

Budapest University of Technology and Economics

Department of Telecommunications and Media Informatics

Robot position estimation using deep neural network with imu datas

Deep learning in practice based on Python and LUA

(BMEVITMAV45)

Homework

MIRA Team:

Balázs Nagy – o668s4

Márk Fekula – VUW9RS

Gergely Hunyady – FPZXIL

Budapest, 2018. 10. 14.

Table Of Content

[1. Project description 2](#_Toc527214358)

[1.1. Main goals 2](#_Toc527214359)

[2. Literature overview 3](#_Toc527214360)

[2.1. Previous work 3](#_Toc527214361)

[3. Data Collection 5](#_Toc527214362)

[3.1. IMU 6](#_Toc527214363)

[3.2. Robotino 7](#_Toc527214364)

[3.3. OptiTrack - Motion capture system 7](#_Toc527214365)

[4. Data Preparation 10](#_Toc527214366)

[4.1. Data understanding and visualization 10](#_Toc527214367)

[5. Modelling 11](#_Toc527214368)

[6. Hyperparameter tuning 12](#_Toc527214369)

[7. Summary 13](#_Toc527214370)

[7.1. Results 13](#_Toc527214371)

[7.2. Further development potentials 13](#_Toc527214372)

[8. References 14](#_Toc527214373)

Table of Figures

**Nincs ábrajegyzék-bejegyzés.**

# Project description

## Main goals

The aim of this study is to examine if it is possible to estimate a robot position from IMU (Inertial Measurement Unit) data. During our research we are using a Phidget Spatial sensor to collect data.

The concept of our research contains the following steps. In the first stage we are focusing on the orientation estimation. There are multiple algorithms that can solve this problem. As a starting baseline we are assuming this stage can be achieved with neural networks as well. The next step is to extend our model with position estimation. We are planning to divide this task into separate stages as well. First stage with position estimation is to examine the 1D case which means the IMU route is a straight line. In the next stage our model will be extended into 2D. This includes complex movements with the robot, so the neural network can learn on a complex data base.

# Literature overview

## Previous work

In a previous study two orientation estimation algorithm have been implemented on the same Phidget Spatial 3/3/3 sensor. Without the full detailed description, the following section contains the main contributions of these algorithms and some major conclusions.

The trivial algorithm to use pose and orientation estimation is Kalman filter, but this method requires the accurate system model and computationally very expensive. Since developing an embedded system on a remote agent, a computationally cheaper algorithm is needed. The first method in our scope is based on the concept of complementary filter [1]. The original algorithm combines the gyroscope and accelerometer data according to fix weights.

An own modified version of the algorithm uses the magneto data to compensate the gyroscope data.

The modified version of the algorithm is able to combine the advantages of the different sensors. The signal of the magnetometer sensor is a bit noise, but it has no offset error, since it provides absolute position. And one of the main disadvantages of the magneto sensor is its slow dynamic response. On the other hand the gyroscope dynamic can follow fast movements but always has an offset error. Figure 1 shows a comparison between the complementary filter performance and other single sensor based algorithms.

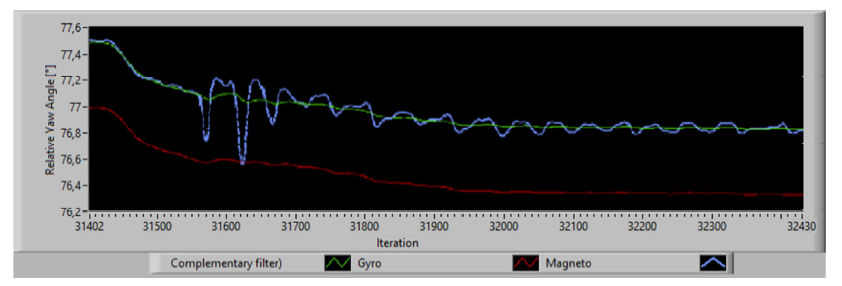


Figure 1: Complementary filter performance compared to magneto meter and gyroscope based algorithms

Another implemented and tested algorithm is a Qvaternion based method. Every rotation in a 3-dimensional space can be described by an axis and a rotation around it. We will use the syntactic of Roberto G., Ivan D. és Jizhong X [2]. The block diagram of the algorithm can be seen on Figure 2.

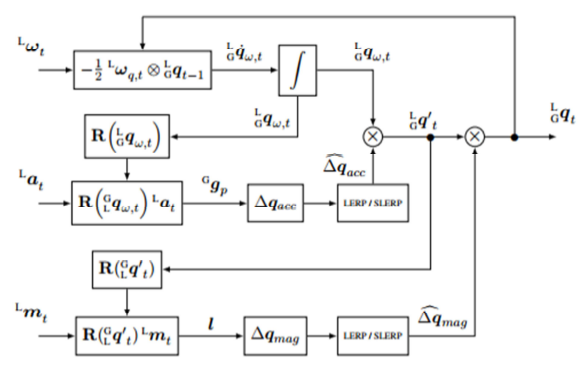


Figure 2: Qvaternion based orientation estimation algoritthm block diagram

After implementation the algorithms has been tested in the MoCap system and the

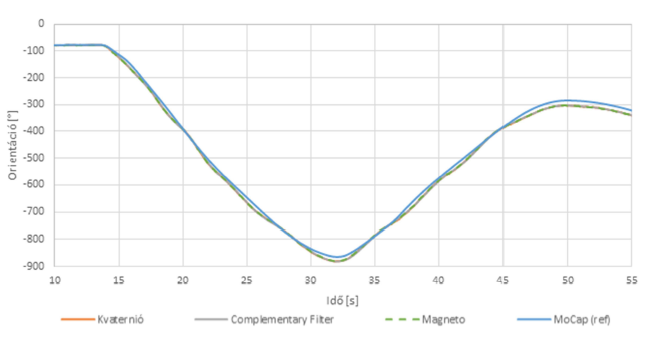


Figure 3: Orientation estimation benchmark

Summarizing the errors made by each algorithm the following table indicates numeric error values.

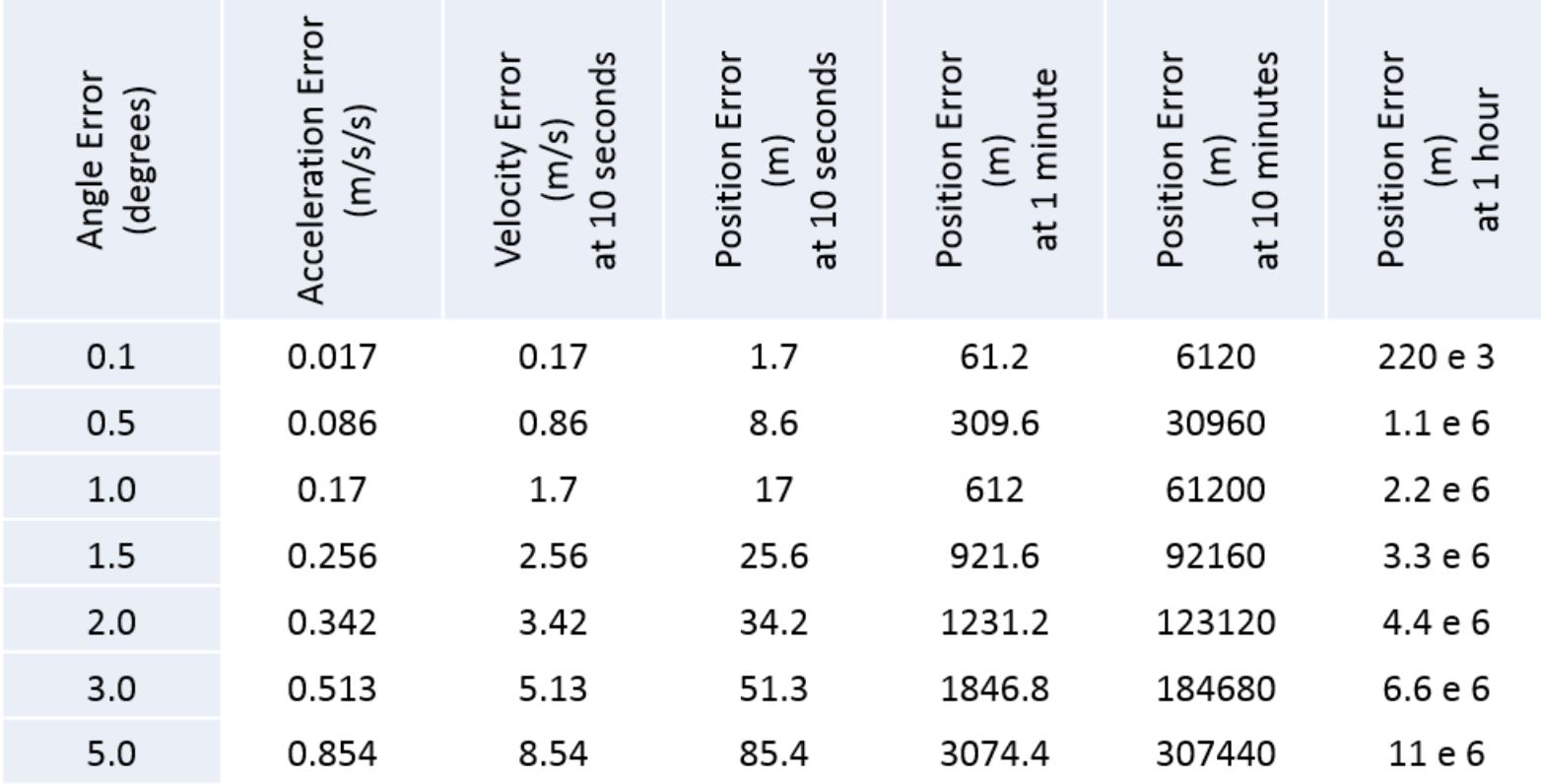
Table 1: Error comparison

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Min error[°] | Max error[°] | Average error[°] | Deviation[°] |
| Complementary | 0.01 | 20.36 | 9.38 | 7.28 |
| Qvaternion | 0.01 | 21.27 | 9.44 | 7.27 |
| Magneto | 0.02 | 21.35 | 9.51 | 7.29 |

Based on the previous study we can assume that the orientation can be estimated with explicit numeric methods. The position estimation with numeric methods without using an external observer is an unsolved task. Due to the white noise disturbance on the measured signal after two integration stage a drift will appear in the signal. Our task is to examine the possibilities of using deep neural networks in this field of research. If there is an explicit numeric solution to a problem a neural network can learn it. From this point of view a neural network can be composed to estimate the orientation of a robot. But deep neural networks can provide solutions to problems which has not an explicit solution so far.

## Related work

As it has been noticed before double integration of acceleration data won’t yield adequate results: “This isn’t because the accelerometers themselves are poor, but because the orientation of the sensor must be known with a high degree of accuracy so that gravity measurements can be distinguished from the physical acceleration of the sensor. Even small errors in the orientation estimate will produce extremely high errors in the measured acceleration, which translate into even larger errors in the velocity and position estimates.” [3]. As Figure 3 shows, even with lower orientation errors the position error after 10 seconds of moving, becomes significant.

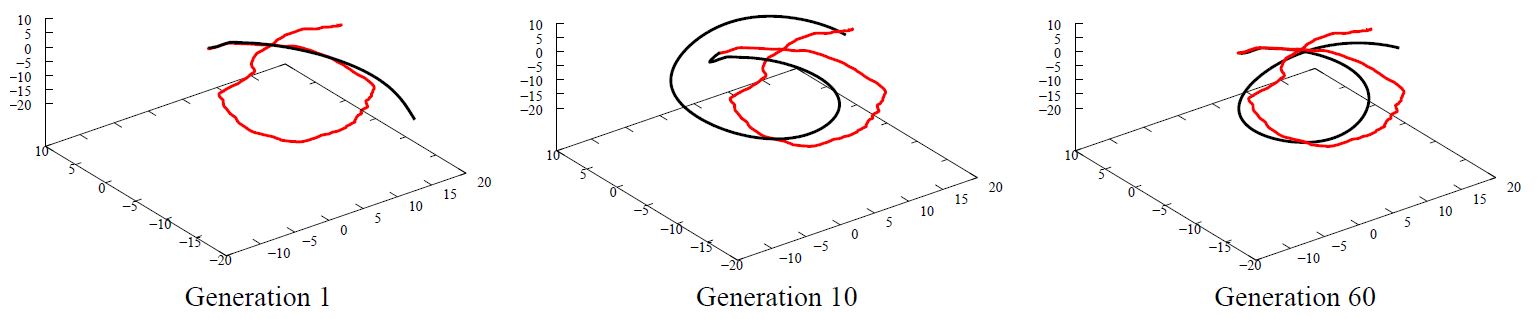


4. Figure Position errors from angle error [3]

In the paper of Maurice Shih et al. [4] the main focus is determining whether double integral, or machine learning produces smaller position error, using only accelerometer data. Provided a constant sampling time integrating twice the data is just a linear combination.

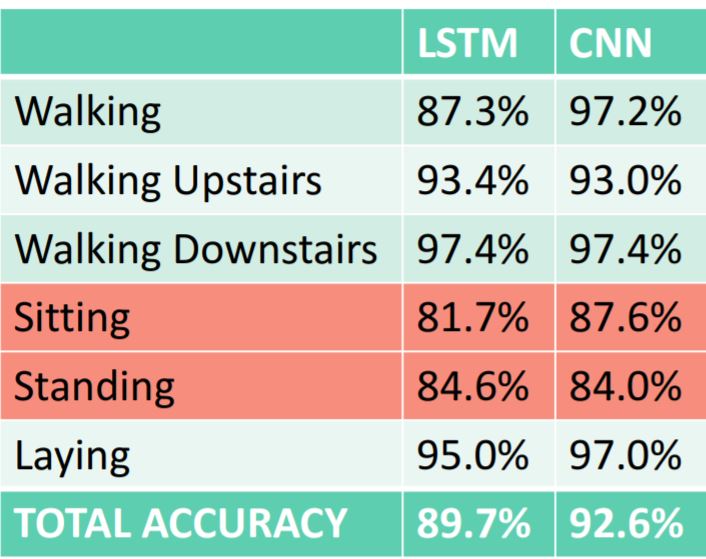
Thus, they tested if using machine learning even a viable solution for this problem. On 1000 training and 1000 test generated samples, linear combination had an error of 6.5%, while linear regression resulted only 2.5%. After proving the viability of machine learning, they tried using non-linear neural network models: Scaled Conjugate Gradient, and Bayesian Regularization. The first is a fast solution, as the second one tends to be slower but more suitable for noisy data. For this test the data was collected from a smartwatch. After fine tuning the models the average error was 13%. This was accomplished with 5 hidden layers, which implicates the complexity level of the problem with this few input parameters.

Hermann Mayer et al. [5] worked on a network, that learns a 3D trajectory based on a few samples. They used LSTM networks with Evolino, which is an evolutionary method, therefore it is better for an error surface with many local minima. The main advantage of LSTM is it has longer memory than other RNN networks, which made it suitable for this trajectory with more than 1000 points. Using a LSTM network with 10 memory cells, end evolving Evolino networks for 60 generations, a smooth and accurate trajectory was created from the training data.



5. Figure 3D plots of trajectory generated after every 20 generations [5]

Beatrix Leung [6] developed a neural network for smartphones, which classifies human activity based on accelerometer and gyroscope data. It is crucial, that while adequate accuracy is maintained, the network has to be lightweight and battery friendly. To meet these requirements, they tested LSTM and CNN based solutions. As the task mainly depends on classification, CNN proved to be more accurate, but LSTM was faster and more power efficient, while not lagging behind so much.



6. Figure Network accuracy for different classes

# Data Collection

Collecting data is an essential part in deep learning era. In order to make enough measurement data we built up our own measurement setup which consist of three main parts. The client program can connect to the Motion Capture system to track and log the position and orientation of the sensor mounted on the robot. It also connect to the robot via wifi to control the movement of the robot. And the program connects the IMU (Inertial Measurement Unit) via USB to collect and log data from it. The sematic of MIRA (MoCap, IMU and Robotino Analyser) can be seen on Figure 3.

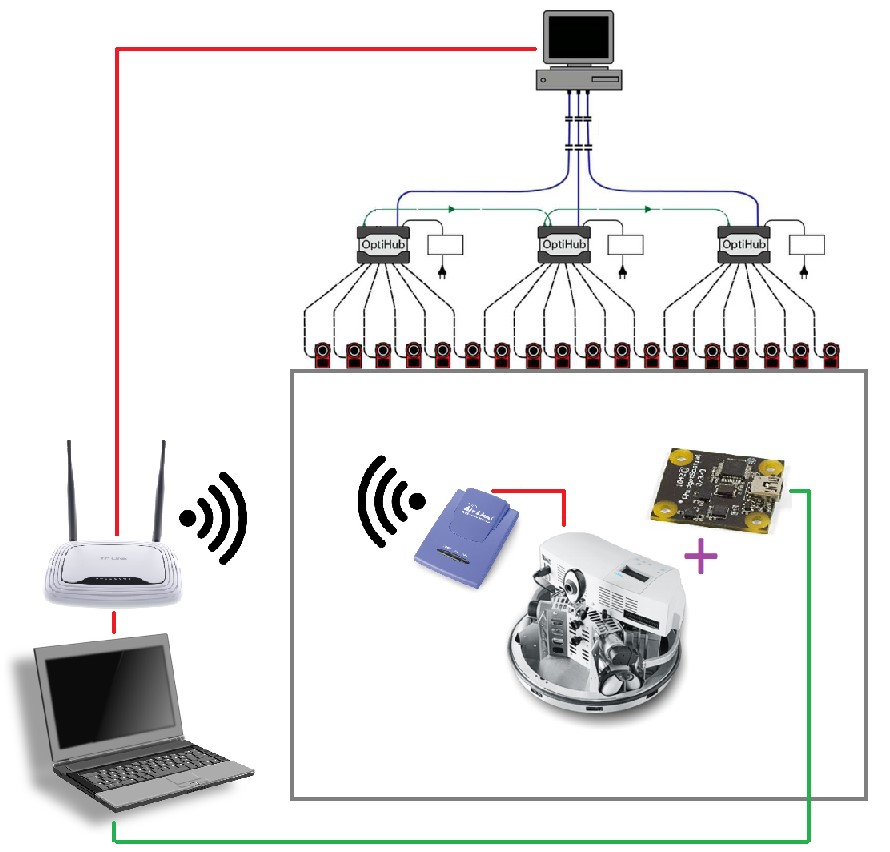


Figure 7: Sematic of MIRA

The following sections contain a detailed description about the three main parts of MIRA.

## IMU

A PhidgetSpatial 3/3/3 sensor, shown in Figure 4, had been used during our study.

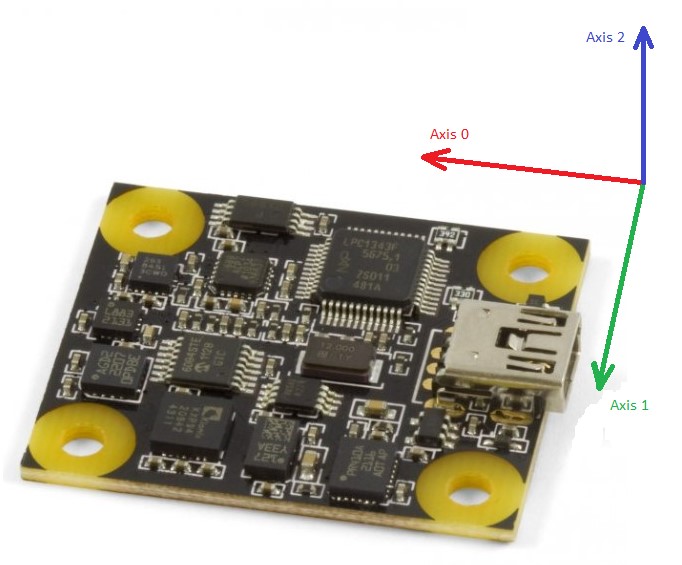


Figure 8: Phidget Spatial 3/3/3 sensor with axis representation

The sensor contains two 3-axis accelerometers, two 3-axis gyroscopes and a 3-axis magnetometer. The exact parameters of the sensor can be found on the official website of the sensor [3]. The sensor is mounted on a carrier robot and plugged to the notebook via USB cable. The USB cable connection is necessary because the robot platform is not compatible with and it is the most secure way to collect data from it.

The data structure from the inertial measurement unit (IMU) is the following:

* acc0 – acceleration along the X axis [g]
* acc1 – acceleration along the Y axis [g]
* acc2 – acceleration along the Z axis [g]
* gyro0 – gyroscope speed around the X axis [°/s]
* gyro1 – gyroscope speed around the Y axis [°/s]
* gyro2 – gyroscope speed around the Z axis [°/s]
* mag0 – magnetic field strength along the X axis [G]
* mag1 – magnetic field strength along the Y axis [G]
* mag2 – magnetic field strength along the Z axis [G]

The different built in sensors have a different sampling speeds. The maximum sampling speed of the accelerometer and the gyroscope is 4 ms/sample, while the maximum sampling speed of the magnetometer is 8 ms/sample. The sampling time can be modified and in case the sampling speed is faster than the magnetometer a zero-order hold filter is applied on the magneto data. In this way we eliminate the harmful effect of the corrupt measurement points.

## Robotino

However, the aim of this study to build up an algorithm which only uses data from the IMU, so it can be platform independent we used a Robotino during our project. Mounting the sensor on a real robot provides real data sets and in this way our algorithm will learn on a real-life use case scenario.

The Robotino -shown on Figure 5- is a small robot with omnidirectional driving system made by FESTO. The robot has its own programming language, but it is not compatible with other languages, so we had to write an API in LabVIEW. With the help of the LabVIEW code we gain access to every data produced by the Robotino, furthermore LabVIEW makes it possible to connect the robot to the Motion Capture system.



Figure 9: Robotino

## OptiTrack - Motion capture system

OptiTrack Motion capture systems are capable of tracking retroreflective markers. The system contains 18 cameras mounted in a room. The official measurement and calibration software for the system is Motive. After connecting the cameras and calibrated the work space with the Motive everything in place to track the designated target. In our project the IMU sensor is the main field of interest but in order to examine it in a real-life situation a carrier robot is needed. First of all, markers were attached to the sensor so the MoCap can track the position and orientation of the IMU. The special sensor holder with markers can be seen on Figure 6.

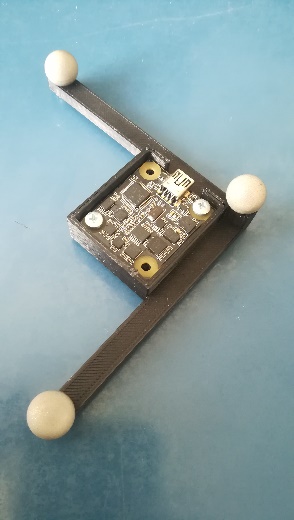


Figure 10: IMU with markers

After the markers were set we mounted the sensor on a Robotino. The robot with the sensor can be seen on Figure 7.



Figure 11: IMU mounted on the Robotino

After the successful setup and system calibration the robot can move in the measurement space.

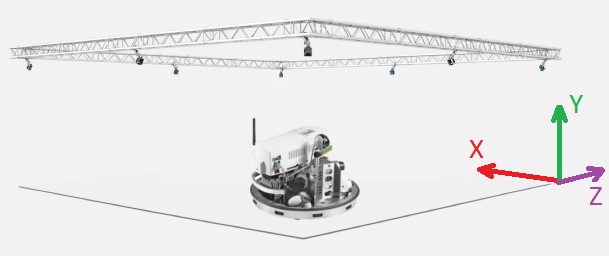


Figure 12: MoCap illustration with axis to better understand the data stream

As shown on the Figure 8 the axis of the MoCap system is not the usual. The Y axis faces up and the robot can move in the X-Z plane. Later on, we have to keep it in mind. The position in the MoCap space is absolute and depend on only the calibration of the system but the orientation of an object is not. It means that every object orientation reference is bound to the original pose when its definition got into the system. Furthermore, the MoCap system provides Roll-Pitch-Yaw representations which is converted into a single orientation along with the X-Z plane.

# Data Preparation

## Data understanding and visualization

A logging session during a measurement produces three files. MIRA outputs a file including the IMU data, a file including the MoCap data and a file including the robot data. Figure 10 shows the MoCap X-Z plane and the routs of the different measurement files.

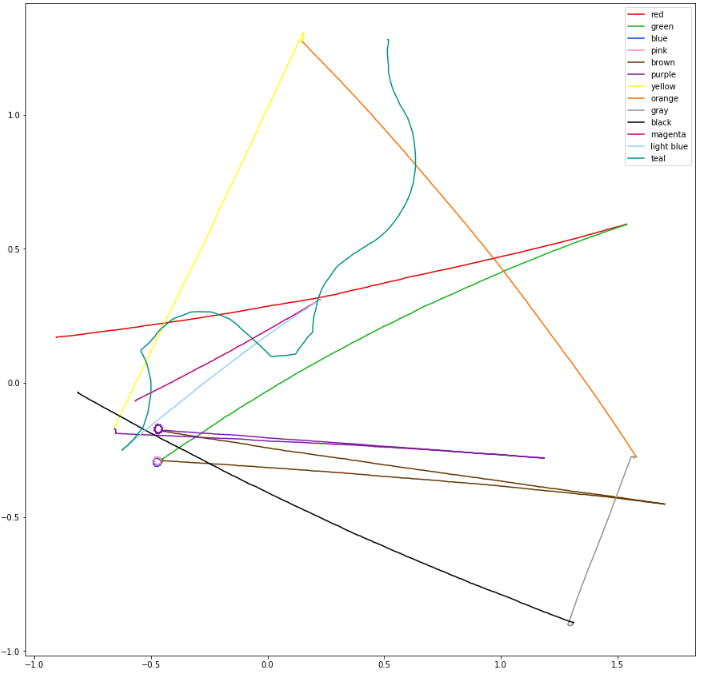


Figure 13: Data sets MoCap plots

Each measurement file contains a specific rout identified with a colour. After choosing a specific rout – for example purple – the detailed measurement data can be plotted. Figure 11 shows the 3 different sensor data plot (accelerometer, gyroscope, magnetometer) in connection with the purple path.

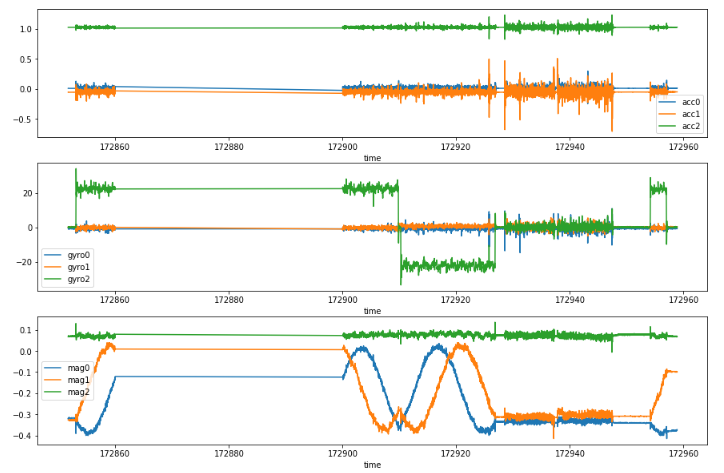


Figure 14: IMU data plot (rout: purple)

# Modelling

# Hyperparameter tuning

# Summary

## Results

## Further development potentials

# References

|  |  |
| --- | --- |
| [1] | P.-J. V. d. Maele, “Pieter-Jan.com,” 2013. [Online]. Available: http://www.pieter-jan.com/node/11. [Accessed 13 10 2018]. |
| [2] | V. RG., D. I. and X. J., “Keeping a Good Attitude: A Quaternion-Based Orientation Filter for IMUs and MARGs,” *Sensors,* pp. 19302-19330, 6. August 2015. |
| [3] | M. Shih és J.-T. Hsieh, „Predicting Short-Range Displacements From Sensor Data,” 2015. |
| [4] | P. Inc., “Phidgets,” 2017. [Online]. Available: https://www.phidgets.com/?&prodid=32. [Accessed 11 10 2018]. |