

Real-Time Position Tracking Using IMU Data

Author: Jermaine Smith

Faculty Mentor: Dr. Yohannes Ketema

1 Introduction

Sensor-based gait monitoring is a method of measuring and estimating characteristics in which a person walks through the use of external sensors [1]. Using data sampled from Inertial Measurement Units (IMUs), previous research has shown that a solid qualitative representation of motion could be acquired through integration of previously gathered accelerometer and angular velocity data [2].

The unique goal of this project and study was to develop and test a real-time position tracking system through the use of low-cost IMUs. The data sampled from the IMU is analyzed in real time using software which detects when steps are taken then estimates the length and direction of the stride to update XY position values.

Using a Bosch Sensortec BNO055 IMU strapped to a test subject's leg and an Arduino UNO microcontroller, multiple tests were done to quantify the accuracy of the system's ability to estimate the subject's final position after walking short distances of varying patterns.

The test results show that the system can give a reasonable quantitative XY position location for short distances with an error of about 10-15% the distance walked. This error could be reduced significantly through the use of a second IMU. Further testing is necessary for long distances and extended periods of time.

2 Design

In this section the overall design of the system (hardware and software) and tests are presented.

2.1 Hardware

This section outlines the hardware used in the position calculation system as well as schematics and setup instructions.

2.1.1 IMU

The IMU used in the position tracking system is the Bosch Sensortec BNO055 IMU, shown in **Fig. 1**.



Fig. 1 Bosch Sensortec BNO055 IMU

<https://www.amazon.com/Adafruit-Absolute-Orientation-Fusion-Breakout/dp/B017PEIGIG>

The BNO055 is an “Intelligent 9-axis absolute orientation sensor”. Using on boards sensor fusion algorithms, the sensor is capable of outputting Quaternion, Euler angles, Rotation Vector, Linear Acceleration, Gravity, and Heading at a rate of 10 hz over I2C and UART lines.[3]. The on-board fusion algorithm makes this IMU an ideal choice for this project, as it allows for an algebraic method of position calculation rather than integration based.

2.1.2 Microcontroller

The board used in the system was an Arduino UNO, shown in **Fig. 2**

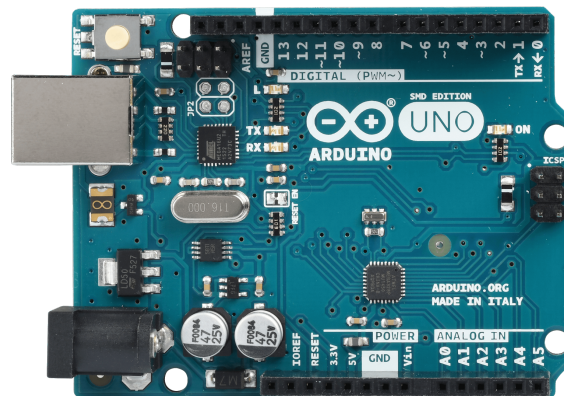


Fig. 2 Arduino UNO

<https://www.reichelt.com/de/en/arduino-uno-rev-3-atmega328-usb-arduino-uno-p119045.html>

This board operates at 5V and can be powered via a USB cable-A male to cable-B male. Thus, the Arduino UNO meets all necessary requirements for operating with the BNO055 and is also the most well documented board offered by Arduino[4].

2.1.3 Schematic & Pinouts

The following figure shows a simplified view of the system as a whole.

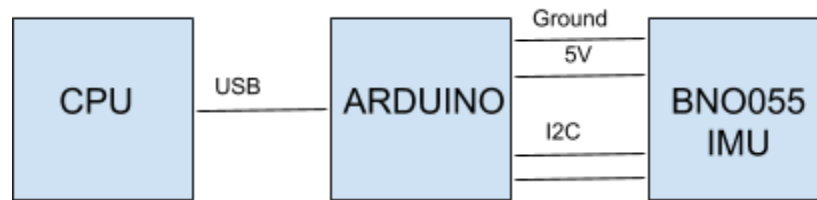


Fig. 3 Simple schematic of position calculation system

The connections between the Arduino UNO and BNO055 IMU are as follows

- Connect Vin (IMU) to 5V pin (Arduino UNO)
- Connect GND (IMU) to GND (Arduino UNO)
- Connect SDA (IMU) to A4 (Arduino UNO)
- Connect SCL (IMU) to A5 (Arduino UNO)

These connections are shown in the following Figure.

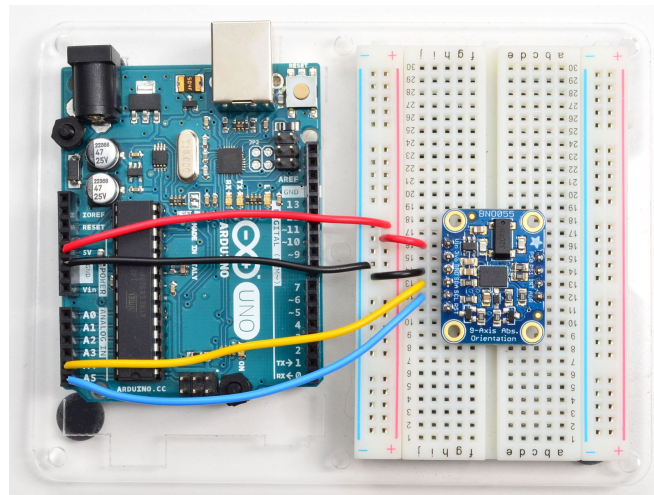


Fig. 4 Wired connections from Arduino UNO to BNO055 IMU

<https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor/arduino-code>

2.2 Software

This section will focus on highlighting and explaining the core functionality of the code used to calculate position. A complete copy of the code used is included in the **Appendix**. The program was based off of the Adafruit Unified Sensor System code (link provided in appendix).

2.2.1 Calibrating & Sampling

First, the calibration of the device is performed. This is vital to proper functionality. The following section of code, which is located in the ‘void setup’ section of the code, performs the device calibration

```
//Calibrating the IMU
uint8_t system, gyro, accel, mag = 0;

while(system != 3)
{
    bno.getCalibration(&system, &gyro, &accel, &mag);

    Serial.print("CALIBRATION: Sys=");
    Serial.print(system, DEC);
    Serial.print(" Gyro=");
    Serial.print(gyro, DEC);
    Serial.print(" Accel=");
    Serial.print(accel, DEC);
    Serial.print(" Mag=");
    Serial.println(mag, DEC);

    delay(100);
}

Serial.println(""); Serial.println("Calibrated");
```

Here, the program waits for the variable ‘system’ to equal 3. This variable equals 3 when the fusion system is ready for use.

The command

```
bno.getCalibration(&system, &gyro, &accel, &mag);
```

is necessary to retrieve the calibration status of the IMU.

The 100 ms delay in the while loop is used throughout the program as it corresponds to the 10hz sampling rate of the IMU.

Once the device is fully calibrated, the loop portion of the code begins. First the euler angles, and acceleration vector are retrieved using

```
imu::Vector<3> euler =
    bno.getVector(Adafruit_BNO055::VECTOR_EULER);
```

```
imu::Vector<3> acc =  
    bno.getVector(Adafruit_BNO055::VECTOR_ACCELEROMETER);
```

then the magnitude of the acceleration is computed using the 'acc' vector

```
acc_magnitude = (sqrt(pow(acc.x(),2) + pow(acc.y(),2) +  
    pow(acc.z(),2)))-9.81;
```

'acc_magnitude' is used in the detection of steps.

2.2.2 Step Detection

Next, the program checks to see if a step has been taken using

```
if (acc_magnitude > 0.9 & abs( euler.z() )<40 )
```

First, acc_magnitude magnitude must be greater than 0.9, this value works well for my gait, but would have to be adjusted for other users. Next, the program checks the Z Euler angle to assure the leg with the sensor is in a forward swing rather than accelerating from its resting position.

When the first step is detected the following code is executed.

```
if (calibration == 0)  
{  
    eX_offset = euler.x();  
    calibration =1;  
}
```

'eX_offset' will be used in calculating position. Specifically, it is useful for estimating the walking direction.

2.2.3 Calculating Position

Finally, the position is calculated. Three different values are calculated in this portion of the program. The values 'pos_x' and 'pos_y' are the x and y values of location in feet. The value 'pos_mag' calculates the magnitude of distance traveled in feet. The positive x axis is located in the direction in which the first step was taken and the coordinate axis is a right-handed system with the z axis pointing downwards.

The following code is used to calculate the three position values.

```

pos_x = pos_x+ 2*(2*leg_len*cos((euler.y()-10)*PI/180))*
        cos((euler.x()-eX_offset)*PI/180)*3.281; // x- position

pos_y = pos_y+ 2*(2*leg_len*cos((euler.y()-10)*PI/180))*
        sin((euler.x()-eX_offset)*PI/180)*3.281; // y-position

pos_mag = pos_mag +
        2*(2*leg_len*cos((euler.y()-10)*PI/180))*3.281;

```

2.2.3.1 Calculating Distance

To explain the math used here, the 'pos_mag' calculation will first be examined.

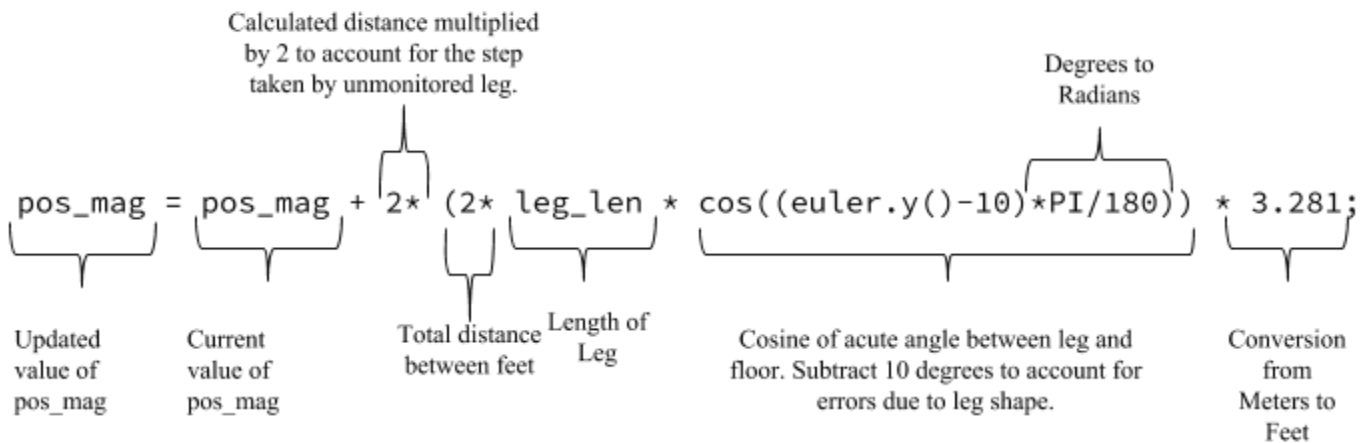


Fig. 5 Labeled explanation of the formula used to calculate distance based off of leg length and euler angles.

Note: A good method of determining the `euler.y()` offset value, which is -10 degrees here, is to measure a step and take note of `euler.y()`. Then solve for the value of `cos(theta)` which would give the proper distance calculation. From here, solve for `theta` and compare it to the value of `euler.y()` and incorporate the offset value required to make the values equal.

The following figure illustrates the calculation

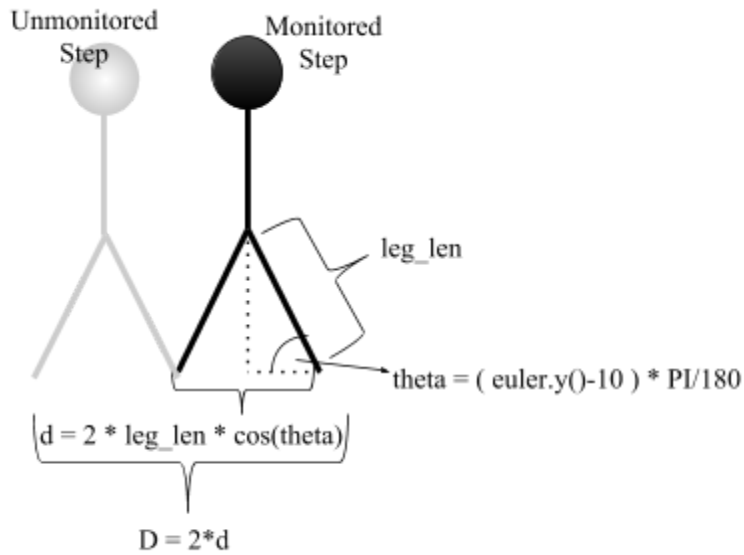


Fig. 6 A visual illustration of the calculation used for distance (Fig. 5).

2.2.3.2 Calculating Direction

The calculated distance D can be resolved into x and y components using the following factors.

$$D * \cos((euler.x() - eX_offset) * \pi / 180) \quad // \quad x\text{- position}$$

$$D * \sin((euler.x() - eX_offset) * \pi / 180) \quad // \quad y\text{-position}$$

The following diagram illustrates the calculation

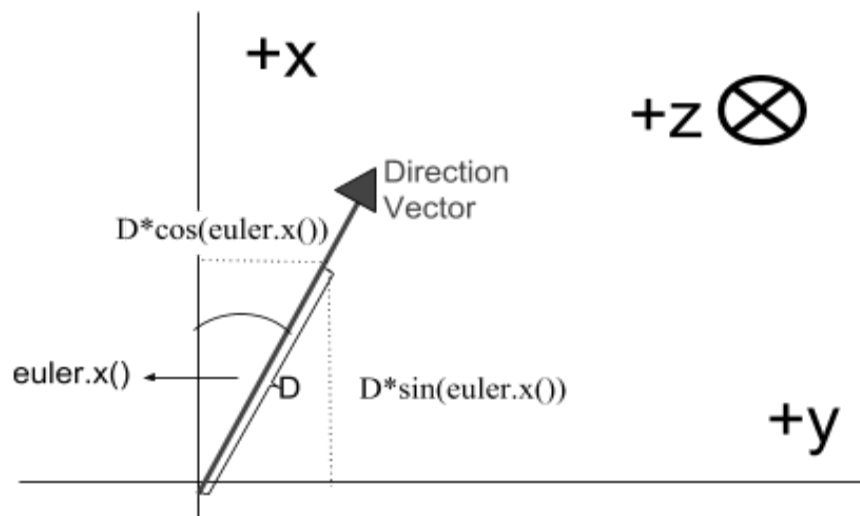


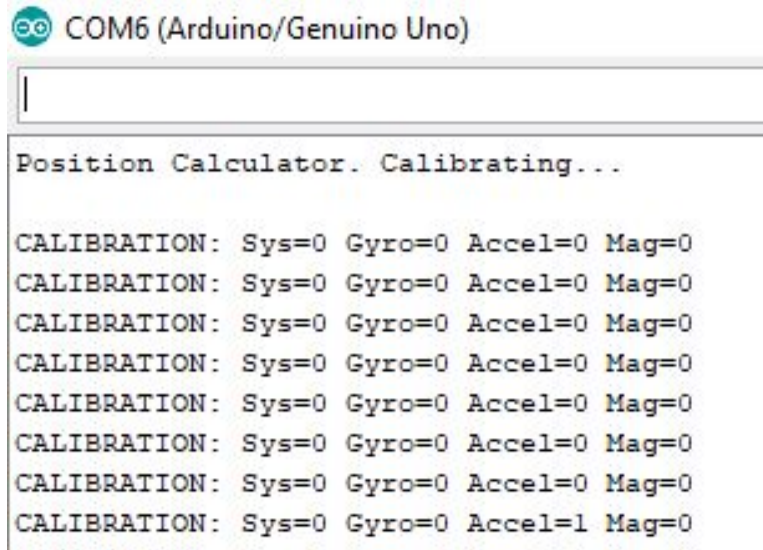
Fig. 7 A visual illustration of the calculation used for direction

Note: The `eX_offset` term causes the +x axis to always be in the direction of the first step the user takes.

2.3 Tests

To test the accuracy of the system, the breadboard containing the IMU was strapped to the right leg of the test subject. The Arduino board was strapped to a belt which could be worn. The Arduino was connected to the breadboard using the pinout shown above and then connected to a laptop via a USB male A-B cable. The Arduino code was then uploaded to the board and the serial monitor was opened.

After a few seconds the serial monitor showed the following.



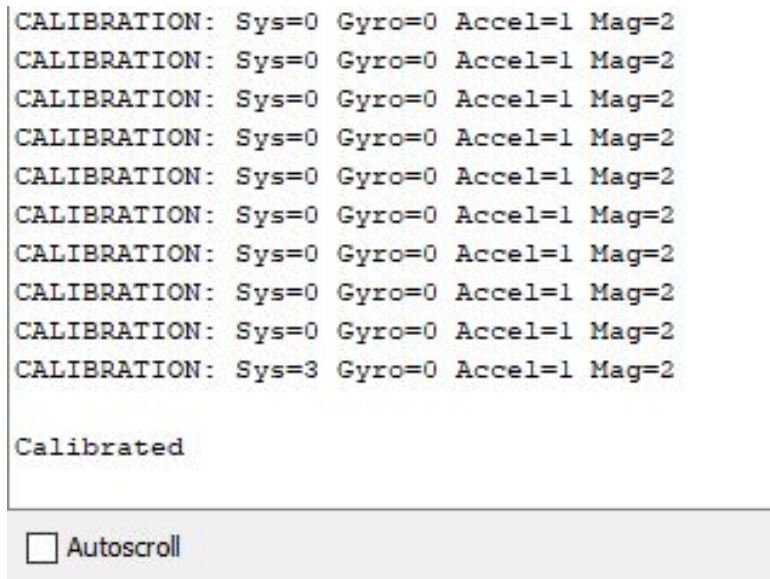
```
COM6 (Arduino/Genuino Uno)

Position Calculator. Calibrating...

CALIBRATION: Sys=0 Gyro=0 Accel=0 Mag=0
CALIBRATION: Sys=0 Gyro=0 Accel=0 Mag=0
CALIBRATION: Sys=0 Gyro=0 Accel=0 Mag=0
CALIBRATION: Sys=0 Gyro=0 Accel=0 Mag=0
CALIBRATION: Sys=0 Gyro=0 Accel=0 Mag=0
CALIBRATION: Sys=0 Gyro=0 Accel=0 Mag=0
CALIBRATION: Sys=0 Gyro=0 Accel=0 Mag=0
CALIBRATION: Sys=0 Gyro=0 Accel=1 Mag=0
```

Fig. 8 Picture of Serial Monitor once program is running. ‘CALIBRATION:...’ is printed until system has been calibrated.

Next, the test subject moved their leg in order to calibrate the sensor.



```
CALIBRATION: Sys=0 Gyro=0 Accel=1 Mag=2
CALIBRATION: Sys=0 Gyro=0 Accel=1 Mag=2
CALIBRATION: Sys=0 Gyro=0 Accel=1 Mag=2
CALIBRATION: Sys=0 Gyro=0 Accel=1 Mag=2
CALIBRATION: Sys=0 Gyro=0 Accel=1 Mag=2
CALIBRATION: Sys=0 Gyro=0 Accel=1 Mag=2
CALIBRATION: Sys=0 Gyro=0 Accel=1 Mag=2
CALIBRATION: Sys=0 Gyro=0 Accel=1 Mag=2
CALIBRATION: Sys=0 Gyro=0 Accel=1 Mag=2
CALIBRATION: Sys=0 Gyro=0 Accel=1 Mag=2
CALIBRATION: Sys=3 Gyro=0 Accel=1 Mag=2

Calibrated

 Autoscroll
```

Fig. 9 Picture of serial monitor once system has been calibrated

The most effective way to do this is by swinging the leg to the side and backwards as is demonstrated in **Figures 10 and 11**.



Fig. 10 Swinging monitored leg to side to calibrate system.



Fig. 11 Swinging monitored leg backwards to calibrate system.

3 Data & Analysis

In this section the results of the tests are presented and analyzed.

3.1 Distance Test

The results of thirteen trials of the distance calculation test show an average error of 2.13 ft. with a standard deviation of 2.11 ft. when estimating an average distance traveled of 18.5 ft. This is approximately an 11.5% error on the total distance traveled.

These results suggest a reasonable estimation for distance traveled over short distances. The errors in estimation is mainly attributed to failure by the detection system to register all steps taken.

3.2 180⁰ Turn Test

The results of ten trials of this test yielded average errors of 3.19 ft. (Std. Dev. = 1.43 ft) and 1.16 ft.(Std. Dev. = 0.93 ft) in the x and y directions, respectively. Because, all motion was concentrated in the x-direction, the larger error in that direction is expected. Furthermore, the error in final position turned out to be 11.2% of the 30 ft traveled by the subject.

These results are very similar to those of the previous test, suggesting an ~11% error can be expected for short distances.

3.3 90⁰ Turn Test

Twenty-one trials of this test resulted in average errors of 2.43 ft.(Std. Dev. = 1.71 ft) and 2.03 ft. (Std. Dev. = 1.47 ft) in final x and y position calculations, respectively. The average error in final position with respect to distance traveled (26 feet) is 12.2%.

Again the error is mainly attributed to failure to detect steps, but the most probable explanation for the increase in percent error in this test is due to any heading drift causing more pronounced errors in this test. Unlike the previous test where heading drift would likely balance itself out during the half turn.

The results of this test are plotted below.

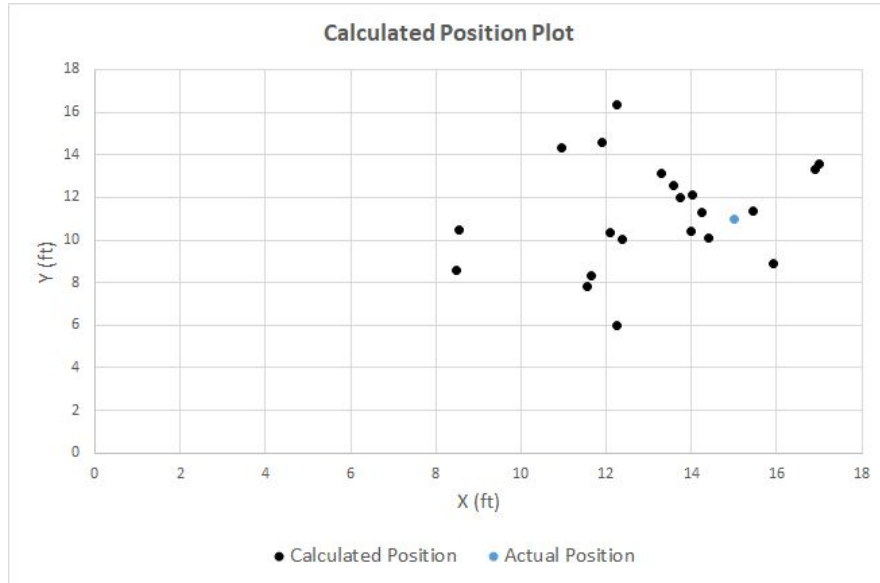


Fig. 13 Plotted results of 90° turn test.

Note: For illustrative purposes the negative y-values which result from a left turn were made positive. This was done for all following plots as well.

3.4 Position Plot Illustration Tests

This test involved saving all collected data from 3 different trials of the 90° test. Trial **3.4.1** was no different than the test in section 3.3. Trials **3.4.2** and **3.4.3** involved walking the path and walking back to the start one time and two times, respectively.

Note: In the following figures, the black and red markers denote the starting and ending positions, respectively.

3.4.1 Path 1

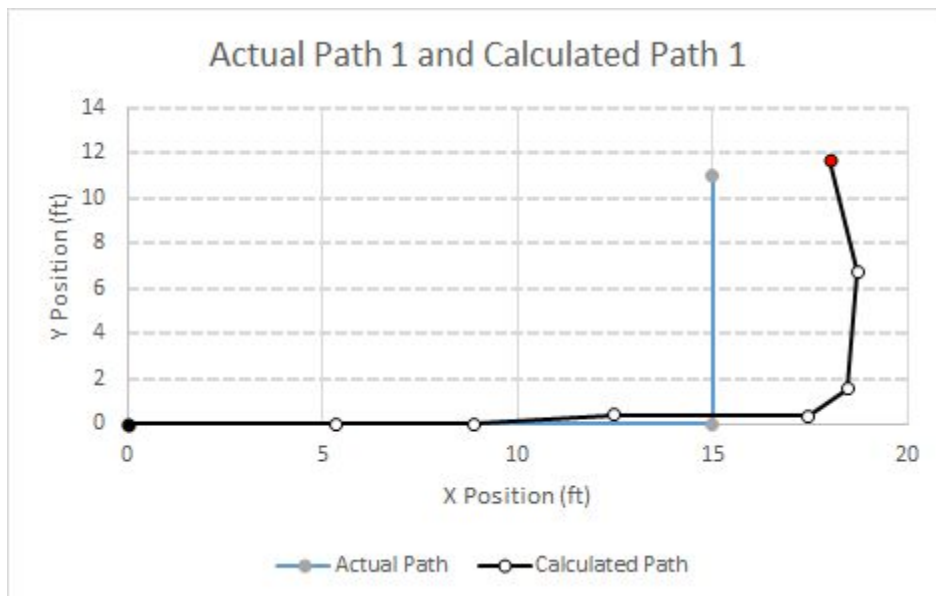


Fig. 14 Actual Path (blue and grey) plotted with Calculated Path (black and white) for Path 1.

The preceding figure shows a relatively precise estimate of position, both qualitatively and quantitatively. The largest error is due to an overestimation of X-position, which is likely due to an inaccurate distance calculation for the first or fourth steps, which are rather large compared to the other steps. Due to space restrictions during testing, walking in the X-direction was not as natural for the test subject as walking in the Y-direction which likely lead to errors throughout this test.

3.4.2 Path 2

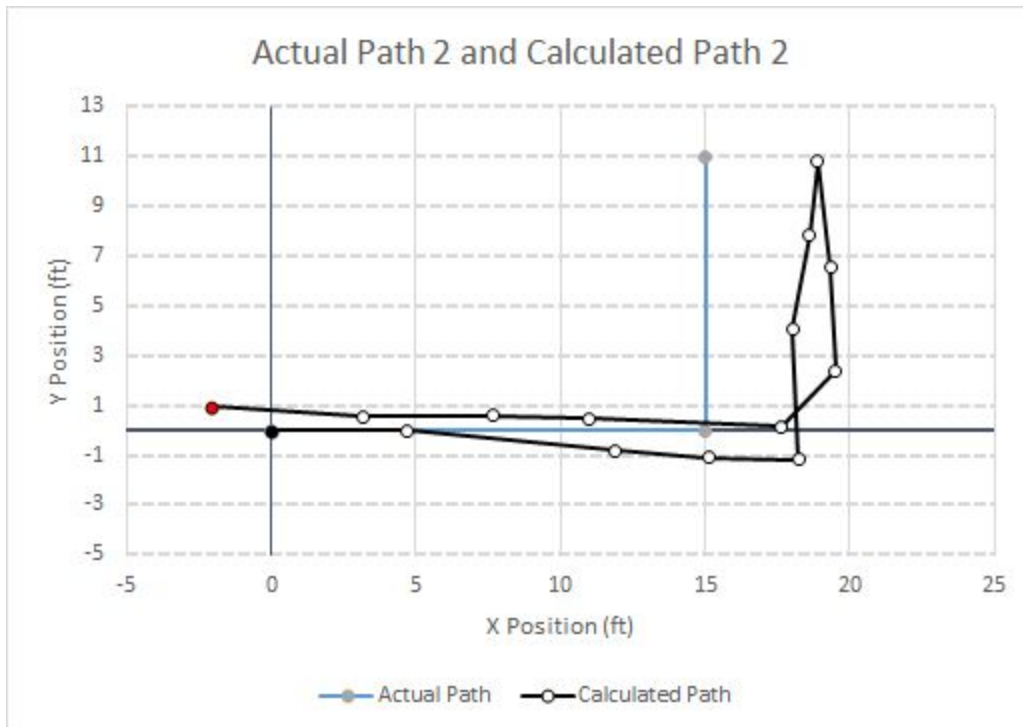


Fig. 15 Actual Path (blue and grey) plotted with Calculated Path (black and white) for Path 2.

Again, the estimated path is both quantitatively and qualitatively reliable. The slight overestimation in X-position is again present, but is still far from a significant error. Furthermore, the calculated final position is very close to the actual final position which was located at the origin.

3.4.3 Path 3

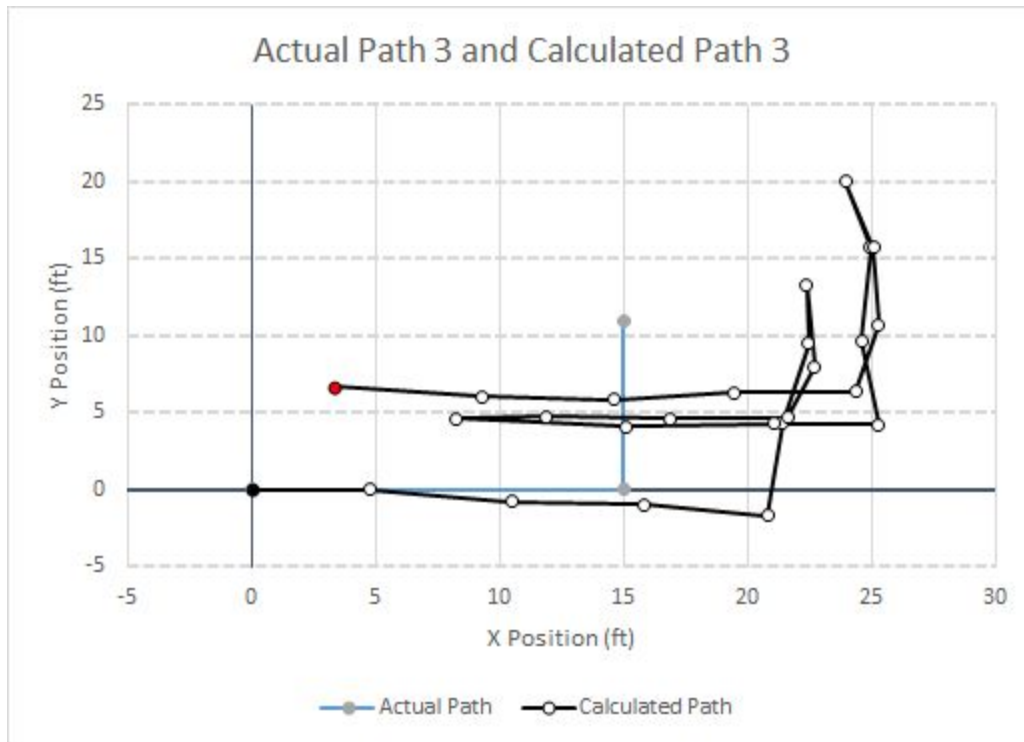


Fig.16 Actual Path (blue and grey) plotted with Calculated Path (black and white) for Path 3.

This trial shows results which are significantly less precise than the two previous trials. This is mainly attributed to the system failing to detect a step during testing, which resulted in a general shift of all following position updates. Nonetheless, the system still did a reasonable job of qualitatively describing the walking path. Moreover, the final position had an error of only 7.2% the distance walked for this trial (104 ft.), suggesting a relatively consistent accuracy in slightly longer ranges.

4 Limitations

Although the results produced were fairly accurate, a few limitations of the system and testing were discovered during the design period.

First, the 10 hz sampling rate of the BNO055 IMU made detecting steps difficult. The true peak of the acceleration spike was often in between samples, resulting in tails of the spike being caught instead. This made quantifying the acceleration magnitude for a test subject very difficult as some steps would result in an acceleration magnitude of nearly 20 m/s^2 but others would only output a magnitude of 0.75 m/s^2 (both measurements exclude gravity).

A possible solution to this issue is a different method to detecting steps. For example, instead of looking for a specific quantity, look at a running average and compare any sampled value to that average value.

Another method would be to hold on to a certain set of recent data points and check the data for a spike, during every iteration. Once a spike is found, the relevant Euler angles could be sampled from that time stamp.

The next limitation of the system was due to the use of only one IMU. First off, the use of only one IMU required the assumption that every step taken by the monitored leg was duplicated exactly by the unmonitored leg. During testing, the test subjects followed this procedure to the best of their abilities, but inherently errors arose from this methodology.

Furthermore, as was noted, the system occasionally failed to recognize a step was taken by the test subject. When this occurred the position estimation would be off by two whole steps, which was measured to be approximately four feet for the test subject.

Next, because only one IMU was used, the test subjects' gait was approximated down to a perfectly symmetrical rigid body system, which is not accurate for the gait pattern of most people. More realistically, at least one knee is significantly bent when a step is taken which significantly changes the geometry of the system, thus the distance of the step. This could be solved by attaching one to two more IMUs on the test subject, or by determining a more realistic distance calculation for more complex gait geometry.

The final limitation was a lack of available testing space. Only relatively short distances were able to be tested which meant the system only ran for one to two minutes at a time before being restarted. This implies that the full effect of the heading drift could not be observed as it is heavily time dependent. In the future this will need to be accounted for more than it was for this project.

Taking these limitations into account, it is a reasonable assumption that the error of the system could be greatly reduced if the suggestions presented were achieved. However, further testing would be required to determine whether or not the system can produce stable and accurate results during long duration and long distance periods.

5 Conclusion and Future Application

This project and analysis has shown that low-cost IMUs have the potential to estimate position over short distances and short periods of time. The accuracy of the system could be vastly increased by implementing a second IMU onto the leg which is currently unmonitored and developing a better real-time step detection algorithm.

As for future applications of this tracking system, there are numerous. As was stated previously, the system could be used to guide the visually impaired or first responders in a smoke-filled environment. Another possible application for the system is to take advantage of the fact that the program is aware of which leg is taking a step. It has been shown that people suffering from Parkinson's disease have a significantly improved cadence when given auditory cues [5]. Through the implementation of a speaker, this system would be able to detect when a

step was taken with either the right or left foot (through the use of two IMUs), and then instruct the user to step with the other foot.

In conclusion, this project provided a solid framework to track position using low-cost IMUs. In the future, with a more robust design and more rigorous testing, researchers could adapt this system to meet the needs a variety of applications.

Appendix

Position Calculation Code. Copy and Paste into a blank Arduino document to use.

```
/*
 * Position Calculator using Bosch Sensortec BNO055 IMU
 * Author: Jermaine Smith
 * Email: smit8332@umn.edu
 * Advisor: Yohannes Ketema
 * Sponsored by UMN UROP Program
 *
 * Last Updated: 8/26/2018
 *
 * Code was modified from
 *
 https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor/arduino-code#adafruit-unified-sensor-system-4-10
 * 'Adafruit Unified Sensor System' code
 */
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <utility/imumaths.h>

#define PI 3.1415926535897932384626433832795

Adafruit_BNO055 bno = Adafruit_BNO055(55);
int start = 0; //variable that indicates sampling may begin
double pos_x = 0; //x- position on 2d plane (m)
double pos_y = 0; //y-position on 2d plane (m)
double pos_mag = 0; //total displacement, used in straight line walking test
double leg_len = 1.0; //leg length in meters
double acc_magnitude; //the magnitude of the acceleration vector excluding gravity
double eX_offset = 0; //This eX_offset value sets whatever direction your first step is in equal to 0 degrees
int calibration = 0; //1 when eX_offset has been set

void setup(void)
{
  Serial.begin(9600);
  Serial.println("Position Calculator Calibrating... Move your leg to calibrate..."); Serial.println("");

  /* Initialise the sensor */
  if(!bno.begin())
  {
    /* There was a problem detecting the BNO055 ... check your connections */
    Serial.print("Ooops, no BNO055 detected ... Check your wiring or I2C ADDR!");
    while(1);
  }

  delay(1000);
  bno.setExtCrystalUse(true);
}
```

```

//Calibrating the IMU
uint8_t system, gyro, accel, mag = 0;

while(system != 3)
{
    bno.getCalibration(&system, &gyro, &accel, &mag);

    Serial.print("CALIBRATION: Sys=");
    Serial.print(system, DEC);
    Serial.print(" Gyro=");
    Serial.print(gyro, DEC);
    Serial.print(" Accel=");
    Serial.print(accel, DEC);
    Serial.print(" Mag=");
    Serial.println(mag, DEC);

    delay(100);
}

Serial.println(""); Serial.println("Calibrated");
start = 1;
}

void loop(void)
{
    imu::Vector<3> euler = bno.getVector(Adafruit_BNO055::VECTOR_EULER);
    imu::Vector<3> acc = bno.getVector(Adafruit_BNO055::VECTOR_ACCELEROMETER);
    //imu::Vector<3> LIN_acc = bno.getVector(Adafruit_BNO055::VECTOR_LINEARACCEL);

    acc_magnitude = (sqrt(pow(acc.x(),2) + pow(acc.y(),2) + pow(acc.z(),2)))-9.81;

    if (start == 1)
    {
        if (acc_magnitude > 0.9 & abs( euler.z() )<40 ) // values within the if statement are characteristics of data when a
step is taken
        {
            if(calibration == 0)
            {
                eX_offset = euler.x(); // This eX_offset value sets whatever direction first step is in equal to 0 degrees
                calibration =1;
            }

            Serial.println("");
            Serial.println("ANGLE SAMPLE");
            Serial.print("eX: ");
            Serial.print(euler.x(),4);
            Serial.print("eY: ");
            Serial.print(euler.y(),4);
            Serial.print("eZ: ");
            Serial.println(euler.z(),4);

            // CALCULATING POSITION: right now it is assumed that every step is duplicated by the second foot

```

```

// Factor of 3.281 used to convert meters to feet.
pos_x = pos_x + 2*(2*leg_len*cos((euler.y()-10)*PI/180)) * cos((euler.x()-eX_offset)*PI/180)*3.281; // x-
position
pos_y = pos_y + 2*(2*leg_len*cos((euler.y()-10)*PI/180)) * sin((euler.x()-eX_offset)*PI/180)*3.281; //
y-position
pos_mag = pos_mag + 2*(2*leg_len*cos((euler.y()-10)*PI/180))*3.281;

Serial.print("X_POS:");
Serial.println(pos_x,4);
Serial.print("Y_POS: ");
Serial.println(pos_y,4);
Serial.print("MAG_POS: ");
Serial.println(pos_mag,4);

//Comment out other print statements and uncomment the following for data which can be easily copied into
spreadsheets
/*
Serial.print(pos_x,4);
Serial.print(",");
Serial.println(pos_y,4);
*/

delay(200); // Helps avoid a single step getting counted twice
}
}
delay(100);
}

```

References

- [1] Ketema, Y, D Gebre-Egziabher, M Schwartz, C Matthews and R Kirker. "Use of gait-kinematics in sensor-based gait monitoring: A feasibility study." *Journal of Applied Mechanics* 4 (2014):
- [2] Ketema, Y., and Gebre. D. "Motion Identification from IMU Data," March 2017.
- [3] "Adafruit BNO055 Absolute Orientation Sensor." *Adafruit*, Adafruit, 22 Apr. 2015, learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor.
- [4] "Arduino UNO Rev3." *Arduino*, Arduino, store.arduino.cc/usa/arduino-uno-rev3.
- [5] Suteerawattananon, M, et al. "Effects of Visual and Auditory Cues on Gait in Individuals with Parkinson's Disease." *Advances in Pediatrics*, U.S. National Library of Medicine, 15 Apr. 2004, www.ncbi.nlm.nih.gov/pubmed/15050439.