Universiteit Twente

Bachelor's Thesis

Redesigning a Gait Detection Algorithm Based on Linear Accelerations to Include Gyroscope Data

Sietse de Groot

Graduation committee members:
M.K. MacLean, PhD (Rai)
F.J. Wouda, PhD (Frank)
Prof.Dr.Ir. M. Sartori (Massimo)

September 6, 2023

Report number: BE-936

Faculty and department: Engineering Technology – Biomechanical Engineering

Abstract

The algorithm presented in this assignment focuses on identifying walking bouts, which allows for the analysis of gait characteristics by other algorithms. A common technique used for data collection for gait detection, is to extract walking data from an inertial measurement unit (IMU), worn by a patient in an uncontrolled environment. An algorithm was previously created to differentiate between walking and non-walking data, using the linear accelerations of an IMU. The aim of this assignment was to improve the accuracy and reduce uncertainty by integrating gyroscope data into the gait detection algorithm. Multiple approaches were explored to include gyroscope data, including a thresholding algorithm in the time domain, a thresholding algorithm in the frequency domain, a machine learning algorithm and a combination of machine learning and a modified version of a part the original gait detection algorithm. A second goal of this assignment was to change the threshold used by the original algorithm to fit a different population, by personalizing the threshold. A personalized threshold, designed by using experimental data of walking and non-walking activities, replaced the generic threshold used by the original algorithm. Experiments were then performed to validate and compare the developed algorithms. Participants were recorded by video and IMU during 5 minutes of free movement. The video data were labeled as ”walking” and ”non-walking” and used for ground-truth validation. Out of all tested algorithms, the combination algorithm showed the highest accuracy, precision and F1-score (accuracy = 0.94, precision = 1.00, recall = 0.92, F1-score = 0.96). The combination algorithm showed a substantial improvement over the original script (accuracy = 0.90, precision = 0.90, recall = 0.95, F1-score = 0.92) except for the recall.

Index

1 Introduction

In 1990, 2.5 million people had Parkinson's disease globally. This number has increased to 6.1 million in 2016 [1]. As there is still no cure to the disease, early diagnosis is important. To diagnose a patient with a neurological disease like Parkinson's disease, their gait can be analyzed, as these diseases influence a person's gait. [2, 3]. Not only can these characteristics be used to diagnose patients, they can also be used to keep track of the progression of the diseases [4].

1.1 Gait Analysis Techniques

Different technological devices can be used to analyze a persons gait. These devices can be split into two categories: wearable sensors and non-wearable sensors. Non-wearable sensors, like floor sensors and image processing techniques, require a controlled environment like a laboratory where the sensors are stationed. Unlike non-wearable sensors, wearable sensors can be used outside of controlled environments.

For this assignment, a wearable sensor called an inertial measurement unit (IMU) is used. IMUs can measure the acceleration and orientation of a body by using an accelerometer and a gyroscope. This data can, for example, be used to distinguish walking from other activities.

When a patient is to be evaluated, they are asked to wear an IMU on their lower back for an extended amount of time during their daily lives. The reason they are not monitored in a controlled environment like a laboratory, is that monitoring in a patients daily life is more sensitive to impairments from neurological diseases [5].

There is however a problem with this method of data collection. It is unknown when a patient was exactly walking, since this would require the patient to keep track of all their activities during the time the IMU is worn. To still be able to separate the walking data from the non-walking data, an algorithm was created by Rai MacLean in MATLAB. This algorithm detects when someone was walking during the data collection, based on the accelerometer data alone. The algorithm also rejects shorter bouts of walking since these do not contain as much data for diagnosis as longer bouts. A more detailed explanation of this algorithm is given in the next section.

1.2 The Original Gait Detection Algorithm

The original gait detection algorithm only takes the linear accelerations of the IMU into consideration. The algorithm can be split into three parts. The first part of the algorithm mainly relies on a threshold. The signal of the three linear accelerations is put through a filter. This filter is a low pass, infinite impulse response with a half power frequency of 0.25hz. This low pass filtered signal gets subtracted from the original signal. This gets rid of low frequency influences on the signal like gravity and drift. As a result, the signal gets centered around zero. After this is subtracted, the magnitude of the three linear accelerations is calculated. Then the algorithm creates a logical vector where 1 corresponds to an IMU data point where the magnitude is higher than the Acceleration Magnitude (AM) threshold, and 0 otherwise. The AM threshold is determined by either the median of the magnitude multiplied by ten, or by a predetermined threshold value of 0.2, depending on which one is lower. The problem with the resulting vector, is that it can switch between walking and non-walking very fast. This means that it is possible, according to the algorithm, that someone walks for a single data point while standing still or the other way around.

To counter the problem of rapid transitions between binary states, the second part of the gait detection algorithm was created. In the second part, the algorithm determines the size of the gaps between walking. This is achieved by calculating when all gaps start and end. Then the *smoothdata* function in MATLAB is used to find the general trend in the vector. The smoothdata function is in this case a Gaussian moving average filter. That means that instead of weighting all points in the moving average window equally, it uses a Gaussian distribution to weight all points. After smoothing the vector, the amount of times the smoothed signal reaches under the Smooth Vector (SV) threshold is summed for each gap. This gap length is then checked to see if it is long enough to be considered a break, by comparing it to the Length of Gap (LoG) threshold. The start and end of the gaps that are long enough, are used to calculate the bout lengths. These bout lengths are then used to reject the short bouts and to make a logical vector again.

The third part of the algorithm filters the walking down even further. It calculates how long each bout is, and if it is shorter than two seconds, it gets rejected. After the short bouts are rejected, the vertical axis of the IMU acceleration is checked. The mean value and standard deviation during walking are calculated. Any walking data points that are outside of this range of the mean $\pm$ the standard deviation are rejected. Since it is possible that new short bouts are created, the short bouts are calculated and rejected again. This creates the logical vector which is returned by the gait detection algorithm. A simplified version of the code is shown in Algorithm 1.

This original algorithm can be used to detect when a patient was walking when they wore the IMU. This information can then be used to analyse gait characteristics. It can also be used to determine the patients activity level. This includes the amount of activity, length of walking bouts and at what time patients were active.

Algorithm 1 Pseudocode: original gait detection algorithm

```
function walkingStatus = originalGaitDetectionAlg(acc, Fs)

% part one
acc = $acc - Lowpassfilter(acc, Fs)$;
mag = $sqrt(acc.X^2 + acc.Y^2 + acc.Z^2)$;

accMagThreshold = $min([median(mag) \times 10, 0.2])$;
logicalVector = $mag > AMThreshold$;


% part two
[startGaps, endGaps] = $gapDetection(logicalVector)$;
generalTrend = $smoothdata(logicalVector,' Gaussian')$;

for i = 1 : $length(startGaps)$ do
    gapLength = $sum(generalTrend(startGaps(i) : endGaps(i)) < SVThreshold)$;
end for

startGaps = $StartGaps(gapLength > LoGThreshold)$;
endGaps = $endGaps(gapLength > LoGThreshold)$;

boutLengths = $determineBoutLengths(startGaps, endGaps)$;
walkingStatus = $rejectShortBouts(walkingStatus, boutLenghts)$;


% part three
walkingStatus = $verticalAccFiltering(walkingStatus, acc)$;

[startGaps, endGaps] = $gapDetection(walkingStatus)$;
boutLengths = $determineBoutLengths(startGaps, endGaps)$;
walkingStatus = $rejectShortBouts(walkingStatus, boutLenghts)$;

end
```

The original algorithm was created to accurately detect walking bouts, while still being computationally efficient. By minimizing the use of for loops and maximizing the use of built-in functionality for vectors, the algorithm manages to process large amounts of data in relatively short time. It also offers a lot of options when calling the function. It is possible to input a different filter and manually change all threshold values. However, the algorithm also has some downsides. For example, it only takes into consideration the linear acceleration of the IMU while most modern IMU contain both accelerometers and gyroscopes. It is also important that the input parameter of the acceleration is formatted correctly, so the vertical axis can be distinguished from the other axis, for filtering purposes. This means that the IMU has to be worn in the correct orientation. Both these problems can be solved by incorporating the gyroscope data into the algorithm. Another problem with the original algorithm is that the AM threshold is based on either a static value of 0.2 or the median of the magnitude signal. This is a simple method for determining personalized thresholds until it exceeds the static value of 0.2. A more advanced method of threshold determination could be useful for creating a personalized threshold value for everyone without limiting the threshold.

1.3 The Aim of This Bachelor Assignment

From the downsides of the affirmations original gait detection algorithm in section 1.2, the following research questions were formulated:

- How can the threshold value Acceleration Magnitude threshold of the original algorithm be personalized without limitations?
- How can gyroscope data be used in the gait detection algorithm of Rai MacLean to reduce uncertainty and possibly improve gait detection?

The original algorithm uses multiple thresholds to determine when an activity is labeled as walking. The AM threshold was determined through visual inspection of acceleration data. Most of this data was collected from elderly people, many of whom had Parkinson's disease. In this report, the participants will be able-bodied people. Since this population is different from the population the AM threshold was determined for, the value will be reconsidered for this assignment. To create the best possible threshold for each participant, the values for the threshold will be personalized. These thresholds will be determined by an algorithm which uses data to personalize the value of the threshold.

After this personalized threshold determination algorithm is created, multiple different gait detection algorithms which include both accelerometer and gyroscope data should be created. These algorithms will then be validated and compared to each other to determine which one is best at detecting gait. To be able to validate and compare the gait detection algorithms, labeled data should be obtained. This data should be representative of walking in the real world, like the walking data obtained after a week of data collection.

2 Method

To be able to create gait detection algorithms with personalised thresholds, data is needed for the personalization and validation of the algorithms. After data was collected, algorithms were created and validated.

2.1 Experiments

Since the experiments have two goals, two different types of measurements were performed. To create a personalized threshold, short experiments for walking and standing still were done. These short experiments will serve as a baseline for determining threshold values. Longer and more realistic measurement were performed for the second goal of the experiments. These longer measurements can be used for validating and comparing the algorithms. The IMU used during these experiments is the Xsens DOT.

2.1.1 Experimental Protocol

Five baseline experiments were performed. For non-walking activities, standing still and sitting for one minute each was chosen. The participants were instructed to sit and stand as they felt was most natural. For the walking activities, three different velocities were chosen: fast, normal, and slow. The participant was asked to walk in a straight line in a corridor for approximately 20 meters, similar to other gait detection studies [6, 7, 8]. To ensure the participants were consistent in their walking velocity, they were instructed with a specific scenario in their daily lives in which they would be expected to walk with that velocity. These scenarios are shown in Table 1. All of these baseline experiments were randomized for each participant to ensure that any influence of order of the experiments was minimized.

To make sure that the recorded data only captured the intended measurements, the duration of the measurements was longer than the collection period. This means for the sitting and standing measurements that the participant started doing the activity before the data collection was started, and only stopped doing the activity after a minute of data collection had been completed. For the walking activities this meant that the participant started walking for approximately two meters before the data collection started and stopped walking two meters after the data collection had stopped. So in total the participant walked approximately 24 meters. This ensured that the collected data only contained the desired IMU data.

Table 1: Instructions for different walking velocities

Walking speed	**Scenario**
Slow	You are walking outside in a park with a friend. You are talking to each other and taking in the scenary while not standing still.
Normal	You are doing groceries and are not in a hurry.
Fast	You are late for a lecture or meeting and you are not running.

Two longer measurements where the participant was free to move however they wanted were performed to test the algorithm in a realistic scenario. During both measurements, the participant was free to stand, sit, and walk as much as they wanted within a five minute time frame. The first measurement was done inside. To challenge the algorithms, participants were also allowed to take the stairs if they wanted to and were also instructed to pick a small item up from the ground at least once. The second longer measurement was performed outside. The reason for doing the same experiment outside was to test how well walking would be identified when the surface is not flat. There were no additional instructions for the measurement outside, as the vertical axis would already be challenged by the uneven terrain. The participant was filmed during these longer measurements so that the IMU data could be labeled afterwards. This measurement was inspired by an article which also wanted to obtain realistic walking data. This was achieved by filming participants at home with a handheld camera while doing daily tasks[9].

During all parts of the experiment one IMU was secured to the participant on the fifth lumbar vertebrae, using medical tape. The participant was informed of all activities and was given specific instructions to ensure consistency between participants. During the two free walking experiments, the participant was filmed. This video was synchronised with the IMU data and was used to label the data. The labeling and synchronization process is further explained in section 2.1.2. An overview of all activities done by the participants is shown in Table 2.

Table 2: Activity list

Activity	**Duration**
Sitting	1 minutes
Standing	1 minutes
Walking straight line inside normal	20 meter
Walking straight line inside slow	20 meter
Walking straight line inside fast	20 meter
Free walking inside	5 minutes
Free walking outside	5 minutes

2.1.2 labeling and Synchronization

The IMU data of both free movement measurements were filmed and labeled afterwards. The experiment was filmed by a Samsung Galaxy A52s filming at 30 fps, while the IMU data was being collected at 60 Hz. This means that one frame of the video was linked to two data points of the IMU. At the beginning of the experiment, the IMU sensor was being filmed and tapped three times. This resulted in a spike in the accelerometer data of the IMU, which was then lined up with the first frame of the video at which visual contact was made with the IMU. By synchronizing these points, an IMU data point can be calculated which coincides with the start of the video. This is calculated for all three points. The average starting IMU data point of the three taps is determined and used to synchronize the IMU data with the video frames.

After the IMU data and video were synchronized, the data was labeled. The activities as well as the transitions between activities were labeled according to Table 3. Since the labels were chosen for each frame, the description is as specific as possible.

Table 3: Label descriptions

Label	**Description**
Calibrating	From the start of the measurement, until the participant lowered their shirt.
Walking	From the moment one stride is completed and the second foot begins to move, until the participant has one foot on the place where they are going to stand still or until the participant places their foot at an angle before sitting down.
Standing	From the moment the participant is not moving their feet and is standing upright with straight knees, until one of their feet move again or their knees bend.
Sitting	From the moment someone is not moving their feet and is sitting down, until one of their feet move again with the intention of standing up or they stop sitting down without moving their feet.
Stairs	From the moment the first foot touches the first step, until the last feet is off the last step. Can be either up or down the stairs.
Picking up item	When the participant starts bending forward to pick up the item until the participant is standing upright again.
Transition	Transition from sitting to standing, standing to sitting, or standing, sitting, calibrating to walking or the other way around.

2.2 Algorithms

There are two types of algorithms created for this assignment. The first one is to determine personalized thresholds for the original gait detection algorithm, and the second one is a new gait detection algorithm which includes both gyroscope and accelerometer data.

2.2.1 Threshold Determination

The Personalized Threshold Determination (PTD) algorithm determines a threshold value for the AM threshold based on the baseline experiments described earlier. These experiments consist of standing or sitting for the non-walking activities, and walking at a near constant speed for the walking activities. The main difference between walking and non-walking activities, is the fact that data from non-walking activities has low magnitudes and small fluctuations, while data from walking activities has high magnitudes with bigger fluctuations. This was the core idea behind the proposed PTD algorithm.

Similarly to the original gait detection algorithm, the PTD algorithm subtracts a low pass filtered copy of the signal from the signal and then determines the magnitude. This magnitude signal is useful, because the main difference between the two, as described before, is the difference in magnitude and fluctuation in magnitude. Then, to determine the threshold, a histogram is created which resembles the distribution of acceleration values over time.
To create the histogram, the magnitude signal is split into 50 bins. This number of bins was determined by visual comparison of histograms with a different number of bins. With 50 bins, there

was a balance of not losing details in the magnitude distribution while avoiding big fluctuations between neighbouring bins. When visual comparison was no longer needed, the MATLAB function *hiscount* was used to calculate the histogram without plotting it. After the points of the histogram are calculated, the data is normalized by dividing the number of data points in a single bin by the total number of data points. As the walking and non-walking data have different sizes, normalizing becomes necessary to be able to directly compare them. Figure 1 shows a histogram of one of the walking data sets.

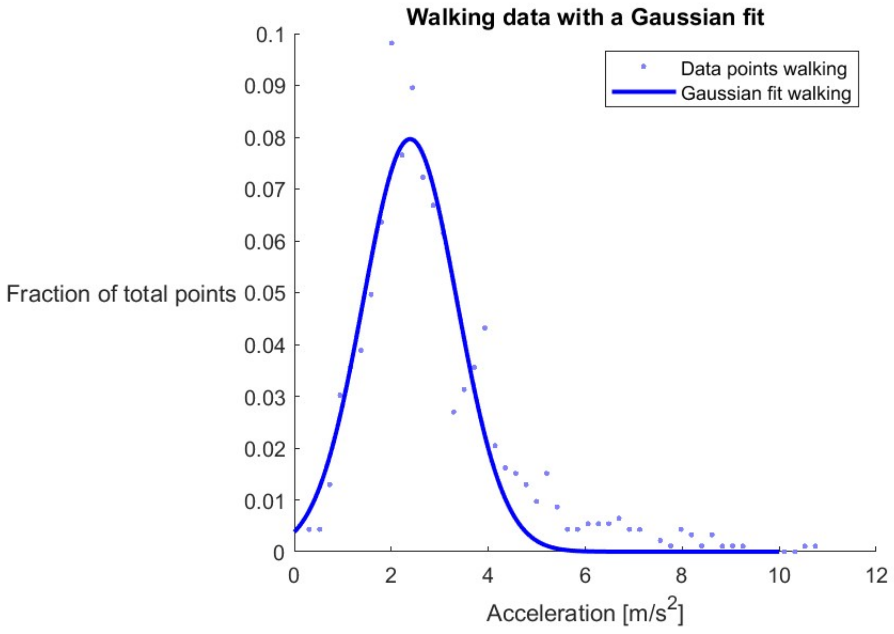

Figure 1: Gaussian fit on walking data points

Visually, the histogram seemed to resemble a Gaussian distribution. To verify if a Gaussian fit was the best option, multiple commonly used fits were tried and their R-squared error was calculated. All fits with an R-squared error higher than 0.85 are shown in Table 4. The Gaussian fits had the highest R-squared error value. Since the "gauss2" fit type only increased the R-squared by 0.04, it was determined that the single Gaussian fit was the best choice, to prevent overfitting. A Gaussian fit on the data of one of the normal walking experiments is shown in Figure 1. Instead of using bins like an actual histogram, dots were used to reduce visual cluttering. A simplified version of the code used to create this fit is shown in Algorithm 2.

Table 4: R-squared error of commonly used fits where R-squared was higher than 0.8

fit type	**R-squared**
"poly6"	0.81
"poly7"	0.86
"poly8"	0.89
"gauss1"	0.95
"gauss2"	0.99

Algorithm 2 Pseudocode: create Gaussian fit

```
function gaussFit = fitGaussianFunction(acc, Fs)

    acc = $acc - Lowpassfilter(acc, Fs)$;
    Mag = $sqrt(acc.X^2 + acc.Y^2 + acc.Z^2)$;

    [binCounts, edges] = $hiscount(Mag, "NumBins", 50)$;
    binFrac = $binCounts/sum(binCounts)$
    binAcc = $edges(1 : end - 1) + (edges(2) - edges(1))/2$;

    gaussFit = $fit(binAcc, binFrac, "Gaussian")$;

end
```

Now that the best fit has been determined, the data from the two baseline conditions can be fitted. This creates one Gaussian function for the standing data, and one for the walking data. An example of these two fits for one participant is shown in Figure 2. There are two intersections between the functions. One intersection is in the middle of the two peaks of the functions while the other is either before the first peak, or after the last. From these two, the one between the peaks of the functions is used as a threshold. A simplified version of this algorithm is shown in Algorithm 3.

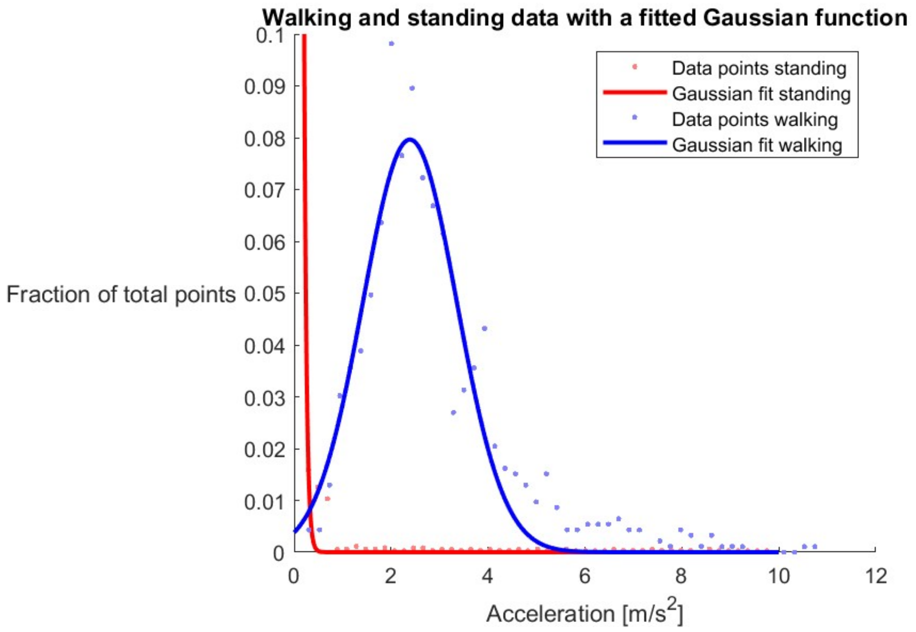

Figure 2: Two Gaussian fits plotted on top of the data points they were fitted on

The training data used for determining the threshold can be experimented with. For non-walking data, the sitting and standing data can be used. For walking data, the walking slow, normal and fast data can be used. Six different training sets are created by matching each non-walking data set to each of the walking data sets, which will be compared to each other to find out which yields the highest accuracy.

Algorithm 3 Pseudocode: personalized threshold determination algorithm

```
function threshold = determineThreshold(walkAcc, nonWalkAcc, Fs)

walkingFit = $fitGaussianFunction(walkAcc, Fs)$;
nonWalkingFit = $fitGaussianFunction(nonWalkAcc, Fs)$;

intersections = $solve(walkingFit == nonWalkingFit)$;
threshold = $intersections(nonWalkingFit > intersections > walkingFit)$;

end
```

2.2.2 Gait Detection

During the development of the algorithms, the advantages of the original algorithm were kept in mind. This meant that the algorithm had to remain simple and computationally efficient. For the first three methods of sensor fusion, the gyroscope and accelerometer are meant as an alternative to part 1 of the original algorithm. The output of these algorithms should be a rough estimation of when the participant was in either a walking state or a non-walking state without considering the time spent within that state. These approaches for sensor fusion can be categorized into two categories: thresholding and machine learning. Thresholding was again divided into two: the time domain and the frequency domain. These approaches are then validated with the labeled data and then compared to each other to determine which one has the highest accuracy. Finally, an algorithm with a combination of both machine learning and a modified version of part 2 of the original algorithm is created.

The thresholding approaches use both gyroscope and accelerometer data of the IMU. Both vectors are compared to different thresholds and combined afterwards. This creates more certainty for each data point which gets labeled as walking.

The thresholding time domain (TTD) algorithm approach uses both the magnitude of the linear accelerations and the angular velocity. In a similar fashion to the original algorithm, the low pass filtered signal is subtracted from the original signal and is then used to calculate the magnitude. Then the signal is compared to an AM threshold to create a logical vector. Then the magnitude of the angular velocity is calculated separately and is then compared to an Angular Velocity Magnitude (AVM) threshold. Both thresholds are determined by the PTD algorithm described in section 2.2.1. Both these logical vectors are combined by a logical AND operator to create the final output of this algorithm. The low pass filter technique is, however, not used for the AVM threshold for determining the threshold value. For the training data of the PTD algorithm, the combination of standing and walking normal were chosen. The pseudocode for this algorithm is shown in Algorithm 4. As stated before, this is an alternative to part 1 of the original gait detection algorithm, so it will be combined with part 2 and 3 to create a complete gait detection algorithm.

Algorithm 4 Pseudocode: thresholding time domain algorithm

```
function walkingStatus = timeDomainGaitDetectionAlg(acc, gyr, Fs)

    acc = $acc - Lowpassfilter(acc, Fs)$;
    magAcc = $sqrt(acc.X^2 + acc.Y^2 + acc.Z^2)$;
    magGyr = $sqrt(gyr.X^2 + gyr.Y^2 + gyr.Z^2)$;

    walkingStatus = $magAcc > AMThreshold$ and $magGyr > AVMThreshold$;

end
```

The thresholding frequency domain (TFD) algorithm approach uses the same magnitude of the accelerations and angular velocities as described before in the TTD algorithm. However, in this algorithm, the angular velocity magnitude is then separated into vectors with a length equal to the sampling rate of the IMU. These vectors are then converted to the frequency domain using a fast fourier transform, where the highest value of the magnitudes of all frequency components is returned. This value should be higher for vectors where the participant was walking, since this would create a periodic signal. This maximum value is determined for all vectors. These values are then compared to an Amplitude Signal Threshold to determine when someone is walking. This threshold is again determined with the same PTD algorithm as described in section 2.2.1. In addition to the angular velocity, the magnitude of the linear acceleration is also compared to an AM threshold. Both logical vectors then get combined by a logical AND operator. The pseudocode is shown in Algorithm 5. This approach is also meant as an alternative for part 1 of the original gait detection algorithm, and will thus be combined with parts 2 and 3 to complete the algorithm.

Algorithm 5 Pseudocode: frequency domain algorithm

```
function walkingStatus = freqDomainGaitDetectionAlg(acc, gyr, Fs)

acc = $acc - Lowpassfilter(acc, Fs)$;
magAcc = $sqrt(acc.X^2 + acc.Y^2 + acc.Z^2)$;

magGyr = $sqrt(gyr.X^2 + gyr.Y^2 + gyr.Z^2)$;

for i = 1 : $length(magGyr) - Fs$ do
    amplitudeSignal(i) = $max(fourier(magGyr(i : i + Fs)))$;
end for

walkingStatus = $magAcc(Fs/2 : end - Fs/2) > AMThreshold$ and $amplitudeSignal > SVThreshold$;

end
```

The machine learning (ML) algorithm approach uses a classification model to label data points as walking or non-walking. The MATLAB classification learner was used to find the best type of classification model. According to the classification learner, the medium k-nearest neighbour(KNN) model was the best model, which is why this is the classification model used in this algorithm. This model classifies data points by calculating which data points in its training set are nearest. The baseline experiments were used as the training set. This way, the algorithm creates a personalized model based on simple measurements. To train the model, the *fitcknn* function was used. The number of neighbours chosen in this case was ten, because that was recommended by the classification learner. A simplified version of this algorithm is shown in Algorithm 6. As the magnitude of the acceleration is not determined in all versions of this algorithm, it will only be combined with part 2 of the original algorithm. Since the designing of the low pass filter for the determination of the magnitude of the acceleration is relatively inefficient, it was decided to leave part 3 out of the algorithm.

Algorithm 6 Pseudocode: machine learning algorithm

```
function walkingStatus = MLGaitDetectionAlg(trainDataIMU, testDataIMU)

model = $fitcknn(traindataIMU, 10)$;
walkingStatus = $predict(model, testDataIMU)$;

end
```

To train the model, a table with sample data is given to the *fitcknn* function. There are three different options for sets of features which can be used for training the classification model. The first possibility involves using all IMU data as sample data. The second possibility uses both the magnitude of the linear acceleration, as well as the magnitude of the angular velocity, similar to the other algorithms. Lastly, the third option is using a MATLAB function called *fscmrmr* to determine the best features for the classification model. This function returns the index as well as their respective scores. All parameters with a score higher than 0.1 are then used to train the model.

Finally, a combination of algorithms is created, by combining machine learning with a modified version of part 2 of the original algorithm. First, the classification model creates the initial logical vector. This logical vector is then smoothed, similar to part 2 of the original algorithm. In contrast to the original algorithm, this algorithm directly compares the smoothed data to a SV threshold, which creates a new logical vector. This direct comparison to a threshold was done to create a more readable and potentially more computationally efficient algorithm than the original algorithm. The best value of the SV threshold will be experimentally determined. After the new logical array is determined, the bout length of each bout is determined in order to reject the short bouts, similar to the original gait detection algorithm. The pseudocode for this algorithm is shown in Algorithm 7. This algorithm will not be combined with any part of the original algorithm, as it already contains a modified version of part 2. Similarly to the ML algorithm, it will not be vertically filtered by part 3, as this would require the determination of the magnitude of the acceleration magnitude, which is computationally inefficient.

Algorithm 7 Pseudocode: combination algorithm

```
function walkingStatus = MLGaitDetectionAlg(trainDataIMU, testDataIMU)

model = $fitcknn(dataIMU, "Activities")$;
walkingPred = $predict(model, testDataIMU)$;

walkingStatus = $smoothdata(walkingPred) > SVThreshold$;
walkingStatus = $rejectShortBouts(walkingStatus)$;

[startGaps, endGaps] = $gapDetection(walkingStatus)$;
boutLengths = $determineBoutLengths(startGaps, endGaps)$;
walkingStatus = $rejectShortBouts(walkingStatus, boutLenghts)$;

end
```

2.3 Validation of Algorithms

The validation was done in two parts. First the PTD algorithm was validated and afterwards the gait detection algorithms were validated and compared. To validate if an algorithm was accurate, the function was tested using data from the data set which was filmed and labeled. The output of the function was then compared to the known label. This was then used to calculate the number of true positives, true negatives, false positives and false negatives. These values were then used to determine the accuracy, precision, recall and F1-score. The accuracy is useful as an overall measurement of the quality of the algorithms. The precision and recall can measure how much of the true positive data points are actually correct, and how many points were missed by the algorithm. The F1-score is a combination of the precision and recall measurement.

The labels of the data points were changed to "walking" and "non-walking", so it could be easily compared to the binary output of the algorithms. All activities labeled as "Walking" and "Stairs" were changed to "walking". All other labels were changed to "non-walking".

The first part of the validation was aimed at validating what training set would be best for the PTD algorithm. This was done by combining the original gait detection algorithm with the PTD algorithm to determine the AM threshold. This algorithm was tested six times, each time with a different training data set, after which the accuracy, precision, recall and F1-score were determined for validation.

The second part of the validation was aimed at validating and comparing the different gait detection algorithms. The algorithms that were compared are: the original gait detection algorithm, the original algorithm in combination with the PTD algorithm, the TTD algorithm, the TTD algorithm and the ML algorithms with three different sample data sets. All algorithms which have the PTD algorithm incorporated, used the same data set of standing as the non-walking data and walking normal as the walking data.

The TTD algorithm and the TFD algorithm were used as replacement of part 1 in the original algorithm, as they were meant to replace that part of the original algorithm. The ML algorithms were only combined with part 2 of the original algorithm, since part 3 required the magnitude of the acceleration which was not available in all ML algorithms
.
The algorithm which combines the ML algorithm with thresholding inspired by the original algorithm, was ran multiple times with different threshold values. The threshold values that were considered are: 0.2, 0.4, 0.6 and 0.8.

The mean accuracy of all algorithms will be shown in a bar graph for comparison. For the algorithms where multiple input parameters were tried, the best one was chosen. The difference between the mean accuracies of the outside and inside measurements are also shown. Finally, to judge the computational efficiency, all algorithms were ran with all available data from the free movement experiments. This will show how long it takes for the algorithms to process roughly 30 minutes of data.

3 Results

The results are split into two parts. The first part focuses on the PDT algorithm, while the second part focuses on the gait detection algorithms. The participants that took part in the experiments for this assignment had an average age of 21 years old (±0), and an average height of 1.84 meters (±0.01).

3.1 Threshold Determination

Table 5 shows the evaluation metrics for the PDT algorithm using different training data sets. All versions with sitting as a data set show lower mean accuracy than their respective counterparts of the standing training data set, except for the combination with walking fast. Both combinations with walking fast score the same over all metrics. The training data sets with the highest mean accuracy, precision, recall and F1-score are the "Sitting + walking fast", "Standing + walking normal" and "Standing + walking fast" training data sets.

Table 5: Mean accuracy, precision, recall and F1-score of the original gait detection algorithms using the PDT algorithm to determine the Acceleration Magnitude Threshold with different training data sets.

Training data	**Accuracy**	**Precision**	**Recall**	**F1-score**
Sitting + walking slow	0.83 ± 0.03	0.96 ± 0.02	0.83 ± 0.04	0.89 ± 0.02
Sitting + walking normal	0.85 ± 0.04	0.96 ± 0.02	0.85 ± 0.04	0.90 ± 0.02
Sitting + walking fast	0.86 ± 0.03	0.96 ± 0.02	0.85 ± 0.04	0.90 ± 0.02
Standing + walking slow	0.84 ± 0.04	0.96 ± 0.02	0.84 ± 0.05	0.89 ± 0.03
Standing + walking normal	0.86 ± 0.04	0.96 ± 0.02	0.85 ± 0.05	0.90 ± 0.03
Standing + walking fast	0.86 ± 0.03	0.96 ± 0.02	0.85 ± 0.04	0.90 ± 0.02

3.2 Gait Detection

The gait detection algorithms which serve as an alternative to part 1 of the original algorithm, are shown in Table 6. The unaltered original algorithm is also shown for comparison. It should be noted that the ML algorithms do not have part 3 of the original algorithm incorporated as stated earlier, to keep the computational complexity low. The version of the ML algorithm is shown in brackets. The original algorithm shows the highest mean accuracy, recall and F1-score and the lowest precision. The highest mean precision is reached by two of the ML algorithms.

Table 6: Mean accuracy, precision, recall and F1-score of the gait detection algorithms with different approaches to part 1 of the original algorithm.

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-score**
Original	0.90 ± 0.08	0.90 ± 0.13	0.95 ± 0.02	0.92 ± 0.07
Original + PTD	0.86 ± 0.04	0.96 ± 0.02	0.85 ± 0.05	0.90 ± 0.03
TTD + PTD	0.86 ± 0.04	0.96 ± 0.02	0.86 ± 0.05	0.91 ± 0.03
TFD + PTD	0.87 ± 0.04	0.95 ± 0.03	0.88 ± 0.05	0.91 ± 0.02
ML(All)	0.81 ± 0.07	0.98 ± 0.02	0.79 ± 0.08	0.87 ± 0.05
ML(magGyr + magAcc)	0.87 ± 0.02	1.00 ± 0.00	0.84 ± 0.03	0.91 ± 0.02
ML(fscmrmr)	0.78 ± 0.09	1.00 ± 0.00	0.76 ± 0.10	0.86 ± 0.06

The results of the combination algorithm which combines one of the ML algorithms with a modified version of part 2 of the original gait detection algorithm is shown in Table 7. The version with the threshold value of 0.8 has the highest scores for all evaluation metrics. The mean accuracy, recall and F1-score all increase with higher threshold values.

Table 7: Mean accuracy, precision, recall and F1-score of the combination algorithm with different threshold values.

Threshold value	**Accuracy**	**Precision**	**Recall**	**F1-score**
0.2	0.87 ± 0.03	1.00 ± 0.00	0.84 ± 0.04	0.91 ± 0.02
0.4	0.90 ± 0.03	1.00 ± 0.00	0.87 ± 0.04	0.93 ± 0.02
0.6	0.92 ± 0.02	1.00 ± 0.00	0.89 ± 0.04	0.94 ± 0.02
0.8	0.94 ± 0.02	1.00 ± 0.00	0.92 ± 0.04	0.96 ± 0.02

The mean accuracies for all 3 participants of all gait detection algorithms are shown in a bar graph in Figure 3. The measurements inside and outside are shown separately together with the mean of all measurements combined. As for the algorithms where different parameters were tried, the version with the highest accuracy is shown. The combination algorithm has the highest mean overall accuracy and highest mean accuracy for the outside measurements. The highest mean accuracy for the inside measurements is reached by the original algorithm. The original, TTD + PTD and TFD + PTD algorithms show higher mean accuracies for the inside measurements, in comparison with the outside measurements. The ML and combination algorithm show higher mean accuracies for the outside measurements.

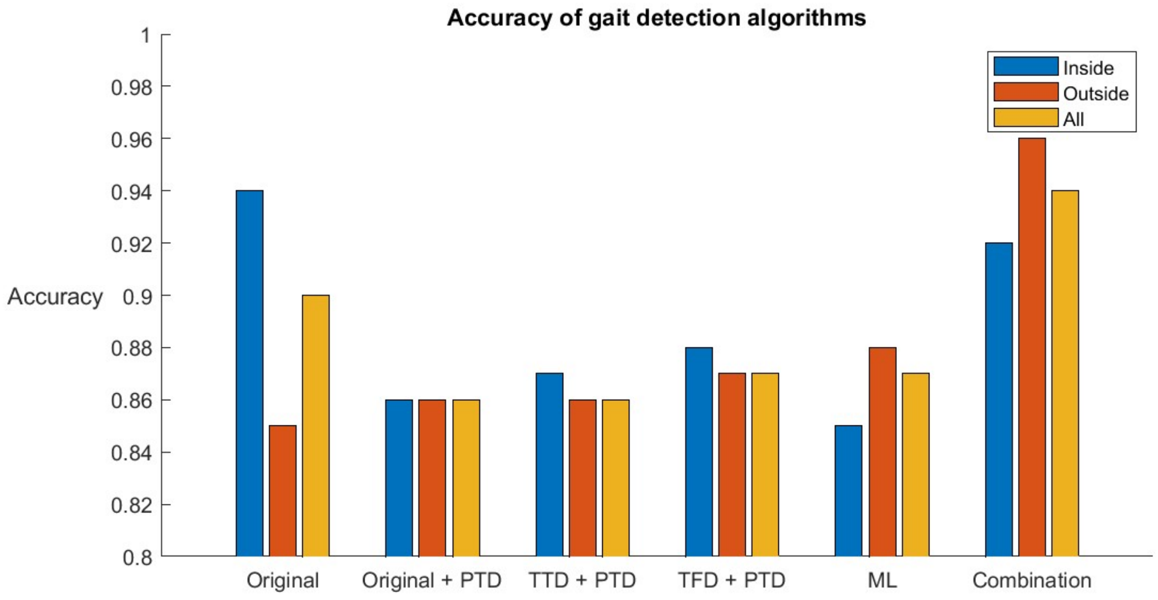

Figure 3: Accuracy of different gait detection algorithms. The mean accuracy for all 3 participants are shown for the inside and outside condition, together with the combined accuracy.

The run times of the gait detection algorithms are shown in Table 8. The original algorithm has the lowest run time, with 4.1 seconds. The highest run time was reached by the TFD + PDT algorithm, with 9.7 seconds.

Table 8: Run time of the gait detection algorithms.

Algorithm	**Run time (s)**
Original	4.1
Original + PDT	6.1
TTD + PDT	7.3
TFD + PDT	9.7
ML (All)	6.9
ML (magGyr + magAcc)	6.1
ML (fscmrmr)	7.1
Combination	5.7

4 Discussion

The discussion will be split into three parts. The results will be discussed in the first part, then recommendations for the algorithms will be discussed, and finally conclusions will be made.

4.1 Results

The best training data set according to Table 5, are the "Sitting + walking fast", "Standing + walking normal" and "Standing + walking fast". The mean evaluation metrics of all training sets are not substantially different from each other, as the difference between the highest accuracy and the lowest are within the standard deviation of each other. When comparing the mean evaluation metrics from the original algorithm using the original threshold determination method in Table 6 with the original algorithm using the PTD algorithm in Table 5, it can be determined that even the best version of using the PTD algorithm has worse mean accuracy, recall and F1-score than the original method. Only the mean precision is higher for the algorithm using the PTD.

The reason for the lower mean accuracy, recall and F1-score of the version of the algorithm using the PTD, might be that the baseline measurements were not representative of the non-walking activities. In particular, the transitions between activities will likely have exceeded the threshold value determined by the PTD algorithm, as transitions generally have more movement than standing or sitting, while also being labeled as non-walking.

From all gait detection algorithms shown in Table 6, the original algorithm had the highest mean accuracy, recall and F1-score. All other algorithms have higher mean precision, meaning that the original algorithm had relatively fewer false negatives, but lower precision as it had relatively more false positives. Out of all ML algorithms, the version with the magnitude of the angular velocity and linear acceleration had the highest mean evaluation metrics. Out of all algorithms with sensor fusion of the accelerometer and gyroscope in Table 6, the TFD + PTD algorithm, together with the ML (magGyr + magAcc) algorithm had the highest accuracy and F1-score. The difference between the two is that the ML algorithm had higher precision, while the TFD + PTD algorithm had higher recall.

The optimal threshold for the combination algorithm is 0.8 according to Table 7. All mean metrics increase with increasing threshold value, except for the mean precision. The version of the combination algorithm with 0.8 as a threshold has the highest mean accuracy, precision and F1-score of all algorithms presented in the results. Only the original algorithm has a higher mean recall.

Also, in Figure 3, it is clear that the combination algorithms has the highest mean accuracy of all algorithms. It also shows that the combination and ML algorithm have a higher mean accuracy outside rather than inside.The original, TTD + PTD and TFD + PTD algorithms show the opposite. They have higher mean accuracies inside than outside. This difference might be explained by the lack of vertical filtering in the ML and combination algorithm, as these are the only two algorithms without it.

As stated before, the combination algorithm has the highest mean accuracy of all algorithms in this report. Its accuracy score of 0.94 is comparable to that of another machine learning algorithm

using a single IMU sensor with an accuracy of 0.96 [10].
The run times of the algorithm in Table 8 show that the original algorithm was computationally most efficient. The worst performing algorithm was the TFD + PDT algorithm. This can be explained by the fact that it uses the PDT algorithm twice, loops over the angular velocity magnitude and uses the low pass filter of which the designing process is known to be computationally relatively inefficient. The combination algorithm which had a lot of focus on being computationally efficient is the second fastest with a run time of 5.7 seconds.

The original algorithm and both thresholding approaches used the threshold determination algorithm. So, while each algorithm had different approaches to creating a logical vector, they all relied on the same algorithm to create the thresholds they operated on. This does create consistency between the algorithms, but can also drag down the algorithms if it does not lead to the optimal threshold value.

As mentioned in the introduction, the orientation of the device is important for part 3 of the original algorithm. This means that all gait detection algorithms which have part 3 of the original algorithm incorporated in them, can be influenced by the patient if the IMU is not worn correctly. This does, however, not pose a problem for the algorithms which do not have part 3 incorporated. For those algorithms, it does not matter what orientation the IMU is worn in, as the result will stay the same.

The ML algorithm using the MATLAB function *fscmrmr* to choose features performed worse than choosing all features. An explanation could be that the threshold of the associated scores was too low, leading to overfitting. This could potentially be solved by choosing a higher threshold for the scores of the features, or by, for example, only using the three highest scoring features.

For the validation of the PTD algorithm, it was chosen to not combine all non-walking and all walking-data. This was chosen because the idea of using a single Gaussian fit would likely not fit the data well. Since each baseline measurement could be described as a single Gaussian fit, if two were added it would likely require a double Gaussian to fit the data correctly. This would increase the computational efficiency of the algorithm, which goes against the goal of the assignment

4.2 Recommendations

The reason that no statistical analysis was done, was because of the relatively small population size of three, the confidence level would be low. To be able to perform a statistical analysis of the proposed algorithms, the size of the experimental group should be increased.

The computational efficiency of the algorithms has only been tested once. To create a more accurate result, the algorithms should be tested multiple times and with different sizes of input data. This was not done in this assignment because of time constraints.

Taking the stairs was considered walking during the validation. It might be better to distinguish walking on stairs and normal walking, because taking the stairs may not be useful for diagnosis as it likely does not contain the same characteristics, since the activities are inherently different.

Part 3 of the current algorithm uses the vertical acceleration to filter out data which can not be walking. This part could be modified to also analyse the angle of the device during walking, and use a threshold to determine when the angle is too extreme to be considered walking.

4.3 Conclusion

An algorithm was created for determining a personalized threshold by fitting two Gaussian functions on the distribution of the magnitude of the linear accelerations of the IMU. The original algorithm using this method of threshold determination resulted in a substantially lower mean accuracy than using the original threshold determination method.

The best way to use gyroscope data to reduce uncertainty and improve the gait detection of the original gait detection algorithm, is by using the combination of the k-nearest neighbor classification model with a modified version of the signal smoothing and bout rejection of the original algorithm. This algorithm had the highest accuracy, precision, and F1-score (accuracy = 0.94, precision = 1.00, recall = 0.92, F1-score = 0.96). The combination algorithm was relatively computational efficient, as the run time of this algorithm was the second fastest of all discussed algorithms, only being beaten by the original gait detection algorithm. It also is not relevant for this algorithm how the IMU is oriented, while that does matter to the original algorithm. Taking all this into consideration, it can be said that the best way to integrate gyroscope data into the original gait detection algorithm, is by using the combination algorithm.

References

[1] E. R. Dorsey, A. Elbaz, E. Nichols, N. Abbasi, F. Abd-Allah, A. Abdelalim, J. C. Adsuar, M. G. Ansha, C. Brayne, J.-Y. J. Choi, and et al., "Global, regional, and national burden of parkinson's disease, 1990–2016: A systematic analysis for the global burden of disease study 2016," *The Lancet Neurology*, vol. 17, no. 11, p. 939–953, Nov 2018.

[2] E. Buckley, C. Mazzà, and A. McNeill, "A systematic review of the gait characteristics associated with cerebellar ataxia," *Gait Posture*, vol. 60, pp. 154–163, 2018.

[3] J. Jankovic, "Parkinson's disease: clinical features and diagnosis," vol. 79, no. 4, pp. 368–376, 2008.

[4] J. Wilson, L. Alcock, A. J. Yarnall, S. Lord, R. A. Lawson, R. Morris, J.-P. Taylor, D. J. Burn, L. Rochester, and B. Galna, "Gait progression over 6 years in parkinson's disease: Effects of age, medication, and pathology," *Frontiers in Aging Neuroscience*, vol. 12, 2020, cited by: 29; All Open Access, Gold Open Access, Green Open Access.

[5] V. Shah, J. McNames, M. Mancini, P. Carlson-Kuhta, R. Spain, J. Nutt, M. El-Gohary, C. Curtze, and F. Horak, "Laboratory versus daily life gait characteristics in patients with multiple sclerosis, parkinson's disease, and matched controls," *Journal of NeuroEngineering and Rehabilitation*, vol. 17, no. 1, 2020.

[6] E. P. Washabaugh, T. Kalyanaraman, P. G. Adamczyk, E. S. Claflin, and C. Krishnan, "Validity and repeatability of inertial measurement units for measuring gait parameters," *Gait Posture*, vol. 55, pp. 87–93, 2017.

[7] S. R. Hundza, W. R. Hook, C. R. Harris, S. V. Mahajan, P. A. Leslie, C. A. Spani, L. G. Spalteholz, B. J. Birch, D. T. Commandeur, and N. J. Livingston, "Accurate and reliable gait cycle detection in parkinson's disease," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 1, pp. 127–137, 2014.

[8] R. Romijnders, E. Warmerdam, C. Hansen, J. Welzel, G. Schmidt, and W. Maetzler, "Validation of imu-based gait event detection during curved walking and turning in older adults and parkinson's disease patients," *Journal of NeuroEngineering and Rehabilitation*, vol. 18, no. 1, 2021.

[9] M. H. Pham, M. Elshehabi, L. Haertner, S. Del Din, K. Srulijes, T. Heger, M. Synofzik, M. A. Hobert, G. S. Faber, C. Hansen, D. Salkovic, J. J. Ferreira, D. Berg, Sanchez-Ferro, J. H. van Dieën, C. Becker, L. Rochester, G. Schmidt, and W. Maetzler, "Validation of a step detection algorithm during straight walking and turning in patients with parkinson's disease and older adults using an inertial measurement unit at the lower back," *Frontiers in Neurology*, vol. 8, 2017.

[10] A. Atrsaei, F. Dadashi, B. Mariani, R. Gonzenbach, and K. Aminian, "Toward a remote assessment of walking bout and speed: Application in patients with multiple sclerosis," *IEEE Journal of Biomedical and Health Informatics*, vol. 25, no. 11, pp. 4217–4228, 2021.

5 Appendix

5.1 Appendix A: Acronyms

Table 9: List of acronyms and their meanings

Acronym	**full name**
IMU	Inertial measurement unit
AM	Acceleration magnitude
AVM	Angular velocity magnitude
SV	Smoothed vector
LoG	Length of gap
PTD	Personalized threshold determination
TTD	Thresholding time domain
TFD	Thresholding frequency domain
ML	Machine learning
KNN	K-nearest neighbour