INDOOR LOCALIZATION USING INERTIAL SENSORS COMMONLY AVAILABLE IN ANDROID DEVICES

NINAD S. INGALE

nsi3177@g.rit.edu

INDEPENDENT STUDY REPORT

GUIDED BY: PROFESSOR DR. LENOID REZNIK

Contents

ABSTRACT:

This paper describes an indoor navigation algorithm based on data collected from commonly available sensors accelerometer, gyroscope and magnetometer in the android devices. We have developed an algorithm which we can predict the next location of the user on the basis of their current location in the real world (X, Y) plane. The motivation behind this initiative was to build a reliable and accurate algorithm which takes sensor readings as inputs and provides us with the prediction of where (next position in real world X, Y plane) would the user be in the next instance. While GPS signals are very useful for such kind of applications, major disadvantage is limited availability of a stable GPS signal in the interiors of buildings like basement, corridors etc. IEEE Sensors Applications Symposium (SAS) [1] provided an opportunity for learning about sensors and their applications to real life scenarios in the form of a challenge data set [2]. The aim of this challenge [2] was to be able to predict the real world (X, Y) coordinates of some fixed fiducial points. The interesting part here is that we do not know how many fiducial points are there and to increase the complexity, we do not know in which order the users of the android devices have visited these fiducial points [2]. We will discuss about this challenge briefly in the following section.

CHALLENGE [2]:

Problem Statement: Use sensor measurements to estimate the x, y coordinates of the fiducial points as accurately as possible.

Given:

1. For the first fiducial point, the phone was held in a vertical position so that the Y axis was pointing up, Z-axis was pointing towards the direction user is facing.
2. Coordinates for the first fiducial point are (0,0)
3. World coordinate system: The X-axis of the world coordinate system is aligned with the Z-axis of the phone and the Y-axis of the world coordinate system is horizontal and pointing towards the right edge of the phone.

Dataset [2]:

- Dataset contains accelerometer, gyroscope and magnetometer readings from various phones given to various users.
- Units of measurement [2]:
  - Accelerometer Readings : (ax, ay, az) -> m/s^2
  - Gyroscope Readings : (gx, gy, gz) -> rad/s
  - Magnetometer Readings : (mx, my, mz) -> (micro tesla)
- Readings from each sensor were collected asynchronously, hence typically there is only readings for one sensor at a given instance. However this is not case for all instances.
- The coordinates for the fiducial points were measured using the laser range finder [2]. However the number of fiducial points is unknown to us.
- Users could follow any pattern while walking. They might have travelled to the fiducial points according their wish and not following a particular pattern.

PREVIOUS WORK:

There is a lot of research being done in the field of sensor applications to real world [3] [4] [5]. In this section we will discuss each of the previous studies done and also discuss about the algorithm we have implemented. We will also talk in depth about the problem to be solved in this [2] particular challenge and then provide our implementation for the same.

In paper [3], the approach used to handle the indoor localization problem is by using the step detection and heading detection direction and prediction modules. For this purpose they have used sensors like accelerometer, gyroscope and magnetometer. NavShoe [3] software splits the walking style of a person into two distinct phases namely stance phase and moving stride phase. When a user holding a phone are in stationary mode the NavShoe will consider the phase as stance phase otherwise it will categorize the movements of the user into moving stride phase. The NavShoe software detects the stance phase of a user and applies the zero velocity updates (ZUPT) [3] and are fed into the Extended Kalman Filter [6]. This makes the step detection algorithm more efficient. If the ZUPT is positive at the end of a stride phase, and the direction of movement was in the north direction, the EKF knows that there has been a drift in the north direction and corrects the readings and update the position of the user in the opposite direction (South). This also adjusts the velocity of the user to 0.

The next important step to determine the position accurately is to find out the angle of rotation i.e pitch and roll for the user at any particular instance. Pitch and Roll at any particular instance can be found by calculating the angle at which the axis (X,Y) is rotated around some third fixed axis (Z). In the case of gyroscope we can assume that the Z-axis is pointing downwards [3].This is because while taking the readings from gyroscope even a small change in the value of gravity will cause a large drift in the readings [3]. Hence we can find the pitch and roll i.e. angle at which X-axis and Y-axis at any particular instance is rotated around Z-axis respectively. This is calculated using arctan(x, z) and arctan(y, z) function which return us the values of the angles in radians.

In paper [4] a similar approach has been opted in order to solve the problem for indoor localization. This method also is based on the inertial sensors and also divides the complete algorithm in similar modules as mentioned in the above [3]. However this [4] paper differs from the above in the manner in which the modules are implemented. In [4] the step detection algorithm is implemented by first using low pass FIR filter with a certain cut off frequency to remove the noise from the data. Once the cut off frequency is determined and the peaks and valleys in the data are found out, the remaining noise in the form of false peaks are removed using Dynamic Time Warping (DTW) [7]. The main function of DTW is to find similarity between two distinct waveforms. The smaller value of DTW will indicate that the waveforms are similar. In this paper [4] the step detection will work as follows. The peaks will be determined at all instances, let the peak at each instance be denoted by S_i. The step detection algorithm works on the basis of the difference between S_i and S_{i+2}. If this difference is lower than the calculated threshold value then the step detected S_i will be considered as a real step.

Next step is to determine the heading direction, in order to do that [4] uses the pitch and roll angles calculated as explained above [3]. Once the pitch and roll are calculated the position of the user in future instances is calculated as below

$$x(t+1) = x(t) + (l(t) + \delta l(t)) * \cos(\Theta(t) + \delta(\Theta(t)))$$

$$y(t+1) = y(t) + (l(t) + \delta l(t)) * \sin(\Theta(t) + \delta(\Theta(t)))$$

Where,

$$x(t) = Current\ state\ vector\ ,$$

$l(t) = Estimated\ step\ length\ at\ instance\ t\ ,$

$\Theta(t) = Estimated\ inferred\ heading\ at\ instance\ t,$

$\delta(\Theta(t)) = Zero\ mean\ gaussian\ noise\ on\ heading\ direction,$

$\delta(l(t)) = Zero\ mean\ gaussian\ noise\ on\ step\ length\ estimation$

DATA PREPROCESSING STEPS:

- Feature Creation :
  - Magnitude of Acceleration: Accelerometer readings are three dimensional. Instead of considering all the coordinates, we have considered the magnitude of the acceleration as it is the most useful quantity when detecting the step. Hence we have created a feature for magnitude of acceleration. The formula used for calculating the magnitude of the acceleration is shown below

    $$a_t = \sqrt{{a_x}^2 + {a_y}^2 + {a_z}^2}$$

    $Where,$

    $a_x = X - intercept\ of\ the\ accelerometer\ reading\ at\ instant\ \ t,$

    $a_y = Y - intercept\ of\ the\ accelerometer\ reading\ at\ instant\ \ t,$

    $a_z = Z - intercept\ of\ the\ accelerometer\ reading\ at\ instant\ \ t$

  - Step length: When a person walks he/she follows a particular pattern of walking. The step length for a particular person is constant but this varies person to person. As the algorithm should be independent of the physical characteristics of the user, this feature would be very useful. Step length is calculated using below formula

    $$S_t = u_t(\delta t) + \frac{1}{2}\, a_t(\delta t)^2$$

    $Where,$

    $S_t = Step\ length\ at\ any\ instance\ t.$

    $u_t = initial\ velocity\ at\ any\ instance\ t. Considered\ 0\ for\ first\ instance,$

$a_t = Magnitude\ of\ acceleration\ at\ any\ instance\ t,$

$\delta t = Time\ difference\ between\ two\ instances.$

- Missing Values :

  As per the given input [2], the readings from sensors are collected asynchronously and hence there is typically data expected from only one of the sensors at each instant. However for the sake of representing an instant completely we have replaced the NULL values for each sensor with the previous value for that particular sensor.

- Smoothing the data:

  In order to smooth the data, we have used the difference between magnitudes of acceleration at instances as the determining factor. As mentioned [3] , the peaks in any waveform can be determined by taking the differences between the values at instances t , t-1 and t+1 i.e. instances that are separated by two units. If the wave follows following rules, we can detect a step for us

  $$N_K - N_{K-1} > 0 \ and\ N_{K+1} - N_K < 0$$

  Where,

  $$N_k = \sqrt{{a_x}^2 + {a_y}^2 + {a_z}^2},$$

  $$a_x = X - intercept\ of\ the\ accelerometer\ reading\ at\ instant\ \ t,$$

  $$a_y = Y - intercept\ of\ the\ accelerometer\ reading\ at\ instant\ \ t,$$

  $$a_z = Z - intercept\ of\ the\ accelerometer\ reading\ at\ instant\ \ t$$

APPROACH AND ALGORITHM:

As we have gained enough understanding about the data and the problem statement for this project, in this section we will proceed with the approach to find real world coordinates of the fiducial points. As we do not know how many fiducial points were there, best way to find the most accurate real world coordinates is to predict the coordinates of the next point the user will be travelling to on the basis of their current position. In order to accomplish this we will be using Kalman filter. In the following figure we will see different modules implemented.

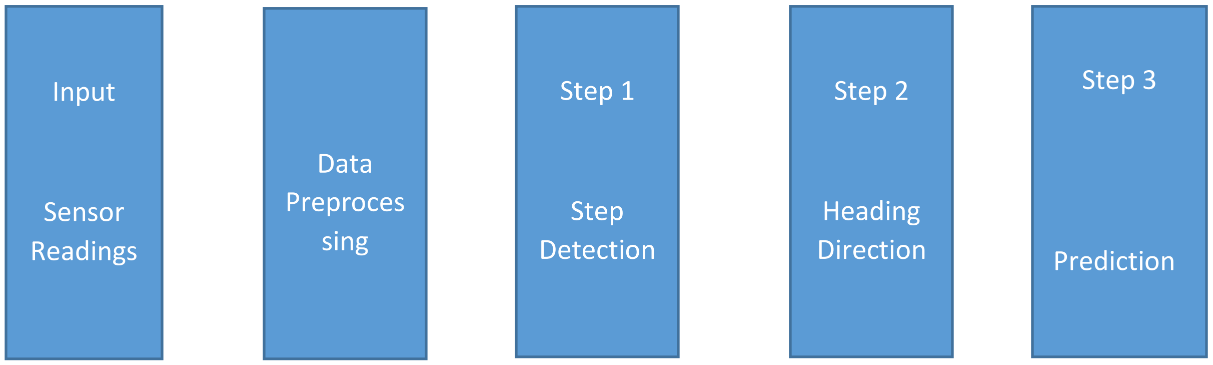

Figure 1: Block diagram for the algorithm.

Step 1: Step Detection

In this module we will be calculating the number of steps a person holding the android phone walks [5]. For this purpose we will be finding the peaks for the acceleration during the entire time the person is walking. Let n_k be the acceleration at any instant k, n_{k-1} be the acceleration at any instant k-1 and n_{k+1} be the acceleration at any instant k+1, then we will determine the peak by using following formula

$$N_K - N_{K-1} > 0 \;\; and \; N_{K+1} - N_K < 0$$

$$\text{Where, } N_k = \sqrt{}(a_x)2 + (a_y)2 + (a_z)2$$

$$a_x = \text{Accelerometer reading at instance K}$$

The above formula finds out all the peaks in the data and recognizes each peak as an individual step. This will help us filter noise records and keep only the steps [5].

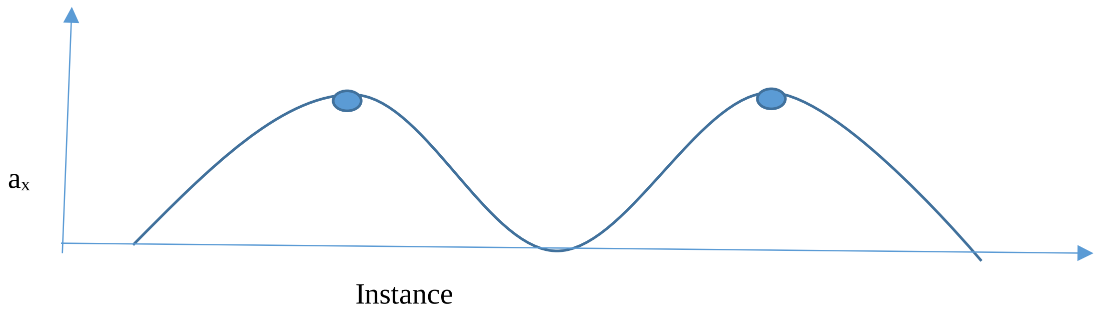

Figure 2: Detection of a step.

In the above figure we will detect the step at the instants highlighted by blue dots. This means that this method will also be used to reduce noise in the data and will increase the accuracy of the prediction using Kalman filter.

Step 2: Heading Detection

About rotational matrices [8]:

Rotation matrices are used in order to find the rotation of an axis around some other axis. In a 2 dimensional environment the rotation matrix is formed referring the figure below. In order to combine the gyroscope and magnetometer readings, we will form the rotation matrices from the gyroscope readings and then multiply the magnetometer readings with the same.

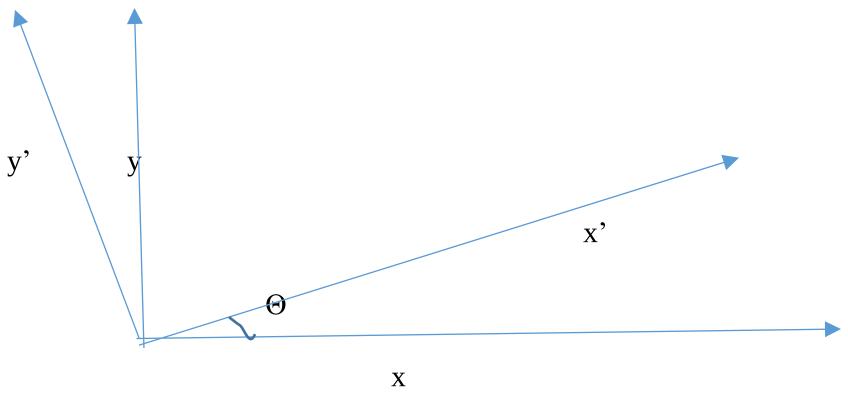

Figure 3: Rotation of axes in a two dimensional space.

From the figure above we can say the following [8]

$$x = x' \cos(\Theta) - y' \sin(\Theta)$$

$$y = \; x' \sin(\Theta) + y' \cos(\Theta)$$

In matrix form the above can be represented as

$$\begin{matrix} x \\ y \end{matrix} = \begin{matrix} \cos(\Theta) & \sin(\Theta) \\ -\sin(\Theta) & \cos(\Theta) \end{matrix} * \begin{matrix} x' \\ y' \end{matrix}$$

As per [8] we can extend this equation to 3 dimensional system by adding the identity for the constant axis (Z-axis in our case). Hence the equation for 3 dimensional rotational matrix can be written as below

$$\begin{matrix} x' \\ y' \\ z' \end{matrix} = \begin{matrix} 1 & 0 & 0 \\ 0 & \cos(\Theta) & \sin(\Theta) \\ 0 & -\sin(\Theta) & \cos(\Theta) \end{matrix} * \begin{matrix} x \\ y \\ z \end{matrix}$$

Combination of Gyroscope and Magnetometer readings:

From the gyroscope readings, we can calculate the pitch and roll, the rotation around X axis and around Y axis respectively as below

$$\Theta = atan2\left(\frac{g_x}{g_z}\right)$$

$$\beta = atan2\left(\frac{g_y}{g_z}\right)$$

Where, $\Theta =$ pitch of the device

$\beta =$ Roll of the device.

g_x = Gyroscope X axis intercept.

g_y = Gyroscope Y axis intercept.

g_z = Gyroscope Z axis intercept.

From above, we can create rotation matrices for adjusting the magnetometer readings

$$R_\Theta = \begin{matrix} 1 & 0 & 0 \\ 0 & \cos\Theta & \sin\Theta \\ 0 & -\sin\Theta & \cos\Theta \end{matrix} \qquad R_\beta = \begin{matrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{matrix}$$

Now multiplying magnetometer readings with the above two rotational matrices, we can calculate the heading as below

$$M_r = M * R_\Theta * R_\beta$$

$$\alpha = atan2\ (m_{Ry}/m_{Rz})$$

Step 3: Prediction

Using Kalman filter we can predict the next state of the device. Kalman filter formula is as below

$$X_k = X_{k-1} + (T * a_{k-1})$$

Where,

$$X_k = \begin{matrix} a_x \\ a_y \\ \alpha \end{matrix} \qquad \text{at any instance k.}$$

$$T = \begin{matrix} \cos \alpha_{k-1} \\ \sin \alpha_{k-1} \\ 0 \end{matrix}$$

$$a_{k-1} = \ Step\ length\ at\ the\ previous\ step.$$

The above formula is for the non-linear kalman filter. The reason for using nonlinear kalman filter is the time difference between the readings taken for different devices is not the same [5]. Hence it will be difficult to convert the time series to the frequency series and hence we cannot use the linear kalman filter equation for this particular task.

ABOUT IMPLEMENTATION CODE:

We have used Java for implementing the algorithm mentioned above. We have divided the implementation into two separate programming units as preprocessing and prediction steps. In the preprocessing step we have done operations on the data and we have created some new features like magnitude of acceleration, rotation angles for X-axis, Y-axis around Z-axis, also we have created the step length at each instance. For this we have used the formulae written in section for PREPROCESSING section.

In the next module which is responsible for predicting the next steps taken by the user, the programming module is again divided into the methods for reading the preprocessed data, calculating the angles of rotation and then implementing the prediction formula specified in step 3 of the algorithm for kalman filter. In this module we have used the JAMA libraries [9] for the sake of ease for multiplication with the rotational matrices.

Once we have predicted the value, the next step is to plot the predicted results. This is done using the libraries for JFree Chart [10]. By using these libraries we have plotted the original points collected in the data and then plotted the points we have predicted using our algorithm. We will discuss about the results in the next section.

RESULTS:

Following figure represents the difference between the actual points and predicted values

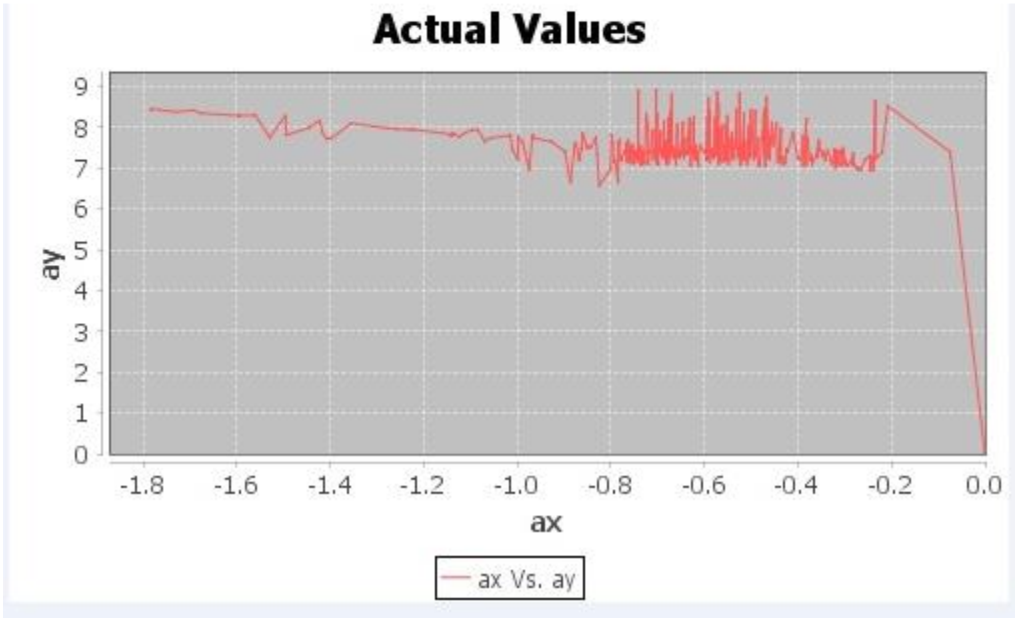

Figure 4: Actual preprocessed data.

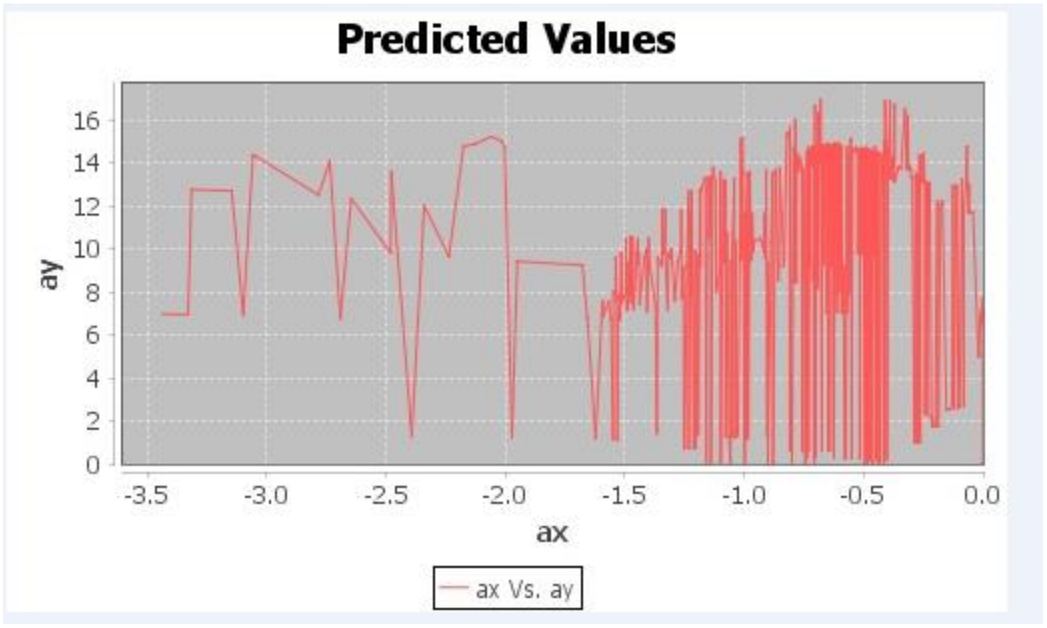

Figure 5: Predicted values using the algorithm for Kalman Filter.

It is difficult to deduce the accuracy of the model developed as we do not know the number of fiducial points for which we have to find the coordinates for. Only given fact about the fiducial points is the position of the phone when reached at the fiducial point. We know that the phone is held vertically at all fiducial points with the Y-axis of the phone pointing upwards Z-axis aligned with the direction of the user facing and X-axis aligned with that of the Z-axis. However we are unsure about whether the user has followed a particular pattern of walking or not. All these limitations make it difficult to determine the exact coordinates for the fiducial points.

Looking at the graphs above we can say that the step detection algorithm has detected the stepd properly as we can see the density of the steps detected (peaks) similar in the left regions of the graph. Also the drift in the coordinates of the accelerometer can be associated with the drift of the gyroscope and magnetometer readings. This drift can be associated with a fact that there is no calibration for the magnetometer is used when calculating the heading direction of the user.

ISSUES FACED:

As mentioned in [2], it is expected that we should typically have data for only one sensor at any instance of time. However there are many instances at which we have found the data related to more than one sensors present. Another issue was missing values for sensors, as we did not know the initial value for the sensor we had to assume the starting value of the sensor and this assumption can affect the quality of the results obtained. Next issue which we faced was absence of a test condition with which we can verify if we have reached a particular point. Due to this absence of condition it is very hard to measure the accuracy of the algorithm implemented.

FUTURE WORK:

In future we need to work on the accuracy of the prediction algorithm by taking into consideration the drifts associated with gyroscope and magnetometer. We also need to find a mechanism in order to compare accuracy of our algorithm in comparison with the actual readings of the fiducial points. We also plan to test this algorithm by collecting the data from mobile phones for various users and then run the algorithm on the each device data and all the data combined.

CONCLUSION:

After implementing this indoor localization navigation algorithm, we can say that inertial sensors of any device can be very helpful in order to predict the next location. These sensors can be used reliably and efficiently in places where we do not have wifi or GPS connectivity. However in order to get the most accurate results we need to perform preprocessing on the data like filtering/smoothing the data using

REFERENCES:

1. 2017 IEEE Sensor Applications Symposium March 13-15 , 2017 , Glassboro , New Jersey , USA

   http://2017.sensorapps.org/index.php

2. Indoor Localization challenge/dataset

   https://github.com/IndoorLocalization/datasets

3. E. Foxlin, "Pedestrian tracking with shoe-mounted inertial sensors," in *IEEE Computer Graphics and Applications*, vol. 25, no. 6, pp. 38-46, Nov.-Dec. 2005. doi: 10.1109/MCG.2005.140
4. Li, F., Zhao, C., Ding, G., Gong, J., Liu, C., & Zhao, F. (2012). A reliable and accurate indoor localization method using phone inertial sensors. Paper presented at the 421-430. doi:10.1145/2370216.2370280
5. Säll, J. & Merkel, J. (2011). Indoor Navigation Using Accelerometer and Magnetometer. (Student paper). Linköpings universitet. http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A446848&dswid=1340
6. L. Ljung, "Asymptotic behavior of the extended Kalman filter as a parameter estimator for linear systems," in *IEEE Transactions on Automatic Control*, vol. 24, no. 1, pp. 36-50, Feb 1979. doi: 10.1109/TAC.1979.1101943
7. Jablonski, B. (2012;2011;). Quaternion dynamic time warping. IEEE Transactions on Signal Processing, 60(3), 1174-1183. doi:10.1109/TSP.2011.2177832
8. Derivation of 2D Rotation Matrix
   https://engineering.purdue.edu/~bethel/rot2.pdf
9. JAMA : A Java Matrix Package
   http://math.nist.gov/javanumerics/jama/
10. JFreeChart
    http://www.jfree.org/jfreechart/