Master of Science in Mechatronics Engineering

Gait recognition basing on IMU sensor and machine learning methods

Albi Matteo
Cardone Andrea
Oselin Pierfrancesco

Department of Industrial Engineering
Academic Year 2021 - 2022

Contents

1 Introduction

The aim of this project is to achieve gait phases recognition using one IMU sensor.

Walking is a mostly periodic action, assuming a constant speed of advancement. Each walk cycle can be divided into several phases, ranging from two up to a maximum of eight.

A full gait cycle is defined as a periodic cycle involving two legs from the IC (Initial Contact) of one foot on the ground to the following occurrence of the same event with the same foot. Typically, a gait cycle is divided into two main phases. The stance phase, which represents approximately 60% of a gait cycle, starts when the foot strikes the ground (IC) and ends when it leaves the ground (TO, Toe off). The swing phase, which accounts for approximately 40% of the remaining gait cycle, starts when one foot leaves the ground (TO) and lasts until it touches the ground again (IC).

The number of detectable phases depends on the methods and sensor systems adopted to gather the signal for recognition of the phases and events. Using one IMU sensor, as in our case, no studies have been reported in the literature where more than four phases have been found [1].

Doing research on the publications present in the academic world, a review of the gait phase detection algorithms by [1] was particularly interesting. In this article, the main methods of dealing with this problem have been outlined. In particular numerous valuable methods of event and phase detection were presented. These computation methods are categorized into two main domains. Firstly, the algorithms based on the threshold method, time-frequency analysis, and peak heuristic algorithms, which are also variations of the threshold method. Secondly, Machine Learning (ML) approaches are now among the most popular techniques to detect phases and events with various models.

1.1 Threshold methods

The Threshold method is the simplest computation method of gait detection. There are different types of threshold algorithms as a set of value rules that figures out certain characteristics of the gait phases or events. In particular, the gyroscope signal which indicates rotation in the sagittal plane is used. In most cases the gait events are found by setting threshold values

or by looking for certain peaks in the signal. A more specific analysis of the adopted method will be discussed later.

1.2 ML methods

ML methods are among the most popular techniques used to classify gait phases in off-line data as well as in real-time data. Many gait phase recognition methods have been developed using different types of machine learning approaches such as HMM, NN model, DLNN, and CNN. ML-based methods were overall an effective approach method for the gait phase detection. Many studies showed excellent offline results, however the main problem of real-time detection is the computation costs and therefore the delay of the detection.

Among the various sensors that can be used, IMUs consisting of a combination of gyroscopes, accelerometers, and magnetometers have become widely utilized. They are portable, cheap, durable, reliable, consume low energy, and can easily be mounted on different body parts. These sensors also provide rich signal information about the gait angular velocity and acceleration that can be used for an accurate prediction of the gait events and phases, which is the main reason why IMU signals are a good option to use in a machine learning approach.

1.3 Method Description

For this project we have been equipped with an IMU sensor and foot pressure sensors. The method used and the various reasons for the choices made will be introduced below.

First of all we chose to place the sensor on the leg, just above the ankle. In fact, in many publications this position has been chosen as it allows to obtain a signal with a lot of information regarding walking. In this position it is possible to obtain a good signal very little affected by noise, moreover this position is better for comfort, compared to example on the foot. This choice was also dictated by the fact that, having only one IMU sensor, it was necessary to place it in a position that provided the best possible signal.

The methods based on ML have shown great performance, moreover

they are very effective as regards the recognition of the phases with respect to the variability of the signal obtained through the IMU sensor worn by different people. For this it was decided to implement an ML-based method. In fact the threshold algorithms can be rather rigid and not very efficient with regard to this problem. In fact, they may have parameters that must be adjusted manually according to the needs of the case. Both to overcome this problem and to obtain a real time application, these algorithms become remarkably complex.

The main problem we encountered with the ML method is having a labeled dataset with which to perform the training of the neural network. One way to solve this problem is to use foot pressure sensors to get a label automatically. This method has the disadvantage of making data acquisition more laborious, as it is necessary to position the insoles correctly every time you change the subject on which to take the data. Furthermore, if it is necessary to re-train the network, it is necessary to redo the entire acquisition of the labeled dataset, which also requires the pressure sensors and may not be a quick procedure.

The proposed method is based on a threshold method performed offline to label the data collected by the IMU. Once the data has been collected, it will be sufficient to modify few hyperparameters of this auxiliary algorithm, if necessary, and carry out the labeling automatically. In this way, the whole system is based solely on one sensor, especially as regards the possible need for a future retrain.

The threshold based algorithm was implemented taking inspiration from various publications [2–4], Further tests were conducted with the use of pressure sensors in order to validate this method.

Once the neural network has been trained, it can be used for both offline and online gait phases recognition. All sections of the project will be described in detail below.

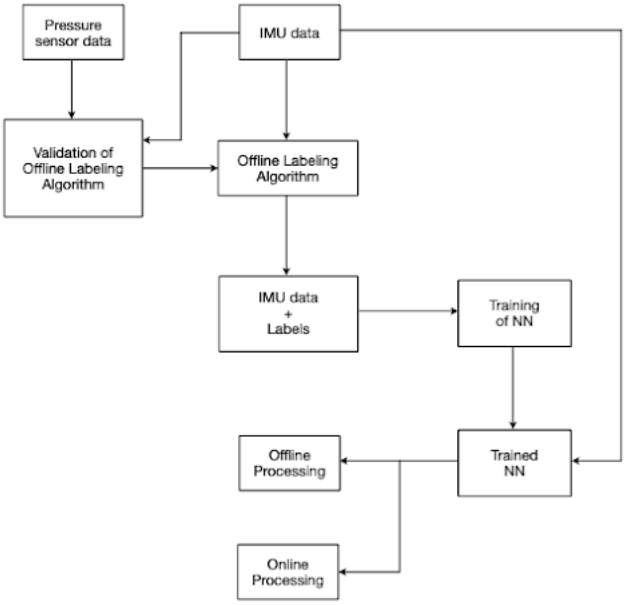

Figure 1: Scheme of the proposed algorithm for gait phases detection.

2 Sensors

In this project two different sensors were used: an IMU Bluetooth sensor and an insole pressure-array sensor. The IMU sensor will be used in the final application of the project as data collector for the gait recognition, the insole has been used for data labeling and validation during the project development.

2.1 IMU sensor: LPMS-B2

The IMU sensor provided for the accomplishment of the task is called LPMS-B2, built by LP-RESEARCH Inc. It's equipped with the main inertia sensors (accelerometer, gyroscope, magnetometer and altimeter) and also some extra sensors, not used for the project (thermometer and barometer).

The sensor communicates only via Bluetooth and implements a pre-data processing of the raw signals from the accelerometer, gyroscope and magnetometer. It's capable of estimating the Euler angles of the system, linear accelerations, angular velocities and computes the quaternions associated

with the actual pose. It also provides filtering of the data, calibration of the magnetometer and variable ranges of measurement for all sensors.

The LPMS-B2 has two different communication modes: streaming mode and command mode. In command mode data is sent to the PC only on request, in streaming mode the sensor keeps collecting and sending data to a specific frequency. We wrote a MATLAB script called *LpmsBT.m* to manage the sensor, building the following functions

- Connect automatically via Bluetooth to the device using sensor's name
- Disconnect from the device
- Specify the communication mode (command and streaming)
- Specify which data should the sensor transmit
- Specify the streaming frequency
- Read the current configuration
- Request data

More information about the sensor and all its functionalities can be found in the LPMS Operator's Manual.

2.2 Insole pressure-array sensor: MITCH

For data validation an insole pressure sensor, called MITCH, has been used. A large part of the project, as stated later, is the data labeling of the IMU signals with the different gait phases to train a deep neural network.

We dedicated a specific session to the usage of two MITCHs, one per foot, along with the LPMS-B2, to collect data in parallel and compare the results obtained with the developed algorithm and the pressure sensors. This way we were able to validate the reliability of the algorithm dealing with the labeling task.

Indeed, using pressure sensors located in different positions of the footbed, it's immediate to recognise the gait phases, and being a robust method we used it as reference for our algorithm.

The MITCH contains sixteen pressure sensors, mounted on a foot-shaped plastic film in different positions, and a data collection device, that includes the read-out circuit for every pressure sensor and an IMU. In the following image the id associated with each pressure sensor is shown (in red the sensors that didn't seem to work). The device can communicate with the

PC via USB port and can automatically collect data when unplugged. Using a dedicated software we first synchronized the timestamp of the two MITCHs, then the LPMS-B2 and the two MITCHs have been subjected to the same periodic acceleration along a common axis. This way we were able to synchronize offline the three different signals obtained by the sensors.

We later used this data to build the labeling algorithm, as explained in the following chapter.

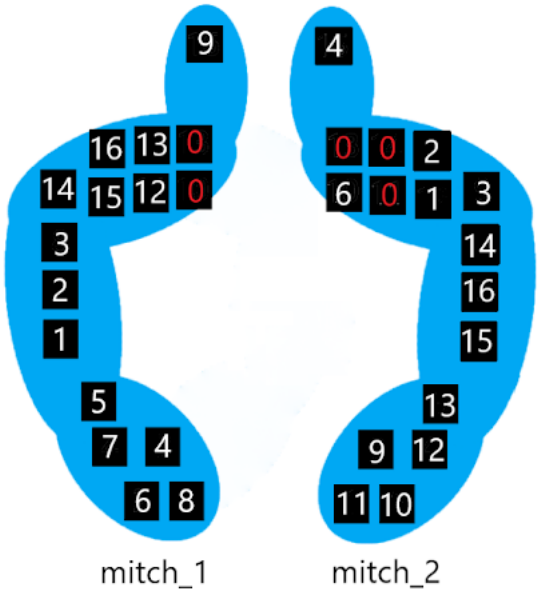

Figure 2: Scheme of the device listing the position of the pressure sensors according to their index in the registered values. Indicated as zero the not working sensors.

2.3 Data collection

Most data has been collected using a treadmill in a gym. For every subject we collected two sessions of about two minutes, at the speed of 4 and 6 km/h (one of the subjects ran at 5.8 km/h because shorter than average, to avoid a running behavior).

We also properly cut the data collected in the lab session while we were using the MITCHs. These data helped to add variability to the dataset being the speed of the subject variable.

3 Labeling method

3.1 Threshold Algorithm

As found in academic publications [1–4], the gyroscope signal indicating the rotation of the leg in the sagittal plane has a characteristic periodic pattern. Figure 3 shows the signal for a few seconds. As you can see, the signal is characterized by a maximum peak (red square), followed by a rapid negative peak (green circle), followed by a fairly stable piece that ends before a rapid descent(orange triangle) and finally a negative peak (purple x). Then the pattern repeats itself throughout the walk.

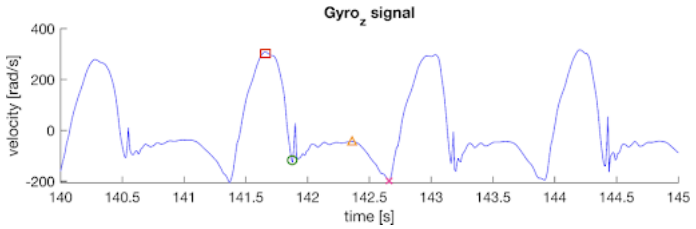

Figure 3: Signal of the gyroscope with highlighted the four notable points.

The maximum peak corresponds to the moment with the maximum rotation speed of the leg while the foot is in flight, called MSw (Major Swing), the negative peak that comes after corresponds to the initial contact of the foot with the ground, called IC (Initial Contact) . From this point on, the foot is in contact with the ground. This phase has been divided into two parts, separated by the point where the heel starts to rise. This point, called MSt (Major Stance) corresponds approximately to the last peak of this phase, before the steep descent to the next phase. At the next lowest point the foot lifts off the ground, this point is called TO (Toe Off).

Thanks to the fact that this algorithm is offline, we can easily find the peaks of interest. The input of the labeling function is the whole data acquired from the IMU sensor, and the output is an updated version of the input file with an extra column for the labels. It is suggested to cut the signal related to the time interval of walk out of the whole signal. In addition It is suggested to check the correct behavior of the algorithm, plotting the gyro signal with the phases, in case of need tuning the cutoff frequency and the second threshold (secondary threshold for a correct detection of MSt, not so significant for the correct detection) it should make the algorithm work correctly. The algorithm is described in the following procedure.

1. Filter with a low-pass filter to find only one maximum peak per period, the cutoff frequency must be not too small to preserve a good signal
2. Filter with a low-pass filter with lower cutoff frequency to find good MSt
3. Calculate an automatic threshold: weighted average between the positive
4. part of the signal and its maximum.
5. Create a matrix to store the conditioned signals and phases
6. Start for loop on all the signal
7. Find maximum peaks indicating (MSw)
8. Once found MSw backward for loop to find the nearest minimum (TO)
9. Once found TO continue backward for loop to find the nearest maximum (MSt)
10. Once found MSt go on with forward loop from MSw to find the nearest minimum (IC)
11. Repeat from point 7

A label is assigned to every sample of the signal, the phases are

- $phase_1$: from IC to MSt
- $phase_2$: from MSt to TO
- $phase_3$: from TO to MSw
- $phase_4$: from MSw to IC

Figure 4 summarizes the algorithm

In figure 5 is shown the original signal and the two filtered signals, in figure 6 are shown the two thresholds with respect to the signal and in figure 7 is shown the original signal with the labels of the algorithm. For future uses of this algorithm it is important to check if the phases detected are qualitatively equal to those shown in image xx.

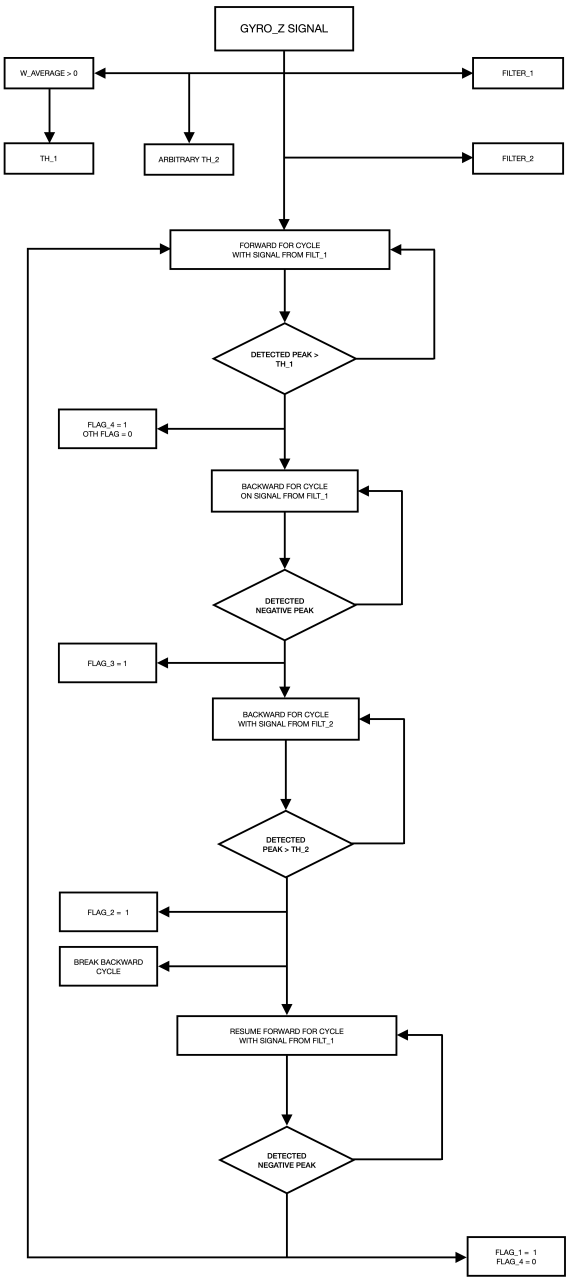

Figure 4: Flow diagram of the threshold algorithm

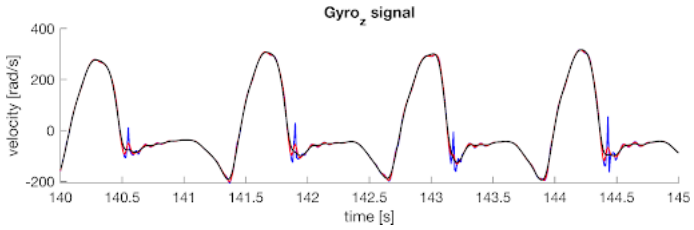

Figure 5: Original signal (blue), filtered signal(red), high filtered signal(black).

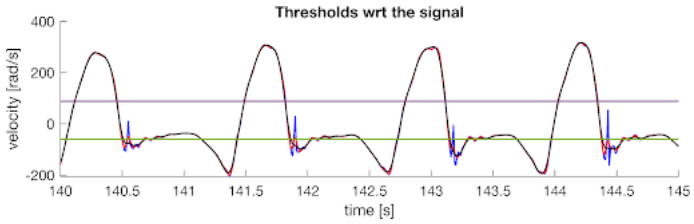

Figure 6: Original and filtered signal with high threshold (purple) and low threshold (green).

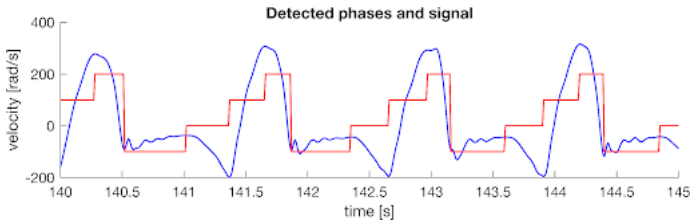

Figure 7: Filtered signal and detected phases, note the stepped trend of the gait phases.

3.2 Validation

The validation of the threshold algorithm consists in comparing the labels given by our algorithm with the labels obtained through the Mitch pressure sensors. These insoles feature a series of pressure sensors, positioned in different parts of the sole of the foot as shown previously. Three sensors were selected on the sole: one on the heel, one in the center of the foot and one on the toe. As can be seen in figure 8, the gyroscope signal is represented along with the three pressure sensor signals. The pressure sensor signals turn out to be quite noisy, so they have been filtered out. In figure 9 the original and filtered signals are shown in detail, the latter have a slightly bell shape compared to the original ones, this is due to the action of the

filter. The sensors have been transformed into switches, a threshold equal to 2 has been chosen, beyond which the sensor is ON and below which it is OFF. The threshold was chosen as the meeting point between the original signal and the filtered one.

So the notable points of the phases were chosen as: IC: heel sensor ON TO: toe sensor OFF, after heel sensor OFF MSt: minimum difference between the toe sensor signal and the sensor in the center of the foot, this means that the pressure is approximately equally distributed between the midfoot and the front. Heel pressure generally drops to zero shortly after initial contact. In figure 10 the difference between the two signals has been added. MSw: this point is not detectable with pressure sensors as the foot is in the swing phase.

Figure 11 shows the walking phases with the two methods with respect to the gyroscope signal. The threshold algorithm demonstrates an excellent detection of the IC and TO points, while the detection of the MSt point is generally slightly different from the point obtained with the insole. This is due to the fact that this phase is not determined by a net event such as rapid change of speed, the whole phase in which the foot is in contact with the ground is characterized by a low speed with many oscillations with small amplitude. So it is difficult to detect a particular event. In general, the gyro signal exhibits a relatively rapid descent towards the TO point. This part was interpreted as the acceleration phase before the foot release and therefore the last peak before the descent phase was chosen as the MSt point.

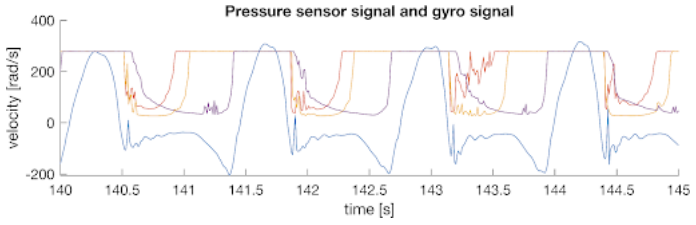

Figure 8: Gyroscope signal (blue) and signals of pressure sensors, heel (red), midfoot (yellow), toes (purple).

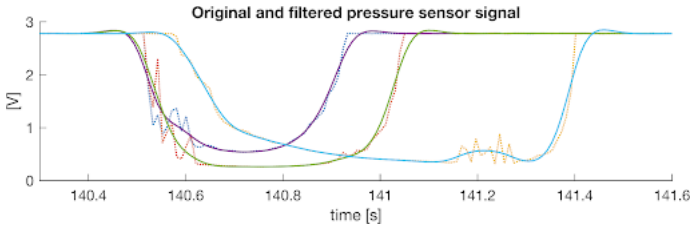

Figure 9: Original signals (dotted lines), filtered signals (continuous lines).

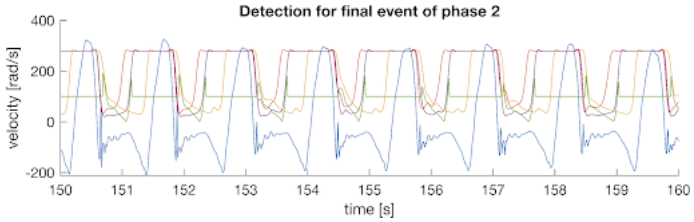

Figure 10: Original signal with filtered pressure sensors signals and difference between toe and midfoot signals (green).

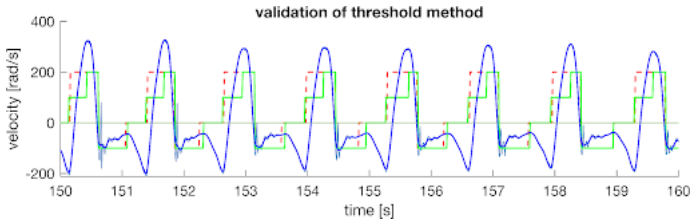

Figure 11: Final comparison between threshold method (green) and insole method (dotted red), note that the fourth phase is not detectable with insole.

4 Machine learning methods

Due to the complex variation and shape of the acquired data, simple mathematical models are not powerful enough to correctly analyze and manipulate the information hidden in them. Because of this, a robust and smart solution is relying on machine learning methods.

Machine learning is the span of (linear and non-linear) mathematical algorithms whose performances can improve automatically through experience, by leaning on and correcting themselves according to loss functions or

quality measures.

There exists plenty of machine learning methods, each one with peculiarities and niche applications, however generally speaking we can distinguish two main approaches: supervised learning and unsupervised learning. Supervised learning is the set of ML algorithms which can be implemented when data labels are known. A comparison between predicted and true output is made for estimating the misclassification error and properly tuning the algorithm's parameters. Unsupervised learning is the set of ML algorithms which must be implemented when there is no information about data labels. Examples must be therefore classified according to their similarity, in terms of means or probability distributions. In this case, since we were able to classify data in 4 different classes by means of the threshold method (section 1), we have been able to take into account supervised learning. However, as briefly mentioned above, the acquired data are characterized by strong and complex variation, namely non-linearity. Because of this, among all the possible supervised algorithms, linear ones such as linear discriminative functions (one-vs-all and all-pairs) or linear SVM (both hard and soft margin) have been immediately discarded.

Among the non-linear methods there exist non-linear SVM, kernel machines and deep neural networks (DNNs). Kernel machines have been immediately discarded because of the required complexity: first of all designing a valid kernel, which corresponds to a certain dot product between features in feature space, is quite difficult. Moreover, a common error in applying kernel machines without any experience is to make the system learn a linear combination of kernels, not the right combination of data.

Features mapping, namely non-linear SVM, could have been a good solution but it would have required a feature mapping from input space to the so-called Hilbert space. For small datasets this is not a problem but for hundreds of thousands of data it dramatically impacts computation performances.

Moreover, with these approaches the main characteristics of our dataset would have been neglected: time dependency. We do not have to forget we are trying to understand and analyze an activity that is time dependent, so events (and therefore data acquisitions) happen sequentially. This is a crucial point in order to correctly manipulate the information flow. Because of this, deep neural networks seem to be a good solution, in particular recurrent neural networks.

Recurrent Neural Networks (RNN) are deep neural networks in which

a cell state is propagated along the chain in order to take into account sequentiality of data. In particular, the fundamental unit is not a perceptron (linear combination of inputs and activation function) but it is a unit cell which takes as input an incoming data and the output of the previous (layer wised) cell.

In this unit some linear and nonlinear manipulations occur and the generated output is fed into the next cell as well as returned for intermediate evaluations. In particular, the number of unit cells is equal to the number of input data we consider at a time. Therefore, according to the kind of prediction and time delay we want to implement, we have to choose the number of units.

RNN implementations have been quite recently developed and they guarantee high performances according to the different unit cell configurations. Among all the different types, the most famous ones are: long short-term memory networks, gated recurrent units and transformers. They will hereby briefly discuss.

4.1 Long Short-Term Memory Networks (LSTM)

LSTMs are one of the most popular implementations of RNNs. These networks are composed of a layer-wised sequence of identical cells. In particular, the number of implemented units is determined by the number of samples we want to consider at the same time in the analysis.

A representation of the cell's structure is shown in the picture below. The unit receives two inputs: the first one is the input data and the second one is the output of the previous cell.

Inside the unit these inputs are combined in order to propagate the correct information flow. In detail three layers, known as gates, are generally present: the *forget gate*, the *input gate* and the *output gate*.

The forget gate selectively forgets parts of the previous cell state through the sigmoid function which is present at the end of it. In particular, if its value is brought to 0 it means the previous state has been reset; if its value is set to 1 it means that the previous state has been unaltered and if its value is in the range [0,1] a downscaling has occurred.

The input gate is responsible for discarding or considering the incoming input through a sigmoid function. Moreover, its output is combined with the hyperbolic tangent of the input itself and if the overall value is different from zero, it is summed to the filtered previous state. Eventually the output gate is responsible for applying the last combination between the current cell

state and the input, by the means of a sigmoid function.

It is important, however, to keep in mind that these gate layers are parametric and during training they are tuned iteration-by-iteration in order to fit the data.

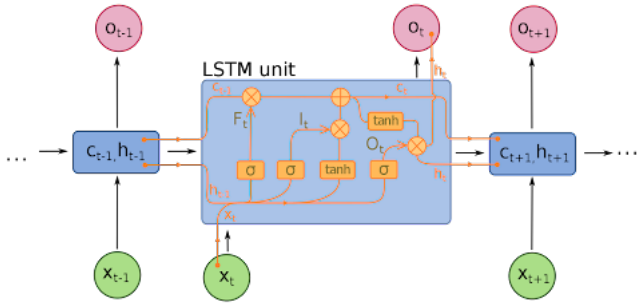

Figure 12: LSTM unit cell structure

4.2 Gated Recurrent Units (GRUs)

GRUs networks are recurrent neural networks very similar to LSTMs. The biggest difference between them is that the former have less parameters than the latter, because of the lack of the output gate.

LSTMs and GRUs show very similar performances at tasks such as speech recognition and natural language processing but the latter exhibit better scoring for smaller and less frequent datasets.

Both long short-term memory and gated recurrent units have been exploited in this project, since their behaviors fit quite well this learning task.

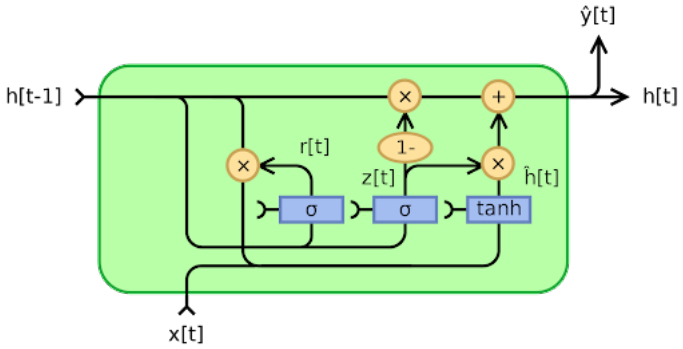

Figure 13: GRU cell structure

4.3 Transformers

Transformers methods are deep neural networks quite similar to LSTMs. Recently developed, the biggest difference with respect to the latter is the introduction of the so-called self-attention. This mechanism allows to learn how to predict the output by relying on the input encoding and the previously generated output. With this approach sequentiality is taken into account but some information is weighted more than others. Transformers are nowadays deeply implemented in speech and text recognition, since in such scenarios key information can be spread along the stream of data. For this project they have been discarded because IMU data are not as complex to understand as the meaning of a text. In activities such as walking or running information is spread uniformly across the stream and not aggregated in specific points.

Figure 14: Transformer logic scheme

4.4 Unsupervised learning

Since the so far considered deep neural networks belong to the same type, a good idea to verify the choice is comparing the obtained performances with a completely different approach: unsupervised learning.

As briefly introduces above, unsupervised learning (UL) is the very only solution when no information about an incoming dataset is provided: if labels, shape or parameters of a distribution are not given, common techniques such as bayesian decision theory, parameter estimation (maximum likelihood estimation, bayesian estimation) or interference cannot be applied.

The basic idea behind this learning class is to aggregate data according to their similarity. In particular, among the different UL types, k-means clustering is one of the most implemented algorithms. It does not require any assumption on the distribution's shape (such as, for example, Gaussian Mixture Model methods) but only the number of total clusters on which run the algorithm.

The core strategy is representing clusters with their means. According to them, data assignments are performed and means are re-computed. The

process runs over and over until there are no more variations in the mean's values. This is a very powerful tool however, as it will be further shown, by performing classification according to data similarity and not data sequentiality we observe a dramatic drop in performances.

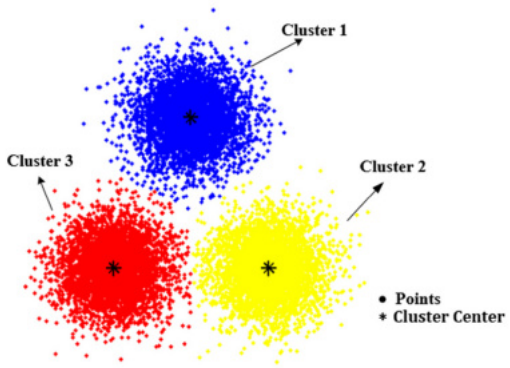

Figure 15: Visual representation of 2D k-Means algorithm

5 Network training and evaluation

5.1 Data preprocessing

In order to train the network, the first step is to reorganize the data in the proper shape as needed by the functions trainNetwork/classify of MATLAB.

The input data is in .csv format, so it's immediate to import each file as a table in the workspace. The different tables are divided in data addressed for the train and for the test of the network. Then the function dataPreprocessing prepares the data to be used by the network.

First, using the function *detectPhases_3* that implements the method explained previously, we label the data adding a new column to the table called ID. After that we delete unnecessary columns for the training (defined by program, associated to the specific LPMSB2 sensor) and rows the *detectPhases_3* function was not able to label. The last step is to split labels and features and transpose the table. At the end we obtain two data structures

- a Y containing for each file a single row where are listed the label associated with each timestamp;
- a X containing for each file the feature stream. Every row corresponds

to a different feature, and every column is associated with each timestamp, as for the labels.

These data shapes are necessary both for training and testing purposes; the function takes as input a cell array of table and returns two cell array structures X and Y, where each cell is a matrix/array where features/labels are listed.

5.2 Neural network structure

To solve this classification problem that takes a datastream as input, a recursive neural network was chosen. The network structure is defined as follow

1. A sequence input layer that deals with the data stream input, as option the number of feature is specified;
2. "Recurrent layer", formed by a specified number of recurrent hidden layers. This layer can belong to two different architectures: GRU or LSTM. As option is also specified the output mode as sequence, so that the network will assign a label to every timestamp;
3. A fully connected layer as decision layer, as parameter the number of labels to classify;
4. A softmax layer, as activation function;
5. A classification layer, to resolve the requested problem.

5.3 Training and evaluation

To test multiple networks with different parameters we defined a structure where save the parameters used for the network definition/training, the trained network and the achieved accuracy. To evaluate the accuracy we used three different values

- Phase accuracy, calculated on all test data, which express how precise is the network to predict each different phase;
- Test accuracy, calculated on all test data, which express how precise is the network to classify the entire dataset of each file;

- Stream accuracy, calculated on a single file using a specific function: simulateStream. It simulates a datastream from the sensor and upon a new timestamp is received, it predicts the associated label. The accuracy is computed between labels predicted in this specific way and the expected ones.

The parameters we vary when training a network are both in the structure and in the training options. In the structure we vary the architecture of the recursive layers(GRU or LSTM) and the number of hidden layers (50, 100 or 150), in the option we modify the number of epochs (150, 175 or 200) and the gradient threshold (1, 1.5 or 2). In total we trained 54 networks. We obtained that the most performing network has the following parameters: GRU architecture, 50 hidden layers, 150 epochs for training, gradient threshold equal to 1. This network has the best values of accuracies on phases: respectively 0.8720, 0.9228, 0.9631, 0.8998 (with mean equal to 0.9144, standard deviation of 0.0385) and stream: 0.9358. In the test accuracy it scored a mean value of 0.9145, very close to the phase accuracy. This result is promising because it shows that the faster and simpler net is also the most accurate.

The network with best accuracy on test is always a GRU architecture network, with 150 hidden layers, that scored a mean equal to 0.9258; however its phases accuracy are: 0.9388, 0.8245, 0.9572, 0.9069 (with mean equal to 0.9069 and standard deviation of 0.0587). So, differently from the first case where phase and test mean accuracies are almost equal, in the second case the two values are very different. This means the second net is more capable of recognizing some phases than others. More specifically, the lower score is on the second phase, which is also the one with more variability and harder to perfectly detect with the labeling algorithm.

So we can conclude that the fist solution is the best one given its higher mean and lower standard deviation on the phase accuracy, leading to a higher accuracy in the stream simulation.

The following graphs show for both the architecture types, as function of each different parameter varied during the train, the mean with respect to the other parameters of stream, phase and test accuracy.

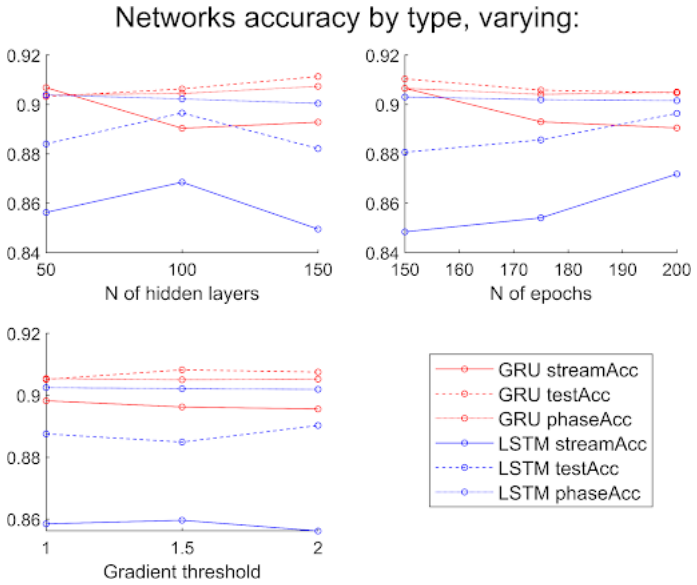

Figure 16: Mean accuracies with respect to the variations of different parameters of the network and its training.

We can deduce that the gradient threshold hasn't much impact on the performance of the network. About the number of epochs, it seems that the GRU architecture tends to lower the accuracy when increasing the epochs, viceversa the LSTM architecture benefits from more epochs in training. Probably, being the GRU type more simple, it soffers earlier overtraining; opposite case for the LSTM, which is more complex.

Looking at the different values for the hidden layers number, it's hard to identify specific trends, but for the LSTM it seems the optimal value is 100, and for the GRU type we see that a growing number of epochs had significant negative impact on the phase accuracy, while the other figure of interest slightly increase; given this info and the previous results, a lower number of hidden layers is the best solution for the GRU architecture.

This analysis justifies the resulting best network.

5.4 Unsupervised learning

As briefly mentioned above, unsupervised learning has been implemented in order to have a tool for comparing performances and properly judging the final accuracies.

In particular, k-Means has been exploited: this algorithm consists in clustering data according to their similarity. Since each cluster is represented

by its mean, at the first step these parameters are randomly chosen.

Step by step, data are assigned to the class with the closest mean: once the assignment procedure is completed, means are re-computed and the process is repeated.

Eventually, when no more changes in means occur, the last assignments are returned.

The algorithm has been executed both on input space and feature space: the dimensions of those are respectively 12 and 84 (7 feature mapping for 12 dimensions). In particular, the considered map were

- Mean
- Median
- Variance
- Range
- Root mean square level
- Mode
- Mean or absolute deviation

Unfortunately, with this approach performances dropped dramatically, achieving an accuracy between 20% and 30%. This variation is due the stochastic initialization of the algorithm.

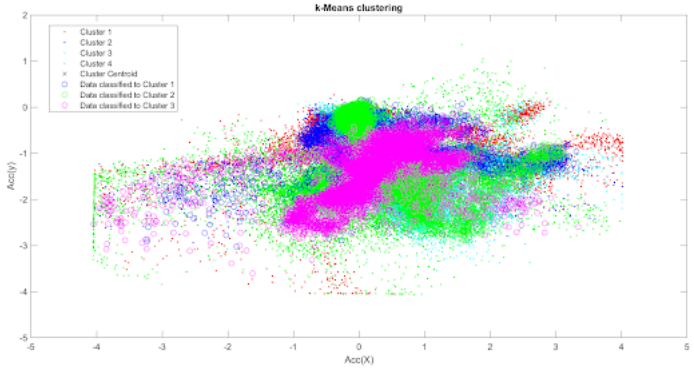

Figure 17: k-Means on raw data (12 dimensional space)

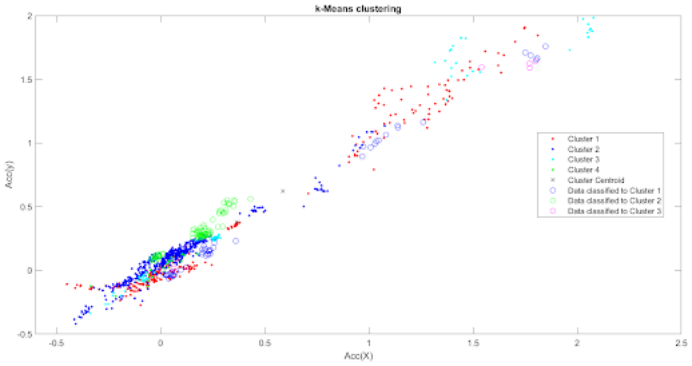

Figure 18: k-Means on feature mapped data (84 dimensional space)

6 Conclusion

We can conclude that the threshold labeling algorithm, validated by pressure sensor data, has been able to correctly classify the gait phases.

Moreover, the implementation non-linear machine learning methods works very well in classifying data coming from an IMU sensor.

In particular, recurrent neural networks allow reaching very high performances while clustering algorithms or, more in general, non- sequential related algorithms leads to worse performances.

References

[1] H. T. T. Vu, D. Dong, H.-L. Cao, T. Verstraten, D. Lefeber, B. Vanderborght, and J. Geeroms, "A review of gait phase detection algorithms for lower limb prostheses," *Sensors*, vol. 20, no. 14, p. 3972, 2020.

[2] D. Gouwanda and A. A. Gopalai, "A robust real-time gait event detection using wireless gyroscope and its application on normal and altered gaits," *Medical engineering & physics*, vol. 37, no. 2, pp. 219–225, 2015.

[3] M. Zakria, H. F. Maqbool, T. Hussain, M. I. Awad, P. Mehryar, N. Iqbal, and A. A. Dehghani-Sanij, "Heuristic based gait event detection for human lower limb movement," in *2017 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, pp. 337–340, IEEE, 2017.

[4] P. Catalfamo, S. Ghoussayni, and D. Ewins, "Gait event detection on level ground and incline walking using a rate gyroscope," *Sensors*, vol. 10, no. 6, pp. 5683–5702, 2010.