Microprocessors and Microsystems 77 (2020) 103118

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Analyzing gait symmetry with automatically synchronized wearable sensors in daily life

Tobias Steinmetzer*, Sandro Wilberg, Ingrid Bönninger, Carlos M. Travieso

Tobias Steinmetzer, Universitätsplatz 1, Senftenberg 01968, Germany

ARTICLE INFO

Article history:
Received 15 May 2019
Revised 15 April 2020
Accepted 21 April 2020
Available online 15 May 2020

Keywords:
Synchronization
Gait analysis
Inertial sensors
Dynamic time warping
Convolutional neural networks
Symmetry

ABSTRACT

Gait deviations such as asymmetry are one of the characteristic symptoms of motor dysfunctions that contribute to the risk of falls. Our objective is to measure gait abnormalities such as asymmetry of the lower limbs in order to evaluate the diagnosis more objectively. For the measurement we use inertial measurement unit (IMU) sensors and force sensors, which are integrated in wristbands and insoles. To extend the battery life of wearable devices, we only save data of the activity *gait* within the wearables. Therefore we perform activity recognition with a smartphone. Using convolutional neural network (CNN) we achieved an accuracy of 94.7 % of the activity *gait* recognition. Before recording we synchronize the wearable sensors and reach a maximum latencies of 3 *ms*. Before the analysis of the symmetry we detect the strides by using a CNN with an accuracy of 98.8 %. For the symmetry evaluation we used dynamic time warping (DTW). The DTW enables us to calculate symmetry of the complete time series of human gait.

1. Introduction

One of the greatest achievements of modern medicine is that the life expectancy of the global population is continuously increasing [1]. However, it also creates new problems. The epidemiology of old-age diseases is a significant part of the problem. These include, for example, Parkinson's disease (PD) [2]. PD is often combined with motor dysfunctions. For this reason, it is essential to assess the patient's motor skills at regular intervals. Furthermore, the measurement of motor skills can be used for conclusions about the progress of the disease and treatment success. Bradykinesia, rigor, tremor, postural instability, and walking disorders are typical symptoms of the disease. The aim of our cooperation with the Niederlausitz Clinic Senftenberg is to measure gait abnormalities in PD, such as the asymmetry of the upper and lower limbs in daily life. Thus, the diagnosis can be more objective and therapy more effective. Furthermore, the continuous measurements should give the patient feedback to the therapy success and thus can be motivated for the therapy. For this reason, we created a mobile system that provides objective measurement data to the physician to be able to evaluate motor disease quantitatively. Gait deviations such as asymmetry are one of the characteristic symptoms of patients with PD that contribute to the risk of falls [3]. For the continuous measurement of motion in daily life, we propose the use of wearable microcontroller-based systems. For this reason, we propose a system that:

1. uses sensors that the user can comfortably carry the whole day
2. records and transmits data in an energy-saving manner
3. synchronizes the arm and leg sensors in real-time
4. filters out the activity *gait* from any *other* movement activity
5. analyses the symmetry of legs individually and to each other

This work is divided into seven sections. Relevant works for this paper are introduced in section II. Section III describes the developed hardware, the test performed by the users. All carried out procedures we need for the analysis of the A-symmetry are described in Section IV. The results achieved are presented in Section V. Section VI indicates the discussion and Section VII the conclusion and further development.

2. Related works

For the symmetry analysis of the gait in daily life, time series must be recorded with the wearable sensors. Therefore it is necessary to extract the activity *gait* from all activities such as sitting, standing, climbing stairs, or walking. To be able to calculate symmetry values from this time series, the system must work synchronously.

* Corresponding author.
E-mail addresses: tobias.steinmetzer101@alu.ulpgc.es, tobias.steinmetzer@b-tu.de (T. Steinmetzer).

https://doi.org/10.1016/j.micpro.2020.103118
0141-9331/

2.1. Sensors

Already in 1992, motion and symmetry of the lower extremities were recorded with cameras and markers [4]. Technological advances and the cost-effective development of depth cameras have opened up new possibilities for motion analysis by Kinect from 2010. The depth camera extended the RGB camera. Thus the gait could be analyzed with new methods [5]. The disadvantage of camera systems is that they are stationary.

In the following years, the use of gyroscopes, accelerometers magnetometers, and force sensors were further developed, and the popularity of the sensors increased, as the many smartphones and game console controllers are equipped with these sensors. The advantage of these sensors is that these are integrated into small devices and that they are wearable. Thus it is possible to analyze the human gait independent of location. These sensors are often used to determine strides or activities [6–15].

2.2. Activity recognition

Many smartphones have a gyroscope, accelerometer, and magnetometer. In many studies, this has been used to try to identify the activities of people [13–15]. One possibility to implement this is to choose a fixed window width and collect all statistical values for this window, which serve as a characteristic for the classification. The use of a neural network has proven to be useful here [14,15]. Another possibility is the use of CNN's [13].

Activity detection is usually used to reflect the time a person has been moving throughout the day. This is sufficient for an activity estimation of a person in general. The quality of smartphone sensors is adequate to estimate the activity of a person.

2.3. Symmetry

The situation is different when wearables assess diseases related to movement disorders. In this case, IMU sensors are attached to specific joints or integrated into clothing. To measure and store the time series of gait wearable containing microcontrollers in combination with IMU sensors are often used [6–11,16–22]. In most cases, the motion of the lower extremities is measured [6–11,22]. Thereby conclusions can be made about the stride length, cadence, stride duration, gait phases, and symmetry [6,10,11,23].

There are different methods for the calculation of symmetry. One approach is that different calculated features like step length, step duration, standing time, or swing time of the legs are put into relation [16,20,24]. The disadvantage of this method is that only average values of the calculated characteristics for the gait can be assessed, but not the entire time series. This is different for stationary systems, which are camera-based. With these systems, the complete body can be recorded synchronously [17]. Both types of symmetry evaluation are useful. However, in our opinion, a direct comparison of the time series is most useful, because differences in the related arms and strides can be measured directly.

The symmetry of arms and legs, as well as the symmetry of the upper and lower limbs with each other, investigate only a few papers [25–27]. Changes of interlimb coordination in individuals with PD and healthy older adults while systematically manipulating walking speed are compared to determine the impact of PD symptoms on interlimb coordination [25]. Markers were placed on the foot, heel, ankle, knee, hip, thigh, wrist, elbow, shoulder, and head. A point estimate of the relative phase (PERP) between body segments was calculated by using the moment at which the positive maxima were reached for the angle of each body segment. To assess change in asymmetry over time is the objective in [26]. The changes in movements are assessed by a single neurologist specializing in movement disorders. A robust ordinal logistic regression model that includes a control for clustering due to repeated observations within-person for evaluating the relative change in asymmetry is used.

Another system focuses on the study of the impact of PD on synkinesias (i.e., the symmetry of movement) during walking, and the effect of medication on the gait symmetry [27]. Every patient was tested and measured using IMU-sensors in his ON and OFF state. The trend symmetry value is calculated as a ratio of the variabilities of two eigenvectors, which are calculated from the kinematic motion data of the left and right limb. An up-to-date overview of symmetry analysis systems for movements is shown in [28].

The use of identical time points for the determination of A-symmetry is of the highest importance. That means data transmission has to be synchronized.

2.4. Synchronization

The video-based systems have a synchronized recording of all extremity movements. The disadvantage is that the measurements cannot be carried out in daily life. Only camera systems for laboratory measurements were found in the literature [27]. Wearable systems, in contrast, could be an alternative for making symmetry measurements of gait in daily life, but they are not time-synchronized.

To closing this gap, the microcontrollers must be synchronized with each other. Several approaches have already been pursued this. A possible solution is to build up a sensor network in which the sensors are connected by wires [27]. Another work presents a system where a docking station serves as a charging station and for synchronization [18]. The docking station can synchronize four wearable sensors, but it has a time drift after a longer runtime. Others use the system of MbientLab [20]. To determine the symmetry of the gait, we need four synchronized sensors (one at each limb). In earlier works, we had tested the system of Mbientlab, but it can only record three synchronized sensors [20].

Advantages and disadvantages of current systems:

- Camera-based systems can measure synchronized time series of each limb. But they are stationary and therefore not suitable for measurements in everyday life.
- A smartphone is useful for detecting gait activities. But it's too imprecise for clinical measurement.
- IMU systems are an alternative to camera-based systems. But they have to be synchronized.

To calculate the gait symmetry of time series using wearable sensors in daily life, we propose a system with two wristbands with IMU sensors, two insoles each with one IMU and ten force sensors, and a smartphone for activity detection. For the measurements, we synchronize all sensors if the activity walk is detected in daily life. We propose a method to calculate the symmetry from all measured values of the gait cycle instead of the symmetry calculation with parameters. For our prototype, we only used data from healthy subjects.

3. Material

3.1. Hardware

3.1.1. Smartphone

For activity detection, we used various smartphones and tablets with the Android operating system. To be independent of a specific device. However, the device must be able to provide linear acceleration and rotation data. We recorded both sensor data with

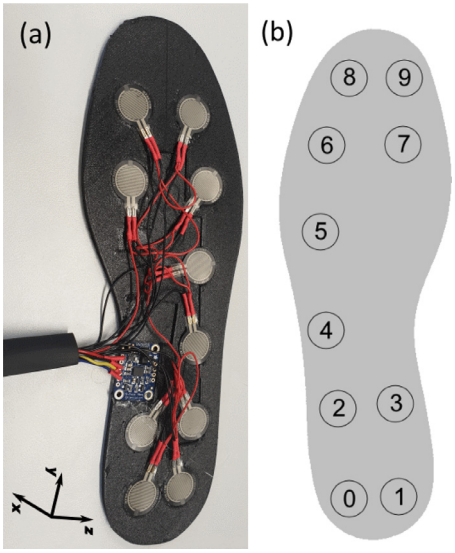

Fig. 1. Insole with force and IMU sensor.

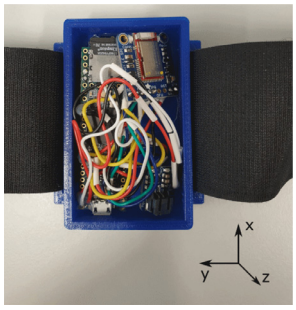

Fig. 2. Wristband and IMU sensor.

a frequency of 50 %. To make the system energy efficient, we use smartphones for activity detection. Only if the smartphone has detected the activity *gait*, the wearables record the data.

3.1.2. Wearables

For the motion measurement, we use Bosch BNO055 IMU sensors consisting of a gyroscope, accelerometer, and magnetometer [29]. The sensors are mounted in insoles, see Fig. 1, and in wristbands, see Fig. 2. The BNO 055 sensor has an integrated co-processor for the sensor fusion that calculates the absolute orientation and linear acceleration. So angle velocity, acceleration, quaternions, Euler angles, and linear acceleration values are received with 100 *Hz*. For force measurement, we use ten FSR 402 force sensors in the insoles [30]. Furthermore, the natural rolling motion can be measured by the horizontal arrangement of the sensors. The parallel arrangement of the sensors should later make it possible to calculate the balance, see (b) in Fig. 1. The insoles were printed with a 3D printer. We used flexible material to achieve a lower bias by using the insoles. To get a lower bias and higher comfort by a foreign insole, we made the insole out of flexible material.

3.2. Data set

3.2.1. Activity recognition

In this data set, we use recordings of 20 healthy subjects to test the system. For this propose, we developed an Android App. The subjects specified the start, end, and type of activity via the App before each recording. We recorded linear acceleration and rotation data of the Android operating system with a frequency of 50 *Hz*. The users had to specify in which activity they performed. In total the following activities were recorded *gait, cycling, go stairs, lying, sitting, standing, smartphone lying around* (table or desk), *smartphone in use* (writing a message or play a game), and *use transport* (drive by car or train). We have reduced the problem to a binary problem and use in the following only the classes *gait* and *other*. The class *other* contains the activities *cycling, go stairs, lying, sitting, smartphone lying around, smartphone in use, standing, and use transport*.

3.2.2. Daily life

For the daily life data set, we have a total of 7 recordings of 7 different healthy persons. The age of the persons was between 25 and 54 years. The persons passed the following test:

1. sitting on a chair for 1 minute
2. stand up and standing for 1 minute
3. walking for 1 minute
4. ascending stairs over three floors
5. descending stairs over three floors
6. walking for 1 minute
7. standing for 1 minute in front of the chair
8. sit down
9. sitting on a chair for 1 minute

4. Methods

4.1. Methodology

The whole process for the recognition of gait data is based on the communication between our Android App and four wearable devices (two wristbands and two insoles). Fig. 3 shows the process. We have separated the functional tasks of the smartphone and wearables with a dotted line. However, the wearables work only as slaves, so the smartphone must always send a signal for starting a function. For this reason the tasks *Stop Recording, Data Transmission, Synchronization* and *Start Recording* are involved by both devices.

At the beginning of the workflow, we make an activity recognition. Thus, we want to distinguish the activity *gait* again, the activity *other*. The activity detection is designed to keep the wearable sensors in standby mode until the activity *gait* is detected. This activity detection extends the usage time of the wearable devices.

When a person does the activity *gait*, the app checks if a recording is in process. If not, the wearable devices have first synchronized, and then the recording of the movement starts. When

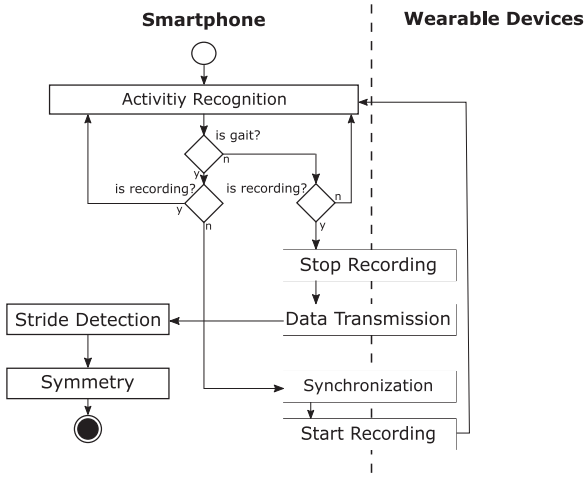

Fig. 3. Process of synchronize, record, and evaluate data.

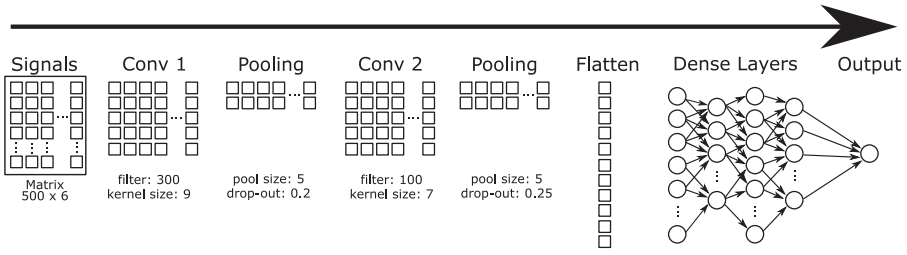

Fig. 4. Schema of the CNN layers for activity recognition.

the activity *other* as *gait* has been detected, and the recording is in process, the recording stopped, and the data transmitted to the smartphone.

For the symmetry calculation, we need a more accurate detection of the strides than with the activity recognition. For this reason, we perform a stride detection by using CNN to detect individual strides of the foot. After that, the symmetry of the strides can be calculated.

4.2. Activity recognition

To enable energy-efficient use of the wearable devices, they are only powered when they are in use. The energy-efficient use means that the wearable devices only have to be switched on during recording. For this reason, we decided to use a binary activity classifier in the smartphone device. This classifier enables us to distinguish the activity *gait* from *other* like cycling, go stairs, lying, sitting, smartphone lying around, smartphone in use, standing, and use transport.

For the activity detection, we use data of the linear acceleration and rotation data of the Android operating system (OS) at a frequency of 50 *Hz*. As features, we use a fixed window width of 10 *s* and an overlap of 50 %. We use as input the x-, y-, and z- axis of the linear acceleration and rotation data of the complete window as input for a 1D CNN classifier. Fig. 4 shows the design of the CNN. We chose CNN because other researchers have also achieved good results with CNN [13,14].

For the construction of the model, we use the activation function rectified linear unit (ReLU) function except for the output layer. The first layer is a convolutional layer with 300 filters and a kernel size of 9. Next is a max pooling with a size of 5 and a drop-out with 0.2. Then follows another convolution layer with 100 filters and a kernel size of 7. Then again, a max pooling with a size of 5 and a drop-out with a probability of 0.25. Next comes a flatting layer. In the following, there are different dense layers with 30, then 10, and finally 50 neurons. The last layer is the output layer, which uses a sigmoid function as the activation function.

For training, we have separated the data by persons. This ensures that the same person is not included in the training and test data set. We split the data that 66 % (13 subjects) are used for training, and 34 % (7 subjects) for testing. During training, we use different epochs and batch sizes. In our case, the setting of 100 epochs and 100 batch sizes has proven good results.

4.3. Synchronization

4.3.1. Process

The synchronization takes place according to the following scheme, see Fig. 5. The master device is the smartphone, and the slaves are the four wearable devices.

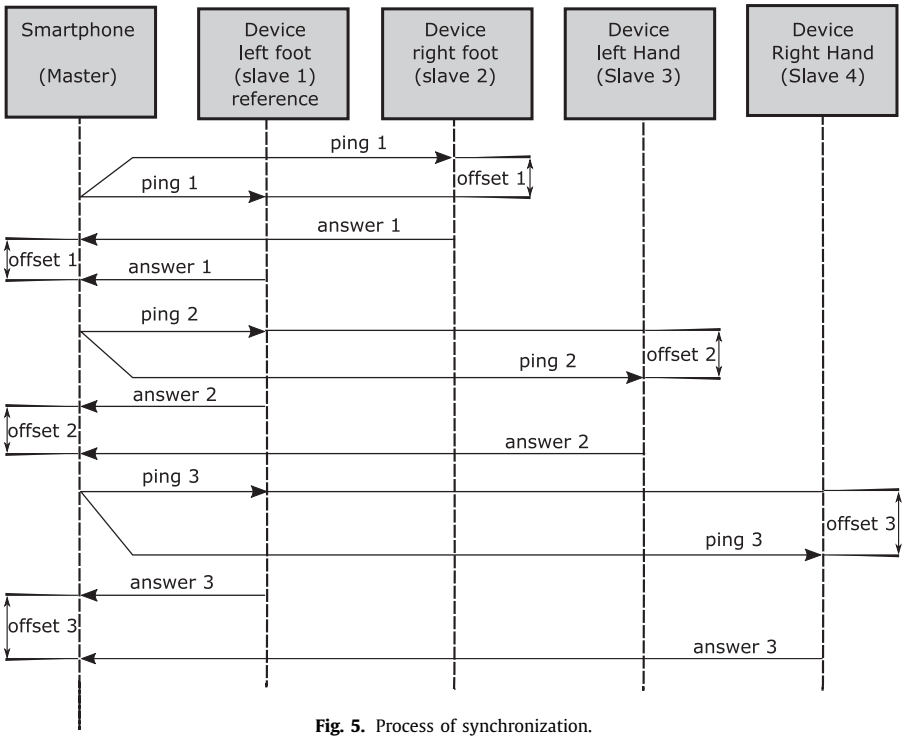

Fig. 5. Process of synchronization.

1. The master sends an empty Bluetooth packet to the first slave (reference slave) and the second slave.
2. Each slave replies with its device time (milliseconds since power-on) as soon as it receives the packet of the master.
3. The master receives the responses and measures the time difference between them.
4. To avoid random response times (e.g., caused by physical influences), the process is repeated multiple times.
5. The median is the time difference between the reception of the packets by the reference slave and the second slave.
6. The master calculates the offset of the second device (see formula 2) and sends it to the second device.
7. Now, the second slave knows its offset compared to the reference slave.
8. Steps 1 to 7 are repeated with the first slave and the third slave.
9. Steps 1 to 7 are repeated with the first slave and the fourth slave.

As a result, we get the offsets between the reference device and the other three measuring devices. The four measuring devices record data synchronously. Unfortunately, a drift can occur between the devices. To prevent this, the synchronization is repeated before each recording.

4.3.2. Verification test setup

To be able to measure the Bluetooth latency correctly, we have wired the microcontrollers to each other. Two microcontrollers are connected by wire using their serial output and input. The output pin of one device is connected with the input pin of the other device and vice versa. In the beginning, the outputs are set to "low". One of the two devices now initiates a "high" output and starts a timer. The other device registers this event by reading a "high" input and answers to the first device by setting its output pin to "high", too. The first device stops the timer as soon as it registers a "high" input. According to repeated measurements, the serial latency is lower than 1 ms Because this latency measurement does not influence the synchronization latency, it is ignored in the following section.

4.3.3. Latency bluetooth

In the next step, we measure the latency and offset of the Android OS to send a Bluetooth packet to the microcontrollers. Furthermore, we also measure the time at which packets are received if the microcontrollers have sent them. For the measurement of the offset between the microcontrollers, we use the test setup from Section 4.3.2. For the test setup, the microcontrollers (M1) and (M2) were placed at the same distance to the Android smartphone (A) so that the same signal strength exists for all devices. Otherwise, this can corrupt the result. The results of the measurement can be seen in Table 1. The first and second columns *A send M1* and *A send M2* are the time stamps of the Android OS in milliseconds (*ms*) when the commands were executed. *Diff 1* is the difference in *ms* between column $(AsendM2) - (AsendM1)$. The *RL* column is the wired offset between the two microcontrollers measured using the method in the 4.3.2 section. This offset is the real offset. *R1* and *R2* are the timestamps in *ms* of the received Bluetooth packets of the microcontrollers. The last column *l* reflects the difference of the columns $R2 - R1$ in *ms*. This offset, the Android OS uses to calculate the device offsets. Perfect synchronization is achieved when $RL = l$.

The measurement was repeated three times. Therefore each measurement is separated in the table by a double line. The columns *RL* and *l* show a correlation to each other. For this reason, it was written in bold.

The Table 1 shows that the packets are sent with different priorities by the Android OS. Thus the column *Diff 1* does not correlate with *RL*. The values of the three measurements of *A send M1* and *A send M2* by Pearson correlation give the following results 0.034, -0.272, -0.617. This means that there is no correlation. On

Table 1
Latency between Android and two microcontrollers between sending and receiving time for three different executions. *A send M1* and *A send M2* are the timestamps when the commands are executed by the Android OS. *Diff 1* is the difference of $(A\ send\ M2) - (A\ send\ M1)$. *RL* is the wired latency between both microcontrollers when receiving the packets. *R1* and *R2* are the times when the Android OS has received the packets from the microcontrollers. *l* is the difference of $R2 - R1$. Bold shows the correlation between *wired* *l* and calculated *l*.

A send M1	A send M2	DIFF 1	RL	*R1*	*R2*	*l*
1552473386320	1,552,473,386,332	12	**23**	1,552,473,386,373	1,552,473,386,396	**23**
1552473402974	1,552,473,402,985	11	**27**	1,552,473,403,025	1,552,473,403,054	**29**
1552473411800	1,552,473,411,804	4	**23**	1,552,473,411,851	1,552,473,411,872	**21**
1552473415820	1,552,473,415,829	9	**30**	1,552,473,415,863	1,552,473,415,897	**24**
1552473418088	1,552,473,418,100	12	**28**	1,552,473,418,131	1,552,473,418,160	**29**
1552473420520	1,552,473,420,527	7	**28**	1,552,473,420,564	1,552,473,420,593	**29**
1552473422598	1,552,473,422,604	6	**31**	1,552,473,422,638	1,552,473,422,672	**34**
1552473424513	1,552,473,424,523	10	**29**	1,552,473,424,557	1,552,473,424,585	**28**
1552473426282	1,552,473,426,295	17	**28**	1,552,473,426,327	1,552,473,426,355	**28**
1552473428201	1,552,473,428,209	8	**28**	1,552,473,428,239	1,552,473,428,267	**28**
1552475001845	1,552,475,001,849	4	**43**	1,552,475,001,888	1,552,475,001,929	**41**
1552475013106	1,552,475,013,111	5	**19**	1,552,475,013,159	1,552,475,013,179	**20**
1552475014617	1,552,475,014,624	7	**33**	1,552,475,014,658	1,552,475,014,730	**72**
1552475015951	1,552,475,015,964	13	**32**	1,552,475,015,996	1,552,475,016,029	**33**
1552475017448	1,552,475,017,460	12	**23**	1,552,475,017,495	1,552,475,017,517	**22**
1552475018910	1,552,475,018,920	10	**24**	1,552,475,018,958	1,552,475,018,979	**21**
1552475020302	1,552,475,020,309	7	**31**	1,552,475,020,346	1,552,475,020,380	**34**
1552475021704	1,552,475,021,713	9	**31**	1,552,475,021,751	1,552,475,021,780	**29**
1552475022982	1,552,475,022,995	13	**29**	1,552,475,023,021	1,552,475,023,049	**28**
1552475024662	1,552,475,024,670	8	**28**	1,552,475,024,709	1,552,475,024,737	**28**
1552477976517	1,552,477,976,523	6	**42**	1,552,477,976,564	1,552,477,976,605	**41**
1552477978212	1,552,477,978,222	10	**29**	1,552,477,978,258	1,552,477,978,287	**29**
1552477979992	1,552,477,980,003	11	**16**	1,552,477,980,046	1,552,477,980,067	**21**
1552477981607	1,552,477,981,616	9	**31**	1,552,477,981,651	1,552,477,981,685	**34**
1552477983153	1,552,477,983,165	12	**31**	1,552,477,983,193	1,552,477,983,227	**34**
1552477984567	1,552,477,984,581	14	**31**	1,552,477,984,611	1,552,477,984,645	**34**
1552477986029	1,552,477,986,036	7	**43**	1,552,477,986,078	1,552,477,986,120	**42**
1552477987445	1,552,477,987,455	10	**28**	1,552,477,987,486	1,552,477,987,515	**29**

the other hand, the columns l and RL correlate strongly by Pearson correlation with the following values 0.743, 0.580, 0.982.

This can be explained by the fact that the microcontrollers M1 and M2 process the commands sequentially, and thus all commands are equally authorized. Furthermore, it is possible to receive signals at the app in real-time because there are several threads available.

Out of this knowledge, we can say the receive time of the smartphone is the offset in which the microcontrollers sent the signal. We use this fact for synchronization. In summary, we can note that when sending packages of two microcontrollers at the same time, these also arrive simultaneously.

4.3.4. Synchronization algorithm

Based on the data from sections 4.3.2 and 4.3.3 we can now propose a solution to synchronize two microcontrollers via Bluetooth. The following steps describe the procedure of the algorithm:

1. The Android device A sends a packet to microcontrollers $M1$ and $M2$.
2. $M1$ sends a packet to A. The packet holds a timestamp of the system time $t1$ directly before sending it.
3. $M2$ sends a packet to A. The packet holds a timestamp of the system time $t2$ directly before sending it.
4. A receives a packet from $M1$ at a real-time $R1$.
5. A receives a packet from $M2$ at a real-time $R2$.

With this information we can calculate our receive latency l in formula (1) where c represents an possible error.

$$l = R2 - R1 + c \tag{1}$$

To determine the offset o from $M2$ to $M1$, we use the formula (2).

$$o = (t2 - t1) + l \tag{2}$$

Of course, in the proposed algorithm, errors can occur, which are the result of disturbances in the magnetic field or other physical effects. Therefore we perform the algorithm eleven times and use the median of the latency to determine the best synchronization between the devices $M1$ and $M2$.

4.4. Stride detection

After the activity recognition algorithm has identified the phases of the activity *gait*, we use the transmitted, synchronized sensor data of the wearable devices to perform a stride detection. In the version of our proposed system, we used only the insole data for the stride detection and symmetry calculation. To train the classifier, first, we manually labeled the data After labeling, the data was normalized and resampled to a uniform length. Then, we trained our CNN to get a model for stride detection.To detect strides from the daily life data set, we use automatic framing to extract fragments from the recording. By fragments, we mean different parts of a recording. These fragments we normalized and resampled. The CNN model classifies these fragments for possible strides. This process is shown in Fig. 6.

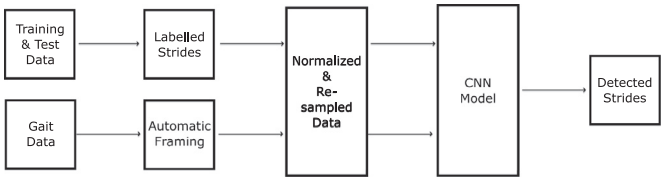

Fig. 6. Process of stride detection with CNN.

4.4.1. Normalization

We use a Min-Max-Normalization to normalize all data between range 0 to 1. The normalization were executed for every $x_i \ \forall \ i \in \{0, \ldots, N-1\}$ of the feature X, where N is the length of feature X, see formula 3. The result is a normalized vector X^{norm} with the values $x_i^{norm} \in X^{norm}$. The functions $min(X)$ and $max(X)$ return the minimum and maximum of the feature X.

$$x_i^{norm} = \frac{x_i - min(X)}{max(X) - min(X)} \tag{3}$$

4.4.2. Resampling

For using the CNN classifier, we need a uniform signal length. Therefore we transform all signals to a uniform length of 100 values. The Python function *resampling* in the library SciPy use a Fast Fourier Transformation (FFT) based method [31].

4.4.3. CNN

To build the CNNs model, we use a sequential network, see Fig. 7.

As the activation function, we use the rectified linear unit (ReLU) function with except at the output layer. As input we use the x-, y-, and z-axis of the linear acceleration and Euler angles. The first one-dimensional convolutional layer creates 100 filters with a kernel size of 3. To reduce the filters, we apply a max pooling with a pool size of 3 and a drop-out with probability 0.2. After the second convolutional layer consists of 100 filters and a kernel size of 5, this is followed by max-pooling again with a pool size of 3. A drop-out follows them with probability 0.2, where single connections are randomly deleted [32]. After that, we have a flattening layer to adjust the dimensions for the neural network (NN). Next, we have two dense layers. The first has 20 neurons, and the second 30 neurons. Last we have an output layer with a sigmoid function as the activation function. As a result, we obtain a probability of the signal being a stride.

4.4.4. Automatic framing

A real signal cannot be manually labeled. Thus, an algorithm should be that task. For this reason, we use automatic framing. The automatic framing creates dynamic window sizes, which we use as input for the CNN in the stride detection. For detection, we use dynamic window sizes. The average duration of a stride is 1.1 s [33] that is equivalent to 110 values of the data. Therefore we use an average window size of 110 $\pm$ 30 values. The window w can

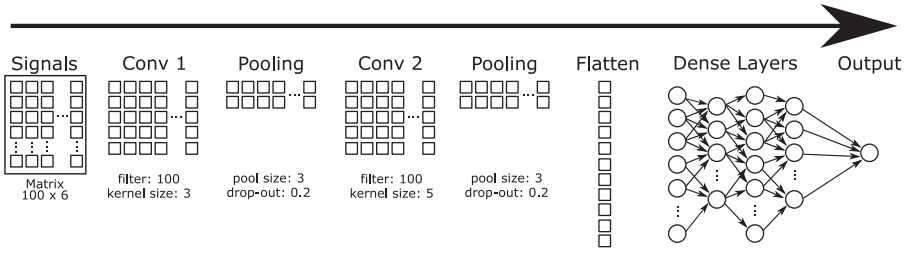

Fig. 7. Schema of the CNN layers.

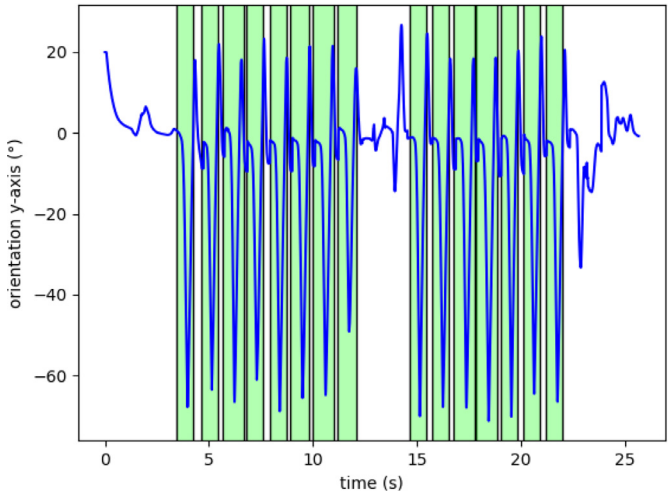

Fig. 8. Predicted strides.

have the following sizes

$$w = \{80, 90, 100, 110, 120, 130, 140\}. \quad (4)$$

From the signal to be classified, windows with all sizes are used.

All dynamic windows are resampled to a uniform length of 100 values and normalized with the functions from Section 4.4.1 and 4.4.2. If the CNN detects a stride with more than 70% probabilities within a window, it is saved in a list. The stride with the highest probability from the list is only used and defined a valid stride. We mark the absolute minimum within a stride $\pm$ 10 *ms*. To distinguish the strides from each other, we use overlapping, see Eq. 5. Overlapping allows us to separate new strides from others. If the predicted stride lies within this range, it is marked as detected stride, see Fig. 8.

$$overlapping = detected\ stride + (average\ stride \cdot 0.8) \quad (5)$$

4.5. Symmetry

The calculation of the symmetry of two strides is possible by the previously performed synchronization in Section 4.3. To calculate the symmetry, we first do a stride detection like in Section 4.4.

So that we always have a fixed reference timestamp (e.g., right foot) point within a stride. We use the time stamp of the minimum inside a stride, see Fig. 9.

To measure the symmetry distance between the time series (strides) of the right and left foot, we use the DTW. DTW has become very well established in the analysis of time-series signals. In contrast to Euclidean distance, this method can compensate for time warping. Based on this flexibility, it is a popular method for the analysis of time series in medicine, science, and industry. The idea with DTW is that not the distance of two indices is calculated, but the distance to the most fitting one. Thus allows comparing time series with each other if they recorded with different duration or frequency.

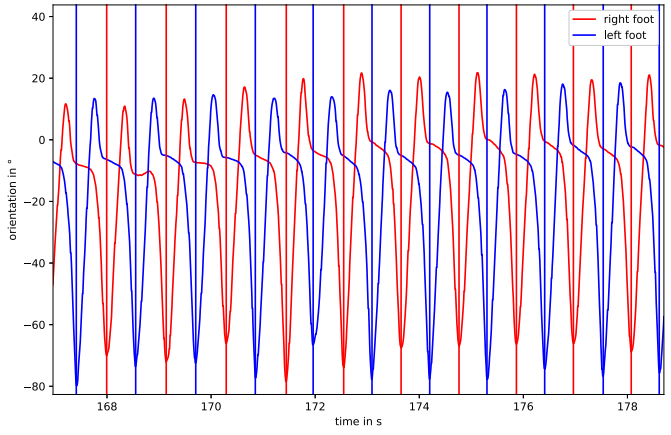

Fig. 9. Orientation data of the left and right foot with the corresponding minima.

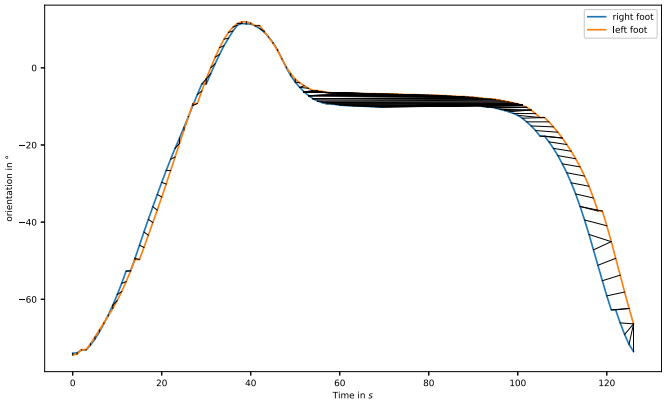

Fig. 10. DTW Symmetry result at person 1 with a distance value of 130.52.

In the first step, the algorithm calculates distances between the time series $(x_i)_{1 \le i \le n}$ (e.g. orientation angle of the right foot) of length n and $(y_j)_{1 \le j \le m}$ (e.g. orientation angle of the left foot) of length m, resulting in a n times m matrix $D = D_{ij}$ containing distances D_{ij} between y_j and x_i. The distances within the matrix are calculated by the sum of the current distance and the minimum distance of a previous neighboring element, see Eq. 6 [34].

$$D_{ij} = (x_i - y_j)^2 + min\{D_{i-1,j}, D_{i-1,j-1}, D_{i,j-1}\} \quad (6)$$

A distance D_{ij} of 0 means 100% symmetry of the measured values. The higher the value D_{ij}, the lower is the symmetry of the two feet, see Fig. 10.

5. Results

5.1. Activity recognition

For recognition of activity *gait*, we have performed a five-fold cross-validation. The results are shown in Table 2. For the results we have specified precision, recall, F1-Score and Accuracy. For each column we have given the average value and standard deviation.

5.2. Synchronization

In the Tables 3, 4, and 5 the measured values of a synchronization are shown. In the tables, the first column is a numbered index. It is followed by the receiving time of the reference microcontroller and the third column of the to be synchronized microcontroller. Column four is the calculated latency of both microcontrollers, and column five is the wired measured latency over the wires. The last column shows the error from calculated to measured latency. For the most accurate timestamp, we calculate the median of the latencies *l*.

Table 2
Results for recognition of activity *gait*.

	precision	recall	F1-Score	accuracy
CNN	0.958 ± 0.031	0.683 ± 0.023	0.884 ± 0.011	0.947 ± 0.005

Table 3
Latency between microcontroller M1 and M2.

index	M1	M2	l	wired l	c
1	1,552,919,014,830	1,552,919,014,946	116	81	35
2	1,552,919,014,999	1,552,919,015,047	48	8	40
3	1,552,919,015,776	1,552,919,015,154	-622	-665	43
4	1,552,919,015,838	1,552,919,015,897	59	55	4
5	1,552,919,015,950	1,552,919,015,972	22	20	2
6	1,552,919,016,005	1,552,919,016,014	9	6	3
7	1,552,919,016,055	1,552,919,016,064	9	7	2
8	1,552,919,016,098	1,552,919,016,106	8	5	3
9	1,552,919,016,147	1,552,919,016,156	9	8	1
10	1,552,919,016,191	1,552,919,016,199	8	4	4
11	1,552,919,016,240	1,552,919,016,262	22	21	1

Table 4
Latency between microcontroller M1 and M3.

index	M1	M3	l	wired l	c
1	1,552,919,585,669	1,552,919,585,703	34	34	0
2	1,552,919,585,731	1,552,919,585,728	-3	-5	2
3	1,552,919,585,770	1,552,919,585,781	11	8	3
4	1,552,919,585,824	1,552,919,585,827	3	4	1
5	1,552,919,585,862	1,552,919,585,877	15	16	1
6	1,552,919,585,924	1,552,919,585,914	-10	-8	2
7	1,552,919,585,961	1,552,919,585,978	17	17	0
8	1,552,919,586,024	1,552,919,586,014	-10	-8	2
9	1,552,919,586,061	1,552,919,586,078	17	17	0
10	1,552,919,586,124	1,552,919,586,114	-10	-9	1
11	1,552,919,586,161	1,552,919,586,171	10	9	1

Table 5
Latency between microcontroller M1 and M4.

index	M1	M4	l	wired l	c
1	1,552,988,793,978	1,552,988,794,019	41	41	0
2	1,552,988,794,060	1,552,988,794,056	-4	5	9
3	1,552,988,794,110	1,552,988,794,119	9	8	1
4	1,552,988,794,154	1,552,988,794,145	-9	-4	5
5	1,552,988,794,203	1,552,988,794,199	-4	-5	1
6	1,552,988,794,227	1,552,988,794,236	9	9	0
7	1,552,988,794,258	1,552,988,794,262	4	3	1
8	1,552,988,794,295	1,552,988,794,313	18	17	1
9	1,552,988,794,359	1,552,988,794,348	-11	-10	1
10	1,552,988,794,395	1,552,988,794,404	9	11	2
11	1,552,988,794,451	1,552,988,794,442	-9	-8	1

Table 6
Latency between microcontroller M1 and M2.

index	l	c
3	-622	43
8	8	3
10	8	4
6	9	3
7	9	2
9	**9**	**1**
5	22	2
11	22	1
2	48	40
4	59	4
1	116	35

In the Tables 6, 7, and 8 the latencies are shown in sorted and the median is printed bold. All three tables provide a positive error of 1 *ms* to the reference device. Thus, the total latency is 1 *ms*. In other measurements, we have a total error of 3 *ms*. Since we record the sensor data with 100 *Hz*, this error is tolerable for symmetry calculation.

Table 7
Latency between microcontroller M1 and M3.

index	l	c
6	-10	2
8	-10	2
10	-10	1
2	-3	2
4	3	1
11	**10**	**1**
3	11	3
5	15	1
7	17	0
9	17	0
1	34	0

Table 8
Latency between microcontroller M1 and M4.

index	l	c
9	-11	1
4	-9	5
11	-9	1
2	-4	9
5	-4	1
7	**4**	**1**
3	9	1
6	9	0
10	9	2
8	18	1
1	41	0

Table 9
Daily life stride detection.

	recall	precision	F1-Score	Accuracy
CNN	0.978	0.978	0.974	0.988

Table 10
Results of the symmetry calculation.

subject	number strides	DTW median
1	90	130.52
2	45	309.91
3	86	576.42
4	37	351.40

5.3. Stride detection

For stride detection, we have performed a seven-fold cross-validation. The results are shown in Table 9. For the results we have specified precision, recall, F1-Score and Accuracy. For each column we have given the average value and standard deviation [35].

5.4. Symmetry

The Table 10 shows the results for the symmetry of four different healthy persons. The first column is the subject number. Column two is the number of strides used to calculate symmetry. In the last column, the median distance of the DTW is shown. All persons had no motor dysfunctions. Person 3 has the largest symmetry deviation. An earlier operation of one knee probably causes these motor dysfunction. The median distance of DTW from person 3 is shown in Fig. 11. In contrast, the median distance of DTW of person 1 in Fig. 10 is smaller than that of person 3. Thus, the gait symmetry of person 1 is more accurate than that of person 3.

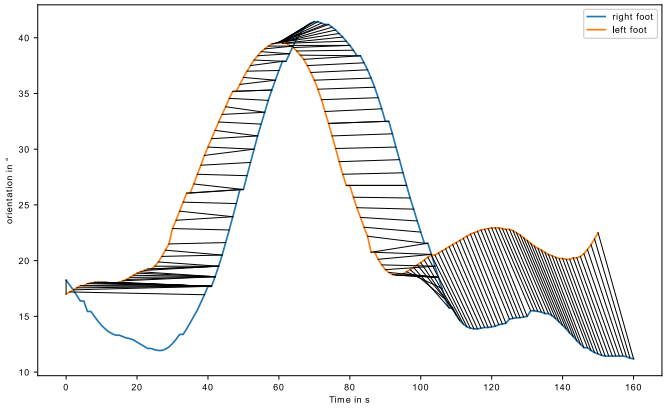

Fig. 11. DTW Symmetry result at person 3 with a distance value of 576.42.

6. Discussion

We have presented a system for real-time analysis of gait symmetry that can be used comfortably by people in their daily life and is independent of the location. The system can be used for measurement of human gait.

By developing an Android app for activity recognition, we were able to show that a smartphone can distinguish between activity *gait* and *other* activities such as standing, lying, cycling, or writing messages. With an accuracy of 94.7% we obtain similar results to other researchers [13–15]. Activity recognition allows us to switch on the wearable sensors, and data recording only activity *gait* is recognized. This method is an energy-efficient solution.

Furthermore, we present a solution to synchronize several wearables sensors. In literature, this problem has already been recognized, and there were several approaches. However, the problem is that the devices of Mbientlab can only synchronize three devices [20]. We synchronize four wearables for four extremities. Another solution was to synchronize the time during charging by cable [18]. However, this solution has the disadvantage that in more extended use, a drift of the clock occurs. We synchronize the wearables before each recording (recognition of activity *gait*). This way, we start each recording without drift of the clock.

In most of the papers dealing with symmetry, they use the stride length, stride duration, and different gait phases to calculate the ratio of the left and right leg [6,10,11,23]. In contrast, our symmetry calculation considers the complete time series. However, the synchronization of the sensors is essential for this. For stride detection, we use a combination of automatic framing and CNN. The use of CNN's for stride detection has proven to be very useful for us. Other work has already been able to benefit from the technology [35]. The symmetry of the legs is analyzed with DTW.

7. Conclusion

With our work, we were able to present a complex system that can analyze the human gait symmetry with the help of wearable devices in daily life. For future work, we want to calculate further features from the time series. By synchronizing the wearable devices, more fundamental symmetry characteristics can be calculated, like cadence, cyclogram, mono pedal phase, or bipedal phase. Other features such as symmetry ratio, symmetry index, gait asymmetry, symmetry angle, stride length, or stride height are also possible. These additional features provide a wide range of features to evaluate human gait. With all these features, more accurate classification of PD stage in the use of machine learning should be possible.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] World health organization, World report on ageing and health, World Health Organization, 2015.
[2] N. Ball, et al., Parkinson'S disease and the environment, Front. Neurol. 10 (2019) 218.
[3] M. Zhang, N. Artan, H. Gu, Z. Dong, L. Burina Ganatra, S. Shermon, E. Rabin, Gait study of parkinson's disease subjects using haptic cues with a motorized walker, Sensors 18 (10) (2018) 3549.
[4] G. Vagenas, B. Hoshizaki, A multivariable analysis of lower extremity kinematic asymmetry in running, International Journal of Sport Biomechanics 8 (1) (1992) 11–29.
[5] O.F. Ince, et al., Gait Analysis and Identification Based on Joint Information Using RGB-Depth Camera, in: 2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), IEEE, 2017.
[6] J. Barth, et al., Stride Segmentation During Free Walk Movements Using Multi-dimensional Subsequence Dynamic Time Warping on Inertial Sensor Data, in: Sensors 15.3, 2015, pp. 6419–6440.
[7] V.N. Bobić, et al., Challenges of Stride Segmentation and Their Implementation for Impaired Gait, in: 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2018, p. IEEE.
[8] X. Tao, et al., Precise displacement estimation from time-differenced carrier phase to improve PDR performance, IEEE Sens. J. 18.20 (2018) 8238–8246.
[9] Koroglu, M. Taha, A. Yilmaz, C.J. Saul, A Deep Learning Strategy for Stride Detection, in: 2018 IEEE SENSORS, 2018, p. IEEE.
[10] J. Hannink, et al., Mobile stride length estimation with deep convolutional neural networks, IEEE J. Biomed. Health Inform. 22.2 (2018) 354–362.
[11] T. Watanabe, T. Miyazawa, J. Shibasaki, A Study on IMU-Based Stride Length Estimation for Motor Disabled Subjects: A Comparison under Different Calculation Methods of Rotation Matrix, in: 2018 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI), 2018, p. IEEE.
[12] Y. Kim, O.S. Eyobu, D.S. Han, ANN-Based Stride Detection Using Smartphones for Pedestrian Dead Reckoning, in: 2018 IEEE International Conference on Consumer Electronics (ICCE), 2018, p. IEEE.
[13] M. Gadaleta, M. Rossi, Idnet: smartphone-based gait recognition with convolutional neural networks, Pattern Recognit. 74 (2018) 25–37.
[14] M.M. Hassan, M.Z. Uddin, A. Mohamed, A. Almogren, A robust human activity recognition system using smartphone sensors and deep learning, Future Generation Computer Systems 81 (2018) 307–313.
[15] L. Cao, Y. Wang, B. Zhang, Q. Jin, A.V. Vasilakos, GCHAR: An efficient group-based context'aware human activity recognition on smartphone, J. Parallel Distrib. Comput. 118 (2018) 67–80.
[16] S. Clemens, K.J. Kim, R. Gailey, N. Kirk-Sanchez, A. Kristal, I. Gaunaurd, Inertial Sensor-based Measures of Gait Symmetry and Repeatability in People with Unilateral Lower Limb Amputation, in: Clinical Biomechanics, 2019.
[17] S.J. Crenshaw, J.G. Richards, A method for analyzing joint symmetry and normalcy, with an application to analyzing gait, Gait & posture 24 (4) (2006) 515–521.
[18] M. Mancini, L. King, A. Salarian, L. Holmstrom, J. McNames, F.B. Horak, Mobility Lab to Assess Balance and Gait with Synchronized Body-worn Sensors, in: Journal of Bioengineering & Biomedical Science, volume 007, 2011.
[19] A.R. Anwary, H. Yu, M. Vassallo, An automatic gait feature extraction method for identifying gait asymmetry using wearable sensors, Sensors 18 (2) (2018a) 676.
[20] A.R. Anwary, H. Yu, M. Vassallo, Wearable Sensor Based Gait Asymmetry Visualization Tool, in: Twenty-Fourth Americas Conference on Information Systems, New Orleans, 2018.
[21] A.R. Anwary, H. Yu, M. Vassallo, Optimal foot location for placing wearable IMU sensors and automatic feature extraction for gait analysis, IEEE Sens. J. 18 (6) (2018c) 2555–2567.
[22] T. Steinmetzer, et al., Clustering of Human Gait with Parkinson's Disease by Using Dynamic Time Warping, in: 2018 IEEE International Work Conference on Bioinspired Intelligence (IWOBI), IEEE, 2018.
[23] X. Jiang, et al., Exploration of Gait Parameters Affecting the Accuracy of Force Myography-based Gait Phase Detection, in: 2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob), IEEE, 2018.
[24] I. Loiret, C. Villa, B. Dauriac, X. Bonnet, N. Martinet, J. Paysant, H. Pillet, Are wearable insoles a validated tool for quantifying transfemoral amputee gait asymmetry? Prosthet. Orthot. Int. 43 (5) (2019) 492–499.
[25] C.C. Lin, R.C. Wagenaar, The impact of walking speed on interlimb coordination in individuals with parkinson's disease, J. Phys. Ther. Sci. 30 (5) (2018) 658–662.
[26] C. Miller-Patterson, R. Buesa, N. McLaughlin, R. Jones, U. Akbar, J.H. Friedman, Motor asymmetry over time in parkinson's disease, J. Neurol. Sci. 393 (2018) 14–17.

[27] S. Viteckova, P. Kutilek, J. Lenartova, J. Kopecka, D. Mullerova, R. Krupicka, Evaluation of movement of patients with parkinson's disease using accelerometers and method based on eigenvectors, in: In 2016 17th International Conference on Mechatronics-Mechatronika (ME), IEEE, 2016, pp. 1–5. December
[28] S. Viteckova, P. Kutilek, Z. Svoboda, R. Krupicka, J. Kauler, Z. Szabo, Gait symmetry measures: a review of current and prospective methods, Biomed. Signal Process. Control 42 (2018) 89–100.
[29] BNO055, Intelligent 9-axis absolute orientation sensor, in: BST-BNO055-DS000-14, Rev. 0273414209, Bosch Sensortec, 2016. June
[30] FSR 400, Series Data Sheet, in: Interlink Electronics, PDS-10004-C,
[31] P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, et al. Scipy: Open source scientific tools for python, 2019, Available online: http://www.scipy.org/ (accessed on 2 December 2019).
[32] N. Srivastava, et al., Dropout: a simple way to prevent neural networks from overfitting, The Journal of Machine Learning Research 15.1 (2014) 1929–1958.
[33] J.M. Hausdorff, et al., Gait variability and basal ganglia disorders: stride–to-stride variations of gait cycle timing in parkinson's disease and huntington's disease, Movement disorders 13.3 (1998) 428–437.
[34] E. Keogh, C.A. Ratanamahatana, Exact indexing of dynamic time warping, Knowl Inf Syst 7.3 (2005) 358–386.
[35] T. Steinmetzer, I. Bönninger, M. Reckhardt, F. Reinhardt, D. Erk, C.M. Travieso, Comparison of Algorithms and Classifiers for Stride Detection Using Wearables, in: Neural Computing and Applications, 2019, pp. 1–12.

Tobias Steinmetzer Graduation 2013 - Bachelor of Science in Computer Science at Brandenburg University of Technology Cottbus - Senftenberg. Graduation 2016 - Master of Science in Computer Science at Brandenburg University of Technology Cottbus - Senftenberg. Since 11/2018 - PhD Student at Universidad de las Palmas de Gran Canaria