POLITECNICO DI TORINO

Department of ELECTRONICS AND TELECOMMUNICATIONS (DET)

Master's degree in MECHATRONIC ENGINEERING

Master's Degree Thesis

Machine Learning for Running Analysis with Wearable Inertial Sensors

Supervisors

CHIABERGE Marcello
ANGARANO Simone

Candidate

MERCURI Edoardo

July 2023

Abstract

Ground contact time and stride length analysis is crucial in high performance running. However, the equipment currently available in the market to perform such analysis, like optical sensors, is often only accessible to professional athletes due to their high cost and the complexity of installation.

This study aims to explore a cost-effective alternative to optical sensors by utilizing inexpensive inertial measurement units placed on the athlete's ankles, offering a wearable solution accessible to a larger audience.

A machine learning approach was used to address the problem, with the creation of a training set that associates accelerometer and gyroscope recordings when the athlete is running, to external measurements of ground contact time, obtained with pressure sensors in the insoles, and stride length, evaluated with a videocamera. The training set has been used to create a model that will predict these values solely based on the data from the inertial measurement units.

The results demonstrate the suitability of this type of data for a machine learning environment and, with the proper model training and training set, it is possible to achieve results comparable to those obtained using optical sensors.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Track and field competitions, like many other sports, make an extensive use of measurements. Taking in consideration a sprinting race, the winner is the one who take less time to cover a distance; in jumping competition the one who jumps the longest or the highest. But the measurement of a performance is not only performed during competitions: in every daily training it is important to keep track of variations and improvements to organise and adapt the training schedule.

Focusing on running, the best friend of a trainer is the stopwatch, but while taking the time of a repetition on 100 meters is enough for most athletes, there may be other factors a trainer would love to measure and consult, if given the possibility. This is why we can see Marcel Jacobs using additional equipment installed on the track during his trainings (figure 1.1): **foot-ground contact times** or stance duration, and analysis on **stride lenghts** is crucial for high performance athletes.

Figure 1.1: Marcel Jacobs, winner in the 100 meters in the Tokyo 2020 Olympic Games, on the starting blocks with optical sensors on the track

Ideally, the measurement equipment should not only not interfere to the movements of the athlete, but also take little time to setup and be cheap and accessible to as much people as possible.

1.1 Biomechanics of sprint running

Several studies have been done on motion analysis, however, the greatest number of studies focus on the analysis of walking or running at low speed while with regard to high speeds there are fewer articles found in the literature. Indicatively, the velocity associated with a walk is around 2 to 3 m/s, increasing the speed goes to jogging (4-5 m/s) and then up to the characteristic sprinter speeds that exceed, for elite athletes, 10 m/s.

The spatio-temporal parameters are derived from the running step cycle (figure 1.2). It is defined from the placing of the foot on the ground, the initial

contact, until the next placing of the same foot. Within this cycle there is a succession of two phases: stance and a swing phase.

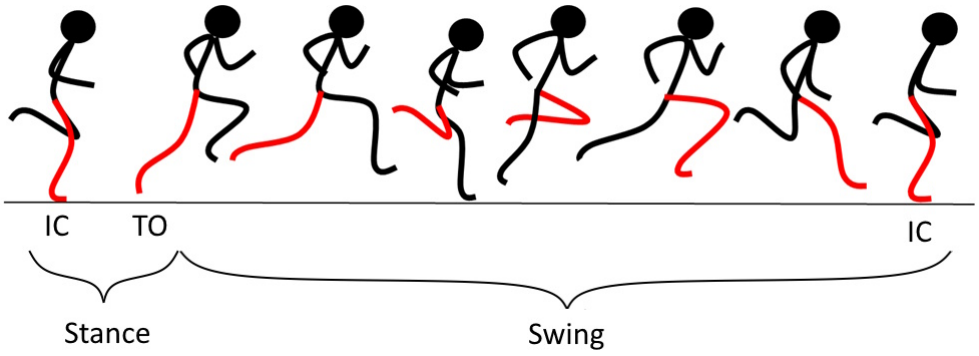

Figure 1.2: Step cycle scheme

In high-intensity running professional sprinters are able to minimize the braking phase in order not to lose the speed gained, unlike walking where the beginning of the stride cycle starts with the heel rest on the ground, thus with a relevant braking phase; to do this the foot's contact with the ground never occurs with the heel but with an area closer to the forefoot. The time and mode of contact of the foot on the ground is very important[1], because it allows the athlete to project forward in search of the highest possible speed: the propulsion phase, in fact, allows a more effective flight phase with a faster recovery of the free limb and with the knee high.

Important factors are stride frequency and stride amplitude, which are the parameters that make it possible to run faster, since the best compromise, between a high number of steps in the unit of time and a large distance covered per stride, leads to the best possible result for each athlete. Personal characteristics such as height, weight, quickness, and elasticity can also vary greatly among sprinters so different combinations of stride frequency and amplitude can lead to the same chronometric results for different athletes.

This thesis is going to focus on the measurements of the stance duration, or foot-ground contact times, and the distance between a foot lift off, and the next step, which is different from the distance covered by the swing phase of a single foot.

Chapter 2

State of the art

The gait analysis is a big subject and is mainly studied for medical purposes to detect pathologies[2], and is performed with different methods and sensors[3, 4].

All this methods share a controlled environment and low speed, which is not the typical case for sprint running, where the athlete wears spiked shoes and run on a track.

For this type of gait analysis the possible instrumentation adopted are **optical sensors** and wearable sensors like **clinical insoles** and **inertial measurement units**.

2.1 Optical sensors

Optical sensors in track and field events are often associated to photo-finish and distance measurements, but they can be employed for gait analysis.

This type of sensors are placed along the runway and one side transmit an array of infrared signals, and the bar on the other side has an array of receivers.

Figure 2.1: Optojump modular system (source www.https://training.microgate.it/it/prodotti/optojump-next/il-sistema-modulare)

When the feet hit the ground they obscure some transmitters from the receivers allowing to measure the ground contact time and stride length. The high sampling rate of 1000 Hz and sensor placement at 1,04 cm guarantee a very high resolution in both the measurements.

As mentioned in the introduction, this type of sensors are very expensive, in the range of tens of thousands of Euros, for a 10-20 meters installation.

2.2 Clinical insoles

Clinical insoles exploit a high number of pressure sensors to determine, among other gait parameters, the ground contact time. These are highly used for walking gait analysis, but can be employed for sprinting too. A commercially available option provide 150 Hz sampling rate, so a 6.67 ms in ground contact time resolution.

As experienced directly in this thesis, this solution is not the best in terms of user experience and reliability, because the insole do not stay fixed in place when sprinting multiple times and this inconvenience may alter the gait and the parameters detection.

2.3 Inertial measurement units

Inertial measurement units (referred as **IMUs** from this point forward) are sensors that are very lightweight and relatively cheap.

These devices are often a cluster of different sensors, including tri-axial accelerometer and tri-axial gyroscope, which are related to movement as they track respectively the acceleration and angular velocity.

Many different algorithms have been developed in order to interpret this type of data, but were developed either with high sampling rate devices, and therefore very costly, or in non-sprinting pace and do not represent the real world conditions on track.

2.3.1 Gait features assessment using gryoscope data

Yonatan Hutabarat et al.[5, 6] propose a heuristic-threshold based algorithm applied to the gyroscope data of two IMUs with 148 hz sampling rate to detect various gait events: their results are relatively good for either temporal and spatial features of gait, as they registered an error of 4.22 ± 15.48 ms and -8.31 ± 21.02 ms (mean$\pm$ standard deviation) in initial and final contact measurements, and an overestimation of 7.72 cm in the stride length against a force plate gold standard.

The top speed analysed is 10 km/h, which is way below the speed an average athlete can reach.

2.3.2 Sprinting ground contact times with high sampling rate IMUs

Using relatively high sampling rate with IMUs, it is possible to find some patterns in the acceleration and gyroscope data. Brendan Purcell et al.[7] used 250 Hz IMUs and achieved an error of 0 ± 12 ms, -2 ± 3 ms, and -1 ± 1 ms (mean$\pm$ standard deviation) in jog, run and sprint conditions respectively, when compared to a force plate runway gold standard.

Marcus Schmidt et Al.[8] validated this method using 1 kHz IMUs with the Optojump gold standard, obtaining similar results.

Chapter 3

Machine Learning approach

Machine Learning is referred to the method of programming an algorithm without explicitly programming it: given some data and a task, the computer develops a model that can perform such task.

Two of the main tasks it can perform are **classification** and **regression**: these are supervised tasks, which means that the training data you feed to the algorithm includes the desired solutions, called **labels**.

In the case of classification, the model must **sort** the given data in a number of predefined classes; an example can be, given a set of images with a single shape drawn on it, to recognise if the shape is a triangle, a square or a circle, sorting the data in three classes.

In a regression instead, the model must predict a **value** given a set of features: an example can be to predict the price of a car based on mileage, age, and brand.

Exploiting these tools it may be possible to extrapolate parameters of performance of an athlete using indirect measurements and data. Humans (better if they are trainers) actually do it all the time: they analyse visual data coming from their eyes and give feedback to the athlete, but having a numerical data on parameters which cannot be measured easily is a clear advantage.

This approach has already been explored and resulted **successful in walking gait analysis**[9, 10], and in this thesis aims to apply it to sprinting gait.

3.1 Model training and type

The data that will be used to train the machine is called "dataset", which is a collection of samples and corresponding labels. Many datasets are available online for many different purposes, but in this case it will be created from the ground up.

Once the dataset is ready it is then divided into a **training and a test set**: the model is trained using the training set and then it is **evaluated** comparing its predictions on the test set.

To avoid the risk of overfitting, a portion of the training set is designated as **validation set**. During the training the model is constantly evaluated also on the validation set, in this way it knows if it is becoming too specialized at predicting the training set samples, but not as good with new samples.

A **regression** approach may be the correct way to tackle the problem: the features provided may be some data collected during the analysed event, and the value to predict is a **continuous output**. Although that does not mean that some classifications steps may be employed for a more complex model.

Among the various regression models available, the state of the art suggest the use of **Deep Neural Networks** and **Convolutional Neural Networks** for this type of task.

After this preliminary planning, the following step is to chose a set of measurable parameters to assemble a dataset from which extrapolate the ground contact time and the stride length.

3.2 Dataset creation - data collection

A possible equipment that satisfy the characteristics of the equipment defined in chapter 1 is the employment of **inertial measurement units** as they are very lightweight and are relatively cheap.

They can be placed on both ankles, so they are **unobtrusive** to the natural movements of the athlete, and can track relatively closely the **movements of the feet**: they will not measure directly the ground contact time or the stride length of each step, but their data will be used to extrapolate those measurements.

It is theoretically possible to get a position relative to a predefined origin through geometrical considerations and double integration of the acceleration, although it is not advisable, as the low sampling rate yield too much error.[11]

Mounting the IMUs on the ankle of a running person it is possible to see that every step is different, but has a somewhat recognisable shape: for example

when the foot lands on the ground there are very big fluctuation in the vertical axis of the accelerometer (figure 3.1), and every time a leg overtake the one on the ground there is a peak in the mediolateral axis of the gyroscope (figure 3.2).

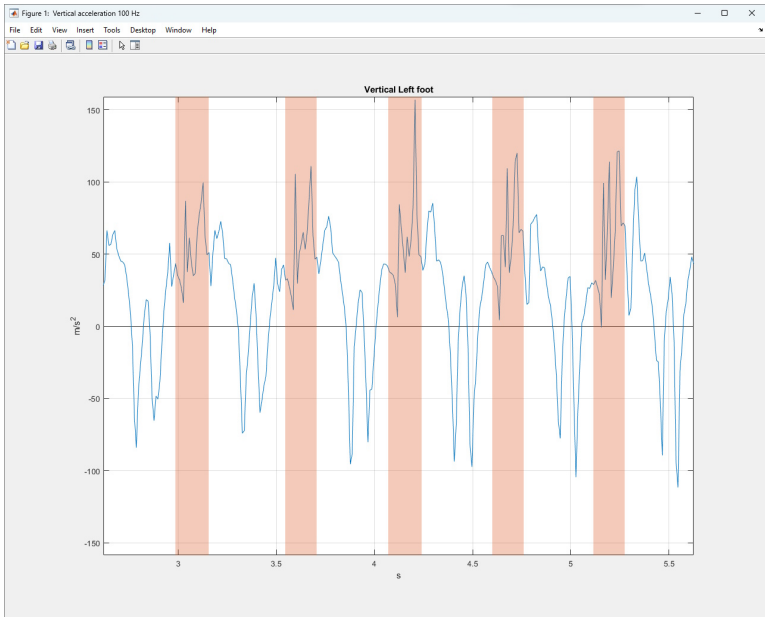

Figure 3.1: Acceleration readings in the vertical axis at 100Hz; highlighted in red are the moments in which the foot is touching the ground, data obtained using pressure insoles

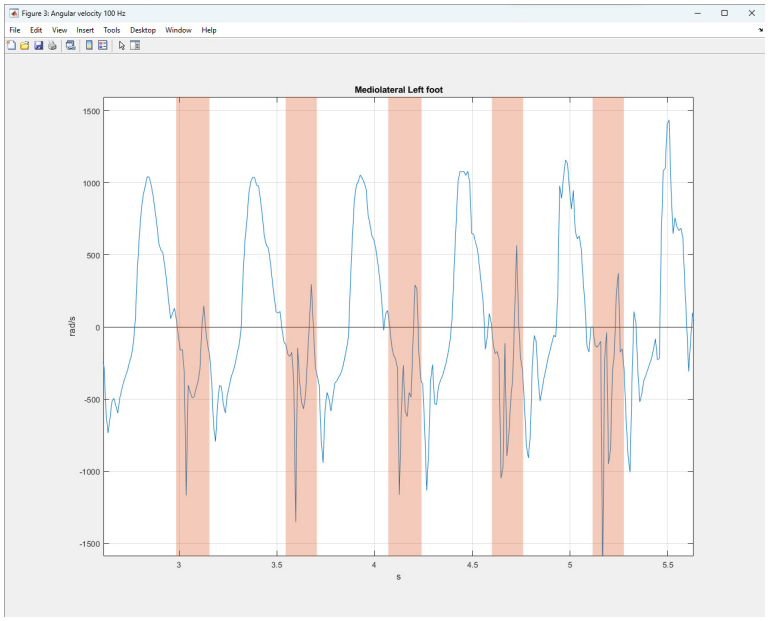

Figure 3.2: Angular velocity readings in the mediolateral axis at 100Hz; highlighted in red are the moments in which the foot is touching the ground, data obtained using pressure insoles

These patterns can be spotted and exploited in a machine learning environment, where a model can be trained to recognise that the values recorded by the sensors correspond to a certain ground contact time and a stride length.

3.3 Dataset creation - labels collection

The IMUs provide the data to be trained, but what about the labels? Additional equipment is needed to provide the target values, on top of the IMUs measurement.

The additional equipment must provide contact time and stride lengths, so the **optical sensors** highlighted in chapter 2 are actually perfect, but unfortunately those were not available.

Another somewhat available option is an **instrumented treadmill**, which can provide both contact time and stride length for each step; the downside of using a treadmill is that it cannot be used in conjunction with *spiked shoes* (which are mandatory for high intensity running trials).

Lastly a combination of a **video-camera** and **pressure sensors in the shoes** to get respectively **spatial** and **temporal** data of the trials; this is the setup that was used in this thesis.
The camera is responsible of determining the positions of each step using pixel counting, while pressure insoles are inside the shoes of the athlete which determine the moments in which the foot hit the ground and when it takes off with pressure sensors.

All systems needs to be **synchronised** in order to have a correspondence in time of IMUs data and external measurements: the video-camera can be synchronised with a visual indication in the video recording while the pressure insoles can exploit internal clock synchronisation in conjunction with an external trigger.

In the end, two different dataset are created: the first comprehend the samples in the form of **time windows of data from the IMUs in a neighbourhood of when the foot is on the ground**, and its labels are the **ground contact times measured by the pressure insoles**; and the second comprehend the samples in the form of **time windows of data from the IMUs in a neighbourhood of the moments between a step and the next one** and its labels are the **stride lengths calculated after the elaborations of the video recordings**.

The bigger the datasets, the more accurately the model will be able to recognise the target values.

Chapter 4

Materials

4.1 ProMove-mini - IMUs system

4.1.1 Inertial Measurement Units

Inertial measurements are acquired using the ProMove-mini IMU system. From the ProMove-mini user manual [12] "The default system consists of a number of ProMove-mini sensor nodes, the Advanced Inertia Gateway and Inertia Studio (PC Software) for monitoring and logging the inertial data. The standard setup is depicted in Figure 4.1. The sensor nodes communicate wirelessly with the gateway in the 2.4 GHz ISM band. The gateway is connected through the mini-USB connector or the Ethernet connector with a PC that runs Inertia Studio."

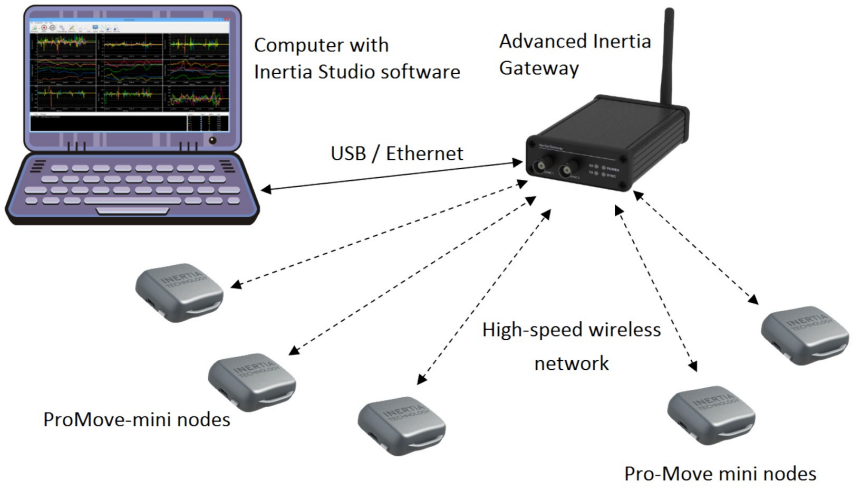

Figure 4.1: ProMove-mini example setup

The nodes have an internal configuration controlled by the Inertia Studio software. The configuration specify which of the multiple sensors inside the

nodes are active, which one are disabled, and their sampling frequency.
The active sensors are:

- Accelerometer
- Accelerometer for high g
- Gyroscope

And their specifications are the summed up in table 4.1.

Tri-axial accelerometer	
Range	Selectable: ±2, ±4, ±8, ±16g
Resolution	62 μg @ ±2 g range
Sampling rate	Up to 1000 Hz
Tri-axial gyroscope	
Range	Selectable: ±250, ±500, ±1000, ±2000 °/s
Resolution	0.007 °/s @ ±250 °/s range
Sampling rate	Up to 1000 Hz
Tri-axial accelerometer for high g	
Range	Selectable: ±100, ±200, ±400g
Resolution	49 mg @ ±100 g range
Sampling rate	Up to 1000 Hz

Table 4.1: Inertia node sensors' specifications

Software settings

The acquisition frequency is set to 1000 Hz, the data willi be downsampled to 100 Hz in the processing of the data.

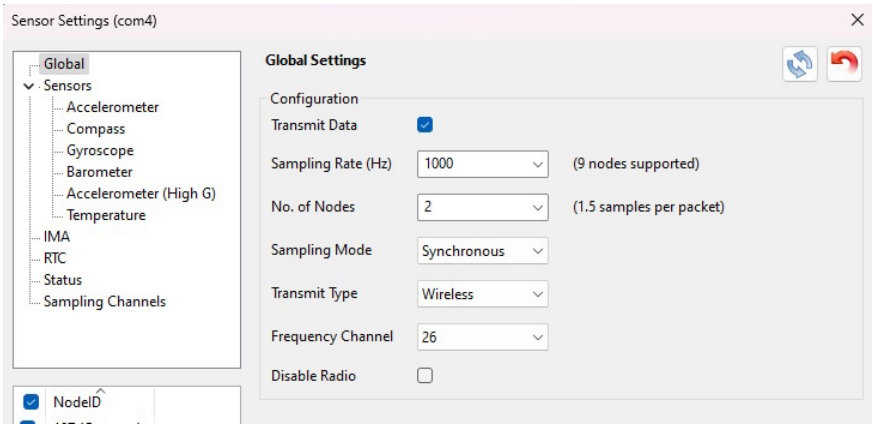

Figure 4.2: Inertia Studio's software settings, global settings

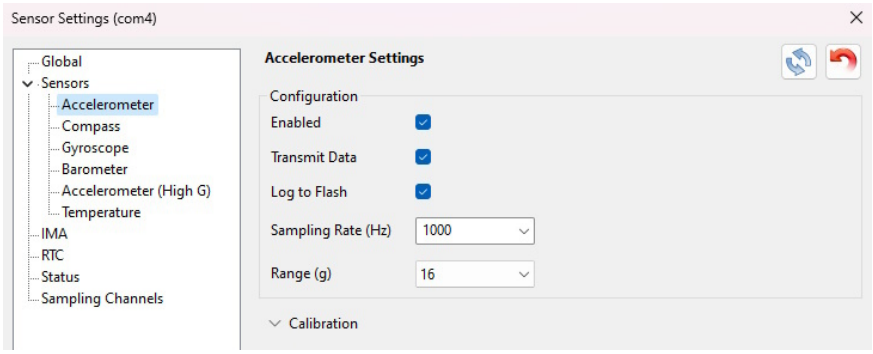

Figure 4.3: Inertia Studio's software settings, accelerometer settings

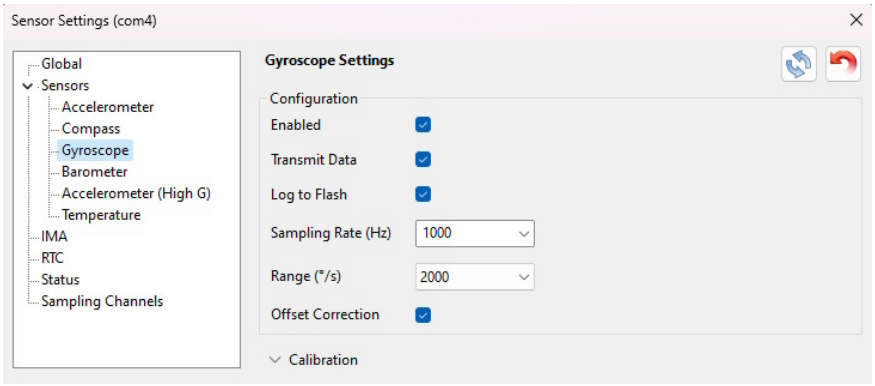

Figure 4.4: Inertia Studio's software settings, gyroscope settings

The accelerometer for high g will not practically be used, nonetheless is active in case further studies find this data to be useful.

4.1.2 Wireless gateway

The Inertia Gateway is connected and powered via USB to the PC. The gateway is the element that enable the synchronization of the data between the different systems thanks to its I/O ports that can be set to high or low signals via software.

The port 1 is connected to an external **LED**, and to a **sensor** as shown in figure 4.5.

Figure 4.5: Gateway connection to PC

Figure 4.6: LED placement

The LED is connected to the right side of the BNC splitter and is placed in the field of view of the camera: a high value on the port 1 can be detected in the video footage when the LED turns on; at the same time this change of state (rising edge) is recorded in the IMUs log, allowing to **synchronise the IMUs data to the video**.

The other sensor connected to the left side has an internal clock synchro-

nised to the insoles system, and every time it registers rising or falling edge in the signal it saves a timestamp, and is used in later elaborations to **cut and elaborate the insole data**.

Since they are connected to the same port, once the video is cut upon the LED turn on detection **every data is synchronised**.

The I/O configuration in the Inertia Studio software is shown in figure 4.7. The output trigger is fired manually on the software during acquisitions, since automatic actions on start and stop of flash logging is not reliable due to logging lag.

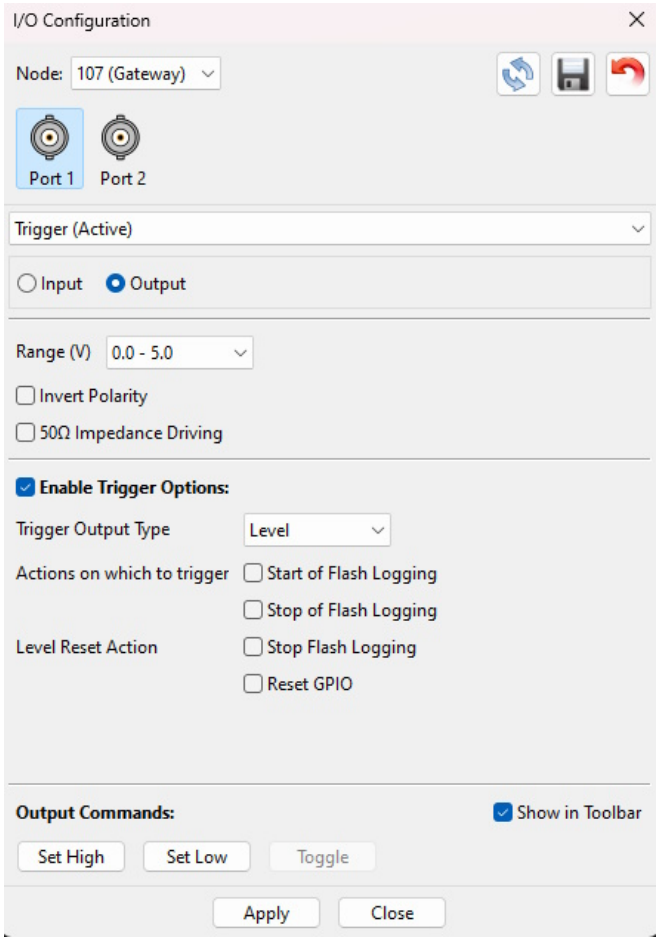

Figure 4.7: Inertia Studio's software settings, I/O ports configuration

4.2 INDIP system - pressure insoles

To provide the **ground contact time of each step** it employed the INDIP[13][14] system.

The complete INDIP system is a combination of magneto-inertial and distance sensors connected to instrumented insoles with pressure sensors. These sensors operate synchronously and save their readings in a on-board storage.

The pressure insoles (figure 4.8) are responsible for the ground contact times; they host 16 pressure sensors operating at 100 Hz sampling frequency. These sensing elements are based on force sensing resistor, which exhibit a resistance which is inversely proportional to the force that is applied to them. Gait events are identified exploiting the sensor redundancy and a cluster-based strategy.

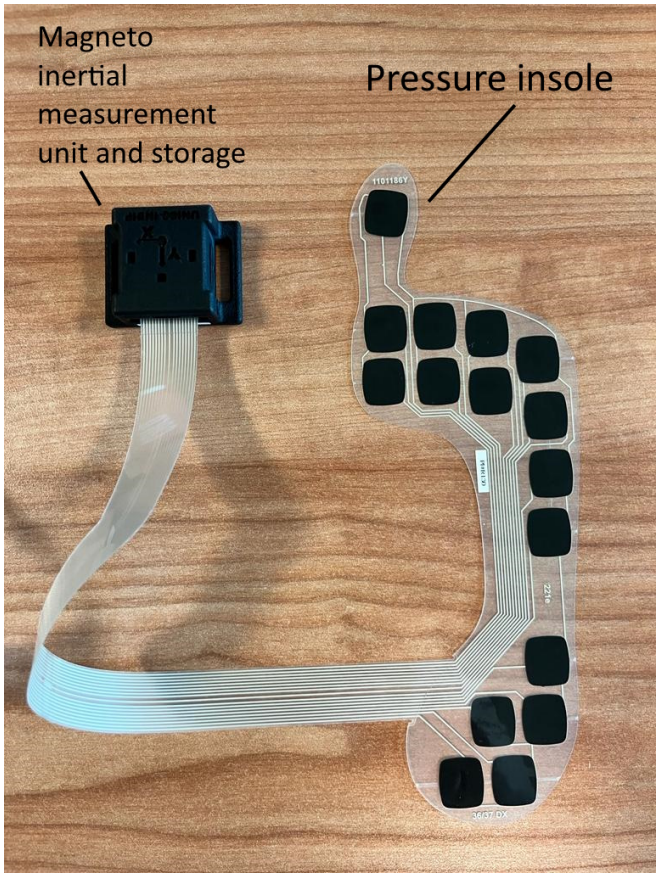

Figure 4.8: Pressure insole and magneto inertial measurement unit of the INDIP system

This system is the golden standard for the ground contact time, which is associated to the inertial movement. However, the proposed standard resolution of 0.01 s not ideal for high performance athletes, for comparison high end optical sensors have a resolution of 0.001 s.

For this reason the system firmware has been modified to achieve a **sampling frequency of 200 Hz**, by reducing the number of **active pressure sensors to 8**.

4.3 Camera

The camera used to determine the position of the steps during the acquisitions is the GoPro HERO10, which allows to record videos at 240Hz, 2.7k (2704 by 1520 pixels) resolution at a wide angle.

Figure 4.9: GoPro HERO10 (source www.gopro.com)

Figure 4.10: Camera placement with respect to the LED and the gateway

Chapter 5

Methods

In order to get a meaningful amount of data, estimated around 2000 samples, to submit to the model for training, it has been chosen to test **13 different subjects**, each running **50 meters for 15 times**.

Subject	Gender (M/F)	Age (y.o.)	Height (cm)	Weight (kg)
1	M	25	180	68
2	M	25	174	66
3	F	23	166	49
4	M	20	182	65
5	F	22	162	48
6	M	22	178	78
7	M	22	184	71
8	F	21	188	65
9	M	19	178	74
10	M	24	171	71
11	M	20	182	82
12	M	33	183	76
13	M	21	181	74
Avg.	77% M	22.8	177.6	68.2
Std.	-	3.4	7.2	9.7

Table 5.1: Summary of participants for track acquisitions

For each trial around 19 steps are sampled, so with a simple multiplication the acquired samples should be around 3705. This is not the case due to unreliability of IMUs connection, camera recordings, insoles faultiness or simply less trials performed by the athlete. So the overall final datasets are smaller:

- 1676 samples for contact time;
- 2216 samples for stride length.

Subject	CT[a] trials collected	% weight on the CT model	SL[b] trials collected	% weight on the SL model
1	8	8.4	10	6.0
2	7	7.4	10	6.0
3	10	10.5	12	7.2
4	10	10.5	15	9.0
5	10	10.5	15	9.0
6	10	10.5	15	9.0
7	-	-	7	4.2
8	-	-	8	4.8
9	-	-	15	9.0
10	4	4.2	15	9.0
11	14	14.7	15	9.0
12	12	12.6	15	9.0
13	10	10.5	15	9.0

Table 5.2: Summary of acquired data for each subject

[a] Ground Contact Time
[b] Stride Length

5.1 Acquisition setup

The athletes run for 50 meters, but only the last 40 meters are framed by the camera to get a reasonably defined image of the athlete during the trial: the camera is placed in the middle at 20 meters and displaced transversally by 15 meters.
The camera points at the center where there is an indicator with a cone.

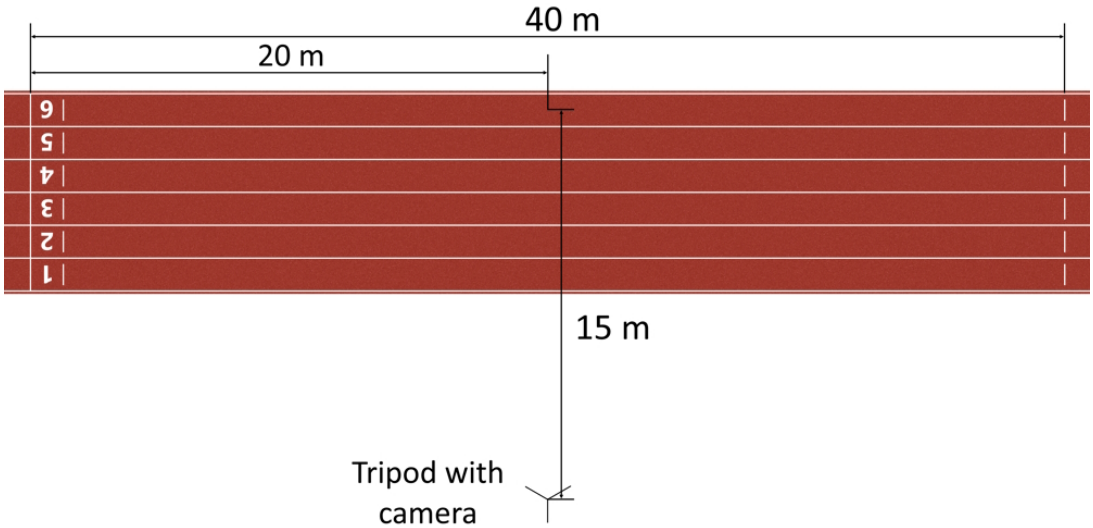

Figure 5.1: Camera and cone placement scheme

The indicator at 20 meters is used in the video post processing in order to center and correct the image (figure 5.2), while the LED is connected to the Pro-Move base to get the trigger signal of every trial: this light is detected in the video elaboration and is used to sync the video with the other sensors.

Figure 5.2: Camera and cone placement

The pressured insoles must be connected and are the first to start the acquisitions; one of these sensors is connected to the ProMove base to get and register the trigger signal of every trial, used to sync the pressured soles' data to the other sensors and video.

The ProMove sensors are placed on the outer sides of the ankles as shown in figure 5.3.

Figure 5.3: Sensors on the athlete

5.2 Per-run operations and data collection timeline

A set of actions performed in a specific order is needed to get the data from the sensors and camera.
As already mentioned, the pressured soles start acquiring data at the start of the acquisitions and they are not turned off until the end, while the camera and the ProMove nodes can acquire data just when actually needed during the trials.

When the athlete is ready to start a trial the camera and the ProMove sensor start the acquisition; it is not needed to start simultaneously, and the recording of the IMUs start when the athlete is near the gateway in order to ensure that the "start recording" command is received correctly. The athlete can now go to the 50 meters mark on the track, and when ready the camera starts recording.

The next step is to provide a way to synchronize the data: the synchronization is performed using the I/O port of the ProMove Gateway, initiated via software.

When the output trigger is set to "High" the ProMove software register the timestamp to the log; the output is connected to the sensor of the pressured soles and the LED: the sensor register its trigger timestamp, while the LED is an indicator for the video.
This trigger is used to reset every timer and get the same start.

After the 40 meters run, the output trigger is set to "Low" and after that the camera and ProMove sensors can stop recording: also in this case we wait for the athlete to be near the gateway to ensure that the "Stop recording" command is received correctly.

The video would be processed from the LED detection to its end.

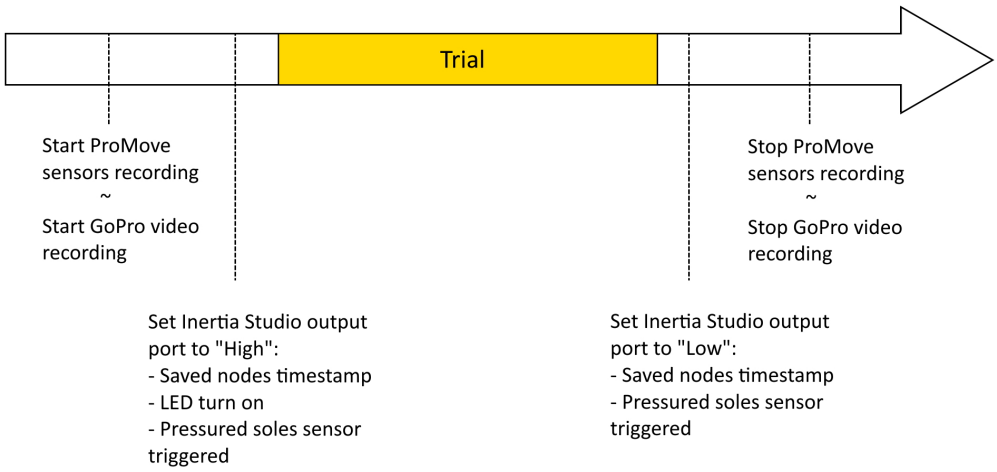

Figure 5.4: Per-run operations timeline

5.3 Raw data collection and organization

At the end of the acquisition the data is gathered in 3 types of file:

- a `00X_results.mat` file containing a structure where each field contains the data of one run in its own sub structure: here there are, initial and final timestamps of the contacts of the foot on the ground obtained by the pressured soles;
- a set of `RunX.mat` files for each athlete containing the acceleration and gyroscope measurements with their corresponding timestamps
- a set of `RunX.mp4` video files for each athlete

5.3.1 Insoles and IMUs data integrity check

A **preliminary visualization** of the acquired data is done in MatLab[15] and the source code can be found in the GitHub repository[16]: for every trial summary graph of the accelerations and angular velocities is shown, overlapped with the pressured insoles contacts.

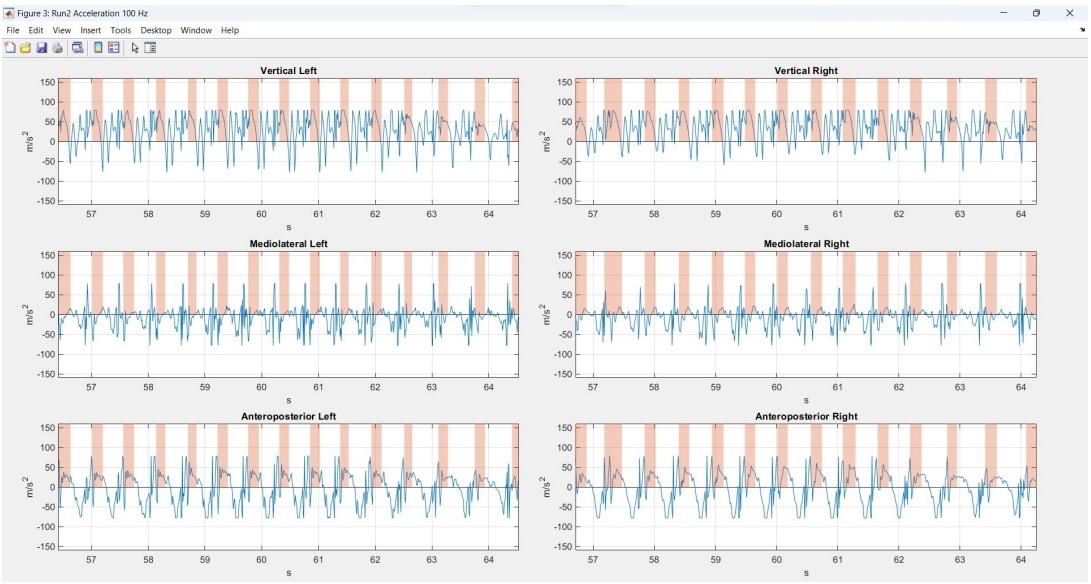

Figure 5.5: Every window shows the left and right foot measurements on the three axes, in blue IMUs data, in red the contacts detected by the insoles

If any problem arise with one of these two systems the run gets discarded for the contact time dataset creation, for example if the IMUs is missing some data or the insoles contact times measurements are blatantly wrong like in the case of right and left foot contacts overlapping.

5.3.2 Axes orientation mismatch

Since the IMUs are oriented in a different manner depending on which ankle are placed, as shown in figure 5.6, the data needs to be changed in sign in some of the axis of one of the two feet in order to get a similar behaviour of the data in similar movements of the feet.

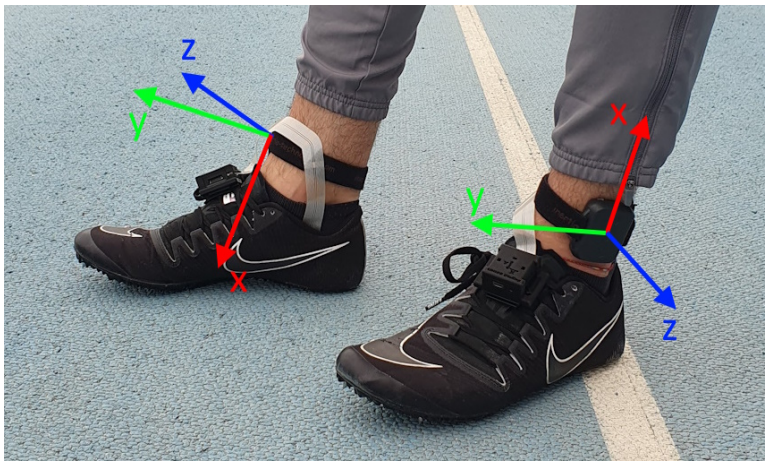

Figure 5.6: Axes orientation of the IMUs

For what concernes **accelerations** only x axis (also called **vertical** axis) needs to be changed, while the y (**anteroposterior** axis) axis is always towards the front from the athlete, and the z axis (**mediolateral** axis) is always towards outside from the athlete; the latter have opposite absolute directions but this is intended as only in this way the two sides have a similar behaviour.
For the **gyroscope**, conversely, the mediolateral and the anteroposterior axes need to change sign in one foot: in order to have positive angular velocity in both feet when the tiptoe of the foot rotates upwards the left foot axis needs to be changed in sign, and the same reasoning can be made for the anteroposterior axes.

Problems like in figure 5.7 where the IMUs data started logging after the start due to a connection error can be detected and that particular trial gets discarded.

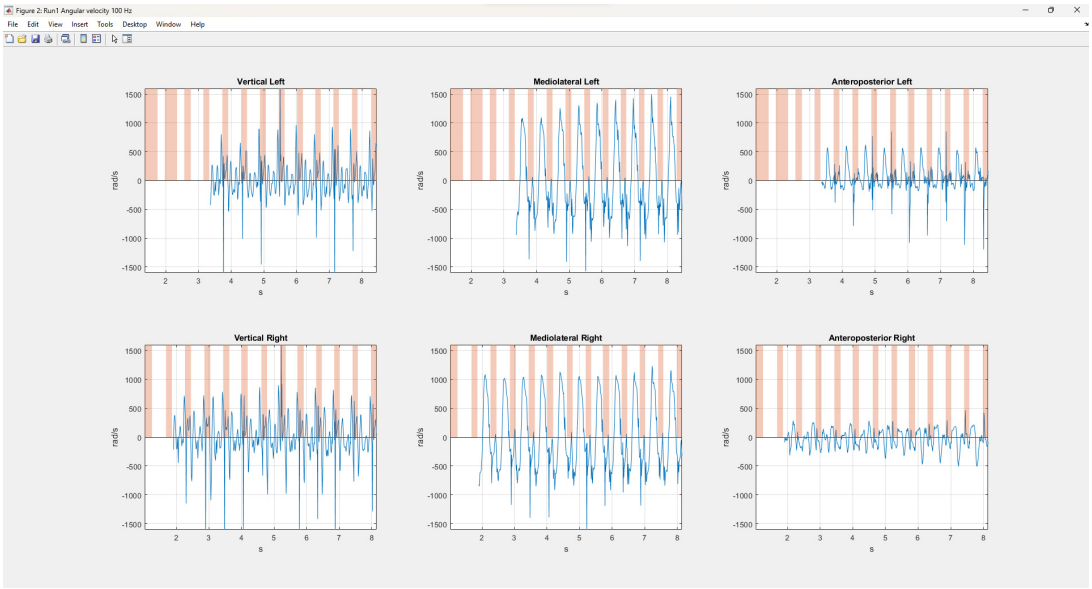

Figure 5.7: IMUs and insoles data visualisation of subject n° 5, Run1

5.4 Contact time dataset

A dataset needs to have a large list of samples, all of which are tied to a corresponding value called "label".

In this specific case the sample is the **data from the IMUs in the neighbourhood of the contact time of the foot on the ground**, and the corresponding label is the **duration of the contact** of the foot on the ground in seconds, given from the insoles data.

Since a-priori it is not possible to know when a contact take place only from the data of the IMUs it is needed to find a way to **isolate each step** by just looking the IMUs data: the adopted solution is to exploit the **positive local maxima in the data of the mediolateral axis of the gyroscope**.

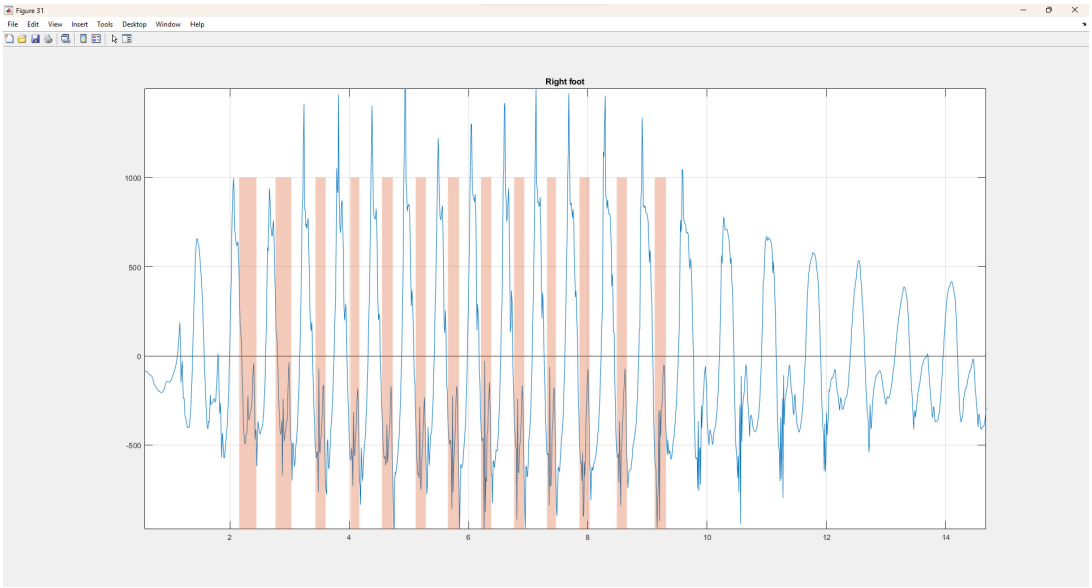

Figure 5.8: Data of the right foot, in blue the mediolateral axis of the gyroscope, the areas in red are the contacts detected by the insoles

As it is shown in figure 5.8 the contacts almost systematically falls in between two positive peaks of the mediolateral axis of the gyroscope, and it has been developed a method to reliably include the data around the contact:

- filter the signal to remove the local maxima with a moving average filter; chosen periodicity value is 6;
- filter through a positive threshold, enough high to take only high intensity steps; chosen threshold is 400;
- identify the middle point in time between the peaks;
- save the IMUs neighbourhood of fixed size around these middle points; chosen 500 ms wide.

The processed data and identified middle points are shown in figure 5.9.

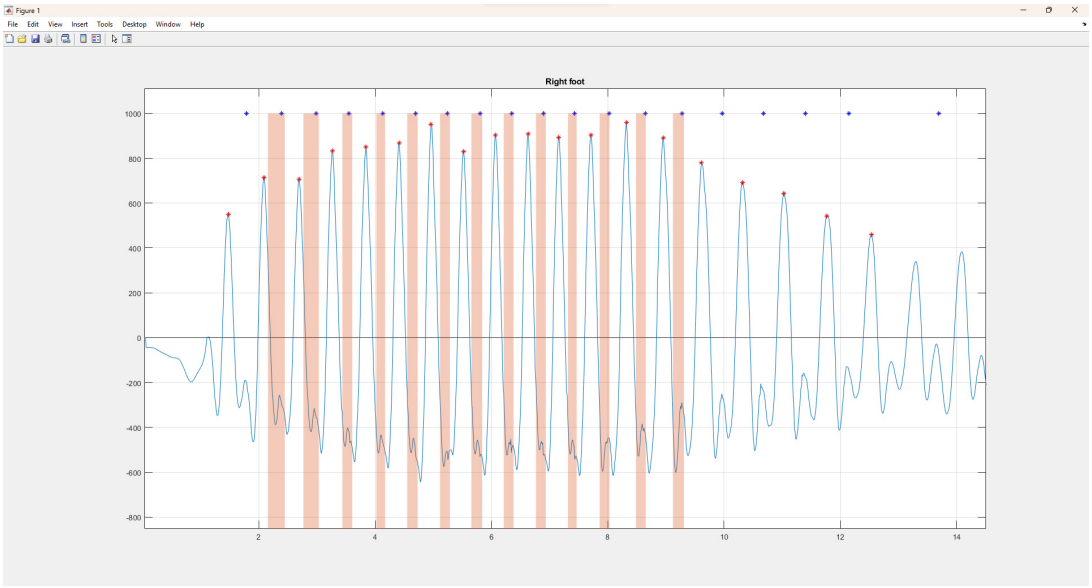

Figure 5.9: Data of the right foot, in blue the mediolateral axis of the gyroscope filtered using a moving average filter; the red asterisks are the local maxima; the blue asterisks are the middle points between the local maxima; the areas in red are the contacts detected by the insoles

This is a simple method to identify and isolate the steps. From this identification it is needed to create samples with coherent size in terms of matrix dimension, expanding from the middle points.

The data of the insoles is used to only save the neighbourhoods of which there is a contact time to associate to the sample.

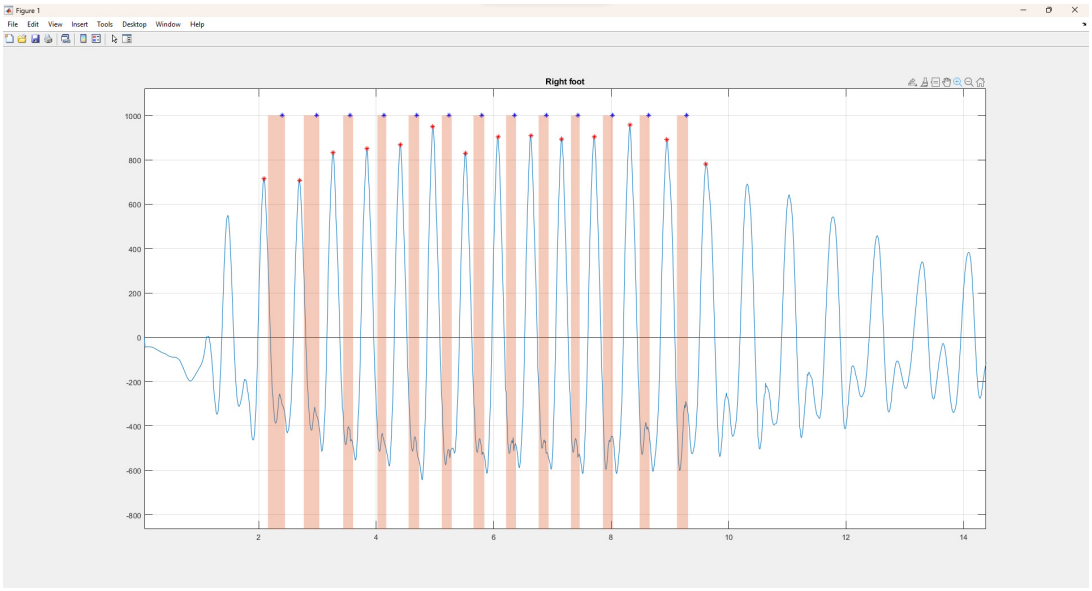

Figure 5.10: Kept middle points after filtering

In figure 5.11 it can be seen that the final saved "windows" of the mediolateral axis of the gyroscope is wide enough to comprehend the time in which the foot is on the ground: the chosen width of these windows is 400 ms.

The final sample is a collection of all the 6 axes.

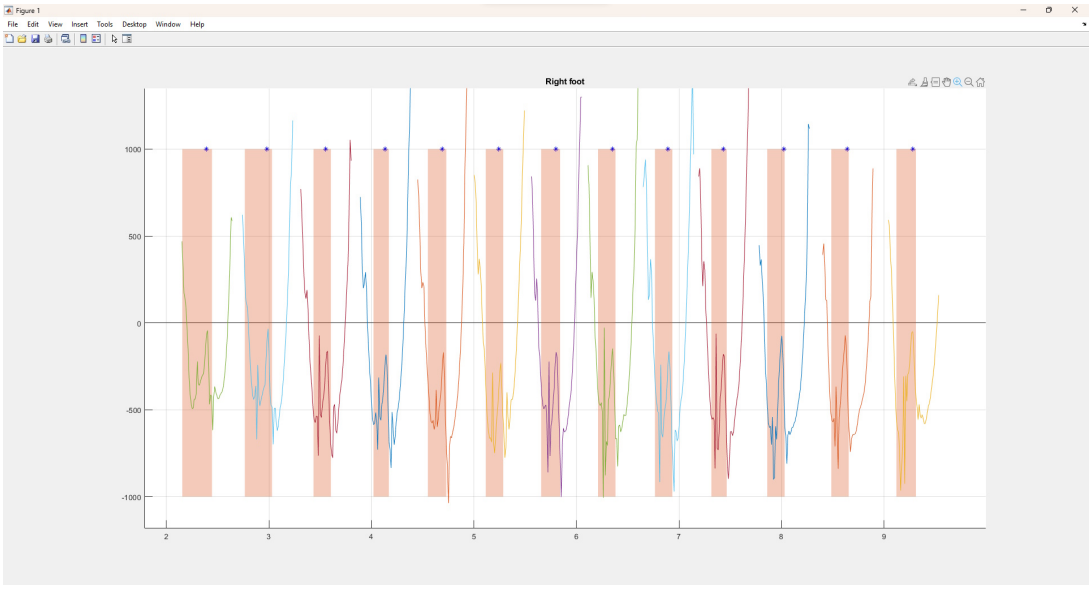

Figure 5.11: Saved mediolateral axis data of the gyroscope for each sample

The source code can be found in the GitHub repository[16] and the final dataset has **1676 samples**.

5.5 Video elaboration

The video recording of each run is used to determine the position of each step and, therefore, the **stride length**. The GoPro HERO10 can shoot videos up to 240 frame per second at a wide angle, but the image is affected by **barrel distortion**, the typical look of a wide angle lens (figure 5.12), and for this reason is not possible to just do a simple pixel to meters proportion. Furthermore the video needs to be synchronised to the IMUs data in order to be sure to have the correct association of the stride length.

Figure 5.12: Typical look of a wide angle lens, straight lines are bent

To achieve these tasks it has been developed a Python[17] script: this environment allow the employment of different libraries to solve all the issues, like OpenCV[18] which provide a large amount of functions for image and video transformation, and MediaPipe[19] which allow to identify of a person in an image, and by extension the positions of the feet.

The source code can be found in the GitHub repository[16]

5.5.1 Python script outline

Start - LED detection area

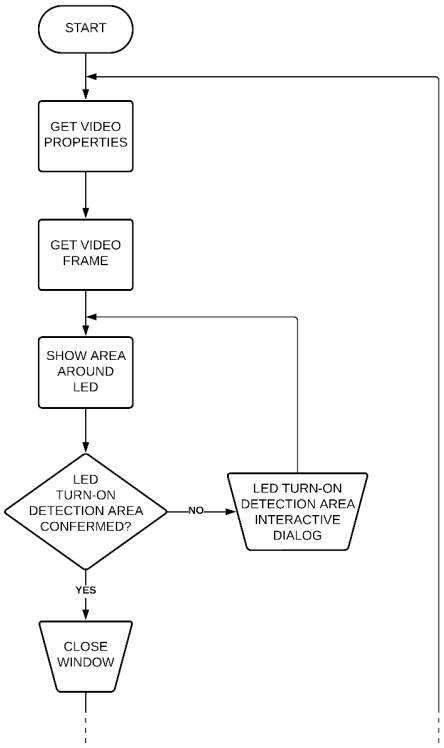

Figure 5.13: Part 1 of flow chart of video elaboration

Before the video gets undistorted and subjected to pose detection there are some preliminary steps.
The LED turn-on detection is performed on specified coordinates, but since from run to run there may be little movements of the camera, this area needs to be checked. It is possible to click on the image, the coordinates selected become the new center of the detection area, in this way it is possible to center it on the LED.

LED turn-on detection - synchronisation

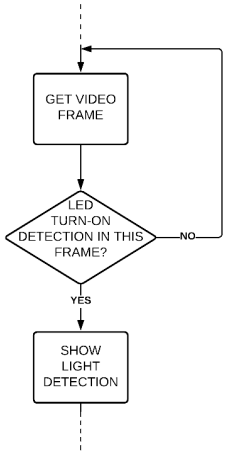

Figure 5.14: Part 4 of flow chart of video elaboration

Start of the LED turn-on detection, each new frame gets compared to the starting frame where the LED is off: the difference of the color values filtered with a threshold is used to determine when the LED turns on. Only after this frame the video is elaborated and saved in new files, so they are synchronised with the IMUs' and the insoles' data.

Video leveling

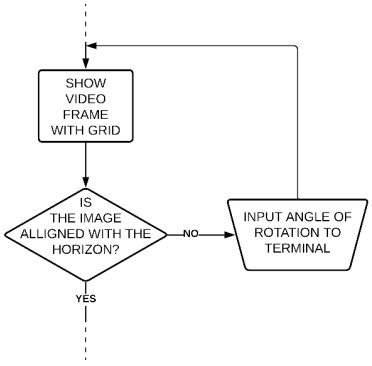

Figure 5.15: Part 2 of flow chart of video elaboration

The next step is to make sure that the video is leveled on the horizon, for this reason it is shown a frame of the video with a grind overlapped: on the terminal is possible to input an angle of rotation; if the input is valid (a float number) a new rotated image is shown, if it is leveled, press enter on the terminal to go on.

Runway center coordinates

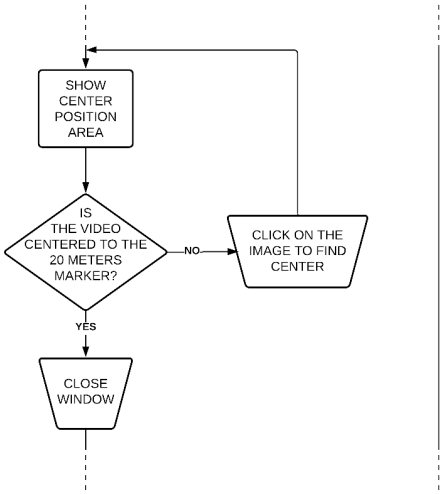

Figure 5.16: Part 3 of flow chart of video elaboration

Now an area around the center of the image is shown, and as with the LED turn-on area check, it is possible to click on the image to center it on the marker placed at 20 meters from the start. This action is needed to get the coordinates of the center of the runway for the correction of the barrel distortion.

Video elaboration

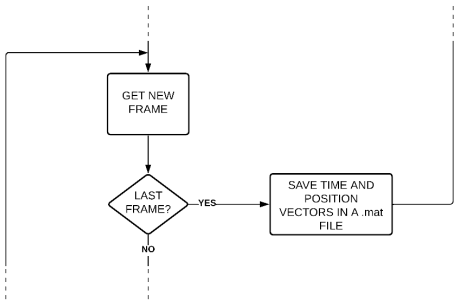

Figure 5.17: Part 5 of flow chart of video elaboration

Start of loop where every remaining frame of the video gets elaborated. After the last frame, a new video file go through the preliminary steps again.

Correct barrrel distorsion

The image gets rotated by the angle specified in the preliminary steps, then, in order to reduce the workload on the successive steps, the image gets cropped to just keep the central horizontal portion where the athlete is running.

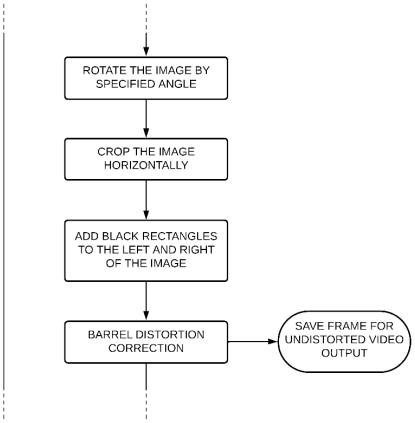

Figure 5.18: Part 6 of flow chart of video elaboration

Two black borders are added on the sides of the image in foresight of the next step: the image gets corrected of its lens distortion and the portions on the outer sides are elongated, therefore without adding the borders the image would have been cropped, removing some of the captured runway.

The final result can be see in figure 5.19, and the image is saved to be part of a new video file.

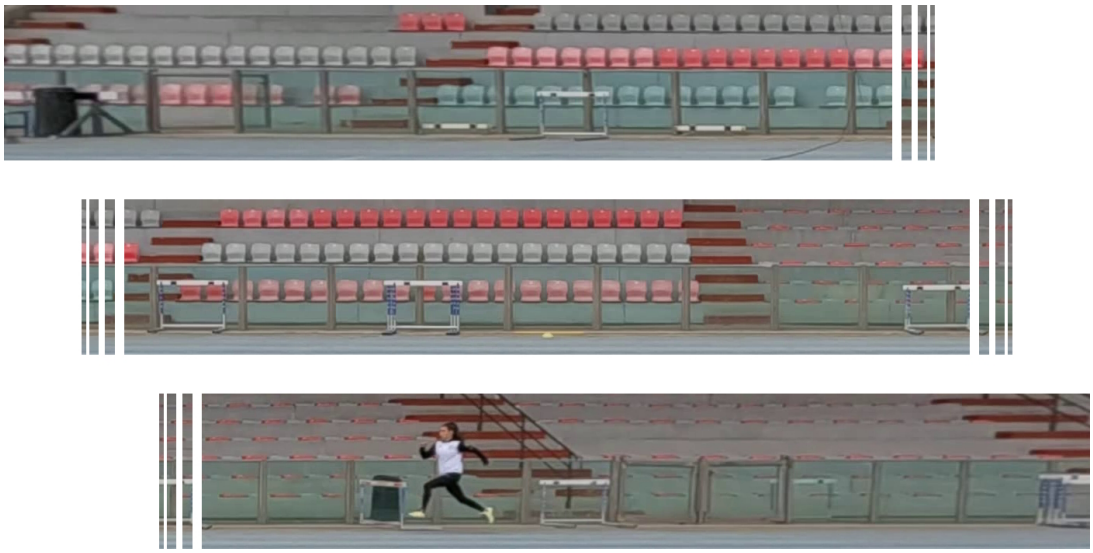

Figure 5.19: Cropped and lens corrected image, for displaying reasons it has been divided in three

After the lens correction a set number of pixels reflect the same real world length, so it is now possible to do a simple pixel to distance proportion to calculate the stride length. Given x as stride length in pixels, and y as stride length in meters:

$$y = \frac{x \cdot 40m}{3424px} \tag{5.1}$$

where 3424 is the image width, which is showing 40 meters.

Pose detection

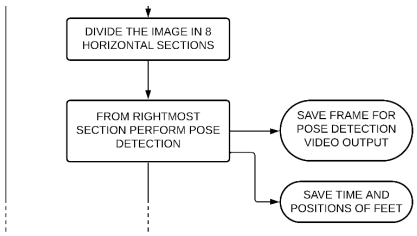

Figure 5.20: Part 7 of flow chart of video elaboration

Now it is possible to the determine the position of each steps using the MediaPipe[19] Python library, but since the athlete is so small compared to the image dimensions it is needed to crop it: the method used is to take a portion of the starting image which is 8 times smaller in width, then perform pose detection starting from right border (figure 5.21), and save each image to a new video file.

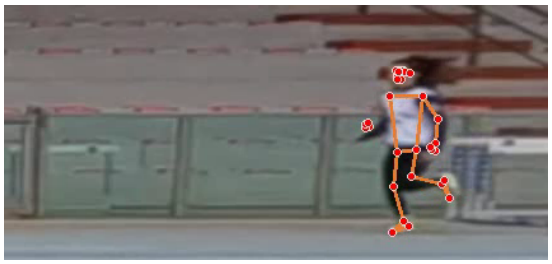

Figure 5.21: One of the first frames of the pose detection

The output of the pose detection is the position in percentage of the image dimensions for each node of the body. This values are scaled back by the image dimensions to have the position in pixels in the current section, and based on the current section position with respect to the original one, it can be scaled back to the coordinates of the whole 3424 pixels image and therefore have also the position in meters using the equation 5.1.

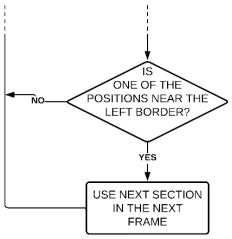

Figure 5.22: Part 8 of flow chart of video elaboration

Once the athlete enters the image, the pose detection will start and when one of the foot is near the left border, the detection area will shift to the left keeping half of the current image (figure 5.23).

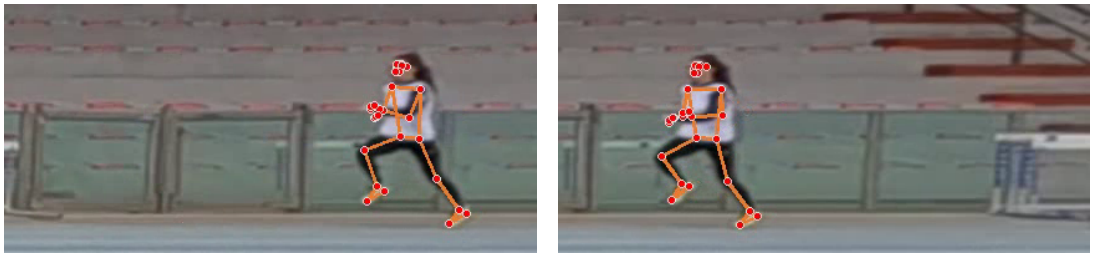

Figure 5.23: Chronologically saved from right to left, the athlete is still inside the image before and after the section position change

At the end of the video is is shown a graph of the detected positions in time (figure 5.24), this is just to check if everything went well, and the vectors containing the positions and timing pairs are saved in a *.mat* file, to use it in further elaborations.

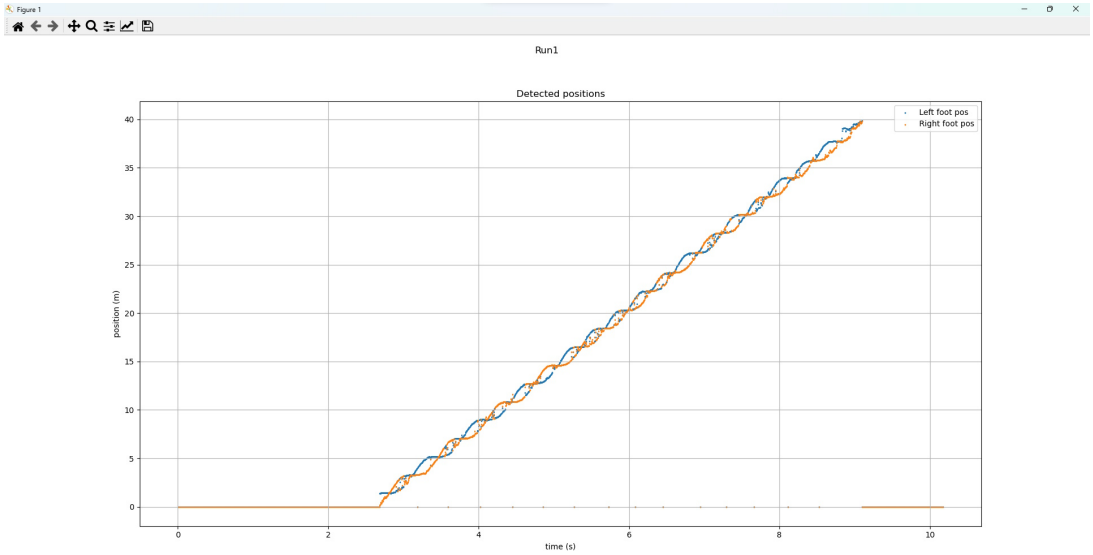

Figure 5.24: Graph showing feet positions detected in time

5.6 Steps positions check

The pose detection using MediaPipe is just a starting point to speed up the process of the manual selection of the feet positions. In fact another MatLab script

has been employed to interact visually with the data and correct the positions.

Using the data from the insoles combined with the positions given by the pose detection it is possible to show a frame of the athlete with a foot on the ground for each step, and adjust the position clicking on the tip of the foot.

In case of missing data from the insoles it is possible to locate the contacts on the ground by just looking ad the position graph, so even without contact time information is still possible to preserve the stride length data.

The source code can be found in the GitHub repository[16]

5.6.1 MatLab script outline

Steps positions through insoles timings

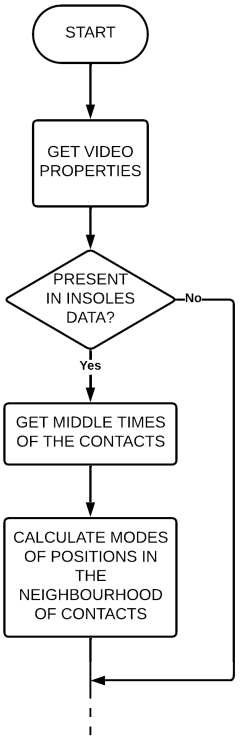

Figure 5.25: Part 1 of flow chart of position check

Each lens corrected video (figure 5.19) will be used to check the position of each step, and the first action is to check if there is data present from the insoles: if it is present a vector is created containing the average of the initial and final contacts of each step; this time coordinate is used to perform a mode on the feet positions neighbourhood to find the step position.

The mode is used because the pose detection often confuses left and right feet, but as can be seen in figure 5.26, when a feet is on the ground an horizontal line can be seen on the graph and the value on the y axis is position of the foot: the mode performed on both the right and left foot positions, with a tolerance of 2 cm; this ensures that the most stable position is chosen, regardless of which foot is on the ground.

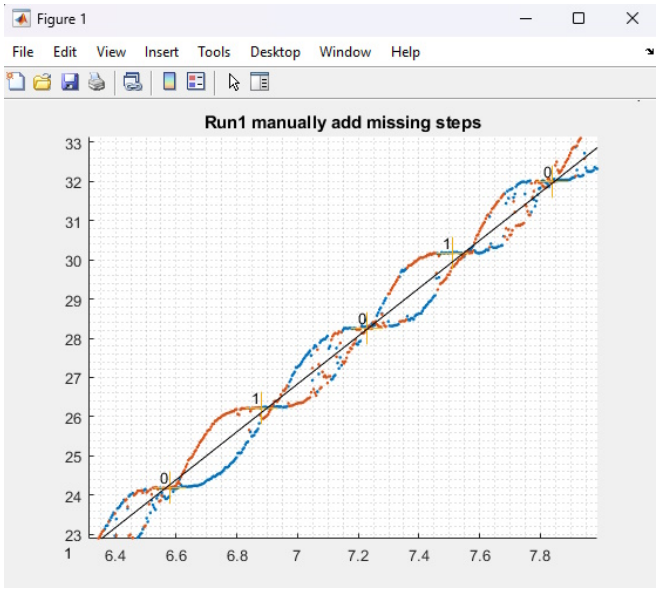

Figure 5.26: In orange: right foot positions; in blue: left foot positions; in yellow: steps positions calculated using mode on a neighbourhood wide 40 values, the label is 0 for right foot, and 1 for left foot; in black, linear regression of the steps

Add missing steps

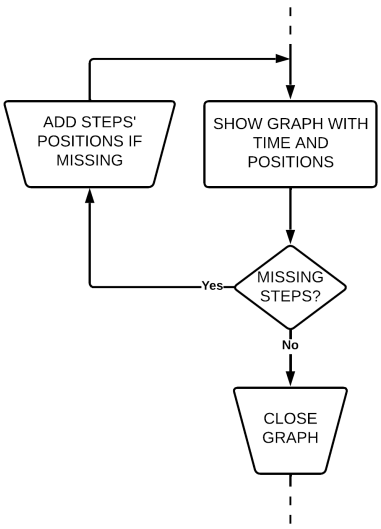

Figure 5.27: Part 2 of flow chart of position check

After this initial determination of the steps position, since the insoles may have missing values at the start or the end, it is needed to add missing steps. As we can see in figure 5.28, the last recognisable step before the 9^{th} second it has not been marked, so the script wait for a click in that area: it is important to get the right position only on the x axis in this stage, since the position will be checked later, and to do so it is needed to click on the crossing of the right and left feet, as shown in figure 5.29. Also the black line is a linear regression of the previous steps, and it is displayed as a guide for the new steps

to add manually.

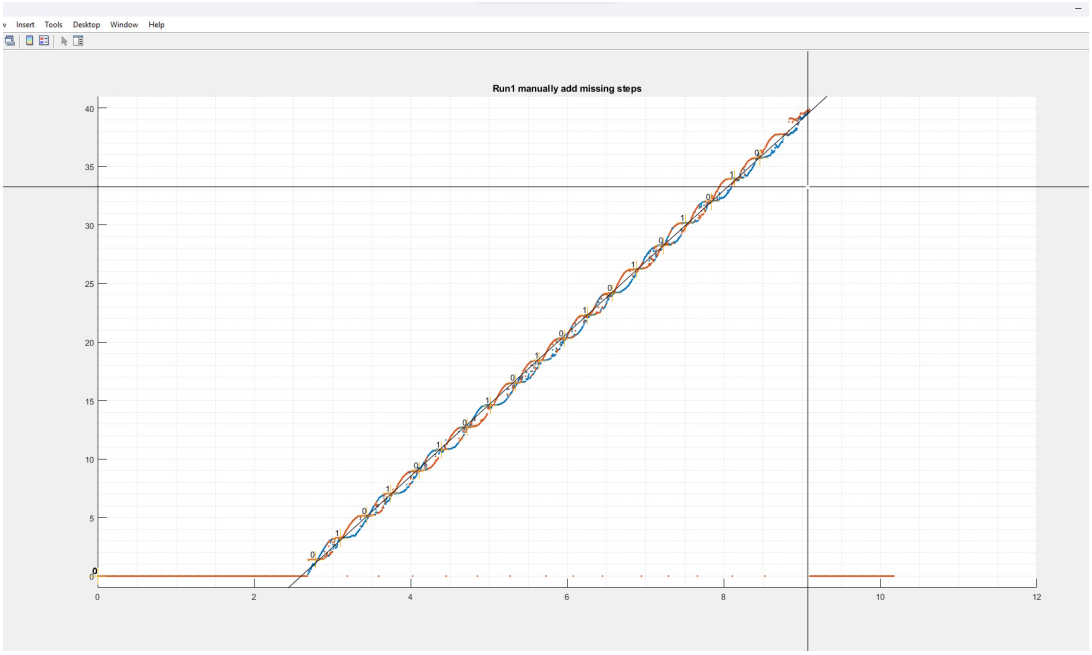

Figure 5.28: The interface is waiting for click input

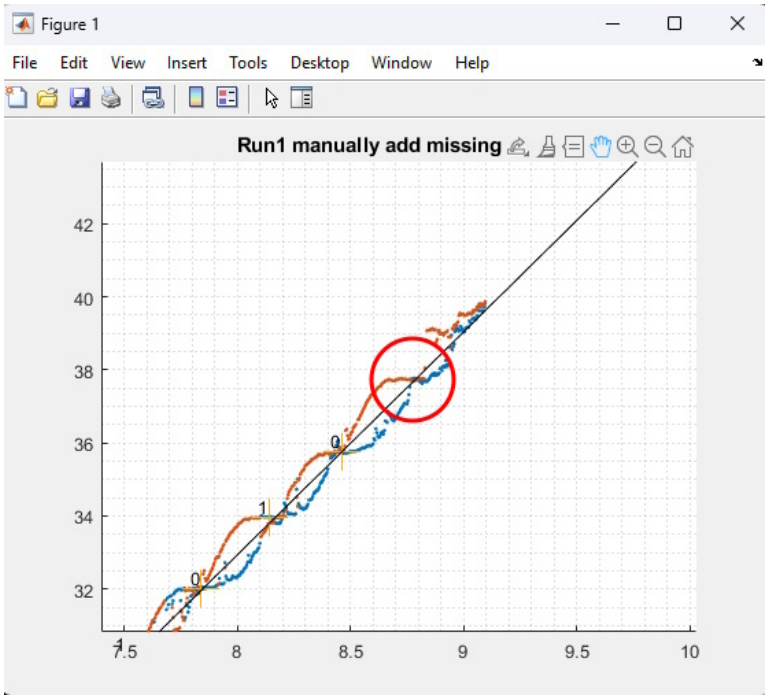

Figure 5.29: Zoom in around the 9th second

In the case of missing insoles information, this interface is presented right away, and all the steps needs to be input chronologically.

Side assignment

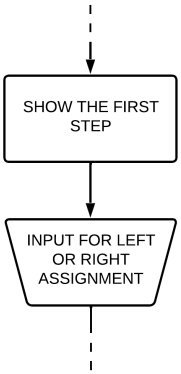

Figure 5.30: Part 3 of flow chart of position check

Since some of the first steps may be missing form the insoles data, or missing altogether, it is needed to assign again which foot has taken the steps. To do this the first step is shown, and clicking on right or the left side of the image it is assigned to the first step, and the other are assigned alternating left and right proceding chronologically.

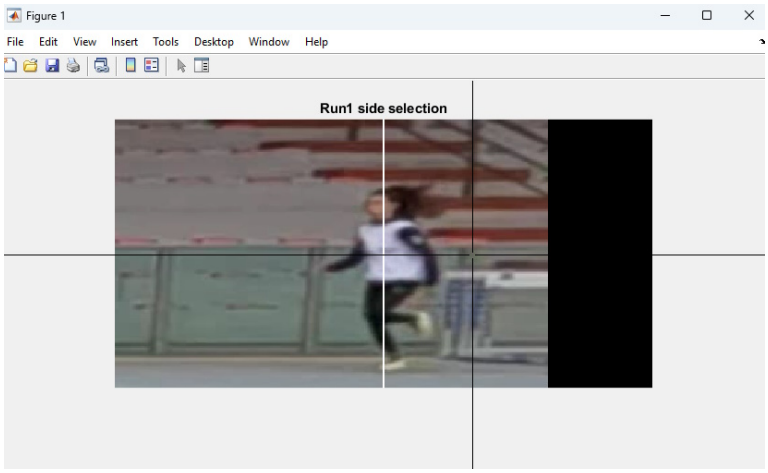

Figure 5.31: First step shown, the interface is waiting for a click input, in this case it is a right foot step, so the click will have to be on the right half side

After the side selection, a recap graph is shown with all the steps selected with a label associated with it: a "0" if it is a rigth step, a "1" if it is a left step, which as can be seen in figure 5.32, are alternated.

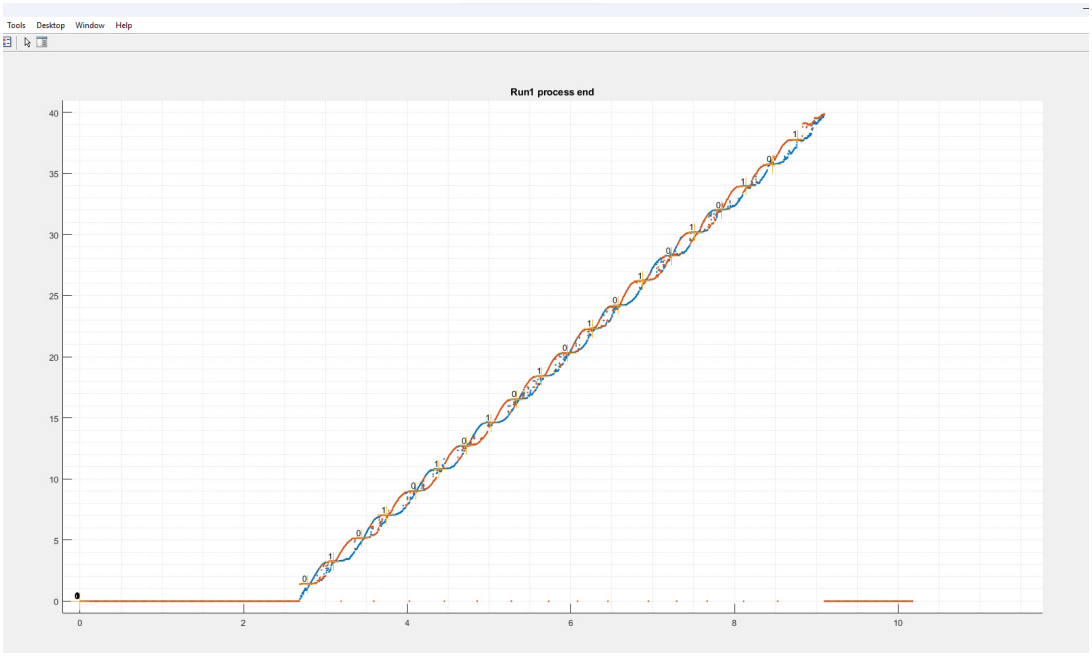

Figure 5.32: Recap graph of the selected steps and side assignment

Steps position check

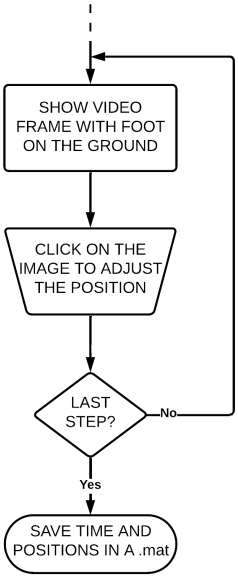

Figure 5.33: Part 4 of flow chart of position check

For each saved step is is shown an image taken from the lens corrected video: it is centred on the position of the step and a white line show the current saved value; now the interface wait for a click input to correct the position, or close it to confirm and procede to the next step. In figure 5.34 it is shown the first step and the cursor is placed where the correct position needs to be; the vertical coordinate is irrelevant, the position needs to be adjusted horizontally.

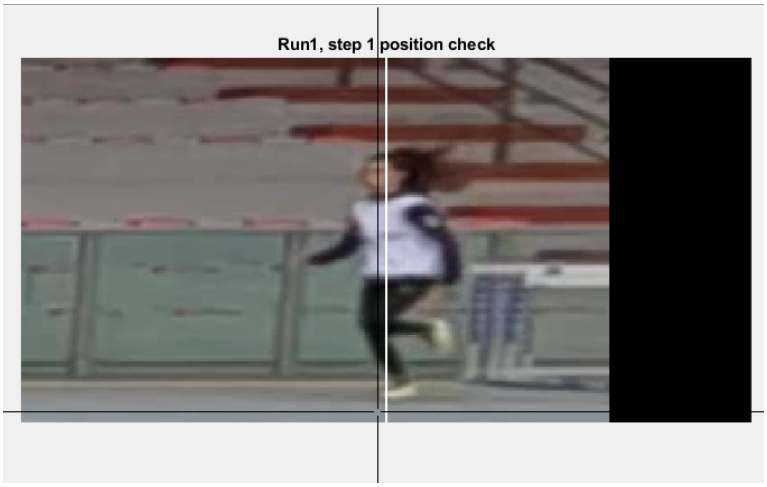

Figure 5.34: Original position shown, the interface is waiting for an input

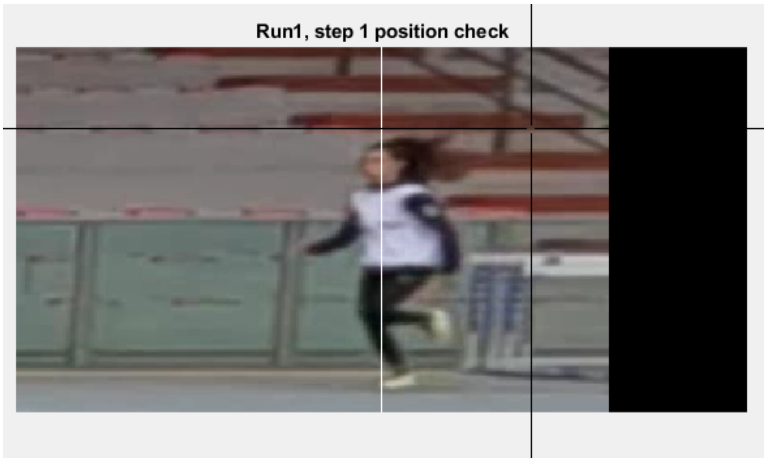

Figure 5.35: Image shown after clicking. It is still waiting for an input

After clicking, the same image but centered on the new position will be shown (figure 5.35). It is still waiting for an input but if the new position is correct it is needed to just close the window to make the next step check interface appear.

After all the steps and all the trials are processed, the times positions and sides of the steps are saved in three matrices and stored in a *.mat* file for further elaborations.

5.7 Stride length dataset

Similar to the contact time dataset, the stride length dataset needs to associate a large number of samples to their labels. In this case the samples are the IMUs data from when a foot lift off from the ground, to when the other touches the ground, and the labels are the stride lengths.

An addition label is assigned to the samples, marking if the lift off foot is the right, marked as "0" or the left one, marked as "1".

Figure 5.36: The stride length is the distance between two subsequent steps

Since both of the feet are interested in this running action (which is not the case in the ground contact time), both feet data will be part of the sample.

The general idea of isolating the steps using the mediolateral axis of the gyroscope is the same as the contact time dataset, but the elaboration differs in the last steps:

- filter the signal to get only big local maxima with a moving average filter; chosen periodicity value is 6;
- filter through a positive threshold, enough high to take only high intensity steps; chosen threshold is 400;
- merge left and right local maxima and find the middle point in time between both left and right local maxima;
- save the IMUs neighbourhood of fixed size around these middle points; chosen 500 ms wide.

The last steps are different because as it can be seen in figure 5.37, now each local maxima falls in the proximity of the contact of the other foot, so in order to find a suitable middle point, the average time between two local maxima of both feet has been calculated for the IMUs neighbourhood data selection middle point (figure 5.38).

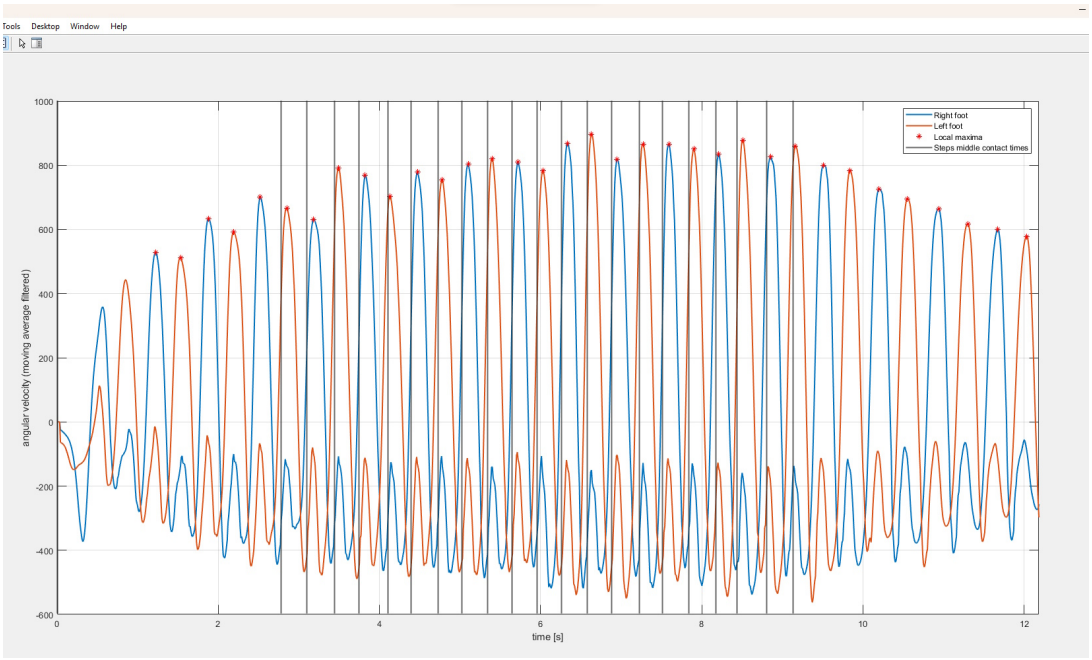

Figure 5.37: The local maxima are in the proximity of the middle point of contacts of the feet

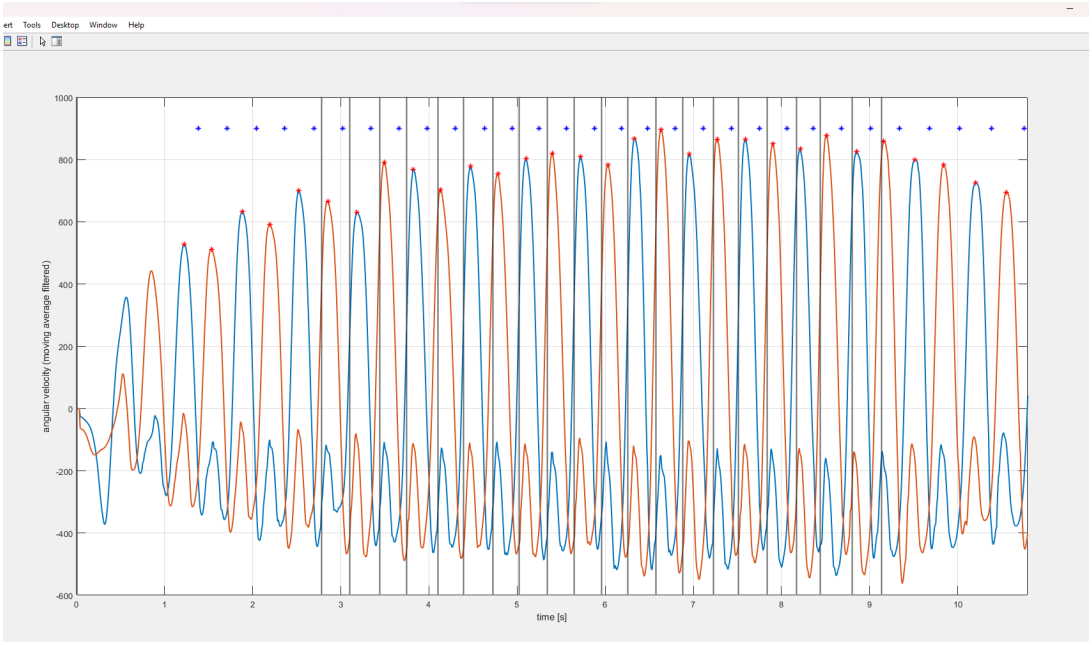

Figure 5.38: In blue are shown the middle points of the local maxima

Knowing the the timings of the steps of which position in known, only the corresponding local maxima are selected (figure 5.39).

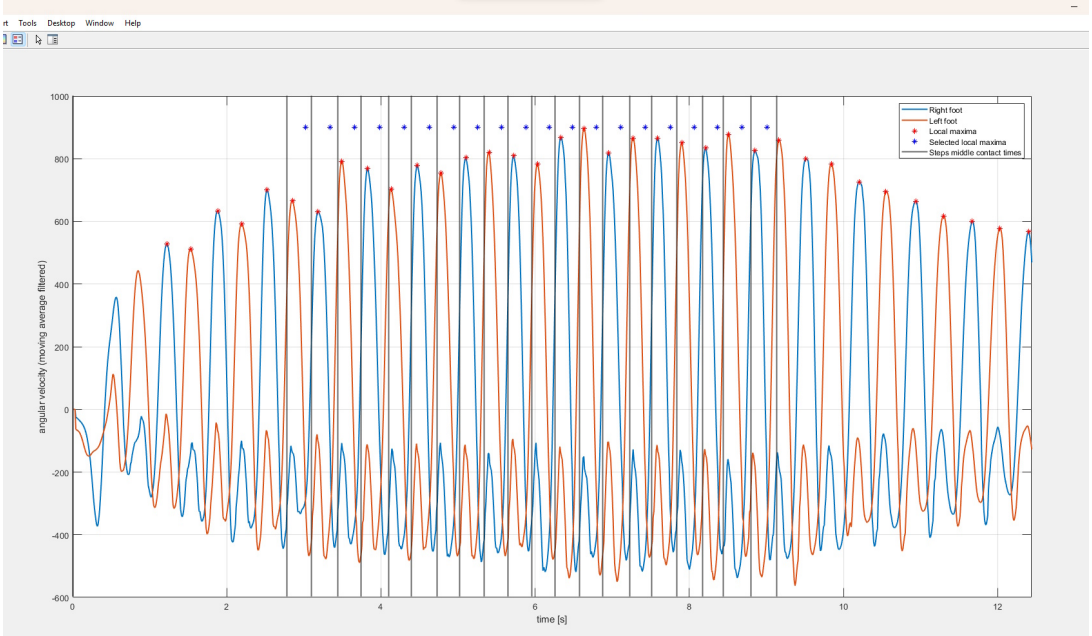

Figure 5.39: In blue are shown only the selected local maxima that will be the center point of the data of the samples

Now it is possible to store the IMUs data in the neighbourhood of these middle points as shown in figure 5.40; note that two lines are shown for each sample, as both left and right IMUs data are stored.

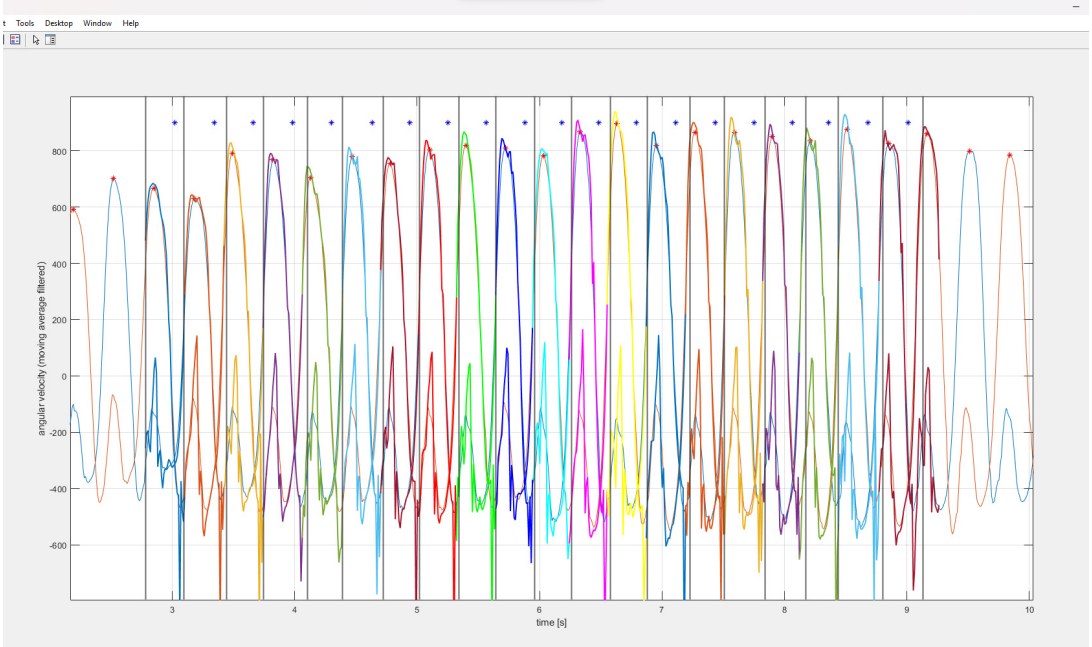

Figure 5.40: Selected gyroscope mediolateral axis neighbourhood data for each local maxima

The source code can be found in the GitHub repository[16] and the final dataset has **2216 samples**.

5.8 Ground contact time model training

With the dataset ready a model has been trained to see if it is possible to recognise ground contact time and the stride length from just IMUs data.

The source notebook can be found in the GitHub repository[16]

5.8.1 Import

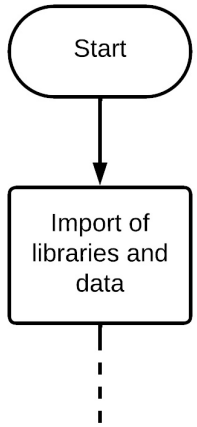

Figure 5.41: Part 1 of flow chart of the ground contact model

The first steps of a ML model is to import the necessary Python **libraries** and the **dataset**. The most notable libraries imported are Sklearn, TensorFlow and Keras:

- **Sklearn**[20] provide simple and efficient tools for data preprocessing and models creation;
- **TensorFlow**[21] provide the tools necessary to build a model with Deep Neural Networks;
- **Keras**[22] which is a TensorFLow interface to build user-friendly and modular Deep Neural Networks, allowing for easy implementation of the convolutional layers on top of other layers like dropout, batch normalization and pooling.

5.8.2 Data preprocessing

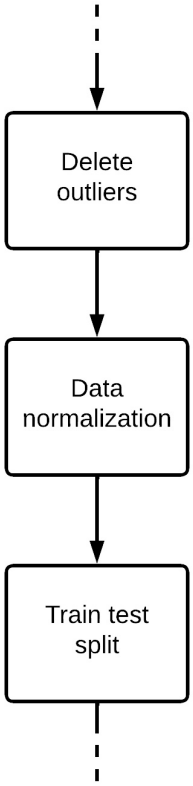

Figure 5.42: Part 2 of flow chart of the ground contact model

Some data gets excluded based on the labels values: if the label is bigger than 300 ms or smaller than 80 ms, the sample is probably an **outlier** as these are unrealistic measurements and should not be included in the training. Subsequently the data gets **normalized** from 0 to 1:

$$x_N = \frac{x}{x_{Max} - x_{Min}} + \frac{x_{N,Max} - x_{N,min}}{2}$$

$$x_N = \frac{x}{2 \cdot sensor_range} + \frac{1-0}{2}$$

the accelerometer and gyroscope sensor's range are 32 g and 4000 $^\circ/s$ respectively. (figure 5.45) This step aims to bring the data to the same scale, and additionally the values being closer to each other helps in these types of models.

Finally the data is **split in training set and test set** using the train_test_split tool from scikit-learn, with a ratio of 0.2.

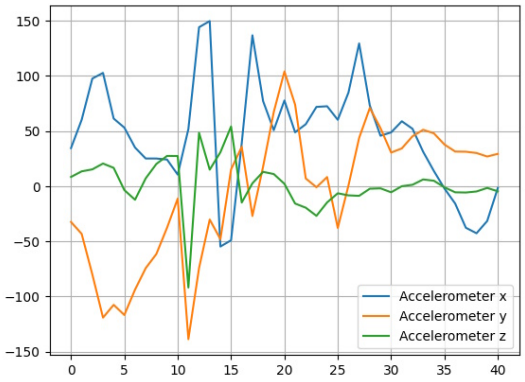

Figure 5.43: Acceleration data of the sample

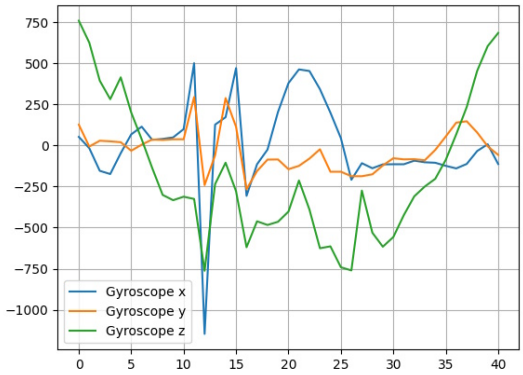

Figure 5.44: Gyroscope data of the sample

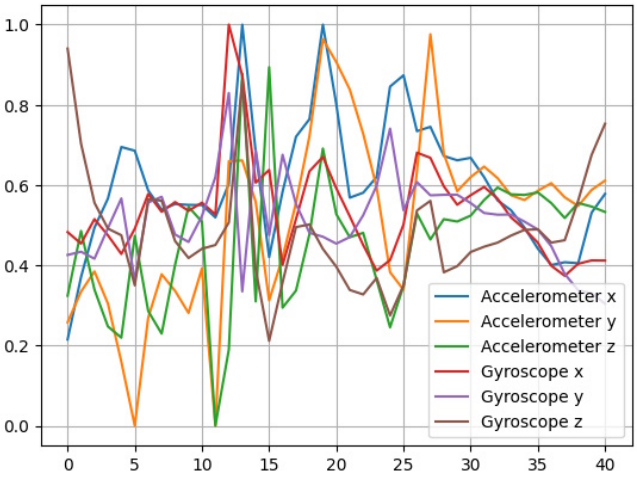

Figure 5.45: Normalized sample data, not the same sample as the previous two images

5.8.3 Model building and evaluation

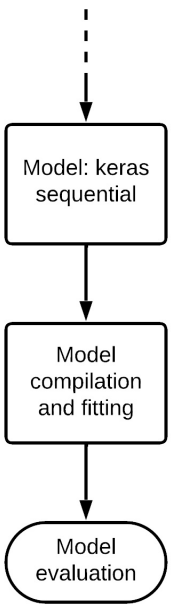

Figure 5.46: Part 3 of flow chart of the ground contact model

The model itself has been build with a Keras Sequential which allows to assemble different subsequent layers.

- **Input** layer, the dimension of the sample provided is $[41, 6, 1]$: timesteps, features and depth of the data; the last dimension is useful for the convolutional 2D layers;
- **Convolutional 2D** layer with 32 filters, kernel size 8 by 1: this layer perform 32 filters of size 8 along each feature to the 41 by 6 samples. The output of this layer is 32 filtered variants of each sample;
- **Convolutional 2D** layer with 64 filters, kernel size 4 by 1. The output is 64 filtered versions of each sample;
- **MaxPooling 2D** layer with pool size 2 by 1 and stride 2 by 1 select the highest value between two adjacent one with a stride of two, effectively shrinking the size of the sample to 21 by 6, keeping 64 variants.

- **Dropout** layer that randomly sets input units to 0 with a frequency of 0.5 rate at each step during training time. This helps prevent overfitting. Inputs not set to 0 are scaled up by 1/(1 - rate) such that the sum over all inputs is unchanged. The input have shape (timesteps, features) and to set the same dropout mask to be the same for all timesteps a noise_shape argument needs to be specified with a 1 in the time dimension: in this case is $[None, 1, 6, 64]$;
- **Flatten** layer merge all the features in one dimension;
- **Dense** layer, or fully-connected layer of output size 256 with "softmax" activation;

- **Dropout** layer with 0.1 rate;
- **Dense** layer of output size 1, without activation to have one continuous output.

After the creation of the model the compile() method is used to specify the loss function and the optimizer: in this case the **"nadam"**[23]) optimizer (adaptive moment estimation[24], plus Nesterov momentum has been used in combination with the **mean absolute error** as loss function.

The fit() method is then used to actually train the model with the train set. The arguments allow to specify the number of **epochs** and a **validation set ratio**. The number of epochs is the number of times the model tries to improve from the previous training based on the value of the loss function: after generally 50-100 epochs the model is not able to improve further and stabilizes around a loss value. This value is set to 100.

The validation set ratio specifies the portion of the training set devoted to the validation process during the training to prevent overfitting and is set to 0.2.

Showing the history of the loss throw the epochs it can be seen that the validation loss expressed in mean absolute error, stabilizes around **12 ms**.

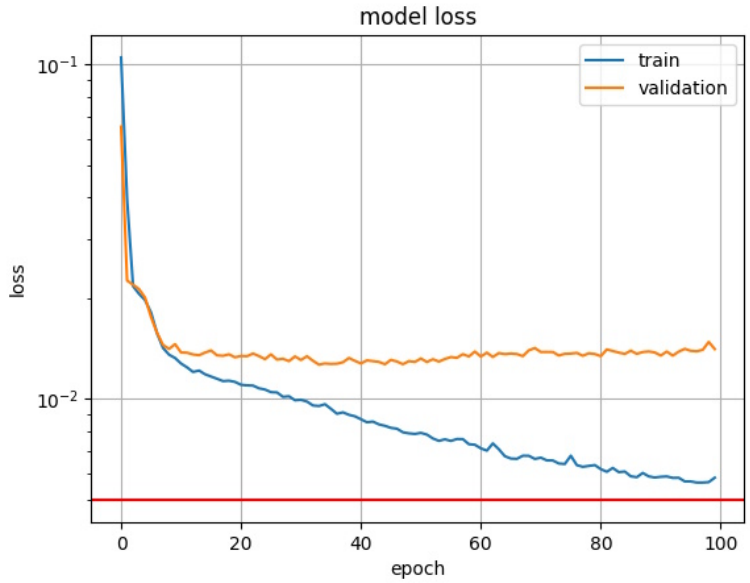

Figure 5.47: Mean absolute error evaluation of each epoch of the training set and the validation set. In red is the insole resolution of 5 ms

As it can be seen in figure 5.47 the model performs better around the 35th epoch on the validation set, and the later epochs show that the model is overfitting the training set while keeping mean absolute error around the same value.

5.9 Stride length model training

The stride length model has been structured in **two steps**, the first is a **classification**, the second a **regression**.

The first 6 columns is always data from the right foot, but this data could be a **lift off from the ground or a landing** movement. So first a classification is performed to determine **which foot is lifting off from the ground**, and in order to have a coherent dataset describing the same movement for the regression step, **the first 6 features get swapped with the rest**. Since the data is specular is really easy for a model to recognise and classify correctly. This classification uses the first column of the "y" matrix for the training where the label is "0" when the right foot lift off, and "1" when the right foot lands.

The source notebook can be found in the GitHub repository[16]

5.9.1 Classification model

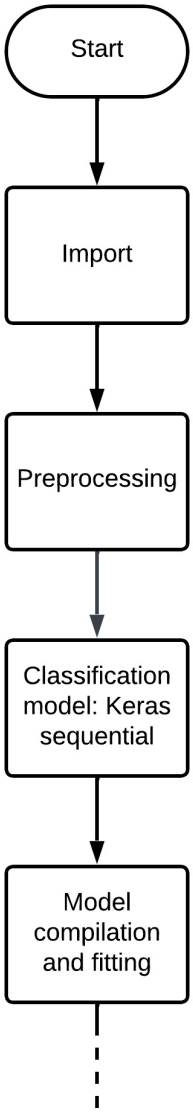

Figure 5.48: Part 1 of the flow chart of the stride length model

The data and **libraries import** and **data preprocessing** steps are the same as the ground contact model, with the exclusions of labels which are above 3 meters, which is most likely an error in the data collection.
The classification model is created using a Keras Sequential method, but no convolutional layers are needed for this very easy task.

- **Flatten** layer merge all the features in one dimension;
- **Dense** layer, or fully-connected layer of output size 2 with "softmax" activation for 2 class classification.

  The model is compiled using the "adam" optimizer and accuracy as loss function, and the fitting is performed with 30 epochs and 0.2 validation ratio.

  The model reaches almost **100% accuracy**, leading to 2 prediction errors on 443 test samples.

5.9.2 Regression model

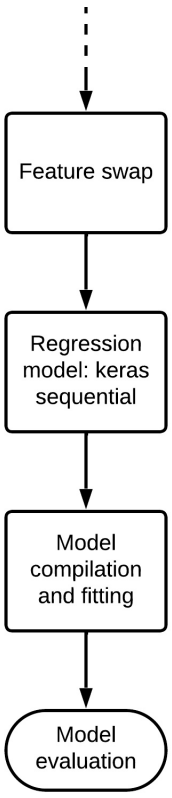

Figure 5.49: Part 2 of the flow chart of the stride length model

After the classification all the samples that are recognised as category "0" gets the first 6 columns of data swapped with the remaining 6: in this way the dataset is now coherent where the first 6 columns always describe a foot that is landing, and the last 6 columns describe a foot that is leaving the ground. This could have been the other way around, the goal is to achieve a coherent dataset.

The regression model is very similar to the ground contact time one:

- **Input** layer, the dimension of the sample provided is $[50, 12, 1]$;
- **Convolutional 2D** layer with 32 filters, kernel size 8 by 1. The output of this layer is 32 filtered variants of each sample;
- **Convolutional 2D** layer with 64 filters, kernel size 4 by 1. The output is 64 filtered versions of each sample;
- **MaxPooling 2D** layer with pool size 2 by 1 and stride 2 by 1 select the highest value between two adjacent one with a stride of two, effectively shrinking the size of the sample to 25 by 12, keeping 64 variants.
- **Dropout** layer with 0.1 rate and noise_shape $[None, 1, 12, 64]$;
- **Flatten** layer merge all the features in one dimension;
- **Dense** layer, or fully-connected layer of output size 256 with "relu" activation;
- **Dropout** layer with 0.1 rate;

- **Dense** layer of output size 1, without activation to have one continuous output.

Showing the history of the loss throw the epochs it can be seen that the validation loss expressed in mean absolute error, stabilizes around 6,2 cm.

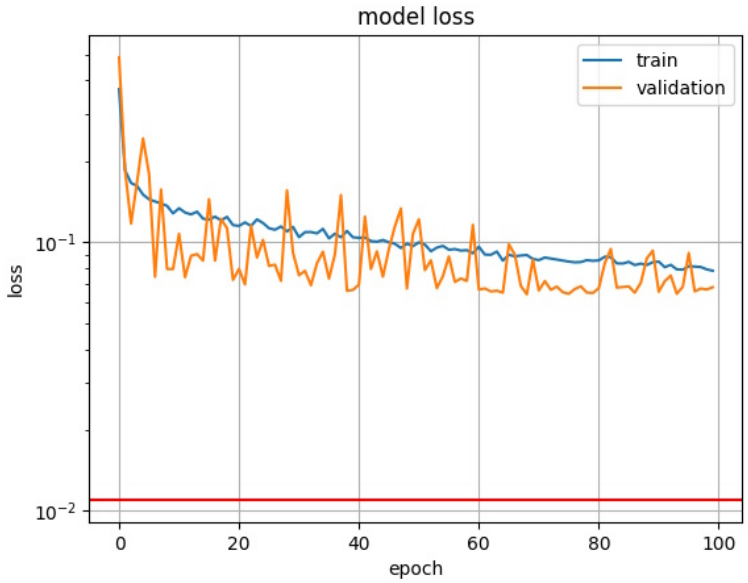

Figure 5.50: Mean absolute error evaluation of each epoch of the training set and the validation set. In red is the position resolution of 1.1 cm

Chapter 6

Results

6.1 Ground contact time model performance

The model is evaluated comparing its predictions of the ground contact time on the test set.

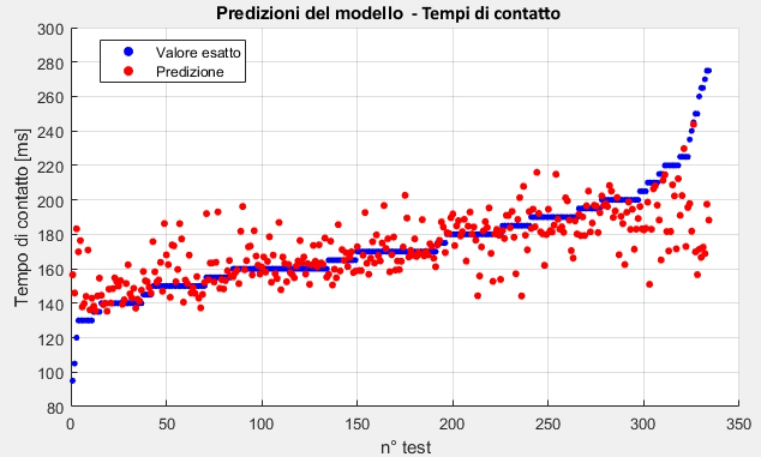

Figure 6.1: On the x axis is the order number associated with the sample; in blue the labels, in red the predictions

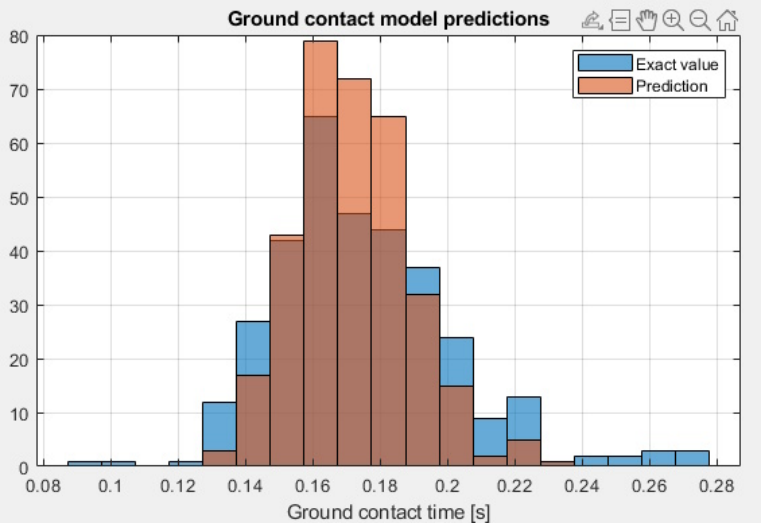

Figure 6.2: Histogram showing the distribution of the test labels and the predictions

The mean absolute error value on the test set is very similar to the MAE on the validation set, around 12.5 ms.

The test dataset is being sorted in ascending order in figure 6.1 and in figure 6.2 it can be seen that the model tends to concentrate the predictions on the middle values; this is probably due to the lack of enough data on the edge cases.

6.2 Stride length model performance

The model is evaluated comparing its predictions of the stride length on the test set.

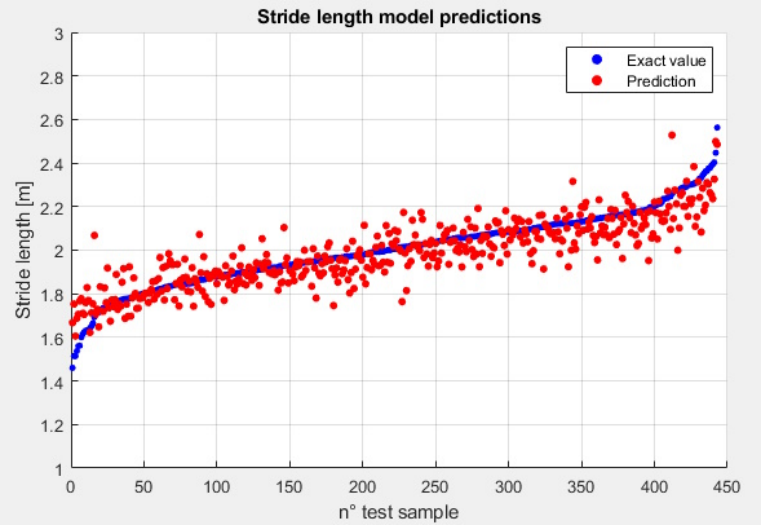

Figure 6.3: On the x axis is the order number associated with the sample; in blue the labels, in red the predictions

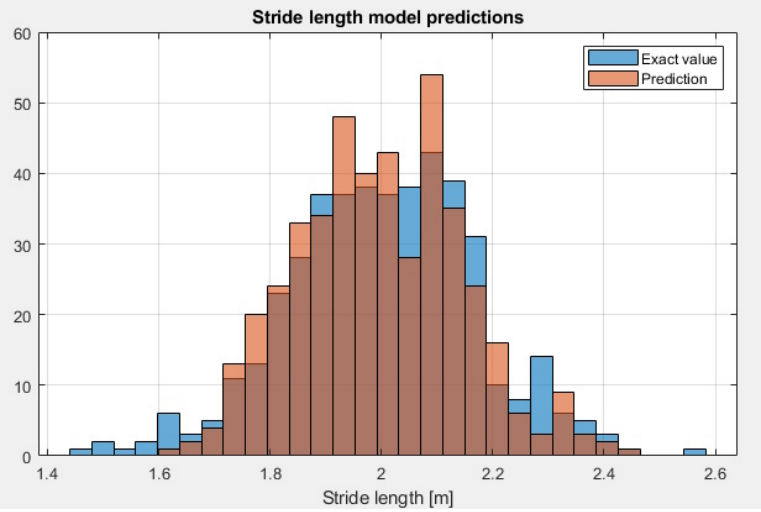

Figure 6.4: Histogram showing the distribution of the test labels and the predictions

The mean absolute error registered on the test set is 6.0 cm, a bit lower than the validation set MAE.

As it is highlighted in figure 6.3, where the labels have been sorted in ascending order, and figure 6.4 the MAE is pretty uniform along all the stride length possible values, with a bit of overestimation on lower values, and a little underestimation when above 2.1 m.

Chapter 7

Conclusions

7.1 General considerations

The results shows that movements recorded with IMUs operating at 100 Hz are a type of data that is trainable in a machine learning environment: despite that fact that so much movement data is lost between each cent of seconds, the little variations between the steps are still present and detectable, allowing the recognition of the target parameters.

The ground contact time model has its labels assigned by the insoles data: they have a resolution of 5 ms, and the model has a mean absolute error of 12.5 ms which is just over two times the original resolution.
This result is not good enough in terms of usability: the smaller ground contact times ranges from 150 to 100 ms and even below, an error of 12.5 ms would be around 10% of the total measurement, so it cannot be employed in everyday use. Ideally it is required to reach at least 10 ms of resolution or less and could be achievable with further developments.

The stride length model based its labels on the videocamera recordings, which has a resolution of 1.1 cm. The final mean absolute error of the model is 6 cm, which make it just below 6 times higher; it may seem a bad result, but since the average stride length is around 2 meters, this error represent just the 3% of the the measurement, and it is totally acceptable even in this state.

7.2 Current model limitations

The models are limited by the acquisitions setup itself which lead to the creation of a pretty specific dataset.

7.2.1 Sampled running phase

The subjects run for 50 meters, but the first 10, where the athlete accelerate, were not captured by the videocamera, and the insoles do not register the first steps. This means that currently the model is only trained to recognise steps of cruise phase of running.

7.2.2 Sampled running intensity

The subjects had to run many times (up to 16 times), therefore they could not push their speed to the limit. Lower speed, generally means higher ground contact times[25], so there is a lack of smaller ground contact times samples.

7.2.3 Datasets acquisitions

Acquiring new data is not easy in the current setup, as it took 2 hours for each athlete, with cumbersome preparations and many systems to take care of.

Furthermore the track had to not be crowded as if a person would get in the shot, it would interfere with the videocamera recordings. This meant that it was difficult to find subjects to test in times of the day where few people trained on the track. This is the main reason why the datasets are of modest dimensions.

7.2.4 Datasets augmentation

The datasets have not been submitted to data augmentation, because the model needs to be trained only on real differences between the samples, and not from artificially augmented ones.

One way to possibly create a bigger dataset is to select multiple samples to associate to the same label as shown in figure 7.1.

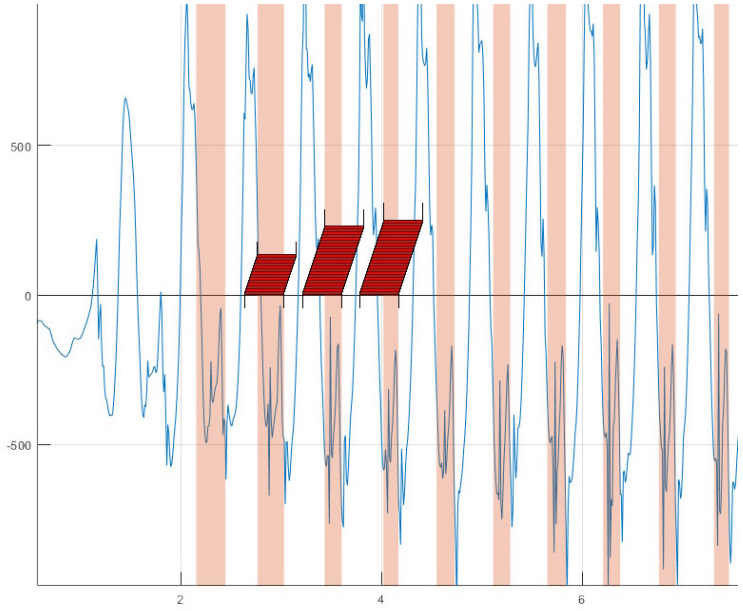

Figure 7.1: Multiple samples are selected for each contact, only three contact shown as example

Instead of taking the approach of the steps isolation looking at the local maxima of the mediolateral axis of the gyroscope (which can still be employed for hen steps have to be isolated in the actual use of the final product), the IMUs data is taken multiple times with fixed width windows around the contact moments, making sure to always comprehend them.
The figure 7.1 also highlight how smaller ground contact durations would lead to more acquired samples with the same label.

7.2.5 Insoles unreliability

A final inspection of the labels provided by the insoles highlighted a behaviour that is possibly not intended, and if fixed could improve the ground contact time performances.

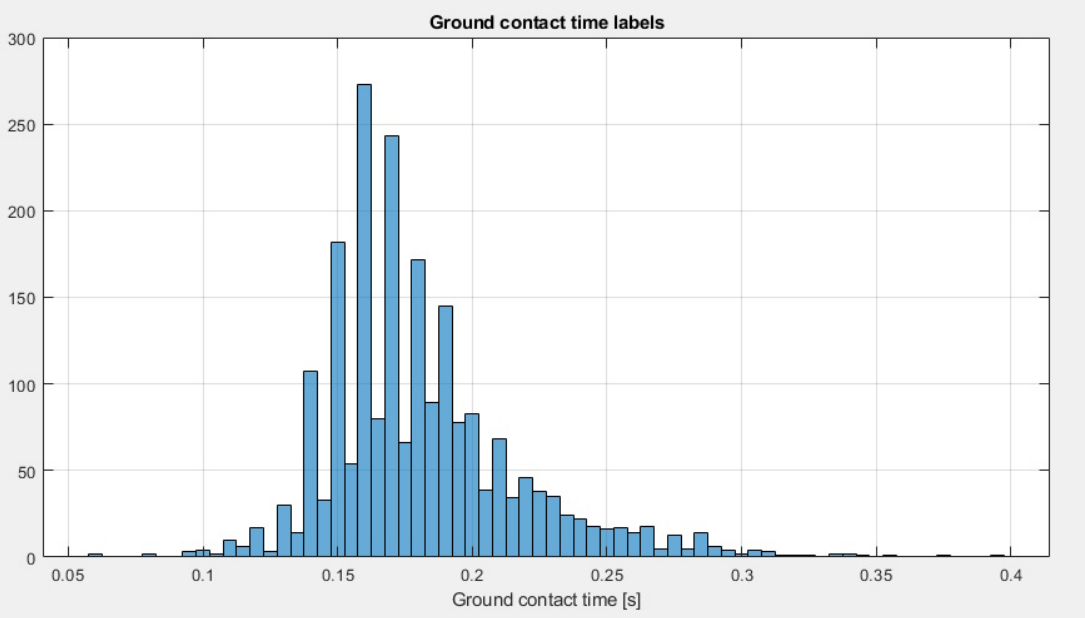

Figure 7.2: Histogram showing the distribution of the labels provided by the insoles

As it is highlighted in figure 7.2, the better part of the labels present a "0" in the 3rd digit after the comma. In theory this should not happen, and the values should be "0" or "5" equally distributed.

7.3 Dataset possible improvements

While the IMUs sensors are proven to be employable for this type of task, it is possible to change the labels collection using different types of sensors.

Using optical sensors would reduce the resolution to 1 ms for the ground contact times, while keeping almost the same resolution in the stride length, around 1 cm, but the acquisition setup would be simplified as the requirement to have an empty track would not be required anymore.

Simplifying the acquisitions and reducing the labels resolutions would benefit greatly in the creation of a bigger and more precise dataset.

Chapter 8

Direction of future developments

8.1 Final user experience

As this thesis proves that it is possible to measure some running parameters using cheaper sensors, let us define the user experience the athlete could face while using this new equipment to complement its training gear.

The final system could be a combination of the two sensors and a smartphone with an application to interface the data provided by the sensors. And the experience could something like the following:

- sensors turn on;
- bluetooth pairing to the smartphone app for internal clock and data synchronization;
- the sensors start recording after being paired for the first time after turn on;
- the smartphone app act a stopwatch to define the start and stop of the running trials to elaborate;
- when the sensors are in the proximity of the smartphone, the app requests the samples relative to the defined start and stop by the app;
- the app uses the models to elaborate the data and show the results of each run.

The proposed user experience highlights the need for the **development of custom sensors and application** capable of bluetooth pairing and data transferring.

8.2 Stride length predictions refinements

Currently the model treats every sample as standalone, but in a real life implementation, most of the times the total length of the trial is known, and this could be an additional data to exploit.

Given the precise start and stop of the trial and the total distance covered by the athlete, the sum of the model's predictions needs to be equal to that distance. In this way the model could refine its predictions.

8.3 Stopwatch function with smartphone camera

Using the touchscreen of a smartphone to define the start and stop of a trial is not as comfortable as a physical stopwatch, but at least the stop function could be replaced with a function in the application.

Every smartphone is equipped with a high resolution camera, and they are able to record at least at 60 at lower resolutions like 720p: 60 frames per second means a frame every 16 ms, and this time resolution is good enough to use the smartphone camera as a "photo-finish" of the trial.

This implementation would require the user to setup the smartphone on a tripod, aligned to the finishing line, and use some kind of video elaboration to detect when the athlete finishes the trial, and therefore record the stop time.

Bibliography

[1] Weyand PG et al. Faster top running speeds are achieved with greater ground forces not more rapid leg movements. *J Appl Physiol*, 2020. doi: 10.1152/jappl.2000.89.5.1991.

[2] Jacquelin Perry and Judith M. Burnfield. Gait Analysis: Normal and Pathological Function. *J Sports Sci Med*, 2010.

[3] JMuro de-la-Herran et al. Gait Analysis Methods: An Overview of Wearable and Non-Wearable Systems, Highlighting Clinical Applications. *Sensors*, 2014.

[4] Lin Zhou et al. Validation of an imu gait analysis algorithm for gait monitoring in daily life situations. *IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 4229–4232, 2020. doi: 10.1109/EMBC44109.2020.9176827.

[5] Yonatan Hutabarat et al. Quantitative Gait Assessment With Feature-Rich Diversity Using Two IMU Sensors. *IEEE Transactions on Medical Robotics and Bionics*, 2(4):639–648, 2020. doi: 10.1109/TMRB.2020.3021132.

[6] Yonatan Hutabarat et al. Seamless temporal gait evaluation during walking and running using two imu sensors. *IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 6835–6840, 2021. doi: 10.1109/EMBC46164.2021.9629492.

[7] Brendan Purcell et al. Use of accelerometers for detecting foot-ground contact time during running. *Proceedings of SPIE - The International Society for Optical Engineering*, 2006.

[8] Marcus Schmidt et al. IMU- based determination of stance duration during sprinting. *Elsevier Ltd.*, 2016.

[9] Belkacem Chikhaoui and Frank Gouineau. Towards automatic feature extraction for activity recognition from wearable sensors: A deep learn-

ing approach. *IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 693–702, 2017. doi: 10.1109/ICDMW.2017.97.

[10] Paolo Tasca. A machine learning approach for spatio-temporal gait analysis based on a head-mounted inertial sensor. Master's thesis, Politecnico di Torino, december 2022.

[11] Jermaine Smith. Real-time position tracking using imu data. Technical report, Univesity of Minnesota, 2018.

[12] Inertia Studio. *ProMove-mini Wireless Inertial Sensing Platform User Manual*, 2023.

[13] Francesca Salis et al. A multi-sensor wearable system for the assessment of diseased gait in real-world conditions. *Frontiers in Biofabrication*, 2023.

[14] Francesca Salis et al. A method for gait events detection based on low spatial resolution pressure insoles data. *Journal of Biomechanics*, 2021.

[15] MATLAB. *9.11.0.1837725 (R2021b) Update 2*. The MathWorks Inc., Natick, Massachusetts, 2021.

[16] Edoardo Mercuri. Collection of scripts and notebooks developed for "Machine Learning for Running Analysis with Wearable Inertial Sensors" Master's Thesis, Politecnico di Torino, July 2023., July 2023. URL `https://github.com/PIC4SeRThesis/EdoardoMercuri`.

[17] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

[18] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[19] Camillo Lugaresi et al. Mediapipe: A framework for perceiving and processing reality. In *Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR) 2019*, 2019. URL `https://mixedreality.cs.cornell.edu/s/NewTitle_May1_MediaPipe_CVPR_CV4ARVR_Workshop_2019.pdf`.

[20] F. Pedregosa et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[21] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[22] François Chollet et al. Keras. `https://keras.io`, 2015.

[23] Timothy Dozat. Incorporating Nesterov Momentum into Adam. *Stanford*, 2015.

[24] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*, 2015.

[25] Peter G. Weyand, Deborah B. Sternlight, Matthew J. Bellizzi, and Seth Wright. Faster top running speeds are achieved with greater ground forces not more rapid leg movements. *Journal of Applied Physiology*, 2000.