Politecnico di Torino

Universidade de Coimbra

Step length estimation

By

Mohammed Ali

A thesis work for a master's degree

in mechatronics engineering

September, 2018

Prof.Mahmoud Tawakoli **Prof.MarcelloChiaberge**

Index

Universidade de Coimbra

POLITECNICO DI TORINO

3. Step length estimation

4. Algorithm

· U C ·

UNIVERSIDADE DE COIMBRA

POLITECNICO DI TORINO

Abstract:

In this work, we present a method for step length estimation and total walked distance estimation using pressure insole for measuring the total pressure and a 9 DOF IMU for measuring the three acceleration components and the three Euler angles (roll, pitch, yow).

The pressure insole was worn in the shoe to sense the applied pressure by the foot for gait event detection, while the IMU was attached on the user's ankle....

All data were received via Bluetooth through an electronic dongle to transmit data from the sensors to the pc.

The algorithm is based on the classical double integration method in the period where the foot is swinging with a zero-velocity update for each step when the foot hits the ground...

Results were validated by walking with a normal and abnormal gait for 10 m. The total error in the walked distance was (1%-3%) & (3%-7%) in the first and second case respectively...

Chapter1:

Gait definition:

Human gait represents this human activity responsible for walking or running...

These activities are characterized by spatial-temporal parameters. The temporal parameters are swinging time which is the time in which the foot is in the air, stance time which is the time in which the foot remains in contact with the ground....

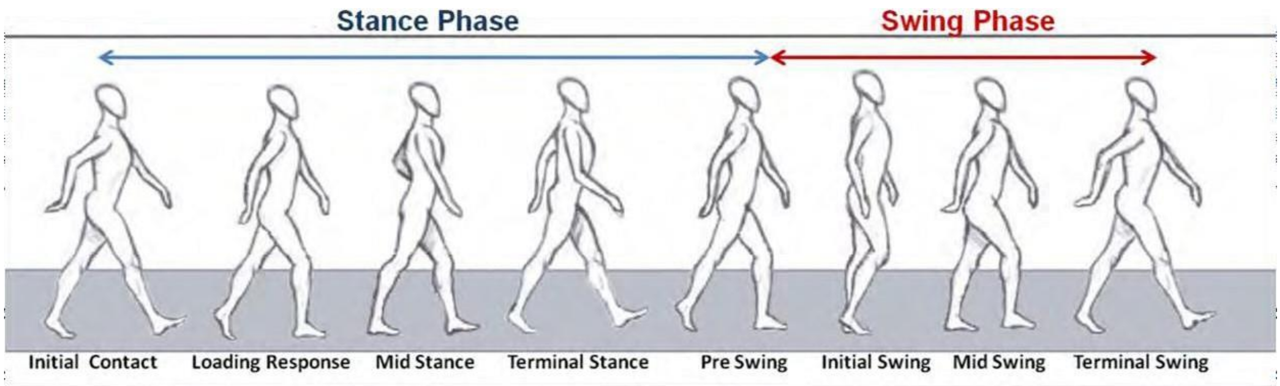
The spatial parameters are the number of steps, the step velocity which is the average velocity of the step, stride length which represents the distance between the toe in the initial position and toe in the final position for each step, the same applies for the heel...

2.1 Gait phases:

The human normal gait is mainly composed of two essential phases, the swing phase and the stance phase.

The human gait is characterized by the cyclic repetition of these two phases...

The stance phase occurs between heel strike which represents the initial contact with the ground and the toe off which represents the last contact with the ground and the start of the swing phase which starts with the toe off and ends with the second ipsilateral foot strike.



As shown in the figure, in the case of normal gait, the stance and swing phases can be further divided in 8 sub phases as follows[1] :

a. **Initial contact**: corresponds to the first contact with the ground and it defines the beginning of the stance phase.
b. **Loading responses**: occurs when the foot is in total contact with the ground.
c. **Mid stance**: starts when the other foot leaves the ground and ends when the other leg advances.
d. **Terminal stance**: starts when the heel leaves the ground and ends when the other foot touches the ground.
e. **Pre-swing**: corresponds to the end of the stance phase and the beginning of the swing phase.
f. **Initial swing**: it begins when the foot leaves the ground and ends when the two legs are in the same position.
g. **Mid swing**: starts when the first leg advances on the second one and ends when the tibia is vertical.
h. **Terminal swing**: occurs before the heel's contact.

2.2gait abnormalities:

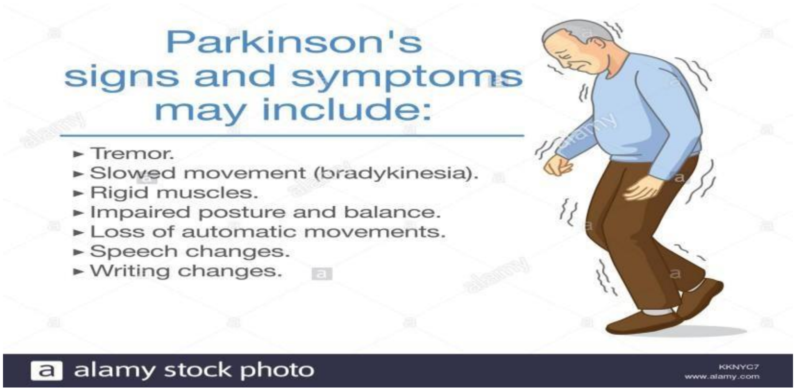
In the walking activity, many parts of the body interact together such as the skeleton, the muscles...

Different factors can affect the normality of the gait, so any pathological nature will require a distinction between *normal gait* and *pathological gait*.

Normal gait refers to the normal activity defined previously

Pathological gait refers to a gait affected by a disease such as muscle weakness or Parkinson disease.

Parkinson disease affects the nerve cells in the brain that produce dopamine which cause a slowed movement and a loss of the gait normality. [2]



2.3 Gait event detection:

It is very important, for the accuracy of our algorithm, to detect correctly the swing and stance phase, because a wrong detection of these phases will lead to an error in the stride length estimation.

In general, we take as reference the period when the foot is flat on the ground since it is the most easy to be identified. When the foot is in total contact with the ground, this corresponds to the exact mid-stance phase.

For the different phases detection we make use of a pressure insole that gives us the total sum of pressure.

Our algorithm is based on threshold detection, such that: if the measured pressure is above the threshold the stance phase is detected, similarly if the pressure value is below the threshold the swing phase is detected. Our threshold corresponds to the pressure measured in the mid-stance phase which corresponds to (80%-85%) of the maximum pressure that can be measured when a person applies his full weight on the foot where the insole is worn.

Chapter2:

Hardware components

To implement our algorithm we use a pressure sensor and a 9 DOF IMU having a triaxial accelerometer, triaxial gyroscope and triaxial magnetometer.

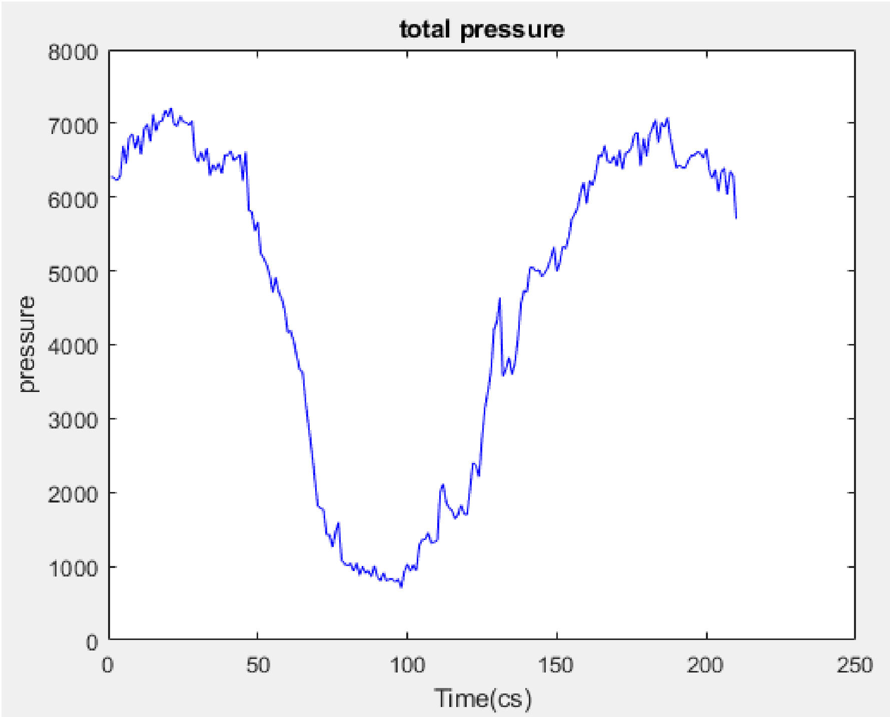
2.1 pressure insole:

The smart insole is composed of 225 pressure sensors mounted on the insole on 40 rows and 10 columns.

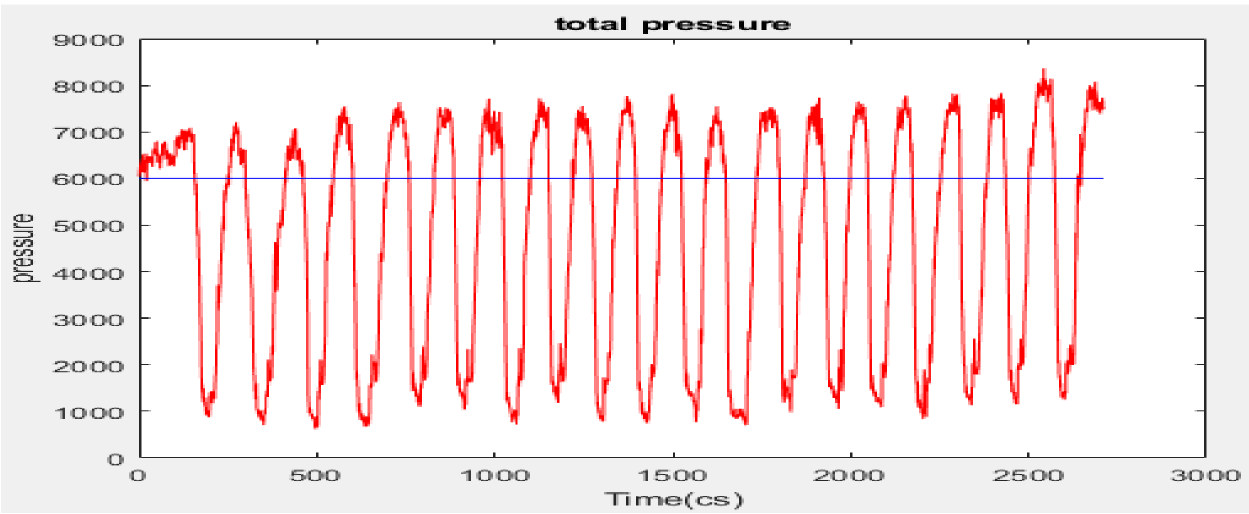
As described before, the insole is responsible of different phases detection.

If we observe the pressure signal during each step we can define the different phases in relation with the measured pressure.

a. Initially we are in full contact with the ground so the pressure is high but it is not the maximum because the weight is distributed on both feet.
b. The second foot starts to move, when the second foot is in the air the pressure reaches a local maximum.
c. The second foot touches the ground, so the pressure starts to decrease again
d. The foot starts to move, so the pressure will decrease gradually reaching a local minimum, when the foot is totally in the air.
e. The foot will touch the ground when the heel strike event occurs, so the pressure starts gradually to increase.
f. When the foot is in total contact with the ground the pressure will more or less constant and the process will be repeated for each step.



In the figure we can observe the different phases and we can see that in this case the threshold was around 6000. The step is detected when the pressure goes below the threshold and then crosses it....



Universidade de Coimbra

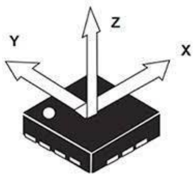
POLITECNICO DI TORINO

2.2 Accelerometer:

An accelerometer is a sensor used to measure the acceleration of a body exploiting the second dynamic's law (F=m*a).

It is composed of a proof mass suspended by springs and constrained to move along a predefined axes.

The proof mass movements can be detected by measuring the capacitance's change.



The IMU are divided into two main categories:

The Gimbaled IMU and the strapdown. These systems obtain the measurements in different frames [3].

We define the sensor frame as the frame attached to the IMU, and the navigation frame as the earth frame.

Gimbaled IMU: in which the IMU is mechanically stabilized in the navigation frame in such a way that the accelerations are directly measured in the navigation frame.

Strap-down IMU: in which the sensor is attached to the sensor frame and moves with it, so it's necessary to rotate the measured acceleration in the sensor frame to obtain the acceleration in the navigation frame by means of rotation angles, which requires the knowledge of the three angles which are obtained from the gyroscope.

It is very important to remember the presence of the gravity component along the z axes.

This component must be taken into account and firstly must be rotated to be aligned with z axes of the navigation frame then it must be subtracted.

In this way measured components will be only due to the movement.

It is important to remember that the sensor is subject to a errors such as noises and sensor bias, which can be removed by a simple calibration.

2.3 Gyroscope:

The gyroscope is used to measure the angular rate (angular velocity) with respect to the three axes.

Gyroscopes are based on the Coriolis law, which establishes that a moving object with velocity ***v*** and angular rate ***Ω*** will be subject to Coriolis acceleration ***acor.***

Which can be expressed by the following equation:
$acor = 2v \times \Omega$.

The Coriolis acceleration lies on the plan perpendicular to the plan defined by the velocity and the angular rate.

When the velocity is perpendicular on the angular rate, the Coriolis acceleration is produced; otherwise the perpendicular components are taken.

The Coriolis acceleration modifies the proof mass's motion.

Once this acceleration is measured, the angular rate can be obtained by using the previous equation.

2.4 Magnetometer:

The magnetometer is used along with the gyroscope to provide an absolute orientation.

The magnetometer is an electronic device able to measure the magnetic field intensity or the direction, strength, relative change of a magnetic field at a particular location [wiki].

A compass is a very important application of magnetometer and it is used to determine the direction of the magnetic field in a known location.

In this way we can determine the magnetic field with respect to the north magnetic pole.

It is very important to remember that magnetometers are subject to noise due to the susceptibility to hard iron and soft iron distortions, so it is very important to make a proper calibration for the magnetometer.

2.5 BNO055 IMU:

We are using BNO055 IMU to measure the three accelerations and the three angles.

Our IMU can be configured in different modes depending on the usage. In the following table we find the different operational modes:

Parameter	Value	[Reg Addr]: Reg Value
CONFIG MODE	CONFIGMODE	[OPR_MODE]: xxxx0000b
Non-Fusion Mode	ACCONLY	[OPR_MODE]: xxxx0001b
	MAGONLY	[OPR_MODE]: xxxx0010b
	GYROONLY	[OPR_MODE]: xxxx0011b
	ACCMAG	[OPR_MODE]: xxxx0100b
	ACCGYRO	[OPR_MODE]: xxxx0101b
	MAGGYRO	[OPR_MODE]: xxxx0110b
	AMG	[OPR_MODE]: xxxx0111b
Fusion Mode	IMU	[OPR_MODE]: xxxx1000b
	COMPASS	[OPR_MODE]: xxxx1001b
	M4G	[OPR_MODE]: xxxx1010b
	NDOF_FMC_OFF	[OPR_MODE]: xxxx1011b
	NDOF	[OPR_MODE]: xxxx1100b

Modes can be classified into two main categories:

Non-fusion mode: in these modes we have the option to use only one of the three sensors alone, or to use two sensors of the three together, or finally to use all the three sensors using the AMG mode.

Fusion modes: in these modes we can use two or more sensors together activating the fusion algorithm in such a way that the measured values are correctly acquired in the sensor frame.

Operating Mode		Available sensor signals			Fusion Data	
		Accel	Mag	Gyro	Relative orientation	Absolute orientation
	CONFIGMODE	-	-	-	-	-
Non-fusionmodes	ACCONLY	X	-	-	-	-
	MAGONLY	-	X	-	-	-
	GYROONLY	-	-	X	-	-
	ACCMAG	X	X	-	-	-
	ACCGYRO	X	-	X	-	-
	MAGGYRO	-	X	X	-	-
	AMG	X	X	X	-	-
Fusion modes	IMU	X	-	X	X	-
	COMPASS	X	X	-	-	X
	M4G	X	X		X	-
	NDOF_FMC_OFF	X	X	X	-	X
	NDOF	X	X	X	-	X

Fusion modes can be further divided into two classes depending on the desired orientation of the sensor.

Relative orientation, the fusion algorithm calculates the heading of the sensor with respect to the initial position; in other words when the sensor is turned on, the yaw zero angle corresponds to the initial yaw angle.

Absolute orientation, the fusion algorithm calculates the heading angle with respect the north magnetic pole, in other words the zero yaw angle corresponds to the direction of the magnetic north.

The user is free to use any operational mode, but for our application we need at least 6 DOF so we need to use the accelerometer and the gyroscope. If we want to have 9 DOF because we need more stability in the angles calculation we must use all the three sensors.

If we decide to use the non-fusion AMG mode we have to implement our fusion algorithm, in order to align the gravity vector with the z axes to remove it correctly.

One of the most classic fusion algorithms is the Madgwick&Mahony algorithm. Mahony algorithm is used to fuse gyro and accelerometer data, while the Madgwick algorithm is used to fuse gyro, magnetometer and accelerometer data.

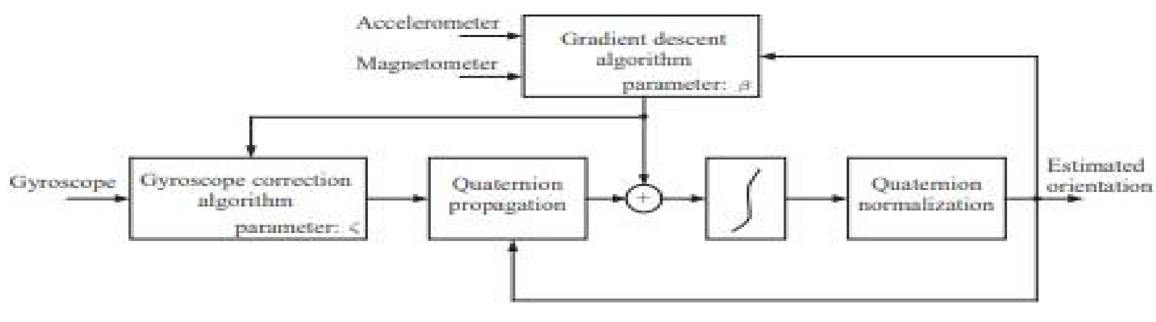
$$\dot{\mathbf{R}} = \mathbf{R}\mathbf{\Omega}_\times \qquad \text{with} \qquad \mathbf{\Omega}_\times = \boldsymbol{\omega}\mathbf{J} = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}, \qquad \text{eq(1).}$$

The main goal is to calculate the kinematic equation for a body rotation. $\omega = (\omega_x, \omega_y, \omega_z)^T$ is the angular velocity, R is the orientation of the sensor frame with respect to the navigation frame.

The Mahony algorithm is based on correcting the input of the input of the previous equation which is the angular velocity by adding a corrective factor $\delta\omega$, calculated by a PI controller based on the error **e,** calculated from the previous estimated attitude and acceleration vector.

Regarding the Madgwick algorithm, it uses a quaternion form so we need to transform the acceleration vector into a quaternion [4].

The main problem is that any rotation around the direction will give the same vector but a different quaternion, to solve this problem we have to find the rotation or the quaternion that aliens the gravity vector **g** in the navigation frame with the measured acceleration vector, to find Ra such that $\mathbf{a} = -\mathbf{R}_a^T \mathbf{e}_z$, where ez is the gravity vector.



http://www.olliw.eu/2013/imu-data-fusing/#refRO2

Fortunately we do not need to use any fusion algorithm in our work, since the IMU does automatically the fusion algorithm in the fusion modes.

The compass, NDOF modes provide absolute orientation, while the IMU & M4G provide a relative orientation.

The absolute orientation is not suitable for our application because subjects must walk inside a room, so the walking direction will be different from the north magnetic pole indicated by magnetometer. In order to avoid complex calculations, the absolute orientation modes will not be used.

For our application we have to use IMU or M4G mode, but the M4G mode is using the accelerometer and the magnetometer so it is not possible to determine the angular rate.

The best mode to use for our application is the IMU mode. Even though it does not have the magnetometer, it is still the best mode to use because it provides the three accelerations and the rotation in angles (roll, pitch, yaw) or in quaternion form (qw, qx, qy, qz).

2.6 sensor calibration:

Accelerometer calibration: it is sufficient to place the sensor in a stable position along the three axes x, y, z to calibrate the accelerometer.

Gyroscope calibration: it is sufficient to place the sensor in a fixed and stable position for few seconds to calibrate the accelerometer.

Magnetometer calibration: to calibrate the magnetometer we have to make random movements like number 8 in the air. The calibration process is very important for the magnetometer, since it is more susceptible to disturbances

For the accelerometer and the gyroscope the offset errors are negligible since they are corrected in the fusion algorithm implemented by the sensor.

Usually the calibration process is very fast...

Chapter 3:

Step length estimation:

In this section we want to give a brief overview on sensor positioning and different algorithms for stride length and walked distance estimation.

Approach	Author	Assumption	Method	Results
Body attaching	Shin *et al.* [17]	- Pedestrians walk or run - Attaching accelerometers to the body	- Using biaxial accelerometer and gyroscope sensors - Counting the number of steps - Estimating the step length as a linear combination of walking frequency and acceleration variance.	- Accuracy of step length estimation for walking cases is 95%, 96% and 96% for slow, normal, and fast walking, respectively - Step length estimation provides an accuracy of 96% for running case
	Shih *et al.* [12]	- Users walk normally in a straight line with average distance of 664.5 cm. - Attaching smartphone on the waist - Placing smartphone on the chest pocket	- Using one triaxial accelerometer and one gyroscope sensor from a smartphone - Using double integral of vertical acceleration to estimate stride length	- Accuracy of distance estimation based on attaching the smartphone on the waist is 97.35%. - Placing smartphone on the chest pocket provides a 96.14% accuracy rate of distance estimation
Ankle attaching	Wang *et al.* [9]	- Users walk along the outside of a sports area that is 559 m long - Attaching triaxial accelerometer on users's ankles	- Using a triaxial acceleration data to analyze gait and estimate the step velocity - Estimating step length as a linear regression model of step frequency and step velocity	- Accuracy of walking distance estimation is 96.42%
Leg attaching	Bennett *et al.* [18]	- Subjects walk in a straight line with average distance of 3.55 m - Placing sensors on the thigh and shin of the right leg	- Modeling human leg as a two-link revolute robot, then using Extended Kalman Filter (EKF) to estimate the displacement in a straight line	- EKF distance estimation had an average error of 2%
Shoe attaching	Alvarez *et al.* [15]	- Subjects walk in a 10 m straight distance - Attaching a sensor module in the front of the users's shoes	- Using a biaxial accelerometer and a gyroscope sensor - Double integrating the horizontal acceleration in the swing phase to estimate the walking distance.	- Mean estimation error rate is 10% with a single sensor module attached on one foot - Result is improved to 7% when mounting a sensor module on each foot
	Wang *et al.* [19]	- Three subjects perform two sets of 40 m level walking, 10-step stair ascending and 10-step stair descending - Mounting a triaxial accelerometer, a gyroscope and orientation sensors to shoes	- Using double integral of acceleration to estimate walking distance - Using zero velocity update (ZUPT) to reset velocity when a foot becomes stationary	- Absolute error of $(3.08 \pm 1.77)\%$ in distance estimation
	Meng *et al.* [16]	- Subjects walk in a straight line for 10 m - In long distance experiment, subjects walk for a distance of approximately 132 m - Attaching an inertial/ magnetic measurement unit in the front of the users' shoes	- Using a module containing a triaxial accelerometer, a triaxial gyroscope sensor and a triaxial magnetometer - Creating a zero velocity update method based on the stride information to further correct the acceleration - An adaptive Kalman Filter is used to estimate the position	- Position error is 0.44 m $\pm$ 0.2 m for short distance (4.4%). - In long distance experiment, the position error is 4.31 m $\pm$ 1.77 m (3.6%)

3.1 Overview:

We can classify step length estimation algorithms into two essential categories: biomechanical algorithms, used for sensors mounted near to the user's centre of mass and they involve some user's parameters as foot or leg length.

The other ones are called parametric algorithms, used for sensors mounted on the user's belt (pelvic) and they are based on some gait parameters such as step frequency or accelerometer variance or both.

a. Inverted pendulum model: in this algorithm the human gait is simulated as an inverted pendulum model. Zijlstra et al [5] proposed a relationship between the step length and the vertical displacement of the COM of human body.
   $SL_{M1} = 2 \times K\sqrt{2hl - h^2}$ Where k is the calibration parameter calculated as the ratio between the mean of the real and mean of the estimated one, l is the user's leg length and h is the vertical displacement in each step.

b. Double pendulum model: it is another biomechanical model for stride length estimation. In this method we consider the human as two different pendulums; the first one is used during the swing phase from foot off to foot initial contact and it is similar to the inverted pendulum. The second one is a different one used in the double stance phase from foot initial contact to foot off.

$$SL_{M2} = SL_{sp} + SL_{dsp} = 2 \times \sqrt{2hl - h^2} + C \times l_{foot} \quad [5]$$

Where the first term is defined previously, lfoot is the length of user's foot and c is a constant.

c. Weinberg's algorithm: it is a parametric algorithm that calculates the stride length as function of the difference between the maximum and minimum vertical acceleration.

$$SL_{M3} = K \times \sqrt[4]{a_{max} - a_{min}}$$

Once again this algorithm has a calibration constant that must be tuned.

3.2 Conventions:

The sensor attitude is provided in angle or quaternion form.

Angle representation has some problems such as singularity. We noticed that when the pitch angle reaches 45 degrees the yaw angle is increased by 180 degrees, the same happens when the pitch angle reaches -45 degrees the yaw angle is suddenly decreased by -180 degrees.

The reason for which this happens is that during the pitch angle rotation the convention assumes as positive rotation the counter clockwise rotation, so when the angle is 45 or -45 degrees the sensor cannot recognize if the sensor is going clockwise or counter clockwise so both solutions yaw0 or yaw0+180 are admissible for this case, but when the pitch reaches 45 the sensors imagines that the rotation sense has changed so it rotates around the z axes by 180 to keep the rotation sense unchanged.

The roll angle is the angle around the x axes and it is defined between (-180&180).

The pitch angle is the angle around y axes and it is defined between (-90&90).

The yaw angle is the angle around the z axes and it is defined between (0&360).

The three elementary rotations are: (roll, x) (pitch, y)

(yaw, z), but in which order rotation are done? Which is the sign of the angles?

$$R_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -sen(\psi) \\ 0 & sen(\psi)0 & \cos(\psi) \end{bmatrix}$$

$$R_y(\vartheta) = \begin{bmatrix} \cos(\vartheta) & 0 & \mathrm{sen}(\vartheta) \\ 0 & 1 & 0 \\ -\mathrm{sen}(\vartheta) & 0 & \cos(\vartheta) \end{bmatrix}$$

$$R_z(\varphi) = \begin{bmatrix} \cos(\varphi) & -\mathrm{sen}(\varphi) & 0 \\ \mathrm{sen}(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

We have different 8 possibilities in the rotation order. In this work we will use the (x, y, z) rotation order.

The sign of the angle is positive if the rotation is performed in counter clockwise sense.

3.3 Quaternion:

Quaternions are 4 numbers defined on the real number field with base {1, i, j, k}, where i, j, k are hyper-complex numbers that satisfy the following anticommutative multiplication rules:

$$\mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 = \mathrm{ijk} = -1$$
$$\mathrm{ij} = -\mathrm{ji} = \mathrm{k}$$
$$\mathrm{jk} = -\mathrm{kj} = \mathrm{i}$$
$$\mathrm{ki} = -\mathrm{ik} = \mathrm{j}$$

A quaternion q is defined as a linear combination expressed in the subspace {1, i, j, k}:

$$\mathrm{q} = q_0 1 + q_1 \mathrm{i} + q_2 \mathrm{j} + q_3 \mathrm{k}$$

Where q0, q1, q2, q3 are real numbers.

Quaternion can be represented as a quadruple of real number:

$$\mathrm{q} = (q_0, q_1, q_2, q_3)$$

Quaternion can be seen also as the sum of a real and a vector part, where the real part is q0 and the vector part is

$$\mathbf{q}_v = q_1\mathrm{i} + q_2\mathrm{j} + q_3\mathrm{k}.$$

So we can write q= (qr, qv), or q=qr+qv.

Quaternions do not present any singularity, so we decided to use quaternion.

The only problem is that we do not know the meaning of these four numbers so if we want to view the angles we must convert quaternions to angles.

To rotate the sensor from the sensor frame to the navigation frame, we have two ways:

a. Rotate the acceleration vector by the given quaternions to bring it to the navigation frame.
b. Convert quaternions to angles and then convert angles into rotational matrixes in order to bring the acceleration vector to the navigation frame.

To rotate a vector qx by a quaternion u, calling qy the new vector we have:

$$\mathrm{q}_x \mapsto \mathrm{q}_y = \mathrm{u}\mathrm{q}_x\mathrm{u}^*$$

Where u* is the conjugate of u, which is represented by the same real part and the opposite in sign vector part.

$$\mathrm{q}^* = q_0 - (q_1\mathrm{i} + q_2\mathrm{j} + q_3\mathrm{k}) = (q_r, -\mathbf{q}_v) = q_r - \mathbf{q}_v = (q_0, -q_1, -q_2, -q_3)$$

Here we notice that the quaternion is multiplied by the vector and the product is then multiplied by the conjugate of the quaternion.

We need to know how quaternion's product is done.

Given the two quaternion h & g, the product h*g is the following:

$$\begin{aligned}\mathrm{hg} \quad = \quad & (h_0g_0 - h_1g_1 - h_2g_2 - h_3g_3)1 + \\ & (h_1g_0 + h_0g_1 - h_3g_2 + h_2g_3)\mathrm{i} + \\ & (h_2g_0 + h_3g_1 + h_0g_2 - h_1g_3)\mathrm{j} + \\ & (h_3g_0 - h_2g_1 + h_1g_2 + h_0g_3)\mathrm{k}\end{aligned}$$

3.4 ZUPT:

The stance phase detection is very important since it is the only moment where the velocity in the sensor frame is theoretically zero, but in the real case the velocity in stance phase is not actually zero either because the accelerometer is subject to noise so the actual velocity is not zero or the sensor attached to the ankle or the shank is still moving even though the is on the ground.

Any non null velocity during this period is considered therefore as an error.

The ideal placement of the sensor for applying the ZUPT is the foot, because in this situation when the foot hits the ground the ZUPT will be applied and this will result in a correct detection of the stance phase and so a correct calculation of the gait parameters.

The zupt can be classified into three approaches:

a. conventional zupt
b. zupt with velocity correction

a. conventional zupt:

In this approach the velocity is set to zero during the stance phase so the error growth is bounded and the step or the total position is calculated only in swing phase.

b. Zupt with velocity correction:

In this case we use the conventional zupt with velocity correction during the stance phase.

The correction is based on (LWI) linear weighted interpolation, which consists in using the mean of the velocity during stance phase as weighting factor.

Considering the not corrected velocity v, considering as period of interest the stance phase or the period from the heel strike to the toe off and considering as tstart the starting time for swing phase, tend the ending time for swing phase corresponding to the start of stance phase, tfinal the end of the step.

The correction factor of the velocity is:

$$\mu = \frac{\sum_{i=i_{end}}^{i_{final}} Vn_i}{(i_{final} - i_{end})}$$

The corrected velocity during stance phase:

$$Vn_i^{¢} = Vn_i - \mu \Big]_{i=i_{end} \cdots i_{final}}$$

The corrected velocity during swing phase:

$$Vn_i^{¢} = (Vn_i - m)\left(\frac{t_i - t_{start}}{t_{end} - t_{start}}\right)\Bigg]_{i=i_{start} \cdots i_{end}}$$

Very similar approach is based on the residual velocity, which is based on the fact that when the stance phase starts the velocity is not actually zero. This value that is supposed to be zero is used as correction value and it is called residual velocity.

The difference between this correction mode and the previous one is that in this one we use the residual velocity as correction value instead of the mean during stance phase.

Chapter 4:

Algorithm implementation:

In this section we are going to discuss the algorithm, the main issues, the results and conclusion.

It is very important to understand how to program the IMU, how deal with angle and rotation conventions, how to select the correct threshold, how to remove velocity's drift, how to deal with sampling data rate, which integration method must be used.

4.1 IMU programming:

The IMU was programmed by a cypress microcontroller

In the following we will see how to program the IMU in order to have the three accelerations, quaternions and angles.

First of all we must declare the acceleration, quaternion and angles variables in the c file.

```
unsigned char  EULER_DATA[6];
unsigned char  QUA_DATA[8];
unsigned char  RAW_DATA_ACCEL[6];
```

Please note the dimension of the different vectors

Universidade de Coimbra

Politecnico di Torino

```
static void BNO55_ReadAccel(unsigned char s_addr) {

   MPU_I2C_Read_Multi(s_addr, LIA_ACCEL_DATA_ADR, 6, &RAW_DATA_ACCEL[0]);    // Read Accel data

   if(s_addr == BNO55_I2C_ADDR_INTERNAL){
       BNO55_Internal.accel.x = (int)((RAW_DATA_ACCEL[1] << 8) | RAW_DATA_ACCEL[0] );
       BNO55_Internal.accel.y = (int)((RAW_DATA_ACCEL[3] << 8) | RAW_DATA_ACCEL[2] );
       BNO55_Internal.accel.z = (int)((RAW_DATA_ACCEL[5] << 8) | RAW_DATA_ACCEL[4] );
    }

    if(s_addr == BNO55_I2C_ADDR_EXTERNAL){
       BNO55_External.accel.x = (int)((RAW_DATA_ACCEL[1] << 8) | RAW_DATA_ACCEL[0] );
       BNO55_External.accel.y = (int)((RAW_DATA_ACCEL[3] << 8) | RAW_DATA_ACCEL[2] );
       BNO55_External.accel.z = (int)((RAW_DATA_ACCEL[5] << 8) | RAW_DATA_ACCEL[4] );
    }

}
```

In this function we read the acceleration data depending on the address whether it is internal or external in order to distinguish between them.

```
static void BNO55_ReadEuler(unsigned char s_addr) {
  //                     Adresse         Register             wohin schreiben
   MPU_I2C_Read_Multi(s_addr, EULER_DATA_ADR, 6, &EULER_DATA[0]);   // Read Accel da
    MPU_I2C_Read_Multi(s_addr,  BNO055_QUA_DATA_W_LSB, 8, &QUA_DATA[0]);   // Read A

    if(s_addr == BNO55_I2C_ADDR_INTERNAL){
       BNO55_Internal.euler.Heading = (int)((EULER_DATA[1] << 8)  | EULER_DATA[0] );
       BNO55_Internal.euler.Roll    = (int)((EULER_DATA[3] << 8)  | EULER_DATA[2] );
       BNO55_Internal.euler.Pitch   = (int)((EULER_DATA[5] << 8)  | EULER_DATA[4] );

       BNO55_Internal.quaternions.W = (int)((QUA_DATA[1] << 8)  | QUA_DATA[0] );
       BNO55_Internal.quaternions.X = (int)((QUA_DATA[3] << 8)  | QUA_DATA[2] );
       BNO55_Internal.quaternions.Y = (int)((QUA_DATA[5] << 8)  | QUA_DATA[4] );
       BNO55_Internal.quaternions.Z = (int)((QUA_DATA[7] << 8)  | QUA_DATA[6] );


    }
```

The same is done with the Euler angles and quaternions to read them in the internal and external address.

```
void BNO55_Read(unsigned char s_addr) {
   //BNO55_ReadTemp(s_addr);

   BNO55_ReadEuler(s_addr);

   BNO55_ReadAccel(s_addr);
// BNO55_ReadGyro();

 //  BNO55_ReadMagnet();
}
```

In the previous function we call the previously defined functions of the accelerometer and the Euler angles, the same can be done for the gyroscope and the magnetometer if needed.
It’s important to give the right address, since it is the only argument of the function.
This function is essential for our IMU, as it allows to read the three components of accelerations, angles and quaternion with the internal or external address.

```
void Config_BNO55(unsigned char s_addr){
   unsigned char dat;
   int i;
   // Select BNO055 config mode
    dat = 0x00;
    MPU_I2C_Write(s_addr, BNO055_OPR_MODE, 1,  &dat);
    CyDelay(50);

    dat = 0b10001;
    MPU_I2C_Write(s_addr, BNO055_SYS_TRIGGER, 1,  &dat
    CyDelay(50);

    // Select page 1 to configure sensors
    dat = 0x01;
    MPU_I2C_Write(s_addr, BNO055_PAGE_ID, 1,  &dat);

    // Configure ACC
    dat = 0b00001101;   // 4g , Bandwidth = 62,5Hz , n
    MPU_I2C_Write(s_addr, BNO055_ACC_CONFIG,1,  &dat);
```

In this funtion we configure the IMU by passing the right address.
A local variable is used to alias the address.
First we define the operation mode, then system trigger and the page ID by passing the proper address.
Note that in order to configure properly the sensors it is important to select page 1.
As first thing we configure the accelerometer.
In the next part we will see how to configure the restant sensors.

```
// Configure GYR
dat = 0b00011000;     // Range 2000dps; Bandw = 47Hz
MPU_I2C_Write(s_addr, BNO055_GYRO_CONFIG_0, 1,  &dat);

dat = 0x00;       // Normal OpMode
MPU_I2C_Write(s_addr, BNO055_GYRO_CONFIG_1, 1,  &dat);

// Configure MAG
// 20Hz output Rate, OpMode = regular, Power Mode Normal
dat = 0b000110;
MPU_I2C_Write(s_addr, BNO055_MAG_CONFIG, 1,  &dat);

// Select page 0 to read sensors
dat = 0x00;
MPU_I2C_Write(s_addr, BNO055_PAGE_ID,1,  &dat);

// Select BNO055 sensor units (temperature in degrees C,
dat = 0x01;
MPU_I2C_Write(s_addr, BNO055_UNIT_SEL,1,  &dat);
```

This code is part of the same function, in which we configure the gyroscope.
Note that there are two modes to configure the gyroscope, a first one for application mode and the other one is for normal operational mode.
The magnetometer is configured in regular operational mode.
Once all sensors are configured we need to switch page ID in order to read them by setting page_ID to 0.

```
// Write Configuration to BNO55 Registers
// EXIT Config mode and switch to selected Operation mode
for(i = 0; i <= 5; i++){       // S 21
//        dat = AMG_CON;
    dat = IMU_CON;
    MPU_I2C_Write(s_addr, BNO055_OPR_MODE, 1, &dat);
    CyDelay(50);
    MPU_I2C_Read(s_addr,  BNO055_OPR_MODE , 1, &dat);
    if(dat == IMU_CON) {

        return;
        }
    }
```

Here we write configuration to IMU registers by calling the writing function and then read the selected configuration. At the end we should exit the configuration mode and switch to selected operational mode to use the IMU properly.
To select the desired operational mode refer to the manual of BNO055.

```c
typedef struct {
  int16 x;
  int16 y;
  int16 z;
} tAxis;
```

```c
typedef struct {
  int16 Heading;
  int16 Roll;
  int16 Pitch;
} eAxis;

typedef struct {
  int16 W;
  int16 X;
  int16 Y;
  int16 Z;
} qAxis;
```

```c
typedef struct {
   tAxis accel;
   tAxis gyro;
   tAxis mag;
   eAxis euler;
   qAxis quaternions;
   short temp;
   float gdt;
} tSensor;
```

Here we define some structs to save the three components of accelerations, gyroscope and magnetometer
The second struct is used for the Euler angles and the corresponding quaternions.
The third struct unifies the precedents structs and considers the temperature and the tolerance.

```c
tSensor BNO55_Internal,BNO55_External;
```

This will be our data struct used in the different functions, the first one for the internal data, the second for the external data.

```
I2CM_Start();
CyDelay(5000);//Wait for system stabilization!!!
    for(k=0;k<10;k++)
    {
    if(BNO55_Detect(BNO55_I2C_ADDR_INTERNAL)==1)
    {   Config_BNO55(BNO55_I2C_ADDR_INTERNAL);
        MPU_I2C_Read(BNO55_I2C_ADDR_INTERNAL,  BNO055_OPR_MODE , 1, buffer);
        if(buffer[0]==IMU_CON)
        {
            int_AHRS = 1;
            break;
        }
    }
    }
```

Here we start to program the IMU acting on its internal address.
If the internal address is passed correctly with the detection function, the IMU is configured based on this and we can select the desired operational mode, we normally select the normal operation mode.
After this we can read the address by the reading function
Pay attention to IMU buffer, to avoid memory stack overflow.

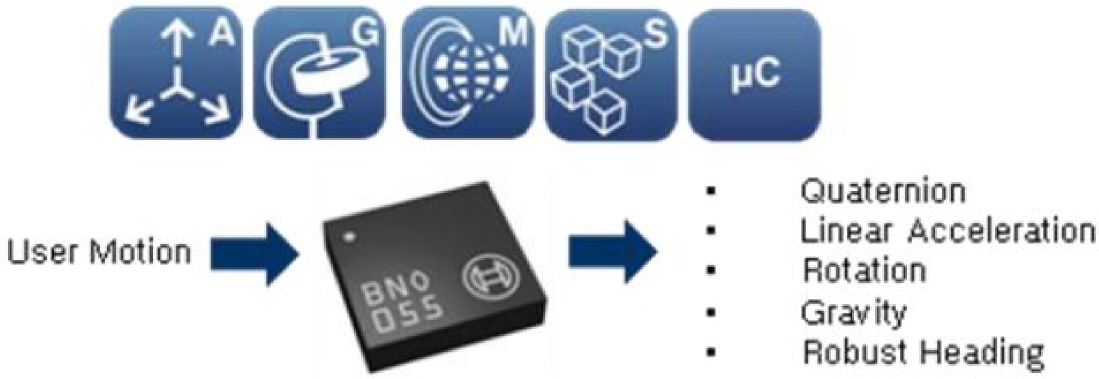
```
short BNO55_Detect(unsigned char s_addr) {
  unsigned char dat = 0;

  MPU_I2C_Read(s_addr, RM_CHIP_ID , 1, &dat);

  if(dat == RM_CHIP_ID_content) {
    return 1;
  }else{
    return -1;
  }
}
```

Here we describe the detect function used to detect the presence of the IMU.
It basically uses the IMU address, then uses the reading funtion and finally verifies if it is passed correctly and corrispondes to chip content, in this condition it gives 1 otherwise it gives -1 and breaks the code.

4.2. IMU BNO055:

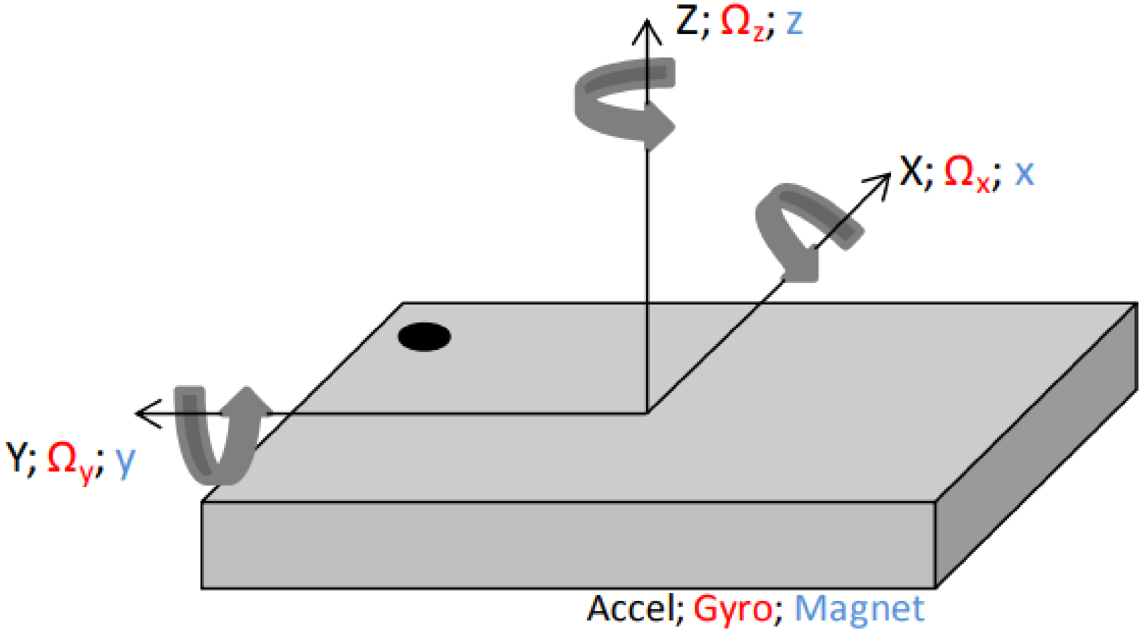


	Operating Mode	Available sensor signals			Fusion Data	
		Accel	Mag	Gyro	Relative orientation	Absolute orientation
	CONFIGMODE	-	-	-	-	-
Non-fusionmodes	ACCONLY	X	-	-	-	-
	MAGONLY	-	X	-	-	-
	GYROONLY	-	-	X	-	-
	ACCMAG	X	X	-	-	-
	ACCGYRO	X	-	X	-	-
	MAGGYRO	-	X	X	-	-
	AMG	X	X	X	-	-
Fusion modes	IMU	X	-	X	X	-
	COMPASS	X	X	-	-	X
	M4G	X	X		X	-
	NDOF_FMC_OFF	X	X	X	-	X
	NDOF	X	X	X	-	X

Here are the different operation modes of the device, in this work we used the fusion mode IMU for simplicity.
This selected mode uses the accelerometer and gyroscope, with relative orientation fusion data.
Note that the CONFIGMODE is intially used to configure the device.

Parameter	Value	[Reg Addr]: Reg Value
CONFIG MODE	CONFIGMODE	[OPR_MODE]: xxxx0000b
Non-Fusion Mode	ACCONLY	[OPR_MODE]: xxxx0001b
	MAGONLY	[OPR_MODE]: xxxx0010b
	GYROONLY	[OPR_MODE]: xxxx0011b
	ACCMAG	[OPR_MODE]: xxxx0100b
	ACCGYRO	[OPR_MODE]: xxxx0101b
	MAGGYRO	[OPR_MODE]: xxxx0110b
	AMG	[OPR_MODE]: xxxx0111b
Fusion Mode	IMU	[OPR_MODE]: xxxx1000b
	COMPASS	[OPR_MODE]: xxxx1001b
	M4G	[OPR_MODE]: xxxx1010b
	NDOF_FMC_OFF	[OPR_MODE]: xxxx1011b
	NDOF	[OPR_MODE]: xxxx1100b

It is important to know how to select the correct operation mode by selecting the right address and write on the corresponding register.
We should consider that to switch between configmode and any other mode it takes 7ms, while for the opposite it takes 19ms.



This is the default coordinate system, that corresponds to the configuration P1 of the sensor.

Axis configuration byte: Register Address: **AXIS_MAP_CONFIG**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved		Remapped Z axis value		Remapped Y axis value		Remapped X axis value	

The default values corrisponde to x axis=x, y axis=y, z axis=z

Axis sign configuration byte: Register Address: **AXIS_MAP_SIGN**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved					Remapped X axis sign	Remapped Y axis sign	Remapped Z axis sign

Positive sign corrispondes to 0, while negative to 1
The default value is 0x00.

For sensors calibration it is important to consider important passegges to allow a correct calibration of the device.

For the accelerometer we should put the device in 6 different positions for few seconds assuring that there is a calm movement between consecutive movements.
We can consult the register CALIB_STAT to verify the calibration status.
For the gyroscope it is sufficient to put the device in any stable position for few seconds to calibrate the device
We can consult the register CALIB_STAT to verify the calibration status.
Regarding the magnetometer it is important to do some random movement as writing the infinite symbole on air.

Chapter 5:
Results and Conclusions:

In this chapter we discuss about the results conducted in the department of electrical engineering at University of Coimbra by trying to follow a normal gait and a distortional gait simulated by the user for around 5 meters.

The pressure sensor was worn by the user and the IMU was placed on the ankle.

In this work the actual step length is compared to the measured one to calculate the error on the step and the total walked distance, then the variance and median are estimated.

Different experiments were held by different users.
The steps were fixed on the ground and real time acquisition was done and saved into mat-lab file and the user had to follow those steps to compare the measured one with the actual one.
For the normal gait 6 different experiment were done for each user to calculate the error and the variance.
For the distorted gait, the same steps were fixed on the ground but only two users simulated a distorted gait by not following the proper or natural walking.
Each user has done 6 different experiments.

The user was wearing the insole inside the shoe and the IMU was mounted on the ankle as shown in the previous picture. You should consider that in this case the angles of the gyroscope roll and yaw will be inverted.

5.1 Normal gait:

First subject

Actual step	Experiment 1	Experiment2	Experiment3
67	72	66	65
43	42	44	38
62	45	58	55
46	42	49	49
28	29	29	28
51	57	51	35
59	55	50	45
75	66	56	75
35	33	38	37
35	33	28	30
45	41	40	31
42	46	48	50
33	40	40	33
45	43	40	41

Actual step	Experiment 4	Experiment5	Experiment6
67	70	62	67
43	39	45	41
62	55	48	63
46	45	50	38
28	22	34	30
51	34	46	45
59	60	55	57
75	63	75	66
35	40	32	34
35	32	34	40
45	45	38	48
42	40	36	40
33	45	36	41
45	37	39	43

Second subject

Actual step	Experiment 1	Experiment2	Experiment3
67	64	59	65
43	38	35	34
62	61	61	58
46	40	45	45
28	26	30	34
51	50	49	46
59	45	51	54
75	78	80	71
35	35	33	40
35	34	30	34
45	42	46	31
42	52	38	41
33	40	25	37
45	39	44	40

Actual step	Experiment 4	Experiment5	Experiment6
67	62	70	67
43	40	38	37
62	69	67	68
46	51	38	50
28	23	28	22
51	49	60	45
59	54	53	65
75	80	74	65
35	33	31	37
35	40	38	38
45	46	49	39
42	42	40	46
33	31	25	37
45	46	38	48

Third subject

Actual step	Experiment 1	Experiment2	Experiment3
67	64	59	65
43	38	35	34
62	61	61	58
46	40	45	45
28	26	30	34
51	50	49	46
59	45	51	54
75	78	80	71
35	35	33	40
35	34	30	34
45	42	46	31
42	52	38	41
33	40	25	37
45	39	44	40

Actual step	Experiment 4	Experiment5	Experiment6
67	62	70	67
43	40	38	37
62	69	67	68
46	51	38	50
28	23	28	22
51	49	60	45
59	54	53	65
75	65	70	70
35	31	30	32
35	38	35	40
45	40	41	55
42	45	48	40
33	25	35	36
45	42	42	50

5.1 Distortion gait:

First subject

Actual step	Experiment 1	Experiment2	Experiment3
67	72	66	65
43	33	35	36
62	55	75	69
46	36	41	38
28	20	23	19
51	61	44	45
59	52	55	51
75	66	71	62
35	49	43	26
35	31	40	30
45	39	40	37
42	34	30	33
33	27	27	25
45	43	40	41

Actual step	Experiment 4	Experiment5	Experiment6
67	70	62	67
43	46	49	52
62	61	48	53
46	34	38	55
28	21	24	35
51	45	47	39
59	48	46	50
75	69	66	77
35	30	25	32
35	31	31	28
45	33	38	41
42	40	36	40
33	45	36	41
45	37	39	43

Second subject

Actual step	Experiment 1	Experiment2	Experiment3
67	72	66	65
43	33	49	41
62	56	69	54
46	43	36	33
28	37	19	34
51	55	63	66
59	57	58	56
75	77	70	85
35	39	40	33
35	33	32	37
45	44	38	36
42	47	49	40
33	39	38	47
45	43	40	41

Actual step	Experiment 4	Experiment5	Experiment6
67	70	62	67
43	36	37	47
62	55	58	74
46	37	47	44
28	22	21	36
51	55	58	62
59	56	52	51
75	67	62	80
35	33	31	28
35	32	44	39
45	41	48	49
42	40	36	40
33	45	36	41
45	37	39	43

Subject ID	Gait type	Mean%	Mean (cm)	Step Error%	Total distance (cm)	Error on total distanc
1	Normal	5,1%	7,5	3,1%	700	2%
2	Normal	3,7%	9,4	3,5%	720	2%
3	Normal	3,3%	8,3	3,3%	690	1%
1	Distorted	6 .0%	15,6	7,2%	780	3,5%
2	Distorted	6.2%	16,2	8,1%	810	4%

References:

https://www.hindawi.com/journals/js/2017/6091261/

https://www.healthline.com/health/stride-length#:~:text=Divide%20the%20number%20of%20feet,1.25%20feet%20(15%20inches).

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5038701/

https://www.scirp.org/journal/paperinformation.aspx?paperid=57213

https://www.sciencedirect.com/topics/neuroscience/gait-analysis

https://www.researchgate.net/figure/The-double-integration-method-for-computing-the-displacements-of-the-body-centre-of_fig1_47790808

https://www.researchgate.net/figure/INS-EKF-ZUPT-algorithm-flow-chart_fig2_324724349

https://www.ceva-dsp.com/ourblog/what-is-an-imu-sensor/

https://study.com/learn/lesson/gait-definition-types-abnormality.html

https://petercorke.com/robotics/roll-pitch-yaw-angles/

https://www.euclideanspace.com/maths/algebra/realNormedAlgebra/quaternions/