

Budapest University of Technology and Economics

Department of Telecommunications and Media Informatics

Robot position estimation using deep neural network with imu data

Deep learning in practice based on Python and LUA

(BMEVITMAV45)

Homework

MIRA Team:

Balázs Nagy – o668s4

Márk Fekula – VUW9RS

Gergely Hunyady – FPZXIL

Budapest, 2018. 12. 07.

Table Of Content

[1. Project description 2](#_Toc531958044)

[2. Literature overview 2](#_Toc531958045)

[3. Data Understanding 4](#_Toc531958046)

[3.1. Data collection method 4](#_Toc531958047)

[3.2. Data visualization 5](#_Toc531958048)

[3.3. Data preparation for deep learning models 7](#_Toc531958049)

[4. Experiments with Different DNN Modells 7](#_Toc531958050)

[4.1. LSTM 7](#_Toc531958051)

[4.2. 1D CNN 7](#_Toc531958052)

[5. Summary 7](#_Toc531958053)

[5.1. Results 7](#_Toc531958054)

[5.2. Further development potential 7](#_Toc531958055)

[6. References 7](#_Toc531958056)

Table of Figures

**No table of figures entries found.**

# Project description

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.

Our research efforts contain the following steps that are documented in the further chapters. You can read a short review on our survey connected to the relevant literature in Chapter 2. As a next step we collected accurate position and orientation data using a special motion capture system presented in Chapter 3. Working with this data we tried to apply different supervised machine learning techniques (namely Long short-term memory networks - LSTM and 1D Convolutional Neural Networks - CNN) to estimate the absolute or relative position and the orientation of the robot. Our findings and experimentations on these methods can be found in Chapter 4. Finally in Chapter 5 we discuss the achieved results and the further development potential.

# Literature overview

In a previous study two orientation estimation algorithm have been implemented on the same Phidget Spatial sensor. The trivial algorithm to use for this problem could be Kalman filter, however this method requires the accurate system model and is computationally too expensive to use in an embedded system. The first method in our scope is based on the concept of complementary filters [1]. The algorithm combines the gyroscope and accelerometer data according to fix weights, and the magneto data is used to compensate the offset error in the gyroscope data. **Hiba! A hivatkozási forrás nem található.** shows a performance comparison between the complementary filter approach and other algorithms based on a single sensor.

Another algorithm is a Qvaternion based method which was presented by Valenti et al. [2]., where every rotation in a 3-dimensional space can be described by an axis and a rotation around it. After implementation, the algorithms have been tested in the MoCap system and the errors achieved by them are shown in Table Table 1.

**Table 1: Error comparison**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Min err[°]** | **Max err[°]** | **Avg error[°]** | **Dev [°]** |
| **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 |

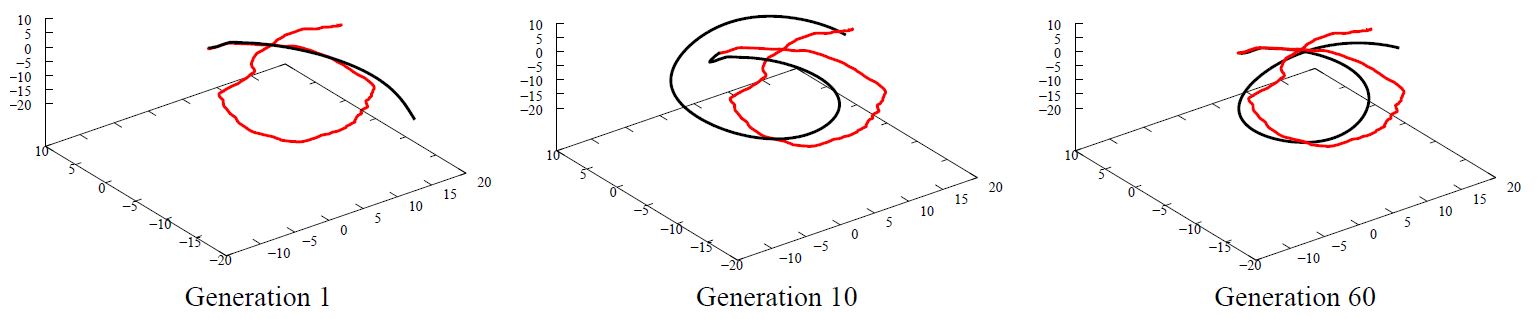
Based on the previous study we assume that the orientation can be estimated with explicit numeric methods. Yet the position estimation with numeric methods without using an external observer is an unsolved task. Due to the white noise disturbance on the measured signal after two integration stage, a drift will appear in the signal. If there is an explicit numeric solution to a problem, a neural network could learn it. But deep neural networks can also provide solutions to problems which has not an explicit solution so far.

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].

In the paper of Shih et al. [4] the main focus is determining whether double integral or machine learning produces smaller position error, using only accelerometer data. With a constant sampling time - , the integration simplifies to a linear combination.

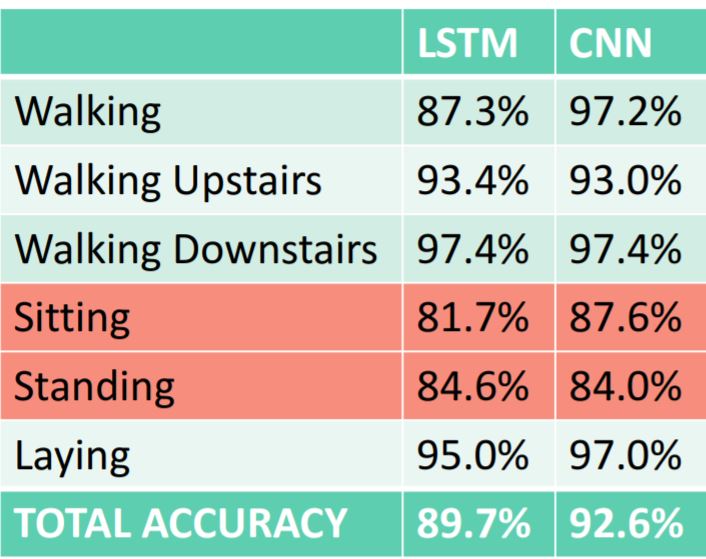
Thus, they tested if using machine learning is even a viable solution for this problem or not. On 1000 training and 1000 test generated samples, linear combination had an error of 6.5%, while linear regression resulted only 2.5%. After that, they tried using neural networks with Scaled Conjugate Gradient, and Bayesian Regularization to get a robust model. 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. Using an LSTM network with 10 memory cells, and evolving Evolino networks for 60 generations, a smooth and accurate trajectory was created from the training data.



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

Leung et al. [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 as well. 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. The results of this research project are shown in Figure 2.



2. Figure Network accuracy for different classes [6]

# Data Understanding

## Data collection method

In order to collect measurement data we built up our own measurement setup which consist of three main parts. Our client program - realized in LabVIEW - can connect to the **Motion Capture system** to track and log the position and orientation of the sensor mounted on a robot. It also connects to the **robot** via wifi to control its movement. And the program connects to an **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. A Phidget Spatial sensor had been used during our study, which provides acceleration [g], gyroscope speed [°/s] and magnetic field strength [G] data along 3 axes. The exact parameters of the sensor can be found on the official website of the sensor [7]. The Robotino - shown on the middle of Figure 3 - is a small robot with omnidirectional driving system made by FESTO.

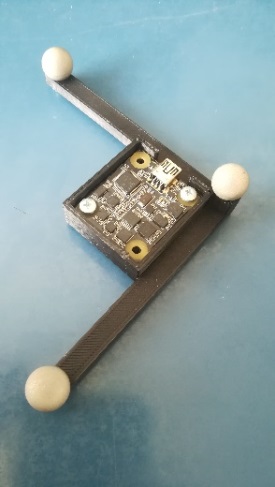
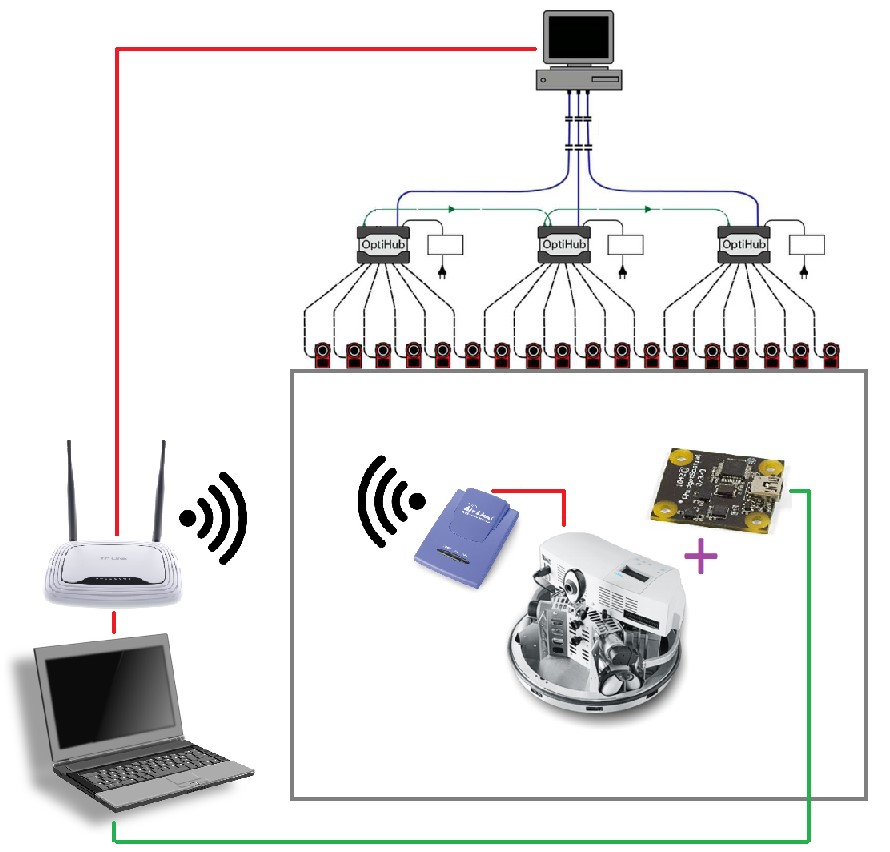


Figure 3: Sematic of MIRA

The OptiTrack Motion capture system is capable of tracking retroreflective markers. The system contains 18 cameras mounted in a room. First, markers – shown on the right side of Figure 3 - were attached to the sensor so the MoCap can track the position and orientation of the IMU. After that, we mounted the sensor on the top of a Robotino. After the successful setup and system calibration the robot can move in the measurement space shown on Figure 4.

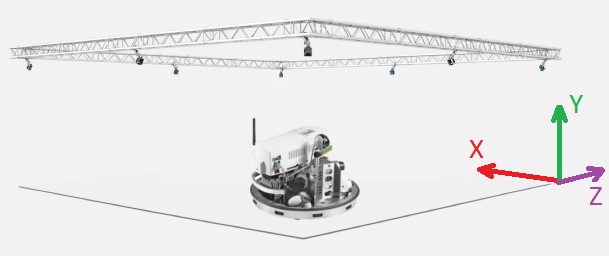


Figure 4: MoCap illustration with axis to better understand the data stream

## Data 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. We needed multiple attempts to finally record error-free data, because synchronizing the IMU and the MoCap turned out to be a difficult task. Figure 5 shows the MoCap X-Z plane and the routes of the robot in the different measurement files.

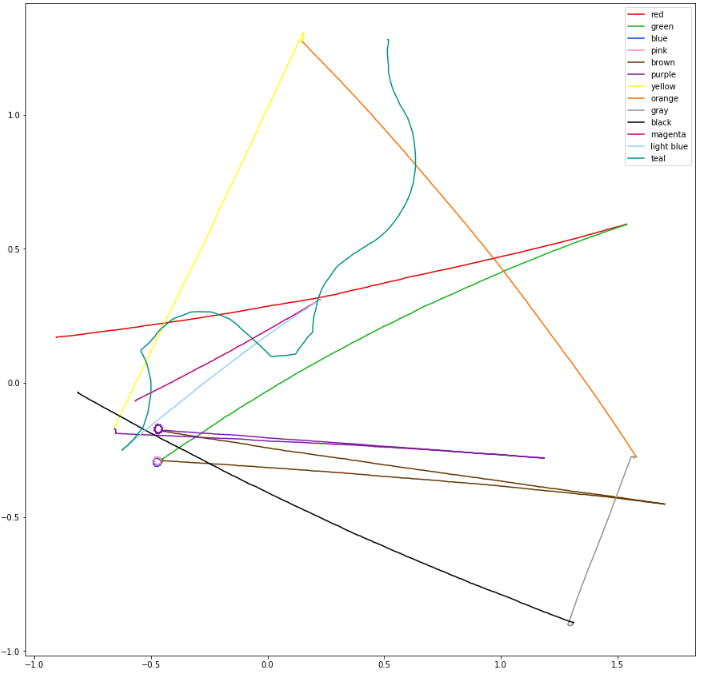


Figure 5: Data sets MoCap plots

Each measurement file contains a specific route identified with a colour. After choosing a specific route – for example purple – the detailed measurement data can be plotted. Figure 6 shows the 3 different sensor data plot (accelerometer, gyroscope, magnetometer) in connection with the purple path.

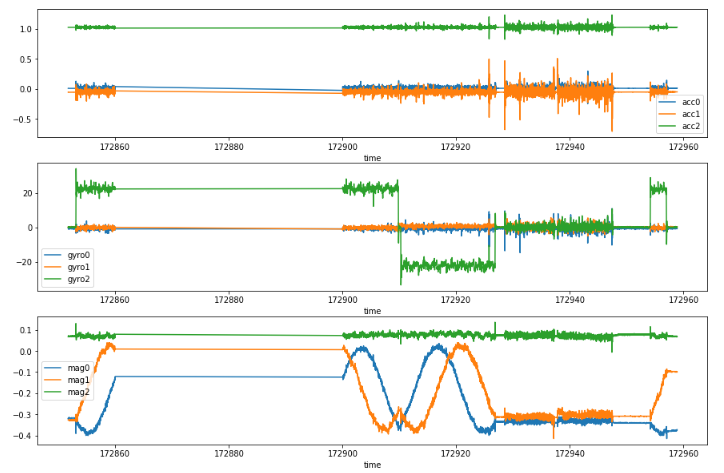


Figure 6: IMU data plot (rout: purple)

## Data preparation for deep learning models

# Experiments with Different DNN Modells

## LSTM

## 1D CNN

# Summary

## Results

## Further development potential

# 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] | "Using Accelerometers to Estimate Position and Velocity," [Online]. Available: http://www.chrobotics.com/library/accel-position-velocity. [Accessed 14. 10. 2018.]. |
| [4] | M. Shih and J.-T. Hsieh, "Predicting Short-Range Displacements From Sensor Data," 2015. |
| [5] | H. Mayer, F. Gomez, D. Wierstra, I. Nagy, A. Knoll és J. Schmidhuber, „A System for Robotic Heart Surgery that Learns to Tie Knots Using Recurrent Neural Networks,” in *International Conference on Intelligent Robotics and Systems*, 2006. |
| [6] | B. Leung, „Optimising Human Activity Classification using Deep Learning for Smart Devices,” [Online]. Available: http://www.cse.unsw.edu.au/~z5014700/docs/TOR.pdf. [Hozzáférés dátuma: 14. 10. 2018.]. |
| [7] | P. Inc., “Phidgets,” 2017. [Online]. Available: https://www.phidgets.com/?&prodid=32. [Accessed 11 10 2018]. |