Sensor Based Motion Tracking and Recognition in Martial Arts Training

Stephan Agojo

Supervisor: Ing. Vratislav Harabiš, Ph.D.

Consultant: FH-Prof. DI Dr. Lars Mehnen

Abstract

In various martial arts, competitors are interested in quantifying and categorising techniques which are exercised during training. The implementation of embedded systems into training gear, especially a portable wireless body worn system, based on inertial sensors, facilitates the quantification and categorisation of forces and accelerations involved during the training of martial arts.

The scope of this paper is to give a brief overview of contemporary technology and devices, describe key methods that are implemented in such devices and as well to realise a motion tracking device

For this reason a literature research was carried out using search engines and databases.

The search yielded that devices are capable of tracking activities of daily life; however only one product was found which can track and recognise techniques applied in martial arts training just for the upper body.

The realisation of an inertial sensor based tracking and offline recognition system which can record and classify different exercises is possible with the method and technology presented.

Even so, different challenges such as implementing an algorithm for online classification need to be overcome. Therefore, this development has to be viewed as an iterative process.

Acknowledgement

I would like to thank Ing. Vratislav Harabiš, Ph.D. for his splendid supervision and support through very useful comments during the development of this paper which as well applies to FH-Prof. DI Dr. Lars Mehnen. I would like to further thank doc. Ing. Jana Kolářová, Ph.D. and Ing. Jiří Sekora and furthermore thank the Department of Biomedical Engineering at the Faculty of Electrical Engineering and Communication at Brno University of Technology

Table of Content

1 Introduction 4
Motion Tracking Technology a Brief Overview 5
Non-Visual Tracking Systems 5
Visual Tracking Systems 7
Currently available inertial sensor based systems 8
Motion Recognition and Analysis 10

2 Methods 11
Discrete Kalman Filter Algorithm 11
Motion Recognition Algorithm 12

3 Material and Realisation 15
Description 15
Connecting the MPU6050 and Raspberry Pi 3 17
Setting up the Raspberry Pi 3 19
Testing the sensors 22
Programming the Raspberry Pi 3 23
Attaching the sensors 25
Recording motion 26
Transferring data from RPi3 to PC with WinSCP 26
Processing data with MATLAB 27

4 Results 28
Data visualization 28
Post-processing 29
Feature extraction 30
Training HMMs 31
Classifying test data 32

5 Discussion 34

Bibliography 37
List of Figures 39
List of Tables 40
Appendix 42

1 Introduction

In professional sports, the performance of human movement can be enhanced by supervising the practitioner's technique by his coach. These respective professional instructions are most useful in sports or activities where the correct execution of the technique is the dominant factor rather than physical structure or physiological capacity, especially in the practice of martial arts. [1]

Sports practitioners strive to improve their performance. The use of embedded systems has increased in recent years and with it, the options to provide improved coaching, more accurate monitoring of the participant and to enhance training experience. The integration of an electronic measurement system into training gear for martial artists can be adapted to various functions such as observing the trainee's vital data, determining the number, force and velocity of the applied technique and as well recognizing it for example whether it is a straight punch, an uppercut or a roundhouse kick and measuring reaction time.

The aim of this diploma thesis is to 1) give a brief overview on motion tracking technology; 2) survey the current market situation on sensor based motion tracking; 3) describe a method for motion recognition and analysis; 4) describe key method used for the development and 5) realize a portable system based on IMUs for punch recognition.

In order to track movement, sensors, preferably inertial measurement units (IMU), which can be accelerometer and gyroscopes or magnetometer, sampled at an adequate rate which accurately measure acceleration and angular velocity are required. Figure 1 depicts a block diagram for a body worn sensor network, which will be modified into an inertial sensor based motion tracking system. The modified system shall be able to recognise pre-recorded motion.

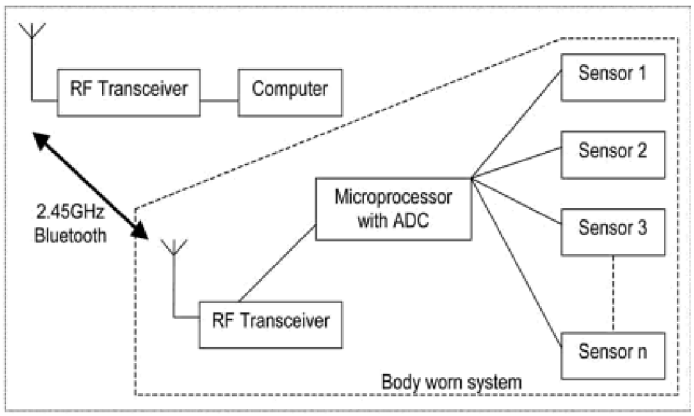


Figure 1: Block diagram of a body worn sensor network [2], which will be modified into an inertial sensor based motion tracking system. The modified system shall be able to recognise pre-recorded motion.

Motion Tracking Technology a Brief Overview

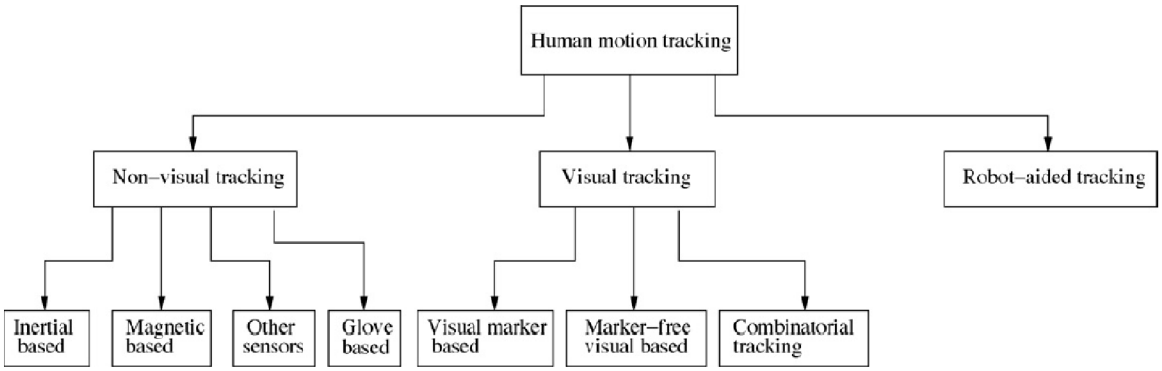


Figure 2: Classification of motion tracking with different sensor technologies [3]

Basically a tracking system can be non-visual, visual or robot-aided. Figure 2 depicts a classification based on available sensor technology. The different approaches to motion tracking will be explained in the following section, however magnetic based, other sensors, glove based and robot- aided tracking will not be further discussed.

Non-Visual Tracking Systems

This system utilises sensors, placed as close as possible to the joints of the human body, in order to retrieve information about movement. Commonly used sensors are inertial, acoustic, mechanical, magnetic, radio or microwave based. Some of them have such a precision that they are capable of detecting small changes in amplitude, e.g. finger or toe movement. Depending on modality-specific, measurement-specific and circumstance-specific limitations each type of sensor, which has its own advantages and drawbacks are affected by different environments. [3]

Inertial based sensors such as accelerometers and gyroscopes are widely spread amongst user electronics (e.g. smart phone for detection of device's orientation). They are an easy to use and cost efficient way to detect full-body human motion. The data output of these sensors can be wirelessly transmitted to a computer for further process, visualisation or analysis. These types of sensors can be very small in size, have a high sensitivity and a large measurement range. Drawbacks are that baseline wander and measurement noise can result in integration drift. [3]

- **MEMS accelerometer**

A moving beam structure composed of two sets of fingers is the main component of a MEMS accelerometer that measures acceleration in g ($1g = 9.81m/s^2$). One set of fingers is fixed to a solid plane on a substrate, while the other is attached to a known mass mounted on a spring that can move depending on acceleration. The capacitance between the fixed and moving beam fingers changes when the sensor is accelerated. The basic structure of a MEMS accelerometer can be seen in figure 3. [4]

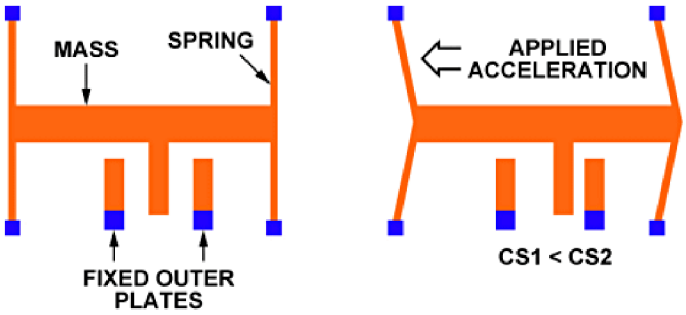


Figure 3: Structure of a MEMS accelerometer [4]

- **MEMS gyroscope**

A gyroscope measures angular rate in $\frac{rad}{s}$ ($1rad/s = \frac{360°}{2\pi} = \frac{180°}{\pi} \approx 57{,}296°/s$) through the Coriolis Effect. A constantly oscillating mass driven in a particular direction experiences a displacement when an external angular rate is applied. This causes a change in capacitance similar as in an accelerometer. The structure of a MEMS gyroscope is illustrated in figure 4. [5]

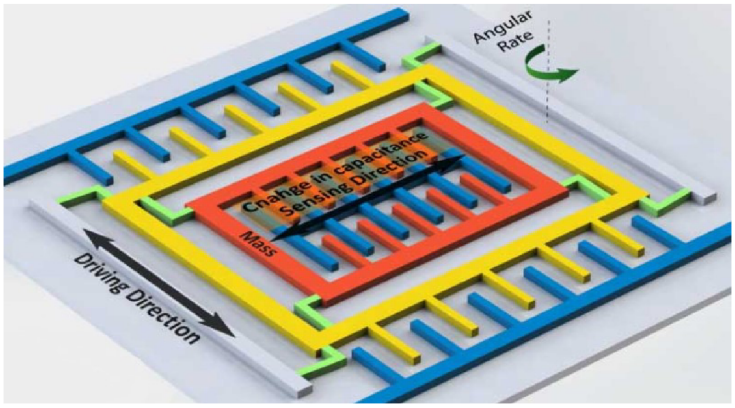


Figure 4: Structure of a MEMS gyroscope [5]

Visual Tracking Systems

Several optical sensors (i.e. cameras) are typically used to improve accuracy in position estimation. A visual tracking system is classified as visual marker based, if indicators of reference points need to be attached to the body, else it is classified as marker free.

- **Visual Marker Based**

This technique applies multiple cameras that are used to capture human movement, with indicators placed over the human body. In human motion analysis, visual marker based tracking systems, such as Optitrack or VICON (fig. 5), which provide low errors around 1mm, hence accurate position information, are often referred to as “golden standard”.

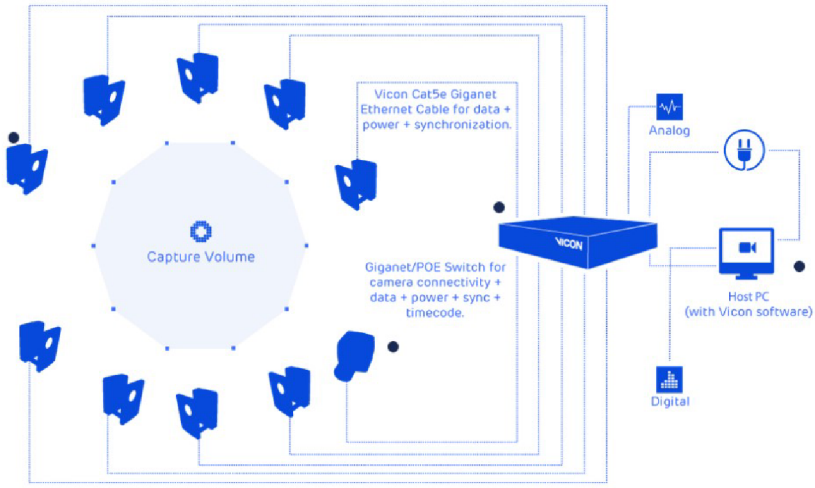


Figure 5: Schematic of a VICON system [6]

It is possible to further distinguish this type of system into passive, active or hybrid, which is a combination of both. Passive marker systems consist of up to 16 cameras, each emitting infrared light. The infrared light is then reflected by the marker and picked up by the camera. Combining 2-D data from the cameras, the system calculates a 3-D position.

In an active marker system, the marker itself emits infrared light, which is detected by the cameras. This type of setup is used when background lighting varies or is unpredictable.

A system using optical sensors with markers have one major drawback, that in case of overlapping body parts or rotated joints, it might not be capable of detecting the movement correctly. This could happen as well in an environment (e.g. home) with many disturbing objects.

- **Marker-free Visual Based**

A marker-free visual based system requires a high resolution camera of a million pixels and a sampling rate above 60 Hz. This method is typically used in surveillance application and as well by Microsoft's Kinect, an interactive input device for a gaming console. The use of this approach is motivated by issues occurring with visual marker based systems: possible unreliable identification of markers, markers placed on soft tissue can move, resulting in noisy data, marker can wobble due to inertia and markers might separate. Intensive computational effort is required to retrieve 3-D localisation and error reduction.

- **Combinatorial Tracking**

Combinatorial tracking combines the advantages of both methods mentioned before in order to reduce the errors, which arise when using each technique separately. If markers on body parts are not within the field of view, boundaries these human body parts can be captured in a motion trajectory. Drawback to this approach is that it requires intensive computation and calibration.

Currently available inertial sensor based systems

Manufacturer	Xsens	Xsens	APDM	Inertial Labs
Product	Awinda	Link	Opal	3D suit
Tracker	17	17	7-24	4-17
Axes	3	3	3	3
Accelerometer	±16 g	±16 g	±16 g, ±200 g	±2 g, ±16 g
Magnetometer	±1.9 Gauss	±1.9 Gauss	±8 Gauss	-
Gyroscope	±2000 deg/s	±2000 deg/s	±2000 deg/s	±1200 deg/s, ±2000 deg/s
Static accuracy (Roll / Pitch)	0.2 deg	0.2 deg	1.15 deg	1 deg
Static accuracy (heading)	0.5 deg	0.5 deg	1.5 deg	1 deg
Dynamic accuracy	1 deg	1 deg	2.8 deg	1 deg
Internal update rate	1000 Hz	1000 Hz	20-200 Hz	1000 Hz
Output rate	60 Hz	240 Hz	50 Hz	60 Hz
Reference	[7]	[7]	[8]	[9]

Table 1: Overview of products with their properties which are available on the market

Each product listed in table 1 comes with proprietary software for real-time motion capturing and visualisation on a 3-D virtual model. The applications for these types of system range from motion capture, for computer generated images, user interaction in a virtual reality to training systems for rehabilitation purposes.

In some systems the number of used tracker can vary, depending on the tracking purpose. For example 4 trackers, each placed on the end of an extremity can coarsely track activities performed by the whole body whereas 7 trackers, where 3 trackers are placed on each segment of each leg and 1 tracker is placed on the trunk can provide more detailed information on activities only performed by lower extremities, e.g. gait analysis. However, information on pricing can just be retrieved upon a written request at the manufacturer.

Elliott Fight Dynamics LLC developed boxing gloves called striketec which are able to track a training session. The system measures translational and rotational acceleration of the fist. The device will be able to measure the punch speed, punch force, punch type and punch count (fig. 6). It will be available for purchase from February 2017. [10][11]

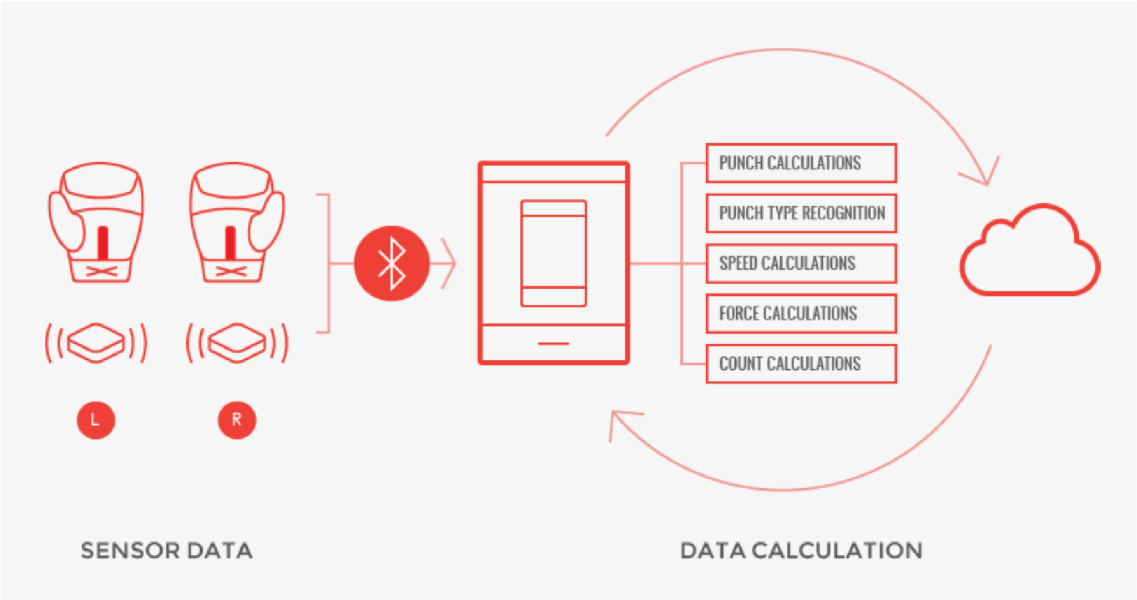


Figure 6: Illustration how striketec works. The sensors embedded in the gloves send their data via Bluetooth to a smart device. All calculations are done on the smart device. Information can be stored and retrieved from the cloud. [11]

Motion Recognition and Analysis

A general-purpose frame work for human motion recognition include data acquisition from sensors, signal pre-processing, signal segmentation, feature extraction and training a classification model, that is summarized as a so-called activity recognition chain (ARC) shown in figure 7.

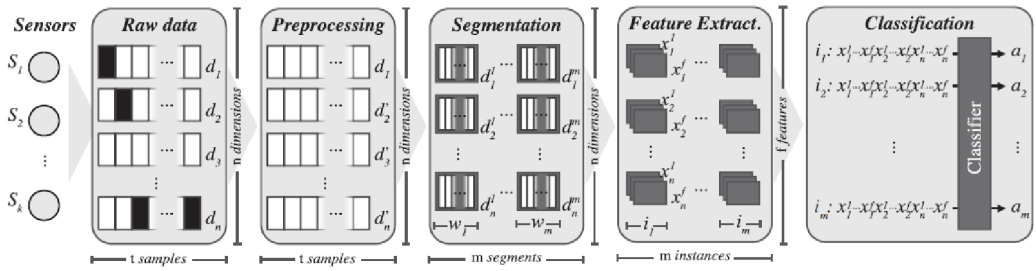


Figure 7: Steps of an activity recognition chain (ARC) from data acquisition to classification [12]

Data acquisition in motion recognition typically involves the design and development of a method to gather data about physical activities performed by the subject. The use of different sensors placed on human body segments is a frequent approach.

Performing data collection in a real-world scenario can result in, that the raw data contain noise, invalid data or missing samples. Therefore, it is crucial to perform signal pre-processing before classification is carried out. After pre-processing signal segmentation can be done with sliding windows. These segments can be transformed into frequency domain to capture temporal information.

Features, such as mean acceleration, can be extracted from these transformed segments. The final stage in the ARC is to train a classification model. Different methods for classifiers are briefly described in the following.

Distance-based methods assume that the data is based on geometric properties and have some kind of similarity and relation. The Euclidean distance and k-nearest neighbour algorithm are commonly used for classification and regression in machine learning algorithms.

Statistical methods assume that data follow a probabilistic function that need to be inferred. An example is the Gaussian naive Bayes-Classifier, which assumes that continuous values related to each class have a Gaussian distribution. [13]

Kernel methods perform pattern analysis based on a similarity function over pairs of data points. Stochastic gradient descent is a technique for discriminative learning of linear classifiers under convex loss functions, such as support vector machines used for classification, regression and outlier detection. A major drawback of this approach is that it requires a large number of parameters to work well. Other classifier can be based on decision trees or neural networks, which will not be further discussed.

2 Methods

Discrete Kalman Filter Algorithm

Dynamic systems are often described in a mathematical representation of a physical system as set of linearly related inputs, outputs and state variables. The Kalman filter, a predictor-corrector algorithm, estimates the state of a discrete-time controlled process with the following models. [14]

State equation (process model)

$$x_k = A \cdot x_{k-1} + B \cdot u_{k-1} \qquad (1)$$

$x_k \ldots state\ estimate\ time\ step\ k$
$x_{k-1} \ldots state\ estimate\ at\ previous\ time\ step$

$A \ldots transition\ matrix$
$B \ldots control\ matrix$
$u_{k-1} \ldots exogenous\ control\ (known)$

Observation equation (measurement model)

$$z_k = H \cdot x_k \qquad (2)$$

$z_k \ldots measured\ value\ at\ time\ step\ k$

$H \ldots observation\ matrix$

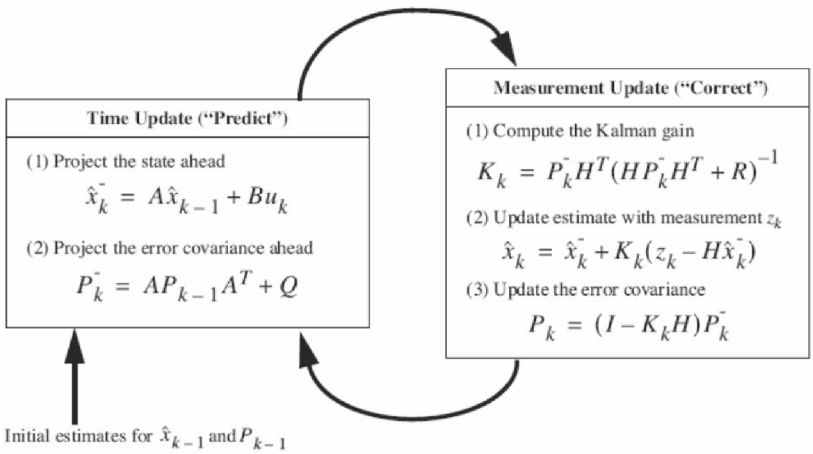


Figure 8: Kalman Filter Algorithm; the time update projects the current state estimate ahead in time. The measurement update adjusts the projected estimate by an actual measurement at that time. [14]

Additionally, random Gaussian white noise source w_k and v_k are respectively added to the state and observation equation. The process noise w has a covariance matrix Q and the measurement noise v, has a covariance matrix R, which are the uncertainties in the states

and correlations within it (fig. 8). At each time step k the algorithm calculates a state estimate x_k and a priori estimate for an error covariance P_k, which indicates the uncertainty associated with the current state estimate. The measurement update equations (corrector equations) give a feedback by including a new measurement value into the a priori estimate to get an improved a posteriori estimate. [14]

The Kalman Gain K_k in figure 8 (measurement update equation (1)) is derived from minimising the a posteriori error covariance, as the measurement error covariance R approaches 0, the actual measurement z is considered to be more and more true, while the predicted measurement $H_{\hat{x}}$ is being more and more considered false.

This recursive method is also known as Sensor Fusion Algorithm that combines data from different sensors in order to achieve more reliable data, which enables the compensation of sensor drift. It is widely spread in real-time applications, for it only requires an actual and previous measurement.

Motion Recognition Algorithm

Bevilacqua et. al. described a real-time gesture analysis algorithm, based on a Hidden Markov Model (HMM), with appliances in performing arts in mind. [15]

Firstly, the time progression of the motion is computed, in order to be able to follow the motion. Secondly, likelihood values between a performed motion and pre-recorded one are calculated and stored. Machine learning algorithms are generally divided into learning and decoding procedures. This approach can work with any type of time-discrete data flow.

- **Learning**

A recorded motion can be seen as temporal profile that is used to create a left-to-right HMM. The direct association of each sampled point with a state in a left-to-right Markov chain builds a model that fits the recorded reference. This can be modelled in a straightforward way (eq. 3). An observable O with a probability b_i is emitted by each state i. As σ_i describes standard deviation, it is obvious that its estimation from a single sample is not possible. Therefore, multiple recordings of the same motion must be taken or modelled based on prior experiments. The learning procedure is depicted in figure 9.

$$b_i(O) = \frac{1}{\sigma_i\sqrt{2\pi}} exp\left[-\left(\frac{O-\mu_i}{2\sigma_i}\right)\right] \qquad (3)$$

$b_i(O) \ldots probability\ of\ observing\ O$
$\mu_i \ldots i^{th}\ sampled\ value\ of\ recorded\ reference$

$\sigma_i \ldots standard\ deviation$

The structure is a statistic model of the data sequence recorded. Additionally a limited number of valid transitions are set by transition probabilities a_0, a_1 and a_2 (self, next, skip transition), which have to satisfy equation 4.

$$a_0 + a_1 + a_2 = 1 \qquad (4)$$

$a \ldots transiton\ probability$

These parameters, similarly as σ_i, cannot be estimated by a single sample. The values are based on either prior knowledge or measurements in specific scenarios. Following cases demonstrate the roles of these parameters.

- $a_0 = a_1 = a_2 = 1/3$ equal probabilities slower or faster performance of motion
- $a_0 < a_1\ and\ a_2 < a_1$ lower probability for speeding up or slowing down
- $a_0 < a_2$ lower probability for slowing down than speeding up
- $a_0 > a_2$ higher probability for slowing down than speeding up

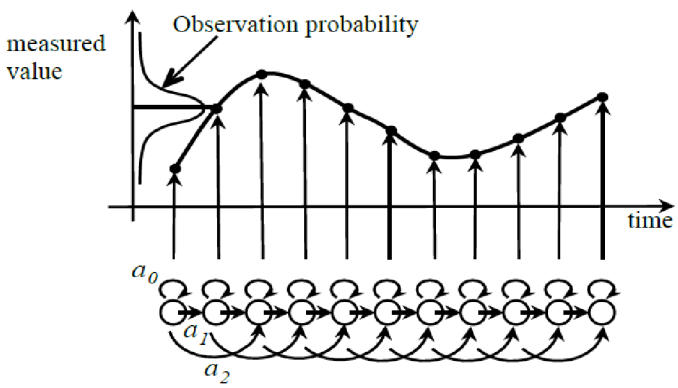


Figure 9: Learning procedure. Left-to-right Hidden Markov Model used to model the recorded reference motion, where a_0, a_1, a_2 are transition probabilities for self, next and skip transitions. [15]

- **Decoding**

The decoding scheme is based on a standard forward procedure in HMM. The computation of $\alpha_i(t)$ is required for the forward procedure, if a sequence of partial observations $O_1, O_2, \ldots, O_t$ are given. The variable $\alpha_i(t)$ corresponds to the probability distribution of the partial observation sequence, until time t and state i. It is inductively computed as follows.

- **Initialisation**

$$\alpha_1(i) = \pi_i b_i(O_1) \qquad 1 \leq i \leq N \qquad (5)$$

$\pi \ldots initial\ state\ contribution$
$b \ldots observation\ probability\ distribution$

- **Induction**

$$\alpha_{t+1}(i) = \left[\sum_{i=1}^{N} \alpha_t(i)\, \alpha_{ij}\right] b_i(O_t) \qquad 1 \leq t \leq T-1, 1 \leq j \leq N \qquad (6)$$

$\alpha_{ij} \ldots state\ transition\ probability\ distribution$

The variable $\alpha_i(t)$ allows the computation of the time progression of the sequence, related to the recorded sample (eq. 7) and the likelihood of the sequence (eq.8). Likelihood can be used as a similarity measure between the performed motion and recorded reference.

$$time\ progression\ index(t) = argmax[\alpha_i(t)] \qquad (7)$$

$$likelihood(t) = \sum_{i=1}^{N} \alpha_i(t) \qquad (8)$$

- **Windowing Technique**

The algorithm based on HMM might result in a very large number of states, which can be challenging for real-time computation. Therefore a sliding window method shown in figure 10, that uses a fixed number of states, can be applied, in order to reduce computational load.

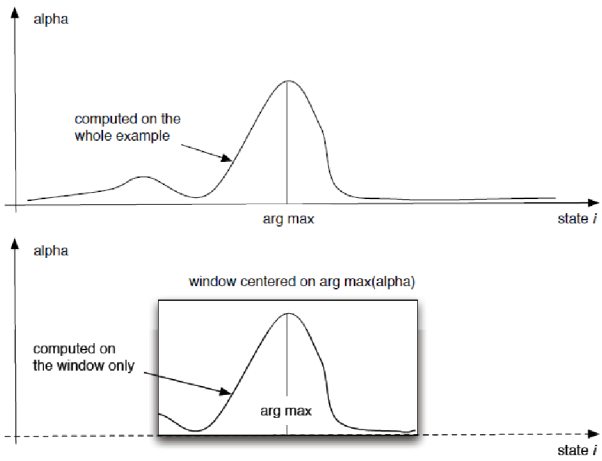


Figure 10: Windowing technique used in decoding in order to reduce CPU load. [15]

The computation of the probability distribution $\alpha_i(t)$, which is done on the entire state structure is required in the decoding procedure. The use of a window, limits this computation to a section of the state structure, where the window is centred on the arg max of the $\alpha_i(t)$ distribution. The window location is moved at each new step of the evaluation. Values of α_i that were not used in the previous window, are initialized to zero in the new window. The CPU load remains constant through the use of this method, as computation is done with a fixed number of states, independent of the length of data in the recorded motion.

3 Material and Realisation

For this diploma thesis the literature research was carried out by using www.google.at, scholar.google.com, www.ncbi.nlm.nih.gov/pubmed and www.youtube.com including following keywords: smart, sensor based, motion, tracking, capture

A reasonable combination of keywords yielded up to several hundred hits. Published papers, available products and products in development concerning sensor based motion tracking were considered as relevant. All useable literature is listed in the bibliography.

The components in table 2 are used for the development of the system:

hardware	software
• 1x Raspberry Pi 3 (RPi3) • 1x micro SDHC card (at least 8GB) • 1x micro SDHC USB adapter • 1x power bank (Output: 5V/2.4A) • 2x IMU GY-521 break out (MPU6050) • 50x jumperwire (Male-Female 20cm) • 1x 5 pin female header • 1x 10 pin female header • 1x breadboard • 5x3 perfboard • 1x USB A to USB B cable • PC (with Windows 10 and Wi-Fi modem) • Soldering station • Iso tape • 1 pair MMA gloves • 1 hoodie (sweater with compartment)	• NOOBS [16] • SD Formatter 4.0 [17] • WinRAR • Putty [18] • WinSCP [19] • RTIMULIB2 [20] • Gesture-recognition-HMM [21] • MATLAB

Table 2: Components used for development

Description

A portable wireless inertial sensor based motion tracking system shall facilitate tracking of activity occurring during a martial arts training session. In such a training session it is usual to exercise different striking techniques like a straight punch, uppercut or diverse kicking techniques and push ups, sit ups or squats. In order to be able to track and recognise the executed technique, 2 IMUs, each with an embedded accelerometer and gyroscope, are connected via I2C to a Raspberry Pi 3, are used. Each sensor will be placed on top of the each user's wrist. The RPi3 will be programmed, through a Python script, to obtain acceleration and angular rate data. That will then be pre-processed through a low pass filter to remove high frequency noise. A Kalman filter will be used to perform sensor fusion. The fusion of accelerometer and gyroscope will yield more reliable angular rate data. All samples will be stored in comma separated value (CSV) text files. These files can be wirelessly transferred to a PC via Secure File Transfer Protocol (SFTP). A Wi-Fi hotspot is required for a wireless connection between PC and RPi3. Figure 11 shows a simple block diagram of the system in development.

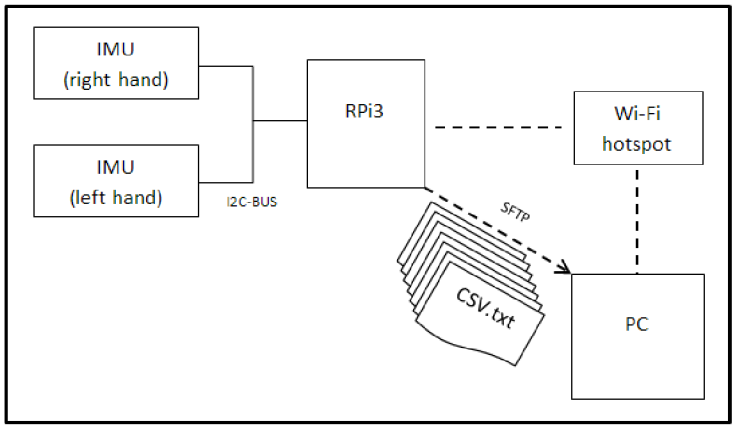


Figure 11: Block diagram of portable motion tracking and recognition system consisting out of RPi3 and two IMUs connected via I2C. Acceleration and angular rate data is stored in CSV text files. File transfer from the RPi3 to a PC through SFTP, possible when both are connected to the same Wi-Fi hotspot. The recognition task is carried out on the computer

MATLAB will be running on the PC, where the data is undergoing further processing and classification. The motions of interest, that will be recorded, are going to be a jab, cross, hook and uppercut. The recorded motions are classified offline with HMMs.

Connecting the MPU6050 and Raspberry Pi 3

The Raspberry Pi 3 (fig. 12) is a low cost credit sized computer. It was originally designed for educational purposes that would improve understanding hardware and programming. It runs an operating system based on Linux. Even though slower than modern laptops, the same CPU can be found in some smartphones as well. [22]

Figure 12 Raspberry Pi 3 is a credit card sized computer with a Linux based operating system [23]

- Raspberry Pi 3 Features [23]

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A

The two IMUs utilised in the system are the MPU6050 from InvenSense. It is available a break out board with a tri-axis accelerometer and a tri-axis gyroscope. [24] A picture of the break out board is shown in figure 13. Along the short edge is the x-axis. The y-axis is along the long edge and z-axis is perpendicular to x-y plane. The MPU6050 has a default hardware address of 0x68. The address can be changed to 0x69 by setting the AD0 pin to logic high.

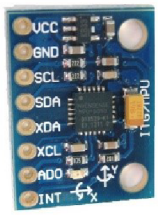


Figure 13: MPU6050 on break out board GY-521 [25]

Figure 14 illustrates a block diagram of the MPU6050. Note that SPI is not available on this chip. Each axis has its own independent accelerometer and gyroscope that are digitized by separate 16-bit ADC. [26]

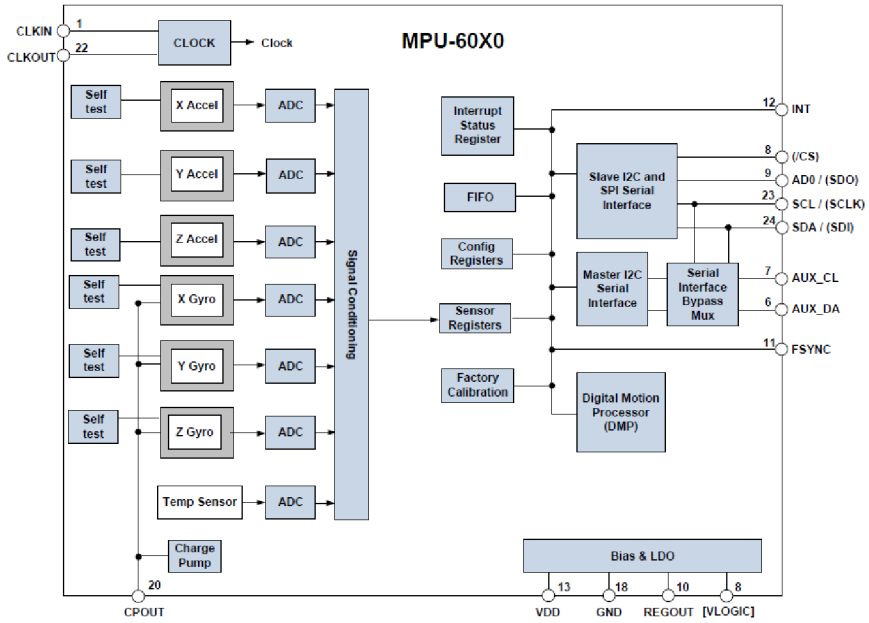


Figure 14: Block diagram of MPU6050 (note: SPI interface not present) each axis has independent sensors that are digitized by separate 16-bit ADCs [26]

- Gyroscope Features [26]

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ±250, ±500, ±1000, and ±2000°/sec
- Integrated 16-bit ADCs enable simultaneous sampling of gyros
- Enhanced bias and sensitivity temperature stability reduces the need for user calibration
- Improved low-frequency noise performance
- Digitally-programmable low-pass filter
- Gyroscope operating current: 3.6mA
- Standby current: 5μA
- Factory calibrated sensitivity scale factor

- Accelerometer Features [26]

- Digital-output triple-axis accelerometer with a programmable full scale range of ±2g, ±4g, ±8g and ±16g
- Integrated 16-bit ADCs enable simultaneous sampling of accelerometers while requiring no external multiplexer
- Accelerometer normal operating current: 500μA
- Low power accelerometer mode current: 10μA at 1.25Hz, 20μA at 5Hz, 60μA at 20Hz, 110μA at 40Hz
- Orientation detection and signaling
- Tap detection
- User-programmable interrupts
- High-G interrupt

The MPU6050 comes with L-shaped male pin headers that have to be soldered onto the breakout board. Both sensors are connected to the RPi3's I2C-bus, 3.3V power supply and ground via coloured jumper wires (see table 3 and figure 15).

Pi3		MPU6050-1		MPU6050-2	
Label	Pin	Label	Pin	Label	Pin
3.3V PWR	1	VCC	1	VCC	1
GND	6	GND	2	GND	2
I2C1SCL/GPIO3	5	SCL	3	SCL	3
I2C1SDA/GPIO2	3	SDA	4	SDA	4
GPIO4	7	AD0	NC	AD0	7

Table 3: Pin out for connecting RPi3 to two MPU6050.The colour indicate the colour of jumper wire used for connection. NC stands for not connected [27]

In figure 15 it can be seen how it could be connected using breadboard that is then replaced by a custom connector built from 5 pin female header, 10 pin female headers and 5x3 perfboard (fig. 19).

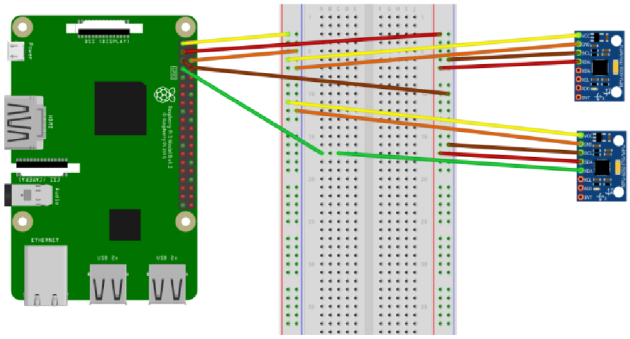


Figure 15 Connecting RPi3 to both MPU6050 with breadboard according to table 2 [28]

Setting up the Raspberry Pi 3

Step 1: Download NOOBS (New Out Of the Box Software) ZIP file from [16].

Step 2: Extract the content of ZIP file to a folder using WinRAR.

Step 3: Insert SD card to USB adapter and connect to PC.

Step 4: Format SD card to FAT32 with SD Formatter 4.0 from [17], in the "Options" menu set "FORMAT SIZE ADJUSTMENT" to "ON"

Step 5: Copy content of the folder, from extracted ZIP file, to SD card.

Step 6: Connect the RPi3 to a screen via HDMI, a keyboard and mouse via USB

Step 7: Insert SD card into Raspberry Pi 3 and power on

Step 8: Set language, keyboard layout and connect to Wi-Fi network.

Step 9: Select Raspian and install

Step 10: When installation is done, hoover with mouse over the Wi-Fi icon in status bar, to get the IP address

Step 11: Open Start Menu -> Preferences -> Raspberry Pi Configuration in Localisation set Locale, Time zone and Wi-Fi Country. A reboot is necessary to apply changes.

Step 12: Open Start Menu -> Preferences -> Raspberry Pi Configuration in Interfaces enable SSH and I2C. A reboot is necessary to apply changes.
Note: enable VNC to connect to RPi3 via remote desktop using VNC software

From this step on it is not necessary for the RPi3 to be connected to a screen and input devices. Since SSH is enabled and the IP address is known, a connection between PC and RPi3 can be established using a terminal such as Putty or using a remote desktop such as VNC if enabled.

Run Putty and enter IP address of RPi3, set port to 22 and select connection type as SSH. The default username is pi and default password is raspberry. When asked to enter password, the keystrokes will not be displayed.

Step 13: Open terminal and download RTIMULIB2 [20] library with the following command:
```
$ git clone https://github.com/RTIMULib/RTIMULib2
```

Step 14: Change to /RTIMULIB2 directory with following command:
```
$ cd RTIMULIB2
```
To open the README file for some basic information about the library enter:
```
$ nano README.md
```
To close README file press CTRL+x.

Step 15: Change to /Linux directory with following command:
```
$ cd Linux
```
Instructions provided in the README file inside this directory are complied with as follows.

Step 16: Add following two lines to /etc/modules:
```
i2c-bcm2708
i2c-dev
```
for this following command is entered:
```
$ sudo nano /etc/modules
```
Then add the two lines above. Exit with CTRL+x and confirm saving changes by pressing y

Step 17: Create a file “raspi-blacklist.conf” in /etc/modprobe.d and add following line:
```
# blacklist i2c-bcm2708
```
to do that enter following command:
```
$ sudo nano /etc/modprobe.d/raspi-blacklist.conf
```
Add the line above. Exit with “CTRL+x” and confirm saving changes by pressing y.

Step 18: Install I2C Tools with following command:
```
$ sudo apt-get install i2c-tools
```

Step 19: Allow other users to access I2C devices as by default it is owned by root.
Create a file 90-i2c.rules in /etc/udev/rules.d and add following line:
```
KERNEL==”i2c-[0-7]”,MODE=”0666”
```
To do that enter following command:
```
$ sudo nano /etc/udev/rules.d/90-i2c.rules
```
Then add the line above. Exit with CTRL+x and confirm saving changes by pressing y.

Step 20: Change the I2C bus speed to 400kHz. Add following line to /boot/config.txt:
dtparam=i2c1_baudrate=400000
Enter following command:
```
$ sudo nano /boot/config.txt
```
Add the line above. Exit with CTRL+x and confirm saving changes by pressing y.
Reboot to apply changes.

Step 21: Install cmake for compiling and installing the library.
Open Terminal again and enter following command:
```
$ sudo apt-get install cmake
```

Step 22: Install Qt for compiling Qt-based GUI demo programs.
Enter following command:
```
$ sudo apt-get install libqt4-dev
```

Step 23: Compile and install libraries and demo programs.
Change to directory /RTIMULib2/RTIMULib with following command:
```
$ cd RTIMULib2/RTIMULib
```
Inside this directory execute next commands after another:
```
$ mkdir build
$ cd build
$ cmake ..
$ make -j4
$ sudo make install
$ sudo ldconfig”
```

The RTIMULib2 is a library that facilitates the use of IMUs. It provides support for many devices from different manufacturer. That library makes it simple to adjust the sensors' address, sampling rate and full scale range. It is available from [20] under an MIT license.

A real-time plotter has been developed which directly sends the IMUs readings to MATLAB for live plotting via TCP socket. Execution of the scripts LIMUTCP.py and RIMUTCP.py requires the installation of the socket library witch: `$ sudo apt-get install socket.` Those scripts need to be executed before running pi_tcp.m script in MATLAB.

Testing the sensors

After setup the sensors' functionality can be tested with a demo program that is already included inside the RTIMULib2 library. The GPIO4 pin, which is connected to AD0 pin on one MPU6050 needs to be set as output and high. This is done in the terminal with following commands: $ `gpio -g mode 4 out` and $ `gpio -g write 4 1`

In order to run the demo program, the directory needs to be switched to /RTIMULib2/Linux/RTIMULibDemoGL. This is done from the home directory with: $ `cd /RTIMULib2/Linux/RTIMULibDemoGL`

The demo program is then started by entering: $ `RTIMULibDemoGL`

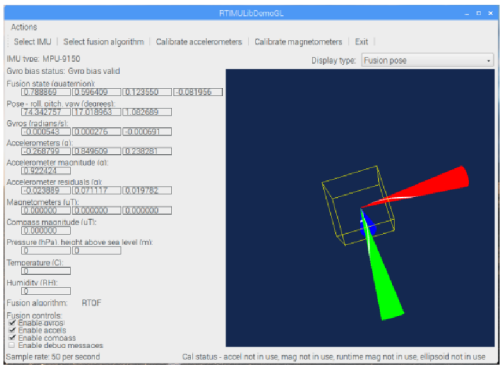


Figure 16 Screenshot of demo program for testing sensors [20]

Figure 16 is a screenshot of the demo. It shows a 3D object that has the same orientation as the sensor on the right side. The left side of the graphical user interface (GUI) shows the angular rate values in degrees and radians per second and as well the acceleration values.

Rotation around x-axis is denoted as roll, around y-axis as pitch and around z-axis as yaw. The demo allows selecting between the two IMUs, select between a Kalman and quaternion filter and calibrating the accelerometer.

The IMU Type will display MPU-9150, which is not relevant, as of the difference to MPU6050, is that the MPU9150 has a magnetometer additionally. Missing sensors are automatically detected during the initialization procedure.

Programming the Raspberry Pi 3

- **Setting up the MPU6050**

A file RTIMULib.ini is created inside the demo folder when starting the demo program. That file is used for setting up the IMUs and needs to be copied or moved to the working directory named tests. It is located in /RTIMULib2/Linux/python/tests. This can be carried out with:
$ `cp RTIMULib.ini /RTIMULib2/Linux/python/tests` or
$ `mv RTIMULib.ini /RTIMULib2/Linux/python/tests`

Inside the working directory tests two python scripts are created, one for each IMU. Therefore a second .ini file is needed. First the directory is switched with:
$ `cd /RTIMULib2/Linux/python/tests`

Then RTIMULib.ini file is opened with:
$ `nano RTIMULib.ini`

The Appendix contains an excerpt of the .ini file highlighting the simplicity of adjusting settings. Inside the file the corresponding values are set to apply following setup.

- IMU type is changed from auto discover to MPU6050 (MPU9150)
- Kalman filter is selected as sensor fusion
- IMU address is set as 104 (0x68)
- Pressure and humidity sensor are changed from auto discover to not in use (not present)
- Sampling rate is set to 250Hz
- Cut off frequency of low pass filter is set to 20Hz for gyroscope and 21Hz for accelerometer
- Gyroscope full scale range is set to ±2000°/s
- Accelerometer full scale range is set ±16g

CRTL+o saves the changes made after naming the file LIMU.ini. The .ini file is opened again, to change the IMU address to 105 (0x69) and then saved as RIMU.ini. The names of the settings files now indicate which sensor shall be placed on which wrist for consistent recordings.

- **Writing Python scripts**

Python is typically used for web, internet development, scientific and numeric computing. [29] It is an object oriented interpreter language, which comes pre-installed on Raspbian. Programs can be written with any text editor, as long the file is saved as .py.

The written scripts are placed into the working directory RTIMULib2/Linux/python/tests. Fusion.py provided by the RTIMULib2 library inside the working directory as an example, is used as basis. Two scripts are produced, again one script for each sensor. Both programs are the same, except for which settings file is used to create a sensor object connected via assigned address. Figure 17 shows a flow diagram of the produced script. The program will constantly poll data, at a sampling rate of 250Hz, for 5 minutes, unless the process is stopped before it finishes. At the end 6 CSV files containing tri-axis accelerometer, tri-axis Kalman filtered angular rate and tri-axis RAW angular rate readings are generated as text files, 3 per each IMU.

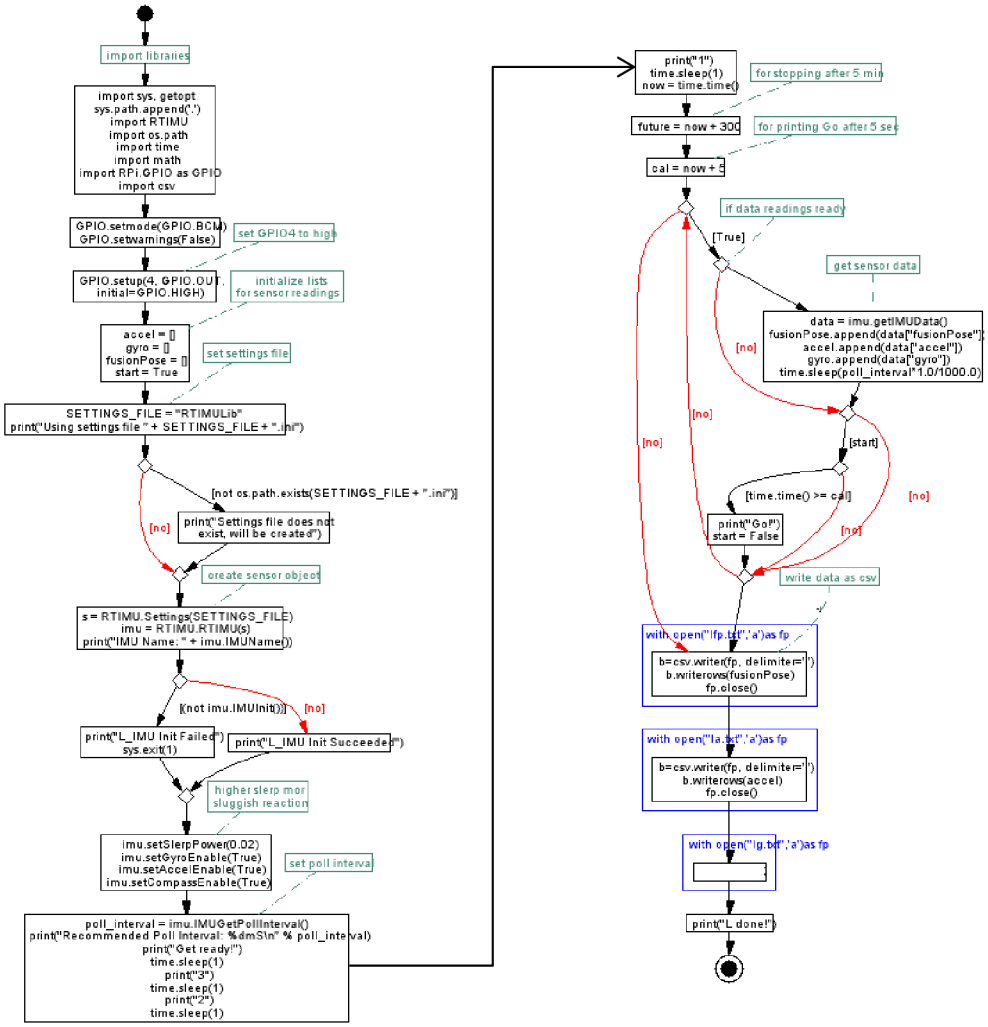


Figure 17 Flow chart of Python script [30]

- **Execution of scripts**

The programs are executed as background process from the working directory with:
$ `sudo python2 LIMU.py & sudo python2 RIMU.py &`

A process IDs, typically 1 and 2 followed by a process number, are assigned to each process upon execution of the script. These are needed in case of prematurely termination.

- **Stopping the scripts**

The background process is called into the foreground with the command $ `fg [process ID]`. Once it is in the foreground it can be stopped with CRTL+c.

Attaching the sensors

A sweater with a compartment in front of the stomach and MMA gloves are used for wearing the tracking system. The RPi3 is placed inside the compartment (fig. 18 left) with a with a power bank as power supply. A hole on the inside of the sweater, allows the sensors and the wiring to be driven to their designated position on the wrist (fig. 18 right)

Figure 18 preparing the system to be worn. RPi3 placed on the inside of the sweater's compartment and wiring driven to their designated position via a hole [31]

The sensor with the green wire is placed on top of the right wrist and fixated with the glove's strapping (fig. 19). The x-axis points towards the fingers. The y-axis faces the direction to that the right thumb points and z-axis going through the wrist. In the same manner, the remaining sensor is attached to the left wrist. Just that the y-axis of the sensor faces the opposite direction of the left thumb.

Figure 19 placing the sensor for attachment [31]

Recording motion

The types of punches under investigation are mentioned above. The following figure 20 shows a series of illustrations that correspond to each kind of strike. The technique is executed from left to right, starting from a ready position and returning to it. From top to bottom a jab performed with the left hand, cross punch with the right hand, left hook, right hook, left uppercut and right upper cut is carried out. A data series has been recorded for each type of strike. The ready position is maintained for at least 5 seconds before starting to deliver punches. This ensures that sufficient samples are gathered for calculating an offset. After initial samples, 5 minute long recordings are taken of one punch with a pause of 2 to 3 seconds in between the execution of the strike.

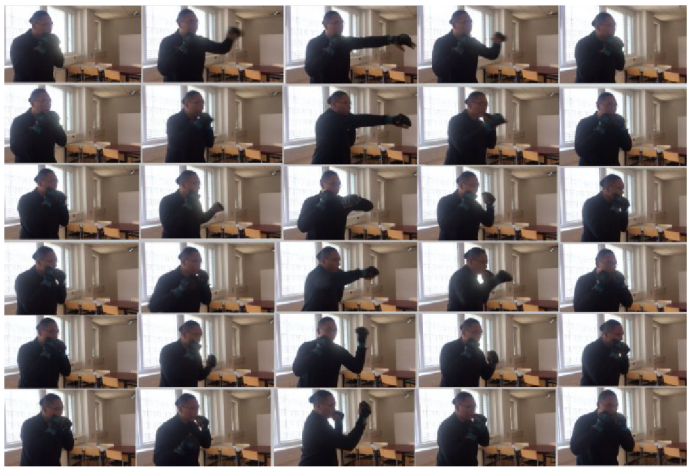


Figure 20 left to right: performance of basic punches; top to bottom: jab, cross, left hook, right hook, left uppercut and right uppercut [31]

Transferring data from RPi3 to PC with WinSCP

WinSCP is a freely available application that enables wireless file transmission via SSH, using a GUI. [19] Once executed on the PC, the IP address of the remote system has to entered, as well as the user name and password. The process of connecting shown in figure 21, is similar to connecting via SSH, with Putty as mentioned before.

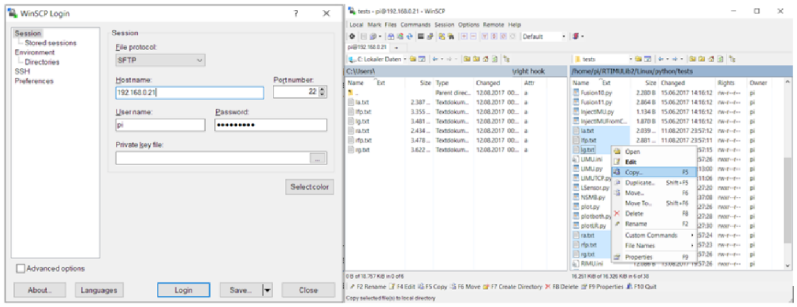


Figure 21: Screenshot of login procedure to RPi3 using WinSCP and copying files from RPi3 to PC into working directory used by MATLAB [19]

The RPi3's IP address is placed under host name, SFTP is selected as file protocol and port 22. Login can be executed after entering user name and password to the respective field. In figure 23 the process shown of copying the text files of interest from the RPi3's working directory straight into the MATLAB's working directory into a folder of the designated punch that has been recorded.

Processing data with MATLAB

- Data visualization

For first inspection a sequence containing each type of punch, is plotted. As visualizing the entire recorded data would render the plot unreadable, two minutes of readings have been selected for further investigation.

- Post-processing

The samples are high pass filtered to remove occurring low frequency noise and drifts at a cut off frequency of 1Hz.

- Feature extraction

A threshold has been determined to identify local acceleration maxima along the x-axis readings, as it has been considered a reliable indicator for the occurrence of a strike. This identification is carried out through a peak finding algorithm that enumerates each found local maxima. The power spectral density (PSD) estimate is as well used as feature. The estimate is obtained with a 50 sample wide rectangular window, overlapping 25 samples.

- Data segmentation

The data has been segmented into 1 second windows, whereas a window is centred on each identified peak value of the acceleration readings along the x-axis. Those windows are then saved as CSV text files that are named with an abbreviation indicating the type of punch it contains and the same enumeration as the peak.

- Training HMMs

A HMM algorithm for MATLAB provided by Ankit Vora on Github [21], is used for training and classification. The first 5 punches of each strike series are used for training HMMs, whilst the rest is used for classification. The first 5 files of each punch are placed inside the folder called train and the rest of the text files are moved to the folder named test. The model parameters inside the ini.m script are changed to 6 classes, 5 files per class, 36 states and 251 cluster centres. The logarithmic likelihood is calculated for each trained punch for 50 iterations after calculating 1 cluster centre per sample using kmeans. Training is started by running the hmmTrain.m script. Pressing any key will skip to the next estimation of likelihood. Furthermore the same task is carried out for the data, with units converted from [g] and [rad/s] are converted into [m/s²] and [deg/s], as well with power spectral density estimate. In this case the number of clusters is set to 129.

- Classifying test data

A total of 369 punches are used as test data. The classification is started by running the testMain.m script. The predictions are skipped through by pressing any key, as like the likelihood functions during training. Predictions are displayed in the command window.

4 Results

Data visualization

The visual inspection of the first data sequence recorded depicted in figure 22, containing all types of punches, reveals that each technique has a unique set of combined profile.

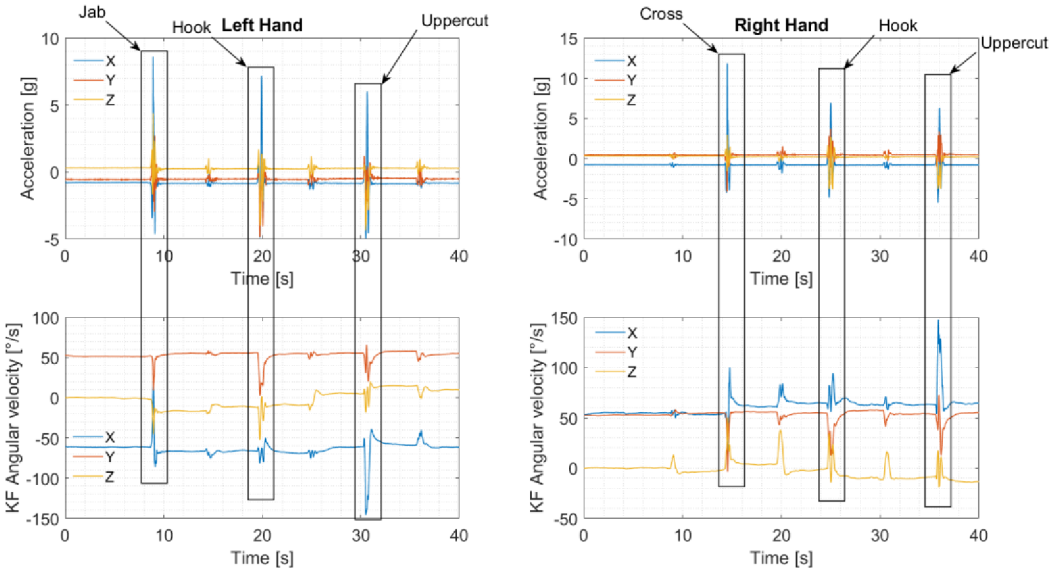


Figure 22: Acceleration and Kalman filtered profile for each type of punch for both hands. It can be seen that the various technique have different kinds of orientation around the respective axis [32]

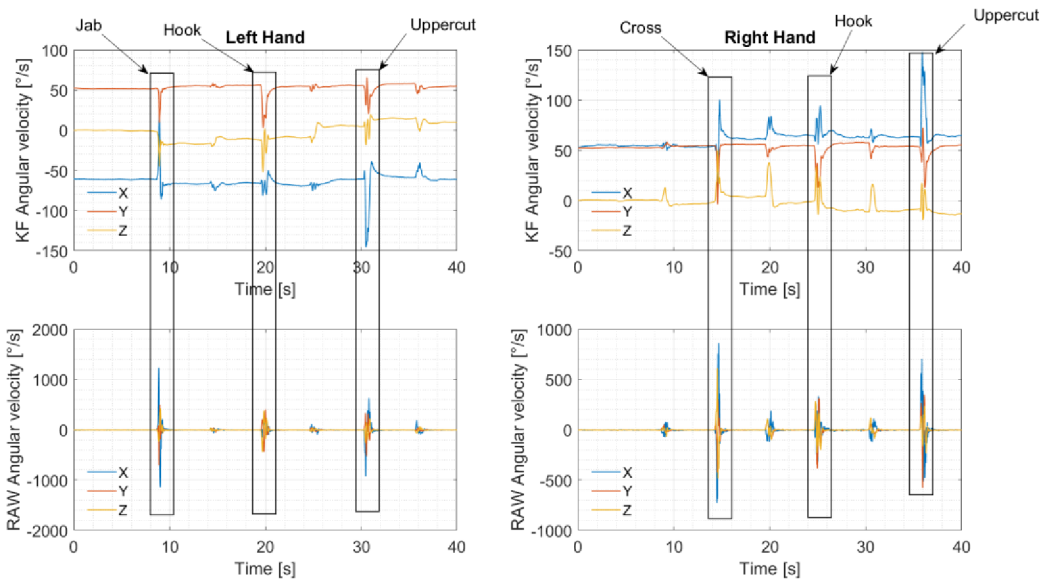


Figure 23: Comparison of RAW and Kalman filtered readings of both hands. The y-axis' limits, fronts that RAW gyroscope data is not reliable and therefore neglected for future investigation [32]

Figure 23 shows a comparison between RAW and Kalman filtered (KF) angular velocity. Future investigation comprising RAW gyroscope data is neglected, as the values are not reliable. The following figure 24 and figure-A 1 to A-5 of the remaining techniques in Appendix are illustrating 2 minutes long sample for each punch type taken out of 5 minutes recordings.

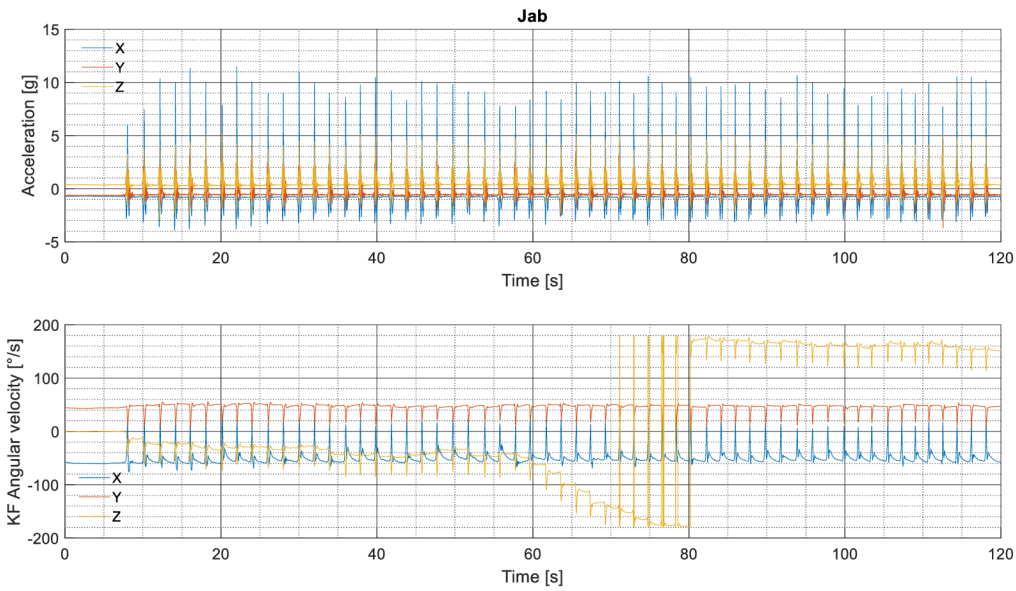


Figure 24: Recorded series of jabs. Strikes are performed after maintaining in ready position for at least 4 seconds. After each strike the subject remains in ready position for about 2 seconds. Each sensor suffers from an offset. In addition the gyroscope's z-axis is prone to drift [32]

Common findings in the figures 24 and A-1 to A-5 are that the entire set of axes suffer from an offset and as well that the gyroscope's z-axis is prone to drift. Those issues are resolved through subtraction of the mean value of the initial 1000 readings that corresponds to taking the mean over the first 4 seconds considering sampling at 250Hz. In order to remove the drift a high pass filter with a cut off frequency of 1Hz is applied afterwards.

Post-processing

As seen before each axis of both the accelerometer and gyroscope has an offset and in addition the readings around the z-axis tend to drift. The offset is removed by removing the mean taken over the first 4 seconds. A 4th order Butterworth high pass filter with a cut off frequency of 1Hz is applied in a zero phase shift manner. This attenuates the low frequency content causing the sensor drift. The outcome of this process is seen in figure 25 and figures A-6 to A-10. After the post-processing it can be seen that, except for the series of left uppercuts, some outliers still remain on the gyroscope's z-axis.

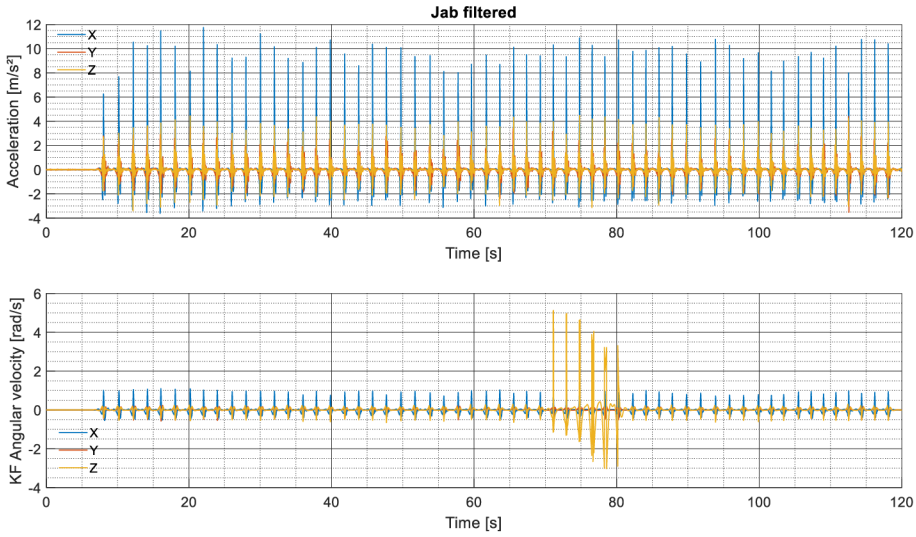


Figure 25: Series of jabs after removing offset and drift through high pass filtering [32]

Feature extraction

As mentioned before a peak finding algorithm is used to detect highest occurring local maxima along the accelerometer x-axis. Each identified peak is numbered upon detection and is demonstrated in the figures 26 and figures A-11 to A-15. Maximum peak is measured in a cross punch with 12.555g at 28.78 seconds with an width of 133ms

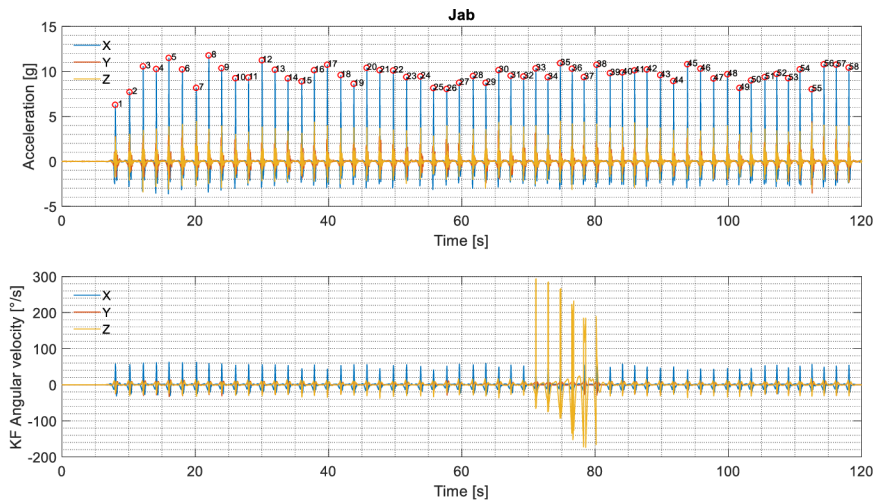


Figure 26: 58 peaks found in series of jabs after setting determined thresholds [32]

All peaks of the accelerometer's x-axis are identified using a minimum peak width of 10 samples and thresholds listed in table 4. The data is then segmented in 1 second windows centered at each peak with a window length of 251 samples, except each final peak.

Unit of data	Jab	Cross	Left hook	Right Hook	Left uppercut	Right uppercut
[g] and [deg/s]	5	5	4	5	1.3	4
[rad/s] and [m/s²]	49	49	39	49	12	39

Table 4: Thresholds for minimum peak used by peak finding algorithm

Training HMMs

The obtained data sets are written into CSV text files and placed into the test folder. First 5 files from each class of strike are moved to the train folder. One HMM is trained per punch with those samples.

A kmeans algorithm that tries to find 251 cluster centres one for each sample in training data through minimizing Euclidian distances. Figure 27 shows the cluster distribution for all 6 punches. This as well is carried out for data sets converted in SI units.

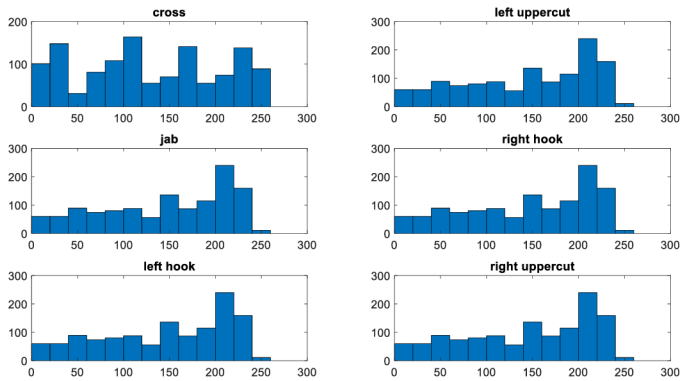


Figure 27: Cluster distribution for each kind of punch [32]

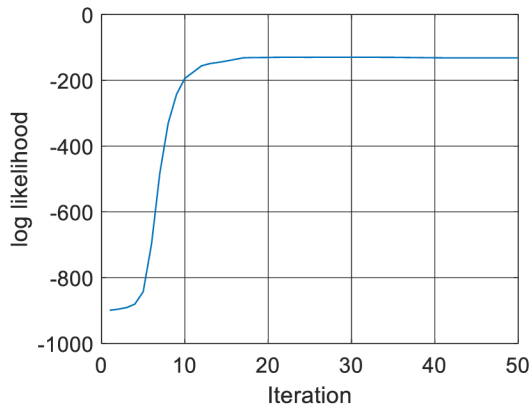
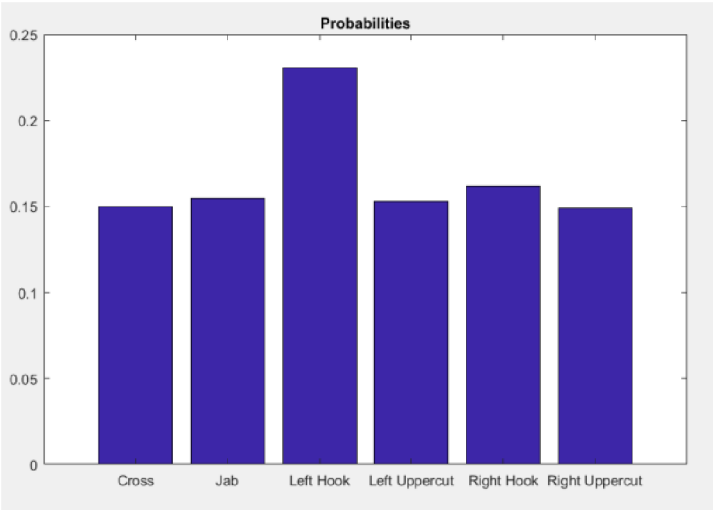


Figure 28: observation probability for 50 iterations [32]

The calculation of the observation probability, or likelihood of observing a certain sequence, is carried out for every punch used to train a HMM. The shape of such a likelihood function, after going through 50 iterations, encoding one strike, is shown in figure 28. It can be seen that training the HMM can already be terminated before, to reduce computational effort.

Classifying test data

After training the HMM, also known as encoding the observation sequence, decoding the test data is done using the model parameters and matching it with the most probable. An example is given in figure 29, showing a bar plot of probabilities and corresponding output of the command window.




```
'True Label'    'lh12.txt'    ' Predicted Label'    'Left Hook'

Cross#: 62
Jab#: 47
Left Hook#: 3
Right Hook#: 0
Left Uppercut#: 0
Right Uppercut#: 0
Total punches: 112
Hit enter to continue to next file
```

Figure 29 example of probability plot and corresponding output of command window [32]

30 punches, 5 per each class, are used to classify 369 punches. This is done three times, once with the accelerometer and gyroscope data in units of [g] and [deg/s], then in units of [m/s²] and [rad/s] and finally with the PSD estimate taken from the data given in SI units. The estimate taken from a jab is given as an example in figure 30.

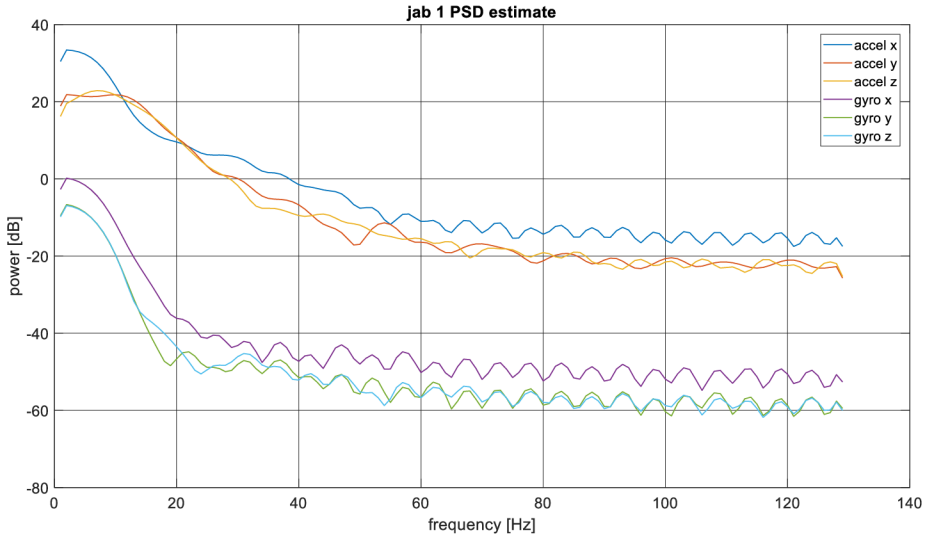


Figure 30: Example of PSD estimate used for classification showing a peak at 2 Hz [32]

About 6.23% namely 23 strikes are misclassified when using the data given in [g] and [deg/s]. However, the misclassification can be traced to the gyroscope's corrupted z-axis data. As soon as the compromised files are removed, every single remaining punch is classified correctly, even though training HMMs with 5 files per class.

The models trained with data in [m/s²] and [rad/s] failed to classify 61 punches. Disqualifying the corrupted test data, 28 techniques remain unrecognised.

Classification under the use of HMMs, trained with PSD estimate as feature, missed to correctly identify 196 out of the entire test set. Neglecting the 23 disturbed data files, the estimate managed exactly 173 out of 346 that are 50%. Table 4 summarises the results.

	[g] and [deg/s]	[m/s²] and [rad/s]	PSD
Corrupted data included	346/369 (93.77%/6.23%)	308/369 (83.47%/16.53%)	173/369 (46.88%/53.12%)
Corrupted data excluded	346/346 (100%/0%)	308/346 (89.02%/10.98%)	173/346 (50%/50%)

Table 5: Hit and miss rate of trained models

5 Discussion

Even though the primary motivation for this project lies in the development of a portable wireless tracking system based on inertial sensors for the use in martial arts training, it may be adapted to satisfy similar purposes for rehabilitation, where tracking training progress is essential. As for rehabilitation and training any kind of sport, they face the same difficulties when it comes to motion tracking.

Typically, optical sensor based tracking systems are generally bound to clinical facilities and require skilled personnel for operation, which is accompanied by high financial expenditures. The correct positioning of markers, whether passive or active, need some experience as well and might not be achievable without any assistance.

During the tracking session the subject has to remain within the capture volume and it has to be ensured that the markers are not obstructed from the optical sensor's field of view. The same applies as well to visual marker-free based tracking systems, where the subject is not allowed to be obstructed. Whereas combinatorial tracking try to combine the advantages of visual marker and marker-free methods, it still demands intensive computational effort and calibration.

These considerations are the main arguments, which make inertial sensor based motion tracking systems attractive. As, state-of-the-art inertial based sensors, are produced as low-cost microelectromechanical systems, that are small in size, have a high sensitivity and a large measurement range. They are a cost efficient approach and basically easy to handle. However, each type of sensor has its own drawbacks, mentioned above, and to obtain reliable sensor data, which is essential for motion recognition, can be a challenge.

Commercially available inertial sensor based motion tracking technology is mainly focused on motion capture, which allows mapping the human motion to a 3-D virtual model. This method is widely used in animated movies and in interactive virtual reality games. They may be used for clinical applications, such as gait analysis or tracking of activities of daily life, which is not an objective of this project.

The striketec boxing glove manufactured by Elliot Fight Dynamics is developed for the boxing community. Some of the functions they advertise, such as the punch type recognition, are what this project aims for. Unfortunately it is not mentioned, which kind of classifier is used, in order to recognise the type of punch. Nevertheless, striketec just uses two sensor nodes and is only able to track movement of the upper limbs, whereas this project, as well aims on recognising kicking techniques that also involves the tracking of the lower extremities.

The method described by Bevilacqua et. al. was used in the context of performing arts, to build an interactive system for music and dance performances. Thus make it interesting for the use in the context of this project. It is based on a Hidden Markov Model (HMM) that underlies the Bayes theorem. The algorithm computes parameters relative to the motion time progression and its likelihood by comparing the executed motion with a stored reference motion, previously recorded during the learning procedure. A windowing technique that reduces computational load (fig. 10), which is always desired, is described as well.

The HMM compared to other classification models seems to be more intuitive, than dealing with f-1 dimensional hyperplanes like in support vector machines or highly graded abstractions, when it comes to artificial neural networks.

The activity recognition chain (ACR) summarises all necessary steps for proper human motion recognition, which comprises data acquisition from sensors, pre-processing the signal, signal segmentation, feature extraction and classification. The development is oriented as the ACR, whereas feature extraction takes place prior to signal segmentation.

Data acquisition, from inertial sensors, such as accelerometer and gyroscopes, does not pose an issue. However, it is known that accelerometer suffer from integration drift, due to measurement noise and gyroscopes from zero-offset drift, due to residual inertia. This is demonstrated in figure 24 and can be seen in figures A-1 to A-5. The challenge, mentioned earlier, is to implement a drift free sensor in order to obtain reliable sensor data, which is further required for correct classification. Therefore, a sensor fusion algorithm based on a Kalman filter is inevitable as shown in figure 23.

On that account the Kalman filter is described as one of the applied methods. It is a predictor-corrector algorithm, based on Bayesian inference, where an educated guess about the next value is made, which then is corrected by an actual measured value. This method allows to compensate the drift, which the inertial sensor underlies and hence to obtain reliable sensor data. However, even the use of post-processing methods, such as the use of filtering, does not ensure safety from other disturbances.

In figure 25 and figures A-6 to A-10 it can be observed that some artefacts still remain in the signal, although a low pass filter in the MPU-6050 already in use at recording and a high pass filter applied afterwards. Those distortions might be caused by crosstalk, a phenomenon occurring when signal carrying lines interfere with one another. Since both sensors in use are connected via wires over I2C, where the bus speed is set to 400 kHz, being exposed to high accelerations for a short period of time, might induce motion artefacts, as maintaining the correct timings is crucial for the communication between sensor and processor, while polling at a high frequency, as 250Hz, can facilitate that. Especially, when the processor in use, is handling requests over scheduler, which is common in operating systems like Linux. The Raspberry Pi 3 used in this setup operates on Raspbian, a Linux based system. A motive not to use interrupts to acquire sensor readings is the lack of a real time clock, even though possible, with custom interrupt handler.

Since Wi-Fi is featured on the RPi3, a portable device that can record movement and recognise in an offline manner has been developed and tested. The use of a Kalman filter in an IMU based system is unavoidable, as already mentioned and shown in a comparison of sensor fused data and RAW gyroscope data.

The library in use allows simple exploit of sensor communication by providing python wrapper that allows straightforward use, without the necessity to deal with setting bits in registers manually. All required setting can be adjusted inside an initialization file. Two separate scripts are run simultaneously on different threads, as before mentioned timings are crucial.

Even though the programs are executed at the same time, one sensor will acquire more data than the other, as the process handler is not able to provide the same ID for scheduling reasons. However, the disparity in data acquisition is negligible.

The attempt to handle both sensors on a single thread can result in a huge difference in collected samples. This happens, as using both sensors within a script can just be done sequentially. While one IMU is ready to be read, it might happen that it has to dismiss its current sample, as it has to wait for the other sensor finishing its current reading when a buffer is not in use. This is experienced in the early stages of development.

Another issue encountered is that taking measurement for long periods is limited. It is related to the sampling frequency, as a big amount of data can overwhelm the csv writer, thus result in no samples to corrupted samples written to the text file. It is as well noticed that the RPi3 tends to overheat when used extensively. At that point it happens to fail to initialize the sensor object.

In this particular setup, it is possible to record motion and still be wearing the device while transferring data, since it is portable. It is possible to connect multiple IMUs to one RPi3 through the use of a multiplexer, considering running a Sensor Fusion Algorithm for multiple IMUs, even they can take some load of the CPU, on one microcontroller, does not seem feasible, given the computational speed, provided by the microprocessor. A preferred options, since a multiplexer introduces latency as it runs sequentially through the sensors, is the use of an address translator. Nevertheless, connecting multiple IMUs would require wires, which further can introduce motion artefacts that require additional filtering. However, the use of wireless technology comes with different challenges such as synchronisation and electromagnetic interference.

The use of HMM allowed the correct classification of the recorded motion after a long series of trial and error. Identifying the compromised data is facilitated by labelling each dataset with its corresponding content and respective enumerated peak. After neglecting the corrupted data, surprisingly a 100% classification rate was obtained by only having the algorithm trained with 5 files per class. The kmeans algorithm tries to minimize the Euclidean distance so that each centroid found matches one cluster. The distance can be further reduced through increasing the number of clusters, whereas the distance becomes zero when the number of clusters corresponds to the length of data points. Thus, it cannot be assumed when training the model again that it will perform the same way, as kmean algorithm randomly selects data points and optimizes iteration after iteration. However, it is possible to classify all test data, even if there cannot be a single target. The performance for the data set, though different, is still acceptable when scaled by unit conversion.

The use of the PSD estimate, as feature, reduced the sample length, thus increasing computational speed. However, determining the right adjustment for initializing parameter is again a trial and error of educated guesses, since there is no general approach 50% seem fair enough.

In conclusion it can be said that taking the next step is developing a real time standalone online classification system using that shall allow training the model instantly.

Bibliography

[1] D. Knudson, *Fundamentals of Biomechanics*, 2nd Edition, Springer Science+Business Media, New York 2007, p. 5

[2] J.I. Cowie, J.A. Flint, A.R. Harland, *Wireless Impact Measurement for Martial Arts – The Engineering of Sport 7: Vol.1*, Springer-Verlag France, Paris, 2008

[3] H. Zhou, H. Hu, *Human Motion Tracking – A Survey*, Biomedical Signal Processing and Control: Volume 3, Issue 1, Elsevier, 2008

[4] R. O'Reilly, K. Harney, *Acceleration/Vibration Sonic Nirvana: MEMS Accelerometers as Acoustic Pickups in Musical Instruments*, June 1, 2009 – http://www.sensorsmag.com/sensors/acceleration-vibration/sonic-nirvana-mems-accelerometers-acoustic-pickups-musical-i-5852 retrieved 22.07.2017

[5] D. Nedelkovski, Arduino Tutorials - Electrical Engineering, November 19, 2015 http://howtomechatronics.com/how-it-works/electrical-engineering/mems-accelerometer-gyrocope-magnetometer-arduino/ retrieved 22.07.2017

[6] Vicon Motion Systems Limited - https://www.vicon.com/what-is-motion-capture retrieved 21.01.2017

[7] Xsens Technologies B.V. P.O.Box 559 7500 An Enschede, Netherlands, https://www.xsens.com/download/pdf/documentation/mvn-biomech/mvn-biomech.pdf retrieved 21.01.2017

[8] APDM Inc., 2828 SW Corbett Avenue, Portland, OR 97201, http://www.apdm.com/wearable-sensors/ retrieved 21.01.2017

[9] Inertial Labs, 39959 Catoctin Ridge Street, Paeonian Springs, VA 20129 U.S.A., https://inertiallabs.com/downloads/new/Inertial_Labs_3DSuit_Datasheet_rev1.2_March_2015.pdf retrieved 21.01.2017

[10] M. Wooster et al, 2000, *Real time boxing sports meter and associated methods*, U.S. Patent, US6611782 B1, issued August 26, 2003

[11] Elliott Fight Dynamics ,1951 Pisgah Road, Florence, South Carolina 29501 – http://striketec.com/ retrieved 23.01.2017

[12] Y. Saez, A. Baldominos, P. Isasi, *A Comparison Study of Classifier Algorithms for Cross-Person Physical Activity Recognition*, Sensors Vol. 17 Issue 1, MDPI, Basel, 2017

[13] T.M. Mitchell, *Machine Learning*; McGraw-Hill: New York, USA, 1997

[14] S. Mau, *What is the Kalman Filter and How can it be used for Data Fusion?*, http://see-out.com/sandramau/doc/Mau05MathKalman.pdf retrieved 21.01.2017

[15] F. Bevilacqua et al., Continous Realtime Gesture Following and Recognition, Gesture in Embodied Communication and Human-Computer Interaction, 8th International Gesture Workshop, GW 2009, Germany, February 25-27, 2009, Revised Selected Papers

[16] NOOBS, Raspberry Pi Foundation UK registered charity 1129409 (available online: https://www.raspberrypi.org/downloads/raspbian/)

[17] SD Association c/o Global Inventures, Inc. 2400 Camino Ramon, Suite 375, San Ramon, CA 94583 USA (available online: https://www.sdcard.org/downloads/formatter_4/)

[18] Simon Tatham, Cambridge, UK (available online: http://www.putty.org/)

[19] SourceForge Media, LLC dba Slashdot Media PO Box2452 La Jolla, CA 92038 (available online: https://sourceforge.net/projects/winscp/)

[20] richards-tech, LLC Copyright © 2014-2015 (availabe online: https://github.com/RTIMULib/RTIMULib2) retrieved 07.04.2017

[21] Ankit Vora Copyright © 2016 (available online: https://github.com/ankitvora7/Gesture-recognition-HMM) 28.06.2017

[22] Red Hat, Inc., Opensource.com, Rikki Endsley, 100 East Davie Street, Raleigh, NC 27601 USA (available online: https://opensource.com/resources/what-raspberry-pi)

[23] Raspberry Pi Foundation UK registered charity 1129409 (available online: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/)

[24] MPU-6000 and MPU-6050 Product Specification Revision 3.4, InvenSense Inc. 1197 Borregas Ave, Sunnyvale, CA 94089 U.S.A., p. 7

[25] Hompage arduino.cc, Arduino® (available online: https://playground.arduino.cc/Main/MPU-6050) retrieved 10.08.2017

[26] MPU-6000 and MPU-6050 Product Specification Revision 3.4, InvenSense Inc. 1197 Borregas Ave, Sunnyvale, CA 94089 U.S.A., p 7-28

[27] Pinout, Raspberry Pi Foundation UK registered charity 1129409 (available online: https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/)

[28] Figure created with Fritzing, Friends of Fritzing e.V. Paul-Lincke-Ufer 39/40 10999 Berlin Germany (available online: http://fritzing.org/download/)

[29] Python Software Foundation, 9450 SW Gemini Dr., ECM# 90772, Beaverton, OR 97008, USA. (available online: https://www.python.org/about/apps/)

[30] AivostoOy Kylänvanhimmantie16 00640HELSINKI FINLAND (available online: http://www.aivosto.com/about.html)
[31] Picture taken on 17.07,2017
[32] Plot created with MATLAB2015b

List of Figures

Figure 1: Block diagram of a body worn sensor network [2], which will be modified into an inertial sensor based motion tracking system. The modified system shall be able to recognise pre-recorded motion. 4
Figure 2: Classification of motion tracking with different sensor technologies [3] 5
Figure 3: Structure of a MEMS accelerometer [4] 6
Figure 4: Structure of a MEMS gyroscope [5] 6
Figure 5: Schematic of a VICON system [6] 7
Figure 6: Illustration how striketec works. The sensors embedded in the gloves send their data via Bluetooth to a smart device. All calculations are done on the smart device. Information can be stored and retrieved from the cloud. [11] 9
Figure 7: Steps of an activity recognition chain (ARC) from data acquisition to classification [12] 10
Figure 8: Kalman Filter Algorithm; the time update projects the current state estimate ahead in time. The measurement update adjusts the projected estimate by an actual measurement at that time. [14] 11
Figure 9: Learning procedure. Left-to-right Hidden Markov Model used to model the recorded reference motion, where $a0$, $a1$, $a2$ are transition probabilities for self, next and skip transitions. [15] 13
Figure 10: Windowing technique used in decoding in order to reduce CPU load. [15] 14
Figure 11: Block diagram of portable motion tracking and recognition system consisting out of RPi3 and two IMUs connected via I2C. Acceleration and angular rate data is stored in CSV text files. File transfer from the RPi3 to a PC through SFTP, possible when both are connected to the same Wi-Fi hotspot. The recognition task is carried out on the computer ..16
Figure 12 Raspberry Pi 3 is a credit card sized computer with a Linux based operating system [23] 17
Figure 13 MPU6050 on break out board GY-521 [25] ..... **Fehler! Textmarke nicht definiert.**
Figure 14: Block diagram of MPU6050 (note: SPI interface not present) each axis has independent sensors that are digitized by separate 16-bit ADCs [26] 18
Figure 15 Connecting RPi3 to both MPU6050 with breadboard according to table 2 [28] 19
Figure 16 Screenshot of demo program for testing sensors 22

Figure 17 Flow chart of Python script 24
Figure 18 preparing the system to be worn. RPi3 placed on the inside of the sweater's compartment and wiring driven to their designated position via a hole. 25
Figure 19 placing the sensor for attachment.....25
Figure 20 left to right: performance of basic punches; top to bottom: jab, cross, left hook, right hook, left uppercut and right uppercut 26
Figure 21: Screenshot of login procedure to RPi3 using WinSCP and copying files from RPi3 to PC into working directory used by MATLAB 26
Figure 22: Acceleration and Kalman filtered profile for each type of punch for both hands. It can be seen that the various technique have different kinds of orientation around the respective axis. 28
Figure 23: Comparison of RAW and Kalman filtered readings of both hands. The y-axis' limits, fronts that RAW gyroscope data is not reliable and therefore neglected for future investigation. 28
Figure 24: Recorded series of jabs. Strikes are performed after maintaining in ready position for at least 4 seconds. After each strike the subject remains in ready position for about 2 seconds. Each sensor suffers from an offset. In addition the gyroscope's z-axis is prone to drift.....29
Figure 25: Series of jabs after removing offset and high pass filtering 30
Figure 26: 58 peaks found in series of jabs 30
Figure 27: Cluster distribution for each kind of punch 31
Figure 28: observation probability for 50 iterations 31
Figure 29 example of probability plot and corresponding output of command window.....32
Figure 30: Example of PSD estimate used for classification showing a peak at 2 Hz 33

List of Tables

Table 1: Overview of products with their properties which are available on the market..... 8
Table 2: Components used for development 15
Table 3: Pin out for connecting RPi3 to two MPU6050.The colour indicate the colour of jumper wire used for connection. NC stands for not connected [27] 19
Table 4: Thresholds for minimum peak used by peak finding algorithm.....31
Table 5: Hit and miss rate of trained models.....33

List of Abbreviations

ADC	Analog to digital converter
ARC	Activity Recognition Chain
CSV	Comma separated value
e.g.	Example given
eq.	Equation
fig.	Figure
GUI	Graphical user interface
HMM	Hidden Markov Model
IMU	Inertial measurement unit
IP	Internet protocol
MEMS	Microelectromechanical system
PSD	Power spectral density
RPi3	Raspberry Pi 3
SFTP	Secure file transfer protocol
SPI	Serial peripheral interface
SSH	Secure shell

Appendix

Excerpt of RTIMULIB.ini :

```
# ######################################################################
#
# RTIMULib settings file
# General settings
# IMU type -
#   0 = Auto discover
#   1 = Null (used when data is provided from a remote IMU
#   2 = InvenSense MPU-9150
#   3 = STM L3GD20H + LSM303D
#   4 = STM L3GD20 + LSM303DLHC
#   5 = STM LSM9DS0
#   6 = STM LSM9DS1
#   7 = InvenSense MPU-9250
#   8 = STM L3GD20H + LSM303DLHC
#   9 = Bosch BMX055
#   10 = Bosch BNX055
IMUType=2
# Fusion type type -
#   0 - Null. Use if only sensor data required without fusion
#   1 - Kalman STATE4
#   2 - RTQF
FusionType=1
#
# I2C slave address (filled in automatically by auto discover)
I2CSlaveAddress=104
# Pressure sensor type -
#   0 = Auto discover
#   1 = Null (no hardware or don't use)
#   2 = BMP180
#   3 = LPS25H
#   4 = MS5611
#   5 = MS5637
PressureType=1
# Humidity sensor type -
#   0 = Auto discover
#   1 = Null (no hardware or don't use)
#   2 = HTS221
#   3 = HTU21D
HumidityType=1

# ######################################################################
#
#
# MPU-9150 settings
#
# Gyro sample rate (between 5Hz and 1000Hz)
MPU9150GyroAccelSampleRate=250
#
# Gyro/accel low pass filter -
#   0 - gyro: 256Hz, accel: 260Hz
#   1 - gyro: 188Hz, accel: 184Hz
#   2 - gyro: 98Hz, accel: 98Hz
#   3 - gyro: 42Hz, accel: 44Hz
#   4 - gyro: 20Hz, accel: 21Hz
#   5 - gyro: 10Hz, accel: 10Hz
#   6 - gyro: 5Hz, accel: 5Hz
MPU9150GyroAccelLpf=4

#
# Gyro full scale range -
#   0  - +/- 250 degress per second
#   8  - +/- 500 degress per second
#   16 - +/- 1000 degress per second
#   24 - +/- 2000 degress per second
MPU9150GyroFSR=24

#
# Accel full scale range -
#   0  - +/- 2g
#   8  - +/- 4g
#   16 - +/- 8g
#   24 - +/- 16g
MPU9150AccelFSR=24

# ######################################################################
```

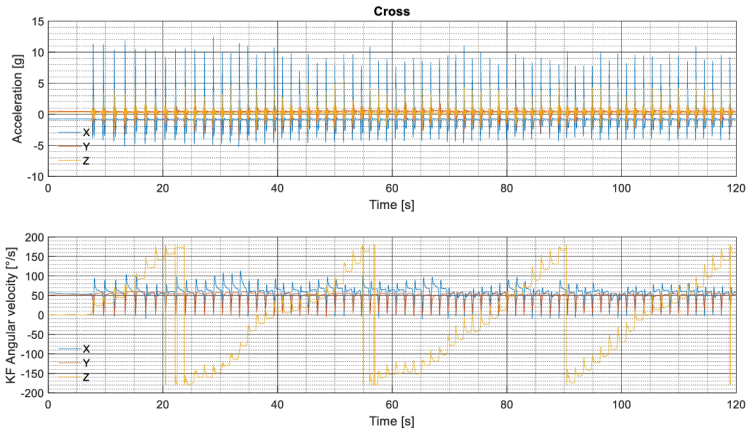


Figure A - 1: Recorded series of cross punches [32]

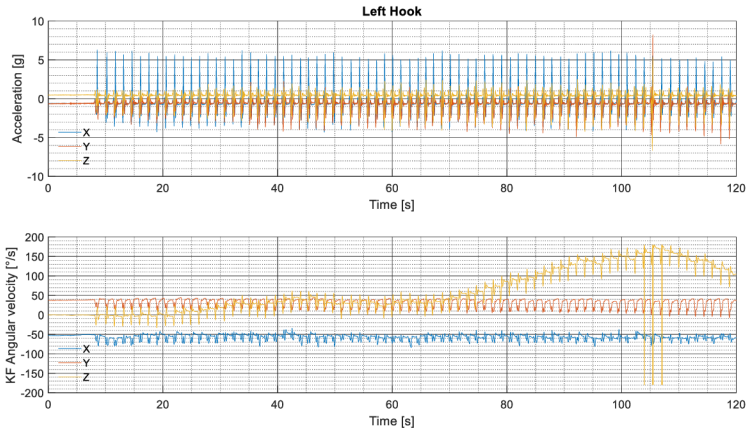


Figure A - 2: Recorded series of left hooks [32]

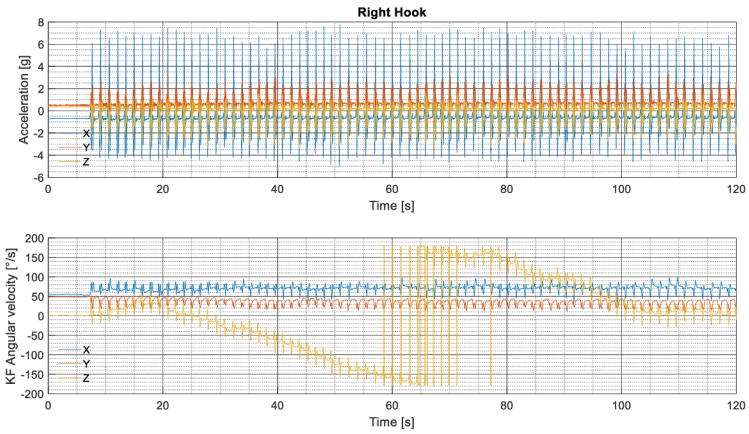


Figure A - 3: Recorded series of right hooks [32]

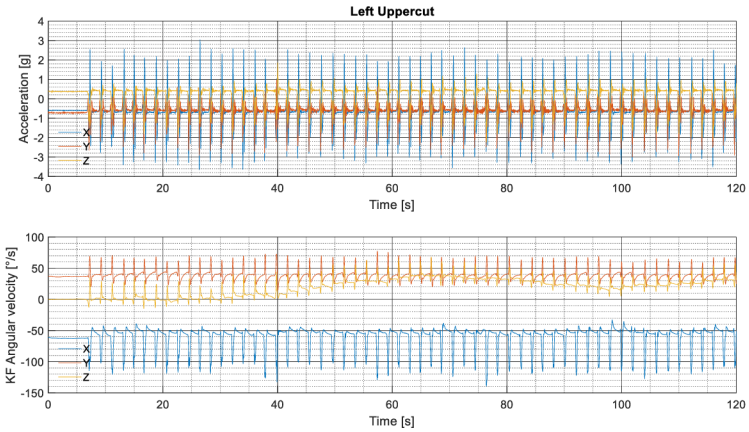


Figure A - 4: Recorded series of left uppercuts [32]

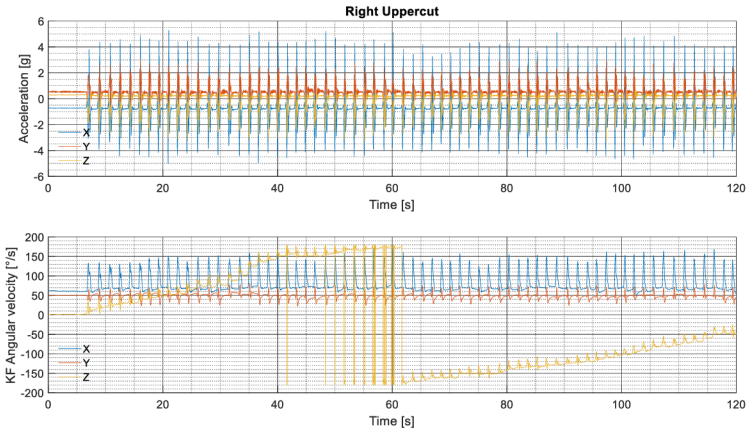


Figure A - 5: Recorded series of right uppercuts [32]

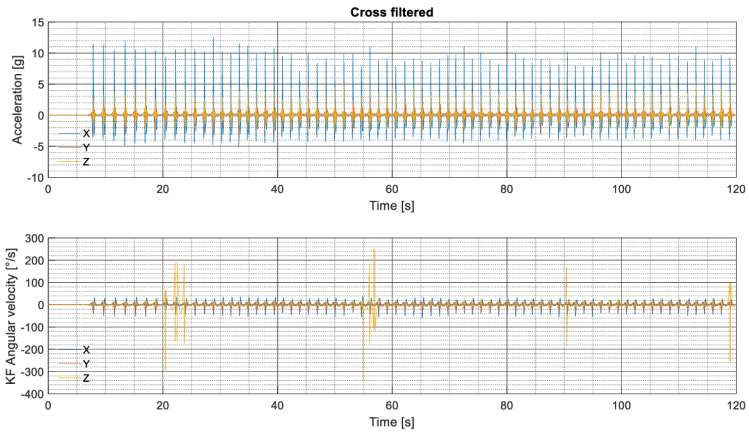


Figure A - 6: Series of cross punches after post-processing [32]

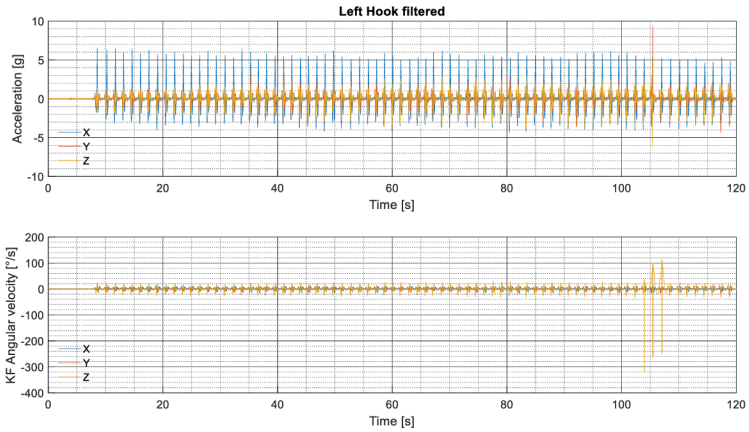


Figure A - 7: Series of left hooks after post-processing [32]

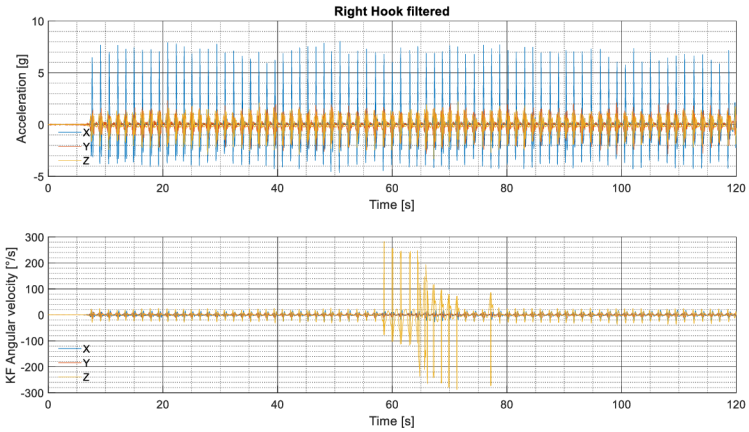


Figure A - 8: Series of right hooks after post-processing [32]

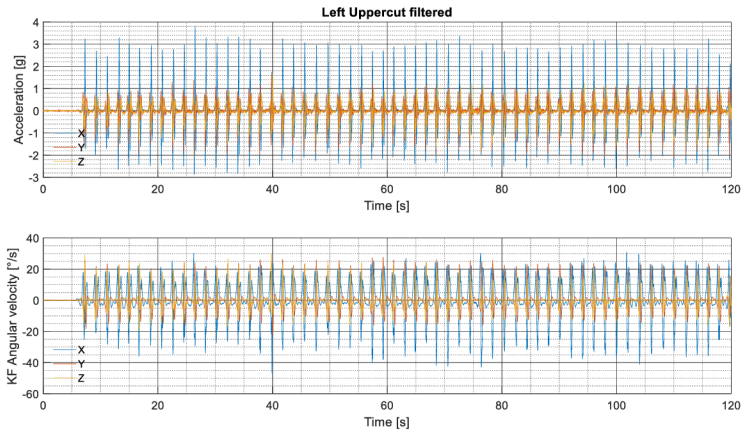


Figure A - 9: Series of left uppercuts after post-processing [32]

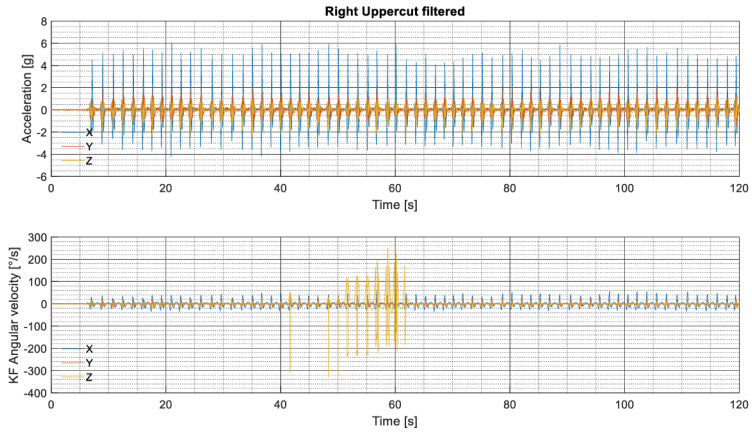


Figure A - 10: Series of right uppercuts after post-processing [32]

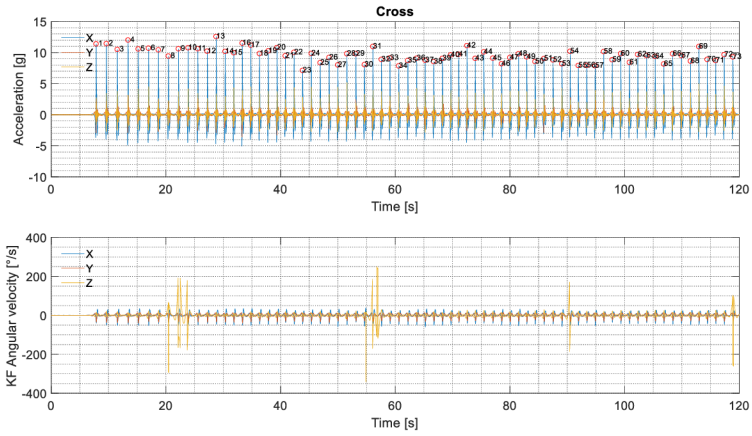


Figure A - 11: 73 peaks found in series of cross [32]

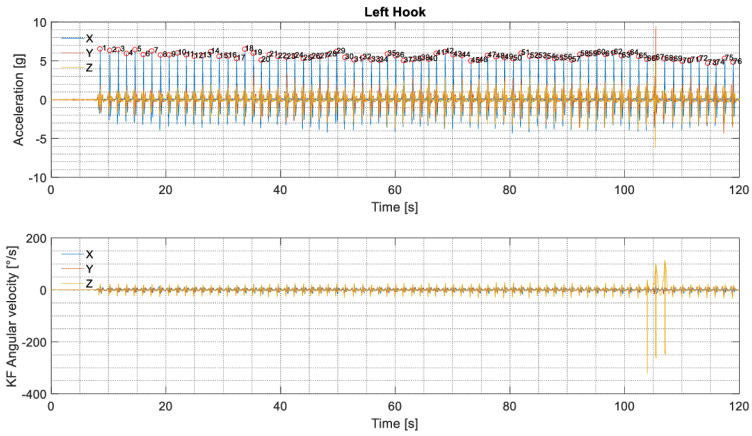


Figure A - 12: 76 peaks found in series of left hooks [32]

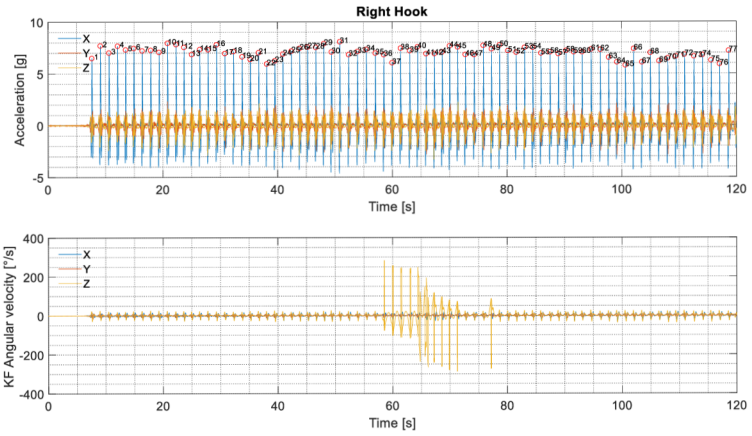


Figure A - 13: 77 peaks found in series of right hooks [32]

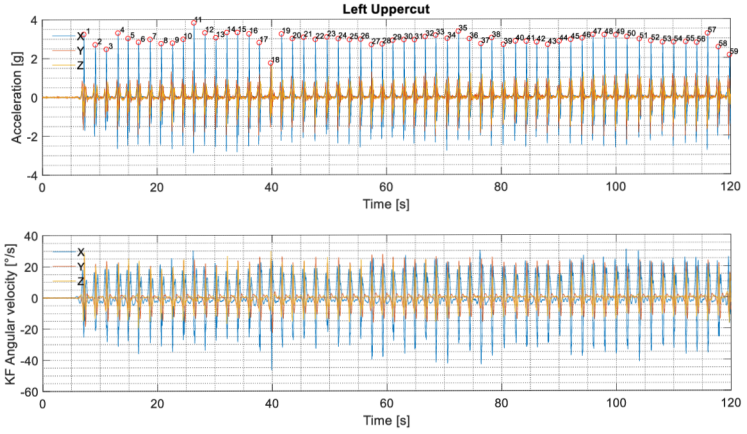


Figure A - 14: 59 peaks found in series of left uppercuts [32]

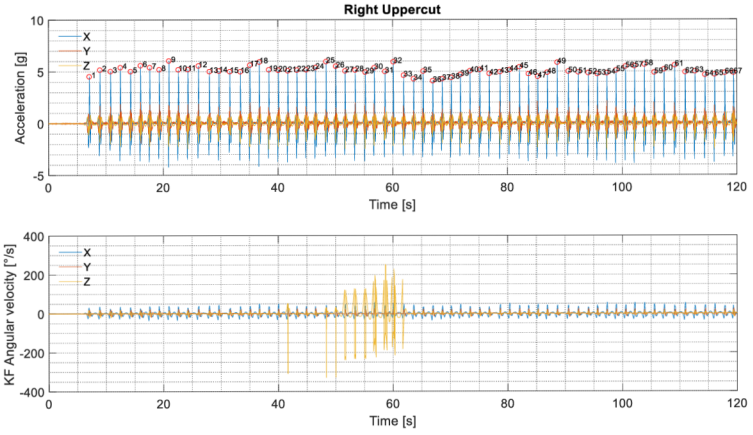


Figure A - 15: 67 peaks found in series of right uppercuts [32]