Sport Coach

Online activity matching using wireless sensor network

Master Thesis

Author
Arie Horst

July 12, 2010

Supervisor:
Dr. Ir. Nirvana Meratnia

Graduate committee:
Dr. Ir. Nirvana Meratnia
Ir. Hans Scholten
Prof. Dr. Ing. Paul Havinga

Enschede
Faculty of Electrical Engineering, Mathematics and Computer Science
Pervasive Systems research group

Abstract

In this thesis we explore the use of Wireless Sensor Networks to perform online activity matching for sport coaching applications. The goal is to find one or more suitable algorithm to match the movement of a trainer and a trainer and to find spatial and temporal differences. Such a system can aid the trainer in group lessons where it is difficult for the trainer to keep track of all the trainees.

In this work we consider fitness like movements such as encountered in aerobic lessons. We limit ourselves to only one one sensor node on the trainer and one sensor node on the trainee, but this well extends to more sensors per trainer and trainee. It also scales well to more trainees per trainer. We use Sun SPOT as wireless sensor nodes, and extend the set of sensors to obtain more inertial measurement capabilities. The accelerometer and gyroscope sensor are used to capture the movements. The gravity vector is extracted and improved with a Kalman filter using the accelerometer and gyroscope data. An automatic segmentation technique is used that examines the movement data for rest and activity periods and changes in movement direction. The segmentation and the movement information is communicated with the node of the trainee where the movements are compared. We choose to use Dynamic Time Warping (DTW) to perform the spatial and temporal matching of the movement. Because DTW is computationally intensive, we developed an optimized technique which we call Fast Incremental Dynamic Time Warping (FIDTW). From the result of the FIDTW algorithm, feedback is then generated and provided to the trainee.

We test all the design choices extensively using experiments, and perform a total system test using different test methods to validate the taken approach. The single person test methods show that the system can reliably discriminate spatial and temporal differences, and provide usefull feedback. The two person test results show that improvements are possible and that more research is needed to make the system give reliable feedback.

Contents

1 Introduction

A sport coach is a person who instructs and gives feedback to other individuals on their performance and correctness. The sport coach is often a trainer and the individuals are often trainees. Such a construct is very common in almost all sport domains, such as soccer, tennis, swimming, fitness, etc. In some of these domains the trainer has to instruct a group on how to perform a movement. Especially with large groups it becomes very difficult for the trainer to keep track of all the trainees and provide feedback to them. We try to address this problem by developing a system that assists the trainer with providing feedback to the trainees using a wireless sensor network. This thesis is therefore entitled very shortly Sport Coach.

A Wireless Sensor Network (WSN) is a network of sensory devices that are wirelessly interconnected through a radio communication link. All these devices, also refered to as nodes, have some sort of processing unit to which sensors are attached and make perception of some physical quantity possible. The processing unit used can be very powerfull in terms of its computational but this will cost more energy. There are also processing units that use very little energy, but are also much more limited in their computational capabilities. The same holds for the wireless communication, where a radio link with higher throughput or with more range will require more energy than a link with very little throughput or a small range. Because of these variables there are a wide variaty of WSN's all geared to a specific application, such as environmental monitoring, structural monitoring, healt care and commercial applications [37][22][3][2][19][13].

Body Sensor Networks (BSN) are a special type of WSN in that they are mostly wireless but do not necessarily need to realize this via a radio link. BSNs are comprised of nodes worn on body that communicate via radio link, via the hosts body, or wired with other nodes on the body. Usually at least one node is capable of wireless communication. Examples of sensors used with BSNs are temperature, heart rate, muscle activity, neural activity and motion [28][21][24][15].

Activity Recognition is a field of research that investigates how to accurately detect different activities a person is performing. Examples of such systems are recognizing activities of daily living like walking, sitting, standing, cooking and eating[1][38][33][27], recognizing gestures from video or motion sensor data [42][49], or recognizing interaction between one or more persons or objects [32][47][10][11]. The latter is also known as Interaction Detection.

This work concentrates on Interaction Detection and, more specificaly, examines the use of BSNs to accomplish the task of finding spatial and temporal differences of human body motion between two persons using inertial sensors in an online and decentralized manner.

Our goal is to develop a system that can autonomicaly give feedback to the user wearing such a BSN regarding his movement with respect to a BSN on another person's body. The feedback given should be on spatial differences such as difference in distance traveled, orientational and temporal differences such as lateness or earlyness. Furthermore, the feedback should be provided sufficiently fast so that the user can correct his movement.

In this thesis we detail the design and implementation of such a system using one sensor node for the trainer and one sensor node for the trainee. Individual components of the system are simulated and tested and the overall system performance is evaulated in terms of accuracy and response time and compared with similar work.

The remainder of this document is organized as follows: First a more detailed problem description is given in section 1.1 followed by the general outline of the solution in section 1.2. Next, related work is address in chapter 2 after which a detailed description of the system architecture is given in chapter 3. After that all the components of the architecture are described in more detail from chapter 4 to chapter 6, followed by a description of the implementation in chapter 7. Finally the evaluation and conclusion section conclude this document.

1.1 Problem Desciption

This thesis is focused on Sport Coaching in a fitness environment, where one sport coach instructs one or more individuals on their movements. We aim at assisting the sport coach in this process by using multiple types of movement sensors and derive from these sensors how a trainee is performing with respect to the trainer.

Important characteristics of typical fitness movements are speed, frequency, direction and displacement. Addequate sensors need to be selected so that these characteristics can be measured. The type of wireless nodes that are used in this thesis is fixed such that the sensors also need to be hardware compatible with the wireless sensor node.

The signals from these sensors need to be sampled and prepared for further use. The sampling frequency needs to be high enough to capture the characteristics of fitness movements. Preparation of a signal is applying appropriate preprocessing such as filtering out noise, depending on the intended use of the signals.

The signals captured by the nodes of the trainer need to be communicated wirelessly to the nodes of the trainee. At least one node of the trainer needs to send his data to one node of the trainee, but other, more complex associations are also possible, depending on the method used for comparing the data.

To compare the sensed data of the trainer with the trainee, one or more methods or algorithms are needed to find the spatial, temporal and directional differences. Furthermore, the algorithms used need to be capable of being deployed on resource limited sensor nodes, which means that algorithms need to run online and in a real-time manner.

From the information produced by the comparison, feedback needs to be given to the user. The form of this feedback however is not the objective of this thesis and is therefore left open.

The goals of this project are:

- Sample and condition the sensor signals appropriately for the method of comparison.
- Transfer the trainer movement data to one or more nodes of the trainee.
- Find an appropriate algorithm for comparing movement data.
- Simulate, design and implement the complete system according to the requirements as descibed below.
- Evaluate the overall performance of the system with real tests using typical fitness movements.

The following requirements are defined for this project:

- The delay between the end of a movement and the feedback must not exceed two seconds.
- The detection must be done online and in real-time.
- The system must be capable of detecting differences in:
  - amplitude of movements with at least 20cm accuracy
  - direction of movements with at least 45 degrees accuracy.
  - duration of movements with at least 500 miliseconds accuracy.

The following assumptions are made for this project:

- Sensor nodes are synchronized with an negligible error.

The problem is simplified further as follows:

- Only movements between 1Hz and 10Hz are considered.
- Only one node for the trainer and one node for the trainee is used and are in direct wireless communication range.
- Energy effeciency will not be considered.
- Sensors are placed at best possible locations.

1.2 Solution Overview

The ultimate goal of this project is to compare the movements of a trainee with the movements of a trainer. This comparison should provide insight in temporal and spatial differences. The solution to this problem involves appropriate preprocessing of the sensor data, sending this data from the trainer to the trainee, comparing the data and generate feedback from the comparison.

Preprocessing is the process of filtering out noise, transform the signal to another representation, or extract features from the signal. Filtering is choosen for two tasks: extracting the gravity vector from the accelerometer data and for smoothing the pure movement acceleration vector. The gravity vector is further improved by using a kalman filter. The kalman filter is choosen because it can make a sub optimal estimation when employed as steady state kalman filter while requiring only few computational resources. The pure movement vector is obtained by subtracting the gravity vector from the raw accelerometer signal. Both the pure movement vector and the gyroscope signal are filtered with a low pass filter to remove high frequency noise from the signal. The sensors are oversampled at 40Hz and downsampled after filtering to 20Hz. All the preprocessing steps are descibed in more detail in chapted 4.

A segmentation algorithm is used to define periods of time to make a verdict of. This time periods are needed by the algorithm that compares the movement data. For segmentation two methods are used of which the first detects activity and rest. The second method detects the change of angle of the movement vector over a short period of time. When the angle change exceeds a specified threshold a segmentation is made. The segmentation algorithm is used only on the node of the trainer, as the trainer defines the movements and the speed. The segmentation algorithm is elaborated further in chapted 5.

The movement vector, gravity vector, gyroscope rate and the segmentation information are then streamed wirelessly from the node of the trainer to the node of the trainee. On the node of the trainee the comparison is then made of his movement data with the received data from the trainer.

Dynamic Time Warping (DTW) is used as algorithm to compare the movement data. The power of DTW is that it can detect patterns in signals even when they are of different lengths or of different speed. This means that DTW, unlike cross-correlation, detects the similarity of signals that are a stretched version of the original signal. The downside of DTW is that it, in its classical form, requires many computational resources. An optimization of DTW is therefore made such that it can be used for online decentralized processing, which is explained in detail in chapter 6. The DTW algorithm produces a path through a distance matrix. From this path information on differences in timing and the distance between the two signals with the given path can be found. For this the segmentation events that are sent by the node of the trainer are used to define which part of the path will be evauluated. Difference distance measure to compare the movement data can be used. How the DTW path is analysed and how different distance measure influences the result is the subject of chapter 6.4.

2 Related Work

In this section a short survey will be given on related work. Activity Recognition is a vast research domain that intersects with the domains of Image Processing, Audio Processing and Motion Sensor Data Processing. We focus on Interaction Detection involving one or more persons using Body Sensor Networks or Wireless Sensor Networks. The subsections are categorized based on the the similarity measure used.

2.1 Distance based similarity measures

A method for detection and classification of interaction between two persons using Feature Distance is presented in the work of Ruzena Bajcsy et al. [6]. The setup considers persons wearing five wireless sensor nodes that stream data wirelessly via a base station to a computer. The sensor nodes are equipped with 3D accelerometer and 2D gyroscope. The goal of the system is to detect if a classified movement deviates from the normal case. If the normal case would be when a care giver helps a care taker to sit down into a chair, this action would deviate when the care giver does this violently or when the care giver is not helping the care taker to sit down. To detect these situations a classification system is used that needs to be trained. Inputs to the classification system are feature vectors generated from accelerometer and gyroscope data. The classification is made by selecting the state of which feature has ''minimum distance'' to the feature that is tested. Using intensive training of the system a detection rate of at least 91% is achieved.

A similar method is used by Davrondzhon Gafurov et al. [14] for Gait Recognition using wearable motion sensors. Gait Recognition can be used in security systems for authentication of persons. Sensors are placed on foot, hip, pocket and arm. Features vectors are extracted from the frequency domain for the arm sensor and from the time domain for the other sensors. The time domain feature vector is found by detecting gait cycles which are then normalized and combined to produce an average gait cycle. The frequency domain feature vector is constructed by selecting the amplitude of some frequency components using Fourier Coefficients. The Euclidean distance of the feature vectors and a template is then matched with a specified threshold. The results in terms of EER (Equal Error Rate) are 5%, 7%, 10% and 13% for the foot, pocket, arm and hip respectively.

Yuji Ohgi [30] uses Dynamic Time Warping (DTW) to analyse the motion of a swimmer's arm stroke and the swing of a golf club. DTW is a method for finding similarity in two timeseries of data that are not necessarily of same length and speed, and mostly used in speech recognition. The collected data from accelerometers and gyroscopes is sampled at 900hz and matched against a reference set of data using DTW. This is done offline on a Personal Computer. The manhattan block distance is used as distance measure for DTW.

Distance measures as used by Ruzena Bajcsy et al. [6] and Davrondzhon Gafurov et al. [14] are simple and generaly cheap in terms of computation. The main disadvantage of the method used by the authors is that it is not possible to detect timing related information. In contrast, DTW is an algorithm that can use any distance meaure to detect the similarity of two data sets. Furthermore, the two data sets do not need to be aligned nor do they need to be of same length. This means that if the two data sets are time series, one can be a delayed or stretched version of the other set. A disadvantage of DTW is that it is computationaly intensive.

2.2 Correlation based similarity measures

Ryan Aylward et al. present in [5] a technique to detect the correlation among dancers and who is leading or lagging using wireless sensors with accelerometers and gyroscopes. Additionally they track the activity of the dancers and contact of body parts of the wearer and between others. This information is then used to control the progression of the music. Using this technique the authors are able to detect at least 0.3 seconds delays between dancers and accurately calculate the correlation. They also conclude that by using time averaging functions and windowed correlation for activity measurement, too long delays are introduced which make the feedback to the music look unnatural.

Correlation is also used by Martin Wirz et al. [46] to detect the formation and dissolving of groups of people. The goal of the research is to develop a mechanism to detect collective behavior patterns in an online decentralized manner using body worn sensors. To overcome the difficulty of people not walking in-step, simple mean and variance are used instead of raw signal data. Pair wise cross correlation is then calculated as a measure for similarity among the individuals. After detecting the groups, a classification is done to find behavior primitives. The proposed system is not implemented but data is collected from real world experiments which is used for a centralized simulation to assess the validity of the approach. These results show that with the approach taken similarity can successfully be measured to detect if two people walk together using a simple classifier.

Marin-Perianu et al. present in their work a lightweight, inexpensive and fast incremental algorithm for calculating cross-correlation [29]. Using this method it is possible to calculate correlation coefficients on small resource constrained devices such as wireless sensor nodes. As validation of the technique an implementation is made that detects if two wireless sensor nodes move together or separately.

The correlation function used by Ryan Aylward et al. [5] and Martin Wirz et al. [46] can detect similarity when the signals are shifted in time, but not when one signal is stretched. Also, these two systems use offline centralized processing. The optimization made by Marin-Perianu et al. in [29] to the correlation function introduces another disadvantage. Their method requires very accurate synchronization, because it cannot detect delays.

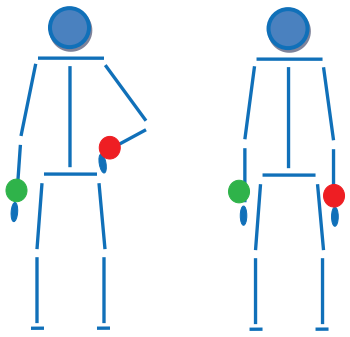
2.3 Coherence based similarity measures

Increasing interactive computing and communication devices with which people work may become problematic for many people in terms of handling and organizing the devices. Additionally many of these devices can or require intercommunication of some kind. As a means to automate this process Jonathan Lester et al. propose a system that detects if two devices are carried by the same person [25]. As a measure for similarity the coherence function is used which is a measure of similarity of two signals in the frequency domain. Using this technique they can successfully detect if two devices are carried by the same person, with detection rates up to 100%, even when the system is fooled by two persons walking in-step.

By using frequency domain signals, all timing information is lost and therefore, detecting delays is not possible. Additionaly, a time stretched signal will have very different frequency characteristics than the orignal signal.

3 System Architecture

In this section a top-down explanation of the system architecture is given. Because we want to focus on the process of comparing movements and not on the most efficient communication method, nodes placed on the trainer are designed as Master and the nodes placed on the trainee are designed as Slave. Furthermore, the mapping of the master nodes to the slave nodes is one-to-one. This means that if a master node is placed on the left wrist the slave node should also be placed ont the left wrist and exchange of sample data happens only between these two nodes. This is defined as a *pair of nodes*. More than one slave can be connected to a master, but we will not consider this case. Figure 1 shows an example of two pairs of nodes placed on the trainer and the trainee. The red pair is attached to the left wrist and the green pair is attached to the right wrist of the trainer and the trainee. A base station is used with which all nodes can communicate. This base station is connected to a computer for visualization and debugging purposes only. All communication between the nodes and the base station is realized through the wireless communication facility of the sensor node. The sensor nodes will be placed at the best possible location. The wrists are choosen to place the sensors on because at this point high accelerations occur, while the sensor can still be attached without much discomfort.



(a) Master (b) Slave

Figure 1: Sensor Placement

Taking the level one step lower to the software level the following components are required on the nodes, of which some are master or slave specific. The hierarchy of the components is shown in figure 2.

Sampling is the process of taking data samples of the sensors. This component is subject to strict and accurate timing as subsequent processing may fail otherwise. This component is needed on both master and slave.

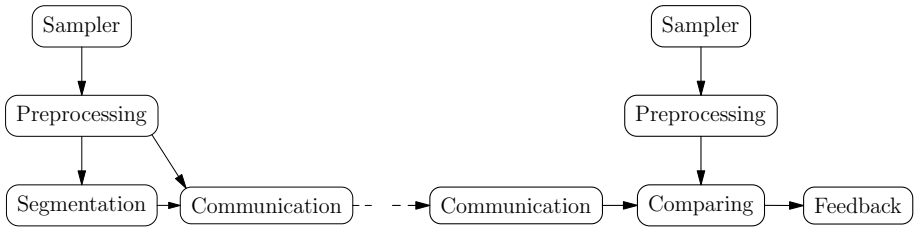
Preprocessing involves conditioning and refining the sensor data and extracting features from the sensor data. This is done in the preprocessing component which is elaborated in chapted 4.

Segmentation is used to mark the beginning, change and end of movements in the stream of sensor data. The segmentation component runs only on the master node as the trainer is supposed to do it right. The methods used to accomplish the segmentation are explained in chapter 5.

Communication between the master and the slave node is done wirelessly using the wireless communication facilities of the sensor node. A simple scheme is used that is explained in section 7.2.5

Comparing of movement data is done on the slave nodes. The comparison identifies differences in the amplitude, direction and timing of movements. The algorithm used for this process is elaborated in chapter 6.

Feedback is generated from the result of the comparison of movement data, and sent to the basestation. A simple feedback mechanism is used which is explained in section 7.2.7.



(a) Master (b) Slave

Figure 2: Software components

4 Preprocessing

When measurements are taken from sensors, these measurements can contain parasitic information in the form of noise. This noise can be introduced at many places in the measurement chain, e.g. when converting from analog to digital representation noise maybe introduced or the physical quantity that is measured may introduce noise. This can be removed via filtering in a preprocessing stage. However, preprocessing is more than just filtering. In this stage also a specific frequency band can be extracted from the sensor data or features such as statistical information of the measurent data can be extracted.

Our first goal is to avoid measuring aliases of the sensor data and to filter out noise. How this is accomplished is explained in section 4.1.

Our second goal in performing preprocessing is to detect the direction of the gravity vector. Firstly, a low pass filter is applied to extract the gravity from the accelerometer data. Next, this gravity vector is improved with the turn rate of the gyroscope using a Kalman filter. Gravity detection is the subject of section 4.2.

4.1 Noise removal

To cancel out noise, a technique called oversampling [45] can be used. Noise tends to have the highest concentrations near the sampling frequency because aliases of all higher frequency noise will show up there, and because of measuring uncertainty of the Analog to Digital Converter. By oversampling this noise is then moved away from the measured frequency band.

The highest frequency of movements expected is emperically found to be 10Hz. According to the Nyquist theorem [44], data should at least be sampled twice the time of the highest frequency component present in the measured signal, such that the sampling frequency should at least be 20Hz. Using oversampling we choose therefore to sample the sensor data at 40Hz.

4.2 Gravity detection

One of our requirements states that the system should be able to detect differences in the direction of the movements of the trainee and the trainer. The signal acquired from the accelerometer is the summation of the accelaration caused by gravity and the acceleration caused by movement of the device. This is an undersirable effect when comparing the movement directions, as it then depends on the orientation of both devices how big the measured difference is.

We choose to extract the gravity vector from the signal of the accelerometer and then subtract this gravity signal from the accelerometer signal to obtain the movement vector. Having a movement vector with gravity eliminated allows for better comparison of the movement direction, and the gravity vector of two devices can be used to compared the device's orientation with respect to gravity.

A good first approximation of the gravity vector can be made using a low pass filter, but this approach may fail when the sum of the imposed acceleration and the gravity becomes very small, or when the orientation with respect to gravity changes too quickly. Here a Kalman filter can help to make a better approximation of the gravity vector using the turn rates of the gyroscope.

To design the Kalman filter we first need to answer the following important questions:

- What method should be used for orientation? Vectorial or angular position?
- What Kalman filter model can be used? normal, extended, or steady state?
- How to make a best possible estimation with a two axis gyroscope?

These questions turn out to be very inter-related. The solution is found by approximating the gyroscopic rates as measured by the gravity vector from the accelerometer. This process is explained in section 4.2.3. With the angular rates of change of both the measured gravity vector and the gyroscope, a linear intergration model can be used for the steady state Kalman filter, which is the subject of section 4.2.4. The method to extract the gravity vector from the accelerometer data is the subject of section 4.2.2. First the principles of Kalman filters are provided in section 4.2.1.

4.2.1 Principles of Kalman filters

A Kalman filter is a linear estimator that gives an optimal estimation given only the previous state and an optional input. The process that is estimated must be a linear process that can be descibed by the following linear difference equation:

$$x_k = A_k x_{k-1} + B_k u_k + w_k \tag{1}$$

where A_k is the state transition matrix of the process, B_k the matrix describing the inputs u_k to the process and w_k is the white ghaussian process noise. The measurements taken from the process are then describe by:

$$z_k = H_k x_k + v_k \tag{2}$$

where H_k is the measurement model of the system and z_k is the new measurement. Both the measurement and the process are expected to have noise. The covariance of this noise is denoted by R_k and Q_k respectively.

The Kalman filter has two stages: predict and update. In the predict stage a model of the process using transition matrix F_k and input matrix G_k is used to predict the current state of the system. Often F_k equals A_k and G_k equals B_k.

The following two equations compute the predicted a priori state $x_{k|k-1}$ and the predicted a priori covariance $P_{k|k-1}$.

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + B_k u_k \tag{3}$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \tag{4}$$

Then using the observation model matrix H_k, the difference of the measurement z_k and the predicted state is computed, known as the measurement innovation:

$$y_k = z_k - H_k \hat{x}_{k|k-1} \tag{5}$$

and the innovation covariance is calculated as:

$$S_k = H_k P_{k|k-1} H_k^T + R_k \tag{6}$$

The Kalman gain is a quantity computed from the predicted a priori (process) covariance $P_{k|k-1}$ and the innovation covariance (measurement) S_k using the following equation.

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \tag{7}$$

The difference between the predicted state and the measured state multiplied with the Kalman gain is now the update applied to the predicted state.

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k y_k \tag{8}$$

The only thing that remains is to update the predicted a priori covariance, calculated as:

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{9}$$

When the state transition matrix is not linear, a linearization can be applied to the state transition matrix by taking the Jacobian (Partial derivative) of the transition matrix. The

Kalman filter is then known as an extended Kalman filter. Another variant is the steady state Kalman filter. This type of Kalman filter can be used when the covariance of the process and measurement noise are constant, in which case the Kalman gain will converge to a constant. The downside of a steady state Kalman filter is that it is not an optimal estimator because it cannot adapt to changes in noise.

4.2.2 Gravity from the accelerometer

To extract the gravity from the accelerometer data we choose to apply a low pass filter. A Finiti Impluse Response (FIR) filter is used because of its numerical stability [43] and the fact that it is easy to implement. We have empirically found that, with regard to our specific type of considered movements, a cut-off frequency of 0.8Hz gives excelent results. This filter has a damping of 10db and 29 filter coefficients using 40 Hz sampling frequency. The gravity vector from this filter is used as a basis and is further improved using a Kalman filter and the rates of change from the gyroscope, which is the subject of section 4.2.4.

4.2.3 Angular rate of change approximation

The used Kalman filter, which is the subject of section 4.2.4, uses the euler angle coordinate system. The gravity vector extracted from the accelerometer however, is expressed in the cartesian coordinate system. To convert from cartesian coordinates to euler angles, at least two vectors are needed that descibe the orientation in cartesian coordinates [40]. Weisstein [40] describes a concise method to make this conversion. This method uses non-linear least squares fitting [41], and the computational requirements of this method are too high to be implemented on wireless sensor nodes. We therefore choose to make the conversion using an approximation.

The gyroscope measures the angular rates ϕ and θ of, respectively, the X and Z axis. The approximation of the angular rate of the Y axis is made by deducing it from the current and previous gravity vector with respect to the two known gyroscope axis X and Z. This is done by rotating the old gravity vector with the two known gyroscope rates of change ϕ and θ, and projecting it on the X/Z plane as shown in figure 3. The angle between the resulting vector and the projection of the current gravity vector on the X/Z plane equals the rate of change ζ_{appr} of gyroscope axis Y.

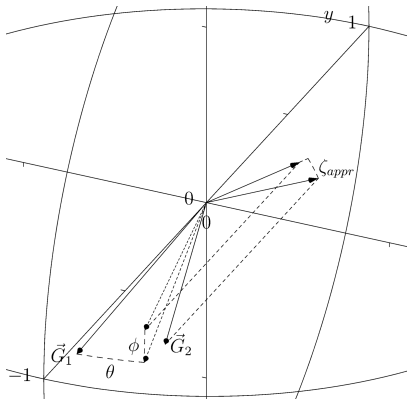


Figure 3: Measuring gyroscopic rate

Using the same method, we make an approximation of the angular rates of change of X and Z as measured by the change of the gravity vector, effectively converting the change of gravity vector to Euler angular rates of change. This is done by taking the old gravity vector and rotate it by ϕ and the previously calculated rate of change ζ_{appr} and projecting it on the X/Y plane. The angle difference of the resulting vector with the current gravity vector projected on the X/Y plane now holds the rate of change θ_{appr} as measured by the gravity vector. This can be done for the rate of change of ϕ_{appr} as well.

4.2.4 Improved estimation of gravity

We choose to use a Kalman filter to improve the estimation of the gravity vector with gyroscope data. This has the advantage of adding more speed when the orientation changes too quickly to be followed by the low pass filter. Also, the gravity extracted by the low pass filter suffers from overshoot, which will be damped by the Kalman filter when the gyroscope data disagrees.

A Kalman filter needs a model of the system to make an estimation of the new state of the system. The goal is to have a very lightweight Kalman Filter with respect to computational requirements. Others have used Kalman filters like the Unscented Kalman Filter in combination with quaternions to estimate orientation of a rigid body [23] or a classic Kalman filter with quaternions to estimate angular velocity and position [50]. Saito H. et al. [34] use in their work a classic Kalman filter to correct joint angles measured from gyroscope with inclination measured by an accelerometer.

We choose to use the euler angles coordinate system so that the much smaller and simpler steady state Kalman filter can be used. The advantage of a steady state Kalman filter is that it is much more efficient from a computational demand point of view. The downside of using the Euler coordinate system is that it gives singularities at $\sim 90°$. The alternative of a vector based model would require a rotation matrix as state transition matrix, which is time dependent. In that case only an extended Kalman filter can be used to overcome the nonlinearity of the state transition matrix, which is too computationaly demanding.

The state model for the Kalman filter that fits our system is the integration of the measured angular rates of change ϕ_{appr} and θ_{appr}, obtained as described in section 4.2.3, into an angular position. The Kalman filter will then correct the measured angular position with the angular position from the gyroscopes. The angular position obtained by the Kalman filter is not absolute, which means that it does not represent the direction of the gravity. Instead, the correction computed by the Kalman filter is also applied to the original gravity vector, resulting in an improved gravity vector.

Figure 4 shows the gravity signal from the low pass filter and the correction by the Kalman filter as dashed lines and solid lines, respectively. The graph shows the plot of the gravity vector collected from an experiment where an arm is swing from the horizontal position upward for 90 degrees. Here, the Kalman filter not only adds turn speed, but also removes the overshoot that from the original gravity vector.

The graph in figure 5 shows the plot of a similar experiment but now the up and down movement is made repeatedly. In this figure the correction from the Kalman filter is visible between the second and third swing, which are made in quick succession.

Another experement is shown in the graph of figure 6 where a sideway movement is made while the orientation of the node is held constant. The direction of movement is in the X direction, as can be seen in the graph where a small deviation is visible on the X axis.

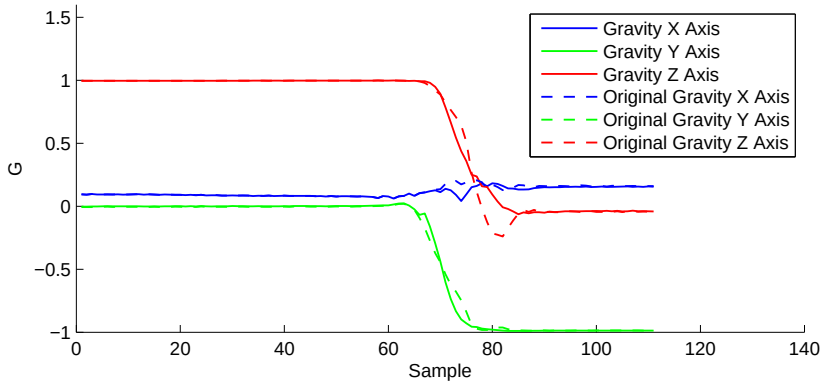


Figure 4: Corrected gravity with Kalman Filter

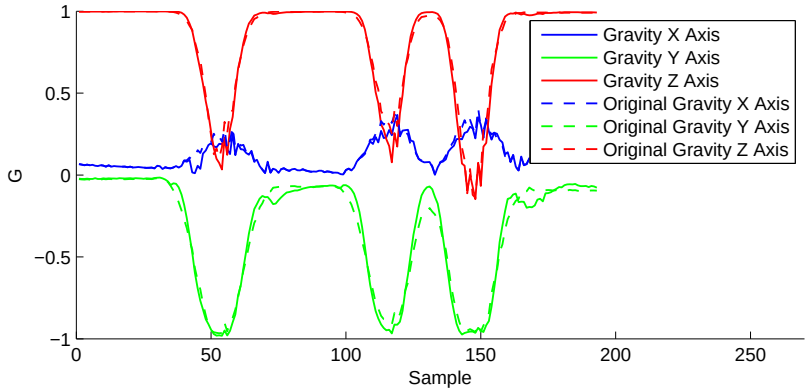


Figure 5: Gravity of repeated swing movement

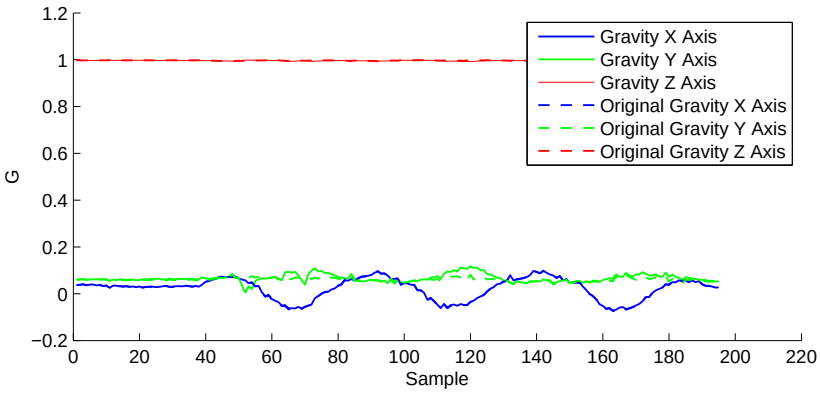


Figure 6: Gravity of repeated sideway movement

5 Segmentation

We have empirically found out that the algorithm to compare the movements of the trainee with the movements of the trainer needs specific periods of time to evaluate over. More specifically the best periods to evaluate are from a beginning to the end of a movement. To this end an algorithm is needed that makes a temporal segmentation of the realtime stream of data, such that each segment will hold at least one movement. This algorithm will run on the node of the trainer and sends segmentation events along with the stream of movement data to the node of the trainee.

Segmentation techniques can be classified in manual segmentation [20] and automatic segmentation [16][17][9]. Manual segmentation can be done offline by examining the data or online using for example a pushbutton. Both of them are regarded as unacceptable as both the trainee and the trainer should not be concerned with such a task and offline segmentation is just not possible as the entire system is to run online in realtime. In [16] Guenterberg et al. present an automatic segmentation technique using windowed standard deviation to detect transitions between sitting and standing states of a human body. Another algorithm also from Guenterberg et al. uses the signal energy to distinguish between rest and activity [17]. Chambers et al. [9] calculate the log likelihood function over a sliding window. Sharp changes of the likelihood values correspond to change of acceleration value. Others have used the most simple form of segmentation by using a sliding window with some overlap [31].

All mentioned automatic segmentation techniques suffer from the problem that stationary signals are detected as rest. As the movements considered in the Sport Coach project can be relatively slow, these methods are not usable. As an illustration we have evaluated the standard deviation of the magnitude of the acceleration of an arm movement, shown in figure 7. The sample rate of the data stream is 20Hz and the standard deviation is evaluated with a sliding window of 20 samples. From this figure it can be seen that the difference between rest and activity is indistinguishable.

Instead we use two separate methods to detect segmentation points. The first method detects whether the node is moving or not, and the second method detects changes in movement direction. The results of these two methods are then combined to generate segmentation events that will be sent to the slave.

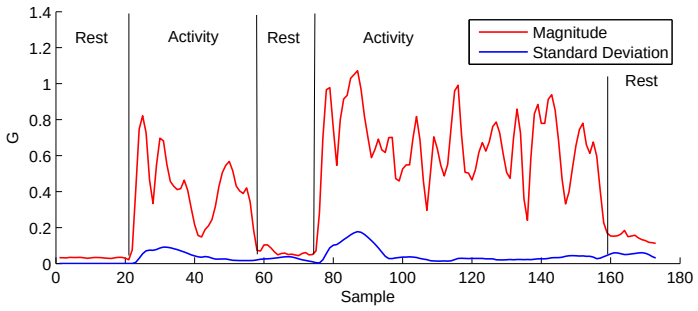


Figure 7: Standard Deviation

5.1 Movement detection

The first method we use takes the magnitude of the movement vector and the absolute value of the rates of change of the gyroscope, to detect activity or rest. Three windowed moving average filters, one for the movement magnitude data and two for the gyroscope data, are used to smoothen the signals. When at least one of these moving averages becomes higher than a certain threshold, the node is considered moving. When all the averages are lower than a certain threshold, the node is considered not moving. A hysteresis between these two thresholds is applied to prevent oscillation of the movement state.

The plot in figure 8 shows the result of an experiment where the averages of the movement magnitude and the gyroscope data were collected. The experiment was made with an arm swing up and down. The sample rate is 20Hz and the averaging window size is 4 samples. There are in total three swings. After the first swing there is a rest of one second. The second and third swing are performed continuously. As the plot shows, the differences between rest and moving is very well detectable.

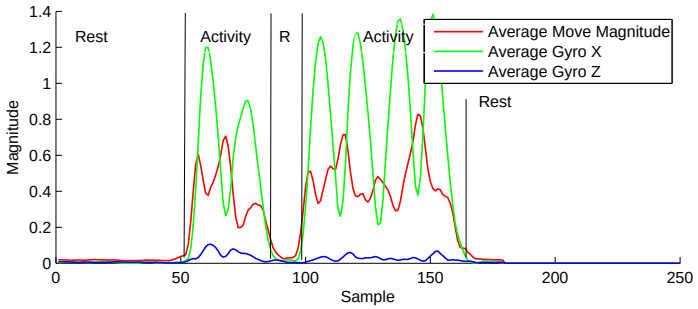


Figure 8: Averages of movement data

5.2 Direction change detection

The second method we use detects abrupt changes in movement direction. This method evaluates the dot product of the current movement direction with a number of movements from recent history, as specified by equation 10.

$$f(i,m,n) = \sum_{j=m}^{n} 1 - \vec{x}_i \cdot \vec{x}_{i-j} \quad m > 0 \quad n > m \quad |\vec{x}_i|, |\vec{x}_{i-j}| > 0.2 \tag{10}$$

The input vector $\vec{x}$ must be normalized so that the dot product produces the cosine of the angle. The result of the dot product is then inverted such that two vectors pointing at the same direction produces zero and two vectors pointing at opposite direction produces two. This is evaluated and summed for a number of vectors from recent history starting at $i-m$ till $i-n$ such that the result is smoother. We have empirically found that the magnitude of the vectors (unnormalized) should be bigger than 0.2 to prevent noise triggering the algorithm.

We choose to use a threshold system to generate a direction change event that will be used by the fusion system that will fuse the results of the two methods. This direction change event is generated only when equation 10 becomes higher than a certain threshold. The figure shows that

Figure 9 depicts the result for the direction change detection algorithm of the same experiment from section 5.1. The data rate is 20Hz, $m = 5$ and $n = 12$ of equation 10, which means that the direction is compared with directions from 6 till 12 samples from history. The Z axis of the gravity vector is also plotted as a reference for when the node is swing up or down. The plot shows that both with and without rest in between a swing, the direction change is very well detectable. The delay introduced by this method is at least 300 miliseconds and at most 600 miliseconds, as is confirmed by the plot where the delay on average is 400 miliseconds.

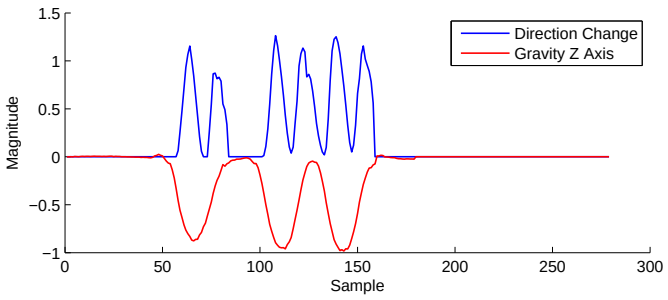


Figure 9: Upward swing

A similar experiment is made using a sideway swing with the same settings. In figure 10 the result of this test is shown. With a sideway swing the algorithm performs slightly worse, but still direction changes are very good detectable. Another test is shown in figure 11. This test is made by ‘‘figuratively drawing a traingle’’ in the air with the sensor node. Four corners are passed of the triangle movement, as can be seen in the figure as well.

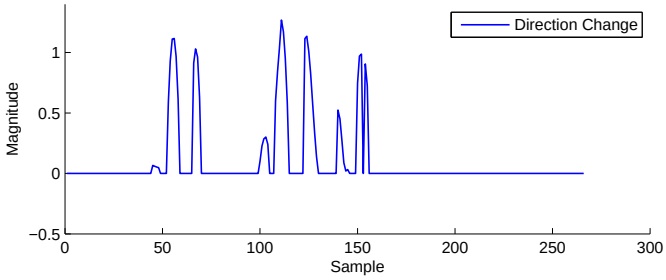


Figure 10: Sideway swing

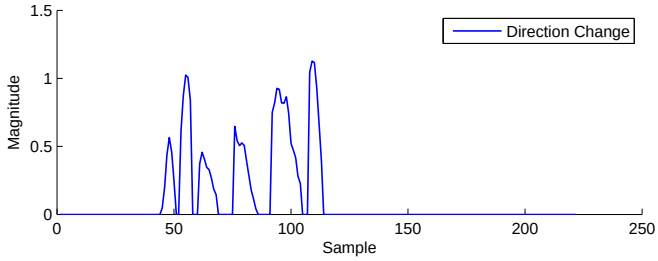


Figure 11: Triangle movement

5.3 Fusion

The two methods are fused by taking the movement state and the direction change event. When the state of movement transits from rest to moving a ‘‘Movement Start’’ segmentation event is generated. A ‘‘Movement Stop’’ event is generated when the state of movement transits from moving to rest. When the movement state is moving, events from the direction change detection are allowed. Because both event types can occur with quick succession in principle, this is prevented by allowing events to occur only when a timeout has expired since the last event.

The experiment of section 5.1 is reused to produce the segmentation event plot shown in figure 12.

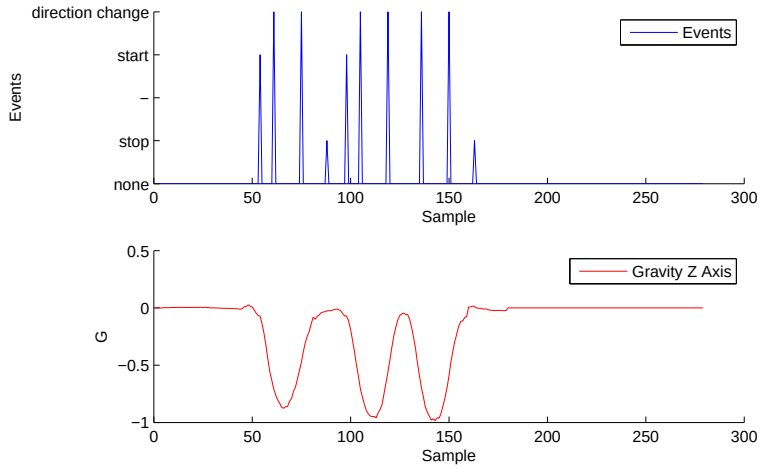


Figure 12: Segmentation

6 Comparing Movements

The measurements of the sensors are a representation of the movements made by the person to which the sensor node is attached. These measurements are then transfered from the node of the trainer to the node of the trainee, where they are compared to find differences in their movements.

There are many methods to accomplish this, such as the work of Ryan Aylward et al. [5] and Martin Wirz et al. [46] who use correlation as a measure of similarity of movements. The standard correlation function is improved by making it incremental by Marin-Perianu et al. in [29], who also successfully used it to measure similarity. Another method is used by Jonathan Lester et al. [25] who used Coherence to measure similarity.

What all these methods have in common is that they produce poor results when movements are very similar and differ only in speed. We choose therefore the similarity measure used by Yuji Ohgi [30], in which Dynamic Time Warping (DTW) is used to assess the performance of swimmers and golfers. DTW in its classic form as used by Ohgi is however not suitable to be used in an online and realtime manner.

To overcome this problem an optimized version of the Classic DTW in terms of computational requirements is presented in this section. Our new technique, Fast Incremental Dynamic Time Warping (FIDTW), computes the optimal shortest warping path and can run on low power and resource constraint devices. Firstly, we provide some background on DTW with a mathematical definition and some notes on related work regarding optimization of the conventional DTW algorithm. The remainder of this section will be devoted to the optimization of the DTW algorithm and will finish with an evaluation of the choosen algorithm.

6.1 Classic DTW

DTW is a general time alignment and similarity measure for two temporal sequences and was first introduced by Bellman [8]. Suppose we have the sequences $C(i), 1 \leq i \leq l, C(i) \in R$ and $T(j), 1 \leq i \leq l, T(j) \in R$. These are called a class sequence and a test sequence, respectively. With these two sequences an $I \times J$ distance table $D(i, j)$ is constructed with which similarity can be measured. From the distance table a warping path W is then calculated which consist of a set of table elements that define a mapping and an alignment between $C(i)$ and $T(j)$.

$$W = \left\{ w(i(q), j(q)) \;\middle|\; \begin{array}{l} q = 1, \ldots, Q \\ max(i, j) \leq Q \leq I + J - 1 \end{array} \right\}$$

where $i(q) \in \{1, \ldots, I\}$ and $j(q) \in \{1, \ldots, J\}$.

This warping path is restricted by *Continuity*, *Monotonicity* and *Endpoint* (the path must start at $i(1) = 1$, $j(1) = 1$ and end at $i(Q) = I$, $j(Q) = J$). By summing the local distances over the warping path, the local distance $DTW(C, T)$ is obtained. One of the possible choices for finding the best alignment between the two sequences is to find the warping path with the minimum DTW distance out of all possible warping paths. With the following recursive steps, the optimal warping path can be found and applies local constraints to the path:

$$D(i, j) = d(i, j) + min \left\{ \begin{array}{l} D(i-1, j-1) \\ D(i, j-1) \\ D(i-1.j) \end{array} \right\}$$

The recursion is generally initialized as $D(1,1) = d(1,1)$ and terminates when $i = I$ and $j = J$. The time and space complexity of this approach is $O(IJ)$.

Matters may become more clear with an example using a class sequence $\{1, 2, 3, 6, 12\}$ and a test sequence $\{2, 1, 2, 3, 6\}$ denoted respectively as $seq1$ and $seq2$. Note that, apart from the first data point, the test sequence is equal to the first four data points of the class sequence. In figure 13, the distance matrix of the two sequences is shown on the left and the accumulated distance matrix with the warping path on the right. From this figure it can be seen that the warping path follows the lowest accumulated distance. Figure 14 shows the time alignment of the two sequences, i.e. the mapping of $seq1$ to $seq2$.

	1	2	3	6	12
2	1	0	1	16	100
1	0	1	4	25	121
2	1	0	1	16	100
3	4	1	0	9	81
6	25	16	9	0	36

	1	2	3	6	12
2	1	1	2	18	118
1	1	2	5	26	139
2	2	1	2	18	118
3	6	2	1	10	91
6	31	18	10	1	37

Figure 13: Classic DTW

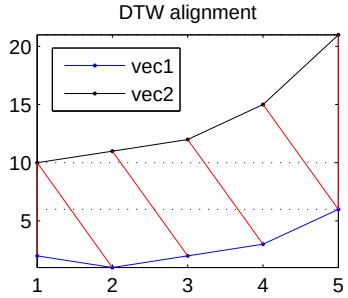


Figure 14: Time alignment of the two sequences

6.2 Existing Optimizations

Previous work on optimization of the DTW algorithm focuses mainly on reducing the time and space complexity and assume offline use[4][35][36][48]. More related work is focused on the optimization of DTW for the alignment of a continuous stream of data with a test sequence.

A notable work on the latter is from Guiling Li et al. [26]. Presented in their work is an algorithm that aligns a continuous stream of data with a static test sequence. By only storing the start vector and the end vector they are able to reduce the time and space complexity to $O(n)$. Although this approach does compute the shortest distance, it is not possible to find the optimal path with this technique, as this requires traversing back trough the accumulated distance matrix. Another downside of not knowing the path is that the dynamic alignment of time, a very valuable piece of information for this project, cannot be measured.

Another optimization approach for DTW is from Dixon [12]. His approach also aligns a stream sequence with a static test sequence. Dixon also incrementally computes the distance matrix similar to the work from Guiling Li et al. [26]. The difference is that Dixon does store the accumulated distance matrix such that the shortest path with respect to a certain bound can be found. This bound specifies how deep the distances are calculated from the current position, and a found path is thus not necessarily the optimal path.

6.3 Fast Incremental Dynamic Time Warping Algorithm

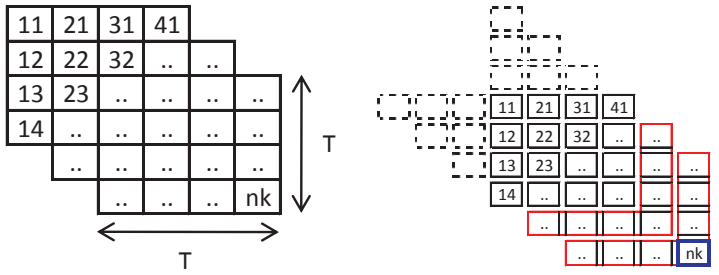
In this section we explain our optimization made to the classic DTW algorithm. One should first note that the aim of the optimization is to make it faster in terms of required computation time and not necessarily make it less resource hungry in terms of required memory space. Secondly the optimization should preserve all the characteristics of the classic DTW, which means that:

- The optimal warping distance should be preserved.
- The optimal warping path should be preserved.
- The optimal warping path must be able to be reconstructed afterwards for analysis.

Additionally, the algorithm should work with two real-time streams that are expected to be synchronized in time with an error $E < \epsilon$. In case of time synchronized data, it is also observed, as can be seen from figure 13, that:

- The diagonal line from top-left to bottom-right represents one-to-one allignment of time.
- A warping path that deviates from the diagonal to the lower left side means that sequence 1 has a delay compared to sequence 2.
- A warping path that deviates from the diagonal to the upper right side means that sequence 2 has a delay compared to sequence 1.

When a maximum positive and negative delay T is considered, the DTW distance matrix can be reshaped as shown in figure 15a, where the lower left and upper right matrix elements that fall outside the delay T are removed. This is justified as the algorithm is required to be able to measure delays up to T. The now appearing shape takes the form of stacked arrows, as can be seen in figure 15b. This arrow shape is from here on called an *arrow object* and all the elements enclosed by the arrow shape *directly belong* to the arrow object. The element of the arrow object which is emphesized by a blue box is from here on called the *center* of the arrow object. The stack form will be exploited to make the algorithm incremental. The main construct of the algorithm will then be a list of these arrow objects. The list is



(a) Reshaped DTW (b) Appearing stacked arrow form

Figure 15: Optimization steps of DTW

extended at the head until a maximum length L, after which one arrow object is removed from the tail everytime an arrow object is added at the head.

In Classic DTW, the accumulated distance matrix is recomputed everytime the first row or column is removed. One reason for this is because otherwise the path distance obtained from the accumulated distance matrix will not be representative anymore. Also from a practical point of view when dealing with stream data, values cannot be added into infinity. One possible way of dealing with this is to recalculate the accumulated distance matrix when needed, but this would still require $\sim O(n^2)$ computation time.

It is observed from the Classic DTW algorithm that, due to local constraints, only immediate neighbour information is needed to calculate the accumulated distance at a certain point in the matrix. This means that when a new arrow object is added, the accumulated distances of this new arrow need only to be consistent with its successor. However, the accumulated distances are also used to find the shortest DTW path by traversing backward through the accumulated distance matrix. This is solved by immediately storing the neigbor with the lowest accumulated distance for all the elements of the new arrow object, so that the shortest DTW path can be found without the accumulated distances. Now that the accumulated distances of the new arrow object are calculated, and the reason to keep the accumulated distances consistent is eliminated, the accumulated distances of the new arrow object can safely be adjusted. The only valid method for adjustment is subtraction, because division would cause range inconsistency of the distances of a new arrow object with the accumulated distances of its successor.

Although it is justified to make the adjustment by subtracting the lowest accumulated distance in the arrow object, only one-eight of the lowest accumulated distance is subtracted to better preserve the scale that these distances represent. This is required for the Flexible Endpoint algorithm that will be explained in section 6.3.1.

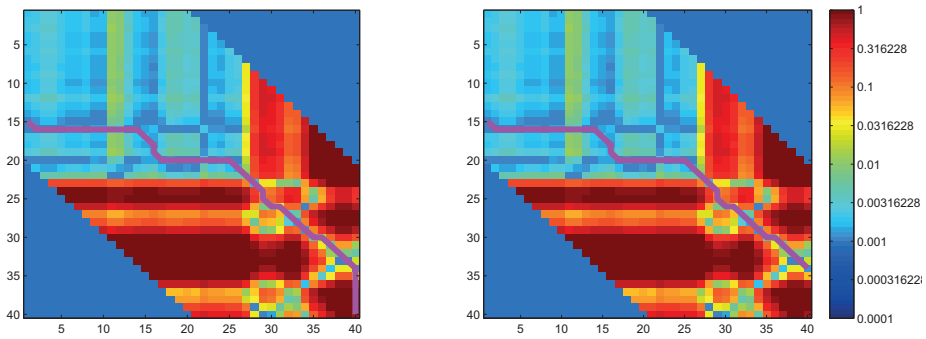
To derive the computational complexity of adding a new arrow object we assume one penalty for computing the distance of an element and three penalties for computing the accumulated distance of an element. This needs to be computed for T elements of an arrow object (see figure 15b) so that the total computational complexity becomes $\sim O(6T)$. There is no penalty for computing the neighbour with lowest accumulated distance because this is already computed when the accumulated distance of an element is calculated, and requires only extra storage per element of an arrow object. However, traversing backwards along the warping path becomes slightly faster. The distance of a warping path is now obtained by adding up all the distances when traversing backward along the warping path, as the elements of the accumulated distance matrix do not represent real distances anymore.

6.3.1 Flexible Endpoint

While testing the algorithm, we found out that the algorithm may find a warping path with a very high distance although the two signals were very similar but slightly delayed. An example of such a case is shown in figure 16a. From this figure it can be seen that the distance in the area at the bottom right corner is very large. This is because the movement of the trainer is finished at this point, but the trainee did not. From figure 16a it can be seen that there is a area with very high distances at the bottom right corner. The DTW path has to cross this area to reach the bottom right corner, which gives the wrong impression that the two movements are very distant.

This problem is solved by allowing flexible endpoint at the start of the path at the bottom right corner. The bottom right corner is the head of the list of arrow objects of the Fast Incremental DTW algorithm. Evaluation of the shortest DTW path starts at the head of the list of arrow objects at the center of the arrow (the blue box in figure 15b). A flexible endpoint algorithm is therefore just finding a more suitable start position in this arrow object. Flexible endpoint algorithms are widely used with DTW. It is unclear who first proposed the use of flexible endpoints, but probably one of the first to propose such a technique is Haltsonen [18].

We choose to accomplish the task by taking the accumulated distance of the center of the arrow object as reference and find an accumulated distance in the arrow that is at least 25% or lower than the reference with a 1% penalty for every element it is further away from the reference. An example of a path produced with the flexible endpoint algorithm is shown in figure 16b. The path is identical, except in the bottom right corner, where the path produced with the flexible endpoint approach does not end in the far bottom right corner.



(a) Classic DTW path

(b) Flexible endpoint

Figure 16: DTW paths

6.3.2 Distance Calculation

The distances for the DTW distance matrix are a combination of distance measures of multiple sources. The used sources are the magnitude of the movement, the direction of the movement and the change of orientation as measured by the gyroscope. The distance between the movement magnitudes are calculated using the squared Euclidean distance function (Equation 11). A magnitude is only one value such that equation 11 simplifies to Equation 12

$$d(p,q) = \sum_{i=1}^{n} (p_i - q_i)^2 \tag{11}$$

$$d(p,q) = (p-q)^2 \tag{12}$$

The changes of orientation measured by the gyroscope are linear, therefore the Euclidean distance can safely be used as a distance measure. It is a two dimensional gyroscope, therefore $n = 2$ in Equation 11.

The distance between the direction of movements is measured using Equation 13, which calculates the angle between two vectors by evaulating the arc cosine of the dot product of the two normalized movement vectors. This distance measure is only evaluated when the magnitude of both vectors is large enough, because the noise becomes more dominant when the magnitude of the movement is small.

$$d(\vec{x}_1, \vec{y}_1) = \arccos(\vec{x}_1 \cdot \vec{y}_1) \quad |\vec{x}_1|, |\vec{y}_1| > 0.2 \tag{13}$$

These three distances are then combined by multiplying them with a weight factor and then adding them up to one distance measure (Equation 14). The largest component of the summation is also stored in the distance matrix as a means to trace back what caused the large distance. This is needed when the DTW path is analysed, which is subject of chapter 6.4.

$$d_{combined} = W_{magn} * d_{magn} + W_{dir} * d_{dir} + W_{orient} * d_{orient} \tag{14}$$

6.4 Path Analysis

With the temporal segmentation information an analysis can be made of the path produced by the FIDTW algorithm. From this analysis appropiate feedback can be given to the trainee. From the DTW path a number of statistical data can be extracted. Figure 17 depicts a typical path produced by the FIDTW algorithm when two movements are very similar. The diagonal elements from bottom-right to top-left of the distance matrix represent perfect time alignment. With respect to the stated direction of the diagonal line, a deviation to the right represents a positive delay and deviation to the left represents a negative delay. Small differences between two movements may already cause small delays, often shifting back and forth from positive to negative delays. Therefore we use the mean of the delays in a segment.

The DTW path distance represents the similarity of two signals. This distance also depends on the length of the path and is therefore normalized by the path length. The smaller the path distance the more similar the movements are.

When the path distance is high, and as such the two movements are detected as dissimilar, the cause of the high distance needs to be found. This information is, for every element, stored in the DTW distance matrix. Then during path analysis this information is recovered from the distance matrix such that it can be used as feedback.

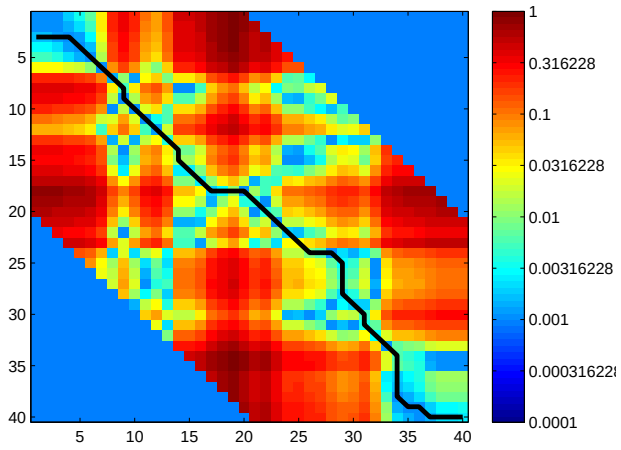


Figure 17: Example DTW path

6.5 Evaluation

In this section we will evaluate the FIDTW algorithm with real world test data obtained from arm swing movement. The movement is made by swingin the arm to which the sensor is attached upward for 90 degrees. The sensor is attached to the wrist, and the movement data is collected and preprocessed using the scheme of section 4. The master node is attached to the right wrist and the slave node to the left wrist of the same person. The pure movement data is used, which means that the gravity has been eliminated.

The weigths of equation 14 are empirically defined. This calibration is made by performing different kinds of movements while examining the response of the DTW algorithm. When the weigth of the direction difference is too high compared to the weigth of the magnitude, the algorithm will be oversensitive to small direction errors and the magnitude will have little contribution to the construction of the path. The same holds for the opposite situation. Unfortunately the direction of a movement is relative to the orientation of a sensor node, such that all experiments must be made using the same orientation for the slave and the master nodes. The gyroscope rate is not used with these experiments, such that the weigth is zero.

In our first experiment we perform an identical movement with both the master and the slave node. In figure 18 the plot of this movement is shown for both the master and the slave. The DTW distance matrix produced by the FIDTW algorithm is shown in figure 19. This result shows that the algorithm detects that the movement is not delayed and very similar. The latter is not very visible from the figure, but under the black line the elements are mostly light blue to light yellow.

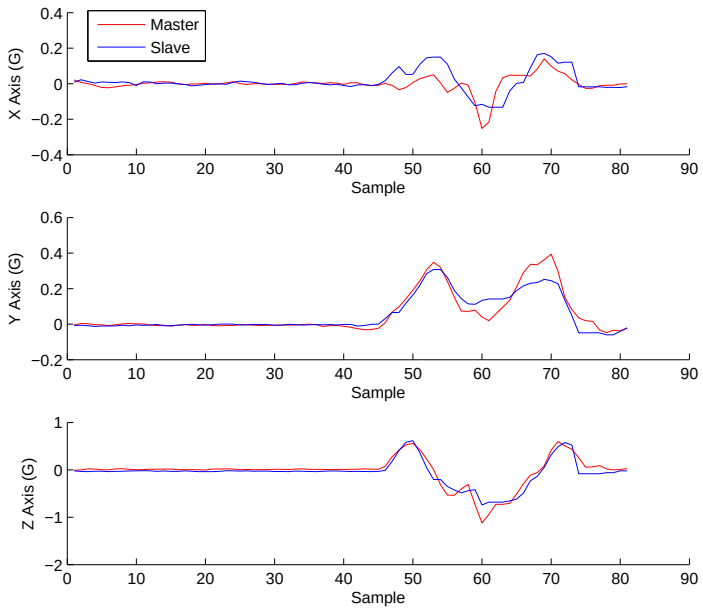


Figure 18: Swing movement of master and slave

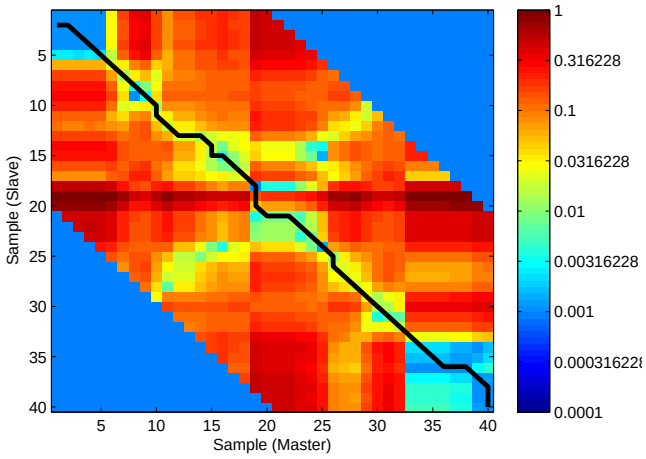


Figure 19: DTW distance matrix of swing movement

With a second experiment using the same movement and settings of the first experiment, we test the measuring of the delay of a movement. This is tested by letting the slave perform the movement slightly later than the master. The plot of this swing movement is shown in figure 20. From this figure we can tell that the slave performed his movement about 250ms after the master. The DTW distance matrix of this movement produced by FIDTW is shown in figure 21. As can be seen in this figure, the DTW path deviates slightly from the diagonal to the upper right corner. The average deviation of the path from the diagonal is exactly 4 samples, which means a measured delay of 200ms.

In a third experiment we test the performance of DTW when two movements are different. This is tested by letting the slave perform a sideway swing instead of an upward swing. The settings are again the same as for the first experiment. The movement performed by the master and the slave is shown in figure 22. The movement of the slave is now most appearent on the X axis. Figure 23 shows the DTW matrix of this movement. As can be seen in this figure, the distances between the two movements are high to very high for the major part of the distance matrix. Also the distances of the elements that are covered by the DTW path are high to very high, which means that the algorithm classifies the movement as dissimilar.

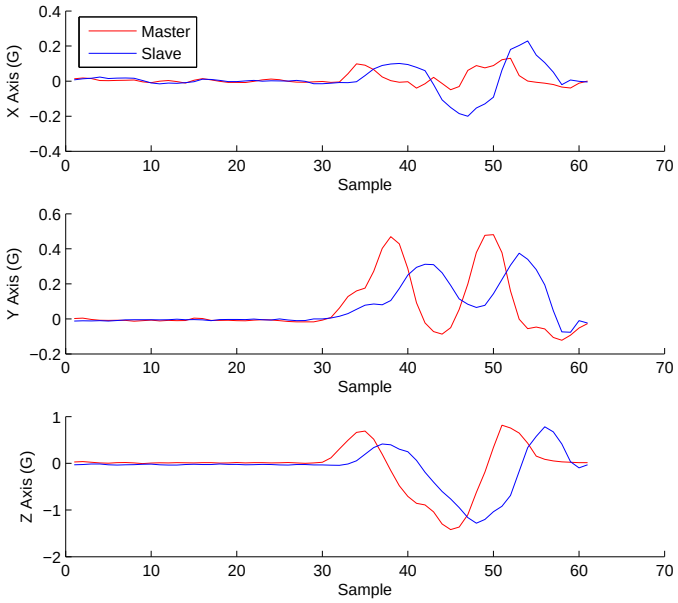


Figure 20: Swing movement of master and slave delayed

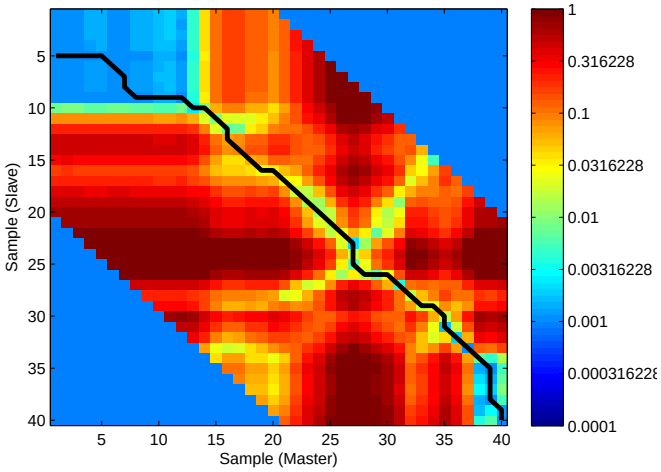


Figure 21: DTW distance matrix of swing movement

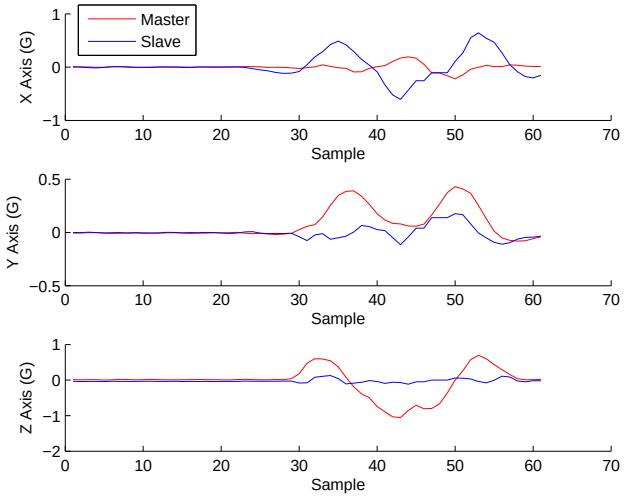


Figure 22: Swing movement of master and slave

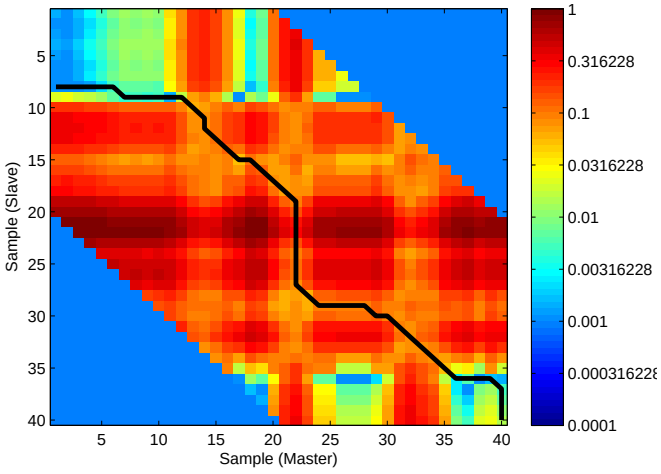


Figure 23: DTW distance matrix of swing movement

7 Implementation

All the modules addressed in the Architecture (section 3) are implemented in Java 2 Mobile Edition - the platform of the SunSpots. With a clock speed of 60Mhz the SunSpots are relatively fast devices, when compared to other wireless sensor nodes. But to be able to fully exploit the capabilities of the Sun SPOT, the implementation must be done with care and optimizations should be made whenever possible. The implementation of all the modules is detailed in this section, but first a description of the used hardware is given.

7.1 Hardware

The hardware used for this project are modified Java Sunspots, displayed in figure 24. Java Sunspots are small wireless devices capable of running java code conform the Mobile Edition of the Java Standard and run at 60 Mhz. Besides basic functionality like pushbuttons, LEDs, etc, the Sunspots are equipped with an accelerometer and there is a breakout port with both analog and digital I/O.

We have made a modification to the SunSpots in the form of an extension of the set of movement sensors. Factory Default the Sunspots are equipped with an accelerometer to measure acceleration. Change of orientation is not measurable with an accelerometer, at most a rough approximation can be made. Put differently, with an accelerometer only displacement can be measured. To include rotation in the set of measurable movements, both a gyroscope and a digital compass are added to the sunspots.

A gyroscope is a device that measures turn rate of a certain axis. Different gyroscopes vary in accuracy, range and the axis's measured. The expected rotation speed is reasonably low and there is no need for high accuracy. However, the set is limited in a sense that it has to be in a prefabricated form such as an extension board, not in the form of a chip. This is required such that the extension boards can easily be attached to the Sun SPOT.

The gyroscope chosen is the LPY530AL from STMicroelectronics that is available on a breakout board from Sparkfun. This is a two axis gyroscope measuring pitch and yaw. The range of this gyroscope is 1500 deg/sec using low precision or 300 deg/sec using high precision. A better option would have been to have a three axis gyroscope but this unfortunately was not available. Also for the digital compass, there was only a two axis version available on breakout board, the HMC6352 from Honeywell. However this sensor is not used in this project but is placed for future use.

A key feature of the Sunspots is the wireless communication, based on the 802.14 standard. With this wireless radio link a distance of 100 meters can be covered. Furthermore, the communication facilities of the Java library for the Sun SPOT provide reliable communication methods, which greatly reduces packet loss caused by weak radio links.

Figure 24: Sun SPOT

7.2 Software

In this section the implementation of the software modules is described. The software for the SunSpots is written for Java 2 Mobile Edition (J2ME). J2ME can be seen as a stripped down version of Java 2 Standard Edition. J2ME is also used for mobile phones where it is mainly used for mobile game development. An application written for J2ME is called a Midlet. Most J2ME virtual machines do not support executing Midlets in parallel, but Midlets can create multiple threads which are executed in parallel. Besides the standard packages there are hardware dependent packages, such as the hardware timer package for the Sun SPOT. Because the software for the SunSpots uses these harware dependent packages, it cannot be compiled for other J2ME devices without modification.

First, an overview will be given using UML diagrams of both the master and slave software.

7.2.1 Overview

Because there are master and slave specific modules there will be master and slave specific Midlet applications. Both types will have the sampling and the preprocessing module but the other modules are dependent on the type of node. First the modules for the master node are described with the aid of a UML class diagram, shown in figure 25. The SensorSampler performs the sampling of sensors and the FilterWorker performs preprocessing. Both these classes are threads and are started at system startup. The sensor readings are then sent from the SensorSampler to the FilterWorker via a stream such that the FilterWorker will never block the SensorSampler. The FilterWorker has control of the filters, the Kalman filter and the PostProcessor. Here the general part of the model which is identical for master and slave software ends and becomes master specific.

The MasterPostProcessor class is a generalization of PostProcessor and performs the segmentation and communication tasks. The transmission of samples to the slave node is handled by the FrameWriter thread. The FrameWriter is implemented as a thread because the MasterPTPConn will block when the slave node is not available or when the connection is temporarily lost. This way the connection with the slave node will not block the FilterWorker thread. A Frame consists of the gravity vector, movement vector, gyroscope rates and segmentation data.

The software for the slave nodes also uses a generalization of PostProcessor named SlavePostProcessor. The SlavePostProcessor class has an FIDTW object and an FrameReader object. Similarly to the master software the FrameReader is a class derived from Thread and is used to receive frames from the master node, also to prevent blocking of the FilterWorker. The FIDTW class performs the comparison of the remote frames with the local frames. Using the segmentation data received from the master node an analysis of the DTW path is made for every segment, as explained in section 6.4.

No synchronisation is done between the master and slave nodes, which means that frames are transmitted when they are ready, and no buffering is done. An update of FIDTW is done when a new frame arives from the master and the current frame of the slave is picked as the other input. This method is choosen because the transmission time is short as only one hop communication is considered, and tests with synchronisation have shown that synchronisation is unreliable because there are small drifts in intervals of timer tasks (see section 7.2.2).

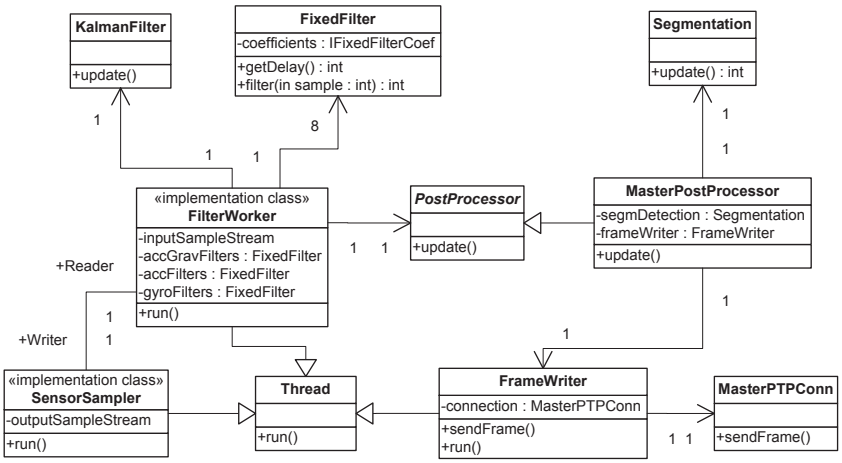


Figure 25: UML class diagram of Master software

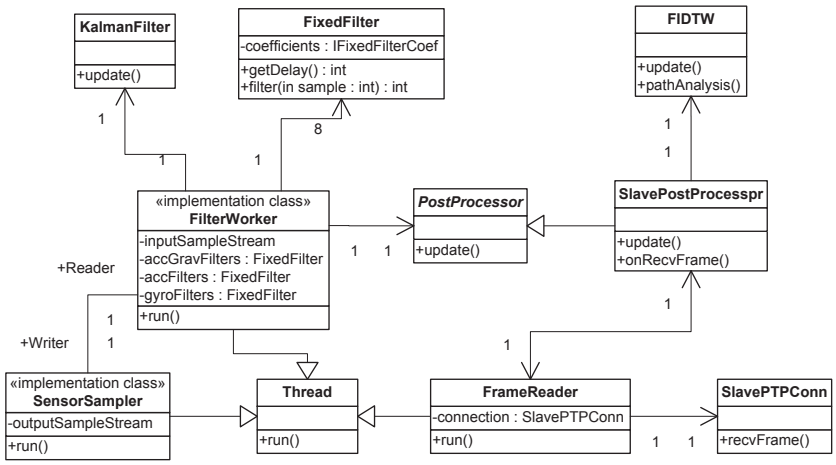


Figure 26: UML class diagram of Slave software

7.2.2 Sampling

Sampling is the process of taking measurements of the sensors at specific time intervals. This interval can be very strict when calculations depend on the time interval between measurements used, or loose when it does not matter if the interval has some slight variations. The Kalman filter used in the preprocessing stage integrates the rate of change of the gyroscope into an angular position, and thus depends on the interval between measurements.

Unfortunately, the Squawk Java Virtual Machine which is the operating system of the nodes, does not support realtime scheduling and no gaurantees are given that a task activated by a hardware timer will also be scheduled in time. Experiments have been held to test if the timed task is reliable enough to be used for sampling when there are multiple tasks competing for processor time. The timer frequency for the test is set to 40Hz and another hardware timer is used to measure the elapsed time between task activation.

The result is shown in figure 27. The mean interval is found to be very close to the specified interval with an error of 0.0178ms. The standard deviation is also very low at 0.0861ms. However, there are also bigger glitches occuring at irregular rates that result in the current interval becoming approximately 1ms longer and the next interval 1ms shorter. This automatic compensation works until some interval after which it becomes too short to be compensated.

According to the reasoning for choosing the sampling frequency in section 4, a sampling frequency of 40Hz is used, corresponding with an interval of 25ms, well above the interval where glitches become a serious problem. Because the error of the mean interval and also the standard deviation are very low, the timing is decided to be reliable enough to be used as a sensor sampler. If the reliability of the timing would have been insufficient, the elapsed time would have to be measured for every sample and used for the integration process in the Kalman filter. Because the result of the intergration is not directly used by any processing other than the Kalman filter, it is justified to use a constant 25ms for integration.

The sampled values are converted to a system wide 16 bit fixed point format. For the gyroscope measurements Q3.12 is used, which stands for 3 bit mantisa, 12 bit fraction and 1 sign bit. The units for the gyroscope measurements are in radians. For the accelerometer measurement Q2.13 is used, with unit G. These measurements are then sent to the preprocessor module via an I/O Stream.

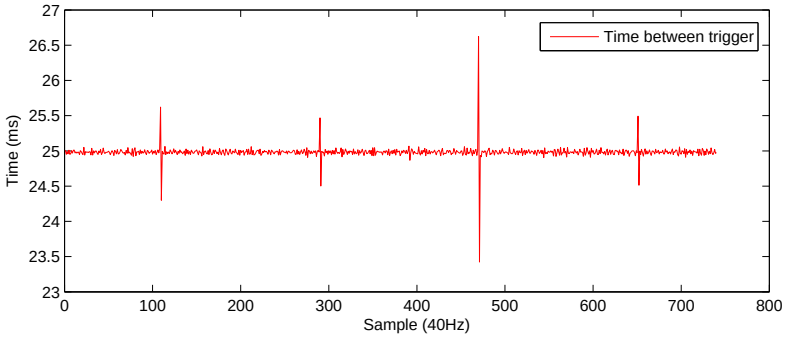


Figure 27: Measured intervals of timer

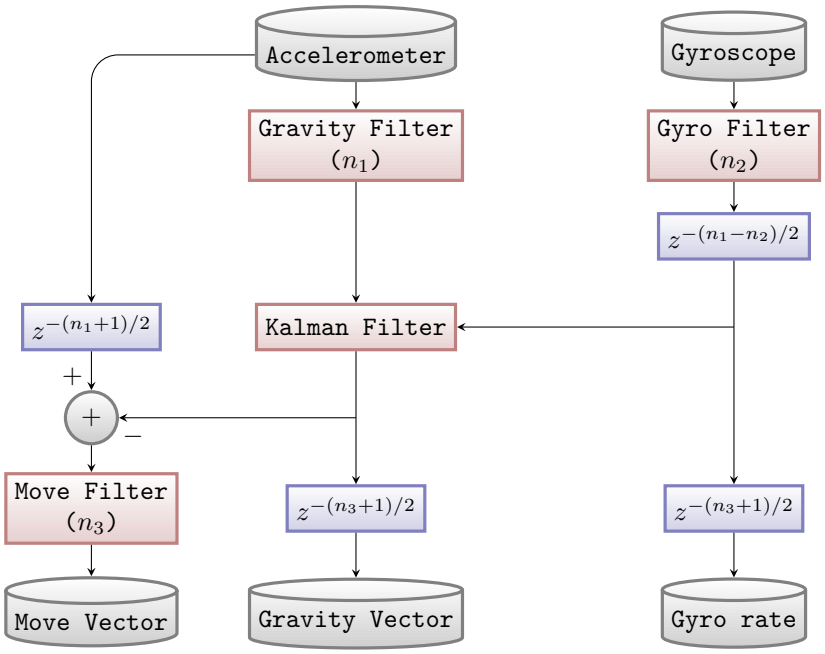


Figure 28: Filter structure

7.2.3 Preprocessing

The preprocessing module implementation is the FilterWorker class that is executed in its own thread. The FilterWorker receives the accelerometer and gyroscope values from the SensorSampler via a stream that blocks until the data is available. The accelerometer data is then fed through a low pass filter to extract the gravity from the acceleration signal. This is a 29th order Finite Impulse Response(FIR) filter with 10db damping and the cutoff frequency at 0.8Hz. The output of this filter is then fed through the Kalman filter which results in the final estimation of the gravity vector.

The gravity vector is then subtracted from the raw accelerometer data to obtain the movement vector. To remove noise the movement vector is then fed through another low pass filter of 5th order with 16db damping and the cutoff frequency at 9Hz. The output of this filter is the final movement vector. The gyroscope measurements are also filtered with a low pass filter of 5th order, 30db damping and cutoff frequency at 8Hz.

A side effect of FIR filtering is the introduction of a delay equal to the number of the filter coefficients plus one and divided by two. To keep all the data synchronized, delays are introduced where required as can be seen in the diagram in figure 28. In this diagram the flow of the sensor data through the filtering mechanism is depicted. The number of filter coefficients are shown in braces for the FIR filters and the delays are represented with the delay operator z.

After all filtering is applied, the update method of the PostProcessor class is invoked which will perform the final processing of the data, depending on the type of node. First the Kalman filter and the required trigonometric approximations are elaborated.

Trigonometric Approximation
To make a better estimation of the gravity vector we use a steady state Kalman filter, as explained in section 4.2.4. The taken approach uses angular rate and position and as a consequence, trigonometric functions are required. The standard floating point implementation of Java is accurate but slow. Instead, an approximation for the sine and cosine functions using polynomials is choosen as presented by Jasper Vijn [39]. The advantage of this method is that it is both fast and accurate and does not use huge lookup tables.

Inspired by this method, an approximation of the arc sine and arc cosine is made with one polynomial and a small lookup table for the coefficients of the polynomial. Only one function needs to be approximated as one follows from the other. The arc cosine function is chosen and is first split in half because the other half is just a mirrored version. The used half of the cosine is then split in six regions that define which coefficients are used for the polynomial. A 5th order polynomial is used which gives a good balance between complexity, speed and accuracy.

The most problematic part of the arc cosine function is at the boundaries of -1 and 1. The further away from the origin, the higher the error of the approximation. For the larger part of the arc cosine function this can be solved by dividing it into regions with their own coeffictients. For an approximation with real numbers an infinite number of regions would be required. Fortunately not real numbers but integers are used and the part of the approximation where the error becomes too large can be approximated with a lookup table, which is only a few numbers depending on required precision. Because the arc cosine expect inputs of -1 up to and including 1, the Q1.14 format is used for input and Q3.12 for output. The coefficients are found using the polynomial approximation tool of Matlab. Pseudocode 1 shows the pseudocode of the algorithm.

A comparison is made with another popular approximation named cordic. The test is made for every digit of precision of Q1.14, which means from -16384 to 16383 with increment 1. Cordic is setup to use 13 iterations to make an approximation. Figure 29 shows the error comparision made by subtracting the approximated value from the real value. For this test the mean error of cordic is 2.5994e-4 and 1.0706e-4 for the polynomial approximation. Using more iterations the cordic approximation becomes better, but converges to 1.2634e-4 from 20 iterations.

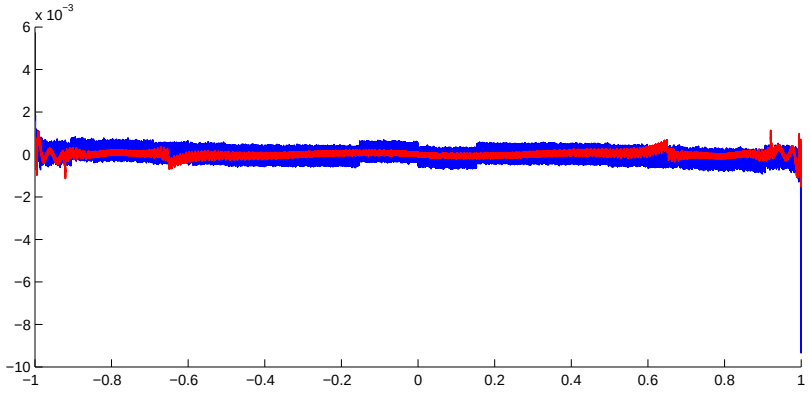


Figure 29: Arc cosine errors; red: polynomial; blue: cordic

Data: P is the current set of coefficients
Input : in: A number between and including -1 and 1 in Q1.14 format
Output: Angle in radians in Q3.12 format

```
P ← getCoefficients();
if use polynomial then
    if in < 0 then
        Sign ← true;
        in = −1∗ in;
    end
    insqr ← fpmul32( in, in);
    k ← P [2] + fpmul32( P [1], in) + fpmul32( P [0], insqr);
    l ← P [4] + fpmul32( P [3], in) + fpmul32( k, insqr);
    out ← ( P [5] + fpmul32( l, in) ) / 4;
    if Sign then
        out ← HALFCIRCLE - out;
    end
else
    out ← getOutFromLookup();
end
```

pseudocode 1: Arc cosine polynomial approximation

Kalman filter
As described in chapter 4.2.4, a steady state Kalman filter is used to improve the estimation of the gravity vector. The inputs for the Kalman filter are approximated angular rate from the change of gravity and the angular rate from the gyroscope. The approximation of the angular rate of the gravity vector is made by first rotating the previous gravity vector by the two known angular rates from the gyroscope and then measure the angle difference with the new gravity vector. Pseudocode 2 lists the steps of the algorithm. The same method is used to approximate the angular rate ϕ_{appr} about the X axis. Figure 30 shows the plot of the approximated angular rate ϕ_{appr} and the angular rate from the gyroscope ϕ in blue and red respectively.

Input : Previous gravity vector $\vec{v}_{prev}$
Input : Current gravity vector $\vec{v}_{cur}$
Input : Angular rate about X axis ϕ
Input : Angular rate about Z axis θ
Output: Approximated angular rate about Y axis θ_{appr}

$\vec{v}_{temp} \leftarrow \vec{v}_{prev}$ rotated by ϕ about the X axis;

$\vec{v}_{temp} \leftarrow \vec{v}_{temp}$ rotated by θ about the Z axis;

$\vec{v}_{projcur} \leftarrow$ Projection of $\vec{v}_{cur}$ onto X/Z plane;

$\vec{v}_{projtemp} \leftarrow$ Projection of $\vec{v}_{temp}$ onto X/Z plane;

if `length(` $\vec{v}_{projcur}$`)` > 0.2 *AND* `length(` $\vec{v}_{projtemp}$`)` > 0.2 **then**
  $\theta_{appr} \leftarrow$ angle difference between $\vec{v}_{projcur}$ and $\vec{v}_{projtemp}$;
else
  $\theta_{appr} \leftarrow 0$;
end

pseudocode 2: Angular rate approximation

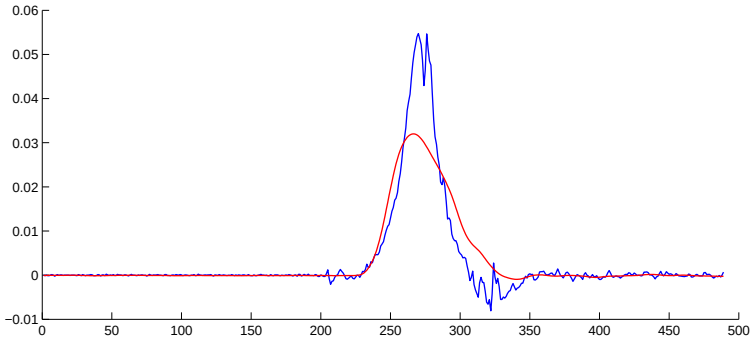
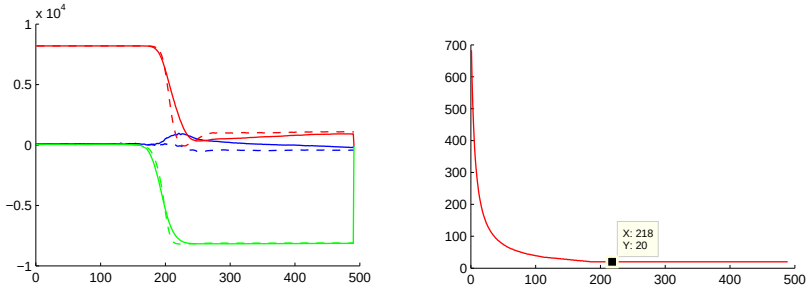


Figure 30: Approximation (blue) and real (red) angular rate

The angular rate approximations ϕ_{appr} and θ_{appr} are then integrated into angular positions which are then fed to the Kalman filter as the measurement input. The angular rates of the gyroscope ϕ and θ are the input of the Kalman filter's process model, which is also an intergration of angular rate into angular position. The R and Q matrices specifiy the ''trust'' that is given to respectively the measurement input and the process model. The values for the R and Q matrices are empirically found with tests from real data.

The way the angular rates are found from the gravity vector is not more than an approximation, and when the gravity vector is close to the axis of rotation it becomes more unreliable. Therefore the trust is put in the measurement such that the Kalman filter is more restrained with the correction it will apply from the process model. Figure 31a shows the result with $R = 0.5$ and $Q = 0.1$, where the dashed lines represent the old gravity vector and the solid lines the new estimation by thr Kalman filter.

Using graph analysis the Kalman gain is empirically found. The plot of the Kalman gain is shown in figure 31b, from which it can be seen that the Kalman gain converges to a constant (in Q3.12 format). This constant is then taken as the Kalman gain for the steady state Kalman filter, such that it does not need to be recomputed at every iteration of the Kalman filter. The functioning of the steady state Kalman filter is shown in pseudocode 3.



(a) Gravity estimation

(b) Kalman gain

Figure 31: Kalman filter test results

Input : Current gravity vector $\vec{v}_{cur}$
Input : Angular rate about X axis ϕ
Input : Angular rate about Z axis θ
Input : Approximated angular position vector $\vec{z}$

Output : Improved estimation of the gravity vector $\vec{v}_{new}$

Data: Kalman state $\vec{x}$
Data: Transition matrix $F = [1; 1] \times 2^{12}$
Data: Measurement matrix $G = [0.025; 0.025] \times 2^{12}$
Data: Measurement matrix $H = [1; 1] \times 2^{12}$
Data: Kalman gain $K = [20; 20]$

$\vec{u} \leftarrow [\phi; \theta]$;

```
// State prediction
```
$\vec{x}_p \leftarrow$ `fpmul32(` F, $\vec{x}$`)` + `fpmul32(` G, $\vec{u}$`)`;

```
// Measurement prediction
```
$\vec{z}_p \leftarrow$ `fpmul32(` H, $\vec{x}_p$`)`;

$\vec{diff} \leftarrow$ Difference between $\vec{z}_p$ and $\vec{z}$;

```
// State update
```
$\vec{x}_p \leftarrow \vec{x}_p +$ `fpmul32(` K, $\vec{diff}$`)`;

```
// Apply difference to gravity vector by rotating it with the
   difference
```
$\vec{v}_{new} \leftarrow \vec{v}_{cur}$ Rotated by $\vec{diff}$;

pseudocode 3: Steady state Kalman filter

7.2.4 Segmentation

Detection of suitable points for segmentation of the sensor data stream is done in the equally named Segmentation class. The segmentation algorithm is excecuted iteratively in the update method, which in turn is invoked by the update method of the MasterPostProcessor class. The segmentation detection is done by the two algorithms from chapter 5. The angle detection algorithm traverses from $T1$ to $T2$ points back in time. The angle difference of the current movement direction and the directions from history are then inverted and added to a total direction change value. If this value exceeds a certain threshold and there is no direction change detected in the previous state, the new state will be set to 'change detected' and an event will be set. When the previous state equals 'change detected' and the total direction change is below a certain threshold the new state will be set to zero. The pseudocode of the direction change detection is shown in pseudocode 4.

The detection of whether the node is being moved or is held still is done by applying a moving average filter of length N over the magnitude of the movement vector and the absolute value of the angular rates of the gyroscope. If at least one of these three values exceeds a certain threshold, and the previous state is 'not moving', then the new state will be set to 'moving'. If the previous state is 'moving', and all three averages are below a certain threshold, the new state will be set to 'not moving'. Pseudocode 5 lists the pseudocode of the movement detection.

The fusion of the direction change detection and the movement detection is shown in pseudocode 6. Depending in the segmentation event, no new events are allowed for some amount of time to prevent too frequent generation of segmentation events. The algorithm uses the movement state and the direction change event to generate a single segmentation event that can be either 'start moving', 'direction change' or 'stop moving'.

Input : Current movement vector $\vec{v}_{cur}$
Input : Previous detection state prevState

Output: New detection state detectionState
Output: Direction change event dirEvent

Data: History of normalized movement vectors moveHist of size $T2$
Data: Total direction change totalChange
Data: Upper bound threshold $C1$
Data: Lower bound threshold $C2$

$\vec{v}_{cur} \leftarrow \vec{v}_{cur}/|\vec{v}_{cur}|$;
totalChange $\leftarrow 0$;
for $i \leftarrow T1$ **to** $T2$ **do**
  $\vec{v}_i \leftarrow$ moveHist$[i]$;
  $angle \leftarrow 1 - (\vec{v}_{cur} \cdot \vec{v}_i)$;
  totalChange $\leftarrow$ totalChange $+ angle$;
end
if (totalChange $> C1$) **and** (prevState! $= CHANGEDETECTED$) **then**
  detectionState $\leftarrow CHANGEDETECTED$;
  dirEvent $\leftarrow CHANGEEVENT$;
else if (totalChange $< C2$) **and** (prevState $== CHANGEDETECTED$) **then**
  detectionState $\leftarrow 0$;
end
Push $\vec{v}_{cur}$ into back of moveHist;

pseudocode 4: Change of movement direction detection

Data: Upper bound thresholds $M1$, $R1$
Data: Lower bound thresholds $M2$, $R2$
Input : Current movement magnitude moveMagn
Input : Current gyroscope rate about X axis ϕ
Input : Current gyroscope rate about Z axis θ
Input : Previous movement state prevState

Output: New movement state newState

Data: History of tuple {moveMagn, ϕ, θ } magnHist of size N

Push tuple { moveMagn, ϕ, θ } into back of magnHist;

averageMagn $\leftarrow 0$;
$\phi_{average} \leftarrow 0$;
$\theta_{average} \leftarrow 0$;

for $i = 0$ **to** N **do**
    averageMagn $\leftarrow$ averageMagn + magnHist$[i]$.moveMagn;
    $\phi_{average} \leftarrow \phi_{average}$ + magnHist$[i].\phi$;
    $\theta_{average} \leftarrow \theta_{average}$ + magnHist$[i].\theta$;
end

averageMagn $\leftarrow$ averageMagn$/N$;
$\phi_{average} \leftarrow \phi_{average}/N$;
$\theta_{average} \leftarrow \theta_{average}/N$;

if ((averageMagn $> M1$) **or** ($\phi_{average} > R1$) **or** ($\theta_{average} > R1$)) **and** (prevState $== NOTMOVING$) **then**
    newState $\leftarrow MOVING$;
else if (averageMagn $< M2$) **and** ($\phi_{average} < R2$) **and** ($\theta_{average} < R2$) **and** (prevState $== MOVING$) **then**
    newState $\leftarrow NOTMOVING$;
end

pseudocode 5: Movement detection

Input : Movement state moveState
Input : Direction event segmEvent
Input : Previous segmentation state prevState

Output: New segmentation state segmState
Output: Segmentation event segmEvent

Data: Forced segmentation event inactivity forceIdleCount

```
// Stay in state until changed detected
segmState ← prevState;
if forceIdleCount == 0 then
    if prevState != SEGM_STATE_IDLE then
        if moveState == NOTMOVING then
            segmState ← SEGM_STATE_IDLE;
            segmEvent ← SEGM_EVENT_MOVESTOP;
            forceIdleCount ← NOMOVE_INACTIVITY_TIME;
        else if dirEvent == CHANGEEVENT then
            segmState ← SEGM_STATE_DIRCHANGE;
            segmEvent ← SEGM_EVENT_DIRCHANGE;
            forceIdleCount ← DIRCHANGE_INACTIVITY_TIME;
        end
    else if prevState == SEGM_STATE_IDLE then
        if moveState == MOVING then
            segmState ← SEGM_STATE_MOVING;
            segmEvent ← SEGM_EVENT_MOVESTART;
            forceIdleCount ← MOVE_INACTIVITY_TIME;
        end
    end
else
    decrement forceIdleCount;
end
```

pseudocode 6: Fusion of movement and direction change detection

7.2.5 Wireless Communication

The sensor samples that are collected on the nodes of the trainer and the trainee need to be brought together such that they can be compared. We choose to use a simple scheme where the node of the trainer is designed as a master and the node of the trainee is designed as a slave. The sensor samples are brought together at the slave, where they are compared by the FIDTW algorithm. The sensor samples of the master thus need to be transfered to the slave node. The transmission of the sensor samples is done wirelessly using the reliable wireless communication stack of the Sun SPOT. The result of the FIDTW algorithm is feedback information which is sent to a base station connected to a computer running a visualization application.

Master Side The implementation of the master side of the communication is split in two parts. The part that deals with the setup of the connection is implemented in the MasterPTPConn class. This class establishes the connection with the slave node. The actual transfer of sensor samples is implemented in the FrameWriter class. This class is a generalization of the Thread class so that non blocking communication is realized. No buffering is performed to prevent misalignment of the sensor samples between master and slave.

Slave Side On the slave side the implementation of the communication is also split in two parts. Similar to the master side, there is a class that establishes the connection named SlavePTPConn, and there is a FrameReader class which is a generalization of the Thread class to prevent blocking of the communication. Additionally the slave has a BaseConn class which handles the transmission of feedback information to the base station with a similar construct as of the master-slave connection.

7.2.6 FIDTW

The FIDTW class implements the FIDTW algorithm, explained in section 6.3. The mechanism of the algorithm is executed iteratively in the updated method of the class, which is invoked by the SlavePostProcessor class. Furthermore there is one private method calculateDistance and a public method pathAnalysis.

Adding new data
The update method takes a local feature and a remote feature, where a feature is a storage type which consist of the direction, magnitude and turn rate of a movement. A history of both the local and the remote features is stored, which is required for latter computation when the distances of the new arrow object need to be calculated. The length of the history depends on how long delays are required to be detectable. We choose a history of 20 such that, with a sample rate of 20Hz, delays of maximum 1 second can be detected.

Firstly, a new arrow object is created and added to the head of the list of arrows. The list of arrows is the storage form we use for the DTW distance matrix, and stores the distance, accumulated distance and the neighbor with the lowest accumulated distance. The list is implemented as a linked list to prevent undesired copying of references. The length of the list is equal to the diagonal length of a DTW distance matrix when a normal matrix would be used. When the list exceeds a certain length, one arrow is removed from the tail. The length of the list of arrow objects depends on the maximum movement duration which need to be detectable. We choose a maximum list length of 40 so that, at 20Hz sample rate, movement durations of 2 seconds can be stored. When the duration of a movement would exceed the length of the arrow list, the DTW path can still be valid. However, when the length of the arrow list is made very small compared to the movement duration, the DTW path becomes unreliable.

Distance calculation
When the new arrow object is created it is filled with the pairwise distances between local feature and the remote feature history, and the remote feature and the local feature history. The center of the arrow contains the distance between the local feature and the remote feature. The distance between two features is calculated from the magnitude and direction information, as is explained in section 6.3.2. Furthermore, this method calculates if the distance between the magnitude or the direction exceeds a certain threshold. This information is also stored in the arrow object for later use when the path is analysed. Pseudocode 7 lists the implementation of the distance calculation method.

Input : Feature Feat1
Input : Feature Feat2

Output: Total distance totalDist
Output: High direction distance highDir
Output: High magnitude distance highMagn

Data: Distance between magnitudes magnDist
Data: Distance between direction dirDist

magnDist $\leftarrow$ (Feat1.$magn$ − Feat2.$magn$)^2;

`// Calculate arc cosine of the dot product`
if (Feat1.$magn > 0.1$) **and** (Feat2.$magn > 0.1$) **then**
  dirDist $\leftarrow \arccos$ (Feat1.$dir \cdot$ Feat2.dir)/(Feat1.$magn *$ Feat2.$magn$);

if magnDist > 0.1 **then**
  highMagn $\leftarrow TRUE$;
else
  highMagn $\leftarrow FALSE$;

if dirDist > 0.1 **then**
  highDir $\leftarrow TRUE$;
else
  highDir $\leftarrow FALSE$;

totalDist $\leftarrow$ magnDist + dirDist;

pseudocode 7: Distance calculation

Accumulated distance calculation
After calculating the distances, the accumulated distances need to be calculated. The accumulated distance is the distance to get to that position in the DTW distance matrix considering the rules as explained in section 6.1. In conventional DTW the accumulated distance matrix is computed from top to bottom and left to right, because otherwise a dependency violation will occur. This means that the accumulated distances of the new arrow object need to be calculated from the outside in. The accumulated distance of an element in the arrow object is calculated by taking the accumulated distance of the neighbor with the lowest one, and then add its own distance. Simultaneously, the neighbor with the lowest accumulated distance is stored for later use when the DTW path must be retrieved. The FIDTW algorithm is a continuous process and therefore the accumulated distance must be compromised to prevent overflow of the storage type. When all the accumulated distances of the new arrow object have been calculated, they are adjusted by subtracting one-eight of the smallest accumulated distance in the arrow object.

Pseudocode 8 lists the steps from creating a new arrow object till the calculation of the accumulated distance.

Data: LocalFeatHist is a history of local features of size MaxHist
Data: RemoteFeatHist is a history of remote features of size MaxHist
Data: ArrowList is the list of arrow objects that make up the dtw matrix
Input: A local feature F_L and a remote feature F_R

```
CreateAndAddNewHead();
if Length(ArrowList) ≥ MaxDtwLen then RemoveTail();
head ← Head(ArrowList);
if Length(ArrowList) > 1 then
    // Calculate distance, accumulated distance and direction for the
       new head
    for i = (MaxHist − 1) to 0 do
        // Up is the part of the arrow that points upward
        // Left is the part of the arrow that points left
        up ← head.up[i];
        left ← head.left[i];
        up.dist ← Distance between F_L and RemoteFeatHist[i];
        left.dist ← Distance between F_R and LocalFeatHist[i];
        loAccDistU ← Lowest accumulated distance of neigbor of up;
        loAccDistL ← Lowest accumulated distance of neigbor of left;
        up.accDist ← up.dist + loAccDistU;
        left.accDist ← left.dist + loAccDistL;
        up.direction ← Direction of neighbor of up with lowest acc distance;
        left.direction ← Direction of neighbor of left with lowest acc distance;
    end
    head.center.dist ← Distance between F_L, F_R;
    loAccDistC ← Lowest accumulated distance of neigbor of center;
    head.center.accDist ← head.center.dist + loAccDistC;
    head.center.dir ← Direction of neighbor of center with lowest acc distance;
    // Adjust accumulated distance
    adjust ← Lowest accumulated distance in head / 8;
    for i = (MaxHist − 1) to 0 do
        head.up[i].accDist − = adjust;
        head.left[i].accDist − = adjust;
    end
    head.center.accDist − = adjust;
else
    // There is only one element: the center
    head.center.dist ← Distance between F_L, F_R;
    head.center.accDist ← head.center.dist;
    head.center.dir ← Direction of neighbor of center with lowest acc distance;
end
```

pseudocode 8: FIDTW

Path retrieval
To find the DTW path a backwards travers must be made starting at the head of the list of arrow objects. First a suitable start position must be found in the head arrow. An element is considered a more suitable start postion than the center when its accumulated distance is lower than 25% of the accumulated distance of the center, with a 1% penalty for every element it is further away from the center. When a suitable start position has been found, the path travers starts at that point. Every element in an arrow object knows the direction to the neighbor with the lowest accumulated distance. By following these directions from the starting point the DTW path is then found. From this travers, an array is constructed which contains the deviation of the path from the diagonal and the high direction difference and high magnitude difference flags, for every point of the path. Also the distance is stored such that the total path distance can be calculated. The pseudocode for path retrieval is split in two parts. In pseudocode 9 a suitable start postion is found, and in pseudocode 10 the path is traversed to construct an array with the path characteristics.

Output : DTW Path (Array) DtwPath

Data: ArrowList is the list of arrow objects that make up the dtw matrix

```
// Find suitable start position
head ← Head(ArrowList);
lowerBound ← head.center.accDist /8;
lowestDist ← lowerBound;
decr ← head.center.accDist /100;
startPos ← 0;
for i = 0 to (MaxHist − 1) do
    if head.up[i].accDist < lowestDist and head.up[i].accDist < lowerBound
    then
        startPos ← i +1;
        lowestDist ← head.up[i].accDist;
    end
    if head.left[i].accDist < lowestDist and head.left[i].accDist < lowerBound
    then
        startPos ← −( i +1);
        lowestDist ← head.left[i].accDist;
    end
    lowerBound ← lowerBound − decr;
end
```

pseudocode 9: Path retrieval (1/2)

```
// Do the path traversal
curArrow ← Head(ArrowList);
diagDeviation ← startPos;
pathLen ← 0;
while curArrow != null do
    if diagDeviation == 0 then
        direction ← curArrow.center.direction;
        causeFlags ← curArrow.center.causeFlags;
        distance ← curArrow.center.distance;
    else
        pos ← abs(diagDeviation) − 1;
        if diagDeviation < 0 then
            direction ← curArrow.left[pos].direction;
            causeFlags ← curArrow.left[pos].causeFlags;
            distance ← curArrow.left[pos].distance;
        else
            direction ← curArrow.up[pos].direction;
            causeFlags ← curArrow.up[pos].causeFlags;
            distance ← curArrow.up[pos].distance;
        end
    end
    DtwPath [pathLen].deviation ← diagDeviation;
    DtwPath [pathLen].causeFlags ← causeFlags;
    DtwPath [pathLen].distance ← distance;
    pathLen ← pathLen + 1;
    switch direction do
        case Up
            if diagDeviation < 0 then
                curArrow ← curArrow.previous;
            end
            diagDeviation ← diagDeviation + 1;
        endsw
        case Diag
            curArrow ← curArrow.previous;
        endsw
        case Left
            if diagDeviation > 0 then
                curArrow ← curArrow.previous;
            end
            diagDeviation ← diagDeviation - 1;
        endsw
    endsw
end
```

pseudocode 10: Path retrieval (2/2)

Path analysis
The pathAnalysis function, as listed in pseudocode 11, analyses the path array constructed during path retrieval. The analysis collects minimum, maximum and average statistics of the deviation from the diagonal and counts the accurrences of high direction distance and high magnitude distance. The latter is a measure for the cause when a path has a high distance. Also, the total distance between the two movements is calculated by adding the distances stored in the path.

This function is invoked when the segmentation algorithm that runs on the master node sends a ''direction change'' event or a ''movement end'' event. The statistics of the path, the path distance and the cause are then sent to a base station connected to a computer.

```
Output: Minimum deviation MinDev
Output: Maximum deviation MaxDev
Output: Average deviation AverageDev
Output: High direction difference count HighDirCount
Output: High magnitude difference count HighMagnCount
Output: Total path distance TotalDistance

Data: DTW Path (Array) DtwPath

MinDev ← LARGEVALUE;
MaxDev ← −LARGEVALUE;
AverageDev ← 0;
HighDirCount ← 0;
HighMagnCount ← 0;
TotalDistance ← 0;

for i = 0 to DtwPath.Length do
    if DtwPath [i].deviation < MinDev then
        MinDev ← DtwPath [i].deviation;

    if DtwPath [i].deviation > MaxDev then
        MaxDev ← DtwPath [i].deviation;

    AverageDev ← AverageDev + DtwPath [i].deviation;

    if DtwPath [i].causeFlags & HIGHDIRECTION then
        HighDirCount ← HighDirCount +1;

    if DtwPath [i].causeFlags & HIGHMAGNITUDE then
        HighMagnCount ← HighMagnCount +1;

    TotalDistance ← TotalDistance + DtwPath [i].distance;
end
```

pseudocode 11: Path analysis

7.2.7 Feedback

Feedback is the response given to the user regarding the movement that he or she made. This feedback should give the user information about how different his or her movement was with respect to the movement made by the trainer, and what was the cause of this difference. There are many different ways the feedback can be given, for example using the LED's of the Sun SPOT, a textual report on a computer screen or a visualization program. We choose to give feedback via vizualization on a computer screen, because this is the most interactive way of giving feedback.

The vizualization application is held as simple as possible, because it is not a requirement for this thesis and only meant as a demonstration application. The program gives feedback via slowly flashing colored rings that change color accoring to the correctness of the movement. When the movement is correct the color is green and changes via orange to red the more incorrect the movement is. The symbol of a hare is displayed when the movement is made before the movement of the trainer, and a snail symbol is shown when the movement is delayed. Likewise, when the major cause of an incorrect movement is the direction, the symbol of a compas is shown and a ruler symbol is shown when the major cause is the magnitude of the movement.

8 Evaluation

The Sport Coach feedback system presented in section 8.1 produces discrete feedback at discrete moments in time regarding the correctness of the most recent movement. This feedback is on three different categories: direction, magnitude and timing, and all these three categories have a degree of incorrectness. This is visually shown via symbols that grow gradually with the incorrectness of the movement.

In this section we assess the correctness of the produced feedback by performing experiments. Firstly the test methodology is described in section 8.1. Then, in section 8.2, we provide the results of the performed tests.

8.1 Test Methodology

It is difficult to test the system for absolute correctness, because the feedback that is sent from the node of the trainee to the visualization application is a general degree of incorrectness constructed from both the direction and magnitude information. The DTW algorithm will find the best match of two movements with the shortest distance. The remaining distance is then the degree of incorrectness, but it has no direct relation with, for example, the difference of the angle of the movements. This means that from the degree of incorrectness it is not possible to tell if it was a short, but big difference or a long, but small difference. Furthermore, if a movement that is in fact wrong, exhibits the same characteristics as a correct but delayed movement, DTW will detect it as a delayed movement.

Also verification itself is a difficult process. When verification would be done by means of experiments with human beings, it is very hard to have a ground truth, but this test is the most representative of a real situation. On the other hand, when performing tests with known ground truth the tests will show if the system correctly labels the movements, but it will not show the resilience of the system.

We choose to assess the correctness of the feedback using three different methods. In the first method the node of the trainer and the node of the trainee are both placed on the same wrist of one person. This test should always label the movements as correct and not delayed. With the second method the node of the trainer is placed on one wrist and the node of the trainer on the other wrist of one person. With this test there is already no real ground truth anymore, other than the feeling of the person that he made the same movement with both arms. The third method uses two persons, where one person wears the master node on the right wrist and the other person the slave node on the right wrist.

The feedback which is sent from the slave node holds information about the incorrectness of the movement, the timing and the cause of the incorrectness. The measure for incorrectness is the distance between the movements measured using the FIDTW algorithm. The timing is either an earlyness or a lateness in the form of a positive or negative average delay. When the distance is above a certain threshold, the movement can be regarded as incorrect, and the cause information then holds ,seperately, the occurences of high direction and magnitude difference.

For the three test methods a test set of four tests is defined. This test set consists of repetitive distinct movements. For every test in the set there is one movement repetition

defined for the master node and multiple variations of this movement for the slave node, which range from identical to distinctly different. What ‘‘different’’ means depends on the actual test. Table 1 depicts all the tests of the test set with an explanatory figure of the movement. The slave variations which are tested are shown in table 2. Table 3 shows the expected result per method for all slave variants. Note that not all test variations of the slave apply for all test methods, and that different thresholds are used for method 3. This is needed because we found, with early experiments, that our similarity measurement method is more sensitive than anticipated. Therefore, the distance of two identical movements measured from two hands of one person is much smaller than two identical movements measured from one hand of two different persons.

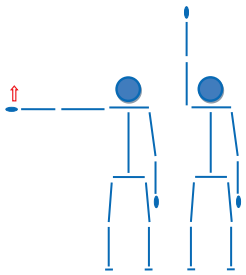
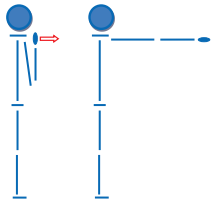
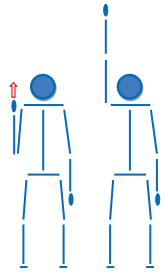
Test	Description	Figure
1	Stretched arm swing of 90 degrees from horizontal to vertical position in the upward direction and then back to the horizontal position.	
2	Right arm is stretched away from the body to the right in a horizontal straight line and then attracted to the body again.	
3	Right arm is stretched away from the body in the upward direction in a vertical straight line and then attracted to the body again.	

Table 1: Test set

Test	Slave variants
1	1. Slave performs movement identical to master. 2. Slave deviates about 45° from the original swing direction. 3. Slave deviates about 90° from the original swing direction. 4. Slave starts about 0.5 seconds later. 5. Slave performs 45° of the swing.
2	1. Slave performs movement identical to master. 2. Slave deviates about 45° horizontally from the original direction. 3. Slave deviates about 90° horizontally from the original direction. 4. Slave starts about 0.5 seconds later. 5. Slave performs half the length of the stretch.
3	1. Slave performs movement identical to master. 2. Slave deviates about 45° horizontally from the original direction. 3. Slave deviates about 90° horizontally from the original direction. 4. Slave starts about 0.5 seconds later. 5. Slave performs half the length of the stretch.

Table 2: Slave variants

Variant	Method 1	Method 2	Method 3
1	Distance < 300 -2 > Delay < 2 Direction < 4 Magnitude < 4	Distance < 300 -2 > Delay < 2 Direction < 4 Magnitude < 4	Distance < 700 -2 > Delay < 2 Direction < 8 Magnitude < 8
2	-	Distance >= 300 -2 > Delay < 2 Direction >= 4 Magnitude < 4	-
3	-	Distance >= 400 -2 > Delay < 2 Direction >= 8 Magnitude < 4	Distance >= 700 -2 > Delay < 2 Direction >= 8 Magnitude < 8
4	-	Distance < 300 Delay < -2 Direction < 4 Magnitude < 4	Distance < 700 Delay < -2 Direction < 8 Magnitude < 8
5	-	Distance >= 300 -2 > Delay < 2 Direction >= 8 Magnitude < 4	Distance >= 700 -2 > Delay < 2 Direction >= 8 Magnitude < 8

Table 3: Expected Results

8.2 Test Results

In this section we present the results of the experiments we performed as specified in section 8.1. The test methods one and two are made with the author, and method three is made with the author and one volunteer. All the tests are made indoors in a room with good radio reception to prevent link losses corrupting the measurements. For method one all the tests are repeated three times, and for methods two and three the tests are repeated two times. The duration of every test is about 20 seconds, during which the movement tested is repeated continuously without rest.

All the feedback output is logged to a file which are then parsed through a classification system according the requirements of table 3. The occurences of good and faulty classifications are then counted which results in True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN) percentages. A classification is a TP when the movement is correct and is classified as correct, and a TN when the movement is incorrect and classified as incorrect. Similarly when the timing of a movement is correct and it is classified as correct, it is a TP. When the timing is incorrect and classified as incorrect it is a TN. A movement is a FN when it is classified as incorrect when in reality it was correct, and a FP when it is classified as correct while in fact it was incorrect.

Method one is the test with both the master and the slave node attached to one arm of the performer. Only test variation 1 applies for this test. Table 4 lists the result of this test. The result shows that under the most ideal conditions, the correct classification is very high.

Method 1	Distance		Delay		Direction		Magnitude	
Variant 1	TP	FN	TP	FN	TP	FN	TP	FN
Test 1.1	100%	0%	100%	0%	100%	0%	100%	0%
Test 1.2	100%	0%	100%	0%	100%	0%	100%	0%
Test 1.3	100%	0%	100%	0%	100%	0%	100%	0%
Test 2.1	100%	0%	100%	0%	100%	0%	100%	0%
Test 2.2	100%	0%	100%	0%	100%	0%	100%	0%
Test 2.3	100%	0%	100%	0%	100%	0%	95.3%	4.7%
Test 3.1	100%	0%	100%	0%	100%	0%	100%	0%
Test 3.2	100%	0%	95%	5%	100%	0%	100%	0%
Test 3.3	100%	0%	100%	0%	95.2%	4.8%	100%	0%

Table 4: Test results using Method 1 and Variant 1

With the second method, the master node is attached to the right wrist of the performer and the slave node to the left wrist. For this method all slave variations apply. All the tests are repeated twice under the same conditions as with method one. The results of this test are listed in the tables 5 to 9 for variations 1 to 5 respectively. The results for test types one and two are very good for all variations. The highest erroneous classification is the direction in test 1.2 of variant two, where only 70 percent of the movements were classified with high direction difference. This can be caused by a small difference in orientation of the sensor nodes, or the DTW algorithm found a more often found paths where the direction difference stayed low. The latter may also clearify the fact that the TP percentage of the magnitude is only 82.6%. The same reasoning may apply to the results of test 1.1 of variant 5, where

both the delay and the direction have high FN. A test which did not perform as expected for method 2, is test 2 for variation 2 and 3. This is caused by the fact that, altough the movements are made in a different direction, these are in fact not very different. Because the sensor nodes do not know their orientation, the direction is relative to the sensor node. As such, the master and slave node measure about the same direction, with only a low TN percentage for variation 2 caused by slight differences in how one performs the movements. Test 2 does perform as expected for the other variations.

Method 2	Distance		Delay		Direction		Magnitude	
Variant 1	TP	FN	TP	FN	TP	FN	TP	FN
Test 1.1	100%	0%	100%	0%	100%	0%	100%	0%
Test 1.2	95.2%	4.8%	100%	0%	100%	0%	100%	0%
Test 2.1	100%	0%	100%	0%	100%	0%	100%	0%
Test 2.2	95.2%	4.8%	100%	0%	95.2%	4.8%	100%	0%
Test 3.1	85.7%	14.3%	100%	0%	90.5%	9.5%	100%	0%
Test 3.2	95.2%	4.8%	95.2%	4.8%	95.2%	4.8%	100%	0%

Table 5: Test results using Method 2 and Variant 1

Method 2	Distance		Delay		Direction		Magnitude	
Variant 2	FP	TN	TP	FN	FP	TN	TP	FN
Test 1.1	8%	92%	100%	0%	16%	84%	84%	16%
Test 1.2	4.3%	95.7%	100%	0%	30.4%	69.6%	82.6%	17.4%
Test 2.1	54.2%	45.8%	95.8%	4.2%	91.7%	8.3%	91.7%	8.3%
Test 2.2	71.4%	28.6%	100%	0%	85.7%	14.3%	95.2%	4.8%
Test 3.1	9.5%	90.5%	95.2%	4.8%	0%	100%	90.5%	9.5%
Test 3.2	14.3%	85.7%	95.2%	4.8%	4.8%	95.2%	90.5%	9.5%

Table 6: Test results using Method 2 and Variant 2

Method 2	Distance		Delay		Direction		Magnitude	
Variant 3	FP	TN	TP	FN	FP	TN	TP	FN
Test 1.1	5.3%	94.7%	100%	0%	0%	100%	89.5%	10.5%
Test 1.2	4%	96%	100%	0%	0%	100%	92%	8%
Test 2.1	92.3%	7.7%	100%	0%	96.2%	3.8%	100%	0%
Test 2.2	79.3%	20.7%	100%	0%	100%	0%	93.1%	6.9%
Test 3.1	12.5%	87.5%	95.8%	4.2%	0%	100%	100%	0%
Test 3.2	14.3%	85.7%	95.2%	4.8%	4.8%	95.2%	95.2%	4.8%

Table 7: Test results using Method 2 and Variant 3

Method 2	Distance		Delay		Direction		Magnitude	
Variant 4	TP	FN	FP	TN	TP	FN	TP	FN
Test 1.1	96.2%	3.8%	3.8%	96.2%	88.5%	11.5%	88.5%	11.5%
Test 1.2	100%	0%	0%	100%	95.2%	4.8%	100%	0%
Test 2.1	100%	0%	15.8%	84.2%	100%	0%	94.7%	5.3%
Test 2.1	100%	0%	11.8%	88.2%	100%	0%	100%	0%
Test 3.1	94.4%	5.6%	0%	100%	94.4%	5.6%	72.2%	27.8%
Test 3.2	100%	0%	17.4%	82.6%	100%	0%	65.2%	34.8%

Table 8: Test results using Method 2 and Variant 4

Method 2	Distance		Delay		Direction		Magnitude	
Variant 5	FP	TN	TP	FN	TP	FN	FP	TN
Test 1.1	0%	100%	85.7%	14.3%	57.1%	42.9%	14.3%	85.7%
Test 2.2	12.5%	87.5%	79.2%	20.8%	70.8%	29.2%	8.3%	91.7%
Test 2.1	0%	100%	100%	0%	80%	20%	5%	95%
Test 2.2	0%	100%	100%	0%	61.1%	38.9%	16.7%	83.3%
Test 3.1	5.6%	94.4%	88.9%	11.1%	88.9%	11.1%	5.6%	94.4%
Test 3.2	11.1%	88.9%	55.6%	44.4%	94.4%	5.6%	0%	100%

Table 9: Test results using Method 2 and Variant 5

The first thing that was noticed when tests were made using two persons for method 3, is that the results of the DTW algorithm are totally different than the results of the one person test for method 2. The path distances produces by DTW for method 3 are much higher than for method 2. We therefore had to adjust the thresholds to compensate for this difference. Another problem proved to be the orientation of the sensor nodes, which must closely match for the direction difference detection to work properly. This is much harder to accomplish with two different persons. Therefore variant 2 is ommited from this test method.

Table 10 to 13 show the results for variant 1, 3, 4 and 5 respectively. Again test 2 shows poor detection rates for variant 3, 4 and 5. For variant 3 the reason is the same as for the corresponding test in method 2. All tests produced acceptable results for variant 1. Also test 3 had good results with variant 4, and test 1 with variant 5. All other results are in general very poor. We have noticed that the algorithm is too sensitive to subtle differences in movements, which show up with this tests between two persons. The distances therefore are much higher, and the thresholds need to be compensated for this. As a result, the algorithm becomes much less accurate, and only very clear movement differences are then detectable.

Method 3	Distance		Delay		Direction		Magnitude	
Variant 1	TP	FN	TP	FN	TP	FN	TP	FN
Test 1.1	75%	25%	100%	0%	100%	0%	83.3%	16.7%
Test 1.2	80%	20%	90%	10%	90%	10%	80%	20%
Test 2.1	77.7%	22.3%	92.6%	7.4%	96.3%	3.7%	59.3%	40.7%
Test 2.2	88%	4%	6%	4%	96%	4%	52%	48%
Test 3.1	95.6%	4.4%	100%	0%	100%	0%	82.6%	17.4%
Test 3.2	90.5%	9.5%	90.5%	9.5%	100%	0%	90.5%	9.5%

Table 10: Test results using Method 3 and Variant 1

Method 3	Distance		Delay		Direction		Magnitude	
Variant 3	FP	TN	TP	FN	FP	TN	TP	FN
Test 1.1	23%	77%	69.2%	30.8%	23.1%	76.9%	76.9%	23.1%
Test 1.2	4.5%	95.5%	31.8%	68.2%	13.6%	86.4%	50%	50%
Test 2.1	40.9%	59.1%	86.4%	13.6%	13.6%	86.4%	36.4%	63.6%
Test 2.2	29.2%	70.8%	79.2%	20.8%	54.2%	45.8%	12.5%	87.5%
Test 3.1	6.5%	93.5%	83.9%	16.1%	51.6%	48.4%	25.8%	74.2%
Test 3.2	5%	95%	70%	30%	15%	85%	5%	95%

Table 11: Test results using Method 3 and Variant 3

Method 3	Distance		Delay		Direction		Magnitude	
Variant 4	TP	FN	FP	TN	TP	FN	TP	FN
Test 1.1	30%	70%	40%	60%	15%	85%	30%	70%
Test 2.1	41%	59%	74.6%	25.6%	100%	0%	56.4%	43.6%
Test 2.2	70.2%	29.8%	57.4%	42.6%	100%	0%	70.2%	29.8%
Test 3.1	100%	0%	30.4%	69.6%	100%	0%	100%	0%
Test 3.2	100%	0%	19.2%	80.8%	100%	0%	100%	0%

Table 12: Test results using Method 3 and Variant 4

Method 3	Distance		Delay		Direction		Magnitude	
Variant 5	FP	TN	TP	FN	TP	FN	FP	TN
Test 1.1	17.4%	82.6%	91.3%	8.7%	87%	13%	13%	87%
Test 1.2	20%	80%	92%	8%	96%	4%	8%	92%
Test 2.1	83.7%	16.3%	95.3%	4.7%	96.7%	2.3%	88.4%	11.6%
Test 2.2	91.7%	8.3%	97.2%	2.8%	100%	0%	100%	0%
Test 3.1	92%	8%	96%	4%	100%	0%	64%	36%
Test 3.2	66.7%	33.3%	83.3%	16.7%	100%	0%	44.4%	55.6%

Table 13: Test results using Method 3 and Variant 5

9 Conclusion

In this thesis we described Sport Coach - Online activity matching using wireless sensor network, in which we present a system to compare the movements of two persons online, in a real-time manner using inertial sensors. The comparison should identify difference in the direction and magnitude of movements as well as the earlyness or lateness of movements. The goal is to have a autonomic system that gives feedback to the user regarding his or her movement. We choose to seperate the concerns for the nodes for the trainer and the nodes for the trainee by designing them as master and slave node respectively. We further simplify the problem by only considering one master and one slave node.

The main problems which this project addresses are the acquisition of movement data, preparing the movement data for processing, communicating the data wirelessly to the slave node and measuring the differences between the movement data. This system is implemented on SunSpots which are wireless sensor nodes running Java. The challenges of this project are to find one or more algorithms to measure the temporal and spatial difference in performed movements and the implementation of such a system on real hardware. Furthermore the response time of the system should be within two seconds.

Our solution first filters the data to remove noise and extract the gravity and the real movement vector from the accelerometer data. We choose to use a Kalman filter to improve the estimate of the gravity vector. Using a segmentation algorithm we identify start and end of movements and also direction changes. This information is then sent wirelessly from the master node to the slave node where the movement data is compared. We present Fast Incremental Dynamic Time Warping (FIDTW) a modification of the traditional Dynamic Time Warping (DTW) algorithm, which can run online on resource constrained sensor nodes. The FIDTW algorithm compares the movement data using the magnitude and the direction of the movement, and finds the temporal and spatial best match of the two movements. This information is then analysed from which conclusion about the timing, the differences of the movement and the cause of the difference is drawn.

In this section we conclude our work by first giving a summary of the design choices in section 9.1 and a summary of our evaluation method in section 9.2, after which we reflect on them in the discussion in section 9.3. This chapter finishes with recomendations for future work in section 9.4.

9.1 Summary of Design Choices

The hardware used for this project is Sun SPOT, which are wireless sensor nodes running the Java virtual machine. By default, the SunSpots are equiped with only a three dimensional accelerometer. We have broaden the the degrees of freedom that can be measured by the Sun SPOT by adding a two dimensional gyroscope sensor and a compass sensor. We however choose to only use the accelerometer and the gyroscope to prevent introducing too much complexity.

The accelerometer and the gyroscope are both analog sensors which need to be converted to a digital representation of the measurement. The Sun SPOT library incorporates a library for reading the accelerometer and also for reading the analog gyroscope. We choose to oversample the sensors at 40Hz to improve the removal of noise in the filter stage.

The measurements from the accelerometer and the gyroscope in their raw state are contaminated with noise from the sensor, measurement and the conversion processes. This noise is largely removed by applying a low pass filter to the sensor sample data as explained in chapter 4. We choose to extract the gravity vector from the accelerometer data such that by subtracting the gravity vector from the accelerometer data will give the real movement vector. The gravity vector is obtained by applying a low pass filter to the accelerometer sensor data and is further improved with gyroscope data using a Kalman filter. After filtering, the sample rate is reduced to 20Hz.

We choose to use a steady state Kalman filter to improve the estimation of the gravity vector which is described in section 4.2. An angular system model is used for the Kalman filter, which results in the need for efficient trigonometric functions. The approach of Jasper Vijn [39] is used for the cosine and sine approximations. Because also the arc sine and arc cosine are needed, we made our own approximation based on Jasper Vijn 's work. With our choice for the angular system model, we also need to convert the gravity vector from the low pass filter to an angular positon representation. Unfortunately, this was more difficult than at first anticipated, because it is impossible to obtain angular positions from only one vector. We therefore choose to make an approximation by using the knowledge of the two dimensions of angular rate from the gyroscope data to find the other dimension of the angular rate from the gravity vector. By applying this process also for the other two dimensions of angular rate, a full three dimensional angular rate representation of the gravity vector is approximated.

In chapter 5, we describe our approach to apply segmentation to the data stream of sensor data. By measuring the average of the magnitude of the movement vector and the gyroscope rates, a distinction between movement and reste can be made. The changes in direction are measured between the current data and multiple data from history. From these detection methods, segmentation events are generated which are sent with the movement data to the slave node.

The communication between the master node and the slave node is explained in chapter 7.2.5. Reliable communication is used to transfer the movement data to the slave node, because we consider only one node and one slave. However, our solution is easily adapted to allow multiple slave nodes on one master, because the similarity measurement is performed on the slave nodes. This would require the use of broadcasting at the master side because otherwise the master node would be too concerned with transmitting packets and the network would get easily congested.

We choose to use Dynamic Time Warping (DTW) to measure the similarity between two movements. DTW can successfully measure similarity even when two sequences are different in length and rate, which makes it ideal to measure similarity of movements which maybe stetched or delayed, when it is known that the sample rate is equal. A disadvantage of DTW is that it requires many computational resources. However, we have made an optimization to DTW which we present in chapter 6. The optimization makes the DTW algorithm incremental, such that the required computations are greatly reduced. This optimized approach makes it possible to use DTW online on wireless sensor nodes. We choose to use the direction and magnitude data from the movement to compare using the FIDTW algorithm. Unfortunately we found that, because the direction data is relative to the node 's orientation, it will sometimes result in faulty results because the orientation of the slave and

the master node was different. Any attempt to compensate for this orientation difference has failed because there is no orientation information available to compensate with.

9.2 Summary of Evaluation

In chapter 8 we evaluated the total system using experiments. Using three different test methods we test how the system performs in different circumstances. The first method is a ground truth test to test whether the system correctly classifies all movements as correct. With the second test method the master and slave nodes are each attached to the two wrists of one person. With this method we test whether the system correctly classifies the differences in movements of the master and slave node. In a third method we test whether the system can also identify and classify the differences of movements of two persons.

Three different test types are introduced which define how the movement must be made. These types are an upward arm swing, a arm stretch in front, and an arm stretch upward. Then, for every test type, there are five variants of movements for the slave. For variant 1 of the test type the movements are identical for master and slave. With variant 2 the slave deviates 45 degrees in movement direction and with variant 3 the slave deviates 90 degrees in movement direction. Variant 4 tests whether delays are correctly measured by letting the slave lag his movement for about half a second. With variant 5 the slave performs half the movement, meaning it either performs only 45 degrees of the arm swing of test type 1 or the arm is stretched only half way for test type 2 and 3.

9.3 Discussion

In this thesis, we performed experiments with our approaches at numerous points. In section 4.2.4 we evaluated the use of the Kalman filter for improving the gravity estimation and our segmentation approach is evaluated in section 5.3. Furthermore, we made real world tests using predefined movements to evaluate the total system in chapter 8. In this discussion we reflect on these results.

In section 4.2.4 we evaluated the performance of the gravity detection approach with a series of experiments. The results of this experiments showed that the Kalman filter gives a small improvement of the gravity estimation compared with only a low pass filter. The Kalman filter adds slightly more speed, and reduces the overshoot measured by the low pass filter. However, the correction that is applied by the Kalman filter is kept low to prevent corruption of the gravity estimation caused by the errors which are present in the approximation of the angular rates. These errors occur when the gravity vector from the low pass filter is close to a singularity in the euler angle coordinate system. An example of such an error can be seen in figure 5 in the Y axis of the improved gravity vector, just at the end of the last swing movement. These singularity errors can be prevented by using quaternions instead of the euler coordinate system, which are immune to such singularities. The cost of this improvement is that the system becomes more complex.

We also evaluated our segmentation approach in chapter 5 with experiments. The results show that movement can reliably be discriminated from rest, and that changes in movement direction can successfully be detected. A shortcomming of this approach is that slower changes of direction, such as a circular rotation, are easily missed by this method. Extending the history with which the current direction is compared might solve this problem, but rotational movements where only the centrifical force is precent will never be detected by

this algorithm. To solve this problem the gyroscope can be used to detect such rotations. In this thesis we do not consider pure rotational movements, so we did not try this approach.

In chapter 6 we presented our approach of measuring similarity and differences between the movements of the trainer and the trainee. The approach is evaluated with experiments using arm movements of one person. We have chosen to use the difference in magnitude and direction of the movement data as input for the FIDTW algorithm. The results of the experiments show that the FIDTW algorithm successfully measures latency and differences in direction and magnitude. A shortcomming of using the direction information is that it is dependent on the orientation of the sensor node. This means that during the experiments the orientation of the slave and master node must very closely match. This is a very undesirable effect, but at this moment we have no other method of detecting differences in direction, and there are no means to correct this because the sensor node is oblivious to it's orientation.

We have performed ground truth test and realistic tests in chapter 8. The ground truth tests show that the algorithm correctly classifies the movement as correct. This test method is not representative for real world situations, and therefore two other test methods are used to evaluate the system. With the second test method, both the master and slave node are attached to the wrists of one person, and with the third test method, the nodes are attached to one of the wrists of two different persons.

The tests with the first method show very good results as expected. The tests with the second method also show good results, with the exception of a few test variants. These exceptions were variant 2 and 3 for test 2, caused by the fact that direction differences are undetectable by the sensor nodes with this test. When the orientation of the sensor node would be known, this problem could be solved. However, obtaining an orientation is not easily accomplished.

The tests with method 3 are not as good as expected. We found that the algorithm is over sensitive to small differences in how one performs a movement. Between the left and right arm these differences are much smaller, such that the algorithm did perform very well with method 2. At this moment we do not have a solution to this problem, but a solution may be found using different distance measure or using the gyroscope to aid or replace the direction difference detection. The problem that only direction differences with respect to the orientation of the node can be detected remains unsolved. The only solution to this problem is to know the absolute orientation of the sensor node. This could be derived with a three dimensional compas together with the gravity vector and may be improved with a three dimensional gyroscope. Currently however, the Sun SPOT are not able to measure all degrees of freedom, such that the orientation can not be derived with the current set of sensors.

9.4 Future work and recommendations

The most important open issue is to make the algorithm work reliable when detecting movement differences between two persons. This is partly caused by the orientation issue, but mostly because the algorithm is over sensitive to subtle movement differences. The solution to this problem may be found in other distance measures and other sensors.

A second open issue is the orientation problem. Due to this problem, the sensor orientation must closely match when attached to the wrist, but also during movements the orientation must closely match. This makes testing with two persons hard and more unreliable. The

problem maybe solved by using different sensors that allow to measure all degrees of freedom, such that the orientation can be derived. This is not an easy task and may not work reliable or fast enough to be used for this application.

The gravity detection may be further improved by using quaternions instead of euler angles, such that singularities are avoided. This will make the Kalman filter more complex, but will improve the reliability of the estimation.

Finally we recommend to not rely on tests where one uses both hands to simulate two person tests. Even when an attempt is made to fool the system with small differences in an identical movement, the results are still much better than with a two person tests.

References

[1] *Bayesian activity recognition in residence for elders*, 2007.

[2] *Wireless Medical Sensor Networks in Emergency Response: Implementation and Pilot Results*, 2008.

[3] T. Arici and Y. Altunbasak. Adaptive sensing for environment monitoring using wireless sensor networks. In *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, volume 4, pages 2347--2352, 21-25 2004.

[4] Ira Assent, Marc Wichterich, Ralph Krieger, Hardy Kremer, and Thomas Seidl. Anticipatory dtw for efficient similarity search in time series databases. *Proc. VLDB Endow.*, 2(1):826--837, 2009.

[5] Ryan Aylward. Sensemble: A wireless, compact, multi-user sensor system for interactive dance. In *Proc. of NIME 06*, pages 134--139, 2006.

[6] Ruzena Bajcsy, Alessandro Borri, Maria Domenica Di Benedetto, Annarita Giani, and Claire Tomlin. Classification of physical interactions between two subjects. *Wearable and Implantable Body Sensor Networks, International Workshop on*, 0:187--192, 2009.

[7] Chris R. Baker, Kenneth Armijo, Simon Belka, Merwan Benhabib, Vikas Bhargava, Nathan Burkhart, Artin Der Minassians, Gunes Dervisoglu, Lilia Gutnik, M. Brent Haick, Christine Ho, Mike Koplow, Jennifer Mangold, Stefanie Robinson, Matt Rosa, Miclas Schwartz, Christo Sims, Hanns Stoffregen, Andrew Waterbury, Eli S. Leland, Trevor Pering, and Paul K. Wright. Wireless sensor networks for home health care. In *AINAW '07: Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops*, pages 832--837, Washington, DC, USA, 2007. IEEE Computer Society.

[8] Richard Ernest Bellman. *Dynamic Programming*. Dover Publications, Incorporated, 2003.

[9] Graeme S. Chambers, Svetha Venkatesh, Geoff A. W. West, and Hung H. Bui. Segmentation of intentional human gestures for sports video annotation. In *MMM '04: Proceedings of the 10th International Multimedia Modelling Conference*, page 124, Washington, DC, USA, 2004. IEEE Computer Society.

[10] Datong Chen, Robert Malkin, and Jie Yang. Multimodal detection of human interaction events in a nursing home environment. In *ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces*, pages 82--89, New York, NY, USA, 2004. ACM.

[11] Diane J. Cook, Aaron Crandall, Geetika Singla, and Brian Thomas. Detection of social interaction in smart spaces. *Cybern. Syst.*, 41(2):90--104, 2010.

[12] Simon Dixon. An on-line time warping algorithm for tracking musical performances. In *IJCAI'05: Proceedings of the 19th international joint conference on Artificial intelligence*, pages 1727--1728, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[13] I.A. Essa. Ubiquitous sensing for smart and aware environments. *Personal Communications, IEEE*, 7(5):47 --49, oct 2000.

[14] Davrondzhon Gafurov and Einar Snekkenes. Gait recognition using wearable motion recording sensors. *EURASIP J. Adv. Signal Process*, 2009:1--16, 2009.

[15] H. Ghasemzadeh, R. Jafari, and B. Prabhakaran. A body sensor network with electromyogram and inertial sensors: Multimodal interpretation of muscular activities. *Information Technology in Biomedicine, IEEE Transactions on*, 14(2):198--206, march 2010.

[16] Eric Guenterberg, Ruzena Bajcsy, Hassan Ghasemezadeh, and Roozbeh Jafari. A segmentation technique based on standard deviation in body sensor networks. In *Dallas-EMBS '07: Proceedings of the IEEE Dallas Engineering in Medicine and Biology Workshop*. IEEE, 2007.

[17] Eric Guenterberg, Sarah Ostadabbas, Hassan Ghasemzadeh, and Roozbeh Jafari. An automatic segmentation technique in body sensor networks based on signal energy. In *BodyNets '09: Proceedings of the Fourth International Conference on Body Area Networks*, pages 1--7, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[18] S. Haltsonen. Improved dynamic time warping methods for discrete utterance recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 33(2):449 -- 450, apr 1985.

[19] Hongwei Huo, Youzhi Xu, Hairong Yan, Saad Mubeen, and Hongke Zhang. An elderly health care system using wireless sensor networks at home. *Sensor Technologies and Applications, International Conference on*, 0:158--163, 2009.

[20] Roozbeh Jafari, Wenchao Li, Ruzena Bajcsy, Steven Glaser, and Shankar Sastry. Physical activity monitoring for assisted living at home. In *BSN '07: Proceedings of the 4th International Workshop on Wearable and Implantable Body Sensor Networks*, pages 213--219, Berlin, Heidelberg, 2007. Springer.

[21] V.M. Jones, R. Huis in 't Veld, T. Tonis, R.G.A. Bults, B. Beijnum van, I. Widya, M. Vollenbroek-Hutten, and H. Hermens. Biosignal and context monitoring: Distributed multimedia applications of body area networks in healthcare. In *Proceedings IEEE 10th Workshop on Multimedia Signal Processing, 2008*, pages 820--825, Cairns, Australia, October 2008. IEEE Signal Processing Society.

[22] Sukun Kim, Shamim Pakzad, David Culler, James Demmel, Gregory Fenves, Steve Glaser, and Martin Turon. Wireless sensor networks for structural health monitoring. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 427--428, New York, NY, USA, 2006. ACM.

[23] Edgar Kraft. A quaternion-based unscented kalman filter for orientation tracking. In *Proceedings of the Sixth International Conference on Information Fusion*, volume 1, pages 47--54, 2003.

[24] Peter Leijdekkers and Valérie Gay. A self-test to detect a heart attack using a mobile phone and wearable sensors. In *CBMS '08: Proceedings of the 2008 21st IEEE International Symposium on Computer-Based Medical Systems*, pages 93--98, Washington, DC, USA, 2008. IEEE Computer Society.

[25] Jonathan Lester, Blake Hannaford, and Gaetano Borriello. Are you with me?" – using accelerometers to determine if two devices are carried by the same person. In *In Proceedings of Second International Conference on Pervasive Computing (Pervasive 2004*, pages 33--50, 2004.

[26] Guiling Li, Yuanzhen Wang, Min Li, and Zongda Wu. Similarity match in time series streams under dynamic time warping distance. In *CSSE '08: Proceedings of the 2008 International Conference on Computer Science and Software Engineering*, pages 399--402, Washington, DC, USA, 2008. IEEE Computer Society.

[27] Lin Liao, Dieter Fox, and Henry Kautz. Location-based activity recognition using relational markov networks. In *IJCAI'05: Proceedings of the 19th international joint conference on Artificial intelligence*, pages 773--778, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[28] Benny P. L. Lo, Surapa Thiemjarus, Rachel King, and Guang zhong Yang. Body sensor network - a wireless sensor platform for pervasive healthcare monitoring. In *Adjunct Proceedings of the 3rd International conference on Pervasive Computing (PERVASIVE'05*, pages 77--80, 2005.

[29] Raluca Marin-Perianu, Mihai Marin-Perianu, Paul Havinga, and Hans Scholten. Movement-based group awareness with wireless sensor networks. In *PERVASIVE'07: Proceedings of the 5th international conference on Pervasive computing*, pages 298--315, Berlin, Heidelberg, 2007. Springer-Verlag.

[30] Yuji OHGI. Mems sensor application for the motion analysis in sports science. *ABCM Symposium Series in Mechatronics*, 2:501--508, 2006.

[31] Shyamal Patel, Chiara Mancinelli, Jennifer Healey, Marilyn Moy, and Paolo Bonato. Using wearable sensors to monitor physical activities of patients with copd: A comparison of classifier performance. *Wearable and Implantable Body Sensor Networks, International Workshop on*, 0:234--239, 2009.

[32] Donald J. Patterson, Dieter Fox, Henry Kautz, and Matthai Philipose. Fine-grained activity recognition by aggregating abstract object usage. In *ISWC '05: Proceedings of the Ninth IEEE International Symposium on Wearable Computers*, pages 44--51, Washington, DC, USA, 2005. IEEE Computer Society.

[33] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L. Littman. Activity recognition from accelerometer data. In *IAAI'05: Proceedings of the 17th conference on Innovative applications of artificial intelligence*, pages 1541--1546. AAAI Press, 2005.

[34] Hiroki Saito, Takashi Watanabe, and Achmad Arifin. Ankle and knee joint angle measurements during gait with wearable sensor system for rehabilitation. In *World Congress on Medical Physics and Biomedical Engineering*, pages 506--509, Berlin, Heidelberg, 2009. Springer.

[35] Yasushi Sakurai, Masatoshi Yoshikawa, and Christos Faloutsos. Ftw: fast similarity search under the time warping distance. In *PODS*, pages 326--337, 2005.

[36] S. Salvadore and P. Chan. Fastdtw: Toward accurate dynamic time warping in linear time and space. In *3rd Workshop on Mining Temporal and Sequential Data*, 2004.

[37] Christopher P. Townsend, Michael J. Hamel, and Steven W. Arms. Telemetered sensors for dynamic activity and structural performance monitoring. volume 4334, pages 228--233. SPIE, 2001.

[38] Du Tran and Alexander Sorokin. Human activity recognition with metric learning. In *ECCV '08: Proceedings of the 10th European Conference on Computer Vision*, pages 548--561, Berlin, Heidelberg, 2008. Springer-Verlag.

[39] Jasper Vijn. Another fast fixed-point sine approximation, 2009.

[40] Eric W Weisstein. Euler angles, 2010. [Online; accessed 11-June-2010].

[41] Eric W Weisstein. Nonlinear least squares fitting, 2010. [Online; accessed 11-June-2010].

[42] Anthony Whitehead and Kaitlyn Fox. Device agnostic 3d gesture recognition using hidden markov models. In *Future Play '09: Proceedings of the 2009 Conference on Future Play on @ GDC Canada*, pages 29--30, New York, NY, USA, 2009. ACM.

[43] Wikipedia. Finite impulse response --- wikipedia, the free encyclopedia, 2010. [Online; accessed 11-June-2010].

[44] Wikipedia. Nyquist–shannon sampling theorem --- wikipedia, the free encyclopedia, 2010. [Online; accessed 31-May-2010].

[45] Wikipedia. Oversampling --- wikipedia, the free encyclopedia, 2010. [Online; accessed 31-May-2010].

[46] Martin Wirz, Daniel Roggen, and Gerhard Troster. Decentralized detection of group formations from wearable acceleration sensors. *Computational Science and Engineering, IEEE International Conference on*, 4:952--959, 2009.

[47] Jianxin Wu, Adebola Osuntogun, Tanzeem Choudhury, Matthai Philipose, and James M. Rehg. A scalable approach to activity recognition based on object use. In *In Proceedings of the International Conference on Computer Vision (ICCV), Rio de*, 2007.

[48] Byoung-Kee Yi, H. V. Jagadish, and Christos Faloutsos. Efficient retrieval of similar time sequences under time warping. In *ICDE '98: Proceedings of the Fourteenth International Conference on Data Engineering*, pages 201--208, Washington, DC, USA, 1998. IEEE Computer Society.

[49] Xiaoming Yin and Ming Xie. Finger identification and hand posture recognition for human-robot interaction. *Image and Vision Computing*, 25(8):1291 -- 1300, 2007.

[50] Xiaoping Yun, M. Lizarraga, E.R. Bachmann, and R.B. McGhee. An improved quaternion-based kalman filter for real-time tracking of rigid body orientation. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, pages 1074--1079, oct. 2003.

List of Figures

List of Tables