Classification of Error Types in Physiotherapy Exercises

With Quaternions

Haakon Drews

Thesis submitted for the degree of
Master in Robotics and intelligent systems
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

2nd May 2017

Classification of Error Types in Physiotherapy Exercises

With Quaternions

Haakon Drews

Abstract

Systems of healthcare and the field of physical therapy in particular are likely to come under increased stress in the decades to come as populations age faster than their workforces can handle. Much of healthcare is dependent on skilled human effort. Intelligent systems could be developed to support the efforts of healthcare specialists and enable patients and users to continue effective treatment with less human supervision, helping healthcare systems to adapt to the change in demographics and their associated rising costs.

This thesis aims to work towards that goal by investigating methods of detecting mistakes made during physiotherapy exercises, which would be a strictly necessary component in any intelligent system that is able to give feedback to a user about her performance. Special consideration is given to the requirements of systems which must function in a home-environment without specialist supervision. For the purposes of this thesis a dataset is built by recording a set of exercises with the help from volunteer test subjects. The dataset is then used to develop a model for classifying exercises and mistakes made during the exercises based on discrete-time quaternion sequences, after which the model is tested using several validation schemes.

The model manages to successfully predict exercises across test subjects and error types with high rates of accuracy given the validation schemes used. However, when classifying error types across subjects, the accuracy proves unsatisfactory. Error types are classified with high rates of accuracy when using template exercises for a given test subject. These results highlight important limitations of the proposed model.

Contents

List of Figures

List of Tables

Glossary

API application programming interface.

DoF degree of freedom.

DTW Dynamic Time Warping.

EMG electromyography.

IMU intertial measurement unit.

k-NN k-nearest neighbor.

offline algorithm is an algorithm which is given its complete input data immediately when it is run and produces a result based on this. This is the opposite of an ..

PCA principal component analysis.

SDK software development kit.

Acknowledgements

I would like to thank my supervisor, Jim Tørresen, for his help and endless patience with this thesis. I would also like to thank Marit Nygård Olsen and Elise Klæbo Vonstad from Sunnaas Hospital for their gracious support and help with all the parts of this thesis related to the topics of physiotherapy and for making available to me the hardware sensors needed for the thesis.

Finally, I would like to thank all my friends, family and loved ones who supported me during the thesis.

Part I

Introduction

Chapter 1

Introduction

1.1 Introduction

Intelligent systems are increasingly being used successfully in many areas of society. These range from fields such as finance and manufacturing, to automating work in warehouses and driving our cars, not to mention helping us find things on the internet. While our technology seems to steadily improve often making human labor obsolete, one area of society which sees a growing demand on specialized human labor is that of healthcare. Many countries in the world, especially in the west, are experiencing a growing demographic of the aging and elderly and a decreasing population of able young people trained to take care of them.

Increasingly, large-scale efforts are being made to use big-data and machine learning to diagnose medical conditions across large groups of people. Meanwhile, intelligent and reliable solutions for effective physiotherapy treatments alleviating the need for trained specialist supervision remain elusive. This thesis aims to work towards getting closer to being able to create such solutions.

1.2 Motivation

The change in demographics manifesting in Europe [29] and the resulting change in workforce demographics may pose an important and tricky problem to solve in the future. As populations grow older, the relative size of the workforce diminishes. This may lead to less taxes being collected and relatively fewer employees in the healthcare sector. This shift in demographics could thereby cause costs to increase dramatically [23]. A part of the strategy to solve these complex challenges might lie in using systems which are intelligent or automated to some degree. This might ease the burden on human specialists when it comes to the treatment of conditions which require physical therapy. Such systems might not just be used after the fact for treatment of existing conditions, but also as a preventative measure to stave off the loss of daily functioning.

In the fields of physical therapy and rehabilitation, exercises are often prescribed to patients to restore or improve motor functioning [20].

Exercises consist of performing specific movement sequences a certain number of times. These exercises should be performed correctly to advance the treatment. Patients can be instructed by therapists or by using brochures [9] as shown in figure 1.1. Using brochures, instead of receiving in-person supervision, patients may be less likely to correctly perform their exercises. This way, the use of brochures may lead to a decreased rate of treatment effectiveness compared to being supervised by a physiotherapist [9]. Additionally, patients receiving supervision in a clinical setting often show greater treatment effectiveness than those in a home setting, which provides for less supervision from therapists [20]. Adherence prescribed exercise plans is often lacking in a home environment [3]. Due to the difference in treatment effectiveness between supervised and unsupervised exercise regimens, patients learning from brochures or performing exercises in a home environment may undergo longer overall treatment. In other words, the real-time feedback and followup patients receive from specialists during a supervised exercise seems to be a crucial factor for the efficacy of an exercise and the treatment overall.

Automated physical rehabilitation monitoring systems could provide useful real-time feedback to patients at locations or times where specialists are not available. These systems could improve recovery time for exercise sessions where no specialist is availble to supervise. Also, due to the resulting higher efficacy of unattended sessions, more sessions could be performed at home overall. Potentially, this may reduce the need for specialist post-checkout support, helping to alleviate the danger of losing treatment progress when a patient leaves the hospital.

1.3 Goals

An exercise-feedback system as described above would need to include the following parts:

- Suitable sensors for capturing motion. Factors determining suitability might include ease of use from the patient's perspective, affordability, ease of integration with existing systems and the type of data the sensors provide.
- A sensor data pipeline which uses the sensors to capture exercise motion data, handling enumeration of sensors and if necessary conversion or processing of data formats. The pipeline must then either stream this motion data to be processed immediately or must save it for later offline use.
- An analysis module containing algorithms which analyze the sensor data provided by the software pipeline in order to evaluate the correctness exercises.

In the real world, a system like this should ideally provide feedback to the user as quickly as possible as errors are being detected. However,

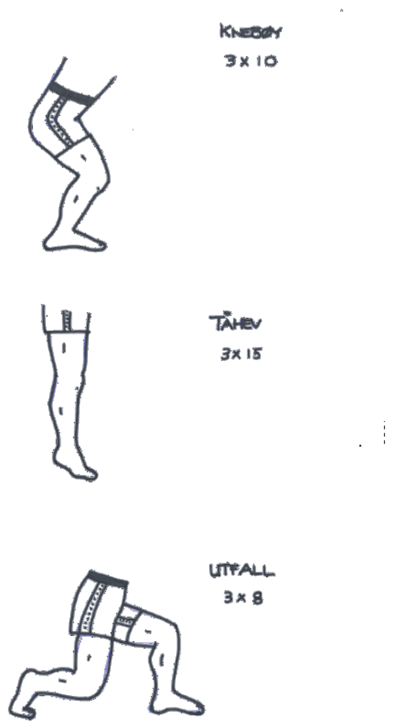


Figure 1.1: Example of a brochure instructable for an exercise. Reproduced from material provided by from Sunnaas Hospital.

real-time implementations of machine learning algorithms can be hard to achieve in their own right, due to both computational and design challenges. Therefore, the specific focus of this thesis with regards to machine learning will be on adapting offline machine learning techniques to rating the correctness and error type for physical therapy exercises. The context and ultimate motivation behind the more narrowly defined goals of this thesis should however be kept in mind throughout. In order to create models that are able to successfully classify errors in exercises, these models would have to be trained. For this purpose, a dataset would need to be produced.

Thus, the actual goals of this thesis can be summarized as follows:

- Evaluating and selecting suitable motion capture sensors.
- Implementing a software pipeline to interface with the sensors and allow real-time streaming or saving of sensor data.
- Designing an exercise program consisting of typical exercises used for physical therapy to use with the system.
- Designing and then executing a recording session protocol in order to obtain testing and training data from test subjects performing the exercise program.

- Implementing machine learning techniques to classify exercises and error types recorded during the recording sessions.
- Train and test the algorithms using suitable validation schemes.

1.4 Outline of thesis

This thesis is split into three parts: an introduction, the implementation and the conclusion, and consists of a total of 6 chapters, followed by an appendix.

- The first part contains two chapters, the introduction and the background. In the introduction, the thesis' goals and the motivations for them are described. The background chapter presents some relevant information on the topics and techniques used later. It also contains a brief presentation of previous work relevant to the thesis.
- The second part contains two chapters, the implementation and the results. The first chapter details the process of planning and implementing the work of this thesis. This includes the building of a dataset and implementing the necessary software and algorithms. The second chapter presents the results obtained and the validation schemes which were employed.
- The last part contains two chapters, the first discussing and interpreting the results obtained, the second containing a conclusion to the thesis and thoughts on possible future work.

Chapter 2

Background

2.1 Sensors and motion capture

Many different kinds of sensors exist, however in this thesis the focus is on sensors which can help us record useful information on motions of the human body in time in general and while performing physical exercises in particular. To give context for the choice of sensors discussed later, the following sections describe relevant types of sensors are along with examples of their typical uses.

2.1.1 Kinetic sensors

Kinetics is the study of forces between moving bodies and is concerned with forces and torques. The category of kinetic sensors includes pressure and torque sensors. Pressure sensors measure the force applied to a point or over an area. Pressure sensors arranged in grids (see figure 2.1) have been used to explore the value and possible correlates in measuring pressure data from footprints for gait analysis [26]. They can also be integrated into treadmills which have been used to calculate the path of the body's center of mass during walking as a way to improve the modeling of human gait patterns [25].

A dynamometer is a force sensor measuring torque. They have previously been used to measure muscle strength data to establish normative values for developing and evaluating rehabilitation programs [12]. To increase the usability of pressure sensors outside of laboratory conditions, efforts are made to develop smaller and less restricting force sensing technologies. In [4] a sensor was developed based on the resistive properties of material used, with the goal of being able to integrate the sensor into clothing. The force exerted on the tensile part of the sensor strip is measured by reading the variable resistance of the sensor material as it is stretched and relaxed (see figure 2.2)

2.1.2 Kinematic sensors

Kinematics concerns itself with the study of the motions of bodies (speed, position) without considering the causes of said motions (forces, torques).

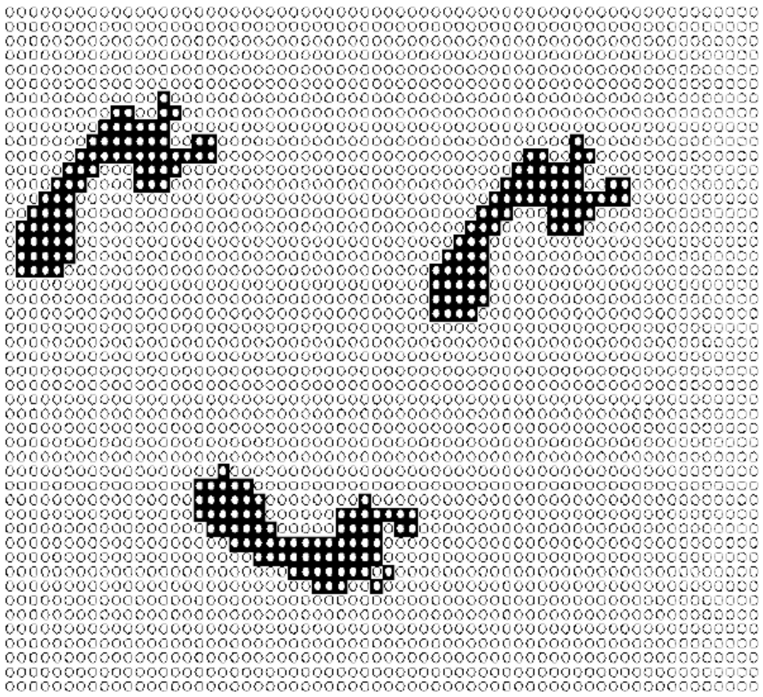


Figure 2.1: Illustration of the pressure points in the Gaitrite pressure sensor system. Individual squares represent single pressure sensors arranged in a grid. [15]

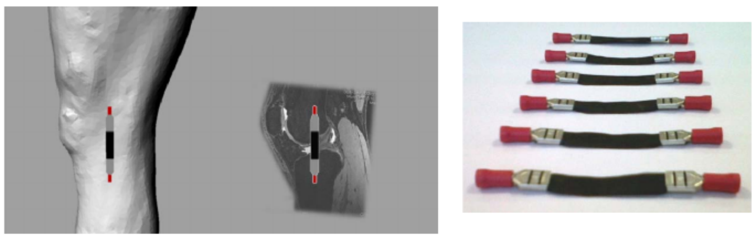


Figure 2.2: Force sensor strips designed to measure torque during knee flexion to use for calculating the knee angle [4]. Example of placement (left), and actual sensor strips (right)

This class of sensors includes IMU and optical motion capture solutions.

Optical motion capturing works by using one or more cameras to observe the subject we want to quantify the kinematics of and processing the data using image analysis techniques.

IMUs are sensors using a combination of a gyroscope, accelerometer and possibly a magnetometer to measure the sensors angular rate, axis acceleration and the local magnetic field respectively. Together, the data from these components may be used to derive the sensor's orientation in space using a technique called sensor fusion. A common algorithm for performing sensor fusion is the Kalman filter.

This thesis focuses on using IMU sensors to measure the kinematics of test subjects during exercises. Therefore, the relevant sensors and their properties are briefly described bellow.

2.1.3 The Myo sensor armband

One of the sensors to be used is the Myo sensor armband developed by Thalmic Labs Inc., Canada. The Myo is a relatively affordable sensor device which at the time of writing is sold at a price of 199 $ on the official Myo store website [6]. The main intended use of the Myo is controlling digital presentations (ie. Powerpoint), control of computer applications (such as media control) and as a controller not unlike a joystick for gaming. It is worn on the upper half of forearm, either right or left.

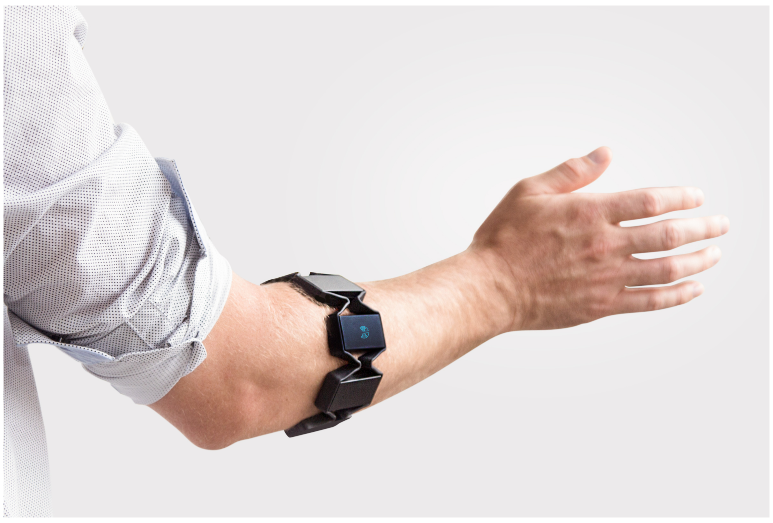


Figure 2.3: Myo sensor armband worn on the right forearm. [19]

The Myo is an armband comprised of 8 "pods", see figure 2.4. The sensors integrated in the armband are a 9 degrees of freedom (DoFs) IMU containing a 3 DoF gyroscope, a 3 DoF accelerometer and a 3 DoF magnetometer in addition to 8 EMG sensors. The EMG sensor contacts are located on the inside of the armband on each of the pods. The Myo is wireless, battery-driven and contains no internal storage except for the firmware and associated settings. The device is charged via a micro-

USB port. Sensor data can only be streamed in real-time via the wireless standard bluetooth 4.0 Low Energy protocol [5]. The default sampling rate of the IMU is 50Hz and the sampling rate of the EMG sensors is 200Hz. The types of data than can be transmitted to a receiver are the vectors of the acceleration, gyroscope and magnetometer sensors as well as quaternions derived from said data, in addition to the sensor data from each EMG pod. The device has integrated hardware support for classification of several hard coded gestures consisting of a combination of arm movements and/or finger movements. To the knowledge of the author, no complete datasheet containing the technical specifications of the device are available at the time of writing. All technical specifications available are either listed on the products website or in the documentation of the official SDK and the bluetooth protocol header file [18] released by Thalmic Labs Inc. Thalmic Labs Inc. provides Software development kit (SDK) for the Windows, Mac OS, Android and iOS operating systems. Various third-party libraries exist for interfacing with other platforms and languages such as Linux or Javascript. Additionally, the Myo uses the bluetooth GATT protocol [10] and Thalmic Labs Inc. have published their specific control API for the sensor [18]. Thus independent development of software for device configuration and data gathering is easier compared to devices using their own proprietary or closed communication protocols. The sensor's firmware can be updated via the USB port.

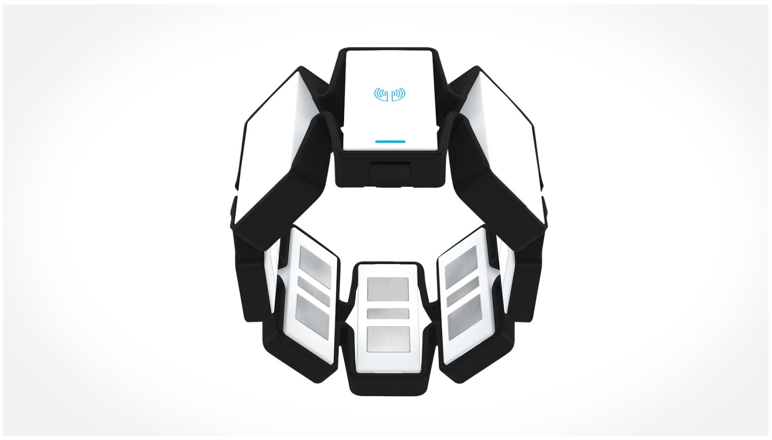


Figure 2.4: Myo sensor armband. The band contains 8 EMG sensors, a gyroscope, an accelerometer and a magnetometer. [19]

2.1.4 The LPMS-B IMU Sensor

The LPMS-B is an IMU developed by LP-Research [28]. The assembled sensor is a simple box, see figure 2.5. At the time of writing, individual units can be purchased as in a "Maker Edition" (requiring assembly" for 299$[16], and for about 1100$ fully assembled and with battery chargers[17]. The sensor is aimed at research oriented use and provides very fine-grained control of its working parameters, from being able to calibrate the magnetometer sensor to account for the local magnetic field or adjusting the parameters of the Kalman filter in use in the IMU. It comes with an Open Motion Analysis Toolkit for the Windows and Linux and C++ software libraries for integrating the sensors directly for Windows, Linux and Java for Android.

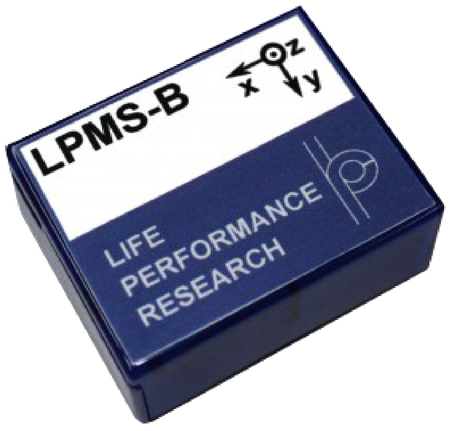


Figure 2.5: The LPMS-B IMU sensor[17]. The dimensions of the sensor are 28 x 20 x 12 mm.

The sensor is battery powered and transmits its data wirelessly over bluetooth, however not via the GATT protocol mentioned previously, but by using the bluetooth connection to emulate a serial connection over which a custom protocol is used. The IMU contains an accelerometer, gyroscope and magnetometer as well as a barometric pressure sensor. The data transmission rate is specified as 400Hz and in addition to the raw data from each individual sub-sensor contained in the IMU, it can provide the orientational data as either Euler angles or quaternions.

2.2 Representing motion and orientation using quaternions

The sensors used in this thesis measure movement primarily through accelerometer, gyroscope and magnetometer sensors, combined in one IMU. The features used later will be based on this sensor data to classify different types of movements and motions and their characteristics. This section provides a brief overview of quaternions as they are heavily used and reasoned about throughout this thesis. It must be noted however that a complete study of quaternions would not be relevant and out of scope for this work, and the point of this section is therefore simply to provide a general notion of what they are and how they may be used.

Quaternions Quaternions are a number system extending complex numbers and can be used to represent orientations in three-dimensional space. Other ways of representing rotation or orientation are Euler angles (specifying the angle of rotation for each 3D axis in a specific order) or rotation matrices.

Quaternions have seen increasing use in the fields of computer graphics and video game development over recent years, partly due to their advantages of easily inverting a rotation, combining rotations and easily interpolating between two rotations. Quaternions also prevent the occurrence of gimbal lock, which is a case where, when using Euler angles, two axes line up and a degree of freedom is "lost".

Quaternions can be thought of as "rotation-deltas", with a quaternion describing a relative rotation in around an axis or vector. Whereas you might need a series of Euler angles to describe a new orientation in space, you only need one quaternion. A quaternion is a four-dimensional vector of the following form,

$$\mathbf{q} = [w, x\mathbf{i}, y\mathbf{j}, z\mathbf{k}]$$

where

$$x, y, z$$

are the vector parts, **i,j,k** are the complex parts and w is the scalar part.

For rotations, we only use unit quaternions, which are defined thus:

$$|\mathbf{q}| = 1$$

The quaternion describing a rotation θ around a unit vector $\mathbf{u}$ can be obtained be the following relationship:

$$\mathbf{q} = \left(cos\left(\frac{\theta}{2}\right), \mathbf{u} sin\left(\frac{\theta}{2}\right) \right)$$

To illustrate, if we wish a quaternion describing the rotation around the vector $\mathbf{v} = (1, 1, 1)$ of $\theta = \pi/2$, using the above expression this will yield

$$\mathbf{q} = \left(cos\left(\frac{\pi/2}{2}\right), (1, 1, 1) * sin\left(\frac{\pi/2}{2}\right) \right)$$

$$\mathbf{q} = \left(\frac{1}{sqrt2}, \frac{1}{sqrt2}, \frac{1}{sqrt2}, \frac{1}{sqrt2} \right)$$

Which we can be normalized to give us the final rotation as a unit quaternion q_n

$$\mathbf{q_n} = \frac{\mathbf{q}}{||\mathbf{q}||} = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2} \right)$$

This rotation can be visualized as shown in figure 2.6. This thesis mainly only uses quaternions to represent orientations and compare them with each other. The rules for rotating a point by a quaternion, combining quaternions to get the resulting sum of rotations and so on have been omitted in this brief overview, as they are themselves not important to the workings of the algorithms dicussed later. Any properties of the math behind quaternions that becomes important to the thesis itself is discussed along the way, when necessary.

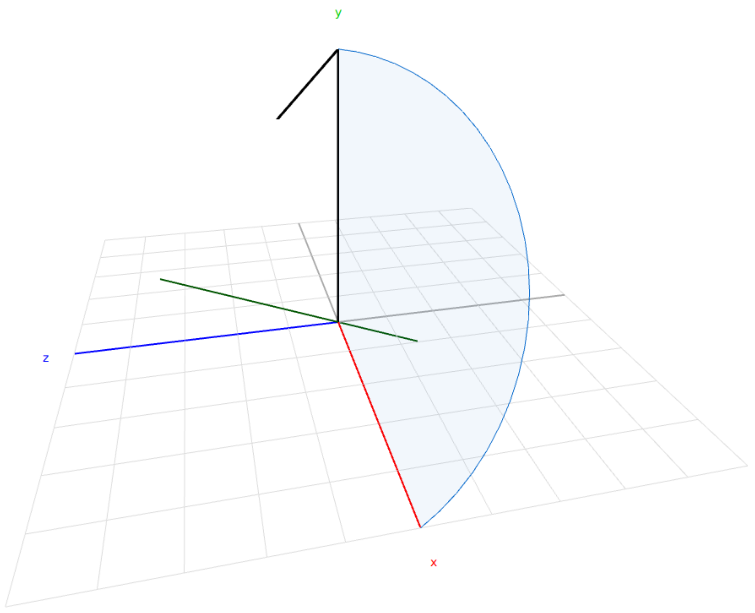


Figure 2.6: Visualization of a rotation quaternion, rotating $\theta = \pi/2$ radians around the vector $\mathbf{v} = (1, 1, 1)$. The rotation is relative, and in this case rotates a point at $(1, 0, 0)$ to $(0, 1, 0)$. Visualization was captured from the tool at [21], useful for visualizing quaternions in space.

2.3 Machine learning

An integral part of the problem of this thesis is evaluating and classifying temporal sequences of data of various dimensions. This challenge is addressed using machine learning algorithms and general classification algorithms. What follows is a brief overview of the topic and the pertinent algorithms.

2.3.1 What is it

Put simply, machine learning is about algorithms taking some data as their input and creating, or learning, a model based on that input to give make predictions about the nature of later input. A model in this context is a description of the nature of the input data and its "behaviour". Often, machine learning means making predictions based on the input data according to the current model. The key point here is that the model of the input is learned by the algorithm instead of it having to be being programmed explicitly, which would simply be man-made heuristics. The creation or learning of a machine learning algorithm's model is called training. A machine learning algorithm is usually trained using training data and tested, to determine its validity or accuracy, using test data. It is important to not use the same data to test the algorithm as was used to train it to avoid overfitting (see 2.3.7).

Machine learning can be split into three categories: unsupervised,

supervised and reinforcement learning. Described briefly, these are:

- Supervised learning - An algorithm is given some input that is already labeled with its correct corresponding output. The algorithm then uses this dataset to generate a model or function which takes an unlabeled input and outputs the predicted label.
- Unsupervised learning - The machine learning algorithm is only given unlabeled input data and needs to create a model of the input data by looking for patterns.
- Reinforcement learning - This can be considered a mix of supervised and unsupervised learning. The algorithm is trained on unlabeled data, but when a prediction is output the model is reinforced by giving it feedback about its predictions (telling it a prediction was "good" or "bad" for instance). This way, the algorithm is nudged in the right direction, but still generating a model itself by looking for patterns.

Different problems might be best served with different algorithms. Supervised learning is only possible when we have a training set with the correct labels, and know the labels in the first place. When we do not yet know the labels, or groupings, we are interested in, unsupervised learning becomes necessary. In cases where we know a better prediction from a not so good one, but don't quite know the correct labels, reinforcement learning might serve best.

2.3.2 No free lunch

The "no free lunch" theorem [30] states broadly speaking that there is no single algorithm which is equally well suited-to every situation or task, and that an algorithm optimized for a certain task will always perform better than the general one. Put differently, to improve the performance of an algorithm means using prior information, or domain specific knowledge, to optimize general approaches to specific problems. This may also be called the lever of knowledge.

2.3.3 Overfitting

Overfitting occurs the model that is learned hasn't learned the actual model which we are interested in, but has only learned to model the training data perfectly. This might be caused by too much fine tuning, resulting in perfect accuracies for the training data, but not being able to classify never before seen samples. Indications that a model is suffering from overfitting is when small changes in the training data lead to great changes in the performance of the predictions of the model.

An example of overfitting is illustrated in figure 2.7. Here, the green line represents a classifier which has lost its ability to generalize, and instead been over-optimized on the training data. Measures that can be taken

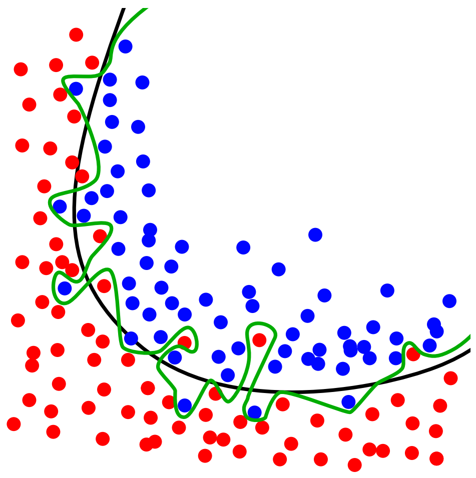


Figure 2.7: Example of overfitting. Illustration from [7].

to avoid overfitting are reducing the complexity of the model and (cross) validation.

2.3.4 Features and feature selection

The input to a machine learning algorithm consists of features. A sample to be classified might have many features, called a feature vector. A feature describes some characteristic of whatever phenomenon we are interested in studying using machine learning algorithms. To obtain good results, it is important to choose features which reduce the complexity of a model, obtain good generalization to avoid overfitting and avoid the curse of dimensionality (see 2.3.4, 2.3.3).

The process of choosing features is called feature selection, and the process of processing existing features or data to obtain new features is called feature extraction. Correct feature selection can be more important than selecting the most optimal machine learning algorithm for a specific task since, if the features do not describe the classes we are interested in learning well, even the best learning algorithm will fail at using them to learn an accurate model of the data.

The curse of dimensionality

The curse of dimensionality refers to the problem of increasing dimensions of data leading to models becoming harder be solved or learned. The higher the number of dimensions of our feature space becomes, the more training data is required to "cover" all the possible inputs we might observe in the entire feature space. Even if the high dimensional features are relevant to the classes we are interested in predicting, the higher the dimensions the smaller the differences in distance between individual samples will become. This will result in all samples seemingly being equally close to all other samples, and the predictions of the classifier will end up basically predicting at random.

Additionally, the higher the dimensionality of the feature space we use, the harder it becomes to intuitively reason about the problem at hand. However, the ability to do so is often crucial, since problem specific prior knowledge is crucial to optimize for that specific problem (2.3.2).

2.3.5 Nearest-neighbor Search

Nearest-neighbor search is the problem of finding the points nearest to another point in a space S. Ideally, with well selected features, the proximity between points in space S is related to the class a point belongs to. In general this concept is used for classification or pattern recognition and is used widely on problem spaces where these can be applied. The proximity or similarity of two points is the result of a proximity function which can be tailored to the problem domain. To illustrate this, three cases for the dimensionality of S and the proximity function are presented below: euclidean distance, quaternions and rectilinear distance. Using the nearest-neighbors we can classify a point using the k-nearest neighbor (k-NN) algorithm, wherein a point is classified by majority vote among the classes of its k nearest neighbors (see figure 2.8).

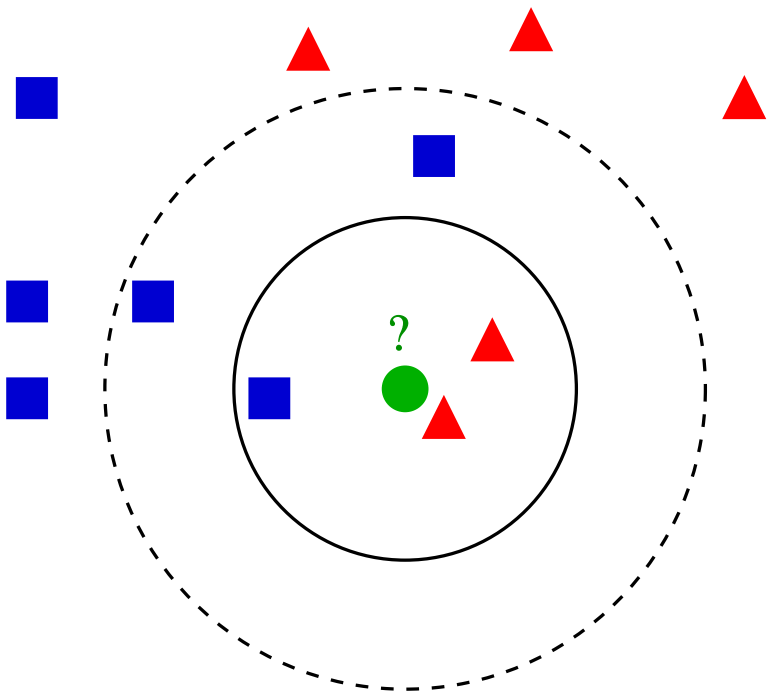


Figure 2.8: Illustration of nearest-neighbor search applied as kNN. The green circle is classified by majority vote as red using its k=3 nearest neighbors, 1 blue and 2 red. [2]

Proximity functions

We can illustrate the nearest-neighbor algorithm as a classifier by viewing two sequences of data as points in a space S. To classify a sequence of data points as as being of a certain kind we can compare it with other sequences we may already know belong to a certain category. One way of comparing sequences is by calculating a measure of distance between the two sequences and using this as a measure of similarity between sequences. This method assumes that the distance function used to calculate the distance between two sequences correlates with their similarity relative to each other in the domain that we're interested in. This means we need a distance function which gives larger output values for inputs that a are dissimilar and lower outputs for inputs that are similar. Thus, the sequences themselves will constitute points in space S of r dimensions, where r depends on the dimensionality of the sequences. In the case of the sequences being viewed as euclidean vectors, we can simply calculate the euclidean distance between the sequences (and each of their respective points) by calculating the respective vector norms. For the sequences **x** and **y** with length n and the distance d between two scalar data points $\mathbf{x_i}$ and $\mathbf{y_i}$ being $x_i - y_i$ the total euclidean distance between the two sequences thus becomes the sum of the distances between their respective data points.

$$distance(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n}(x_i - y_i)$$

The distance function between two points in the sequences depends on the dimensions or domain of the individual elements in the sequences. Imagine we have two sequences comprised of positions in three dimensional space. Calculating euclidean distance (a straight line between two point in space), the distance between two elements p and q of the two sequences could be computed as

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2}$$

However, imagine we have sequences of something other than simple vectors in euclidean space. We might have sequences of quaternions, in which case the distance of one sequence-point to another might be considered the quaternion which represents the rotation from quaternion to the other or the angle of rotation described by that difference-quaternion. In the latter case, the distance $d(\mathbf{p}, \mathbf{q})$ gives the angular distance θ between two quaternions and could be calculated as,

$$d(\mathbf{p}, \mathbf{q}) = \arccos(2 * \langle q, p \rangle^2 - 1) = \theta$$

Euclidean distance is not always the most preferable distance metric in high dimensional spaces however [1] and an alternative to euclidean distance functions called taxicab geometry, also referred to as Manhattan distance, might prove more effective. This way of calculating distance is illustrated in figure 2.9.

A general disadvantage of simply calculating the distance between two sequences element-by-element is that we are constrained to using sequences of equal length. Additionally, if we have discrete-time sequences and one sequence is stretched in the time domain the utility of the

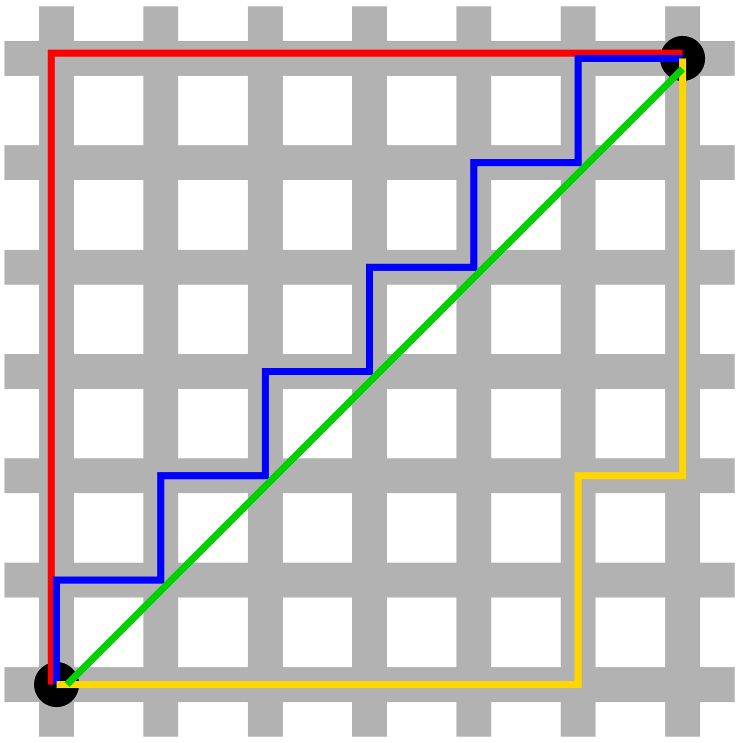


Figure 2.9: Manhattan distance. The blue, red and yellow lines represent the shortest paths between the two black circles in Manhattan distance. The green line represents the shortest path in euclidean distance. [27]

calculated distance will diminish depending on the features we are interested in, as the features (data points) of each sequence no longer line up. This might happen when a two sequences show the same thing (for instance the same exercise), but one was recorded with a higher sample rate, or one was performed more slowly than the other. If both sequences represent the same movement, but one was performed more quickly than the other with the sampling rate being equal, that sequence will appear "compressed" and the calculated distance will indicate a larger distance value in spite of both exercises belonging to the same category. One way of solving this is by using DTW (see 2.3.6). Once distances between sequences can be compared, methods to classify sequences based on their spatial relationship to known sequences can be used. For example, the nearest-neighbor classifier (see chapter 2.3.5) might be used to classify a sequence based on the nearest sequences in terms of the calculated distances.

2.3.6 Dynamic Time Warping

DTW is meant to solve the problem of comparing temporal/ordered sequences of different lengths and stretched or compressed discrete-time sequences.

A regular DTW is an algorithm measuring the similarity or distance between two temporal sequences. The sequences may have different lengths. The name comes from the algorithm essentially "warping" sequences in the time dimension to optimally align with each other. This will align similar features in the two sequences with each other independently from where in the time dimension they are located. The algorithm calculates the optimal distance between each feature and finally the total optimal distance between the two sequences, thus providing a relative measure for their similarity. This measure of similarity is often a more suitable metric than the absolute distance between two sequences for sequences which are similar, but stretched or compressed in time (such as for instance speech, or exercises).

Consider the time sequences

$$\mathbf{x} = x_1, x_2, x_3, \cdots, x_n$$

$$\mathbf{y} = y_1, y_2, y_3, \cdots, y_m$$

where n and m are the lengths of sequence x and y respectively. Now consider that each sequence represents the same thing, but stretched. When simply calculating the difference between each of the points using

$$distance(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} (x_i - y_i)$$

the points which will be compared will not correspond to the same features in each sequence. See the left side of figure 2.10 for an illustration of this. To correctly compare the differences, or distances, between the points each sequences belonging to the same features of the sequences, we can use DTW, thereby getting a more realistic measurement of distance between the two sequences.

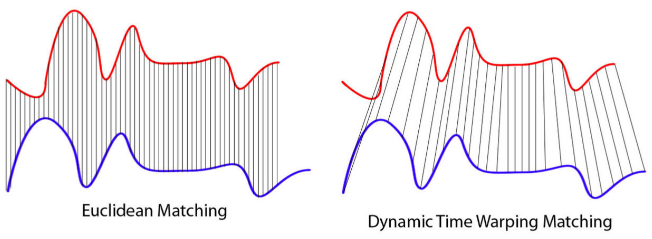


Figure 2.10: An illustration[31] of the difference between calculating the distance direct euclidean distance between two sequences by comparing every element with the same indices (left), or using DTW to compare elements thought to belong to the same features of the two sequences (right).

DTW achieves this by “warping” the sequences in the time domain. Every feature in each sequence is assigned to the feature in the other other sequence which has the shortest distance to it, after warping. The sum of these shortest distances is the similarity or distance between the two sequences. To find the pairings between all elements of the two sequences, we can view every pairing combination in a grid, where the value of a cell is the distance, or cost, of a pairing. This grid is illustrated in figure 2.11. When we look at the problem from this perspective, finding all matching-pairings becomes a problem of finding the shortest path through this grid.

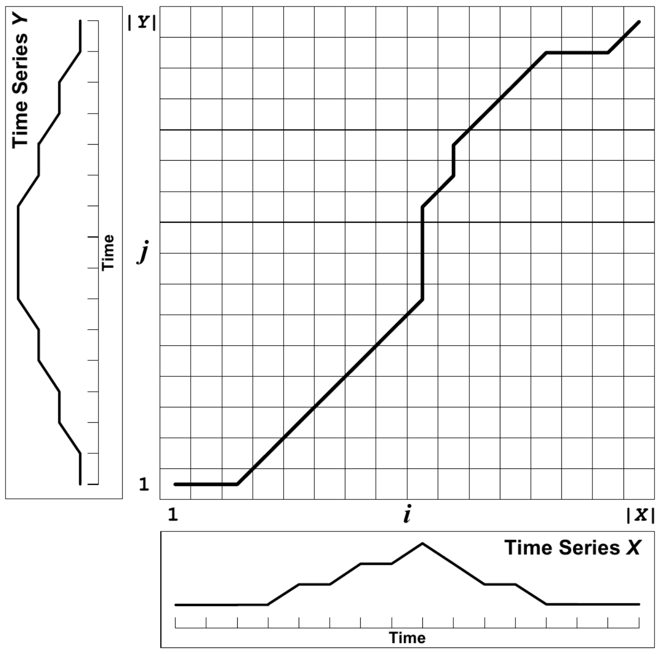


Figure 2.11: Illustration[8] of DTW, using a grid to represent all possible sequence-pair alignments. Finding the best pairings will warp the sequences to match.

To find the shortest path through the grid of possible sequence-feature-pairings, we define the cost of the shortest-path to a cell (x_i, y_j) as the following, where $d(x_i, y_j)$ is the simple cost between a pair $(x_i - y_j)$:

$$distance(x_i, y_j) = d(x_i, y_j) + min(distance(x_{i-1}, y_j), distance(x_i, y_{j-1}), distance(x_{i-1}, y_{j-1}))$$

The total distance for a point (x_i, y_j) is thus the recursive distance including all the points in time before it which formed the shortest path to (x_i, y_i). To find the shortest path to the top right corner, ie. the end of both sequences, we therefore only need to calculate the local and total shortest-costs beginning at the lower left corner, the beginnings of both sequences, and then once completed for the whole grid, backtrack the shorted path from the top right. This is illustrated by the red dotted line in figure 2.11.

Since calculating every possible cost in the grid can quickly become too burdensome in terms of time and resource requirements of the algorithm, some constraints can be imposed. Some examples of these are requiring monotonicity (the path cannot go backwards in time, ie. i and j must never decrease), we can introduce a window size (the shortest path is constraint to a window covering the diagonal of the grid) and window shape (the slope of the path might be constrained to never become too steep or shallow).

2.3.7 Validation

Validation serves the purpose of validating a model we have developed using machine learning algorithms. In other words, it is used to validate that the model has not simply learned the training set by overfitting, but can generalize to new and unseen data not belonging to the training set. Cross- or fold-validation can be used to achieve this.

Cross-validation

When training and testing a model or an algorithm, we might partition the dataset's labeled samples into two groups, training on 70% of the dataset and testing on the remaining 30%, and call it a day. If we do this, we won't know whether our model simply happens to do well at predicting exactly the 30% we tested it on. Overfitting might have happened. Additionally, if our dataset is comparatively small in size, the sample size corresponding to 30% of all samples (or whatever values are chosen) might not give us very much confidence in the actual performance of the model. To solve this, we effectively increase the size of our testing dataset to include the whole dataset (in a way), as well as increasing the chances of noticing overfitting, by iteratively partitioning our dataset into k-folds, where k is the number of partitions or folds.

For $k = 4$, we would partition the dataset into 4 folds, or segments. We then choose one segment as the testing or validaton segment, and use the remaining three to train the model. We do this for every fold, such that each fold is used the testing set once, while the rest are used as the training set. Afterwards, we can calculate a mean error or average accuracy over all of those validations, to give us an a better indication of the model's actual predictive accuracy.

Alternatively, we can do exhaustive cross-validation by splitting the dataset into testing and training segments in every way possible. This is done by leave-p-out cross validation. Here, p-samples are used as the testing or validation set, while the rest of the samples are used as the training set. Like with k-folding validation, we then do this for all ways of splitting the dataset into p-testing samples and the rest. We can then try every p-value as long as p remains smaller than the total number of samples. This last approach however can be argued to quickly become pointless as the training set becomes tiny compared to the testing set. Furthermore, the computational complexity of larger datasets will quickly make exhaustive cross-validation unfeasible.

2.4 Previous work

2.4.1 Previous work in automated rehabilitation evaluation systems

Various automated systems and methods have been proposed to improve the efficacy of physical rehabilitation at home [11, 24, 31] and these systems can potentially give objective metrics as opposed to being reliant on subjective observation even in a clinical setting [14, 24]. Though they may focus on different problem areas and methodologies within the same greater context, many try to provide users, or patients specifically, with automated feedback without the need for an attending specialist. The same solutions may be of interest in both the area of sports and professional healthcare. Some of these previous works and their methods have been presented below in more detail to give context to the methods and experiments chosen for this thesis.

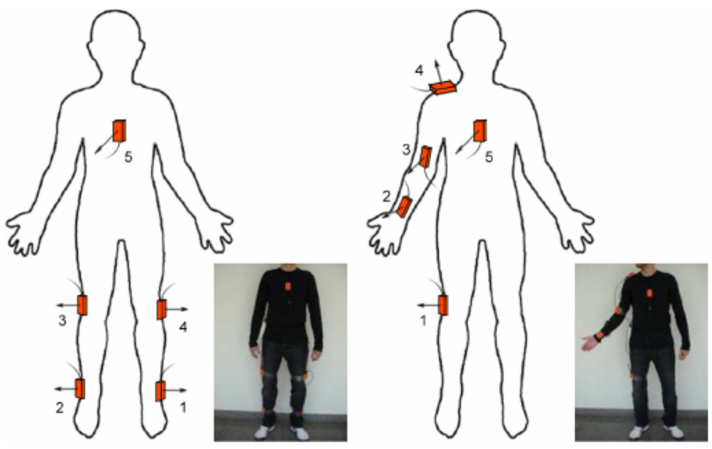


Figure 2.12: Two sensor placement configurations where used in [31]

In [31], an effort was made to develop an autonomous automated system that would solve the problems of automatically evaluating an exercise session from a set of possible exercises correctly. An exercise session in this context consists of a period of activity containing several different types of exercises with idle time in between. The exercises were orthopedic rehabilitation exercises. Motion was captured using five inertial sensors (see figure 2.12). The sensors used were MTx units developed by Xsense [13]. Each of these sensors contains an accelerometer, a gyroscope and a magnetometer. Specifically, the system's goal was to categorize parts of recorded movement sequences as belonging to a specific orthopedic exercise and error type. These sub-sequences were compared to template exercises, recorded once for every test subject, exercise and error type. A machine learning algorithm was developed to address these requirements and both the template exercises as well as the training dataset was recorded using motion data captured from a group of five healthy test subjects.

The algorithm developed to that purpose is an extension of the dynamic time warping algorithm (see 2.3.6). The raw sensor data of all sensors was captured as discrete-time sequences and the sequences for each sensor type (ie. gyroscope, accelerometer etc) were normalized for that sensor type.

Eight orthopedic exercises were included in the experiments and every one of the five subjects performed one standard reference template of every exercise and their respective types. The types of each exercise classified were 1) the correctly performed exercise, 2) the exercise performed incorrectly either by performing it too quickly or 3) the exercise performed incorrectly by performing it with a low amplitude (minimal movements). Thus, every test exercise session dataset would be compared against these reference templates to classify an exercise as incorrectly or correctly performed and optionally to classify the type of error by calculating the DTW distances to the correct reference templates of the two error templates. Idle time between exercises during an exercise session was also taken into account by using thresholding on the distance metrics to prevent classification of data with distances too great to any exercise class. Leave-one-exercise-out was used to test the robustness of the system against unknown exercises.

The results of [31] for a total of 120 reference sets and 1200 test sets are an accuracy of 93.46% for exercise classification only (determining which exercise was performed) and 88.65% for classification of both the type of exercise and execution type (correct or incorrect, ie. too fast or low amplitude) at the same time. Advantages of the developed system are its autonomous nature in that the whole system does not have to be reprogrammed to be used for new exercise types, changing the number of sensors and or altering the default placement of the sensors on the body. Possible disadvantages may include that the algorithm is not real-time and thus cannot provide completely real-time feedback to patients during an exercise. Furthermore, should the execution of an exercise be incorrect in multiple ways, only one will be detected. If an error is too extreme (the exercise might be performed much too quickly) the general exercise type might not be classified at all. The main critical disadvantage in the view of this thesis' author is that it's necessary to record a reference template for each exercise and every one of its possible error types, resulting in a large number of exercises needed to be recorded if detailed feedback on error types is desired. Thus if a new patient were to use the system, this patient would have to record every exercise to be used by her both correctly and incorrectly for every error type to be detected. For every exercise, that means at least three recordings for two error types. These would have to be performed under specialist supervision.

In [11] it is investigated whether the performance (quality of execution) of lower limb exercises in patients can accurately be measured using inertial sensors and whether the reduction of the number of sensors has a significant impact on the accuracy of the measurements. Fifty-eight patients from a physiotherapy clinic were selected to perform ten repetitions of seven lower limb exercises using three inertial sensor units to measure motion data. The sensors were fastened to the thigh, shin and foot of the leg that was exercised (see figure 2.13). The data was processed by an offline algorithm, classifying for binary correct- or incorrectness and the error classes correct/incorrect alignment of movement, speed and quality of movement. Some of the features extracted from the calculated acceleration, magnitude, pitch and roll were signal mean, skewness, kurtosis, signal energy and signal range after which principal component analysis (PCA) was performed. A logistic regression classifier was used

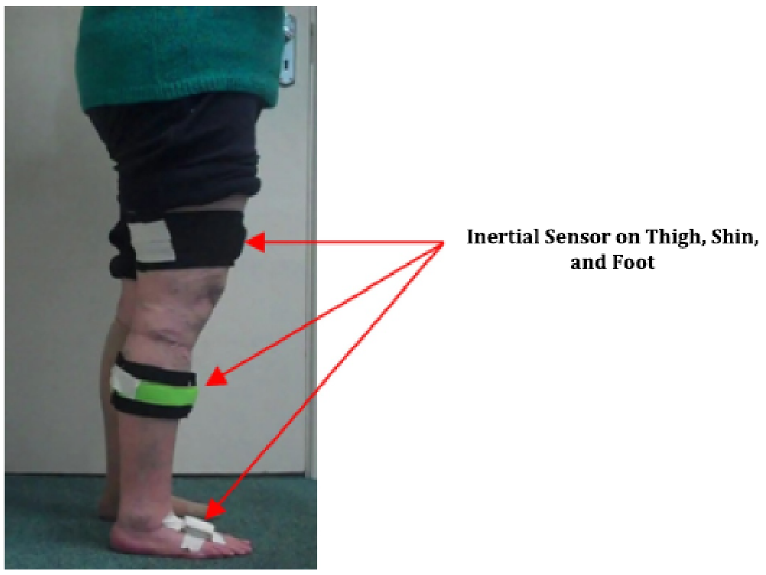


Figure 2.13: Sensors attached to the right light for orthopedic exercise evaluation in [11]

for classification. The results of [11] indicate that a reduction of the number of sensors has relatively little impact on the accuracy of the classification. For binary classification (correct or incorrect exercise), three sensors achieved an average accuracy of 81%, two sensors an accuracy of 82% and using a single sensor gave an accuracy of 83%. The accuracy for multi-label classification (distinguishing between the different error types) gave accuracies of 63%, 61% and 63% for three, two and one sensors respectively. The classifier was trained and tested using leave-one-subject-out cross-validation.

Part II

The project

Chapter 3

Implementation

The aim of this chapter is to follow the path this project took throughout its course and explain the individual choices that were made along the way. Briefly, in 3.1 initial initial assumptions and constraints are presented which served as the starting point of the work that was done. In 3.3, the complete process of obtaining the necessary datasets is explained, with sub-sections dealing with the exercises used in 3.3.2, the data recording sessions themselves in 3.3.3 and the necessary software that had to be built, described in 3.3.4. Finally, 3.4 presents the machine learning aspects of the implementation, discussing classes, features and the choice of algorithms, the results and testing of which can be found in the following chapter, 4.

When planning, it is necessary to the see big picture while also allowing for the chance that a plan might be lacking and in need of changing. The real world process of planning and creating proves to be a rather iterative process of constantly looping between the planning and doing stages. Illustrating this iterative process is difficult in a sequential thematically ordered thesis however, so I will simply note when the evolution of an idea or constraint had an important impact on the core of the thesis itself.

3.1 Starting point

The ultimate aim of this thesis was to work towards solutions for remote physiotherapeutical treatment. As such, a certain degree of realism in terms of the data and algorithms used was decided upon, as all data required for the building of datasets had to be gathered and processed as part of the thesis. These constraints were formulated as:

- Obtain data from actual physiotherapy exercises, in use in clinical settings
- Use error types that are as close to real clinical exercise errors as possible
- Only try to detect one error type at a time
- Use sensors that could conceivably be used in a home environment by non-tech-savy users

- Apply machine learning techniques and select features in the service providing information conceivably useful to those users
- Due to the limited scope of the thesis, only apply offline machine learning

These initial constraints evolved slightly and the changes and challenges are described in the following sections along the way. Real-time error detection was quickly discarded as a goal for this thesis, as the scope of work involved in the data recording proved to be relatively substantial.

3.2 Why build a custom dataset

Most existing, publically available datasets on human movement found by the author at the time of writing were built for the purposes of general activity classification, such as walking, standing, sports activities etc. Some few were for the purposes of classifying hand gestures, for instance for sign language. Since this thesis concerns itself with exercises used in the field of physiotherapy and measuring their correctness, the decision was made to build a custom dataset for this thesis. This would also have the advantage of being able to control which kinds of sensor data would comprise it, giving more control over which kinds of machine learning techniques could be applied to it.

3.3 Building the dataset

To enable machine learning algorithms to output helpful data on the performance of an exercise, data is needed describing that movement in the real world, obtained through sampling using sensors some kind. The data sampled must itself, in theory, contain the information that we wish to obtain using machine learning techniques, and the sampling technique itself should ideally provide the data in a format that is easily amenable to processing using said machine learning techniques. The dataset should be large enough for meaningful results and to enable the use of validation techniques.

Thus, we have two main factors to consider: the actual source of the data having a high information content for what we want to find out, and the format of the data being as easily processed as possible (see 2.3.4 for challenges regarding data of high dimensions).

The following subsections treat the main areas affecting the aforementioned factors: test subjects, recording protocols and sensors.

3.3.1 Test subjects

The data fed to the machine learning algorithms in this thesis comes from real test subjects, instead of for instance simulations. It might be possible to simulate the movement of people for the purposes of analyzing physical exercises, however that problem was deemed out of scope for this thesis. More importantly, many movement patterns exhibited during physical rehabilitation by patients are arguably difficult to simulate even for healthy test subjects due to the neurological defects that cause them creating unique types of movement.

Since the purpose of this thesis is to in some way work towards a solution for treatment of real patients or users to improve their health, ideally the test subjects would be real patients with real disabilities. This was considered, but ultimately dismissed due to the time and regulatory hurdles required to run a study on real patients. Instead, regular healthy volunteer test subjects were decided upon for the data gathering stage. The only constraint put on who might be a test subject was the ability to physically perform the chosen set of exercises correctly.

Without already having the results of the data processing at hand, it would be difficult to predict the number of test subjects necessary to achieve a satisfactory results. Generally more are better, but in practice time is limited. It was assumed that 5-8 test subjects would be sufficient, given the chosen set of exercises (see 3.3.2).

Ultimately, five test subjects were recruited according to the requirements listed above to build the dataset. The test subjects were additionally chosen to have a relatively big range of age and body types in the set of test subjects, to potentially test the robustness of the algorithms employed to these factors. All test subjects were at the time of recording generally healthy.

3.3.2 Exercises and errors

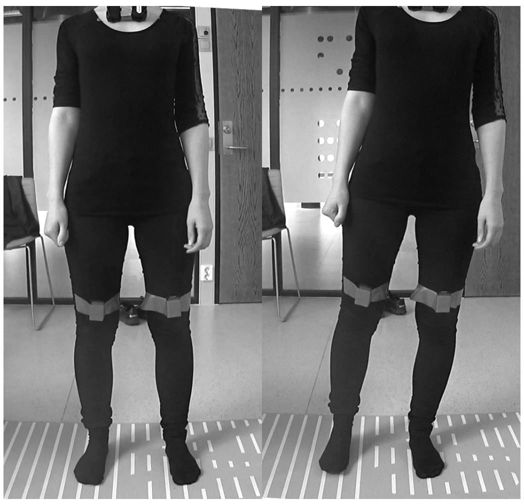
The set of exercises one would want to choose should ideally be simple enough perform with minimal training required and easily provide the types of errors we wish to test for. Otherwise, the exercises should be as realistic as possible and be in use in clinical settings.

There are many ways exercises can be performed incorrectly, and a near infinite number of combinations of errors in the real world. To simplify the initial development and testing of machine learning techniques, each exercise was recorded only with one type of error at a time in addition to its correct form.

Henceforth, the different generalized types of errors made during an exercise - including the class of "correct" - shall be referred to as **error types**. To differentiate between types of exercises (biceps curl vs knee extention for instance), the terms **exercise type** or simply **exercise** will be used. This is done to help differentiate between these two classes predicted for in this thesis.

The three general error types chosen are amplitude (too large or small movements), speed (too fast or slow movements) and trajectory (not moving in the right direction).

Each exercise consists of a movement repeated 10 times. Each exercise was performed in 4 ways to capture all correct and incorrect modes of


(a) Exercise: Sideways lean. Start and middle positions of the can be seen to the left and right, respectively. The end position is the same as the start position.

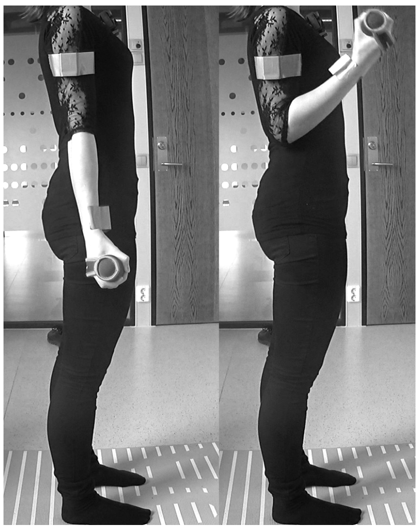

(b) Exercise: Biceps curl. Start and middle positions of the can be seen to the left and right, respectively. The end position is the same as the start position.

Figure 3.1: The biceps curl and sideways lean exercises.

movement (correct and the three error types described above). Every one of these modes was recorded 5 times.

Initially, the number of exercises considered was 5-8. Given this, and the number of 5 test subjects, the size of the resulting dataset would have been 500-1280. The main challenge in choosing which exercises to use proved to be choosing exercises which were easy enough to repeat often for regular healthy test subjects and which were easy to perform without the use of extra training equipment. For the average generally healthy test subject, repeating an exercise an exercise containing 10 repetitions 4 times for each category (correct, and three types of error) quickly became arduous. For practical reasons the author decided to limit the number of exercises in favor of keeping the number of repetitions per exercise. This way, it was hoped, the higher number of repetitions would compensate for unintended errors in a repetition, while a higher number of exercises with fewer repetitions each might have provided a wider test bed for general error detection, but at the expense of confidence in the accuracy of the evaluation of a repetition. Thus, due to time constraints, 3 exercises each consisting of 10 repetitions were deemed an appropriate trade-off.

The time and work required to actually record exercises with test subjects proved more extensive than anticipated, for reasons detailed in 3.3.3. The resulting dataset size of a total of 300 recorded exercises was considered sufficient. The three exercises chosen and a short description of their movements can be seen in figure 3.2, figure 3.1b and figure 3.1a.

The exercises were chosen from a range of exercises suggested by specialists from the hospital of Sunnaas Hospital. This was done to ensure

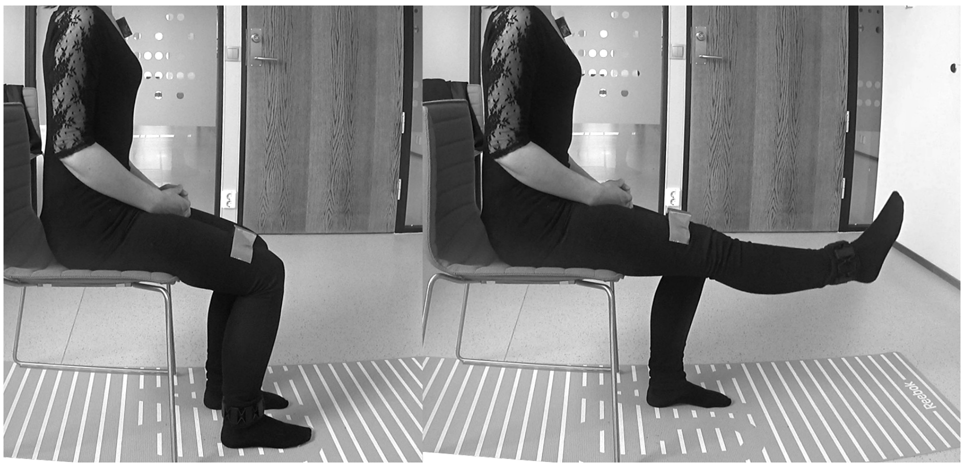


Figure 3.2: Exercise: Knee extension. Start and middle positions of the can be seen to the left and right, respectively. The end position is the same as the start position.

clinical relevance in terms of each exercise, as well as suitability in terms of the thesis goal with regards to error evaluation. Thus, the exercises all share the common traits of consisting of repetitions of movements which have a beginning, middle and end point and provide the opportunity of exhibiting easily recognizable manifestations of the types of errors we are interested in. For instance, when performing the biceps exercise one might lower the arm too quickly, or too far, or potentially both. Additionally, the exercises where chosen such that the arcs of movement in space would be different for each exercise. The purpose of this was to hopefully avoid similar looking datasets, increasing the chance of successfully processing the data according to true general underlying properties of the nature of the errors this thesis is interested in, instead of arriving at solutions that don't generalize well for other exercises.

3.3.3 Recording session protocol

Following a consistent and well-defined protocol for recording the exercise data is a crucial factor for obtaining reliable data. To facilitate this, a protocol was developed detailing the complete process of recording an exercise with the help of a test subject from beginning to end. The protocol was tweaked in a few ways after initial trial runs to account for practical constraints and requirements, especially in terms of time available. These tweaks are detailed along the way. Building the dataset for later computation and analysis was a non-negligible part of this thesis and thus warrants discussing.

Any gathering of data sampled from human test subjects requires strict control and attention to the issues surrounding consent and data privacy. As such, a consent form was formulated and signed by every participant of the recording sessions, specifying for instance the requirements that no personally identifiable information would be published in the results of the work. For identification in the thesis work, each test subject was assigned a numerical id.

While avoiding unintentional variance in the recorded data is important, the main reason real people are used as test subjects instead of for instance using movement data from a simulation is the unpredictability and variance obtained by sampling in the real-world. A hypothetical movement analysis system intended for unsupervised home-use, by its very nature, must be able to handle this. A balance must therefore be struck between allowing each individual's own way of correctly performing the exercises to be represented in the data, while at the same time avoiding any unintended modes of movement that might obscure the underlying characteristics of exercises and error types we are interested in.

Some assumptions were made early on that proved not to be feasible for this thesis. The point of the dataset is to provide realistic data on exercises performed both correctly and incorrectly. Initially, it was assumed that the best way to get this dataset was to simply let test subjects perform the exercises after them receiving general instructions. A panel of specialists would then label those recorded exercises as correctly or incorrectly and if applicable with the correct type of error. In other words, erroneously performed exercises would be "generated" naturally, by letting the test subjects make realistic mistakes. It quickly became apparent however that this approach required too much time for an uncertain benefit. The plan was to let three specialists from Sunnaas Hospital watch video recordings of test subjects and then label those recordings. This would have meant devising a way of handling non-unanimous labeling, where one or all of the specialists disagreed on a label. The labels assigned by the specialists would have then been used to label the sensor data itself, either manually or by having to create an automated system first. Furthermore, the video recordings would have been another factor to be standardized for every recording session, the videos would have had to have been anonymized manually and then been made available to the specialists. Only after this lengthy process it would be clear whether the dataset even contained enough examples of each category. If not, it might then be necessary to record more in the hope of obtaining enough samples of the right kind by chance. Finally, it would be unclear how much more realistic "natural" mistakes during the exercises would be in a clinical context anyways, due to the test subjects being generally healthy as opposed to actual patients. After some initial recordings which followed this strategy and considering an estimation of a total of 75 recorded exercises of at least 60 seconds each, this approach of labeling was deemed out of scope for this thesis. This decision was strengthened by the fact that the main goal is to investigate the general feasibility of this kind of system (see 1.3, with a special focus on the machine learning implementation, not clinical viability.

Instead of following the strategy described in the previous paragraph, it was decided to simply guide the test subjects in a standardized way, formulated with the help of a specialist, to perform each exercise according to the error types required (see 3.3.2). This approach has the advantage assuring enough examples of each error type in the dataset, at the expense of the, albeit questionable, authenticity of “naturally“ erroneous exercise recordings. To minimize the possible impact of inauthentic mistakes, another criterion for the choice of exercises was decided to be that their error variants should be simple and not too complex to “fake”. In practice, this proved to be a non issue during recording sessions, as the error types chosen are easily explained, demonstrated and recognized by eye for the purposes of this thesis.

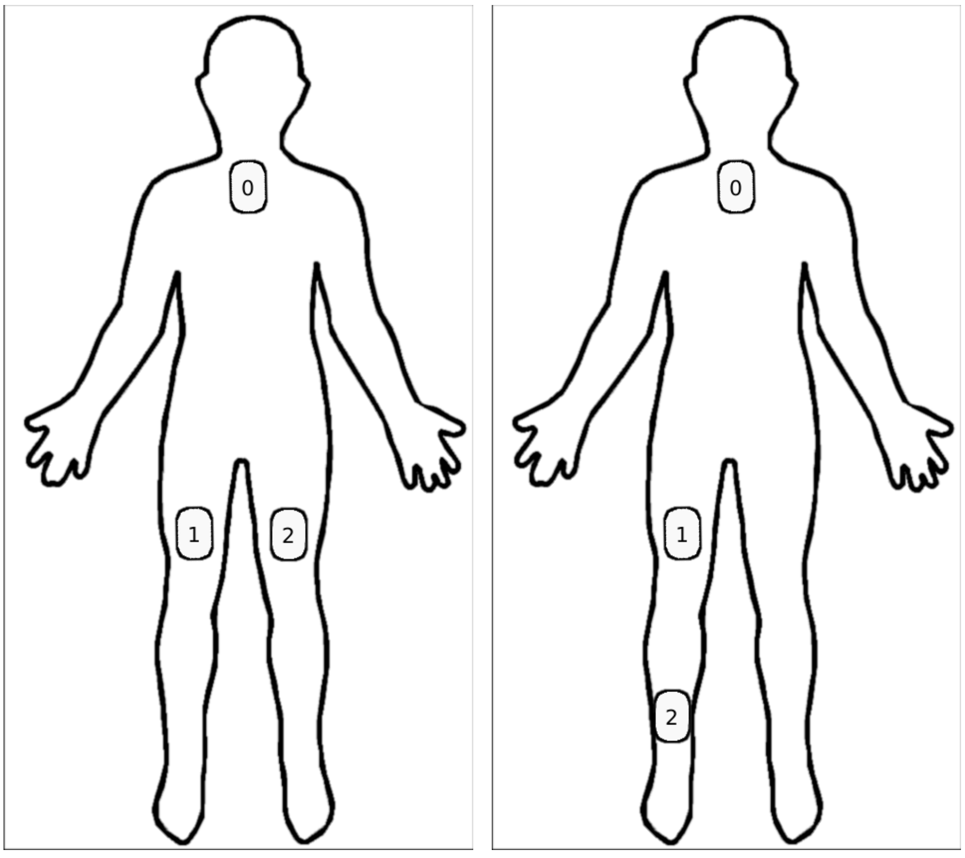


Figure 3.3: a) The placement for the sensors for the sideways lean exercise. Sensor 1 is placed on the right thigh above the knee, sensor 2 is placed on left thigh above the knee, while sensor 0 is on the chest at the top of the sternum. For details, see the figure 3.1a. b) The placement for the sensors for the knee extension exercise. Sensor 1 is placed on the right thigh above the knee, sensor 2 is placed on the right shin bone facing forward, above the ankles, while sensor 0 is on the chest at the top of the sternum. For details, see the figure 3.1b.

Ideally, we want every test subject to perform each exercise and its error types in the same general way. Thus, for each exercise and its error variants a set of standard instructions was formulated to be used while guiding the test subjects through the recording sessions. Furthermore, the recordings must be labeled correctly as a session goes on, and a checklist was used to make sure every mode of every exercise was completed for every test subject.

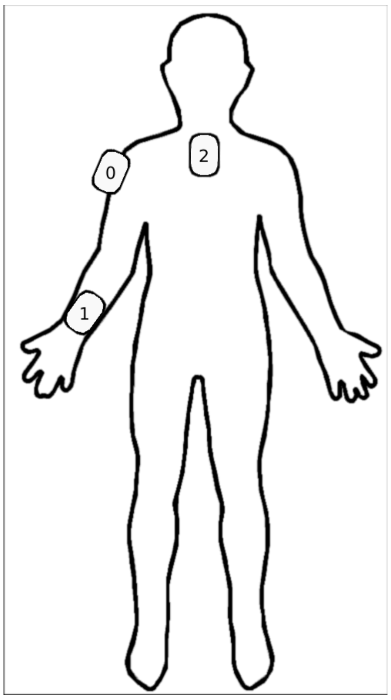


Figure 3.4: The placement for the sensors for the biceps curl exercise. Sensor 0 is placed on the outside of the right upper arm, sensor 1 is placed on the inside of the wrist of the right arm, while sensor 1 is on the chest at the top of the sternum. For details, see the figure 3.1b.

Another important factor to control for during across the recordings is consistent placement of sensors on the body during an exercise. An even distribution was arrived at with the individual sensor's placements aiming to enable the sensors to capture the main movements involved in an exercise's movements. The general sensor placements for the individual exercises are presented in figures 3.3a, 3.4 and 3.3b. Every sensor was mounted with its x-axis pointing down or towards the end of the limb it was attached to, with its z-axis pointing outwards away from the body. Straps and tape were used to attach attach the sensors on the body. The table for which sensor id belongs to which type of sensor is shown at 3.1.

Table 3.1: The sensor indices and which type of sensor they belong to.

Sensor ID	Type of sensor
0	LPMS-B
1	LPMS-B
2	Myo

Having the placement of sensors on the body be the same across all exercises instead was considered, but rejected. The ultimate hope of this thesis was to find ways of classifying error types based on inherent characteristics of the movements these error types exhibit, independent of the type exercise or sensor placement. More importantly however, having the same sensor placement for all exercises might mean that, since the exercises, vary a lot in which body segments are being used, some of the sensors might barely be moved at all for some exercises, but receive a lot of characteristic movement for others. Thus, instead of having the same sensor placement for all exercises, each exercise has its own sensor placement in order to enable the sensors to capture the main movements of every exercise. See chapter 5 for more remarks on this decision.

The final protocol consists of a checklist, exercise instructions and sensor placement instructions.

With the protocol in hand and endeavoring to follow it diligently, we can turn to the task of actually recording the data we need to build out dataset. For this, both hardware and software need to work in tandem, as described in the following section.

3.3.4 Recording hardware and software

Hardware

Many types of movement recording technology exist, some of which are described in 2.1. To keep in line with the states goals for this thesis, the choice of technology was constrained to something affordable and most importantly something which can provide movement data in a way that lends itself to machine learning algorithms and the task of evaluating the exercises for the types of errors we are interested in. Ideally the technology would lend itself to easy deployment in a home or other unsupervised setting, given that the methods for evaluating exercises in this thesis prove to be generally feasible.

Furthermore, the chosen sensor system would need to work wirelessly to ease setup and use. Additionally, it would provide real-time sampling data in a format that requires as little processing before feeding it to the machine learning algorithm as possible. Even though the machine learning in this thesis happens offline (see the glossary for offline algorithm), a future implementation would want as little computational complexity as possible between real-time data sampling and generating feedback, to enable the feedback to be as close to real-time as possible. To summarize the important constraints:

- Affordable
- Wireless
- Deliver meaningful exercise movement data in real-time
- Relatively easy to integrate with custom software
- Theoretically suited for use in a home or unsupervised environment

Given those factors, the sensor technology in thesis was limited to body-worn IMU sensors (see 2.1.2). Off-the-shelf optical systems were dismissed primarily due to their lack of availability and lack of easy integration in custom systems, compared to the IMUs. Higher-end optical systems were discarded due to the amount of work and care required for setup and use making them ill-suited for anything like home use, in addition to their high price.

The final choice of hardware, the Myo sensor armband and the LPMS-B IMU, was ultimately a result of them fulfilling the technical requirements as well as being affordable and available at the time. Two Myo sensors and 2 LPMS-B IMUs were available during the work of the thesis. It was decided to use both types of sensor, since they were both available and using two seemed to add little extra cost in terms of work, and in case it became clear later that the data of one type of sensor was more useful than that of the other. The latter was considered likely, since the LPMS-B is more expensive, research oriented IMU with less observed orientational drift, among other things, compared to the Myo. The Myo proved more prone to drift and its use was plagued with some technical problems. It was thus thought that using both might enable an analysis of which is more suitable for exercise analysis overall as an added benefit(see 5 for that discussion). The technical problems with the Myo lead to the LPMS-B being prioritized for the sensor placement layouts.

Both the Myo and LPMS-B enable wireless transmission of real-time orientational data using an open API. Furthermore, both support transmitting the orientational data in quaternions as well as gyroscope and accelerometer samples, with only the LPMS-B being able to also transmit its raw magnetometer data. Being able to use quaternions, precomputed in the sensor hardware, as well as "raw" sensor data (ie. magnetometer, accelerometer and gyroscope data) was thought to offer the possibility of exploring the utility of quaternions for analysing movement using machine learning, compared to the seemingly more usual way of using the sensor data directly as a base of feature extraction (see 2.4).

Software

Hardware sensors don't work in isolation. Software is necessary to receive the data from the sensors, label it correctly and archive it for later use.

A large amount of the hands-on work required to build a dataset for the purposes of machine learning was in the building of the software necessary to enable recording of the data by interfacing with the sensors, initializing them and saving the data in a practical manner. The requirement was set to have a program that would be supplied with the information about a recording session such as exercise name, type and test subject id, and would then connect to both types of sensors. It would then potentially calibrate them, record and possibly visualize the data (to enable real-time monitoring by eye) and finally save the data from both types of data to the same file.

There were multiple ways to go about this. While the company behind the LPMS-B, LP-Research, supplies a fully featured SDK and software suite to use the sensor, the Myo is only distributed with a limited SDK and no

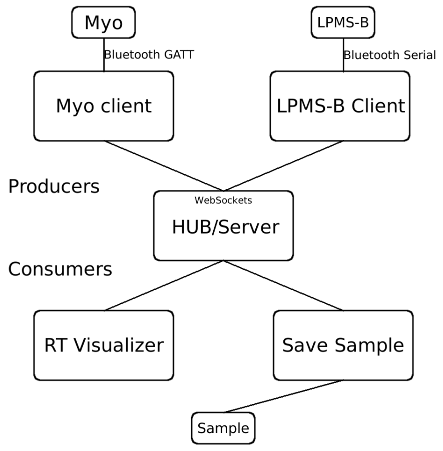


Figure 3.5: Overview of the software implemented to record exercise data, using Myo and LPMS-B type sensors. Two clients, using each sensor's individual API, stream their sensor's data to the hub, which distributes it to any clients, such as a real-time visualizer and the sample saving client which creates the samples later contained in the dataset.

interface geared towards research. Due to the lack of existing suitable software for the Myo sensor, but given the publication of its bluetooth GATT protocol specification (see 2.1.3), some software would have to be written from scratch for it anyways. Since the sensor data from both sensors differs slightly and thus a common format needed to be created anyways it was decided to build one core program and then interface to the sensors using a plugin-like architecture, in theory only needing to write the common functionality needed for recording sessions once (see a discussion of the merit of this decision in 5). Using a plugin-based architecture was also thought to make it easier to add support for more types of sensors later, if needed. An overview of the software implemented is shown in figure 3.5.

To enable quick development and due to the availability of existing machine learning and networking libraries for it, Python was chosen as the programming language to develop the software. As a basis for the plugin architecture, a client-server model was implemented. In this model, the server is simply a kind of bus or central hub distributing information. Clients provide information or consume it, and register with the server accordingly. For each type of sensor a client was written to interface with the sensor, format its data to conform to a common format, and then transmit the data to all consumers, via the server. Two other clients were written, for saving and visualizing data, respectively. This way it is easy to both add more sensors and programs to use that data. WebSockets were chosen as networking protocol for their ease of use. The overview in figure 3.5 details which APIs were used for the sensor clients.

As mentioned above, a client was written to combine the sensor data from both types of sensors into a single file, resulting in one file per recording. Each file is labeled with its exercise, error type (correct or error type) and test subject id. The complete dataset containing data from all 5 test subjects contained 300 of these files. All sensor data available from the sensors was saved, at each sensor's maximum sampling rate. The simple real-time visualizer client that was developed in the hopes of detecting sensor timeouts by eyes is shown in figure 3.6.

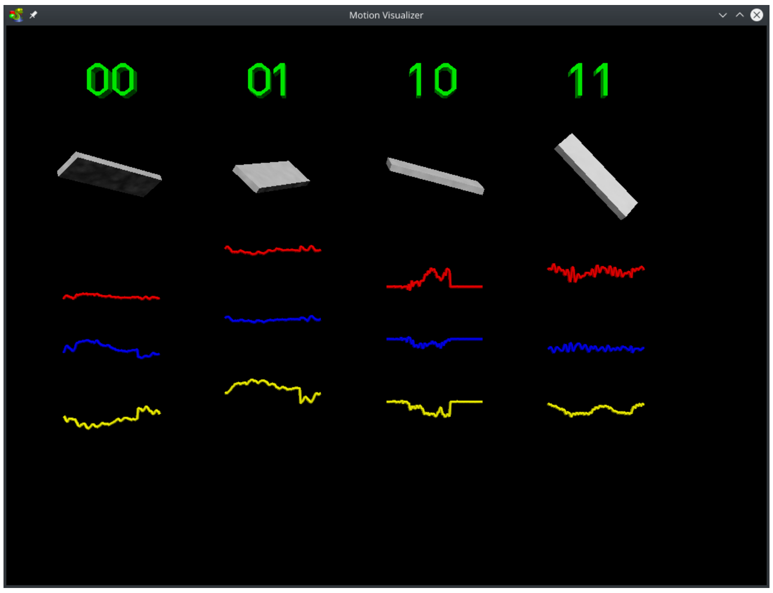


Figure 3.6: The simple real-time motion visualizer developed for the purposes of detecting sensor time-outs and making sure the sensors were working correctly.

Sensor calibration Another point worth mentioning is that of sensor calibration. By their very nature, the IMU sensors always provide relative rotations from some starting point, not absolute orientations in space. This starting point is determined by their orientation when they are turned on. Because of this, initially a calibration procedure was implemented. During this calibration phase a test subject would stand still in the starting position of the current exercise and the orientational data from the pose would be saved as an "offset", to calibrate all subsequent orientations against. This, it was thought, would make sure the data sequences of different recordings would be more comparable due to a common orientational starting point.

The decision was made to discard of the calibration procedure. First, every test subject by nature will have a slightly different stance and body shape, making completely equalized data sequences using calibration points impossible. Secondly, the Myo sensor was found to be drifting so much that calibration would be negated within a short amount of time. Thirdly, it was reasoned that calibration might be inherently unnecessary. The reasoning is as follows: Assuming there is no sensor drift and two recordings are performed exactly the same way and the only difference between the recordings is the starting orientations of the sensors, this "cost" or difference between sequences is a constant factor for each data point of a sequence. It follows that unless this constant cost is so great that it obscures the differences between classes of movements, which are not constant (exercises can be infinitely different from one another, due arbitrary lengths), the cost of not having calibration can be disregarded and calibration becomes unnecessary. The merit of this reasoning is discussed in chapter 5.

3.3.5 The dataset

Following the recording session protocols, recording the samples, using the sensors and software developed as described in the previous sections, the resulting dataset contains 300 labeled points of data, each consisting of a discrete-time series of sensor data from both types of sensors. An overview of the of the dataset obtained is shown in the tables 3.2 and 3.3. As can be seen in the tables, during some recording sessions more than 5 samples were recorded. This happened when there was reason for suspecting a sample had been "fouled" by some unintended mistake, but then after checking was fine after all. This dataset will be the basis of all the following machine learning related work.

Unfortunately, one test subject's sideways lean exercises had to be discarded since the placement of the sensors used for that recording session for that exercise was discovered to be incorrect. This happened at a time when the test subject in question was not able to re-record the exercises. Whether or not this will have an effect on the accuracy of the classification of this exercise or the error types is discussed below.

Table 3.2: Total number of samples recorded for each of the error types.

Error Type	Number of samples
Correct	73
Incorrect speed	70
Incorrect trajectory	71
Incorrect amplitude	72
Total	286

Table 3.3: Total number of samples recorded for each exercise.

Error Type	Number of samples
Biceps curl	101
Knee extension	104
Sideways lean	81
Total	286

3.4 Machine Learning

This section will deal with the machine learning techniques employed to analyze the dataset. First, general constraints are stated as to what kinds of results we are interested in. Then, the process of feature selection is discussed, followed by the machine learning algorithms run on those features. The results and testing are presented in 4.

3.4.1 Purpose

What kinds of answers we want from the machine learning algorithms determines which algorithms and techniques we use. In this thesis the main focus is not on classifying which exercise is performed (ie. biceps curl or knee extension), but rather to detect the occurrence of specific error types (ie. speed or amplitude), regardless of which exercise is being performed.

Thus, the main goals for the machine learning part of this thesis were decided to be the exploration of features and learning algorithms in the service of classifying error types, with the classification of exercise types as an added benefit. A special focus was put upon the role quaternions might play here, as to the authors knowledge little or no work has been done to explore the use of quaternions for classifying physiotherapy exercises.

3.4.2 Feature selection

The selection of features is crucial to obtain decent results from a classifier, as briefly explained in 2.3.4. The main and first focus in this thesis was upon the use of quaternions, or rather quaternion distance, as the basis for features. A very superficial overview of quaternions is presented in 2.2.

Quaternion discrete-time sequences

The error types chosen - speed, trajectory and amplitude - can all be thought to exhibit their individual characteristics in the orientation and time domains. An exercise performed too quickly will theoretically exhibit the same sequence of orientations as the correct variant, but at different times. An incorrect trajectory will differ in orientation, but not necessarily along the time domain. A difference in amplitude will exhibit a difference in both the time and orientation domain, making for an interesting combination. Thus, since the error types we are exploring in this thesis can be considered in the "orientation-time domain", the first candidates for features exploiting this lever knowledge were found in interpreting the sensor data in that context.

Every exercise in the dataset consists of a discrete-time series, where each entry in the series is comprised of the data supplied by the sensors at that time point. One of those data types, representing the orientation of the sensor at that point in time, is a quaternion. The quaternion at this point can be viewed as the result of a type of feature extraction applied to the combined output from the accelerometer, gyroscope and magnetometer sensors contained in each IMU, which yields a quaternion. This feature extraction is done in the hardware of each sensor (see 2.1.4 and 2.1.3). Since the quaternion combines the orientation information contained within the three other data types in a reduced state (one 4-element vector vs three 3-element vectors), this format was hypothesized as suitable. To account for the time domain, each feature would in effect be a discrete-time series, made up of quaternions, for each exercise. The relationship between two recorded exercises is thus the relationships between their quaternion sequences. The resulting feature-space is the space of quaternion discrete-time sequences, each sequence a feature. When using multiple sensors at once, the quaternion sequences belonging to each sensor can be said to form one feature, together. These features are compared using different proximity functions for the k-NN and DTW algorithms.

3.4.3 Learning algorithms

The algorithms in this section were implemented or provided by library functions in Python, as detailed in each sub-section.

k-Nearest Neighbors with Dynamic Time Warping

The first choice of learning algorithm fell to the k-nearest neighbor algorithm for its simplicity, as presented in 2.3.5. A range of values for k were experimented with (see results in 4.1). For the feature space of quaternion discrete-time sequences, a proximity function is needed to find the k nearest neighbors. Since every sequence in the dataset may differ in length, it would be useful to avoid interpolation or decimation to bring all sequences to an equal length. For this reason, dynamic time warping was chosen, as presented in 2.3.6. The DTW module used [8] is based on the fastDTW algorithm proposed by [22]. This is an approximation of DTW which still provides optimal or near-optimal alignment between two series, but at linear time and space complexity. The original DTW algorithms provides optimal alignment between data series, but at the cost of quadratic time and space requirements. The quadratic time requirement of the original DTW algorithm proved too costly for the sizes of quaternion sequences in the dataset used.

The DTW algorithm needs to compare the distance between individual elements of each sequence-pair that is compared. Each element in these sequences is a quaternion (see 3.4.2). Quaternions can be seen as simple 4-element vectors, but also exhibit the inherent mathematical properties of rotations in space as quaternions. As such, we can exploit these properties by comparing two quaternions in rotation space. In addition, we can also compare two quaternions viewing them as simple 4-element vectors in euclidean space. This, it was hypothesized, might indicate the value of choosing to exploit the nature of quaternions themselves for the problems in this thesis. Both of these methods were thus implemented as detailed briefly below.

A range of k-values was experimented with to find an optimum value, the results of which can be found in 4.1.

Multiple sensors per exercise Every recorded exercise, or sample, in the dataset consists of a one sequence of sensor data, per sensor. A maximum of three sensors were used during the actual recording of the dataset. This means that when we wish to compare the distance between two samples in the dataset, we can either compare only the sensor data sequences of one sensor or combine the distances to all pairs of sensors. In this thesis, both were implemented. Thus, a parameter was used to specify which sensor or sensors are to be used.

The scheme for determining the nearest neighbor for a combination of sensor sequences was defined as follows, where *sequenceA* and *sequenceB* are the discrete-time data sequences generated by sensors 0 and 1 for each sample, *dist* is the distance function used for measuring the distance between sequences, and the two samples are *u* abd *v*:

$$distance(\mathbf{u}, \mathbf{v}) = dist(u.sequence0, v.sequence0) + dist(u.sequence1, v.sequence1)$$

Thus, when using multiple sensors during classification, the combined distance for all their sequences for a sample will determine the final proximity. Said plainly, if one of the sensors we use matches a class very well, it might compensate for another sensor not matching so well. Only the sensors with the same id are ever compared to each other, since their placement on the body is equal.

When using only one sensor, for instance sensor 0, proximity is calculated only 0's sequences:

$$distance(\mathbf{u}, \mathbf{v}) = dist(u.sequence0, v.sequence0)$$

Euclidean distance When using euclidean distance to compare two elements *p* and *q*, the DTW algorithm was provided the distance function as follows:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2 + (p_4 - q_4)^2}$$

Quaternion distance When using quaternion distance as measured in rotation expressed as radians needed to get from one rotation, represented by quaternion *q*, to another rotation, represented by quaternion *p*, the following function was provided to the DTW algorithm:

$$d(\mathbf{p}, \mathbf{q}) = \arccos(2 * \langle q, p \rangle^2 - 1) = \theta$$

Chapter 4

Results and validation

In the sections below, the results for all techniques implemented and introduced in 3.4 are presented. Several cross-validation schemes were employed to produce these results and are presented first, to give context to the results themselves, which follow. Results and parameter choices are discussed in 5.1.1.

4.1 k-Nearest Neighbors with DTW

This section presents the results from classifying individual sequences from the dataset as belonging to one of the 4 possible error types: correct, incorrect speed, incorrect amplitude or incorrect trajectory, using k-NN with DTW. For completeness sake, for every technique employed for classification of the error types, the model was also run against the type of exercise itself, to classify an exercise as belonging to one of the three possible exercises (knee extension, biceps curl and sideways lean, see 3.3.2). The k-values used were $\mathbf{k} = (1, 3, 5, 7)$ (k is always chosen to be odd-numbered to avoid ties). Higher k-values were not run for all data as the trend seemed to show diminishing returns (see 5).

For reasons discussed further in 5.1.2, sensor 2 (The Myo sensor) was not used to generate results. As a consequence, the only combinations of the sensor parameter used are 0, 1 and (0,1).

For the different validation techniques, distance functions and parameter and sensor combinations the best result of each is presented in this chapter in the form of a confusion matrix. A confusion matrix shows what all samples in the dataset were classified as, and individual as well as an average accuracy score. The rest, or rather all, results are listed in tables 4.3 and 4.4, to conserve space and provide an overview of the trends for different parameter combinations. The tables show the parameters used for each result, and the result listed in the tables is the average accuracy score for that set of parameters.

Results are shown for both classification of the type of exercise (for instance knee extension) and classification of error types (correct or one of the error types).

4.1.1 Validation

Different forms of fold-validation were used as explained briefly in 2.3.7. To employ cross-validation for the use of k-NN, the dataset of exercises is split in different ways to explore the robustness of the model for classification and to take into account and test the nature of dataset. The reasoning behind the chosen folds follow below.

Leave-one-out validation

When we classify an exercise by looking for its closest neighbors using k-NN, we check against the entire dataset, excluding the exercise we are currently classifying. In other words, the training dataset is the entire dataset minus the entry we wish to classify, which becomes the testing dataset. This is then done for all entries in the dataset.

Leave-me-out-exactly validation

Every type of error type in the dataset is performed 5 times by each test subject. For every exercise, including its error types, 20 samples were recorded.

This means that for samples to be classified, there are 4 (or 19) other samples which can be assumed to be very similar, not just because they belong to the same class, but because they were performed by the same test subject as well.

This validation scheme aims to control for the similarity of recordings that is due to having been performed by the same test subject. Leave-me-out-exactly validation is defined as leaving out all samples in the dataset performed by the same test subject and belonging to the same category to be classified as the current test sample.

In other words, if a sample's type of exercise is to be classified and the sample is a biceps curl exercise and was performed by test subject A, the k-NN algorithm excludes all biceps curl samples performed by A from the training set for that sample.

Leave-me-out validation

This is the same as leave-me-out-exactly validation, except when classifying a sample, all exercises performed by the sample's test subject are excluded from the training set, not just those which also belong to that sample's true class.

4.1.2 Results

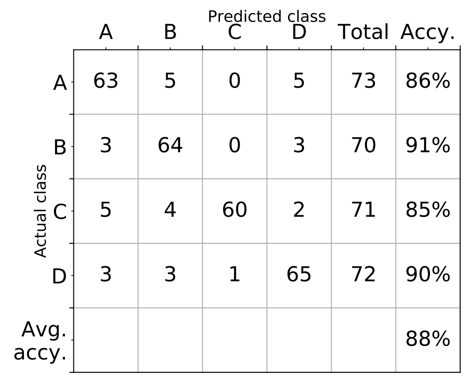
The final list of adjustable parameters is shown in table 4.1. Every possible combination was run for both exercise type and error type classification. This yielded 72 results per class type, and 144 results total. The accuracy of predicting for each class, as well as an average accuracy, were calculated for each combination.

Table 4.1: This table presents the final list of parameters which can be adjusted, each of their possible values, and the total number of possible combinations. The total number of combinations of parameters was run for both exercise type and error type classification.

Parameter	**Possible values**
distance function	euclidean, quaternion
k	1,3,5,7
validation scheme	Leave-one-out, leave-me-out, leave-me-out-exactly
sensors	0,1,(0,1)
Number of possible combinations	72

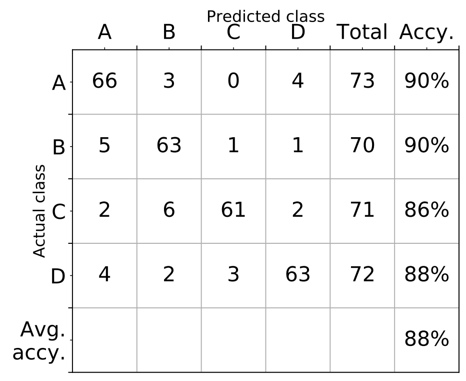
Figures 4.2 4.1 show the confusion matrices of the best results for the classification of the two class types. For each classification type, the best results obtained using each validation technique and distance function (quaternion vs euclidean) are presented. Tables 4.3 and 4.4 show all results with their corresponding parameters. Accuracy averages are rounded to 2-digit significance when not exactly 100%. All results, including every accuracy for every class per parameter combination can be found in the appendix at A.

Top error type classifications for every validation scheme and distance function



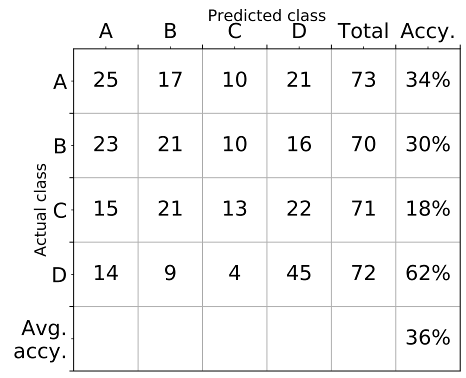
		Predicted class					
		A	B	C	D	Total	Accy.
Actual class	A	63	5	0	5	73	86%
	B	3	64	0	3	70	91%
	C	5	4	60	2	71	85%
	D	3	3	1	65	72	90%
	Avg. accy.						88%

(a)



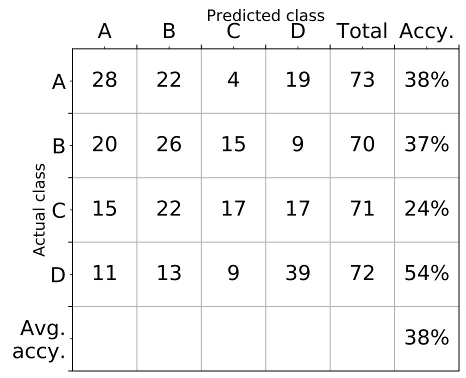
		Predicted class					
		A	B	C	D	Total	Accy.
Actual class	A	66	3	0	4	73	90%
	B	5	63	1	1	70	90%
	C	2	6	61	2	71	86%
	D	4	2	3	63	72	88%
	Avg. accy.						88%

(b)



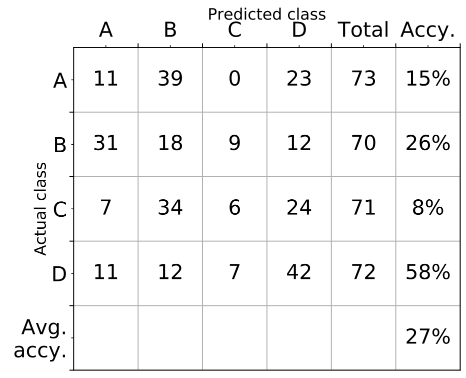
		Predicted class					
		A	B	C	D	Total	Accy.
Actual class	A	25	17	10	21	73	34%
	B	23	21	10	16	70	30%
	C	15	21	13	22	71	18%
	D	14	9	4	45	72	62%
	Avg. accy.						36%

(c)



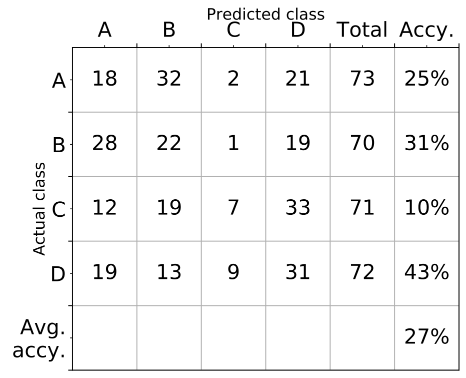
		Predicted class					
		A	B	C	D	Total	Accy.
Actual class	A	28	22	4	19	73	38%
	B	20	26	15	9	70	37%
	C	15	22	17	17	71	24%
	D	11	13	9	39	72	54%
	Avg. accy.						38%

(d)



		Predicted class					
		A	B	C	D	Total	Accy.
Actual class	A	11	39	0	23	73	15%
	B	31	18	9	12	70	26%
	C	7	34	6	24	71	8%
	D	11	12	7	42	72	58%
	Avg. accy.						27%

(e)

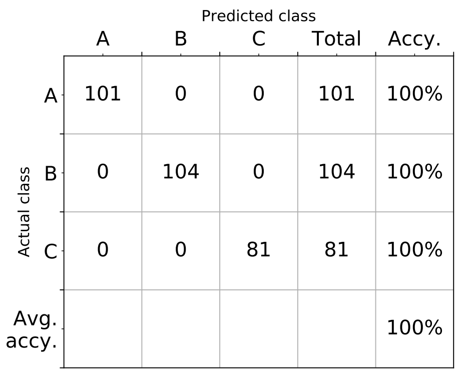


		Predicted class					
		A	B	C	D	Total	Accy.
Actual class	A	18	32	2	21	73	25%
	B	28	22	1	19	70	31%
	C	12	19	7	33	71	10%
	D	19	13	9	31	72	43%
	Avg. accy.						27%

(f)

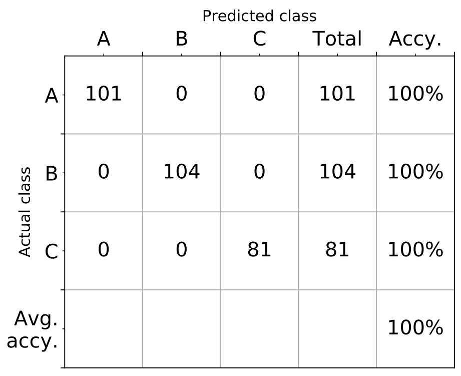
Figure 4.1: Best error type results for every validation scheme and distance function used. Category labels: A=correct, B=incorrect speed, C=incorrect trajectory and D=incorrect amplitude. Parameters, distance function, k-value, sensors used and validation scheme, for each:
a) quaternion, k=1, sensors=(0,1), leave-out-out.
b) euclidean, k=1, sensors=(0,1), leave-one-out
c) quaternion, k=1, sensors=(0,1), leave-me-out-exactly
d) euclidean, k=7, sensors=(0,1), leave-me-out-exactly
e) quaternion, k=1, sensors=(0,1), leave-me-out
f) euclidean, k=7, sensors=0, leave-me-out.

Top type of exercise classifications for every validation scheme and distance function


Predicted class
Actual class

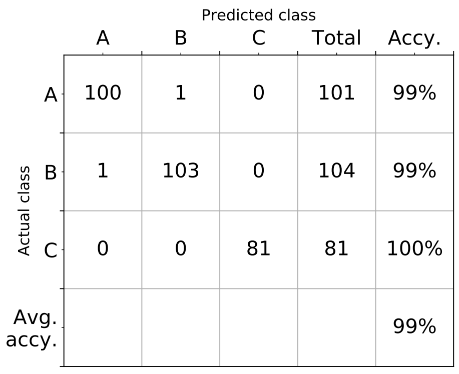
	A	B	C	Total	Accy.
A	101	0	0	101	100%
B	0	104	0	104	100%
C	0	0	81	81	100%
Avg. accy.					100%

(a)


Predicted class
Actual class

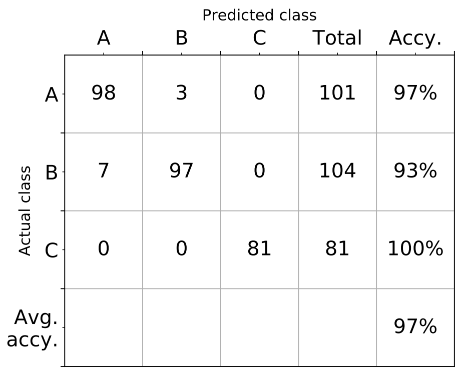
	A	B	C	Total	Accy.
A	101	0	0	101	100%
B	0	104	0	104	100%
C	0	0	81	81	100%
Avg. accy.					100%

(b)


Predicted class
Actual class

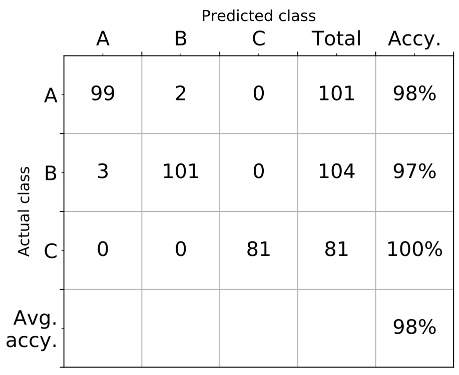
	A	B	C	Total	Accy.
A	100	1	0	101	99%
B	1	103	0	104	99%
C	0	0	81	81	100%
Avg. accy.					99%

(c)


Predicted class
Actual class

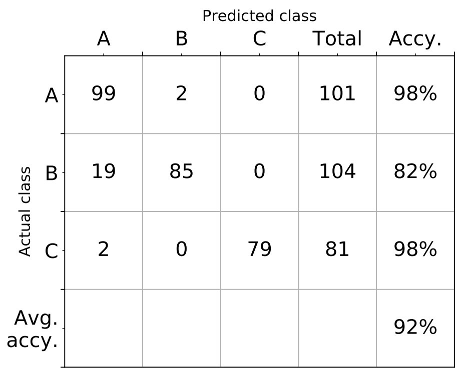
	A	B	C	Total	Accy.
A	98	3	0	101	97%
B	7	97	0	104	93%
C	0	0	81	81	100%
Avg. accy.					97%

(d)


Predicted class
Actual class

	A	B	C	Total	Accy.
A	99	2	0	101	98%
B	3	101	0	104	97%
C	0	0	81	81	100%
Avg. accy.					98%

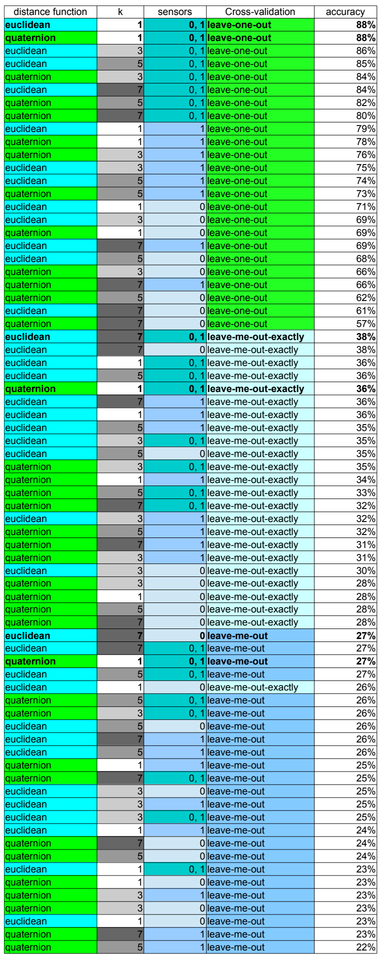
(e)


Predicted class
Actual class

	A	B	C	Total	Accy.
A	99	2	0	101	98%
B	19	85	0	104	82%
C	2	0	79	81	98%
Avg. accy.					92%

(f)

Figure 4.2: Best exercise type results for every validation scheme and distance function used. Category labels: A=biceps curl, B=knee extension, C=standing lean. Parameters, distance function, k-value, sensors used and validation scheme, for each:
a) quaternion, k=1, sensors=1, leave-one-out
b) euclidean, k=1, sensors=(0,1), leave-one-out
c) quaternion, k=1, sensors=1, leave-me-out-exactly
d) euclidean, k=1, sensors=1, leave-me-out
e) quaternion, k=7, sensors=(0,1), leave-me-out
f) euclidean, k=1, sensors=1, leave-me-out-exactly



distance function	k	sensors	Cross-validation	accuracy
euclidean	**1**	**0, 1**	**leave-one-out**	**88%**
quaternion	**1**	**0, 1**	**leave-one-out**	**88%**
euclidean	3	0, 1	leave-one-out	86%
euclidean	5	0, 1	leave-one-out	85%
quaternion	3	0, 1	leave-one-out	84%
euclidean	7	0, 1	leave-one-out	84%
quaternion	5	0, 1	leave-one-out	82%
quaternion	7	0, 1	leave-one-out	80%
euclidean	1	1	leave-one-out	79%
quaternion	1	1	leave-one-out	78%
quaternion	3	1	leave-one-out	76%
euclidean	3	1	leave-one-out	75%
euclidean	5	1	leave-one-out	74%
quaternion	5	1	leave-one-out	73%
euclidean	1	0	leave-one-out	71%
euclidean	3	0	leave-one-out	69%
quaternion	1	0	leave-one-out	69%
euclidean	7	1	leave-one-out	69%
euclidean	5	0	leave-one-out	68%
quaternion	3	0	leave-one-out	66%
quaternion	7	1	leave-one-out	66%
quaternion	5	0	leave-one-out	62%
euclidean	7	0	leave-one-out	61%
quaternion	7	0	leave-one-out	57%
euclidean	**7**	**0, 1**	**leave-me-out-exactly**	**38%**
euclidean	7	0	leave-me-out-exactly	38%
euclidean	1	0, 1	leave-me-out-exactly	36%
euclidean	5	0, 1	leave-me-out-exactly	36%
quaternion	**1**	**0, 1**	**leave-me-out-exactly**	**36%**
euclidean	7	1	leave-me-out-exactly	36%
euclidean	1	1	leave-me-out-exactly	36%
euclidean	5	1	leave-me-out-exactly	35%
euclidean	3	0, 1	leave-me-out-exactly	35%
euclidean	5	0	leave-me-out-exactly	35%
quaternion	3	0, 1	leave-me-out-exactly	35%
quaternion	1	1	leave-me-out-exactly	34%
quaternion	5	0, 1	leave-me-out-exactly	33%
quaternion	7	0, 1	leave-me-out-exactly	32%
euclidean	3	1	leave-me-out-exactly	32%
quaternion	5	1	leave-me-out-exactly	32%
quaternion	7	1	leave-me-out-exactly	31%
quaternion	3	1	leave-me-out-exactly	31%
euclidean	3	0	leave-me-out-exactly	30%
quaternion	3	0	leave-me-out-exactly	28%
quaternion	1	0	leave-me-out-exactly	28%
quaternion	5	0	leave-me-out-exactly	28%
quaternion	7	0	leave-me-out-exactly	28%
euclidean	**7**	**0**	**leave-me-out**	**27%**
euclidean	7	0, 1	leave-me-out	27%
quaternion	**1**	**0, 1**	**leave-me-out**	**27%**
euclidean	5	0, 1	leave-me-out	27%
euclidean	1	0	leave-me-out-exactly	26%
quaternion	5	0, 1	leave-me-out	26%
quaternion	3	0, 1	leave-me-out	26%
euclidean	5	0	leave-me-out	26%
euclidean	7	1	leave-me-out	26%
euclidean	5	1	leave-me-out	26%
quaternion	1	1	leave-me-out	25%
quaternion	7	0, 1	leave-me-out	25%
euclidean	3	0	leave-me-out	25%
euclidean	3	1	leave-me-out	25%
euclidean	3	0, 1	leave-me-out	25%
euclidean	1	1	leave-me-out	24%
quaternion	7	0	leave-me-out	24%
quaternion	5	0	leave-me-out	24%
euclidean	1	0, 1	leave-me-out	23%
quaternion	1	0	leave-me-out	23%
quaternion	3	1	leave-me-out	23%
quaternion	3	0	leave-me-out	23%
euclidean	1	0	leave-me-out	23%
quaternion	7	1	leave-me-out	23%
quaternion	5	1	leave-me-out	22%

Figure 4.3: All results for error type (correct/type of incorrect) classification, sorted by average accuracy. The entries in bold font correspond to the confusion matrices in figure 4.1.

distance function	k	sensors	Cross-validation	accuracy
euclidean	**1**	**0, 1**	**leave-one-out**	**100%**
euclidean	3	0, 1	leave-one-out	100%
euclidean	5	0, 1	leave-one-out	100%
quaternion	**1**	**1**	**leave-one-out**	**100%**
quaternion	1	0, 1	leave-one-out	100%
quaternion	3	0, 1	leave-one-out	100%
quaternion	5	0, 1	leave-one-out	100%
quaternion	7	0, 1	leave-one-out	100%
euclidean	1	1	leave-one-out	100%
euclidean	7	0, 1	leave-one-out	100%
quaternion	3	1	leave-one-out	100%
quaternion	**1**	**1**	**leave-me-out-exactly**	**99%**
quaternion	1	0, 1	leave-me-out-exactly	99%
quaternion	3	1	leave-me-out-exactly	99%
quaternion	3	0, 1	leave-me-out-exactly	99%
quaternion	5	1	leave-one-out	99%
quaternion	7	0, 1	leave-me-out-exactly	99%
quaternion	5	0, 1	leave-me-out-exactly	99%
euclidean	3	1	leave-one-out	99%
euclidean	5	1	leave-one-out	99%
euclidean	7	1	leave-one-out	99%
quaternion	7	1	leave-one-out	99%
quaternion	5	1	leave-me-out-exactly	98%
quaternion	**7**	**0, 1**	**leave-me-out**	**98%**
quaternion	7	1	leave-me-out-exactly	98%
quaternion	1	0, 1	leave-me-out	98%
quaternion	3	0, 1	leave-me-out	98%
quaternion	5	0, 1	leave-me-out	98%
quaternion	7	1	leave-me-out	98%
quaternion	3	1	leave-me-out	97%
quaternion	5	1	leave-me-out	97%
quaternion	1	1	leave-me-out	97%
euclidean	**1**	**1**	**leave-me-out**	**97%**
euclidean	3	1	leave-me-out	96%
euclidean	5	1	leave-me-out	95%
euclidean	7	1	leave-me-out	94%
euclidean	1	0, 1	leave-me-out	94%
euclidean	3	0, 1	leave-me-out	94%
euclidean	5	0, 1	leave-me-out	94%
euclidean	7	0, 1	leave-me-out	94%
euclidean	**1**	**1**	**leave-me-out-exactly**	**92%**
euclidean	3	1	leave-me-out-exactly	91%
euclidean	5	1	leave-me-out-exactly	90%
euclidean	1	0, 1	leave-me-out-exactly	89%
euclidean	3	0, 1	leave-me-out-exactly	89%
euclidean	5	0, 1	leave-me-out-exactly	88%
euclidean	7	1	leave-me-out-exactly	87%
euclidean	7	0, 1	leave-me-out-exactly	85%
quaternion	1	0	leave-one-out	85%
euclidean	1	0	leave-one-out	83%
quaternion	3	0	leave-one-out	82%
euclidean	3	0	leave-one-out	79%
quaternion	5	0	leave-one-out	79%
quaternion	7	0	leave-one-out	77%
euclidean	5	0	leave-one-out	77%
euclidean	7	0	leave-one-out	70%
quaternion	1	0	leave-me-out	60%
quaternion	7	0	leave-me-out	59%
quaternion	3	0	leave-me-out	59%
quaternion	5	0	leave-me-out	57%
quaternion	1	0	leave-me-out-exactly	57%
quaternion	7	0	leave-me-out-exactly	57%
quaternion	3	0	leave-me-out-exactly	56%
euclidean	1	0	leave-me-out-exactly	56%
quaternion	5	0	leave-me-out-exactly	54%
euclidean	1	0	leave-me-out	53%
euclidean	5	0	leave-me-out-exactly	52%
euclidean	3	0	leave-me-out-exactly	51%
euclidean	5	0	leave-me-out	50%
euclidean	7	0	leave-me-out	50%
euclidean	3	0	leave-me-out	50%
euclidean	7	0	leave-me-out-exactly	46%

Figure 4.4: All results for classification by exercise type (biceps, knee extension, sideways lean), sorted by average accuracy. The entries in bold font correspond to the confusion matrices in figure 4.2.

Part III

Conclusion

Chapter 5

Discussion

In this chapter, the results presented in 4 will be interpreted and discussed. Furthermore, high-level decisions made during the implementation phase of this thesis are evaluated seen in the context of the results.

5.1 General discussion

5.1.1 Results of k-nearest neighbor and dynamic time warping

When we look at the complete results for exercise and error type classification in tables 4.4and 4.3, we can discern several trends and patterns. It should be kept in mind that when accuracies in general are talked about, what is referred to are the average accuracies per combination of distances function,sensors used, k-value and cross-validation scheme.

Discussion of parameters and their effects on the results

To begin with, it is clear that error type (correct or error type) classification seems to yield far lower accuracy rates than the classification of only which exercise was performed. The top 10% of the results for error type classification lie between 82%-88%, whereas the top 10% results of exercise type classification all show 100% accuracy. Furthermore, the range of average accuracy for error type classification at 66% is greater than that for exercise classification at 54%, indicating a steeper drop of average accuracy.

To dig deeper, when one looks at the top results for both exercise and error type classification, we see that they were obtained using leave-one-out cross-validation.

For error type classification especially, there's a clear stratification of validation schemes for the results (see figure 5.2), sorted by accuracy. The top segment of results is all leave-one-out, the middle is composed of leave-me-out-exactly and the bottom is populated by leave-me-out. Within the stratification of the validation schemes, we can discern similar stratification for which sensors were used. This is especially clear in with the leave-one-out segment, where the top part is dominated by sensors=(0,1), followed by sensors=1 and then mostly sensors=0 at the bottom. The same general pattern seems to hold for the middle layer, leave-me-out-exactly validation.

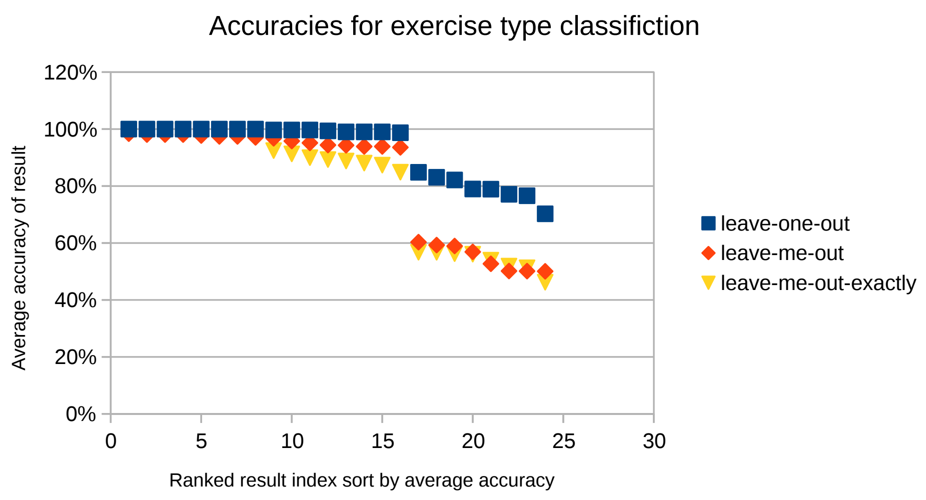


Figure 5.1: Scatterplot showing the stratification of results by validation scheme for exercise classification.

The bottom layer of leave-me-out seems to still generally show that using a combination of sensors 0 and 1 gives better results, but the sensor combinations are more scattered, with only a general indication of sensor=0 having worse performance.

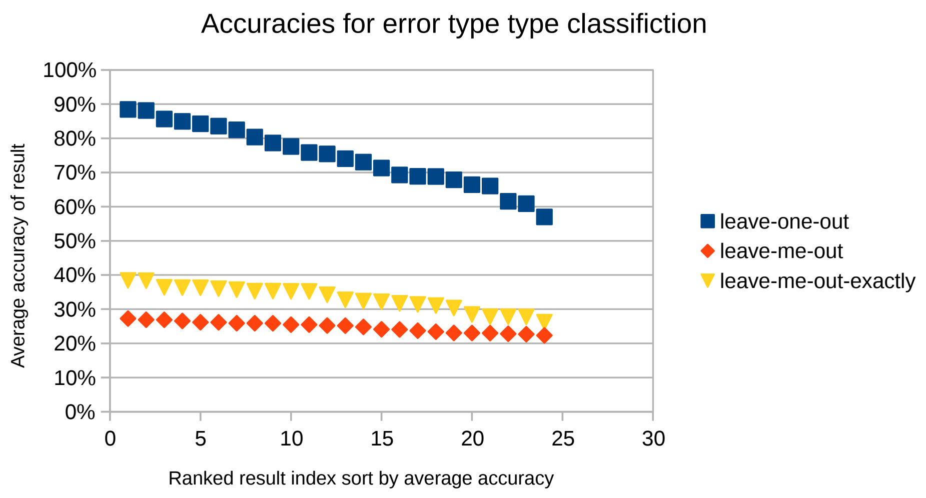


Figure 5.2: Scatterplot showing the stratification of results by validation scheme for error type classification.

For exercise classification, some stratification of validation schemes still seems to exist (see figure 5.1), but it is more mixed, and the effects of the validation scheme seem to be overcome somewhat by the effects of distance function and sensor(s) used. Whereas for error type classification there seems to be no discernible stratification of the type of distance function used, for general exercise classification there is a slightly stronger stratification of distance function, with the quaternion function being over represented in the top 50% of the results. However, the significance of this is questionable, since the total range of accuracy for exercise classification is only about 54% (46%-100%), with the top half of the accuracies only having a range of about 6%, and the stratification of the distance function still being somewhat mixed. A stronger stratification can be discerned for which sensors were used. Here, the same trend can be detected as for error type classification, with the sensors=(0,1) more or less dominating the upper regions of every layer of stratification of validation scheme. The difference between the accuracies using sensors (0,1) or 1 and using sensor 0 only are especially noteworthy, with almost the entire lower third of the results having been obtained using sensor 0 only. In this region the accuracy rate also starts dropping a lot faster, with a range of 40%, contrasted to the 15% spanned by the results above it.

Finally, the k-value seems to lead to no clear stratification in the results. It seems however that a value of k=1 consistently obtained a higher accuracy within the various strata of the other parameters compared to higher values of k. The same seems to roughly hold for the k-values of 3,5 and 7, with a lower k-value generally seeming to give better results.

Summary We might summarizes as follows:

- Leave-one-out correlates with far greater average accuracy than the other validation schemes. The difference is especially marked for error type classification.
- A stark stratification of validation schemes can be observed in the sorted average results
- Exercise classification sees far better average accuracies compared to error type classification overall
- A sensor-value of 1 scores higher than sensor=0, but sensors=(0,1) score higher still.
- Lower values of k seem to be preferable to high values of k

Interpretation of the results

The model The biggest effect on accuracy seems to come from the type of validation scheme used. To interpret this effect, one must regard the methods of validation in the context of the nature of the dataset.

To elaborate, when using leave-one-out validation, there will always be another sample of the same class, recorded by the same test subject, in the training set. This is then similar to having recorded a “template exercise” for every class by each test subject, as was done in [31], briefly presented in 2.4. When using leave-me-out or leave-me-out-exactly validation, all samples of the same class and recorded by the same test subject will be excluded from the training set, in effect leading to classification without the use of template exercises.

From this, some important points can be drawn regarding the fitness of the algorithm and features used for classifying exercises and error types. First, that the algorithm works well for classifying the exercise type, even without template exercises, exemplified by the results obtained not using leave-one-out. In other words, even without template exercises there is a clear clustering of samples belonging to the same exercise type in this model, independent of test subject and error type. Second, the algorithm and features do not work well for classifying error types. When classifying error types with template exercises (leave-one-out), the samples do cluster according to their class, but those clusters are highly dependent on having a template exercise or rather, they are dependent on test subject and exercise type. This becomes clear when classifying error types without a template exercise, as the accuracies drop dramatically (from 84% with templates (leave-one-out) to 38%(not leave-one-out)).

This leads us to conclude that when classifying error types successfully with the use of template exercises, what is actually happening is not classification of error types in general. We are interested in classifying the error types (correct, incorrect speed/trajectory/amplitude) independently from the exercise type they occur in or the test subject that performed them. What we seem observe, however, is the classification of ″emerging“ classes instead. A class emerges for each combination of error type, exercise type and test subject, and this is what the samples cluster around when using leave-one-out. The distance, in orientational terms, between two of these emerging classes (for example ″correct-biceps by test subject A“ vs ″correct-biceps by test subject B“) can be assumed to usually be much smaller than between two regular exercises types (for example biceps vs knee extension). Thus, we would expect that even with the use of template exercises for error type classification, the top accuracy would be lower than that for just exercise classification, since noise will have a greater chance of affecting a prediction. That is exactly what can be observed from the results, since, with leave-one-out (ie. template exercises), the top error type classification result is 88% while the top exercise classification result for the same validation scheme is 100%.

The difference in accuracy between results using leave-me-out and leave-me-out-exactly is slightly puzzling at first glance, as both eliminate what would correspond to a template exercise from the training set, but leave-me-out-exactly consistently performs better than leave-me-out, though, in the case of error type classification, not by much. The reason for this is unclear, however one possibility might be that since leave-me-out-exactly still retains the other samples from the current sample's test subject in the training set, more correct classifications can occur in the cases where error types are sufficiently distinct to a test subject to allow for correct predictions.

Value of orientational exercise representation Finally, with these interpretations of the results, some conclusions can be drawn about the worth of using features based on orientational time series as the basis for classifying exercises and error types. The features used by the k-NN algorithm were the distance between quaternion sequences (or the sequences themselves, depending on how one wishes to define it), and each quaternion sequence was located in this feature space.

From the results, it would seem that series of orientational data serve well to represent the characteristics needed to distinguish one specific sequence of movement, an exercise for instance, from another. They do not seem to serve well, however, to represent the types of exercise errors this thesis was set to investigate. Instead, trying to differentiate between different error types using orientational data in the way this thesis did, seems only to generate more fine-grained clusters around the corresponding "test subject-exercise-error type" combination. Therefor, they would seem to serve well only to detect specific versions of the exercises, and not to detect the error types independent of type of exercise or a specific test subject.

The role of the sensors Three sensors were used for building the data set. Due to technical difficulties discovered too late, the results were generated using only two of the sensors, sensor 0 and sensor 1. These are both LPMS-B sensors. As can bee seen in the results, using only sensor 0 yielded a significantly lower accuracy regardless of the other parameters. It is assumed that the reason for this is that the body placement of the sensor for all exercises captured less or even generally too little of the movements characteristic of each exercise. Combining both sensors however yielded higher accuracies than only using sensor 1, which seems to validate the use sensor 0 as a supporting factor for classification. In the case of error type classification, combining both sensors improved accuracy from using only sensor 1 at 79% to 88%. The results for exercise type classification are so tightly grouped and high-scoring that it seems difficult to argue that that a sensor combination of (0,1) scores higher than only sensor 1, since the difference in score is miniscule.

The lack of a calibration procedure for the sensors seems not to have negatively impacted exercise type classification in any significant way. Exercise classification, for both leave-me-out and leave-me-out-exactly validation, managed to achieve top average accuracies of 99%.

The lack of a calibration procedure might have impacted the results for error type classification more strongly, since, as posited below, error type classification seems to operate on much smaller margins of differences between samples. Since it by its nature is constant with sequence length, it likely was not the deciding factor for error type classification achieving lower accuracy scores in general.

The missing exercise recordings from one of the test subjects As mentioned in 3.3.5, the recordings for the sideways lean exercise had to be discarded for one of the test subjects. It had been discovered very late that the sensor placement for that exercise, for that test subject, had been incorrect. The results however seem not to be very much affected by this, as the accuracies for the sideways lean exercise are approximately the same as those for the others, on average, nor did it obscure the effects discussed above, self-evidently. Of course, it would have been preferable not to lose those samples from the dataset. However, the reasons for the low rates of accuracy for the classification of error types are presumably, as discussed above, not due to a lack of samples, and the accuracy rates for the classification of exercise type is high regardless of the samples missing.

Summary Again, one might summarize:

- The features and algorithms are well-suited to classify exercise types, even without having template exercises recorded by the same test subject in the training dataset.
- The features and algorithms are not well-suited to classify error types generally. Using template exercises only gives the illusion of good prediction, unmasked by the validation schemes of leave-me-out and leave-me-out-exactly. The clustering of samples actually happens around parameter combinations of "exercise-error type-test subject", not error types in general.
- Sensor calibration seems to be unnecessary for classifying on discrete-time orientation-data sequences as used in this thesis, provided that the features and model itself are a good fit.

5.1.2 Decisions and lessons learned

This section presents some discussion on the merit of various decisions taken and choices made during the work of this thesis, seen in the light of the results and their interpretations presented in the preceding section.

Building your own dataset with real people

Building your own dataset presents both benefits and challenges. It enables one to shape the dataset such that one might define one's own classes and types to test for, as well as make sure that the necessary raw sensor data exists within it to accomplish a task. On the other hand, when working under a time constraint, this will lead to there being less time remaining to actually implement the machine learning methods. As the final implementation is strictly dependent on having a dataset at all, dataset recording delays may delay the entire project.

In the end, the choice of building one's own dataset was still easy one, as there was no other ready-made dataset to be found providing the data this thesis required.

Choice of exercises The exercises chosen for this thesis were the biceps curl, knee extension and sideways lean. As far as can be gleaned from the results obtained, these exercises are well suited for this thesis. They operate on different regions of the body, while the knee extension is similar enough in principle to the biceps curl to investigate the robustness of the chosen features to distinguish between them. The sideways lean exercise seems the most subtle, requiring little in the way of bending joints in relation to each other, but rather exhibits general movement patterns followed by the entire body. In theory, these exercises should also be well suited to test for the various error types themselves, as the exercises in their correct form are easily distinguished from their incorrect forms. However, the latter is not confirmed by this thesis, since, as the results show, the algorithms and features chosen were not able to satisfactorily classify error types independent of exercise and test subject. This is not an indication that the exercises are ill-suited for this, only that the combination of exercises, features, learning algorithm and error types in this thesis were not able to achieve this goal. It is hypothesized by the author that the fault does not lie with the exercises themselves, but with the bad fit of the chosen features.

Number of exercises, number of repetitions, number of test subjects During the work of building the dataset by recording exercises, it was found that most time was spent preparing the session and then training new test subjects who had not participated in the recordings before. It was found that, after already having participated once, when re-recording a set of samples everything went much faster. As such, the number of test subjects is in hindsight considered sufficient, as the number is high enough to show the robustness of the algorithms and features chosen when using leave-me-out(-exactly) validation. More test subjects likely would not have made the effect of leave-me-out starker, but would have increased the workload for building of the dataset.

In light of the number of comparatively few test subjects, a high number of 10 repetitions per exercise and 5 repeats of each exercise recording seems a good trade-off.

Even though it sufficed in this case, only having 3 exercises is probably on the very low end of what is useful. Having more exercises might not have made any of the trends discussed in the previous sections any clearer, but having only 2 exercises would have been entirely pointless. If possible, thus, it would probably have been advisable to use more exercises than 3, in case one proved to have inherent flaws.

Placement of sensors The placement of the sensors can mostly be judged as acceptable. It can be argued that by having a unique placement of sensors for each type of exercise, each exercise's distinct movement patterns were accounted for and captured by the sensors, instead of hoping for the best by having one sensor placement layout for all exercises. One sensor placement layout for all exercises might still be preferable however, if more sensors were available to cover all body segments involved in all the exercises. However, it would then become necessary to test a sample against all available sensors, instead of only testing against the sensor or sensor which are known to, in the case of specialized placement for each exercise, capture the most characteristic parts of an exercise's movement.

In any case, by always having a sensor placed to capture the main part of a movement and one that does less so, it became apparent in the results section that in fact it is important having a sensor covering the main body segments involved in an exercise, at least with the features and learning algorithm chosen in this work. Furthermore, it was shown that having a weakly-predicting sensor placement can be offset by a stronger one, and that even a strongly predicting sensor placement might be enhanced by a supporting sensor.

Choice of software architecture The building of software to interface with both types of sensors in real-time to enable the saving of data to a common and unified format consumed a large amount of time during the work of this thesis. In hindsight, and in terms of the experiences made with the two types of sensors, it would have been more advisable to first stress-test both types of sensors in practice during real recording sessions, each on their own. It might then have been discovered that only using one type of sensor was beneficial, saving a lot of work to be spent on other areas instead. Once built however, the plugin or client-server architecture proved to be a success in that it was easy to connect both sensors using their separate APIs and stream their data to one synchronized data sequences, as well as easily implementing a real-time motion visualizer.

Choice of sensor hardware In theory, both types of sensors are ideal for the work in this thesis. In practice, the Myo armband sensor proved prone to technical difficulties such as timing out without a disconnect being advertised, making it hard to detect a disconnect while recording a session. Additionally, the two Myo sensor's used during the work of this thesis presented very high amounts of angular drift, especially compared to the LPMS-B sensor. The issue was further complicated by the two Myo sensors manifesting different technical problems from each other, due to as it was later discovered different hardware firmware versions. These difficulties ultimately lead to the discarding of the sensor data from the Myo sensor from the dataset in its entirety, since it was deemed too unreliable and was found to have "holes" at random places in the sequences in the dataset.

As mentioned regarding the choices of software architecture, in hindsight it would have been beneficial to stress-test both sensor types under real recording-session conditions, before any investment in real-time unifying of data streams was made.

Machine learning

Choice of features The choice of quaternion sequences as features was an alluring one. Most work reviewed by the author for this series uses the accelerometer, gyroscope or magnetometer data from IMU sensors directly. Investigating the use of quaternions, readily available as precomputed by the IMU sensors, was hypothesized as a potential feature extraction method reducing the computational complexity necessary. It was also thought that investigating the use of quaternions themselves would be interesting.

As a matter of fact, using quaternion sequences to classify specific movement sequences such as exercises seems to work very well, both using euclidean and quaternion distance functions to compare the proximity of quaternion sequences. However, as the results show, using them, at least with k-NN and DTW, does not provide high accuracies when classifying properties of the movement sequences themselves, which is what the error types are.

Choice of learning algorithms Using k-NN as a simple learning algorithm was hypothesized to be a good starting point for investigating the value of quaternion sequences as features, and this was born out by the results. Further, using DTW to compare the features themselves for use with k-NN seemingly proved a good idea, albeit lacking comparison with alternatives. DTW proved a good solution to comparing the features, since they consist of sequences of varying lengths.

Both distance functions employed, euclidean and quaternion, performed well, with no clear winner in terms of accuracy. Computationally however, the time required by the euclidean distance function (295 minutes) is roughly 140% the time required by the quaternion distance function (209 minutes) when calculating all distances for the entire dataset on the same machine. It would therefor seem like quaternion distance is preferable when quaternions are already pre-computed and when reducing computational complexity is desirable.

Choice of Validation schemes The validation schemes employed proved highly important to the results and their interpretation for in this thesis. Using only leave-me-out validation it would seem that the average accuracies for the classification of error types is a lot higher than is really the case, assuming one is interested in independent "true" classification of the error types, without always having a template exercise of the exact exercise, error type and test subject (or user) combination available in the training set. The use of leave-me-out and leave-me-out-exactly illustrated the underlying workings of the algorithms with the use of quaternion sequences as "only" classifying specific movement sequence, seemingly not being conducive to classifying general characteristics of movements, such as the error types.

Chapter 6

Conclusion

6.1 Conclusion

The goals stated for this thesis in chapter 1 were:

- Evaluating and selecting suitable motion capture sensors.
- Implementing a software pipeline to interface with the sensors and allow real-time streaming or saving of sensor data.
- Designing an exercise program consisting of typical exercises used for physical therapy to use with the system.
- Designing and then executing a recording session protocol in order to obtain testing and training data from test subjects performing the exercise program.
- Implementing machine learning techniques to classify exercises and error types recorded during the recording sessions.
- Train and test the algorithms using suitable validation schemes.

The context for the goals of this thesis is the growing need for solutions enabling patients to undergo therapy, treatment or perform simple preventative exercise at home by guiding them automatically to some degree, without constant help and supervision by professionals.

Even though a perfect autonomous system for evaluating exercises intelligently has not yet become reality, strictly speaking this thesis achieved what it set out to do. Sensors were evaluated and selected. A dataset was successfully built, albeit suffering from technical difficulties involving the sensors, requiring triage on the dataset. An exercise program was designed and incorporated into a recording session protocol. Machine learning techniques were implemented to classify exercises and error types. The use of quaternion sequences and different proximity functions for use with k-NN and DTW was investigated. The models were validated and tested successfully, uncovering more-or-less hidden structure and meaning underlying the data. While the classification of exercise types yielded satisfactory results, the same can not be said for using quaternion sequences to classify error types independently from other factors, at least the way they were used in this thesis. Error type classification did yield satisfactory results when using what amounts to template exercises. Furthermore, the different validation schemes made it possible to more precisely describe the best uses of quaternion sequences in the context of exercise classification. This also made the role of template exercises evident when trying to classify error types using these methods. Lastly, the results indicate that sensor calibration is unnecessary if the model itself is a good enough fit for the problem.

In hindsight, though the above results provided useful insights into the use of quaternions for exercise classification, less time could have been spent on needless sensor integration and the technical difficulties involved with the recording sessions. More time could have been spent on discovering other features more amenable to being used to represent the error types that were investigated. Even though reliable error type classification without using template samples was not achieved, the classification of exercises and error types yielded some promising results and granted a deeper understanding of the problem domain.

6.2 Future work

Future work in this area and related to the original motivation of this thesis might include an extended search for features which can be used to represent the error types discussed. It has become clear to the author that many, if not all, error types seem only to be representable in combination with the correct incarnation of an exercise. An incorrect trajectory is defined relative to its correct counterpart, and might have no other characteristics marking it as inherently erroneous. An interesting idea would thus be to not attempt to classify error types in isolation of the knowledge of which exercise they are ocurring in, but take the knowledge of the base exercise and then have a training set comprised of that exercise, containing different error types, and then classifying on that using features more suitable for error type classification. These feature would need to embody the relative nature of error types, being a function of both the correct type of an exercise and the the incorrect type.

Once more optimal features have been discovered, future work might also include real-time implementations based on them.

Part IV

Appendix

Appendix A

Raw results

Table A.1: All results for exercise classification. A=biceps curl, B=knee extension, C=sideways lean.

distance function	k	sensors	validation	A	B	C	avg. acc.
quaternion	1	1	leave-one-out	100%	100%	100%	100%
quaternion	1	0, 1	leave-one-out	100%	100%	100%	100%
quaternion	3	0, 1	leave-one-out	100%	100%	100%	100%
quaternion	5	0, 1	leave-one-out	100%	100%	100%	100%
quaternion	7	0, 1	leave-one-out	100%	100%	100%	100%
euclidean	1	0, 1	leave-one-out	100%	100%	100%	100%
euclidean	3	0, 1	leave-one-out	100%	100%	100%	100%
euclidean	5	0, 1	leave-one-out	100%	100%	100%	100%
quaternion	3	1	leave-one-out	99%	100%	100%	100%
euclidean	1	1	leave-one-out	99%	100%	100%	100%
euclidean	7	0, 1	leave-one-out	99%	100%	100%	100%
quaternion	1	1	leave-me-out-exactly	99%	99%	100%	99%
quaternion	1	0, 1	leave-me-out-exactly	99%	99%	100%	99%
quaternion	3	1	leave-me-out-exactly	99%	99%	100%	99%
quaternion	3	0, 1	leave-me-out-exactly	99%	99%	100%	99%
quaternion	5	1	leave-one-out	99%	99%	100%	99%
quaternion	7	0, 1	leave-me-out-exactly	99%	99%	100%	99%
quaternion	5	0, 1	leave-me-out-exactly	99%	98%	100%	99%
euclidean	3	1	leave-one-out	97%	100%	100%	99%
euclidean	5	1	leave-one-out	97%	100%	100%	99%
euclidean	7	1	leave-one-out	97%	100%	100%	99%
quaternion	7	1	leave-one-out	99%	97%	100%	99%
quaternion	5	1	leave-me-out-exactly	98%	97%	100%	98%
quaternion	7	0, 1	leave-me-out	98%	97%	100%	98%
quaternion	7	1	leave-me-out-exactly	98%	97%	100%	98%
quaternion	1	0, 1	leave-me-out	98%	96%	100%	98%
quaternion	3	0, 1	leave-me-out	98%	96%	100%	98%
quaternion	5	0, 1	leave-me-out	98%	96%	100%	98%
quaternion	7	1	leave-me-out	96%	97%	100%	98%
quaternion	3	1	leave-me-out	96%	96%	100%	97%
quaternion	5	1	leave-me-out	96%	96%	100%	97%
quaternion	1	1	leave-me-out	95%	96%	100%	97%
euclidean	1	1	leave-me-out	97%	93%	100%	97%
euclidean	3	1	leave-me-out	96%	91%	100%	96%
euclidean	5	1	leave-me-out	94%	91%	100%	95%
euclidean	7	1	leave-me-out	92%	92%	99%	94%
euclidean	1	0, 1	leave-me-out	93%	92%	98%	94%
euclidean	3	0, 1	leave-me-out	93%	92%	96%	94%
euclidean	5	0, 1	leave-me-out	93%	92%	96%	94%
euclidean	7	0, 1	leave-me-out	93%	91%	96%	94%

distance function	k	sensors	validation	A	B	C	avg. acc.
euclidean	1	1	leave-me-out-exactly	98%	82%	98%	92%
euclidean	3	1	leave-me-out-exactly	97%	82%	95%	91%
euclidean	5	1	leave-me-out-exactly	93%	82%	95%	90%
euclidean	1	0, 1	leave-me-out-exactly	95%	83%	90%	89%
euclidean	3	0, 1	leave-me-out-exactly	95%	84%	88%	89%
euclidean	5	0, 1	leave-me-out-exactly	94%	84%	86%	88%
euclidean	7	1	leave-me-out-exactly	91%	81%	90%	87%
euclidean	7	0, 1	leave-me-out-exactly	95%	82%	78%	85%
quaternion	1	0	leave-one-out	87%	85%	83%	85%
euclidean	1	0	leave-one-out	86%	78%	85%	83%
quaternion	3	0	leave-one-out	83%	82%	81%	82%
euclidean	3	0	leave-one-out	78%	76%	83%	79%
quaternion	5	0	leave-one-out	81%	74%	81%	79%
quaternion	7	0	leave-one-out	79%	73%	79%	77%
euclidean	5	0	leave-one-out	73%	75%	81%	77%
euclidean	7	0	leave-one-out	67%	73%	70%	70%
quaternion	1	0	leave-me-out	78%	64%	38%	60%
quaternion	7	0	leave-me-out	75%	61%	42%	59%
quaternion	3	0	leave-me-out	79%	61%	37%	59%
quaternion	5	0	leave-me-out	75%	60%	36%	57%
quaternion	1	0	leave-me-out-exactly	77%	68%	25%	57%
quaternion	7	0	leave-me-out-exactly	67%	68%	35%	57%
quaternion	3	0	leave-me-out-exactly	73%	68%	27%	56%
euclidean	1	0	leave-me-out-exactly	66%	67%	35%	56%
quaternion	5	0	leave-me-out-exactly	71%	63%	27%	54%
euclidean	1	0	leave-me-out	66%	51%	41%	53%
euclidean	5	0	leave-me-out-exactly	57%	59%	40%	52%
euclidean	3	0	leave-me-out-exactly	58%	60%	36%	51%
euclidean	5	0	leave-me-out	59%	49%	42%	50%
euclidean	7	0	leave-me-out	55%	53%	42%	50%
euclidean	3	0	leave-me-out	59%	50%	41%	50%
euclidean	7	0	leave-me-out-exactly	54%	47%	37%	46%

Table A.2: All results for error type classification. A=correct accuracy, B=incorrect speed accuracy, C=incorrect trajectory accuracy, D=incorrect amplitude accuracy.

distance function	k	sensors	validation	A	B	C	D	avg. acc.
euclidean	1	0, 1	leave-one-out	90%	90%	86%	88%	88%
quaternion	1	0, 1	leave-one-out	86%	91%	85%	90%	88%
euclidean	3	0, 1	leave-one-out	85%	83%	86%	89%	86%
euclidean	5	0, 1	leave-one-out	84%	84%	82%	90%	85%
quaternion	3	0, 1	leave-one-out	84%	84%	79%	90%	84%
euclidean	7	0, 1	leave-one-out	81%	84%	80%	89%	84%
quaternion	5	0, 1	leave-one-out	82%	83%	75%	90%	82%
quaternion	7	0, 1	leave-one-out	81%	77%	72%	92%	80%
euclidean	1	1	leave-one-out	82%	80%	70%	82%	79%
quaternion	1	1	leave-one-out	81%	84%	65%	81%	78%
quaternion	3	1	leave-one-out	79%	80%	61%	83%	76%
euclidean	3	1	leave-one-out	78%	66%	72%	86%	75%
euclidean	5	1	leave-one-out	77%	64%	70%	85%	74%
quaternion	5	1	leave-one-out	74%	77%	56%	85%	73%
euclidean	1	0	leave-one-out	78%	74%	69%	64%	71%
euclidean	3	0	leave-one-out	64%	74%	69%	69%	69%
quaternion	1	0	leave-one-out	71%	70%	69%	65%	69%
euclidean	7	1	leave-one-out	66%	61%	66%	82%	69%
euclidean	5	0	leave-one-out	62%	73%	63%	74%	68%

distance function	k	sensors	validation	A	B	C	D	avg. acc.
quaternion	3	0	leave-one-out	67%	69%	63%	67%	66%
quaternion	7	1	leave-one-out	67%	70%	46%	81%	66%
quaternion	5	0	leave-one-out	60%	69%	54%	64%	62%
euclidean	7	0	leave-one-out	56%	69%	49%	69%	61%
quaternion	7	0	leave-one-out	53%	61%	48%	65%	57%
euclidean	7	0, 1	leave-me-out-exactly	38%	37%	24%	54%	38%
euclidean	7	0	leave-me-out-exactly	41%	24%	31%	57%	38%
euclidean	1	0, 1	leave-me-out-exactly	27%	41%	23%	54%	36%
euclidean	5	0, 1	leave-me-out-exactly	36%	31%	25%	53%	36%
quaternion	1	0, 1	leave-me-out-exactly	34%	30%	18%	63%	36%
euclidean	7	1	leave-me-out-exactly	37%	39%	21%	47%	36%
euclidean	1	1	leave-me-out-exactly	26%	41%	25%	50%	36%
euclidean	5	1	leave-me-out-exactly	33%	36%	25%	47%	35%
euclidean	3	0, 1	leave-me-out-exactly	32%	34%	23%	53%	35%
euclidean	5	0	leave-me-out-exactly	36%	21%	32%	51%	35%
quaternion	3	0, 1	leave-me-out-exactly	36%	26%	18%	61%	35%
quaternion	1	1	leave-me-out-exactly	27%	36%	8%	65%	34%
quaternion	5	0, 1	leave-me-out-exactly	34%	23%	14%	60%	33%
quaternion	7	0, 1	leave-me-out-exactly	36%	21%	11%	61%	32%
euclidean	3	1	leave-me-out-exactly	25%	33%	23%	49%	32%
quaternion	5	1	leave-me-out-exactly	34%	29%	7%	57%	32%
quaternion	7	1	leave-me-out-exactly	29%	29%	11%	57%	31%
quaternion	3	1	leave-me-out-exactly	26%	30%	8%	60%	31%
euclidean	3	0	leave-me-out-exactly	30%	17%	30%	44%	30%
quaternion	3	0	leave-me-out-exactly	42%	6%	24%	42%	28%
quaternion	1	0	leave-me-out-exactly	38%	10%	23%	40%	28%
quaternion	5	0	leave-me-out-exactly	40%	9%	21%	42%	28%
quaternion	7	0	leave-me-out-exactly	42%	9%	15%	44%	28%
euclidean	7	0	leave-me-out	25%	31%	10%	43%	27%
euclidean	7	0, 1	leave-me-out	21%	29%	13%	46%	27%
quaternion	1	0, 1	leave-me-out	15%	26%	8%	58%	27%
euclidean	5	0, 1	leave-me-out	21%	27%	14%	44%	27%
euclidean	1	0	leave-me-out-exactly	22%	21%	25%	36%	26%
quaternion	5	0, 1	leave-me-out	21%	26%	4%	54%	26%
quaternion	3	0, 1	leave-me-out	21%	24%	4%	56%	26%
euclidean	5	0	leave-me-out	18%	33%	11%	42%	26%
euclidean	7	1	leave-me-out	15%	26%	21%	42%	26%
euclidean	5	1	leave-me-out	16%	23%	23%	42%	26%
quaternion	1	1	leave-me-out	18%	26%	1%	57%	25%
quaternion	7	0, 1	leave-me-out	19%	24%	4%	54%	25%
euclidean	3	0	leave-me-out	15%	30%	17%	39%	25%
euclidean	3	1	leave-me-out	15%	21%	21%	43%	25%
euclidean	3	0, 1	leave-me-out	18%	24%	14%	43%	25%
euclidean	1	1	leave-me-out	15%	19%	23%	40%	24%
quaternion	7	0	leave-me-out	25%	21%	11%	39%	24%
quaternion	5	0	leave-me-out	26%	21%	13%	35%	24%
euclidean	1	0, 1	leave-me-out	16%	24%	11%	42%	23%
quaternion	1	0	leave-me-out	22%	17%	21%	32%	23%
quaternion	3	1	leave-me-out	16%	23%	1%	51%	23%
quaternion	3	0	leave-me-out	27%	17%	15%	32%	23%
euclidean	1	0	leave-me-out	11%	26%	24%	31%	23%
quaternion	7	1	leave-me-out	16%	23%	1%	50%	23%
quaternion	5	1	leave-me-out	19%	24%	1%	44%	22%

Bibliography

[1] Cc Aggarwal, A. Hinneburg and Da Keim. 'On the surprising behavior of distance metrics in high dimensional space'. In: *Database Theory - Icdt 2001, Proceedings* 1973 (2001), pp. 420–434. ISSN: 0302-9743.

[2] Antti Ajanki AnAj. *Example of k-nearest neighbour classificationnb.* 28th May 2007. URL: https://commons.wikimedia.org/wiki/File:KnnClassification.svg (visited on 01/05/2017).

[3] Sandra Frances Bassett. 'The assessment of patient adherence to physiotherapy rehabilitation'. In: *New Zealand Journal of Physiotherapy* 31.2 (July 2003), pp. 60–66.

[4] Jeroen H. M. Bergmann et al. 'An Attachable Clothing Sensor System for Measuring Knee Joint Angles'. In: *IEEE Sensors Journal* 13.10 (Oct. 2013), pp. 4090–4097. ISSN: 1530-437X, 1558-1748. DOI: 10.1109/JSEN.2013.2277697. URL: http://ieeexplore.ieee.org/document/6578074/ (visited on 20/03/2017).

[5] *Bluetooth Low Energy Bluetooth Development Portal.* URL: https://developer.bluetooth.org/TechnologyOverview/pages/ble.aspx (visited on 21/08/2016).

[6] *Buy Myo | Myo Gesture Control Armband for $199 USD.* Thalmic Labs. URL: https://store.myo.com/ (visited on 01/05/2017).

[7] Chabacano. *diagram showing overfitting of a classifier.* Feb. 2008. URL: https://commons.wikimedia.org/wiki/File:Overfitting.svg (visited on 01/05/2017).

[8] *fastdtw 0.2.2 : Python Package Index.* URL: https://pypi.python.org/pypi/fastdtw/0.2.2 (visited on 27/04/2017).

[9] M Friedrich, T Cermak and P Maderbacher. 'The effect of brochure use versus therapist teaching on patients performing therapeutic exercise and on changes in impairment status'. In: *Physical therapy* 76.10 (1996), p. 1082. ISSN: 0031-9023.

[10] *Generic Attribute Profile (GATT) Specification | Bluetooth Technology Website.* URL: https://www.bluetooth.com/specifications/generic-attributes-overview (visited on 01/05/2017).

[11] Oonagh M Giggins, Kevin T Sweeney and Brian Caulfield. 'Rehabilitation exercise assessment using inertial sensors: a cross-sectional analytical study'. In: *Journal of NeuroEngineering and Rehabilitation* 11.1 (2014), p. 158. ISSN: 1743-0003. DOI: 10.1186/1743-0003-11-158. URL: http://jneuroengrehab.biomedcentral.com/articles/10.1186/1743-0003-11-158 (visited on 20/03/2017).

[12] Thomas Harbo, John Brincks and Henning Andersen. 'Maximal isokinetic and isometric muscle strength of major muscle groups related to age, body mass, height, and sex in 178 healthy subjects'. In: *European Journal of Applied Physiology* 112.1 (Jan. 2012), pp. 267–275. ISSN: 1439-6319, 1439-6327. DOI: 10.1007/s00421-011-1975-3. URL: http://link.springer.com/10.1007/s00421-011-1975-3 (visited on 20/03/2017).

[13] *Home - Xsens 3D motion tracking*. URL: https://www.xsens.com/ (visited on 02/05/2017).

[14] Fay Horak, Laurie King and Martina Mancini. 'Role of body-worn movement monitor technology for balance and gait rehabilitation'. In: *Physical therapy* 95.3 (2015), pp. 461–70. ISSN: 1538-6724. DOI: 10.2522/ptj.20140253.

[15] CIR Systems Inc. *GAITRite Electronic Walkway Technical Reference*. Mar. 2016.

[16] *LPMS-B Bluetooth IMU Maker Edition - LpResearch - InMojo*. URL: http://www.inmojo.com/store/lpresearch/item/lpms-b-bluetooth-imu-maker-edition/ (visited on 01/05/2017).

[17] *LPMS-B-HM STD (LPMS-B STD with Heave Motion)*. URL: http://www.hibot.co.jp/en/products/motion-sensor_11/lpms-b-hm-std-lpms-b-std-with-heave-motion_48 (visited on 01/05/2017).

[18] *Myo Bluetooth header spec*. GitHub. URL: https://github.com/thalmiclabs/myo-bluetooth (visited on 01/05/2017).

[19] *Official Thalmic Labs Press Kit.zip*. URL: https://thlodestone.s3.amazonaws.com/press/Official%20Thalmic%20Labs%20Press%20Kit.zip (visited on 21/08/2016).

[20] Ferda Özdemir et al. 'Comparing stroke rehabilitation outcomes between acute inpatient and nonintense home settings'. In: *Archives of Physical Medicine and Rehabilitation* 82.10 (Oct. 2001), pp. 1375–1379. ISSN: 00039993. DOI: 10.1053/apmr.2001.25973. URL: http://linkinghub.elsevier.com/retrieve/pii/S0003999301286675 (visited on 20/03/2017).

[21] *Quaternions - Visualisation*. URL: http://quaternions.online/ (visited on 01/05/2017).

[22] S. Salvadora and P. Chan. 'Toward accurate dynamic time warping in linear time and space'. In: *Intelligent Data Analysis* 11.5 (2007), pp. 561–580. ISSN: 1088-467X.

[23] Brenda C. Spillman and James Lubitz. 'The Effect of Longevity on Spending for Acute and Long-Term Care'. In: *The New England Journal of Medicine* 342.19 (2000), pp. 1409–1415. ISSN: 0028-4793. DOI: 10.1056/NEJM200005113421906.

[24] Christina Strohrmann et al. 'Monitoring motor capacity changes of children during rehabilitation using body-worn sensors'. In: *Journal of NeuroEngineering and Rehabilitation* 10.1 (2013), p. 83. ISSN: 1743-0003. DOI: 10.1186/1743-0003-10-83. URL: http://jneuroengrehab.biomedcentral.com/articles/10.1186/1743-0003-10-83 (visited on 20/03/2017).

[25] Luigi Tesio et al. 'The 3D path of body centre of mass during adult human walking on force treadmill'. In: *Journal of Biomechanics* 43.5 (Mar. 2010), pp. 938–944. ISSN: 00219290. DOI: 10.1016/j.jbiomech.2009.10.049. URL: http://linkinghub.elsevier.com/retrieve/pii/S0021929009006423 (visited on 20/03/2017).

[26] Ekaterina B. Titianova, Plamen S. Mateev and Ina M. Tarkka. 'Footprint analysis of gait using a pressure sensor system'. In: *Journal of Electromyography and Kinesiology* 14.2 (Apr. 2004), pp. 275–281. ISSN: 10506411. DOI: 10.1016/S1050-6411(03)00077-4. URL: http://linkinghub.elsevier.com/retrieve/pii/S1050641103000774 (visited on 20/03/2017).

[27] User:Psychonaut. *Figure illustrating Manhattan versus Euclidean distance. The red, blue, and yellow lines all have the same length (12), whereas the green line has length.* 25th Apr. 2006. URL: https://commons.wikimedia.org/wiki/File:Manhattan_distance.svg (visited on 02/05/2017).

[28] *Welcome to the Home of LP-RESEARCH.* LP-RESEARCH. URL: https://www.lp-research.com/ (visited on 01/05/2017).

[29] Andrea Winkelmann - Gleed. 'Demographic change and implications for workforce ageing in Europe: raising awareness and improving practice.(Report)'. In: *Contemporary Readings in Law and Social Justice* 3.1 (2011), p. 62. ISSN: 1948-9137.

[30] D. H. Wolpert and W. G. Macready. 'No free lunch theorems for optimization'. In: *Evolutionary Computation, IEEE Transactions on* 1.1 (1997), pp. 67–82. ISSN: 1089-778X. DOI: 10.1109/4235.585893.

[31] Aras Yurtman and Billur Barshan. 'Automated evaluation of physical therapy exercises using multi-template dynamic time warping on wearable sensor signals'. In: *Computer Methods and Programs in Biomedicine* 117.2 (Nov. 2014), pp. 189–207. ISSN: 01692607. DOI: 10.1016/j.cmpb.2014.07.003. URL: http://linkinghub.elsevier.com/retrieve/pii/S0169260714002910 (visited on 20/03/2017).