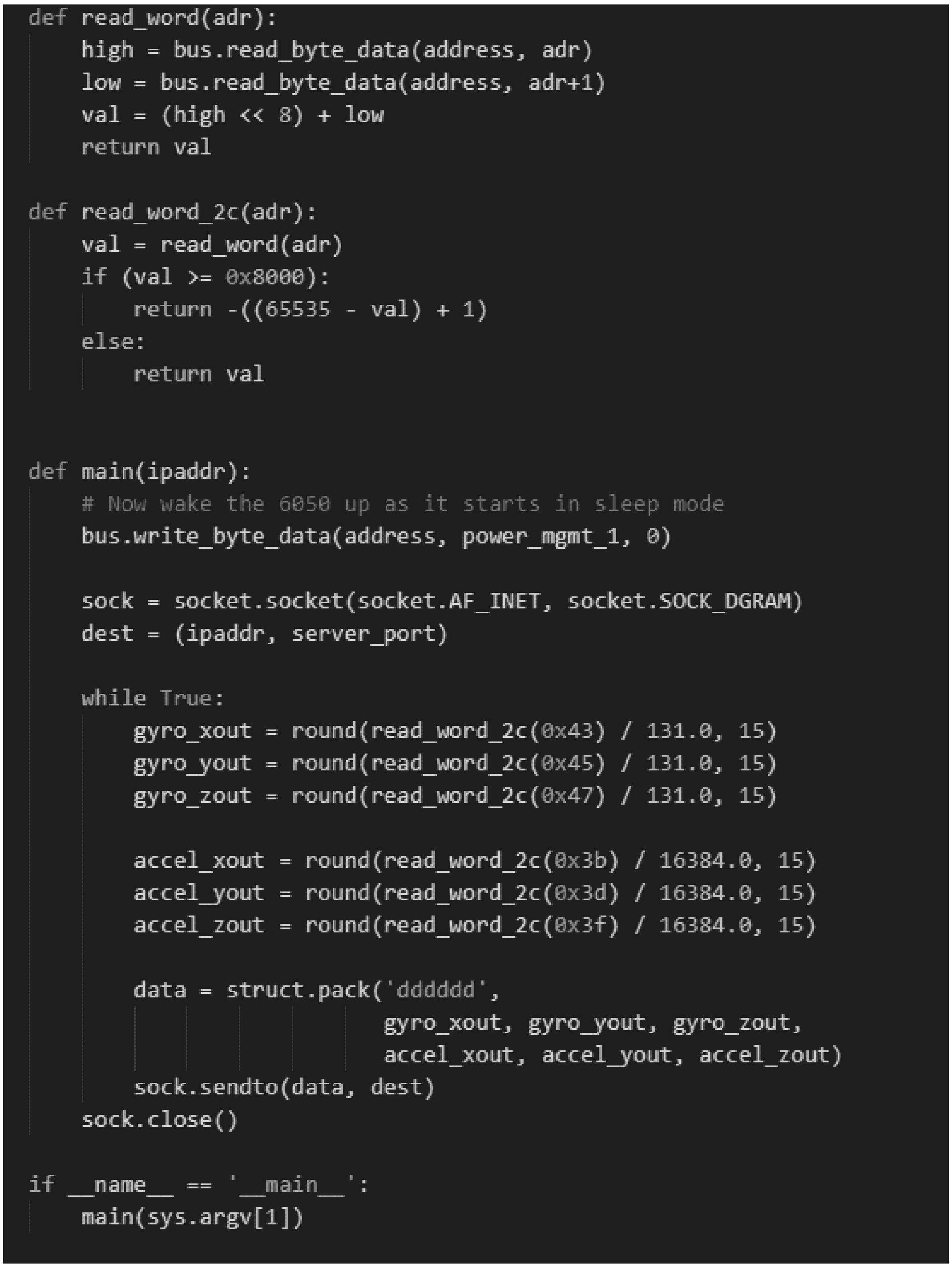
The Raspberry Pi is powered by a power bank. Get the measurement data through the sensor using I2C communication. On the Raspberry Pi, python code gets the values from the sensor and sends them to the main computer using a UDP socket. The main computer receives the data and processes it using C++ code to get the final angle of rotation. Finally, the 3D cube is rotated using OpenGL.

Below is the python program for the module that acquires the sensor data from Raspberry Pi:



```python
# mpu6050.py
import smbus
import math
import socket
import struct
import sys

server_port = 48050

# Power management registers
power_mgmt_1 = 0x6b
power_mgmt_2 = 0x6c

bus = smbus.SMBus(1)
address = 0x68

def read_byte(adr):
    return bus.read_byte_data(address, adr)
```

**Figure 8 Source code for MPU-6050 sensor data acquisition**



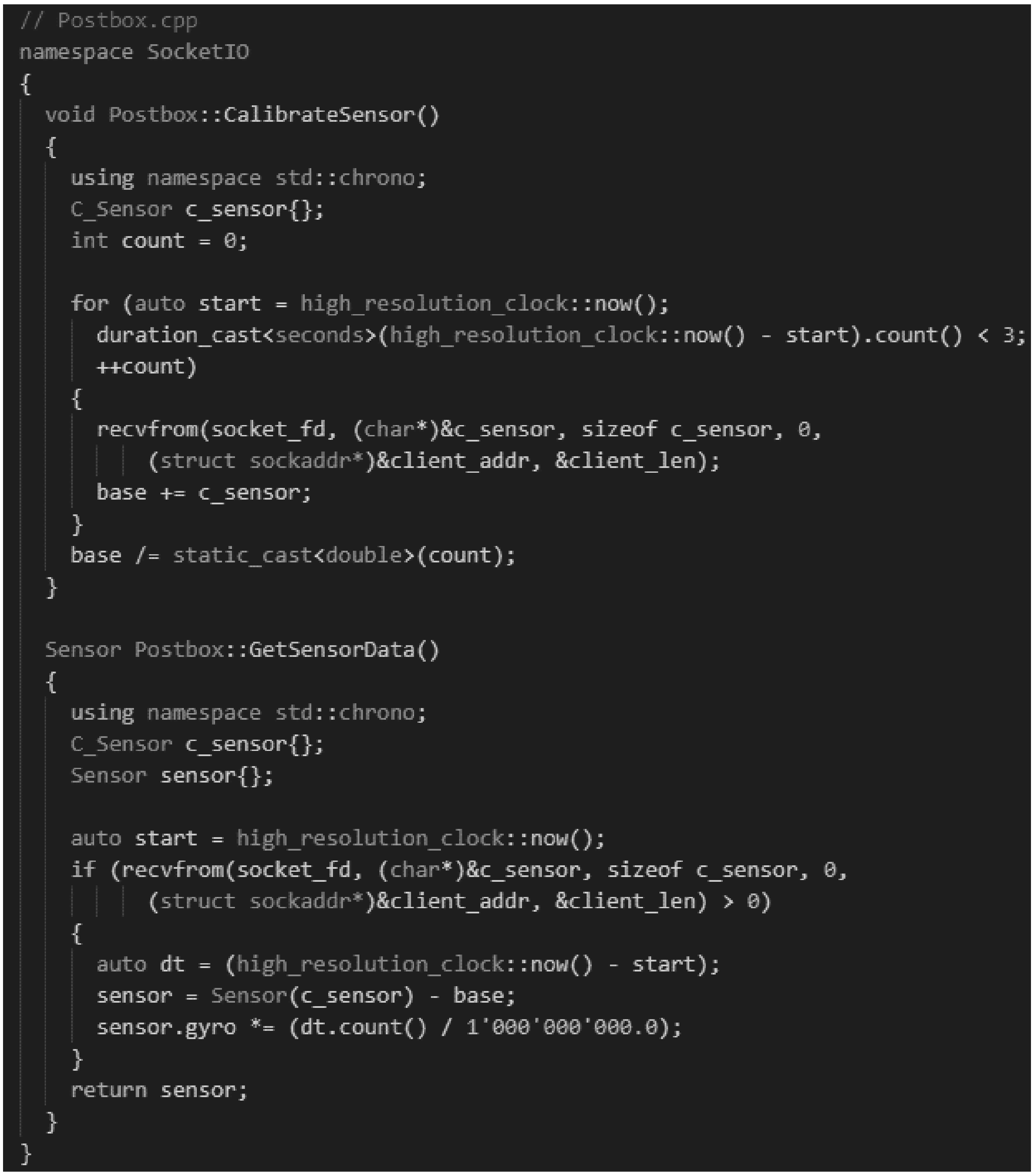
```python
def read_word(adr):
    high = bus.read_byte_data(address, adr)
    low = bus.read_byte_data(address, adr+1)
    val = (high << 8) + low
    return val

def read_word_2c(adr):
    val = read_word(adr)
    if (val >= 0x8000):
        return -((65535 - val) + 1)
    else:
        return val


def main(ipaddr):
    # Now wake the 6050 up as it starts in sleep mode
    bus.write_byte_data(address, power_mgmt_1, 0)

    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    dest = (ipaddr, server_port)

    while True:
        gyro_xout = round(read_word_2c(0x43) / 131.0, 15)
        gyro_yout = round(read_word_2c(0x45) / 131.0, 15)
        gyro_zout = round(read_word_2c(0x47) / 131.0, 15)

        accel_xout = round(read_word_2c(0x3b) / 16384.0, 15)
        accel_yout = round(read_word_2c(0x3d) / 16384.0, 15)
        accel_zout = round(read_word_2c(0x3f) / 16384.0, 15)

        data = struct.pack('dddddd',
                           gyro_xout, gyro_yout, gyro_zout,
                           accel_xout, accel_yout, accel_zout)
        sock.sendto(data, dest)
    sock.close()

if __name__ == '__main__':
    main(sys.argv[1])
```

**Figure 9 Source code for MPU-6050 sensor data acquisition**

Use the smbus module of python to communicate i2c with the MPU6050. The family of read_word() function is used since the MPU6050's gyroscope and accelerometer ADC word lengths are 16 bits. The read_word_2c() function converts the value to a 16-bit integer.

It uses an infinite loop to send the measured value to the server. 131 and 16384 are the default sensitivity scale factors for the gyroscope and the accelerometer, respectively. No changes were made from the default setting. The hexadecimal values in the read_word_2c() function are the addresses of the registers in the MPU6050 where the corresponding sensor measurements are stored. Since the C code is using a double-type to store the measurements, it is rounded to 15 decimal places. You can use the struct.pack() function to read into struct from C/C++ all at once.

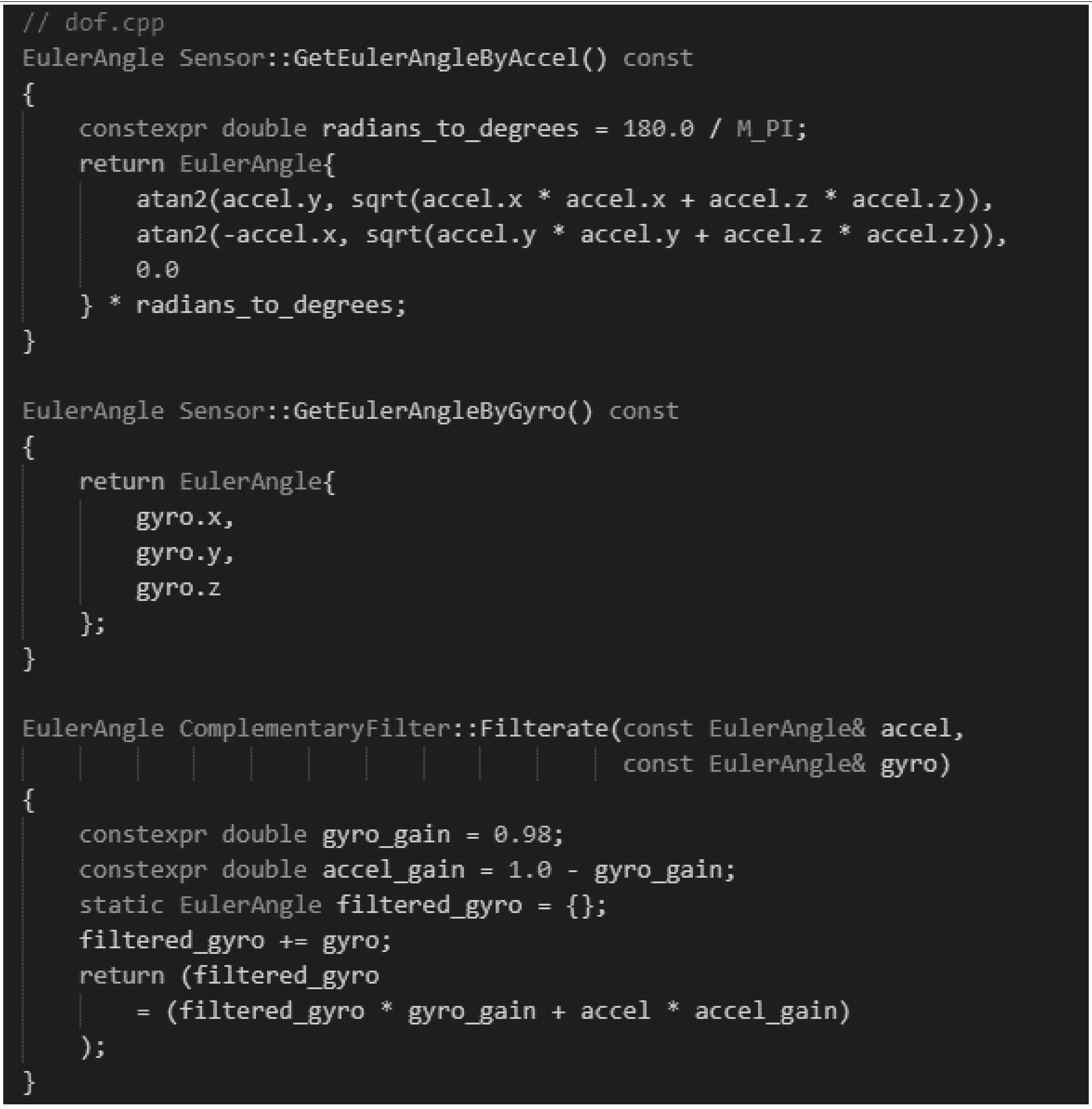
Here is the C++ code I am using for filtering and visualization on my main computer:



```cpp
// Postbox.cpp
namespace SocketIO
{
    void Postbox::CalibrateSensor()
    {
        using namespace std::chrono;
        C_Sensor c_sensor{};
        int count = 0;

        for (auto start = high_resolution_clock::now();
            duration_cast<seconds>(high_resolution_clock::now() - start).count() < 3;
            ++count)
        {
            recvfrom(socket_fd, (char*)&c_sensor, sizeof c_sensor, 0,
                    (struct sockaddr*)&client_addr, &client_len);
            base += c_sensor;
        }
        base /= static_cast<double>(count);
    }

    Sensor Postbox::GetSensorData()
    {
        using namespace std::chrono;
        C_Sensor c_sensor{};
        Sensor sensor{};

        auto start = high_resolution_clock::now();
        if (recvfrom(socket_fd, (char*)&c_sensor, sizeof c_sensor, 0,
                    (struct sockaddr*)&client_addr, &client_len) > 0)
        {
            auto dt = (high_resolution_clock::now() - start);
            sensor = Sensor(c_sensor) - base;
            sensor.gyro *= (dt.count() / 1'000'000'000.0);
        }
        return sensor;
    }
}
```

**Figure 10 Source code for receiving sensor data**

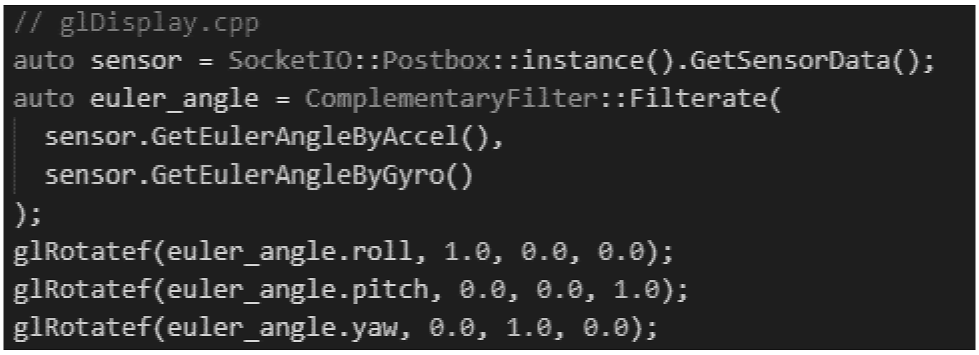
Postbox::CalibrateSensor() is a function to calibrate the initial point of the inertial sensor. The average value acquired from the first 3 seconds is determined as the initial point.



```cpp
// dof.cpp
EulerAngle Sensor::GetEulerAngleByAccel() const
{
    constexpr double radians_to_degrees = 180.0 / M_PI;
    return EulerAngle{
        atan2(accel.y, sqrt(accel.x * accel.x + accel.z * accel.z)),
        atan2(-accel.x, sqrt(accel.y * accel.y + accel.z * accel.z)),
        0.0
    } * radians_to_degrees;
}

EulerAngle Sensor::GetEulerAngleByGyro() const
{
    return EulerAngle{
        gyro.x,
        gyro.y,
        gyro.z
    };
}

EulerAngle ComplementaryFilter::Filterate(const EulerAngle& accel,
                                          const EulerAngle& gyro)
{
    constexpr double gyro_gain = 0.98;
    constexpr double accel_gain = 1.0 - gyro_gain;
    static EulerAngle filtered_gyro = {};
    filtered_gyro += gyro;
    return (filtered_gyro
        = (filtered_gyro * gyro_gain + accel * accel_gain)
    );
}
```

**Figure 11 Source code for the Euler angle transformation and the complementary filter**

The angular velocity measured by the gyroscope is integrated over unit time, and the acceleration measured by the accelerometer is converted to roll-pitch-yaw using the formula above. ComplementaryFilter::Filterate() is an implementation of a complementary filter.



```cpp
// glDisplay.cpp
auto sensor = SocketIO::Postbox::instance().GetSensorData();
auto euler_angle = ComplementaryFilter::Filterate(
  sensor.GetEulerAngleByAccel(),
  sensor.GetEulerAngleByGyro()
);
glRotatef(euler_angle.roll, 1.0, 0.0, 0.0);
glRotatef(euler_angle.pitch, 0.0, 0.0, 1.0);
glRotatef(euler_angle.yaw, 0.0, 1.0, 0.0);
```

**Figure 12 Source code for rotating 3D objects**

Take the sensor data from the Raspberry Pi and run it through a complementary filter to rotate a 3D object around each axis. Since the coordinate system of OpenGL is right-handed and the MPU6050 is left-handed, we exchange the pitch and yaw.

## 4. Results

The full source code for the implementation can be found at https://github.com/droplet92/mpu6050_mocap.

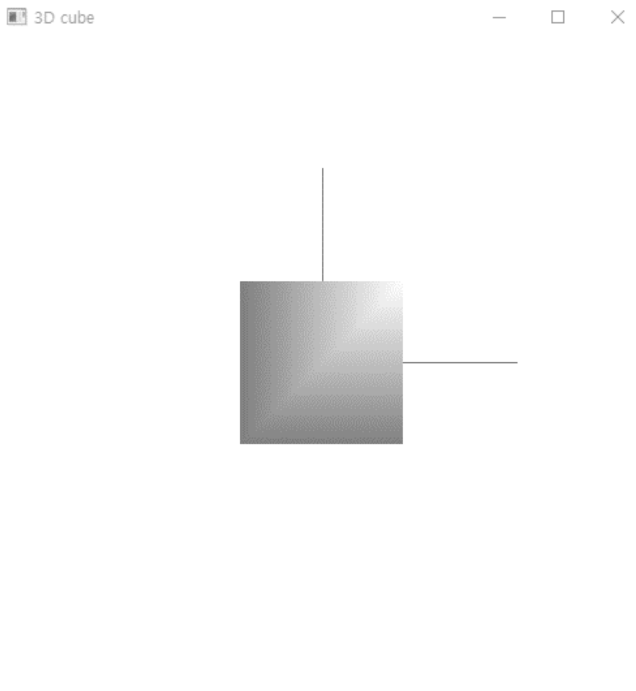


**Figure 13 – Fixed 3D cube object**

### (1) Rotated about one axis

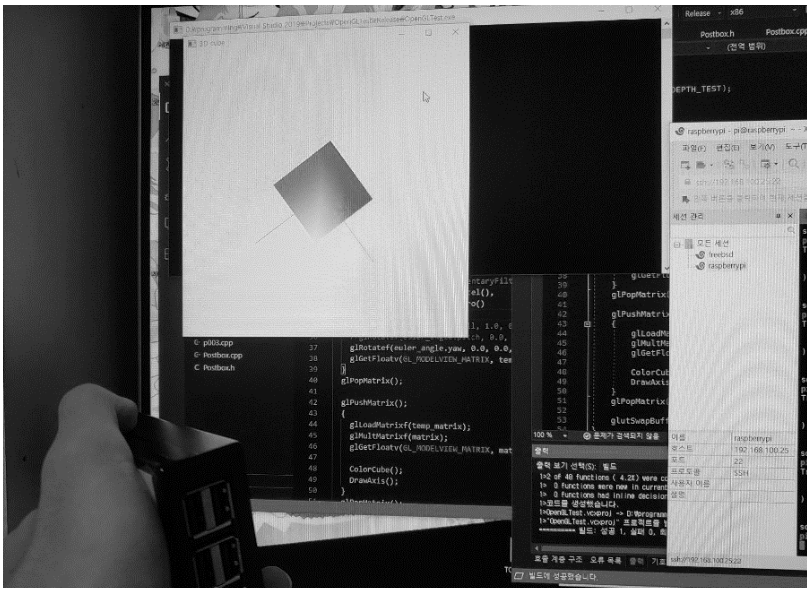


**Figure 14 – Results reflecting only yaw**

When only one of the three rotations (roll, pitch, or yaw) was reflected, the results looked satisfying. However, it would rotate slightly even when it was still, or it would rotate suddenly and rapidly due to gimbal lock.

### (2) Results reflecting all rotation data

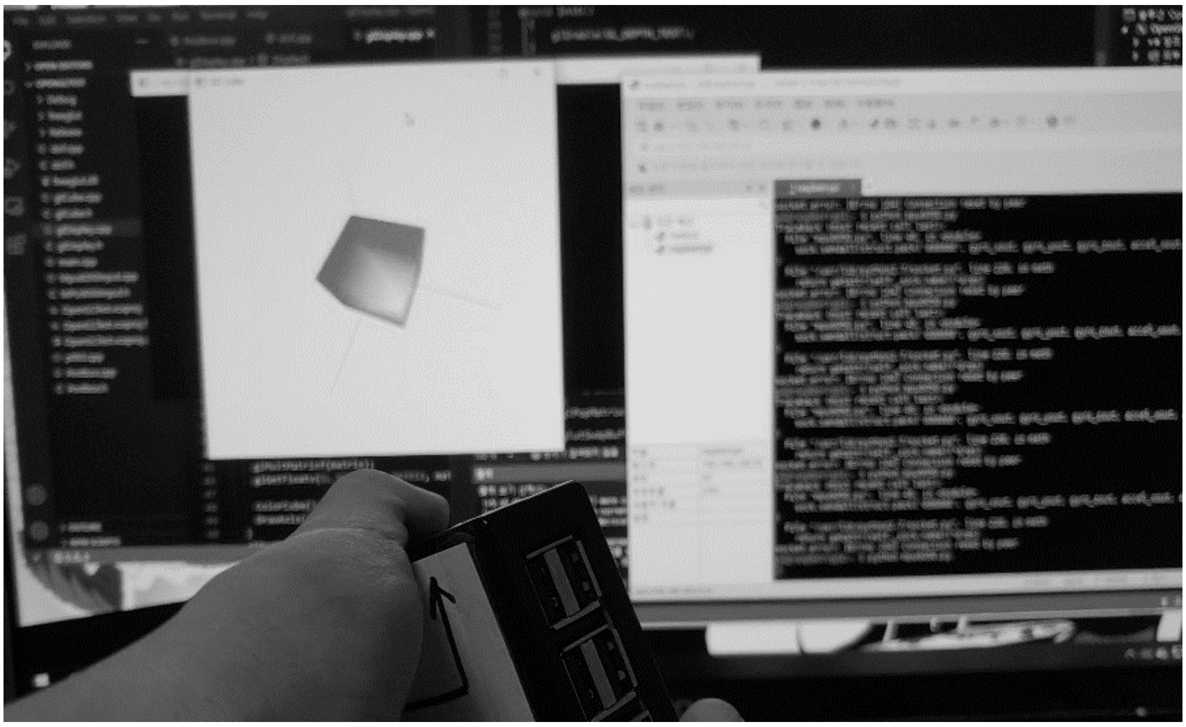


**Figure 15 – The result of including all roll, pitch, and yaw**

They tend to move together according to the movement of the sensor, but they rotate at a high speed even when still, making it difficult to see the overall picture. It seemed like there were too much instantaneous data, which resulted in a lot of noise. By artificially reducing the data from the sensors, I was able to observe better results with the naked eye.

## 5. Conclusion

We were able to understand the concept of inertial motion capture; and the principles and limitations of gyroscopes and accelerometers. Although the real-time motion capture experiment was not as successful as it could have been, we were able to see some results with the naked eye. We can improve the problem by revisiting the code and formulas for errors and applying better methods such as Kalman filters and quaternions. Other tools such as a geomagnetic sensor or a DMP may be utilised to improve the results..

## Reference

| | |
|---|---|
| Papers/Articles | ➢ 정철희, 이명원. 2008. "호환성 있는 인체 애니메이션을 위한 모션 데이터 정의." 컴퓨터그래픽스학회논문지 14 (2): 35-41. doi: 10.15701/kcgs.2008.14.2.35<br>➢ 김종민, 이제희. 2011. "적은 개수의 3 차원 모션 센서들을 이용한 실시간 동작 생성." HCI 2011: 150-152. |

Websites

- "YouTube About" (Sep 5, 2020) YouTube. 날짜 없음, https://www.youtube.com/intl/ko/about/press/
- "Twitch Press Center" (Sep 5, 2020) Twitch. last modified Sep 10, 2020, https://www.twitch.tv/p/press-center/
- "연예인 못지 않은 인기...버츄얼 유튜버 아시나요" (Sep 5, 2020) IT Chosun. last modified Sep 15, 2019, http://it.chosun.com/site/data/html_dir/2019/09/11/2019091102627.html
- "[보는 게임 시대] ③ 플랫폼 간 e 스포츠 유치 경쟁... 인기 스트리머 쟁탈전도" (Sep 5, 2020) 아주경제. last modified Apr 24, 2020, https://www.ajunews.com/view/20200424001614811
- "집콕 시대...e 스포츠는 최대 호황" (Sep 5, 2020) MK. last modified May 1, 2020, https://www.mk.co.kr/news/it/view/2020/05/451247/
- "트래비스 스콧, '포트나이트' 게임 내 가상 콘서트 영상 공개" (Sep 6, 2020) HYPEBEAST. last modified Apr 24, 2020, https://hypebeast.kr/2020/4/travis-scott-astronomical-fortnite-concert-livestream-kid-cudi-collaboration-the-scotts
- "Motion capture" (Sep 7, 2020) Wikipedia. last modified Sep 10, 2020, https://en.wikipedia.org/wiki/Motion_capture#Advantages
- "xyz and pitch, roll and yaw systems" (Sep 14, 2020) ResearchGate. 날짜 없음, https://www.researchgate.net/figure/xyz-and-pitch-roll-and-yaw-systems_fig4_253569466
- "오일러 각" (Sep 6, 2020) Wikipedia. Aug 31, 2020, https://ko.wikipedia.org/wiki/오일러_각
- "오일러 각 Pitch Roll Yaw" (Sep 14, 2020) Tistory blog gammabeta. last modified Nov 11, 2019, https://gammabeta.tistory.com/1761
- "MPU6050 Accelerometer and Gyroscope" (Sep 14, 2020) ProteShea. last modified Jul, 2019, https://proteshea.com/process/learn/embedded-systems/uno-rev3-projects/mpu6050-accelerometer-and-gyroscope-with-arduino-uno/
- "[12 호]왜 두 개의 센서를 융합하는가? 2 부" (Sep 14, 2020) Media Contents. last modified Apr 25, 2012, http://www.ntrexgo.com/archives/4000
- "상보필터" (Sep 14, 2020) MechaSolution. last modified Jul 01, 2015, http://mechasolutionwiki.com/index.php?title=상보필터
-