

DISSERTATION



CLAAS AHLRICHS

Development and Evaluation of AI-based Parkinson's Disease Related Motor Symptom Detection Algorithms

A Thesis submitted to
Faculty 3 of University Bremen
in (Partial) Fulfillment of the Requirements of the
Degree of
Doktor-Ingenieur
(Dr.-Ing.)

Claas Ahlrichs
Department of Computer Science
University Bremen

First Reviewer: Prof. Dr. Michael Lawo
Second Reviewer: Dr. Albert Samà

July 6, 2015

Preface

This is the work of a computer scientist, not a medical professional. This statement is not meant to have an impact on quality and dedication with which this research has been conducted. However, it is intended to affect expectations regarding its contents and employed methods. In other words, this statement serves as a reminder that you, as a reader, may come from a very different background and your expectations may be very different depending on your particular field of expertise. Here, machine learning (ML) and artificial intelligence (AI) methods are more evident than biological or neurological methodologies. Having said this, I sincerely hope that you enjoy the results of this work and fruits of my labor.

Abstract

Parkinson's Disease (PD) is a chronic, progressive, neurodegenerative disorder [12, 45, 69, 73, 146] that is typically characterized by a loss of (motor) function, increased slowness and rigidity. It is generally attributed to elderly people and is rarely diagnosed before the age of 40. Despite radical advancements in medicine since its discovery, reason and cause of PD remain unknown [44, 69, 83, 146]. Thus, most treatments aim at reducing severity and frequency of motor complications. Treatments are yet to be found and shown to cure or even slow the progression of PD. Due to the lack of feasible biomarkers, progression cannot easily be quantified with objective measures. For the same reason, neurologists have to revert to monitoring of (motor) symptoms (i.e. by means of subjective and often inaccurate patient diaries) in order to evaluate a medication's effectiveness. Replacing or supplementing these diaries with an automatic and objective assessment of symptoms and side effects could drastically reduce manual efforts (by patients as well as neurologists) and potentially help in personalizing and improving medication regime. In turn, appearance of symptoms could be reduced and the patient's quality of life increased.

This work presents a review of publications related to detecting several PD motor symptoms and side effects. Building on this knowledge a set of methodologies for recognizing several motor deficiencies is proposed and eventually evaluated. The proposed approach roughly consists in post processing the classification results of a support vector machine (SVM) (a supervised learning algorithm) that has been trained to recognize a symptom (or side effect). Accuracy and other measures are determined by applying the approach to a large database (DB) with recordings from 92 patients. The resulting accuracies were found to outperform state-of-the-art techniques in case of two symptoms (i.e. resting tremor and freezing). However, data usage was slightly penalized. As for dyskinesia (i.e. a side effect of some PD medications), the methodology yielded to acceptable accuracies (above 90%) but it does heavily penalize data usage and it does not quite reach the performance of state-of-the-art techniques.

The second part of this work focuses on a framework for data mining (DM) applications and time series. The literature may have proposed a set of time series frameworks for this purpose [21, 22, 56, 175], however, many authors still prefer handcrafted solutions rather than utilizing a common framework. Reasons for this preference include: lack of maintenance, inapplicability to the domain or lack of flexibility and functionality. All of these complicate development and evaluation of new algorithms. Consequently, new algorithms can easily be published but fail to provide fundamental comparisons against an adequate subset of competitive state-of-the-art techniques. This thesis highlights the development process of a software framework that aims to nullify some of

these drawbacks. The framework is constructed around the concepts of modularity, reusability and configurability.

Kurzdarstellung

Bei der Parkinson-Krankheit handelt es sich um eine chronische, progressive und neurodegenerative Erkrankung [12, 45, 69, 73, 146], welche sich typischerweise durch den Verlust von motorischen Fähigkeiten, zunehmender Langsamkeit und Starrheit auszeichnet. Im Allgemeinen wird die Krankheit älteren Menschen zugeschrieben und wird selten vor dem vierzigsten Lebensjahr diagnostiziert. Trotz enormer medizinischer Fortschritte seit dem Bekanntwerden der Krankheit bleiben Ursache und Grund weiterhin unbekannt [44, 69, 83, 146]. Demnach sind die meisten Behandlungen darauf ausgerichtet, Stärke und Häufigkeit von motorischen Komplikationen zu verringern. Leider gibt es keine Chance auf eine Genesung beziehungsweise darauf den Krankheitsverlauf zu verlangsamen. Durch fehlende Biomarker und Indikatoren kann das Fortschreiten der Krankheit nur schwierig quantifiziert werden. Daher sind Neurologen dazu gezwungen auf die Kontrolle und Beobachtung von (motorischen) Symptomen zurückzugreifen (dies geschieht häufig in Form von subjektiven und ungenauen Patiententagebüchern) um die Wirkung der Medikamentierung zu bewerten. Das Ersetzen bzw. Ergänzen dieser Tagebücher mit einer automatischen und objektiven Erhebung von Symptomen und Nebenwirkungen könnte manuelle Bemühungen (sowohl von Patienten als auch von Neurologen) verringern. Des Weiteren könnte dies zu einer Verbesserung der Medikamentierungen führen und damit auch zu einem verminderten Auftreten von Symptomen. Im gleichen Schritt würde die Lebensqualität von Patienten verbessert werden.

Diese Arbeit stellt eine Reihe von Veröffentlichungen mit Bezug zu verschiedenen (motorischen) Parkinson-Symptomen und Seiteneffekten vor. Auf diesem Wissen aufbauend, werden Vorgehensweisen zur Erkennung solcher motorischer Schwächen vorgestellt und schließlich evaluiert. Die Herangehensweise basiert im Kern auf der Nachverarbeitung von Klassifikationsergebnissen einer Support Vector Machine (SVM) (einem Verfahren aus dem Bereich des überwachten Lernens). Die SVM wird darauf trainiert (oder angelernt) motorische Symptome und Seiteneffekte wieder zuerkennen. Im weiteren Verlauf werden die Klassifikationen der SVM zusammengefasst und im Rahmen einer so genannten Meta-Analyse verwertet. Die Auswertung der Methode geschieht anhand einer großen Datenbank mit Aufzeichnungen von 92 Patienten. Bei der Erkennung von zwei Symptomen (d.h. Tremor und akinetischen Phasen) übertrafen die daraus resultierenden Ergebnisse aktuelle Veröffentlichungen. Allerdings führte das Verfahren zu einer (wenn auch akzeptablen) Verringerung der Datennutzung. In Sachen Dyskinesie (d.h. eine Nebenwirkung von manchen Medikamentierungen) führte das Verfahren zwar zu akzeptablen Ergebnissen (über 90% Genauigkeit) aber die Datennutzung wurde stark verkleinert und die Leistungen aktueller Veröffentlichungen konnten nicht ganz erreicht werden.

Der zweite Teil dieser Arbeit konzentriert sich auf ein Software-Rahmenwerk für Anwendungen im Bereich von Zeitreihen und Data Mining. Zwar wurden in der Literatur einige solcher Rahmenwerke bereits vorgestellt [21, 22, 56, 175], allerdings ziehen viele Autoren eigenständige Lösungsansätze dem Einsatz dieser Rahmenwerke vor. Gründe für diese Abneigung sind unter anderem fehlende Wartung (bzw. Instandhaltung), Unbrauchbarkeit im Bezug zum Anwendungsfeld (Rahmenwerk ist auf ein anderes Feld ausgelegt) oder einfach fehlende Flexibilität und Funktionalität. All dies erschwert die Entwicklung und Bewertung von neuen Algorithmen. Aus diesem Grund kann es leicht vorkommen, dass neue Verfahren veröffentlicht werden, es aber nur Vergleiche und Auswertungen mit einer ungenügenden Auswahl von konkurrierenden Verfahren gibt. Diese Arbeit hebt den Entwicklungsprozess eines Software-Rahmenwerkes hervor, welches zum Ziel hat einige dieser Nachteile aufzuheben. Das Rahmenwerk ist mit Hinblick auf Konzepte wie Modularität, Wiederverwendbarkeit und Konfigurierbarkeit angelegt.

Acknowledgment

Having worked on this thesis for the past three years, I came to realize that it is not an easy task to acknowledge all those people that deserve to be named here. For agreeing to be my supervisor, supporting this work and helping me to manoeuvre all the way to the end of it, I thank Prof. Dr. Michael Lawo. For his constant feedback and countless productive discussions, I deeply thank my personal friend and supervisor Dr. Albert Samà. For accepting me as a contributor and doctoral candidate, I thank the REMPARK project and its consortium members. For having endured me over past years, her creative feedback and visually pleasing contributions, I truly thank Nadine Freye. Furthermore, I would like to thank my mother Bärbel Ahlrichs and father Thomas Ahlrichs, brothers Jan and Hauke Ahlrichs and sisters Swantje, Wibke, Lina and Rike Ahlrichs without whom this work may very well never have been written or attempted. I also like to express my gratitude toward Prof. Dr. Stefan Edelkamp, doctoral candidate Galina Stoyanova and Carlos Pérez-López. Last but not least, I thank all participants (that contributed to the REMPARK database) without whom this thesis would not have a sufficient set of data samples.

Contents

Preface	i
Abstract	iii
Acknowledgment	vii
Table of Contents	ix
List of Figures	xiii
List of Tables	xvii
Listings	xxiii
1 Introduction	1
1.1 Motivation	1
1.2 Research Questions	4
1.2.1 Scope and Limitations	5
1.2.2 Requirements	6
1.3 Thesis Organization	7
2 A Brief Review of Parkinson’s Disease and Time Series	9
2.1 Definitions and Notations	9
2.2 Parkinson’s Disease	13
2.2.1 Symptoms	17
2.2.2 Diagnosis	21
2.2.3 Treatment	22
2.2.4 Rating Scales	24
2.3 Temporal Data Mining	25
2.3.1 Selection	26
2.3.2 Preprocessing	28
2.3.3 Transformation	33
2.3.4 (Temporal) Data Mining	37
2.3.5 Interpretation	40
2.4 Summary	41

3	Related Work Regarding Symptom Detection	43
3.1	Frameworks for Time Series Analysis	43
3.1.1	Waikato Environment for Knowledge Analysis	44
3.1.2	Massive Online Analysis	44
3.1.3	Unstructured Information Management Architecture	45
3.1.4	Framework: streams	46
3.1.5	Others	47
3.1.6	Summary	48
3.2	Identifying Parkinson’s Disease and Its Symptoms	51
3.2.1	Tremor at Rest	51
3.2.2	Bradykinesia	53
3.2.3	Akinesia / Freezing of Gait	54
3.2.4	Drug-Induced Dyskinesia	55
3.2.5	Dysarthria and Dysphagia	56
3.2.6	Others	58
3.2.7	Summary	61
3.3	Parkinson’s Disease in Research Projects	65
3.3.1	HELP	65
3.3.2	REMPARK	66
3.3.3	PERFORM	67
3.3.4	TREMOR	68
3.3.5	Summary	68
4	A Framework for Time Series Analysis	71
4.1	Design and Development	71
4.1.1	Design Challenges	72
4.1.2	Structure and Design of Modules	73
4.1.3	Flow In-between Modules	87
4.1.4	Data Sources and Sinks	98
4.1.5	Summary (of Used Design Patterns)	103
4.2	Extensibility	104
4.2.1	Adding Modules	105
4.2.2	Wrapping and Decorating Modules	106
4.2.3	Data Sources and Sinks	106
4.2.4	Links	107
4.2.5	Functions Across the Entire Graph	108
4.2.6	Alternative Traversal Methods	108
4.3	Applications and Scenarios	109
4.3.1	Recognizing PD Motor Symptoms	109
4.3.2	Generating Trading Decisions	109
4.3.3	Analysis of Network Traffic	110
4.3.4	Quality Control of OpenStreetMap-Data	110
4.4	Included Modules and Algorithms	110
4.5	Summary	110
5	Database: Patients and Their Symptoms	113
5.1	Data Acquisition and Labeling	113
5.1.1	Sensors	113
5.1.2	Criteria and Demography	114
5.1.3	Protocols	114

5.1.4	Tests	118
5.1.5	Labeling	119
5.2	Contents of Database	119
5.2.1	Tremor at Rest	127
5.2.2	Dyskinesia	127
5.2.3	Bradykinesia	128
5.2.4	Freezing of Gait	128
5.2.5	Summary	129
6	Indication of Parkinson's Disease Motor Symptoms	131
6.1	Tremor (at Rest)	131
6.1.1	Variation 1: Naive Approach	132
6.1.2	Variation 2: Aggregation	138
6.1.3	Variation 3: Two-Sided Aggregation	142
6.1.4	Further Variations	144
6.2	Dyskinesia	144
6.2.1	Variation 1 to 3:	148
6.2.2	Further Variations	152
6.3	Akinesia / Freezing of Gait	152
6.3.1	Variation 1 to 3:	153
6.4	Summary	159
7	Benchmark of Symptom Detecting Algorithms	161
7.1	Tremor (at Rest)	161
7.2	Dyskinesia	163
7.3	Freezing of Gait	165
7.4	Discussion	166
8	Conclusions	169
8.1	Conclusions	169
8.2	Contributions	170
8.3	Future Work	171
	Appendix	195
A	Parkinson's Disease	197
A.1	Symptoms	197
A.2	Diagnosis and Treatment	204
B	Implementation Resources	211
B.1	Algorithm Samples	211
B.2	Framework Samples	212
B.3	Modules of Framework	223
B.4	Road-Map	227
C	Additional Results in Signifying Motor Symptoms	231
C.1	Tremor (at Rest)	231
C.1.1	Any Tremor at Waist	231
C.1.2	Lower Tremor at Waist	231
C.1.3	Upper Tremor at Waist	231
C.2	Dyskinesia	231

C.2.1	Any Dyskinesia at Waist	231
C.2.2	Trunk + Strong Limb Dyskinesia at Waist	238
D	Questionnaires	241

List of Figures

1.1	Illustrates structures of the brain related to the basal ganglia circuit. Latter is located in the upper end of the brain stem. The image is based on a figure which has been retrieved from Wikimedia Commons and belongs to the public domain.	2
2.1	Shows the therapeutic window of PD and how its boundaries narrow over time. This Figure is intended to indicate general tendencies with regard to disease progression (i.e. decrease in effectiveness and narrowing of therapeutic window). These graphs are not necessarily scaled properly, thus units on both axis have been avoided. Areas identified by number 1 correspond to periods in ON, number 2 represents periods in OFF and number 3 highlights periods with dyskinesias.	16
2.2	Example of a person suffering from hypomimia or otherwise called loss of facial expression. The masked face can be a result of bradykinesia or related symptoms. The image has been retrieved from Wikimedia Commons and belongs to the public domain. . .	19
2.3	Example of a person showing signs of micrographia. PD symptoms like bradykinesia may lead to a small, cramped handwriting. The image has been retrieved from Wikimedia Commons and belongs to the public domain.	19
2.4	The general flow of information and the five steps of the knowledge discovery in databases (KDD) process are outlined. Note that the actual process is not as linear as suggested by this representation. Results may improve if steps are performed multiple times. The image is based on a figure by Fayyad et al. [54]. . . .	27
2.5	Shows the body temperature of a child over a period of three days. Note the two clear outliers below the “average” temperature line (dashed). For more details on the values refer to Table 2.2. . . .	30
2.6	Illustrates <i>mean</i> and <i>median</i> binning if applied to body temperature data samples shown in Table 2.2. The size of a bin is four, so that a (single) bin represents measurements from an entire day.	31
2.7	Illustrates the results when a 3-point central moving average (CMA) and a simple 4-point weighted moving average (MA) are applied to the body temperature data samples listed in Table 2.2.	31
2.8	Shows resampled body temperature data listed in Table 2.2. For an easier reference, the original data samples are also included. .	32

2.9	Illustrates <i>min-max</i> and <i>z-score</i> normalization applied to the body temperature example from Table 2.2. The required (minimum, maximum, etc.) values have been determined by searching the entire dataset. For convenience, these values are: minimum = 35.8°C, maximum = 37.8 °C, mean = 37.1 °C, standard deviation = 0.6 °C ²	35
4.1	Illustrates the structure of the Strategy design pattern as defined by [57, p. 349]. The relationship among <i>ConcreteStrategy</i> classes and abstract <i>Strategy</i> is highlighted.	74
4.2	Shows the implementation of the Strategy design pattern. Here the <i>Processor</i> interface and a few exemplary implementations are highlighted.	75
4.3	Shows the life circle of a module within the framework.	76
4.4	Shows the implementation of the Strategy design pattern in the context of configurability. The classes <i>ConfigurableAdapter</i> and <i>Configurable</i> are shown.	77
4.5	Shows the implementation of the Strategy design pattern for constraining configuration parameters. The <i>Condition</i> interface and several concrete implementations are highlighted.	78
4.6	Illustrates the structure of the Observer design pattern as defined by [57, p. 293]. The relationship among <i>ConcreteObserver</i> class and abstract <i>Observer</i> is highlighted as well as their relationship to <i>ConcreteSubject</i> and abstract <i>Subject</i>	79
4.7	Shows the implementation of the Observer design pattern. Here the <i>Observer</i> interface and an exemplary implementation are highlighted.	80
4.8	Illustrates the structure of the Adapter design pattern as defined by [57, p. 157].	81
4.9	Shows the implementation of the Adapter design pattern as part of the framework. Here, the interaction between various configuration related parts of the framework (i.e. <i>Configuration</i> , <i>Condition</i> and <i>Observer</i>) is highlighted.	82
4.10	Illustrates the structure of the Template Method design pattern as defined by [57, p. 360]. The <i>AbstractClass</i> defines a set of template methods as well as a set of function-specific operations. <i>ConcreteClass</i> classes are meant to change the behavior of these operations. The template methods are typically based on these operations as well.	83
4.11	Shows the implementation of the Template Method design pattern. The <i>ConfigurableAdapter</i> class contains both function-specific methods (i.e. <i>getParameter</i> , <i>setParameter</i> and <i>getParameters</i>) and template methods (e.g. <i>getBoolean</i> , <i>setBoolean</i> , etc.).	84
4.12	Illustrates the structure of the Decorator design pattern as defined by [57, p. 196]. The generic <i>Component</i> interface is shown as well as the <i>Decorator</i> classes. Their relationship is highlighted.	85
4.13	Shows the implementation of the Decorator design pattern in the framework. Several variations of the <i>Decorator</i> class are shown.	86

4.14	Shows possible structures that can be created with tree-like and graph-like representations. Items 4 and 5 demonstrate the main difference between both approaches. In order of appearance, the modeling situations refer to: “single node”, “single child, single parent”, “multiple children, single parent”, “single child, multiple parents” and “cycle”. Each box represents a single module whereas each sub-figure depicts an “atomic” modeling situation. One can replace a box with any other modeling situation and still have a functional structure.	88
4.15	Illustrates the concept of layering graphs. Module “A” represents the entry point of the graph (i.e. data source). From there information is passed to module “B” and “C”. Continuing this path the information from “B” is going to reach “E” one iteration before the data from “D” would (left). This is due to the extra module and can be avoided (if wished) by grouping modules “C” and “D” into a single one (middle). Alternatively, an extra “dummy” module could be inserted between “B” and “E” (right).	89
4.16	Illustrates the structure of the Composite design pattern as defined by [57, p. 163]. The generic <i>Component</i> interface is shown as well as <i>Leaf</i> and <i>Composite</i> elements (including their implementations).	90
4.17	Shows the implementation of the Composite design pattern. It supports treating compositions (e.g. class <i>CompositeNode</i>) as well as individual components (e.g. class <i>PlainNode</i>) in a homogeneous way.	90
4.18	Defines the structure of the Iterator design pattern [57, p. 257]. The <i>Aggregate</i> and <i>Iterator</i> interfaces are shown as well as specializations of both.	93
4.19	Illustrates the implementation of the Iterator design pattern. Furthermore, the adapted <i>Iterator</i> and <i>Node</i> (former <i>Aggregate</i>) interfaces are shown. Concrete implementations are presented as well.	93
4.20	Shows realizations of the Iterator design pattern: <i>InfiniteLevelOrder</i> (bottom) and <i>OneShotLevelOrder</i> (top).	94
4.21	Shows the implementation of the Strategy design pattern for linking modules together. It illustrates the relationship among <i>Link</i> , <i>Configurable</i> and concrete implementations.	96
4.22	Illustrates the structure of the Visitor design pattern as defined by [57, p. 366]. Hierarchies for <i>Visitor</i> and <i>Element</i> classes as well as their interaction are shown.	97
4.23	Shows the implementation of the Visitor design pattern as part of the framework.	98
4.24	Shows the implementation of the Strategy design pattern for reading and writing arbitrary data.	99
4.25	Illustrates the structure of the Bridge design pattern as defined by [57, p. 171].	101
4.26	Shows the implementation of the Bridge design pattern.	102

5.1	Highlights the sensor locations in relationship to a (human) skeleton. The wrist sensor is placed on the left hand. The waist sensor is located near the anterior superior illiac spine (ASIS). The image has been retrieved from Wikimedia Commons and belongs to the public domain.	115
5.2	Illustrates the order and length of the recording sessions.	117
6.1	Shows the frequency spectrum of an individual with tremor and another individual without tremor. It can be noted that during episodes of tremor frequencies between 4 - 6 Hz are apparent (as well as their harmonics).	133
6.2	Shows the behavior of geometric mean and accuracy with unbalanced datasets (number of positive and negative labels diverges).	136
6.3	Illustrates the results of an evaluation for varying window sizes with respect to Parkinsonian tremor. For each window size, an SVM has been trained on the training dataset and evaluated on the test dataset.	137
6.4	Indicates the effect of window aggregation t and threshold th on geometric mean. The results are shown for all four conditions.	141
6.5	Shows the overall methodology for detecting tremor.	145
6.6	Illustrates the results of an evaluation for varying window sizes with respect to dyskinesia. For each window size, an SVM has been trained on the training dataset and evaluated on the test dataset.	147
6.7	Indicates the effect of window aggregation t and threshold th on geometric mean. The results are shown for all four conditions.	151
6.8	Illustrates the results of an evaluation for varying window sizes with respect to freezing episodes. For each window size, an SVM has been trained on the training dataset and evaluated on the test dataset.	156
6.9	Indicates the effect of window aggregation t and threshold th on geometric mean. The results are shown for all four conditions.	157
B.1	Illustrates a possible way of implementing the algorithms presented in Chapter 6.	228

List of Tables

2.1	Highlights the relationship among various commonly used terminology from a ML point of view. Here terms like true positive and false positive are listed.	11
2.2	Lists the body temperature of a child over a period of three days. The average temperature is 37.1°C. The numbers are represented as time series A with a length of $m = 12$ and $n = 3$ attributes. . .	30
2.3	Shows the calculated results based on the data samples shown in Table 2.2 for both binning methods (i.e. <i>mean</i> and <i>median</i>). . . .	30
2.4	Lists DM algorithms of various categories. This is not intended to be a complete list, but rather an overview of commonly used techniques.	39
3.1	Highlights the applicability of mentioned frameworks to the comparison criteria. Those points that are enclosed in brackets are partially fulfilled (✓ and ✗ indicate whether the criteria closer is to being fulfilled or not fulfilled).	50
3.2	Summarization of state-of-the-art publications on PD symptom indication algorithms. For each symptom (T: tremor, B: bradykinesia, F: freezing of gait (FoG), D: dyskinesia) and reference, the employed classification techniques and utilized sensors (A: accelerometer, G: gyroscope, E: electromyograph (EMG) sensor) are highlighted. Furthermore, their results are indicated. It should be kept in mind that the results among these papers are not directly comparable due to employment of different datasets.	64
3.3	Lists several related publications. The author(s) and title of their respective publications are highlighted. This list is intended to supplement state-of-the-art publications (shown in Table 3.2) with additional noteworthy and relevant papers.	66
5.1	Inclusion and exclusion criteria for participation in data acquisition. The contents have been summarized from parts of the REMARK project documentation (i.e. case report form for screening and baseline). The original document can be found in the appendix.	116
5.2	Lists the amount of time (in minutes) that each patient experienced a particular symptom. Here resting tremor, dyskinesia, bradykinesia and FoG are listed. The values were extracted from those parts of the DB where a gold standard was available. . . .	122

5.3	Lists the length of recordings (in minutes) with respect to resting tremor. The amount of time is given for combinations of sensor location (i.e. wrist or waist), motor state (i.e. ON, OFF and intermediate) as well as varying types of tremor (which can occur simultaneously). These values were extracted from those parts of the REMPARK DB for which a gold standard was available. . . .	123
5.4	Lists the length of recordings (in minutes) with respect to dyskinesia. The amount of time is given for combinations of sensor location (i.e. wrist or waist), motor state (i.e. ON, OFF and intermediate) as well as varying types of dyskinesia (which can occur simultaneously). These values were extracted from those parts of the REMPARK DB for which a gold standard was available.	124
5.5	Lists the length of recordings (in minutes) with respect to bradykinesia. The amount of time is given for combinations of sensor location (i.e. wrist or waist), motor state (i.e. ON, OFF and intermediate) as well as varying types of bradykinesia. These values were extracted from those parts of the REMPARK DB for which a gold standard was available.	125
5.6	Lists the length of recordings (in minutes) with respect to FoG. The amount of time is given for combinations of sensor location (i.e. wrist or waist), motor state (i.e. ON, OFF and intermediate) as well as varying types of FoG. These values were extracted from those parts of the REMPARK DB for which a gold standard was available.	126
6.1	Lists the number of windows (before aggregation) in each dataset that are used for signifying tremor (i.e. left hand tremor).	134
6.2	Shows the full set of features used for tremor detection. The reduced feature set is comprised of index 1.	134
6.3	Outlines results in recognizing tremor with the naive approach (i.e. variation one). Various measures are listed for both datasets.	138
6.4	Summarizes results in recognizing tremor with the one-sided approach (i.e. variation 2).	141
6.5	Summarizes results in detecting tremor with the two-sided approach (i.e. variation 3).	143
6.6	Lists the number of windows (before aggregation) in each dataset that are used for signifying dyskinesia (i.e. trunk dyskinesia). . .	146
6.7	Shows the full set of features used for dyskinesia detection. The reduced feature set is solely based on index 1.	146
6.8	Outlines results in recognizing dyskinesia with the naive approach (i.e. variation 1). Various measures are listed for both datasets. .	150
6.9	Summarizes results in recognizing dyskinesia with the one-sided approach (i.e. variation 2).	151
6.10	Summarizes results in detecting dyskinesia with the two-sided approach (i.e. variation 3).	152
6.11	Lists the number of windows (before aggregation) in each dataset that are used for detecting FoG.	156
6.12	Shows the full set of features used for FoG detection. The reduced feature set is comprised of index 1.	156

6.13	Outlines results in recognizing FoG with the naive approach (i.e. variation 1). Various measures are listed for both datasets. . . .	157
6.14	Summarizes results in recognizing FoG with the one-sided approach (i.e. variation 2).	158
6.15	Summarizes results in detecting FoG with the two-sided approach (i.e. variation 3).	158
7.1	Summarizes results from various algorithms for detecting Parkinsonian tremor. Results found in Chapter 3 and Chapter 6 are briefly summarized. “D.U.” is used as a shorthand for “data usage”. More details can be found in Table A.4.	162
7.2	Summarizes results from various algorithms for detecting dyskinesia. Results found in Chapter 3 and Chapter 6 are briefly summarized. “D.U.” is used as a shorthand for “data usage”. More details can be found in Table A.6.	164
7.3	Summarizes results from various algorithms for detecting FoG. Results found in Chapter 3 and Chapter 6 are briefly summarized. “D.U.” is used as a shorthand for “data usage”. More details can be found in Table A.5.	165
A.1	Lists a collection of motor symptoms that are associated with PD. This table has been constructed based on data found in a publication by Jankovic [73].	198
A.2	Shows a set of non motor symptoms known to be associated with PD. The contents of this table have been gathered from publications by Jankovic [73] and Hou et al. [69].	199
A.3	Lists a series of publications related to automatic indication algorithms for bradykinesia. For each paper a set of algorithm-related points are highlighted. Various insights on the used datasets are also given (i.e. length, number of participants, kind of activities / tasks). It should be noted that some values (i.e. length of dataset and results) were estimated in order to keep the rows somewhat comparable. Whether or not the publication was referenced in chapter “Related Work Regarding Symptom Detection” is indicated by the “State of the Art” column. Fields marked by “-” indicated that no information was given in the corresponding paper.	200
A.4	Lists a series of publications related to automatic indication algorithms for tremor. For more details on the column descriptions, please refer to Table A.3.	201
A.5	Lists a series of publications related to automatic indication algorithms for FoG events. For more details on the column descriptions, please refer to Table A.3.	202
A.6	Lists a series of publications related to automatic indication algorithms for dyskinesia. For more details on the column descriptions, please refer to Table A.3.	203
A.7	United Kingdom (UK) PD Society Brain Bank’s clinical criteria for the diagnosis of probable Parkinson’s Disease. The contents have been reproduced based on a publication by Jankovic [73]. .	205

A.8	National Institute of Neurological Disorders and Stroke (NINDS) diagnostic criteria for PD. The contents have been reproduced based on a publication by Jankovic [73] and Gelb et al. [58]. . . .	206
A.9	Clinical diagnostic criteria for idiopathic PD. The contents have been reproduced based on a publication by Samii et al. [146]. . .	207
A.10	Proposed inclusion and exclusion criteria for deep brain stimulation (DBS). The contents have been reproduced based on a publication by Samii et al. [146].	208
A.11	Lists UK PD Society Brain Bank's criteria for diagnosis of parkinsonian syndrome. The contents have been reproduced based on a publication by Davie et al. [45].	209
A.12	Proposed diagnostic criteria for histopathologic confirmation of PD. The contents have been reproduced based on a publication by Gelb et al. [58].	210
C.1	Lists the number of windows (before aggregation) that are used for signifying tremor (i.e. any tremor).	231
C.2	Results (any-tremor-waist) for detecting tremor using the naive approach.	232
C.3	Results (any-tremor-waist) for detecting tremor using the one-sided approach.	232
C.4	Results (any-tremor-waist) for detecting tremor using the two-sided approach.	233
C.5	Lists the number of windows (before aggregation) that are used for signifying tremor (i.e. lower tremor).	233
C.6	Results (lower-tremor-waist) for detecting tremor using the naive approach.	233
C.7	Results (lower-tremor-waist) for detecting tremor using the one-sided approach.	234
C.8	Results (lower-tremor-waist) for detecting tremor using the two-sided approach.	234
C.9	Lists the number of windows (before aggregation) that are used for signifying tremor (i.e. upper tremor).	234
C.10	Results (upper-tremor-waist) for detecting tremor using the naive approach.	235
C.11	Results (upper-tremor-waist) for detecting tremor using the one-sided approach.	235
C.12	Results (upper-tremor-waist) for detecting tremor using the two-sided approach.	236
C.13	Lists the number of windows (before aggregation) that are used for signifying dyskinesia (i.e. trunk dyskinesia).	236
C.14	Results (any-dyskinesia-waist) for detecting dyskinesia using the naive approach.	236
C.15	Results (any-dyskinesia-waist) for detecting dyskinesia using the one-sided approach.	237
C.16	Results (any-dyskinesia-waist) for detecting dyskinesia using two-sided approach.	237
C.17	Lists the number of windows (before aggregation) that are used for signifying dyskinesia (i.e. trunk dyskinesia).	238

C.18 Results (trunk-and-strong-limb-dyskinesia-waist) for detecting dysk-	
nesia using the naive approach.	238
C.19 Results (trunk-and-strong-limb-dyskinesia-waist) for detecting dysk-	
nesia using the one-sided approach.	239
C.20 Results (trunk-and-strong-limb-dyskinesia-waist) for detecting dysk-	
nesia using the two-sided approach.	239

Listings

B.1	Demonstrates the implementation of <i>mean</i> -binning using the Awk programming language.	211
B.2	An example of a <i>Processor</i> implementation. The shown example forwards its input values.	212
B.3	An example of a <i>DecoratorProcessor</i> implementation. The shown example generates statistics on the processing time that a module requires.	213
B.4	An example of a <i>StreamHandlerImpl</i> implementation. The shown example provides raw access to a file.	215
B.5	An example of a <i>DataHandler</i> implementation. The shown example provides access to queue.	216
B.6	An example of a <i>Link</i> implementation. The shown example demonstrates the case where one would want to filter the flow of data between two modules.	218
B.7	An example of a <i>Visitor</i> implementation. The shown example initializes all modules within a graph by calling the <i>setUp</i> method.220	
B.8	An example of an <i>Iterator</i> implementation. The shown example provides sequential access to a graph of modules in such a way that the items are returned with respect to their depth within the graph (i.e. shortest distance from a root). It is a level ordered <i>Iterator</i>	221

Chapter 1

Introduction

1.1 Motivation

This thesis focuses on development and improvement of algorithms for detecting Parkinson's Disease (PD) related symptoms in time series data. PD is a disorder of the central nervous system resulting in a loss of motor function, increased slowness and rigidity. Artificial intelligence (AI)-based techniques can be utilized to detect symptoms such as tremor or bradykinesia for the purpose of monitoring and treatment improvement. Those affected by PD bear a great burden and have to cope with a rather reduced quality of life. Considering the leading role of Germany, this is an even more pressing issue. In 2004, Germany inhabited the largest number of people with Parkinson's within Europe [9].

Even though it can manifest itself at any age, PD is among other diseases (e.g. Alzheimer's, dementia, chronic bronchitis) usually attributed to elderly subgroups of the population. Considering demographic changes of the last decades, the number of cases and burden of PD is expected to increase [61, p. 36]. The World Health Organization (WHO) estimates that around 5.2 million people were suffering from PD worldwide in 2004 [101]. Depending on the estimating organization, Europe inhabited 1.2 [61] - 2.0 [101] million of them in the same year.

In the early 19th century, James Parkinson first described in "An essay on the shaking palsy" [119] six cases showing (motor) symptoms such as a shaking hand or slowness of movement. Even after 200 years of advancements in medical technologies, the cause of PD remains unknown [44, 69, 83, 146]. As to what triggers the disease, it can only be speculated. Researchers are investigating various possibilities including viruses, environmental factors, aging and genetic causes [12, 45, 146], but no definitive answer can be given at this point in time. As a consequence, people with Parkinson's cannot be cured yet. Current treatments aim at slowing the progression of the disease, focus on symptomatic relief and attempt to lift the enormous burden of PD.

PD is a chronic, progressive, neurodegenerative disorder [12, 45, 69, 73, 146] which is generally characterized by a gradual loss of (motor) function (e.g. slowness and rigidity). The cardinal symptoms are bradykinesia, rigidity, tremor and postural instability [9, 45, 69, 73, 83, 146, 154]. These symptoms result from a dopamine deficiency in the substantia nigra [10], a part of the brain that is

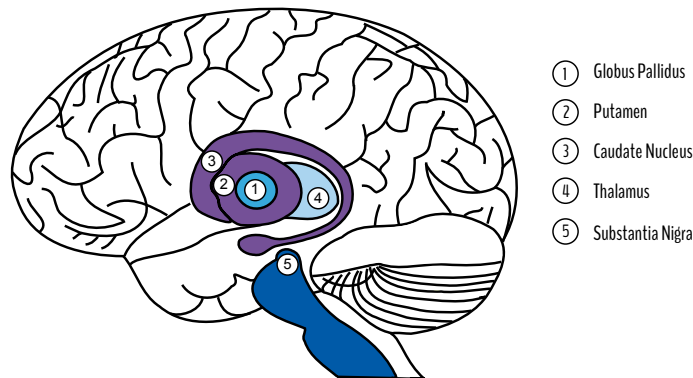


Figure 1.1: Illustrates structures of the brain related to the basal ganglia circuit. Latter is located in the upper end of the brain stem. The image is based on a figure which has been retrieved from Wikimedia Commons and belongs to the public domain.

located within the basal ganglia circuit (see Figure 1.1). Dopamine is a neurotransmitter involved in movement control [10, 174]. Usually by the time of diagnosis, a great number of dopamine-producing neurons have already diminished [146].

PD is a great burden, not just for people suffering from the disease but also for those being indirectly affected (i.e. relatives and caretakers). In an advanced stage of the disease and without proper treatment, patients are no longer capable of taking care of themselves. In the Global Burden of Disease (GBD) study, the WHO rated PD to be on the same disability level as: amputated arm, congestive heart failure, deafness, drug dependence and tuberculosis [101, p. 33]. Considering neuropsychiatric disorders, PD is after Alzheimer and Epilepsy the third most frequent cause of death in the world [101, p. 56].

A large number of symptoms have been shown by people with Parkinson's [69, 73]. The most visible and easily noticeable symptoms are related to motor functions. However, quality of life is also affected by numerous non motor symptoms (e.g. depression, sleep disorder, cognitive / neurobehavioral abnormalities, autonomic and gastrointestinal dysfunction) [9, 69, 73]. As the disease progresses, symptoms change and fluctuate (i.e. some symptoms simply disappear, while others (re-) appear), creating a unique symptomatic history for each individual patient. Unfortunately, in an advanced stage of Parkinson's further (drug-induced) symptoms may become apparent. Dyskinesia is one of these symptoms and results from a lengthy pharmacological treatment (i.e. several years). It manifests itself as an involuntary movement of body parts (e.g. rhythmical moving of upper body).

In order to reduce symptoms and compensate for the loss of dopamine, patients take medication such as levodopa. In early stages of the disease, prescribed medication is commonly given in pulses (e.g. pill taken every few hours). In later stages of PD, a continuous dose of medication may be administered to minimize symptoms. This change is commonly justified as effectiveness of these medications decreases and “wears off” with progression of PD. This results in

reappearance of symptoms before the next dose is taken. Furthermore, the so-called therapeutic window, in which the medication produces the desired effects, narrows. Consequently, moderate and advanced patients cycle between states in which they can move almost normally (ON state) and in which they show symptoms because the effects of medication have “worn off” (OFF state). Keeping the patient in the ON state is desirable. Depending on the stage of the disease, patients will fluctuate between both ON and OFF states several times during the day (e.g. medium-advanced patients cycle three to four times).

Various tools and techniques exist to help and assist PD patients. Unfortunately, the loss of dopamine cannot be monitored in the same way as it can be done with other diseases such as diabetes. Here, the blood sugar can be observed and a system can adjust the amount of required medication based on this observation. With PD this is a difficult and invasive task because measuring dopamine levels requires direct access to the brain stem. Thus most of the PD cases can only be confirmed after death through an autopsy [58]. To counteract this shortcoming, neurologists require detailed records on symptoms and motor states. However, instead of objective measures, neurologists have to rely on time-consuming manual and subjective measures (e.g. diaries) to improve treatment.

A lack of reliable and objective monitoring options is one of the primary reasons why no successful system for monitoring PD has been built so far. Furthermore, the large amount of symptoms and the fact that these change as the disease progresses thwart the building of such a system. Nonetheless, some projects pursue this very goal. E.g. HELP [65] and REMPARK [133] are two European research projects with a focus on building a system for detecting PD symptoms (and therefore the ON or OFF state) and administer medication accordingly.

Publications reveal a great number of techniques for automatic detection of PD motor symptoms which employ various AI-based methods such as neural networks (NNs) [13, 18, 34, 36, 37, 51, 78, 140], hidden markov models (HMMs) [136] and support vector machines (SVMs) [18, 34, 122]. Depending on the symptom and utilized sensors, various features are calculated (e.g. entropy [18, 34, 110, 122], spectral or fractal features [13, 31, 32, 77, 111, 115, 143, 155, 162, 171]). Over time, sensor signals are analyzed and compared to known samples of each symptom in order to recognize them. No matter whether these AI methods are continuous or window-based, all of them can be viewed as either data mining (DM) techniques and / or time series (analysis) algorithms. Much literature exists that presents algorithms for detecting a single symptom (e.g. [13, 33, 34, 37, 51, 98, 113, 136, 161, 171]). Considering the heterogeneous nature of symptom profiles, this is not sufficient in case of PD. Few publications focus on detecting multiple motor symptoms (e.g. [36, 122, 140, 143]), but even those rarely consider enough symptoms for use in real-world scenarios. In reality, multiple symptoms may overlap, thus increasing the chance of false negatives and false positives.

As there is no convenient way to effectively measure dopamine levels within the basal ganglia circuit, detection of ON-OFF state is essential for systems like HELP [65] and REMPARK [133]. They rely on proper recognition of PD (motor) symptoms in order to estimate the ON-OFF state. Thus, part of this thesis focuses on improvement of detection algorithms and development of new approaches. These efforts could provide a foundation for better monitor-

ing systems. Treatment of PD patients can be improved and burden reduced, consequently lowering the inconveniences for everyone involved (i.e. patients, caretakers, relatives, friends, etc.).

Unfortunately, no unified framework for time series analysis and DM applications exists that can be easily utilized for development of the above algorithms. Instead several frameworks and libraries would have to be used which are more or less well maintained. The development of a unified software-framework is also part of this thesis. Such a framework can be used for a systematic evaluation of PD symptom detection algorithms and thus providing a foundation for comparing different detection approaches. The framework is going to emphasize a stream-based processing of data samples. In doing so, the software-architecture of the framework implicitly reminds developers to implement their algorithms in a way that processes data samples iteratively and as they become available (i.e. online processing). Once completed, such a framework enables an easier development and evaluation of PD detection algorithms.

To summarize, the objective of this thesis is two-fold: (primary) development and improvement of algorithms for detecting PD related motor symptoms and (secondary) to develop a framework for time series analysis, whereas the algorithms of the primary objective will leverage AI-techniques.

1.2 Research Questions

This section formulates the research questions of this thesis. Furthermore, requirements, scope and limitations of a striven solution are outlined.

Due to the lack of a practical option to measure dopamine levels within the substantia nigra (basal ganglia circuit), ON-OFF state estimation is based on detection of motor symptoms. The difficulty of not being able to directly infer drug needs makes the development of PD monitoring systems (e.g. HELP [65], REMPARK [133]) unnecessarily difficult. In addition, no unified framework for time series analysis and DM applications exists which further increases development efforts. As it has been described above, the proposed outcome of this thesis are: to compensate the lack of a unified framework for time series analysis and to provide algorithms for detecting various motor symptoms of PD.

In more detail, the overall goal of this thesis is to pursue and answer the following research questions.

1. **How can a time series be represented? What is an adequate software-architecture for a DM and time series analysis framework?**

Depending on the data, a time series can be represented in different ways. However, a unified representation is essential for a proper time series analysis framework as it greatly simplifies implementation and architectural work. Analysis of time series can be tedious work, especially when algorithms need to be (re-) implemented from scratch or adapted every time a new data source (e.g. sensor) is added.

2. **Which Parkinson’s Disease symptoms can be detected and how can they be detected?**

Considering the wide range of symptoms, those must be identified that

are likely to be recognizable with off-the-shelf sensors. Whereas a reasonable low interaction and unobtrusiveness for use in daily life are highly desirable. Among the countless features of sensor signals and algorithms, that could be used to classify each symptom, representative features and appropriate algorithms need to be identified.

3. How can published state-of-the-art techniques for detecting motor symptoms of Parkinson’s Disease be improved?

Alone, the lack of a working PD monitoring system and youth of the field suggest that further progress is possible. As does the fact that a large number of publications consider a single motor symptom instead of multiple ones. Many publications claim to have achieved a specificity and sensitivity of up to 90% [34, 78, 114, 142, 182], which can still be improved. State of the art techniques need to be reviewed. The used kinds of sensors, extracted features from their signals and applied AI algorithms should be identified. This can then be used to develop new and / or improved approaches.

4. How well do the new / improved approaches perform when compared to state-of-the-art techniques?

Competitive algorithms need to be compared to one another. Where possible, state-of-the-art algorithms and newly developed approaches should be evaluated (i.e. in terms of speed, number of false positives and false negatives, online vs. offline detection, etc.).

These research questions can be seen as a guideline of what will be discussed throughout this thesis. As may be judged from the number of questions, most attention is devoted to improvement and development of PD motor symptom detection algorithms. Consequently latter research questions are more important and less effort is put into answering the first question.

1.2.1 Scope and Limitations

This thesis proposes the design and implementation of a software-framework for time series analysis and DM with reusable modules. The framework is intended to treat all data sources and data processing modules in a unified way. A model-driven approach and a number of design patterns, which have been described by Gamma et al. [57], are utilized.

Even though, the design and architecture do not suggest a specific programming language nor are they restricted to a specific one, the implementation is done in Java (version 1.7). This limits the potential working environments to those operating systems and devices supporting Java. However, future releases might target different programming languages such as C++, Objective-C or .Net programming environment. Thus supporting further devices and operating systems.

The second part of the thesis focuses on improvement and development of PD related motor symptom detection algorithms. Due to the wide variety of PD symptoms, only motor symptoms are taken into consideration. The development will be done with respect to the state-of-the-art, knowledge about the symptoms as well as labeled sensor signals of several PD patients that show a variety of symptoms and that have been diagnosed with PD. The development of the

algorithms may be considered to have acceptable results if both sensitivity and specificity are above 80%. For the final algorithms, a threshold of at least 90% is expected to be reached.

In general, it is the idea to provide neurologists with information (e.g. motor state or symptoms) which can then be used to make educated decisions to improve and personalize treatments. However, this thesis merely develops the tools and algorithms to support this scenario. The evaluation of these tools and algorithms is done with real data from PD patients, but they are not validated in a clinical setting.

1.2.2 Requirements

All requirements in this section have been derived from the above mentioned research questions. For easier affiliation, they have been grouped by research topic. The anticipated answers will be viewed in the light of these requirements.

The following requirements are related to the software-architecture of the time series analysis framework that is developed in Chapter 4. Here, the general idea is to implicitly promote online and modularized analysis in stream-based data.

- **Stream-based:**

The framework is expected to highlight a stream-based processing approach. By doing so the framework reminds application developers that a definitive start or end of data may not always be known. Instead, an online analysis for each data sample is implicitly suggested.

- **Iterative:**

Each data sample is read, possibly transformed and processed on its own. A module within the framework accepts individual data samples or a bulk of them and processes them one after another.

- **Scalability:**

Samples in data streams are not necessarily limited to a few hundreds or thousands, but may in fact be several million samples long (or even infinite). A framework for stream-based analysis must be able to cope with such diversity. Not all algorithms in the framework are required to adhere and handle such quantities, but the framework in itself certainly has to provide the means to handle a potentially infinite stream of information.

- **Flexibility:**

The flow of data, from one module to another module, must not necessarily be static. Although in many cases a static (unchanging) flow of information is sufficient, there are expectations in which a more flexible way (i.e. conditional branching) can be beneficial. E.g. incoming data does not have a statically predefined format and may vary. Depending on the variation, a separate module may be utilized to process the received data.

- **Reusability:**

The reusability of framework components and modules is essential. Use of design patterns [57] and unified interfaces for modules and their parameterization is intended.

- **Extensibility:**

The striven time series analysis framework must be expandable in the sense

that adding new data sources and algorithms (i.e. adding new modules) is not cumbersome. The cost of adding a new module must be weighted against the usability of the rest of the framework. The frequency of adding a new module should also be taken into account.

- **Support for distribution:**

A potential distribution of framework modules and components across multiple threads, processes or devices is desirable.

Analogous, the remaining items are related to the algorithms for detecting symptoms related to PD.

- **State of the art:**

The implementation of modules for newly developed algorithms and / or improved approaches must be compared to state-of-the-art implementations.

- **Real-time:**

The algorithms should be able to provide results in real-time, where real-time means that all required sensor signals can be either processed online and provide results either continuously or at appropriate intervals (e.g. every second, 10 seconds, etc.).

- **Portability:**

It is desirable to provide algorithms that could (at least in principle) run on less computational capable devices such as mobile phones (e.g. Android devices) or wearables.

- **Sensors:**

Data signals are to be retrieved from body-mounted sensors. In the spirit of portability and wearable computing, the data sources should be ideally placed as close as possible to the symptoms' origin, rather than having camera recordings of patients or other external measurements.

It should be kept in mind that these requirements merely present an outline. A number of (sub-) problems and constraints are connected to each of them. All requirements will be referred to in their corresponding chapters (mainly Chapter 4 for the framework and Chapter 6 for the algorithms).

1.3 Thesis Organization

The remaining chapters are organized as described in the following.

- **Chapter 2 ‘A Brief Review of Parkinson’s Disease and Time Series’:** Presents background information on the two main topics of this thesis: Parkinson’s Disease and time series. Symptoms, diagnosis and treatment of PD are outlined. Several common algorithms related to time series analysis and DM are summarized. This chapter establishes a common understanding of terminology that is used throughout the thesis.
- **Chapter 3 ‘Related Work Regarding Symptom Detection’:** Introduces various approaches of detecting symptoms of Parkinson’s Disease. Related projects and systems with a special focus on PD and detection of its symptoms are shown. Additionally, a number of frameworks and toolkits for time series analysis are summarized. This chapter answers

research question two and it provides a basis for discussing the striven solutions to the remaining research questions.

- **Chapter 4 ‘A Framework for Time Series Analysis’:** Shows the development of a modular framework for DM and time series analysis. Furthermore, a few applications and scenarios are summarized. With respect to the requirements and related work, this chapter proposes an answer to the first research question presented in Section 1.2.
- **Chapter 5 ‘Database: Patients and Their Symptoms’:** Briefly describes data acquisition and contents of the utilized database (DB). The idea of this chapter is to get an impression on the contents of the DB.
- **Chapter 6 ‘Indication of Parkinson’s Disease Motor Symptoms’:** Discusses several AI-based approaches for detecting motor symptoms related to Parkinson’s Disease. In the course of this chapter, a few algorithms for detecting motor symptoms of PD patients are developed. Thus, an answer to research question three is proposed. With regard to the requirements and related work, a solution is formulated.
- **Chapter 7 ‘Benchmark of Symptom Detecting Algorithms’:** Describes benchmarks of several symptom detecting algorithms that have been introduced in Chapter 3 and Chapter 6. The results are highlighted and discussed. This chapter holds an answer to the fourth research question with respect to the requirements and related work.
- **Chapter 8 ‘Conclusions’:** A summary of contributions made to the research fields as well as a list of conclusions and possible directions of future work are presented. This chapter closes the thesis.

Chapter 2

A Brief Review of Parkinson's Disease and Time Series

The purpose of this chapter is to constitute a common definition and understanding of various terminologies and notations. Therefore, terms like Parkinson's Disease (PD) and time series are going to be described in the course of this chapter. At first, notations and definitions are introduced. This is followed by an overview of PD and its symptoms as well as an introduction to the field of time series analysis and data mining (DM). This chapter concludes with a brief summary.

2.1 Definitions and Notations

This section describes a set of notations that are used throughout this work. Furthermore, several definitions are provided. It is the intention to supplement for potentially different notions depending on the reader's field of origin. The author encourages the reader to look through this section in order to verify whether an intuitive understanding of terminology and notations is present.

In later chapters, real-world data from PD patients are analyzed and processed. There, it will be the goal to recognize symptoms when they appear. This is a typical classification task and assumes availability of pre-recorded and annotated data. Part of the (annotated) data are used to train a machine learning (ML) algorithm in recognizing these symptoms, whereas the remainder of the (annotated) data are used to verify and evaluate the trained algorithms. Even though, the task of classification is not the only task that is commonly performed in a ML context (e.g. regression, outlier detection and clustering are further examples), the following notations and definitions do focus on this task in particular.

- **Algorithm:** Similar to a recipe, an algorithm includes a set of ordered instructions to reach a predefined goal. In case of a classification task, this could consist of all steps that are required to recognize certain symptoms

in a database (DB) of recorded signals (i.e. including all preprocessing, application of ML techniques as well as post-processing).

- **Data sample (or observation):** A value or a set of values that is considered to have been measured at the same instant of time. An example would be the water depth of a harbor or the location of an remote controlled (RC) plane. Without loss of generality, an observation can be represented by a vector A_i measured at time T_i such as

$$A_i = (a_{i,1}, a_{i,2}, \dots, a_{i,n}) \quad (2.1)$$

where $A_i \in \mathbb{R}^n$; $T_i \in \mathbb{N}$; $i, n, m \in \mathbb{N}$; $i, n, m > 0$ and $i \leq m$. Here n defines the number of attributes within vector A_i (i.e. the set's size). $a_{i,j}$ represents the j^{th} attribute within the data sample A_i ($j \in \mathbb{N}; 0 < j \leq n$). i marks the data sample's position in a time series A with a length of m . T_i refers to the point in time at which the observation A_i was made.

- **Time series:** A series of data samples measured over the course of time. The data points are ideally spaced at uniform time intervals (i.e. $T_i - T_{i-1}$ is constant $\forall i \in 2 \dots m$). However, this is not a requirement. An example would be to measure water depth as flood and ebb come and go. Without loss of generality, a time series can be modeled by the matrix such as

$$A = (A_1, A_2, \dots, A_m)^T = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \dots & & & \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{pmatrix} \quad (2.2)$$

where A_i with $0 < i \leq m$ ($A_i \in \mathbb{R}^n$; $i, n, m \in \mathbb{N}$; $i, n, m > 0$ and $i \leq m$) are the corresponding data samples within the time series A . m defines the number of observations in the time series A and n represents the number of attributes per data sample A_i . Again, $T_i \in \mathbb{N}$ identifies the point in time at which the measurement was done.

- **Sampling rate (or sampling frequency):** Defines the speed with which measurements or observations are made. Here, this corresponds to the number of data samples A_i that are observed within a second. Without loss of generality, the sampling rate for time series A can be represented by

$$S_r = \frac{m}{\sum_{j=2}^m T_j - T_{j-1}} \quad (2.3)$$

where $T_i \in \mathbb{N}$; $i, m \in \mathbb{N}$; $i, m > 0$ and $i \leq m$. Here, m corresponds to the length of time series A and T_i identifies the point in time at which a data sample A_i was measured. The sampling rate S_r is assumed to be constant (e.g. if $S_r = 10Hz$ then A has a length of 50 observations after 5 seconds). In more general terms, the number of observations m (in time series A) increases linearly with $m = S_r * t$ where t is the timely length of time series A .

- **Dataset:** A collection of time series (i.e. any number of time series). Typically, at least two different datasets are available (i.e. one for the training phase and another one for the testing phase).
- **Online Processing:** Measured data samples are processed as they become available. In some cases, it may be beneficial to group multiple data

	condition positive	condition negative
test positive	true positive (TP), hit	false positive (FP), false alarm, type I error
test negative	false negative (FN), type II error	true negative (TN), correct rejection

Table 2.1: Highlights the relationship among various commonly used terminology from a ML point of view. Here terms like true positive and false positive are listed.

points and process them in a bulk. The group can also be considered as a single data point, but as part of time series with a lower sampling rate. Nevertheless, the amount of data should not add up to more than a few seconds or minutes in the context of PD (this number depends on application and context). Calculating a moving average (see Section 2.3.2) is an appropriate example.

- **Offline Processing:** Data samples are first captured and stored for later analysis and processing. As opposed to handling data samples as they become available, the time difference between capturing and actual processing is rather large (i.e. typically several hours, days or even weeks). An example would be location data (e.g. global positioning system (GPS)) from an RC plane during a flight. There the data is first recorded and later visualized or processed.
- **Real-time:** A time constraint that guarantees the availability of a response to an input within a predefined period of time (under reasonable conditions). The actual time interval depends on the context in which the term is used. E.g. in real-time computer gaming the response should be available within a fraction of a second whereas the recognition of bradykinesia may well be delayed by several seconds and can still be considered real-time.
- **Gold standard (or ground truth):** The best available labeling for a dataset (under reasonable conditions). An example would be labels from medical professionals which describe the occurrence of tremor at rest or other symptoms. A less preferable (but in some cases still acceptable) labeling would be from medical novices. Without loss of generality, the labels for a time series A can be represented in the form of another time series B of equal length m with $A_i \in \mathbb{R}^n$; $B_i \in \mathbb{R}^p$; $i, m, n, p \in \mathbb{N}$ and $i, m, n, p > 0$. Here m represents the length of time series A and B . i identifies a particular data sample or label within these time series. n and p correspond to the number of attributes.

The described terminology should provide an idea of their meaning in the context of this work (staying within the realm of classifying PD symptoms from pre-recorded time series). Furthermore, a set of goodness and badness measures is shown below. These enable assessing previously trained algorithms. A few complementary definitions can be found in Table 2.1.

- **Positive Predictive Value (a.k.a. precision):** The percentage of correctly recognized positive labels. The precision of an algorithm in detect-

ing a symptom (such as resting tremor) can be described with this ratio. It highlights the number of actually detected tremor episodes among all the tremor episodes that were believed to have occurred by the algorithm.

$$PPV(TP, FN, TN, FP) = \frac{TP}{TP + FP} \quad (2.4)$$

- **Negative Predictive Value:** The percentage of correctly recognized negative labels. In case of an algorithm for detecting resting tremor, this highlights the number of correctly classified data samples without resting tremor among all those data samples that were believed to show no resting tremor.

$$NPV(TP, FN, TN, FP) = \frac{TN}{TN + FN} \quad (2.5)$$

- **True Positive Rate (a.k.a. recall and sensitivity):** The ratio of correctly classified positive data samples among those that should have been classified as positive data samples. For the previously mentioned algorithm, this corresponds to the number of tremor episodes that were correctly detected among those that should have been detected (i.e. not those episodes that are just believed to be tremor episodes by the algorithm). It shows how sensitive the algorithm is in picking up episodes of tremor.

$$TPR(TP, FN, TN, FP) = \frac{TP}{TP + FN} \quad (2.6)$$

- **True Negative Rate (a.k.a. specificity):** The ratio of correctly classified negative data samples among those that should have been classified as negative data samples. In case of the tremor algorithm, this highlights the number of data samples that were correctly detected to not exhibit tremor among those data samples that should have been detected (i.e. not just those data samples that were believed to not show signs of tremor by the algorithm). It shows how specific the algorithm is in picking up episodes of tremor.

$$TNR(TP, FN, TN, FP) = \frac{TN}{TN + FP} \quad (2.7)$$

- **Fall-Out:** The percentage of incorrectly recognized negative data samples among those that should have been recognized as negative data samples. This measure basically represents the opposite of the “true negative rate” (i.e. $FO = 1 - TNR$). For the previously mentioned algorithm, this highlights the percentage of data samples that were detected as tremor but should not have been classified as such.

$$FO(TP, FN, TN, FP) = \frac{FP}{TN + FP} \quad (2.8)$$

- **Accuracy:** The overall percentage of correctly recognized labels. In case of the tremor algorithm, this highlights its accuracy and shows the number of correctly detected labels within the (entire) time series.

$$ACC(TP, FN, TN, FP) = \frac{TP + TN}{TP + FP + TN + FN} \quad (2.9)$$

2.2 Parkinson's Disease

PD is a chronic, progressive, neurodegenerative disorder [12, 45, 69, 73, 146] that has been first described by James Parkinson in 1817 [119]. It affects the movement of those suffering from the disease and it is typically characterized by a loss of (motor) function, increased slowness and rigidity. Despite radical advancements in medicine and medical technology over the previous two centuries, reason and cause of PD remain unknown [44, 69, 83, 146]. Thus, most treatments aim at reducing severity and frequency of motor complications. They do not cure nor have they been shown to slow the progression of Parkinson's. Due to PD's heterogeneous nature (i.e. genetic vs. non-genetic, etc.), it is not even clear whether PD can be considered a single disease [83].

Suffering from PD is a great burden as it considerably decreases the quality of life. This is due to a gradual loss of function and increasing disability to take care of oneself. The World Health Organization (WHO) considers the burden of PD to be on the same disability level as an amputated arm, drug dependency, or suffering from congestive heart failure, deafness and tuberculosis [101]. A great number of motor and non motor symptoms are related to the disorder (see Table A.1 and Table A.2 in Appendix). The cardinal symptoms are bradykinesia, rigidity, tremor and postural instability [9, 45, 69, 73, 83, 146, 154]. However, a number of non motor related symptoms (e.g. sleep disturbances, depression, psychosis, autonomic and gastrointestinal dysfunction as well as dementia) may occur as well [45, 69, 73, 85, 146].

The disease is generally attributed to elderly people and is rarely diagnosed before the age of 40. It is estimated that the mean age of onset is about 65 years [146]. In contrast, the prevalence is about 1.5% for people at the age of 60 or above [170]. Thus, given the aging (European) population PD is expected to become a public health problem of increasing magnitude. Nonetheless, the disease can manifest itself at any age and cases are known in which individuals were in their twenties or thirties. In addition, PD is after Alzheimer and Epilepsy the third most frequent cause of death among neuropsychiatric disorders [101, p. 56].

The WHO estimated that about 5.2 million people were suffering from PD worldwide in 2004 [101]. In the same year, it has been estimated that 1.2 million [61] to 2.0 million [101] people were suffering from PD in Europe alone. As the authors of "Cost of disorders of the brain in Europe 2010" [61] note, a patient bears a yearly cost of about 11.200 Euro, which adds up to a total cost of almost 14.000 million Euro within Europe in 2010. To set this into perspective, Germany inhabited the second largest number of people with Parkinson's in the same year (highest number of subjects with PD was in Italy with close to 240.000 people, compared to about 220.000 people in Germany). They are followed by France and United Kingdom (UK) with 190.000 and 110.000 people respectively. In Germany, the total cost of PD was roughly 2.800 million Euro whereas on average each subject bore a cost of about 12.800 Euro.

Although cause and reason remain unknown, it has been shown that PD results from a loss of dopamine-producing neurons. These neurons are located in the substantia nigra within the basal ganglia, which can be found in the brain stem (see Figure 1.1). Those neurons produce a neurotransmitter called dopamine that is used by the basal ganglia circuit to control motor functions such as movement of hands, legs, etc. In order to provoke movement this neu-

rotransmitter is utilized to pass messages from the brain to the corresponding body part and its muscles. Concerning people with Parkinson's, these messages are not smoothly transported to the corresponding part of the body. This causes difficulties in controlling movement, a predominant sign of PD. Planning, initiating and executing movements may be affected [73]. Furthermore, it has been shown that a link between extent of Lewy bodies (LBs) / Lewy neurites (LNs) and clinical symptoms exists [96, 126].

Unfortunately symptoms start to appear at a stage in which a great number of dopamine-producing neurons have already diminished [83]. It has been noted [83] that not all dopaminergic tracts are equally affected. Furthermore, additional neurotransmitter abnormalities may also occur in PD and manifest themselves in one or more of the previously mentioned non motor symptoms such as depression or sleep disturbances. As to what actually triggers the degeneration of neurons it can only be speculated. Theories include aging, genetic causes, environmental exposures (such as exposure to toxic material) and viruses [12, 45, 158]. For more details on the genetic landscape of PD the interested reader may want to have a look at [23].

To summarize, PD motor symptoms are mainly due to a lack of dopamine production in the substantia nigra. As a result, the smooth transmission of messages from the brain to the muscles is thwarted. Unfortunately taking dopamine by itself does not have the desired effect as it cannot pass the brain-blood barrier. Thus, it cannot migrate from the blood-stream to the brain where it is needed. However, a precursor of dopamine called levodopa resolves this issue, because the body is able to convert it into dopamine. Thus, the supply of dopamine is refreshed by taking levodopa and patients regain control over their movements. Nonetheless, due to reduced dopamine production or lack thereof in the basal ganglia the refilled dopamine does not last and motor symptoms are bound to reappear unless another dose of medication is taken. Thus patients experience periods of mobility and immobility.

In an early stage of the disease, these fluctuations are predictable and can almost entirely be avoided. However, as PD progresses the desired effects of levodopa "wear off". Thus the effectiveness of medication has a tendency to decrease, thus causing symptoms to reappear earlier than before [45].

With continued progression (degeneration of dopamine producing neurons) and reduced effectiveness of medication, the only working option is to increase the dose in the hope of rendering the patient mobile. Consequently, the dosage needs to be adjusted from time to time. This also means that fluctuations between periods of mobility and immobility happen more frequently as the disease progresses. As mentioned above, at first these fluctuations are predictable. However, fluctuations become less deterministic and eventually totally unpredictable (ON-OFF phenomenon) in later stages of the disease [116].

The fluctuations between periods in which individuals show almost no motor symptoms (known as ON period) and periods in which motor symptoms are present (known as OFF period) are a major problem for people with Parkinson's. Depending on the stage of the disease several fluctuations between ON and OFF periods may occur every day (e.g. medium-advanced patients cycle three to four times per day).

In general, ON and OFF periods can be described as follows.

- **ON:** While being in an ON state (i.e. patient is on medication), motor

symptoms are almost invisible. Usually only professionals recognize them in an ON state. Patients report to feel fairly fluid and in control of their movements.

- **OFF:** While being in an OFF state (i.e. patient is off medication), patients may experience symptoms such as tremor, freezing of gait, bradykinesia, etc.

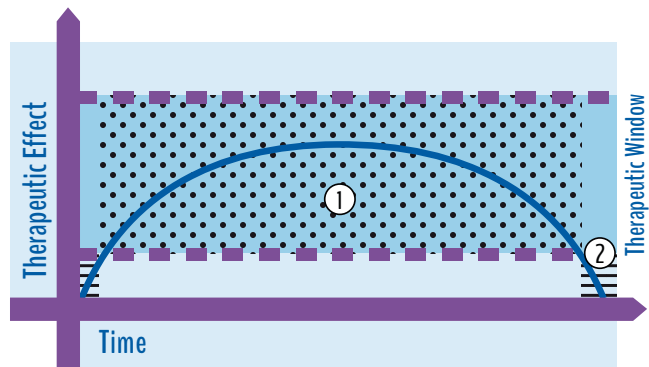
Recent research investigates continuous medication intake as an option to further optimize PD treatment. It is believed that continuous intake of medication (e.g. through the means of a pump) may be more effective than use of an impulse-based intake (e.g. in form of pills taken at regular intervals). Impulse-based intake usually delivers a higher dose at a time than continuous intakes would require. This makes sense as pills only last for a certain period of time and thus need to deliver more medication in order to gap the time between the actual intakes. However there is also more medication “lost” due to metabolism outside of effective areas. A continuous medication intake could potentially help patients to maintain a constant (and healthy) level of dopamine. Thus preventing patients from entering the OFF state. Furthermore, the overall medication intake could potentially be reduced as well as loss of medication due to metabolism while being just as effective or even more effective at the same time.

Due to side effects of medication further symptoms may occur. Dyskinesia is probably the most noticeable and certainly one of the most disabling (motor) symptoms. It presents itself as an involuntary movement of body parts (e.g. upper body or lower extremities) and becomes apparent after having taken an “overdose” of dopamine. The length of dopaminergic treatment and dose have been shown to correlate with the appearance of dyskinesia [97, 148]. Thus, it is desirable to avoid unnecessarily high doses and prolong this type of medical treatment as long as possible.

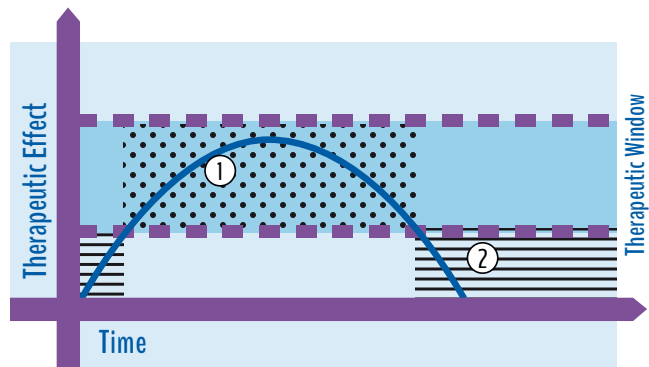
The social implications of PD are a major problem for those that are suffering from the disorder. During ON periods PD patients are almost indistinguishable from healthy people. However, the appearance of freezing events (i.e. the patient literally freezes and is unable to continue walking) or dyskinesia can be most inconvenient and embarrassing. Reasons for such events include improperly timed medication intake (e.g. either because a pill is not taken or the dosage is not correct) or simply because the disorder is in an advanced stage (i.e. medication is no longer as effective as it used to be for that particular patient).

When trying to minimize the frequency of OFF states the only working solution is usually to increase the dose of medication. However, this is likely to give rise to dyskinesia at some point in time. Therefore, it is a common problem for neurologists to find a medication dose within the boundaries of the so-called therapeutic window. In other words, they try to find a dose that is just large enough to avoid OFF periods (lower boundary of therapeutic window) but at the same time not large enough to provoke dyskinesia (upper boundary of therapeutic window). A major obstacle is that the therapeutic window narrows with disease progression (see Figure 2.1). This is because the effectiveness of medication decreases and further dopamine neurons continue to diminish as Parkinson’s advances.

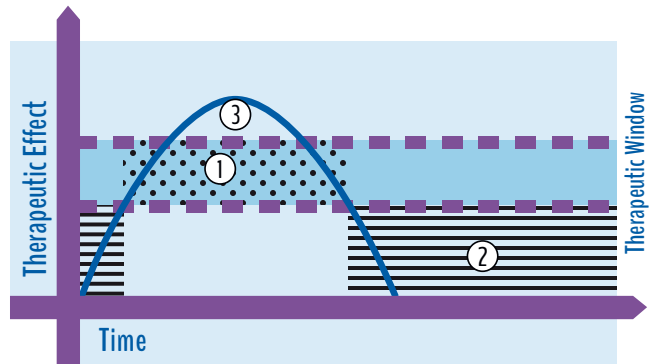
This concludes the general introduction of PD, remaining subsections will focus on specific aspects of the disorder. Motor and non motor symptoms as well



(a) Initial stage of PD. The therapeutic window is relatively large and medication lasts relatively long.



(b) Medium / moderate stage of PD. Effectiveness of medication and therapeutic window have shortened.



(c) Advanced stage of PD. Therapeutic window has narrowed a lot and medication wears off relatively soon. Dyskinesias are likely if therapeutic window is overshoot.

Figure 2.1: Shows the therapeutic window of PD and how its boundaries narrow over time. This Figure is intended to indicate general tendencies with regard to disease progression (i.e. decrease in effectiveness and narrowing of therapeutic window). These graphs are not necessarily scaled properly, thus units on both axis have been avoided. Areas identified by number 1 correspond to periods in ON, number 2 represents periods in OFF and number 3 highlights periods with dyskinesias.

as diagnosis and treatment of PD are described. Furthermore, several rating scales regarding the state and progression of PD are presented.

For a more detailed picture on how the disorder might actually progress, Korczyn et al. [84] provide an interesting summary on the works and theories by Braak and colleagues [25–27].

2.2.1 Symptoms

This section highlights a number of classical motor and non motor symptoms of PD. As mentioned above, those suffering from Parkinson’s may show a broad range of symptoms (see Table A.1 and Table A.2 in Appendix). With advancement of PD, the symptomatic profile of individual patients changes. Not only do these symptoms fluctuate in terms of rate of progression and severity but also in terms of appearance (i.e. symptoms may disappear while others (re-) appear). Everyone’s experience is unique [73, 85].

In general, the symptoms can be categorized as follows.

- **Motor symptoms**, which are symptoms related to (any kind of) movement. They include but are not limited to: tremor, rigidity, freezing (motor blocks), falls, bradykinesia (slowness of movement) and postural instability. Among other things, speech may also be affected.
- **Non motor symptoms**, which are not related to movement. These symptoms include but are not limited to: dementia, sleep disturbances, depression, psychosis, autonomic and gastrointestinal dysfunction.

The symptoms begin to emerge after dopamine levels within the substantia nigra have reduced by around 50%-70% [45, 83]. This may imply that a “long pre-clinical state, of 20 years or more, predates the appearance of motor symptoms” [83].

Given the nature of this thesis, most attention is devoted to motor symptoms. Nonetheless, the latter category is not ignored and a few common non motor symptoms are highlighted toward the end of this section. At first, various cardinal and classical features are described.

Cardinal And Classical Symptoms

Several cardinal and classical symptoms are listed as follows.

- **Tremor at rest** (a.k.a. rest tremor or resting tremor) is an easily recognized motor symptom of PD. It manifests itself as an involuntary, unilateral (one-sided) shaking of an extremity (e.g. hand, foot, etc.). In general, extremities of the upper body are more affected than those of the lower body [85, 146]. The shaking generally occurs at a frequency between 4-6 Hz [73]. However, various different frequencies can be found in literature (e.g. 4-8 Hz [12], 3-5 Hz [146] and 4-10 Hz [48]). Given the name tremor at rest, the tremor is only present when muscles are at rest and dissolves during sleep as well as with action (i.e. voluntary movement of affected extremity) [85].

Tremor at rest is a very common symptom of PD and most people with Parkinson’s experience a shaking or trembling of one of their hands at an early stage of the disease [146].

- **Rigidity** is a common symptom shown in early stages of PD. Generally speaking, rigidity refers to stiffness [85] and manifests itself as an increased resistance of limbs. Leading to a reduced flexibility of the ankles [85]. Rigidity may be apparent both distally (e.g. wrist and ankles) and proximally (e.g. neck and shoulders). Interestingly, the symptom is more prominent during mental task performances [146] and reinforcing maneuvers at the contra-lateral limb [73, 146]. It is also associated with pain [73].

An increased rigidity may give rise to postural deformities (e.g. flexed neck, knees or elbows). However this usually becomes apparent at later stages of PD.

- **Bradykinesia** refers to slowness of movement [73, 85]. It usually appears in very early stages of the disease [146]. Bradykinesia is a common symptom of PD and other movement disorders [73, 146] and it is characteristic for basal ganglia disorders [73]. Furthermore, bradykinesia may also be apparent as loss of facial expression (see Figure 2.2) [45, 73, 85], impaired swallowing [73, 85], cramped handwriting (see Figure 2.3) [45], reduced blinking [73] or decreased arm swing while walking [73].

Depending on the severity, movements may not only be slowed (bradykinesia), but also diminished (hypokinesia) or completely abrogated (akinesia). Often family members notice a slow performance of daily activities and decreased reaction times before affected subjects become aware of the symptom themselves. Thus bradykinesia may be present even before PD has been diagnosed.

Experiments indicate a dependency between bradykinesia and emotional state of the subject. I.e. despite being immobile, subjects who become excited were able to free themselves and make quick movements such as catching a ball or start running if someone screams “fire”. Thus subjects may simply have problems to initiate or continue movement without an external trigger but otherwise have intact motor programs [73].

- **Freezing of gait (FoG)** (a.k.a. freezing or motor blocks) is a form of akinesia, which presents itself as an inability to initiate or continue movement [73, 146]. Motor blocks are a common symptom, experienced by people with Parkinson’s (although it does not occur uniformly) and can affect various extremities (e.g. arms and legs) as well as the face [73]. After onset of the symptom, it typically lasts for several seconds and disappears afterward. In severe cases, it can be apparent for several minutes.

Freezing greatly impairs the quality of life of those affected and is one of the most disabling symptoms. It is usually not seen in early stages of PD and it is a common cause of falls [73, 146].

Freezing happens in several situations such as beginning to walk (start hesitation), crossing a busy intersection or walking through a narrow doorway (hesitation in tight quarters). Five subtypes of situations have been identified: start hesitation, turn hesitation, hesitation in tight quarters, destination hesitation and open space hesitation [73, 147].

- **Postural instability** refers to a reduced balance [73, 146]. An increasing number of falls is a common result of this symptom [73, 146]. It usually occurs in late stages of the disease and results from a loss of (postural) reflexes [73].



Figure 2.2: Example of a person suffering from hypomimia or otherwise called loss of facial expression. The masked face can be a result of bradykinesia or related symptoms. The image has been retrieved from Wikimedia Commons and belongs to the public domain.

Catherine Metzger
13 Octobre 1859

Figure 2.3: Example of a person showing signs of micrographia. PD symptoms like bradykinesia may lead to a small, cramped handwriting. The image has been retrieved from Wikimedia Commons and belongs to the public domain.

In addition, several other symptoms may be apparent in people with Parkinson's. E.g. a reduced stride length or gait speed and falls are common. As noted earlier, symptomatic profiles change with progression of the disease. Symptoms fluctuate in both severity and composition resulting in a unique experience for each patient [85].

In early stages of the disorder, further motor related symptoms such as dysarthria, dysphagia, hypophonia and sialorrhoea [73] may also be apparent in people afflicted with PD. Such speech disorders can be marked by word finding problems ("tip-of-the-tongue phenomenon"), breathy speech as well as soft and monotonous speech [73].

In medium and advanced stages, drug-induced symptoms are likely to become evident. Dyskinesia is a common symptom that falls into this category. It manifests itself as an involuntary, uncontrollable movement (that unlike resting tremor does not disappear with voluntary movement). This stands in contrast to the otherwise typical symptoms of slowness and rigidity. A long-term treatment of levodopa is known to increase the risk of dyskinesia.

Postural instability and FoG are common causes for falls. However, it has also been noted that alone the fear of falling can have further negative impacts on balance of patients.

Non Motor Symptoms

Even though this thesis focuses on recognition of motor symptoms, a few non motor symptoms are mentioned for the sake of completeness. A more detailed list may be found in the Appendix (see Table A.2).

- **Psychosis and hallucinations** may occur in PD [69, 84, 146]. Hallucinations are usually visual in nature and have a tendency to show in later stages of the disease [69]. Patients have reported to see small animals, children or deceased relatives [69].
- **Sleep abnormalities** are a common problem [68, 69, 73, 146]. People have reported to experience problems sleeping (e.g. due to depression, illusions, restless legs and / or other causes). Another common problem of people with Parkinson's is excessive sleepiness [69]. In some cases, patients having trouble to sleep at night also endure excessive (daytime) sleepiness due to their sleep hygiene or lack thereof. Even during regular daily activities, patients have been seen to fall asleep within seconds [69, 141]. This symptom is more often apparent in an advanced stage of the disease and male patients [69].
- **Sensory abnormalities** may also be present in people with Parkinson's [69, 73, 128]. E.g. sense of smell and vision may be affected. Furthermore, changes in weight and / or pain can be experienced [84]. All of which can be apparent in early stages of PD and sometimes even before a clinical diagnosis [84].
- **Autonomic dysfunctions**, such as excessive sweating, erectile impotence and / or problems with the urinary system, may also occur in PD [69, 73, 76, 84]. Some of which have been shown to occur prior to clinical manifestations of motor symptoms [84]. However, medication intake may exaggerate these features.

- **Cognitive and neurobehavioral disorders** present a great burden to those affected. Dementia and cognitive slowing have been shown to be apparent in early stages of the disease [69, 84]. However, it occurs more frequently in patients in later stages of the disease [69, 73, 146].

It should be noted that some of the above described non motor symptoms are not a direct effect of PD but rather a product of pharmacological treatment. However, this does not make them more bearable. They are a reality for people with Parkinson's.

2.2.2 Diagnosis

An accurate diagnosis of PD is difficult, due to the broad range of symptoms and lack of disease-specific biomarkers [73, 135]. Even though the past decades have shown advancements in that matter, diagnosis still remains a clinical one [45]. Patients are diagnosed based on occurrence or absence of motor symptoms, positive response to dopaminergic treatment and their health history.

The most conclusive tool for PD diagnosis is autopsy [146] and many PD cases can only be confirmed post-mortem [58]. It can be utilized to show a decrease in dopamine-producing neurons in the substantia nigra. Even in nowadays, no reliable test for a definite diagnosis of PD has been found [135]. Due to the fact that PD is clinically diagnosed (i.e. without a definite biomarker) and considering the diversity of symptomatic profiles, a fairly high percentage of PD diagnosed patients is estimated to be wrong [12]. Post-mortem studies have highlighted these difficulties and suggest an alternative diagnosis in up to a quarter of patients [45]. This is not surprising, knowing that most symptoms of PD can also be apparent in one or several other disorders and show only minimal or no differences at all [83]. Nonetheless, the diagnosis of PD may be uncomplicated when classical symptoms are shown by a patient. This, however, is not necessarily the case. Especially in early stages of the disease, signs and symptoms can overlap with other syndromes [73, 130, 163], making it difficult to distinguish PD from other forms of parkinsonism. Thus a differential diagnosis is typically performed in order to single out people with PD from people with other parkinsonian disorders and conditions.

LBs can be considered a pathological hallmark of PD. These bodies are thought to accumulate in dopamine containing neurons undergoing degeneration within the substantia nigra [45, 83]. There is a well-established link between PD symptoms and extent of Lewy pathology. Post-mortem studies have found that extent of LBs correlates with clinical symptoms [96]. With rare exceptions, all sporadic and familial PD brains were found to include LBs and / or LNs [96, 126].

Depending on country and institution, several guidelines for diagnosing PD exist. These typically specify supportive and exclusion criteria and usually differentiate between criteria for possible, probable and definite PD. However, it can be noted that the diagnosis of PD is mostly based on the presence of a combination of cardinal symptoms, a positive response to dopaminergic treatment as well as a set of exclusionary features [45, 146]. Several of these guidelines can be reviewed in Section A.2.

Samii et al. [146] summarizes such a guideline (see Table A.9 for more details). Here a patient is diagnosed with clinically possible PD if at least one of the conditions is met: asymmetric bradykinesia, asymmetric rigidity and / or

asymmetric resting tremor. If any two of these conditions are met by a patient then PD is a clinically probable diagnosis. However, for a definite diagnosis, conditions for clinically probable diagnosis need to be met and a positive response to anti Parkinson drugs (such as levodopa or dopamine agonists) must be evident. On the other hand, presence of any exclusion criteria suggests an alternative diagnosis. Among the exclusion criteria are exposure to drugs (which may cause parkinsonism), severe dysautonomia, early gait disturbance, early dementia or family history of PD (in more than one family member).

Jankovic [73] published a similar guideline (see Table A.7) as presented by Samii and colleagues [146]. Just like the previous one, these instructions originate from the UK PD Society Brain Bank. However, in Jankovic’s version the clinical criteria for diagnosis of probable PD is described only. This guideline consists of three subsequent steps: (1) bradykinesia and at least either one of rigidity, rest tremor (4-6 Hz) or postural instability need to be apparent, (2) other causes of parkinsonism must be ruled out and (3) at least three further supportive conditions must be identified (see Table A.7).

Several rating scales have been proposed to quantify disease progression in people afflicted with PD. Section 2.2.4 highlights a set of commonly employed rating scales such as the unified Parkinson’s Disease rating scale (UPDRS) or Hoehn & Yahr scale (HYS).

2.2.3 Treatment

As noted before, no treatment has been proven to cure PD. Instead treatments aim at minimizing and preventing motor fluctuations, dyskinesia and symptoms (motor and non motor) as well as slowing disease progression. Several medical (e.g. levodopa, dopamine agonist and inhibitors) and surgical (e.g. ablation and transplantation) options are available. Even approaches utilizing Chinese Medicine have been proposed [74]. More recent modalities have evolved to use ultrasound and biological therapies [160]. The more traditional options are highlighted in the following.

Medical Options

The most common and widely used drug for treatment of PD is levodopa [12, 45, 83, 146]. It is a precursor of dopamine, which is able to cross the blood-brain barrier. In contrast, dopamine itself cannot pass the blood-brain barrier. For most patients levodopa is the medication of choice as it has positive effects on motor symptoms. To prevent metabolism of levodopa outside the brain, it is often taken in combination with further drugs. However, long-term treatment comes at the cost of further (drug-induced) disabilities which are dependent of the taken dose and duration. E.g. dyskinesia being a typical example of such kinds of side effects. Thus patients often face the dilemma of keeping the same dose and rendering themselves rigid and immobile or increasing the dose and risk an increased chance of dyskinesia. Generally, it is preferable to use as small doses of levodopa as possible (and postponing the appearance of dyskinesia). It gives also rise to the question of when to introduce a dopaminergic treatment. Even though there is no concrete answer to this question, the overall idea is to postpone any medical treatment as long as possible until the symptoms become troublesome [45, 146].

As the disease progresses further dopamine-producing neurons diminish and, thus, higher (and / or more frequent) doses of medication are required to compensate the loss in order to keep the patient mobile. This, however, gives rise to motor fluctuations (i.e. cycling between periods of mobility and immobility). At first, these fluctuations are predictable, but they become less deterministic over time.

Dopamine agonists work by enhancing the effectiveness of dopamine [12]. It is often used in combination with levodopa [12, 146]. They are increasingly utilized as first line treatment (on their own) due to their positive effects on motor symptoms and delayed introduction of levodopa [45, 146]. When comparing to levodopa, dopamine agonists come at a price of increased side effects and decreased UPDRS scores while remaining the same quality of life and reducing motor symptoms [45].

Monoamine oxidase B (MAO-B) inhibitors have also been shown to have a positive effect on motor fluctuations and symptoms [45]. They work by slowing the breakdown of dopamine at the synapses [12]. Catechol-O-methyltransferase (COMT) inhibitors have the ability to lengthen the half-life of levodopa [45, 83]. Thus COMT inhibitors can be used to smooth out motor fluctuations while reducing levodopa doses and periods of immobility [45, 83].

To summarize, various treatments have been proposed in order to minimize the number of motor fluctuations as well as their severity. Nonetheless, levodopa remains the drug of choice at some point in the therapy. It is only a matter of time until levodopa is introduced, until then the increased chance of dyskinesia is balanced against side effects of other treatments. The interested reader may want to have a look at [176, p. 243ff] for a more detailed view on conventional treatment options.

Further medications are known to work against PD but has been found to have insufficient efficiency (e.g. Amantadine [40] or Imipramine [12]).

Surgical Options

Deep-brain stimulation is currently the most preferred surgical option for treatment of PD [95, 146]. This is due to the fact that it is a less irreversible procedure when compared to general ablative and restorative surgery [146]. Realistic surgical options are briefly described as follows.

- **Ablation:** Part of the brain is either surgically destroyed (pallidotomy) or surgically removed (thalamotomy). Such procedures have been reported to reduce tremor and improve motor symptoms. However, this comes at a risk of infarct, seizures and even death [19, 146, 169].
- **Stimulation:** A procedure (deep-brain stimulation) in which one or more electrodes are embedded in certain parts of the brain. Using those electrodes to stimulate the surrounding tissue relieves tremor, but does not improve other symptoms [149]. The general risks of brain surgery apply to this procedure just the same. Further complications regarding implanted hardware and stimulation itself may occur. However, stimulation related problems can be resolved by adjusting stimulation variables (e.g. amplitude and frequency).

Both options have successfully been used to improve symptoms of PD (e.g. tremor and dyskinesia) [29, 87, 146]. Even though ablation is a risky and in-

vasive procedure, there are patients and situations where such procedures are appropriate [146]. However the latter one remains the more preferable and more commonly applied choice [160]. Due to its destructive nature, ablation is generally avoided whenever possible.

Another approach transplants tissue in hopes of restoring neurodegenerated parts of the brain. It has been reported that dopamine-producing neurons of transplanted tissue largely survived [117]. However, despite the survival of the neurons most patients developed dyskinesia which remained even “after withdrawal of dopaminergic treatment” [146]. Thus, at the time of writing, transplantation is not a feasible option and generally avoided.

Similarly to diagnostic procedures of PD (see Section 2.2.2 and Section A.2), there exist clinical guidelines regarding the application of brain surgery in cases of PD. An exemplary guideline for deep brain stimulation (DBS) is shown in Table A.10. Here strong commitment from the patient (and social support by family and friends) to keep showing up at medical appointments, a definite diagnosis of PD (as defined in Table A.9 and Table A.8) as well as clearly defined ON and OFF periods are among the inclusion criteria. Whilst severe brain atrophy, drug-induced parkinsonism, dementia or psychiatric issues count to exclusion criteria.

2.2.4 Rating Scales

Several rating scales have been proposed in order to quantify the rate of progression in people with PD. These scales and questionnaires are applied where objective measures (i.e. biological markers, functional tests, etc.) are not available or not cost-effective. Despite their subjectivity, questionnaires and rating scales present basic tools that are widely used throughout medical and research communities.

The UPDRS and HYS are two commonly utilized instruments for quantifying the evolution of PD. Nonetheless several alternatives such as the Schwab & England scale (SES), Webster scale and others exist. Here, several of these rating scales are briefly described. For a more detailed review, the interested reader is encouraged to examine the references. However, it should be noted that most of these scales require a medical background and that these scales can be quite extensive (e.g. in [60], a thirty-page long questionnaire is shown).

- **Unified Parkinson’s Disease Rating Scale:** The UPDRS is a commonly used multi-domain rating scale. It is used to assess the severity of PD. It resembles a collection of questions that are to be answered by a neurologist. These questions cover many aspects of PD (i.e. tremor, FoG, dyskinesia, depression, sensory complaints, etc.). It is comprised of four sub-scales (i.e. section one: mentation, behavior and mood; section two: activities of daily living (ADL); section three: motor examinations; section four: complications). A revised version of the original UPDRS is described by the Movement Disorder Society (MDS) in [60]. The original scale is described in [53].
- **Hoehn & Yahr Scale:** The HYS [67] is a stage-based approach that has been widely adopted and it is largely accepted. It is focused on postural instability as indicator for impairment and disease progression. Stages range from “Unilateral involvement only usually with minimal or no func-

tional disability” (i.e. stage one) to “Confinement to bed or wheelchair unless aided” (i.e. stage five) in increments of one. However, a modified HYS has also been proposed which uses smaller increments. In [59], the original HYS as well as the modified adaptation are described in more detail.

- **Schwab- & England-Scale:** This scale quantifies the ability of a person to perform ADL. Here the rating is expressed as a percentage indicating the person’s ability to perform ADL (i.e. 100% meaning complete independence while 0% meaning complete dependence / bedridden). More details on the SES can be found in [150].

Depending on the version of the UPDRS, the HYS as well as Schwab- and England-scale are actually included in the UPDRS. For the interested reader, a comparison of the HYS and SES can be found in [102]. Here the correlation between these scales as well as the discrepancies between the patient’s judgment and the doctor’s impression is touched upon. Additional rating scales, such as the clinical impression of severity index (CISI-PD) [100] and rating scale for gait evaluation (RSGE) [99], exist. An overview of rating scales and questionnaires can be found in [176, p. 755ff].

2.3 Temporal Data Mining

Temporal DM refers to a collection of techniques that intend to extract “useful” knowledge from temporal data [105]. The definition of “useful” is highly dependent on the target domain and task (e.g. a stockbroker would certainly be more interested in forecasting abilities rather than knowledge about past stock events). In nowadays, more and more data are being generated, collected, processed and analyzed. The continued growth of the internet and increasing use of social media platforms illustrate this fact. They have been growing in such a way that a manual analysis is no longer possible or at least no longer feasible and associated with great costs. However, growth is not limited to internet-based data but instead applies to almost any kind of data (e.g. digitized books, medical data, personal health records, stock and foreign exchange markets, gene sequences, etc.). Time series (or temporal data) represent a subsection of all available data, which has become increasingly important over the past few decades. Their application is of vital importance in fields like financial data forecasting and medicine.

A great deal of this section is devoted to the process known as knowledge discovery in databases (KDD). Fayyad et al. define KDD as a non-trivial process which aims at “identifying valid, novel, potentially useful, and ultimately understandable patterns in data” [54]. This process consists of five steps which are highlighted in Figure 2.4 along with the typical flow of information. Each of these steps is separately described in the remainder of this section. Even though Figure 2.4 and the structure of this section suggest a fairly rigid and linear procedure, this is not necessarily the case. In many KDD applications all five steps are utilized to some extent. However, they are by no means required at all times. E.g. in some cases, interpretation of patterns (see Section 2.3.5) spawns new questions and might result in repeating one or several steps, such as selecting further attributes and refining applied transformations to the original

data. Thus the KDD process should be considered more like a guideline and good practice rather than a strict algorithm.

At the heart of the KDD process is a concept called DM. Here, temporal DM refers a sub-field of DM but with a stronger focus on the time domain. It inherits the basic ideas and techniques from DM while specializing them and using them in a different context (i.e. harvesting of useful patterns and knowledge from temporal data). Several of these techniques are introduced in the remainder of this section (e.g. classification, regression, clustering).

There have been some confusions regarding the use of terms like DM, time series analysis and KDD. This is likely due to the fact that DM, time series analysis and KDD are very interwoven concepts. In general, the KDD process addresses the problem of “mapping low-level data (which are typically too voluminous to understand and digest easily) into other forms that might be more compact (for example, a short report), more abstract (for example, a descriptive approximation or model of the process that generated the data), or more useful (for example, a predictive model for estimating the value of future cases)” [54]. Time series analysis is very similar to the KDD process. However, it is specialized to leverage information within the time domain. DM originates as a (single) step of the KDD process, it “consists of applying data analysis and discovery algorithms” [54] in order to harvest useful patterns. However, depending on the domain these terms have been used almost interchangeably and have only slightly different meanings. Independent of domain and discipline, they share the overall goal of discovering useful knowledge and / or patterns in data. Also certain steps and actions are common among them (i.e. selection, transformation, preprocessing, etc.). Thus, it makes sense to refer to them in the course of this section. Part of this section is devoted to describing not only techniques of temporal DM but rather a more global view on the whole process of discovering patterns in (temporal) data. Thus a sequence of actions, which is part of any reasonable serious and complex DM application, is highlighted.

For completeness’ sake, it should be mentioned that the KDD process is not one of a kind. Several closely related processes exist (e.g. CRISP-DM [151]). Many of them overlap to a certain extend and are tailored to a particular domain. However, most of them can be viewed as specializations of the KDD process described in [54].

For an easier understanding of the succeeding sections, assume to have access to a (fictive) medical database from a local hospital. Think of it as a children hospital, which specialized in chronic diseases. Several nurses have noticed a strange behavior in some children and have raised concerns regarding their well-being. Therefore, the responsible doctor has asked to review their data (e.g. temperature levels) and look for patterns, which might shed light on their situation. This information may also proof to be useful in the future as this behavior might be recognized sooner and therefore appropriate actions can also be taken earlier.

2.3.1 Selection

The general assumption is that numerous data sources may be available, but not all of them are necessarily useful in the given context or application. Consequently, this first step deals with selecting appropriate data sources and attributes. However, a sufficient understanding of the target domain needs to be

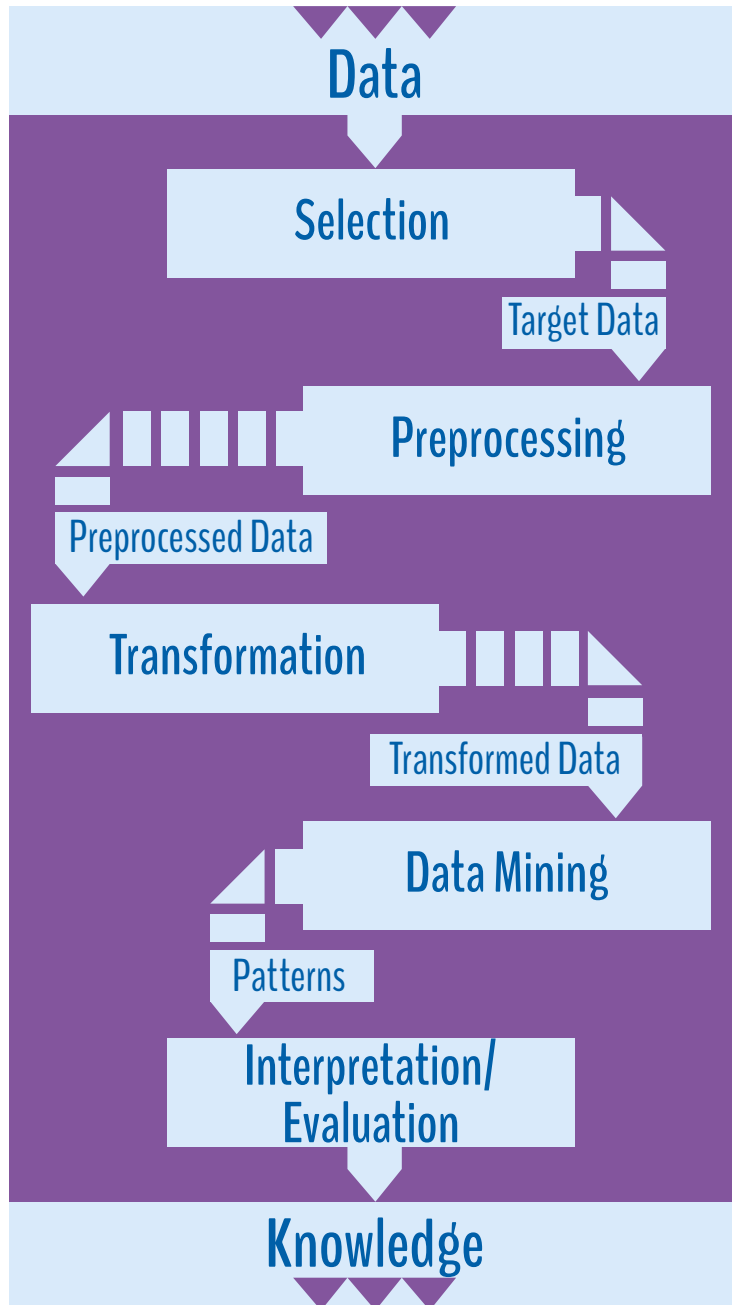


Figure 2.4: The general flow of information and the five steps of the KDD process are outlined. Note that the actual process is not as linear as suggested by this representation. Results may improve if steps are performed multiple times. The image is based on a figure by Fayyad et al. [54].

developed at first. This knowledge can then be utilized for a proper data selection and / or acquisition. It is one of the most important steps of the KDD process as its results will affect all succeeding steps. A sufficient knowledge of the target domain eases a selection of appropriate attributes (i.e. variables) or data samples, which are ultimately used to perform (temporal) DM on. Thus, selecting inappropriate data at this point is likely going to result in discovering meaningless patterns or no patterns at all. Consequently, the process will fail to extract useful patterns and / or knowledge. In the absence of sufficient domain knowledge, however, this might not be as obvious as it should be. One can easily continue to falsely believe in having obtained new, valid knowledge while the discovered patterns are completely meaningless to a domain expert. Instead they are most likely to represent some random patterns in “poorly” selected data.

Furthermore, the amount of chosen attributes should be considered. A selection of too less attributes or data samples will likely result in the discovery of meaningless patterns, if any at all. Thus, the selected data does not carry enough information for the desired patterns to emerge. The same applies to inappropriate attributes or data samples. On the other hand, selecting too many attributes or data samples is also likely to end in discovery of many useless patterns. The more data are presented to DM algorithms the more patterns are potentially found. Consequently, too much information causes many patterns to emerge which makes it unnecessarily difficult to sort out and actual patterns are likely to be overseen in the flood of “potentially” useful patterns.

Imagine to be involved in the above-described scenario, the responsible doctor (i.e. domain expert) has asked you to review the children’s data. He suggested looking for signs of fever and elevated body temperature first. Obviously one should select body temperature and maybe further attributes related to fever. In the course of the KDD process these data are then transformed, analyzed and the resulting patterns are likely to reflect an answer to the doctor’s suspicion. However, not including the body temperature or further correlated variables as an attributes will very unlikely result in any satisfying answer to the doctor’s suspicion. It should be kept in mind that simply narrowing the data selection to a few body temperature related attributes can also result in discovering patterns that are not relevant to the actual problem or may not be “actual” patterns at all (but rather a random finding or observation). Thus if one wants to find temperature-related patterns and one searches long enough, then one is likely to find them (whether these patterns are the true source or not). Analyzing body temperatures is enough to verify the suspicion of fever but will certainly not suggest the true source of the fever (e.g. food poisoning).

In general, the more knowledge about the target domain is available the better. It can be used to rule out certain attributes and utilized to acquire others. Thus, the discovered patterns are more likely to reflect useful and meaningful patterns.

2.3.2 Preprocessing

Cleaning and preprocessing of data are dealt with in the second step of the KDD process. The desired result is a clean and consistent dataset, where the definition of clean and consistent heavily depends on the domain and task. However, when talking about “cleanliness”, there are at least three aspects worth highlighting:

missing data fields, presence of noise and handling of errors.

In most cases, some sort of preprocessing is required especially if multiple data sources are involved (e.g. multiple databases from different hospitals or institutions). In such a case, it is easily possible that different names are used to represent the same or similar attribute (e.g. “temperature” and “temp.”). However, even if attributes have identical names across all databases (or sources), their type, precision and unit may differ nonetheless. E.g. body temperature can be measured in degree Celsius or Fahrenheit. Another data source might not be interested in a precise body temperature at all, instead whether the body temperature is below 37°C, between 37°C and 40°C or above 40°C might be of interest. This highlights that even if multiple sources contain the same (or similar) attributes some form of preprocessing may still be required.

Consider the aforementioned scenario in which the responsible doctor suggested to look for signs of fever. To ensure the well-being of all children, a nurse measures their body temperature every few hours. Usually the measurements are taken at 8:00h, 12:00h, 18:00h and 22:00h. Under normal conditions, several minutes earlier or later do not have a significant influence as the body temperature is not expected to change dramatically within, say, 30 minutes. But what happens if the temperature was not measured? E.g. a reason could be that the particular child was on a field trip with several other children (thus all participating children missed a measurement) or the responsible nurse was tied to a “never-ending” staff meeting (thus she could not take measurements). Those are just two of many possible situations that can prevent measurement of the body temperature. In some cases, missing data fields may simply be left empty. However, there is also the option to replace empty fields with meaningful values, where the definition of meaningful again depends on the domain and task. These fields could be filled with a sensible default value or their value could be estimated by using adjacent measurements (e.g. using linear interpolation) [105, p. 23].

Another aspect of “cleanliness” is the presence of noise (i.e. a random error). This error can be due to several factors like faulty equipment (e.g. thermometer not properly working), environmental conditions (e.g. humidity, warmth, etc.), human error (e.g. nurse did not wait for the measurement to complete, some nurses measure temperature underneath the tongue, while others prefer to measure in the ear or underneath the arm), the device itself has an error / noise rate (even when properly working) and many others. For an example of such noise consider Table 2.2 and the corresponding graph in Figure 2.5. Looking at the graph it becomes clear that the body temperature of this particular child is for the most part around the average body temperature of 37.1°C. However, there seem to be two measurements clearly below the average body temperature. Depending on the context, these measurements may be considered outliers and may need to be smoothed out. Binning and moving averages are two methods, which are commonly used when wanting to smooth out such outliers / anomalies.

In binning, the data are first divided into bins (or buckets) of equal size and then each bin is smoothed. Smoothing of bins can be done with a variety of methods. Two popular approaches are *mean* and *median* smoothing, where all values in a bin are replaced by either their mean or median of that particular bucket. Listing B.1 highlights this process while Figure 2.6 and Table 2.3 illustrate the smoothed data in relationship to the original data. Of course, the size

Index	Day	Time	Temperature
i	$a_{i,1}$	$a_{i,2}$	$a_{i,3}$
1	1	08:00h	37.2°C
2	1	12:00h	37.8°C
3	1	18:00h	37.1°C
4	1	22:00h	36.9°C
5	2	08:00h	35.8°C
6	2	12:00h	37.5°C
7	2	18:00h	37.6°C
8	2	22:00h	37.1°C
9	3	08:00h	36.9°C
10	3	12:00h	36.2°C
11	3	18:00h	37.3°C
12	3	22:00h	37.8°C

Table 2.2: Lists the body temperature of a child over a period of three days. The average temperature is 37.1°C. The numbers are represented as time series A with a length of $m = 12$ and $n = 3$ attributes.

Bin	Mean	Median
37.2, 37.8, 37.1, 36.9	37.25	37.15
35.8, 37.5, 37.6, 37.1	37.0	37.3
36.9, 36.2, 37.3, 37.8	37.05	37.1

Table 2.3: Shows the calculated results based on the data samples shown in Table 2.2 for both binning methods (i.e. *mean* and *median*).

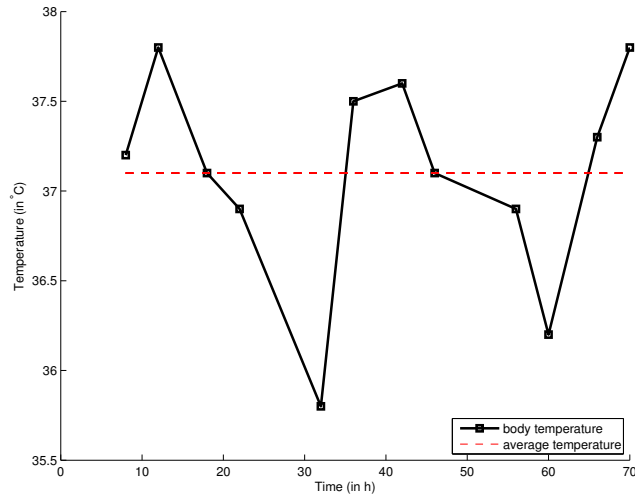


Figure 2.5: Shows the body temperature of a child over a period of three days. Note the two clear outliers below the “average” temperature line (dashed). For more details on the values refer to Table 2.2.

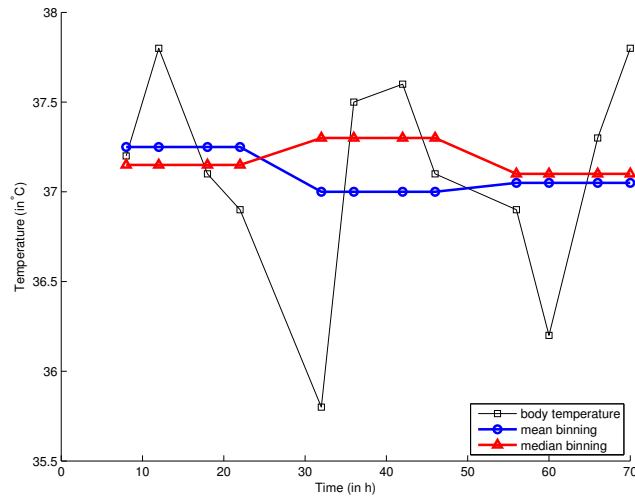


Figure 2.6: Illustrates *mean* and *median* binning if applied to body temperature data samples shown in Table 2.2. The size of a bin is four, so that a (single) bin represents measurements from an entire day.

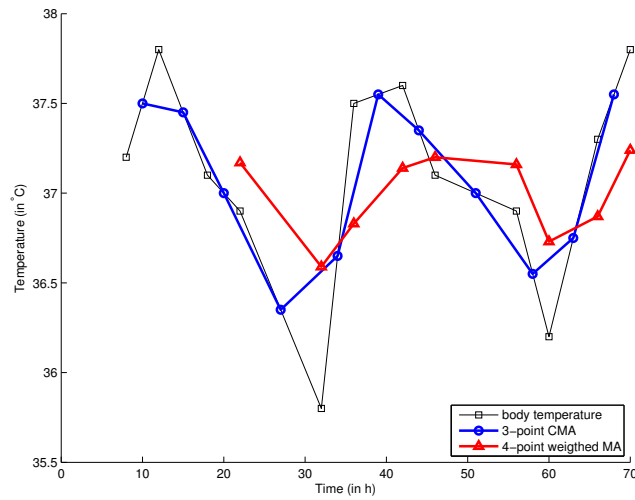


Figure 2.7: Illustrates the results when a 3-point central moving average (CMA) and a simple 4-point weighted moving average (MA) are applied to the body temperature data samples listed in Table 2.2.

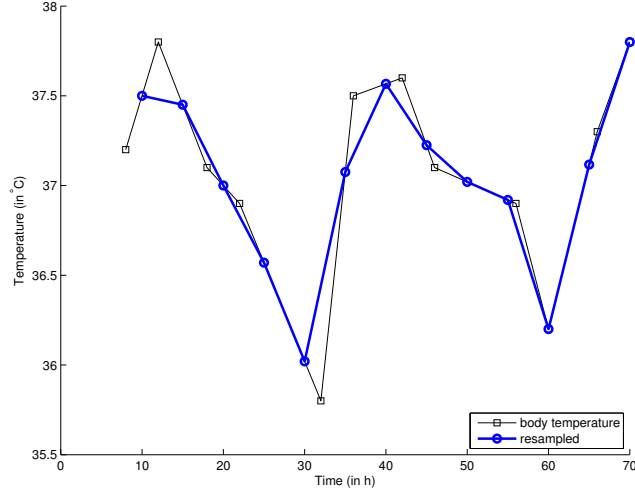


Figure 2.8: Shows resampled body temperature data listed in Table 2.2. For an easier reference, the original data samples are also included.

of the bins depends on the domain and task, so does the smoothing method.

The MA, a.k.a. rolling average (RA), can also be utilized to smooth out short term fluctuations. Many variations of this particular approach exist, however, it can be said that all approaches calculate some form of (weighted) mean with respect to a set of past and / or future values. Depending on the domain, future values may not always be available at the time of calculation (e.g. when working with live streaming data). Thus it is generally preferable to utilize approaches relying on past values only (as opposed to future values) whenever possible. However, even if future values are required by the chosen approach then data can simply be buffered and calculated once enough data has been accumulated. This approach, however, delays calculation by as many future measurements that are required which might not be an acceptable tradeoff in all domains. Also depending on the domain, the number of considered values varies as well as their weighting (i.e. more recent values may be more important than older ones). One approach is called 3-point CMA. This particular approach uses both past and future values around the point of interest [105, p. 24]. They are all equally weighted. The results when applied to the data (shown in Table 2.2) can be reviewed in Figure 2.7. Consider the function

$$CMA(A, n, i, j) = \sum_{p=0}^{n-1} \frac{a_{i-p,j}}{n} \quad (2.10)$$

where $a_{i,j}$ corresponds to an attribute of data sample A_i within the time series A . Here, $i, j, n \in \mathbb{N}$; $i, j, n > 0$ and $n \bmod 2 = 1$. n defines the number of data samples that are consumed for each computation and needs to be odd. Another variation of the MA approach might use the past four values where the most recent value is the most important one (i.e. higher weighting). This is also illustrated in Figure 2.7.

$$WMA(A, n, i, j) = \sum_{p=0}^{n-1} \frac{(n-p) * a_{i-p,j}}{\sum_{p=1}^n p} \quad (2.11)$$

where $a_{i,j}$ corresponds to an attribute of data sample A_i within the time series A . Here, $i, j, n \in \mathbb{N}$ and $i, j, n > 0$. n defines the number of data samples that are consumed for each computation.

Furthermore, it should be considered that at the beginning and end of time series there are usually not enough data points available to apply MA techniques. There are two extremes for dealing with this issue: skip them (no calculation is done where not enough data are present) or calculate them anyway. Here, the first option has been chosen (see Figure 2.7). For the CMA, the average of the first two values denotes the new value for the one at 10 o'clock. This is because the timing must also be taken into account.

Considering temporal aspects contained in the data of Table 2.2 (and Figure 2.5), the sampling rate or frequency may need to be adjusted or resampled for certain DM algorithms (i.e. some algorithms assume that data samples were recorded at equidistant time intervals). Reasons for this include that the samples were recorded either at inappropriate time intervals (e.g. measurements were not taken at equidistant time intervals) or the timing itself is not as accurate as required (e.g. due to measurement errors or uncertainties the timing might be off by a few seconds or minutes). However, this can also be viewed as a specialized missing-values problem. In general when resampling data, one tries to estimate missing values and determine what values would have been measured if they were actually measured at that particular point in time. To estimate them, some sort of interpolation is usually applied (e.g. linear interpolation [105, p. 23, p. 124ff]).

For example, the body temperature from Table 2.2 was not measured at equidistant time intervals. Instead of taking the measurement every six hours, they were taken at intervals of four, six, four and ten hours. This, of course, has obvious practical reasons. Nonetheless, applying a simple linear interpolation technique (as shown by Equation 2.12) may result in an estimated graph shown in Figure 2.8. Here, a new data sample is estimated based on the attributes of two adjacent data samples A_i and A_{i+1} . It will typically reside somewhere between these data samples A_i and A_{i+1} .

$$resample(A, i, j, \Phi) = a_{i,j} + (a_{i+1,j} - a_{i,j}) * \Phi \quad (2.12)$$

where A_i and A_{i+1} belong to time series A . i indicates the data sample within the time series. j represents the j^{th} attribute. Φ identifies the position of the new data sample relative to $a_{i,j}$ and $a_{i+1,j}$.

2.3.3 Transformation

Once a dataset has been cleaned up and consistency is ensured, data are usually transformed in some way. The third step of the KDD process mainly aims at data reduction and projection. As in previous steps, this one also depends on the domain and goal of the task. In general, the idea is to find representable features within the acquired data samples. Furthermore, techniques for dimensionality reduction are applied to reduce the number of attributes (i.e. variables

under consideration) and number of data samples (i.e. decrease amount of data). Finding appropriate invariant representations may also be helpful. Data reduction and normalization are the main objectives in many practical scenarios due to variability and a flood of (available) data. The idea behind this is to relieve the computational burden of DM algorithms.

Some DM algorithms perform better if all values of the variables under consideration are within a certain range (e.g. $[-1, \dots, 1]$, $[0, \dots, 1]$, etc.). In theory, the range of values does not matter. However, the range of values does have an effect on performance of (DM) algorithms in reality where representational and computational limitations exist. Applying normalizations techniques is a common way to avoid performance degradations through large values. Two popular ones, called *min-max* normalization and *z-score* normalization, are described in the following.

Consider the same body temperature history shown in Table 2.2. In order to compute the *min-max* normalization, the minimum value and maximum value must be available. They can be obtained by searching the dataset (i.e. complete search or estimated by a partial search) or they are depended on the particular domain and attribute. In latter case, they can be estimated by domain experts. However, they can also be limited by the precision of hardware used for taking measurements (e.g. a standard clinical thermometer has a predefined operational range and could not be used to measure temperatures above 42°C or below 32°C). Equation 2.13 is used to perform *min-max* normalization on the data from Table 2.2, where *min* and *max* refer to the minimum and maximum value of the dataset respectively. Both values are used in conjunction with a data sample A_i (from the original dataset as opposed to the normalized one) to calculate the normalized value for each of the attributes (of A_i). Figure 2.9 illustrates the results if applied to the temperature example.

$$MinMax(A, i, j) = \frac{a_{i,j} - min}{max - min} \quad (2.13)$$

For *z-score* normalization, mean and standard deviation of the considered variables are assumed to be known. In some cases, those values can be estimated as they are dependent on the domain and considered attributes. However, in general the entire dataset must first be iterated in order to determine them. Once the mean and standard deviation are at hand, they can be used to normalize the data as described by Equation 2.14. The mean and standard deviation are represented by *mean* and *std* respectively. The normalized value is based on the original value *mean*, *std* and $a_{i,j}$ for time series A with data samples A_i and their corresponding attributes. Again the results of applying *z-score* normalization to data samples in Table 2.2 are shown in Figure 2.9.

$$zScore(A, i, j) = \frac{a_{i,j} - mean}{std} \quad (2.14)$$

For a proper normalization, it should be taken into account that there are those time series which are stationary (i.e. mean, variance and auto-correlation do not change [105, p. 122]) and those that are non-stationary. The most pragmatic way of dealing with non-stationary is to utilize some sort of sliding window technique (i.e. partial search of dataset) where the window size highly depends on the task. Here the above algorithms would be applied to each individual sliding window separately.

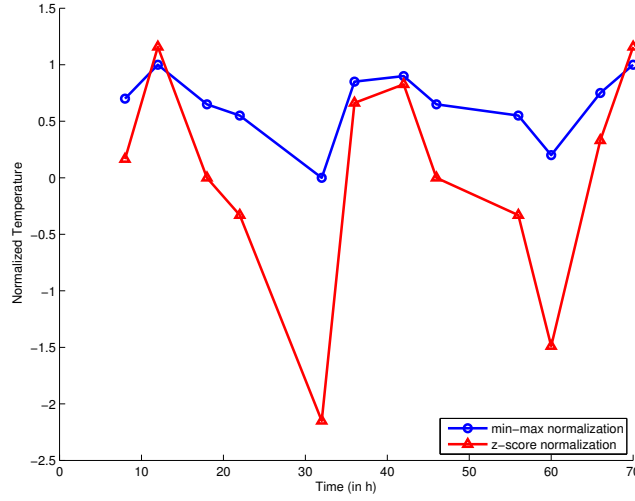


Figure 2.9: Illustrates *min-max* and *z-score* normalization applied to the body temperature example from Table 2.2. The required (minimum, maximum, etc.) values have been determined by searching the entire dataset. For convenience, these values are: minimum = 35.8°C, maximum = 37.8 °C, mean = 37.1 °C, standard deviation = 0.6 °C².

As mentioned, dimensionality reduction also plays an important role in this step. There are several methods for reducing and summarizing data. In [131], Ratanamahatana et al. highlight four representation approaches, which can be used to reduce the amount of data. These approaches are shown hereafter.

- **non-adaptive:** The algorithms can be used independently of the data at hand. This approach will typically reduce the number of data samples (or their dimensionality). Consequently, the representation is fixed by the algorithm (i.e. representation does not change regardless of the data that are being presented to the algorithm). In case of piecewise aggregate approximation (PAA), time series are divided into a fixed number of segments which are replaced by the segment's average. This category also includes but is not limited to: PAA [80], discrete wavelet transform (DWT) and discrete Fourier transform (DFT).
- **data-adaptive:** Representation schemes of this type adapt to the data. Here, the combination of data and algorithm determine the final representation. In case of symbolic aggregate approximation (SAX) (and assuming that a Gaussian distribution is present in the time series), the PAA method is applied and its results are further discretized. The distribution is divided into areas of equals probability which are then used to replace data samples with a symbol that has been assigned to the area. Singular value decomposition (SVD), SAX [90], indexable symbolic aggregate approximation (iSAX) [153], adaptive piecewise constant approximation (APCA) are also examples of this kind of approach.
- **data-dictated:** Time series are also replaced with fewer data samples (or with data samples of reduced dimensionality) but the original data samples influence the final representation. Here representations of time

series are entirely dictated by their data samples. Thus, algorithms of this category can be applied to any time series. One approach, called clipping [16], represents time series data by transforming it into a series of zeros (i.e. value below average of time series) and ones (i.e. value above average).

- **model-based:** The (time series) data are represented by some sort of model. The time series are abstracted into states for which the probability of transitioning from one state to another state is determined. These probabilities can then be used to model the states as well as transitions between them. E.g. hidden markov models (HMMs) [129] as well as various statistical models fall into this category.

Furthermore, the authors of [131] point out that representation schemes typically have two desirable properties: (1) no false dismissals and (2) dimensionality reduction. Reducing the number of dimensions also lowers the complexity (e.g. when comparing multiple datasets with each other). However, the first property refers to the fact that two similar datasets should also be similar in the transformation space. This means that a distance measure can be found in the transformation space that guarantees to be less or equal than the distance in the original data space [153]. This property is also called the lower bounding property.

Regarding summarization techniques, rather global characteristics (e.g. mean, median, mode, run-length, skewness, entropy, etc.) are utilized when compared to representation approaches. Thus they typically result in a great dimensionality reduction and consequently in a (great) loss of information. Consequently, it is quite common to represent data with a multitude of summarizations. This may counteract the loss of information, while still providing a great dimensionality reduction.

The remainder of this section introduces several representation algorithms and outlines various summarization techniques. Their description is not meant to be complete but rather to provide a first impression. For more details, the interested reader is referred to the corresponding references in literature.

Representation

Over the previous decades, numerous time series representations have been proposed and evaluated in literature. Most of them have been introduced or have their roots in the preceding century. Examples for each of the four categories (i.e. model-based, data-adaptive, non-adaptive and data-dictated) have already been given. Here a few of them are briefly highlighted.

PAA works in a similar way to *mean binning*. The authors of [80,90] describe a method in which a time series is divided into unisized segments and eventually each segment is replaced with its mean. This is one of the simplest representation approaches and can rival other more complex approaches like dynamic time warping (DTW) and wavelet transforms [105, p. 37]. The representation does not need to be complex in order to work effectively.

A related approach is APCA [79], a data-adaptive approach. It comes from the makers of PAA. A time series is split in segments of varying length and each segment is approximated by a constant value. The length of each segment is chosen in a way that minimizes approximation errors. Keogh et al. [79] outline

that this approach is generally able to approximate areas of low details with a single segment while multiple segments are used to approximate areas of high detail. However, comparing to PAA, this comes at a cost of requiring to store two values for each segment (i.e. length and value). Mitsa notes that this approach is useful for “bursty” signals [105, p. 43].

Further approximation approaches are SAX and iSAX [90], which are also based on PAA by Keogh et al. [80]. Both are based on results from PAA and further discretize them with symbols (i.e. a string alphabet) of equal probability. This is possible if the normalized time series is Gaussian distributed and thus the time series can be divided into areas of equal probability. Each of the areas is assigned a symbol (e.g. “a”, “b”, “c”, etc.) and all values from the PAA are replaced with their corresponding symbol. This approach has been successfully applied in classification, clustering and anomaly detection scenarios [90]. In [153], Shieh et al. introduced a multi-resolution indexing method called iSAX. It can be applied to very large temporal datasets that were previously out of reach.

Summarization

A large number of summarization techniques have been introduced in literature. To name a few of them: run-length-based signatures, histogram-based signature, statistics-based and statistical measures and many more. These and other approaches are briefly outlined in [105, p. 46ff].

Common statistics-based summarization approaches include but are not limited to: mean, median, mode, variance and standard deviation. They are all easy to calculate and provide a great dimensionality reduction. However, they also provide a rather simplistic and global view on a dataset, a lot of information is lost compared to the original data. Thus, it is common to utilize multiple of these techniques to represent time series. Mitsa briefly outlines all mentioned approaches in [105, p. 47].

The run-length-based signature of a time series highlights the number of occurrences and length of consecutive and identical values. In other words, run-length corresponds to the number of consecutively occurring data points that have identical values. Based on this signature, various measures can be formulated. E.g. the presence of either short or long run-length could be emphasized depending on the domain and task.

In [35], the authors used a histogram-based approach to summarize time series. The histogram shows the frequency of all values within a dataset (or window). Various statistical measures can be defined based on such a histogram. Skewness (i.e. “asymmetry of the histogram’s shape” [105, p. 50]) kurtosis (i.e. “peakiness of histogram” [105, p. 50]) and entropy (i.e. “amount of information and randomness” [105, p. 51]) are examples of these measures.

2.3.4 (Temporal) Data Mining

Fayyad et al. suggest to apply DM algorithms to the resulting dataset after having performed selection, preprocessing and transformation. The intention of the fourth step should be to discover useful and meaningful patterns. It represents the “heart” of the KDD process. Depending on the goal and type of data, several different kinds of algorithms can be utilized. Three common categories of

these algorithms are classification, clustering, and regression. However, several others, such as outlier detection or rule learning, can just as well be utilized.

Classification algorithms belong to the supervised learning approaches. This is because data, used for training, must be labeled or pre-classified. The algorithm tries to generalize from the presented training samples in order to be able to classify future, unknown samples. The term supervised is used because the data are annotated by a (human) teacher either directly or indirectly. A classification algorithm learns a function that maps from some input to a discrete value. Thus, such algorithms can only be used to identify the class or category for a (single) data sample, where the class can be one of a finite number of different values (or choices).

Some learning tasks, however, are required to output a continuous function which is not limited to discrete values. Algorithms of this category are grouped under the term regression. In contrast to classification algorithms, they can also be utilized (to some extent) to predict future values. This ability makes them especially interesting in fields like finance or medicine where predictions about future values can have a great impact. Otherwise, regression approaches also belong to supervised learning tasks.

Clustering is an unsupervised learning approach. Algorithms in this category aim to discover clusters (or groups of similar data samples) within the presented data. These clusters provide information regarding the relationship of data samples within a cluster and with data samples of adjacent clusters. Algorithms in this category do not require a (human) teacher but instead solely depend on the presented data. Thus, they are unsupervised.

In this step, the original goal of DM is revisited and used to decide on the category of algorithms that should be utilized for the task. E.g. learning thresholds for having an increased, normal or decreased body temperature would certainly be a classification problem. However, learning how the body temperature varies over the course of a day, in order to be able to predict temperature development, would be a classical regression problem. Finally, looking for recurring levels of temperatures, but without knowing anything about them, would require a clustering algorithm. The actual selection of a DM approach should primarily be based on the goal of the task. Apart from the previously mentioned examples, the end users' opinion and views should also be taken into account. They should be considered as part of the goal. Some users might be more interested in predictive capabilities (e.g. stockbrokers would certainly be interested in development or forecast of a share; for them, some sort of polynomial regression would be appropriate), while others would rather be able to actually understand the decision making process (e.g. medical professionals would certainly be more interested in being able to understand why a particular patient has been diagnosed with intestinal cancer; for them, a decision tree or other rule-based approach would be more appropriate).

Depending on the chosen DM algorithm, different kinds of data models and parameters might be more appropriate than others. Before continuing, one should verify that the current data matches the algorithms expectations and assumptions (e.g. data types, independence, correlation, Gaussian distribution, etc.). If this is not the case then one should return to the previous step and properly transform the data first. There can be a strong dependency between chosen DM algorithms and the way the data are modeled, which should always be kept in mind. Once the preconditions of the chosen algorithm are met then

Category	Approach(es)
Unsupervised [105, p. 87ff]	k-means, hill climbing, COWEB, BIRCH, CURE, DBSCAN, expectation maximization (EM)
Supervised [105, p. 68ff]	k-nearest neighbors (KNN), Bayes, ID3, C4.5, support vector machine (SVM), neural network (NN)
Regression / Prediction [105, p. 121]	simple linear regression, linear multiple regression, timeweaver, MA, exponential moving average (EMA), auto-correlation, auto-regression, autoregressive moving average (ARMA), ridge regression, lasso regression

Table 2.4: Lists DM algorithms of various categories. This is not intended to be a complete list, but rather an overview of commonly used techniques.

the actual DM can be performed. In other words, the search for (interesting) patterns may begin. The proper handling of preceding steps (i.e. with foresight and creativity) may substantially ease the work of DM algorithms [54].

Table 2.4 lists and references several algorithms of the aforementioned categories. It is intended to provide an overview of common DM algorithms and by no means meant to be an exhaustive list. One way or another, all of these algorithms require a way to determine the similarity of data samples or time series in order to recognize reoccurring patterns. Several options for similarity measures are shown in the following.

Similarity

Similarity measures play an important role in DM algorithms and should be kept in mind during the entire KDD process. A large set of these similarity measures has been proposed and evaluated. Here a few well-known approaches are highlighted. The interested reader may want to have a look at [105] for more details on the matter. The author points out and discusses an extensive collection on similarity measures. A subset of this collection is briefly described here.

- **Euclidean Distance:** This metric is defined for two time series A and B (with m observations and n attributes) as shown by Equation 2.15. The Euclidean distance $dist_{eu}$ can only be applied if both time series do not differ in terms of length, scale and baseline. In addition, gaps and noise will have an impact on the results. This also applies to variations in the time axis.

$$dist_{eu}(A, B, m, n) = \sum_{i=1}^m \sqrt{\sum_{j=1}^n (a_{i,j} - b_{i,j})^2} \quad (2.15)$$

- **Absolute Distance:** The absolute distance $dist_{ad}$ (a.k.a. Manhattan distance) is defined for time series A and B (with m observations and n attributes) as shown in Equation 2.16. Here similarities and dissimilarities are treated uniformly. In contrast to the Euclidean distance, dissimilarities

are not emphasized as much. However, this metric is also sensitive to the presence of noise, gaps and variation in time series.

$$dist_{ad}(A, B, m, n) = \sum_{i=1}^m \sum_{j=1}^n |a_{i,j} - b_{i,j}| \quad (2.16)$$

- **Maximum Distance:** This measure emphasizes the maximum difference between two time series A and B (with m observations and n attributes) as defined by Equation 2.17. The minimum distance can just as easily be utilized. However, both metric are sensitive to noise and require two time series of equal length, scale and baseline.

$$dist_{max}(A, B, m, n) = \max(a_{i,j} - b_{i,j}); i, j > 0, i \leq m, j \leq n \quad (2.17)$$

- **Dynamic Time Warping:** Keogh and Pazzani [81] define the distance $dist_{DTW}$ between two time series A and B (with n attributes and with length m_A and m_B , respectively) by a warping path W with $W = w_1, \dots, w_K; \max(m_A, m_B) \leq K < m_A + m_B - 1$ and $w = dist(A, B, i, j)$ being the distance between two data samples A_i and B_j . The general idea is to find a warping path W that minimizes the warping costs (see Equation 2.18). Here the time series do not need to be of equal length, but noise and outlier do affect this metric in a negative way. Overfitting can occur, thus distorting the actual distance between both time series.

$$dist_{DTW}(A, B, m_A, m_B, n) = \min \left\{ \frac{\sqrt{\sum_{k=1}^K w_k}}{K} \right\} \quad (2.18)$$

- **Longest Common Subsequence:** This metric emphasizes the longest common subsequence (LCSS) of two time series A and B (see Equation 2.19). Here both time series are required to be of equal scale and baseline. LCSS is not sensitive to gaps in both time series and can handle the presence of noise better than DTW. It is also less computational expensive and thus scales better.

$$\begin{aligned} A &= (42, 23, 12, 99, 21, 4, 65, 86)^T \\ B &= (17, 23, 4, 90, 87, 86, 20, 31)^T \\ lc_{ss} &= (23, 4, 86)^T \end{aligned} \quad (2.19)$$

As mentioned, many more similarity measures exist. E.g. slope-based similarity computation, edit distance, various probabilistic models or distance measures based on principle component analysis (PCA), eigenvalues and SVD can also be utilized. An overview is provided by Mitsa [105, p. 27ff].

2.3.5 Interpretation

The fifth and last step of the KDD process deals with interpretation and evaluation of discovered patterns. Afterwards, (newly) discovered knowledge can simply be documented for later use. However, in most cases the knowledge is going to be applied to another system or used to further refine the KDD process. As it has been previously mentioned, the KDD process (as a whole) is not as

strict and linear as suggested by the structure of this section. In fact, in many of the highlighted steps it is possible (in some cases even necessary) to go a step back and refine the data with newly gained findings. The interpretation step is no different in this regard. New knowledge can be used to optimize the selection of attributes, required transformations, etc.

Interpretation of newly discovered patterns may require additional expert knowledge of the particular domain. This further stretches the multidisciplinary nature of the KDD process. If necessary, one can return to any of the preceding steps in order to refine the results of the process. It may be helpful, to remove obvious, irrelevant or redundant patterns at this point. This step can also involve visualization of extracted patterns and models or visualization of the (original) data given the extracted patterns. For a better understanding of patterns, they can be translated into a more comprehensible form for end users.

It should always be a part of this step to evaluate the resulting patterns and knowledge. Another part of this step is that the newly obtained knowledge should be checked against previously discovered knowledge and other previous beliefs [54]. The purpose is to get an idea of how well the DM algorithm has worked and whether it produced the expected kind of results. If not, then an improper selection of data (see Section 2.3.1) or inappropriate feature extraction (see Section 2.3.3) are a probable cause. Depending on the chosen type of algorithm, error measures like precision and recall may give an indication regarding the achieved quality. Even though the DM step in the KDD process received quite a bit of attention, it does not mean that the other steps are less important. In fact, they should be considered equally, if not more, important [54].

2.4 Summary

This chapter discussed various topics related to PD and time series. Additionally, a set of definitions and notations were introduced (e.g. gold standard, false positive, positive predictive value, etc.), which are used throughout the thesis. The Table 2.1 might be especially handy for novices.

With respect to PD, the characteristics of motor symptoms were described and the overall progression of the disease is touched upon. Several non motor symptoms were also introduced and their implications for those suffering from PD were described. Well-known diagnosis methods and established treatment options of PD were enumerated. Furthermore, a few rating scales for quantification of progression in PD patients were highlighted.

As far as time series are concerned, the KDD process was described. Each step of the process was elaborated (e.g. preprocessing, (temporal) DM, interpretation, etc.) and simple examples (e.g. *mean*-binning, z-score normalization, Euclidean distance, etc.) were presented.

Chapter 3

Related Work Regarding Symptom Detection

In this chapter, the focal point is on computerized and automated indication of Parkinson’s Disease (PD) motor symptoms in time series. Thus, an answer to research question two is formulated. Additionally, various machine learning (ML) and data mining (DM) frameworks for time series analysis are discussed. Latter ones are viewed in the light of the requirements presented in Chapter 1. This chapter assumes prior background knowledge on both predominant topics (i.e. PD and ML). For a review on the matter please refer to Chapter 2.

3.1 Frameworks for Time Series Analysis

Literature has proposed a set of time series frameworks for online analysis [21,22,56,175]. Yet, many authors of symptom indication algorithms have settled for handcrafted solutions [34,140,142,143,166] as well as popular tools like Waikato Environment for Knowledge Analysis (WEKA) [62], Matlab and RapidMiner. Consequently, their implementations are not necessarily available to the general public and much less likely to be available in any particular framework (such as WEKA). In turn, a new algorithm can easily be published but fail to provide fundamental comparisons against an adequate subset of competitive state-of-the-art techniques. To facilitate a proper comparison among these algorithms, one would have to integrate a new algorithm in multiple frameworks (potentially having to implement the same algorithm in several programming languages) and one would also have to get the handcrafted solutions working. Furthermore, application of small datasets may make it tough to evaluate the algorithm’s “true” effectiveness as it is unclear how the particular algorithm performs on lengthy or large datasets.

Apart from the aforementioned frameworks, several other frameworks have also been proposed. However, they are not always actively maintained [88], specific to a particular domain [89], or do not provide desired flexibility and functionality [66]. Here, a selected subset of these tools and frameworks is discussed. Preference is given to those solutions that are no longer in a BETA stage (i.e. they have reached a somewhat stable stage), frameworks that are actively maintained, open-source and freely available. Furthermore, frameworks

providing the necessary flexibility in terms of transforming streamed data as well as data types are given preference. A few of their pros and contras are highlighted in the process.

3.1.1 Waikato Environment for Knowledge Analysis

WEKA is an ML / DM workbench [24,62,175] with an ever growing popularity in the research community. Its basic functionalities include but are not limited to: preprocessing (e.g. over 75 data filters and several data sources), classification algorithms (e.g. naive Bayes, C4.5, M5, bagging, boosting, etc.), clustering (e.g. k-means, expectation maximization (EM)-based mixture models, etc.) as well as a set of attribute selection and data visualization methods. The framework has been published under the GNU General Public License (GNU GPL). It has been under active development for about two decades (i.e. since 1993). Later versions of the WEKA workbench switched over to JAVA (first release in 1997). Prior releases resembled a collection of C programs, shell scripts, PROLOG programs, etc. [24,62]. Today's version is widely adopted and used by both researchers and practitioners. Researchers benefit from the availability of a wide range of algorithms to compare against while practitioners can apply these algorithms to their target domain.

The software can easily be extended (e.g. adding classification algorithms, clustering approaches, etc.). WEKA provides several graphical user interfaces, each focusing on a different aspect (i.e. exploration, experiment design, etc.), and a command line interface. These can be utilized to accomplish a wide variety of tasks. It comes with a large amount of documentation (i.e. application programming interface (API) documentation, references to original publications of algorithms, book [175], manuals and publications [20,24,62,70,132]) and a selection of readily available algorithms even from the pre-WEKA era.

Even though WEKA can easily be extended, not everyone can add to the software releases. A strictly limited circle of developers has the privileges to perform changes [24]. However, from a code quality and coherence point of view, this may very well have been a wise decision. Additionally, WEKA has been designed for "traditional" DM and ML applications, thus it does not natively support a stream-based analysis. Furthermore, the time domain and sequence of data samples are not implicitly considered. It is meant to filter data, extract features and apply learning algorithms on a finite set of data samples. The latter are represented by predefined data types. Despite the generally good quality of the code, the class hierarchy and interfaces appear cluttered up. They seem historically grown and their purpose is not always clear. WEKA uses individual unrelated interfaces for each kind of algorithm (i.e. filter, classifier, clustering, etc.) rather than employing a common data processing entity.

3.1.2 Massive Online Analysis

Massive online analysis (MOA) is an open-source software framework that includes a set of online and offline algorithms for clustering as well as classification of evolving data streams. MOA has been published under the GNU GPL. It has been created as a tool for researchers and practitioners, enabling usage of a broad range of readily available algorithms. Bifet et al. [21] have created MOA as an expandable platform where researchers can contribute their algorithms

and test against others. Practitioners can utilize MOA’s capabilities to evaluate real world problems on a set of algorithms and choose the best one.

The MOA framework is related to WEKA. Algorithms in both frameworks can be used interchangeably. Also similarly to WEKA, this framework also provides a fundamental benchmark platform for researchers and their algorithms. It has been noted by Bifet et al. [21] that numerous publications of new algorithms do not provide sufficient benchmarks against competitive state-of-the-art approaches. Furthermore, the capacity to function on large (or even infinite) datasets is not always adequately demonstrated nor is the potential application under varying memory limits shown. Here, MOA can be utilized to create (and re-use) benchmark settings. Thus, researchers and practitioners are enabled to create repeatable experiments as well as reproducible results. This reduces the effort to properly evaluate and compare newly developed algorithms.

The general software architecture allows MOA to be easily extended in three ways: (1) data sources / generators, (2) learning algorithms and (3) evaluation methods. In principle, these points also describe the overall workflow within the MOA framework. First a source is selected (i.e. file, artificial data generator, etc.), then secondly one of the learning algorithms is chosen and configured. Lastly, an evaluation method is chosen for analysis of a given scenario. Thus, a rather simple and sequential workflow is implemented in the framework.

Additionally, MOA supports the notion of constrained computing. It can be configured with varying memory limits to simulate different environments. E.g. sensor networks (memory $\leq 100\text{KB}$), mobile devices (memory $\leq 32\text{MB}$) and dedicated servers (memory $\approx 400\text{MB}$) can be addressed. This enables more realistic tests and allows evaluation of algorithm efficiency for varying applications and environments.

The nature of the MOA framework does only allow a fairly strict workflow. Even though this represents a common scenario and it does make sense in cases where processed data are at hand. However, real-world cases do typically require at least some preprocessing. This functionality would come handy in rapid prototyping scenarios. Of course, one could argue that such preprocessing functionality should not be part of a framework like MOA, and one might even be right. However, this kind of functionality greatly enhances usability and is certainly helpful and eases prototyping efforts. Additionally, more complex workflows are desirable (as opposed to a simple sequential processing of a data source, learning algorithm and evaluation method). E.g. scenarios in which the information flow serves multiple learning algorithms in parallel cannot be modeled with MOA. The handling of streams is limited to an internal stream-like interface rather than input and output stream native to JAVA. Furthermore, the data format handling and stream handling are intertwined thus reducing flexibility. Regression and frequent pattern mining are also not supported.

3.1.3 Unstructured Information Management Architecture

The Unstructured Information Management Architecture (UIMA) project [168] is an open-source software, published under the terms of the Apache License, version 2.0 (ALv2). Its ultimate goal is to aid in the transformation of unstructured information to structured information. UIMA can be used to analyze large amounts of unstructured data with the goal of identifying previously unknown knowledge. One of the main motivations behind the project is the idea to

foster reusability (e.g. reuse of analysis components) and thus reducing overall development efforts. Its architecture was designed to be easily expandable (e.g. adding new components for analysis or annotation). These components can be chained together in order to form complex analysis and processing scenarios.

An implementation of the UIMA architecture is available for the JAVA and C++ programming languages. The main concepts of the architecture are analysis engines (AEs), annotators, common analysis structure (CAS) and subject of analysis (SOFA). However, UIMA is an empty framework on its own. As such it does not include any particular data processing, ML or DM modules. It thrives through participation of developers and enthusiasts. It is primarily employed in search and annotation applications. The analysis occurs through a chain of so called analysis engines, the result of which assigns semantics to elements in the unstructured data. The architecture allows a stream-based processing based on a CAS data type.

UIMA includes support for distributed deployments in different middleware environments, support for multiple-modalities, support for efficient integration across programming languages, component discovery and composition and many more. However, its main purpose remains: turn unstructured information into structured information. Afterward, the structured information can be used by ML or DM algorithms. It supports the analysis of multi-modal contents (e.g. text, audio and video).

3.1.4 Framework: streams

The streams framework is an open-source software framework for stream-based data processing [22] and it is published under the terms of the GNU Affero General Public License (GNU AGPL). It provides the means for an iterative and modularized processing of data samples and enables rapid prototyping of compute graphs (i.e. a graph of interconnected modules that accept data, process or compute data and forward it to the next module in the graph). The framework utilizes XML-based definitions for processes and graphs. Thus, the transfer and exchange of designed experiments among users is simplified which in turn facilitates reproducibility (i.e. by other researchers and practitioners). In this regard, the framework abstracts from implementation details and allows focusing on the actual design task.

Graphs in the streams framework can be translated to topologies of the scalable tool for resource management (STORM) system [56, 159]. Nonetheless, these graphs can also be run within a JAVA virtual machine (VM). Furthermore, the MOA library has been integrated to add support for various ML approaches. The streams framework is intended to provide a high level definition of stream processes. Thus enabling the integration of existing ML and artificial intelligence (AI) libraries such as WEKA or MOA. In doing so, some of the original drawbacks from the integrated libraries can be nullified. The framework provides an “environment for implementing stream processes by combining implementations of existing online algorithms, online feature extraction methods and other preprocessing elements” [22, p. 8].

On the downside, the streams framework treats data sources, data sinks and processing modules in a different way. There should be no need for such distinctions. Furthermore, computation graphs of the streams framework only allow sequential and tree-like structures to be formed, but not fully interconnected

graphs. Additionally, throughput of data samples can only be throttled as opposed to varied in both directions. Also, only restricted data types can be processed (i.e. existing data types cannot necessarily be easily reused). The API's source code is only partially documented. However, a technical report, detailing various configuration and extensibility options, is included in the framework.

3.1.5 Others

Several other frameworks are in existence, a few of them are briefly highlighted as follows.

- **IPUS:** It is a domain-independent framework which is based around formal signal processing models. In contrast to stable environments where a fixed set (and configuration) of signal processing algorithms (SPAs) may be appropriate and sufficient, Integrated Processing and Understanding of Signals (IPUS) can also be applied in environments which can be characterized by “variable signal-to-noise ratios, unpredictable source behavior and the simultaneous occurrence of objects whose signal signatures can mask or otherwise distort each other” [88].

One of the key aspects of the IPUS architecture is its responsive design loop. The front-end signal processing can be dynamically changed in accordance with changes in the environment. Parameterization and SPA can both be adjusted in response to such changes.

IPUS is capable of processing signal data in a block-wise manner. It uses an iterative process to adapt SPAs, their configuration and interpretation of signals. Lesser et al. [88] state that the framework can (potentially) be used everywhere where a “rich underlying theory” is present.

The framework does not automatically adjust parameters of SPAs, instead it reacts to changes in the environment in a predefined manner. The adjustments need to be modeled in the form of a domain-specific formal processing theory. This is then used to initialize IPUS. The learning of models and statistical relationships among components are not covered.

- **SPARK:** The SPARK [156] project is an open-source cluster computing system, written in the Scala programming language and published under the terms of the ALv2. The general idea of SPARK is a system that supports a wide range of large-scale data-intensive applications. Similarly to MapReduce [49], Dryad [72] and Map-Reduce-Merge [178], SPARK allows parallel computing on commodity clusters with automatic fault tolerance as well as the “reuse [of] a working set of data across multiple parallel operations” [180]. The latter type of application is not efficiently supported in the previously mentioned systems. Iterative algorithms (e.g. many ML algorithms) and interactive DM tools belong to this class.

SPARK utilizes resilient distributed datasets (RDDs) to support such applications with similar properties to MapReduce (i.e. scalability and fault tolerance). RDDs represent a collection of objects that are partitioned across multiple machines. This collection is read-only and partitions can be reconstructed if one is lost. These components are accessed by a driver program, which implements the high-level control flow of user applications (and can be used to automatically reconstruct lost partitions). RDDs are particularly well suited for “batch applications that apply the same

operation to all elements of a dataset” [179]. RDDs are less effective for “applications that make asynchronous fine-grained updates to shared state” [179].

- **STORM:** The STORM project [56, 159] resembles a lightweight, flexible and scalable system for management of resources on clusters. The architecture is designed to meet two primary goals: (1) to provide resource management (RM) mechanisms and (2) to support a variety of job scheduling algorithms. These goals have been realized by implementing a set of loosely coupled daemons, which communicate through low-level network messages. STORM incorporates several scheduling algorithms such as gang-scheduling or buffered co-scheduling [56]. Three kinds of daemons are utilized by STORM: (1) machine manager (MM), (2) node manager (NM) and (3) program launcher (PL). They are used to successfully distribute and execute an application. The MM performs global scheduling and resources accounting. Local resource monitoring and scheduling is done by the NM. The PL takes care of the actual application.

The RM tool is designed around a set of five critical points: flexibility, usability, portability, scalability and performance. It takes full advantage of the underlying network hardware [56] and runs in user mode (as opposed to kernel mode). The latter point facilitates experimentation but comes at the cost of being dependent on the operating system (OS)’s scheduler (thus, it is susceptible to its variability).

STORM is an open-source framework and freely available [159]. It is published under the terms of the ALv2. However, it is more an RM tool rather than a data processing and analysis framework. It provides clusters the ability to quickly launch applications, maximize resource utilization and fast responses to user input. Nonetheless, the managed applications may still perform data processing tasks even though the framework does not. It is a relevant work in terms of RM and parallelism but it does not resemble a data processing framework.

3.1.6 Summary

This section outlined several relevant and related frameworks to time series processing. A brief overview to WEKA, MOA, UIMA and streams was given. Some pros and contras were highlighted in the process. Table 3.1 summarizes criteria for comparing the previously outlined frameworks.

These criteria are based on the requirements that were imposed on the software framework being developed in this thesis (see Section 1.2). Consequently, they were also used as a basis for comparison to other frameworks. In the eyes of the author, a framework that fulfills all or at least most of these criteria can be expected to gain popularity among researchers and time series enthusiasts. However, such a framework could not be found by the author.

Most criteria were chosen because their fulfillment is expected to ease development and evaluation within the framework. An open-source license helps in terms of transparency and does not necessarily exclude commercial use but may actually foster it (depending on the selected license). Practitioners would benefit from flexibility, scalability, support for distribution and definitely from included algorithms. On the other hand researchers and developers would benefit from reusability of components (within the framework) as well as extensibility. The

criteria “stream-based” and “iterative” are inherent to a framework related to time series that focuses on online processing.

Most of the presented frameworks in Table 3.1 were published under the terms and conditions of a GNU GPL (or a derivative thereof). All of these frameworks are freely available, open-source and written in Java. Furthermore, all of them utilize reusable components and they are extensible in (at least) one way. From the selected 14 criteria, the average framework fulfills 9 criteria (of which 2 are partially fulfilled). None of the frameworks fulfill all criteria and most of them have their weaknesses in providing preprocessing, classification and related capabilities.

	WEKA	MOA	UIMA	streams
Version	3.6.10	2013.11	2.4.2	0.9.15
License	GNU GPL	GNU GPL	ALv2	GNU AGPL
Programming Language(s)	JAVA	JAVA	JAVA	JAVA
Open-Source	✓	✓	✓	✓
Free	✓	✓	✓	✓
Stream-Based	X	(✓)	(X)	✓
Iterative	(X)	✓	(✓)	✓
Scalability	X	✓	✓	✓
Flexibility	(✓)	X	(✓)	(✓)
Reusability	✓	✓	✓	✓
Extensibility	(✓)	(✓)	(✓)	(✓)
Support for Distribution	X	X	✓	✓
Preprocessing	✓	X	X	X
Classification	✓	✓	X	X
Clustering	✓	✓	X	X
Regression	✓	X	X	X
Frequent pattern mining	X	X	X	X

Table 3.1: Highlights the applicability of mentioned frameworks to the comparison criteria. Those points that are enclosed in brackets are partially fulfilled (✓) and X indicate whether the criteria closer is to being fulfilled or not fulfilled).

3.2 Identifying Parkinson’s Disease and Its Symptoms

Much research has been published with a focus on biological, chemical and genetic aspects of PD. Over the last two decades, an increasing number of publications originate from fields like computer science or AI, focusing on signifying motor symptoms in people with Parkinson’s. This information can then be used to monitor symptom progression, (partially) evaluate treatment effectiveness and could ultimately be used to personalize (and improve) medication regime. Some of these publications are dedicated to detecting single motor symptoms [13, 33, 34, 37, 51, 98, 113, 136, 161, 171] and others to the detection of multiple symptoms [36, 122, 140, 143]. These publications reveal a great number of techniques for automatically indicating the presence of PD motor symptoms (e.g. neural networks (NNs) [13, 18, 34, 36, 37, 51, 78, 140], hidden markov models (HMMs) [136] and support vector machines (SVMs) [18, 34, 122]). Depending on symptom and utilized sensors, various features have been proposed and used in this context (e.g. entropy [18, 34, 110, 122], spectral or fractal features [13, 31, 32, 77, 111, 115, 143, 155, 162, 171]). In the course of this section a strong focus will be on common motor symptoms that are experienced by PD patients. This restriction is due to the otherwise extensive list of motor and non motor symptoms as shown in Table A.1 and Table A.2. Preference is given to those publications that do not focus on a single symptom (as opposed to multiple symptoms) nor make use of synthetic datasets (as opposed to data recorded from sensors on the subject’s body) but rather use unconstrained and unscripted activities of daily living (ADL).

It should be kept in mind that there are many other publications with a focus on PD symptom indication or their severity, but do not make use of body-mounted sensors or otherwise do not resemble the previously elaborated criteria. Despite this reservation, a few selected publications that do not fit these criteria are presented nonetheless.

3.2.1 Tremor at Rest

In an early study by Salarian et al. [142], they were able to achieve a specificity of 98% and sensitivity of 76.6% in detecting resting tremor on a dataset with ten patients and ten control subjects. In total, close to twenty hours of data were captured by the authors. Two tri-axial gyroscopes (i.e. one on each wrist) were used to record data while participants performed a set of scripted everyday activities. Spectral analysis was used to filter interesting regions within the frequency range specific to resting tremor (i.e. $3.5Hz - 7.5Hz$). In a later study [143] based on the same dataset, the data stream was divided into chunks with a length of three seconds to which the Burg method [30] was applied (i.e. a method for estimating power density spectra). Additionally a meta-analysis was introduced to remove isolated segments that were classified to exhibit tremor or tremor-like behavior (e.g. a single segment with tremor surrounded by non-tremor segments). This increased the sensitivity to 99.5% but lowered the specificity to 94.2%.

In a paper by Zwartjes et al. [182], inertial sensor data (i.e. acceleration and angular velocity) were gathered from six patients and seven control subjects.

Zwartjes et al. had captured approximately 1.5 – 2 hours of data while participants were performing a set of scripted activities in laboratory conditions. A multi-staged algorithm is utilized to indicate regions of tremor. At first some preprocessing is applied to the raw data, which is then used to classify the subject’s activity and posture. This pre-classification is used to highlight regions of interest where tremor is more noticeable (e.g. arms are hanging still while standing). If activity or posture were suitable for detection of tremor (and its severity) then those portions of the data stream are divided into segments of three seconds length with $\frac{2}{3}$ overlap. For each segment, the Fourier transform (FT) is used to identify tremor specific frequencies and thus tremor episodes as well. In the algorithm’s last stage, a meta-analysis removes isolated segments of tremor (very similar to the process that was utilized by Salarian et al. in [143]). Zwartjes et al. achieved an accuracy of about 84.7%. However, in comparison to studies by Salarian et al. [142,143], the recorded activities were less constrained.

Rigas et al. [136] achieved an accuracy of 87% in detecting tremor in an accelerometer based dataset with twenty-three participants (i.e. ten patients and thirteen control subjects). All participants performed daily activities in laboratory conditions. The data stream is divided into three second windows with 50% overlap. Having applied standard filtering and analysis techniques (i.e. finite response (FIR) filters, fast Fourier transform (FFT), etc.) an HMM is utilized to detect tremor episodes. This is different from most algorithms for tremor indication. More common approaches rely on spectral features alone [13, 31, 32, 77, 111, 115, 143, 155, 162, 171] while others classify based on NNs [13, 18, 34, 36, 37, 51, 78, 140] or SVMs [18, 34, 122]. Rigas et al. state that HMMs are suitable for tremor indication because “tremor presents time-dependency” [136]. They consider HMMs as a time sensitive extension of the naive Bayes classifier.

Cole et al. [36] were able to detect tremor with a sensitivity of 93% and specificity of 95% in unconstrained and unscripted activities. The dataset contained about 48 hours of acceleration and electromyograph (EMG) measurements from twelve participants (i.e. eight patients and four control subjects). Here a dynamic neural network (DNN) is used in combination with a set of FIR filters to detect tremor. It is stated by the authors that DNNs [172] were utilized because they are more capable of learning and classifying time-dependent classes (e.g. tremor) when compared to regular and static neural networks. Cole et al. divided the data stream into segments of two seconds length for feature extraction. The features were simply passed to the DNN, where artificial neurons did their work. However, the neurons’ outputs were not simply forwarded to the next layer of neurons. Instead, each neuron had an FIR filter attached to it which transformed the output before it was passed to subsequent neurons. Their results were mainly dependent on the choice of training data. Here a handcrafted representative subset of data was chosen.

A dataset with nineteen patients and four control subjects was used by Roy et al. [140] to signify tremor. They achieved a sensitivity of 91.2% and a specificity of 93.4% in EMG and acceleration data. The participants were performing unscripted and unconstrained activities in a home-like environment for several hours. Here the data stream is also divided into two second windows and a combination of DNNs with FIR filters is fed with various features that were extracted from these segments.

Niazmand et al. [114] collected data from accelerometers integrated into a

pullover. Ten patients and two healthy control subjects performed standardized PD motor tasks. An average sensitivity of 80% in indicating postural tremor and resting tremor and a specificity of 98.5% was achieved by the authors. Their algorithm first determines the relative acceleration among the sensors and then determines the movement frequency. This is done because sensors are not fixed on the patient’s body but rather in a garment which position can change depending on executed movements. The raw data are simply filtered, normalized and a noise removal method is applied. For determining the movement frequency, a combination of thresholds and peak counting is utilized.

3.2.2 Bradykinesia

In a paper by Cancela et al. [34], the authors present a motor symptom monitoring and management system. Their work originates from a European research project called PERFORM (see Section 3.3 “Parkinson’s Disease in Research Projects”). Here a set of classification algorithms (e.g. SVM, k-nearest neighbors (KNN), NN, decision tree (DT), etc.) was evaluated. The highest accuracy of 86% was achieved by the SVM. The corresponding dataset consists of acceleration data from twenty patients performing a set of ADL (within the limits of a scripted protocol). A standard analysis procedure is used by Cancela et al. At first a Butterworth filter is applied to raw sensor data then the data stream is epoched in five second segments with a 50% overlap. A set of features (i.e. sample entropy, root mean square (RMS), cross correlation, etc.) is calculated for each segment and passed to the classification algorithms. Here, the algorithms classify presence and severity of bradykinesia. Interestingly, the severity is not derived from standard motor tasks (see Section 2.2.4) but instead from ADL.

Cancela was also involved in a publication by Pastorino et al. [120]. Here a slightly modified version of Cancela’s algorithm is utilized (as in [34]). A dataset from twenty-four patients performing unconstrained and unscripted activities at their home was gathered for a week. Twice a day, a clinician came to visit the patient and performed a short session, which was later used to test the previously developed algorithm. Pastorino et al. show that an additional meta-analysis can improve classification results. Instead of using the generated outputs from the SVM directly, they can be further smoothed to ignore impossible and unrealistic scenarios. Using a patient independent algorithm an accuracy of $68.3\% \pm 8.9\%$ was achieved and a $74.4\% \pm 14.9\%$ accuracy was achieved with the additional meta-analysis. They indicate that a patient specific training of the algorithms would likely lead to improved results.

Salarian et al. were not only involved in detecting tremor in time series data, they were also using gyroscopes on the wrists to indicate the presence of bradykinesia. In [142], ten patients and ten healthy control subjects participated in the collection of twenty hours of data. All participants were performing scripted ADL. Salarian et al. showed that the features rotation of hand (R_H) and mobility of hand (M_H) correlate well with the clinician’s ground truth ($r = -0.84$ and $r = -0.83$ for M_H and R_H respectively and $p < 0.00001$). Several years later, Salarian et al. were able to reproduce their results in [143]. However, window sizes of five minutes and above were used in latter publication. Even though their work does not produce results in real-time, it does give hope that not many sensors are required for a decent accuracy in bradykinesia detection.

The authors Zwartjes et al. [182] were able to identify bradykinesia related parameters that correlate well with the patient’s unified Parkinson’s Disease rating scale (UPDRS) scores. Here a dataset based on accelerometers and gyroscopes from six patients and seven healthy control subjects was analyzed. All subjects performed a mixture of standardized motor tasks and ADLs in a random predefined order. An activity and posture classifier is used to identify a set of elementary activities (i.e. walking, standing up) and postures (i.e. standing and sitting). For upper extremities, an average arm acceleration is calculated while various gait-related features (i.e. step length, step velocity, etc.) are determined from a tri-axial gyroscope and a tri-axial accelerometer that are placed on the foot. These features provide the basis for bradykinesia (slowness of movement) and hypokinesia (poverty of movement) quantification. The authors’ results indicate that a significant correlation is present in almost all bradykinesia-related parameters while “none of the hypokinesia-related parameters were significantly correlated” [182].

3.2.3 Akinesia / Freezing of Gait

In [51], Djurić-Jovičić et al. employed a neural network and a simple thresholding technique to classify walking patterns in PD patients. A set of six inertial measurement units, each containing a tri-axial accelerometer and a tri-axial gyroscope, were attached to the subjects’ legs (i.e. thigh and shin) as well as their feet. The kinematics of four patients (as they were following a predetermined path) were gathered, annotated and used to train a neural network. In total, about 30 minutes of data were collected. The path itself included several (potential) hurdles which have been designed to provoke freezing of gait (FoG) (e.g. start hesitation, destination hesitation, narrow path or turn hesitation, etc.). A combination of heuristically determined thresholds and an NN were utilized to differentiate between “normal” (i.e. standing and regular steps) and pathological (i.e. festination, akinesia, shuffling and small steps) walking patterns. The authors [51] achieved an error rate as high as 16% due to the choice of thresholds (i.e. thresholds were independent of patients, etc.). It should be taken into account that the algorithm was working in real-time (i.e. about 0.5 seconds delay).

A similar technique was developed by Cole et al. [37]. However, instead of a regular (static) neural network a dynamic neural network [172] was utilized. Their indicator algorithm showed a sensitivity of 82.9% and specificity of 97.3% in a dataset containing unconstrained and unscripted activities. Ten patients and two healthy control subjects contributed and helped to gather about two hours of data from several accelerometers (i.e. forearm, thigh and shin) and an EMG sensor (i.e. shin). The authors employed a multi-staged algorithm [37]. In the first stage, a simple linear classifier determines whether the subject is in an upright position (i.e. standing and not sitting or lying). If this is the case then a DNN determines freezing episodes. The idea was to identify periods in the data stream where FoG episodes are more likely to be apparent (both visually and in data stream). In contrast to a static NN, Cole et al. [37] have decided to use a DNN because they are able to better capture time-varying weights that are present in FoG episodes.

An accelerometer based smart garment called MiMed-Pants [112] has been used by Niazmand et al. [115] to extract and analyze gait-related features. In

this case, the measurement device has been successfully integrated in an item that is “suitable for daily use” [115]. The pair of pants can be washed like a regular textile. A sensitivity of 88.3% and specificity of 85.3% has been achieved with this setup. Five accelerometers (i.e. each shin, each thigh and belly button) provided kinematics on six patients while they were performing standard activities [181] (i.e. walking course, including narrow spaces, gait initialization and reaching destination, etc.). In total about one hour of data was collected by Niazmand et al. No use of advanced artificial intelligence methods was made in [115], but instead a linear classification was applied to features that were extracted from the sensors. The algorithms provided feedback with a delay of about two seconds.

In 2009, Bächlin et al. published their work on a wearable and context-aware system for real-time detection of FoG events [32]. The system provides acoustic feedback within a two second window. A set of accelerometers and gyroscopes was utilized (e.g. on shank, thigh and waist). Over eight hours of data were gathered from ten patients performing ADL, as well as walking in a straight line and a random walk. Bächlin et al. claim to have built the first context-aware and wearable system to assist PD patients in detecting FoG events. An overall sensitivity of 73.1% and a 81.6% specificity were achieved with just two features (i.e. energy in $0.5 - 3Hz$ and energy in $3 - 8Hz$ band) and two thresholds. The features are used to calculate the freezing index (FI) whenever the energy exceeds a lower threshold. FoG is detected when the FI is above another threshold. The results were mainly due to different walking styles (that were used by the subjects in their dataset), choice of features and use of patient independent thresholding. They state that a personalized training and choice of threshold might have produced better results.

In [157], Stamatakis et al. were able to show differences in walking patterns between a PD patient and a healthy control subject. The authors identified a set of features (e.g. variability in stride time, variability in stance phase and others) that may be used for differentiating between PD patients and healthy control subjects as well as for detecting FoG events and their duration. Even though no results in terms of accuracy or significance were presented in [157], their proposed features may prove to be beneficial.

3.2.4 Drug-Induced Dyskinesia

Keijsers et al. [78] were able to achieve an accuracy of 96.6% in detecting dyskinesia. Thirteen participants were enrolled in their study. Each subject contributed about 2.5 hours of acceleration data while they were performing a set of scripted activities (approximately 35) in a controlled environment. In total six tri-axial acceleration sensors were attached to the subject’s body (i.e. one on each thigh, one on each shoulder, one on trunk and one on wrist) during their recording session. The algorithm, used by Keijsers et al., classified fifteen minute segments using a regular neural network. The NN was fed with features such as power and velocity in certain frequency bands (e.g. $1 - 3Hz$, $> 3Hz$) and other features. The output of the NN indicated the presence (or absence) of dyskinesia within the segment. Several segment sizes were empirically evaluated (e.g. fifteen and one minute segments). The best accuracy was achieved with the fifteen minutes segments (i.e. 96.6%). However when using one minute segments the accuracy drops to about 80% on the same dataset.

In contrast to Keijsers et al. [78], Tsipouras et al. [166] were able to achieve similar results but on smaller segments. While Keijsers et al. [78] used fifteen minute segments, here two second intervals with 75% overlap are utilized. Tsipouras et al. state to have achieved a 93.7% accuracy using their dataset. This contains inertial sensor data (i.e. acceleration and angular rate) of four patients and six control subjects. All participants were performing a set of scripted activities. In total two gyroscopes (i.e. one on trunk and one on waist) and six accelerometers (i.e. one next to each gyroscope, one on each arm and one on each leg) were used during recording sessions. Features included entropy, mean and standard deviation for each sensor and every 2-second window (with 1.5 seconds overlap). Here five classification methods were evaluated (i.e. naive Bayes, KNN, fuzzy lattice reasoning, DTs and random forests (RFs)). The RFs performed best with 93.7%. C4.5 is close behind with about 93.5% while all other remaining classification algorithms achieved an accuracy around 85%.

Cole et al. [36] used a DNN to better capture the time-based variables. The authors were able to achieve a 91% sensitivity and a 93% specificity in detecting dyskinesia. Here a similar procedure to their work in detecting tremor [36] and bradykinesia [37] was utilized. Their dataset contained several hours of acceleration data and EMG measurements from eight patients as well as four control subjects. Participants were performing unscripted and unconstrained activities during their recording session. A set of features is extracted from a two second sliding window (e.g. dominant frequency and energy in given frequency bands) and fed to the DNN. Additionally, outputs of each artificial neuron (i.e. node within the neural network) are filtered using a five point FIR filter.

Similarly Roy et al. [140] combine DNNs with a rule-based reasoning method. They were able to achieve a sensitivity of 90% and specificity of 93.4% in a dataset containing acceleration data and EMG measurements from nineteen patients and four control subjects. One hybrid sensor (containing a tri-axial accelerometer and an EMG sensor) was located on each arm and leg. All participants were performing unconstrained and unscripted activities in a home-like environment. In total about 30 hours of data were gathered and used by Roy et al. [140]. Again, the extracted features originate from two second segments (e.g. energy in certain frequency bands, dominant frequency, etc.). Their algorithm uses those features to feed two DNNs (i.e. one for mobility states and one for motor states). These DNNs provide preliminary results on patient's mobility state (i.e. sitting walking, standing, etc.) and motor symptoms. They are used in combination with a framework called IPUS [88], which in turn activates different DNNs to maximize symptom recognition rates (e.g. based on the fact that the subject is walking, sitting, etc.).

Patel et al. [124] utilized clustering techniques (and EM) to distinguish various levels of severity in PD patients while they were performing standardized motor tasks. A similar approach was used by Sherrill et al. [152]. Here six patients provided acceleration data to which clustering was applied in order to detect dyskinesia.

3.2.5 Dysarthria and Dysphagia

In [134], Revett et al. employ a rough sets approach for distinguishing healthy subjects and people with PD based on vocal data. Their dataset is based on

thirty-one participants (i.e. twenty-three patients and eight healthy controls) performing a phonation task. Little et al. [91] originally constructed this dataset and donated it to University of California Irvine (UCI) Machine Learning Repository [15]. On average, each participant performed six phonations of the vowel [a], thus resulting in close to 200 samples. The authors document a set of twenty-three features (including spectral features, shimmer, jitter, presence of PD, etc.). They report to have achieved a 100% accuracy when using all available features in their rough sets approach. This holds if the classification category is binary (i.e. healthy subject or PD patient). In this case, several hundred rules are generated to identify a data sample’s category. When trying to reduce the number of rules, the accuracy drops but stays well above 90% with about 100 rules. However, the authors did not attempt to perform classification based on the patient’s duration of the disease, severity or UPDRS scores.

In a publication by Bakar et al. [17], they present a speech-based assessment tool for identifying PD. Here the same dataset (as originally constructed by Little et al. [91]) has been utilized. The authors performed several tests in which they compared testing accuracy, training accuracy, average mean squared error (MSE) as well as average number of iterations of two learning algorithms for NNs (Levenberg-Marquardt (LM) and Scaled Conjugate Gradient (SCG)). Their results indicate that LM outperforms the SCG algorithm. Generally speaking, the LM-approach resulted in better testing and training accuracy as well as a lower MSE while SCG performed better in terms of “number of iterations”. The best result was achieved with a 97.9% accuracy in training and 93.0% accuracy in testing.

Asgari and Shafran [14] utilized a similar set of features for classifying speech and phonation data. They performed a prediction of UPDRS motor scores based on these features. In more than one hundred recording sessions, twenty-one control subjects and sixty-one patients were asked to perform three tasks: sustained phonation (i.e. phonation of vowel [a]), diadochokinetic test (i.e. repetition of syllables [pa], [ta] and [ka]) and a reading task. The actual recordings were done with a dedicated device that was designed to be an at-home testing tool. The dataset itself was analyzed with 100 frames per second using Hamming windows (each of 25ms length). A rather large set of features is extracted for both voiced and unvoiced segments. About 15K features were generated based on their recordings of phonation and speech data (including pitch, frequency, harmony, etc.). an SVM was employed to translate these features into UPDRS motor scores. Depending on the used features (or subset of features), a mean absolute error between 6.1 and 5.7 (UPDRS) points was achieved (total points of motor sub-scale: 108).

In [103], Mekyska, Rektorova and Smekal evaluate a set of features for automatic analysis of speech disorders in PD. The authors provide an extensive summary on various speech-related parameters and highlight a few common problems in automatic speech analysis. Their dataset is composed of forty-two male control subjects and twelve male PD patients. Each participant was asked to pronounce all vowels (i.e. [a], [e], [i], [o], [u]) once in a natural speed and once slowly. The inter-intra class distance ratio (IICDR) method and minimum redundancy maximum relevance (mRMR) method were used by Mekyska et al. in order to sort out the top 20 parameters for each method (after having started with 510 parameters in total). In a second step, the Jarque-Bera test was utilized to see which features show a normal probability distribution. Those with a

normal distribution were used in a multi-factor analysis of variance (ANOVA). Their results show up to three features (i.e. “mean $B - F_1$ ” for $p \leq 5\%$ as well as “mean F_0 ” and “mean NHR” for $p \leq 10\%$) that can be used to distinguish healthy control subjects from those afflicted with PD. Mekyska et al. add that there are also several parameters, which do not show a coherent tendency in published literature. The authors point out several papers in which a particular feature has been shown to be significant, non-significant and indifferent in terms of separating patients from healthy subjects. However, they also comment that these conflicting publications usually used rather small datasets. Thus, they are more prone to random variations and clusters.

Xiuming et al. [177] describe a diagnostic approach to PD based on principle component analysis (PCA) and Sugeno integral. The authors employ a dataset by Little et al. [91] from UCI Machine Learning Repository [15]. Thus, data of thirty-one participants (i.e. eight patients and twenty-three healthy control subjects) was analyzed. Xiuming et al. show five principle components that account for 86.5% of the information within the signal. In order to propose a diagnosis, the Sugeno measure and Sugeno integral are then determined for the top five most relevant features. The authors report a classification accuracy of 81.0%.

3.2.6 Others

In 2000, Hamilton et al. [63] published their work on outcome prediction in pallidotomy in PD patients (see *ablation* in Section 2.2.3). It was their goal to build a reliable tool, which estimates an operation’s outcome based on intra-operational recordings of neural activity. A standard NN was employed and trained with a set of features (e.g. signal power, entropy or fractal dimensions). This system could provide supplementary data and aid surgeons in minimizing risks (e.g. blindness, difficulties in speaking or swallowing, etc.) and maximizing effectiveness. Their results indicate that all evaluated NN performed similarly in terms of overall outcome prediction. Hamilton and colleagues found that their NNs handled exceptional cases well.

In terms of diagnosis Kupryjanow et al. [86] came up with an alternative measurement technique for determining UPDRS sub-scores related to motor tests (i.e. finger tapping and rapid alternating movement of hands). Instead of relying on arguably subjective assessments from neurologists, they present a device called Virtual-Touchpad (VTP). Here a webcam is used to capture movements of hands and translate them into machine-readable features. In comparison to other methods, this approach does not require equipment to be attached to the patient (or mounted on the patient). an SVM recognizes hand gestures and postures. The succession of those postures is used to extract the mentioned features and determine UPDRS scores. The authors did not perform a user study in [86].

Cunningham et al. [43] presented their work on a computerized assessment tool. The work is intended to identify movement difficulties found in people with PD and similar movement disorders. Here the participants’ ability to point and click on targets on the computer screen is compared among those afflicted with PD and healthy control subjects. A benefit of this approach is that patients are not expected to wear “unusual” or specialized hardware. However, on the other hand, it requires the patient to sit in front of a computer

and cannot be mobile (as it would considerably alter their ability to point and click). Their results show a difference in control subjects and PD patients. The control group was generally more accurate (i.e. clicked closer to the target's center and made less accidental clicks) and faster (i.e. required less time to click once the target has been reached). This holds for both computer literates and computer illiterates. Those being computer illiterates and suffering from PD showed a higher variance in terms of accuracy of clicking the target center. In [42], Cunningham et al. present their tool's abilities in indicating akinesia, bradykinesia, dyskinesia, rigidity and tremor.

In a preceding publication by Cunningham and colleagues [41], another study has been performed with ten PD patients. Here participants were asked to use a home-based assessment tool twice a day (i.e. once in ON state and once in OFF state) for a period of four days. It is their goal to differentiate between a participant's ON state and OFF state based on their test performance. As in other studies by Cunningham et al. [43], the subjects clicked on targets while their speed, time, distance and location of click were recorded. Regarding the time, a statistical significance was found when comparing performances in ON and OFF states ($p = 0.017$). The authors also note that a few subjects showed an increased variance in ON-OFF state which indicates that they might not have been in a clearly defined ON-OFF state at the time of testing. Nonetheless, their results appear to be promising and larger follow-up studies are to be seen.

Wang et al. [173] developed a new method for quantitative evaluation of symptoms in people with PD. Their proposed method is based on free spiral drawings with a digitizing tablet. Spiral drawings itself have been used a number of times to quantify motor dysfunctions (in particular [92, 94, 104] were highlighted by Wang et al.). However, these methods were usually employing some sort of guidance or template. In contrast, Wang et al. utilized free spiral drawings. A group of ten participants enrolled in their study at a hospital in Japan (Kaizuka). All of them were asked to draw a spiral that was then used to extract several features (e.g. number of turns, mean of radius, maximum radius, etc.). Their results indicate that healthy control subjects can be clearly separated from the remaining eight patients regarding the *number of extreme points in radius curve*. Most PD subjects were not able to "rapidly enlarge the circle as spiral" [173]. The authors remark that the features *mean value of radius* and *slope of radius curve* demonstrated stiffness and could be used to distinguish between healthy and pathological subjects.

Pradhan et al. [127] considered a similar methodology, but instead of drawing spirals thirty PD patients were tracking waves by applying force to sensors. Here two force sensors (i.e. one for index finger and one for thumb) were "squeezed" in order to track a wave (i.e. simple sine wave and complex wave with multiple frequency components) with and without mental distraction. Their goal was to provide an assessment tool for clinical progression of PD patients. Pradhan and colleagues state that similar studies have been performed but "which may not be effective in documenting subtle changes in motor control" [127]. When comparing their task of wave tracking (involving precision control), other studies were usually employed to quantify surgical results or treatment progression. Three features were considered: *spectral density*, *RMS error* and *lag*. Although some of the features correlated significantly with UPDRS scores, there have been no significant improvements in prediction. Nonetheless, the authors suggest that their test may add an extra objective measure that other tests fail to capture.

In a publication by Brewer et al. [28], a similar approach has been used to predict UPDRS scores. Here twenty-six participants (all PD patients) were exhibiting pressure on force and torque sensors while they were performing wave tracking tasks. The authors used the same parameters to summarize the participants ability to properly track waves (i.e. spectral density, RMS error and lag). These features were evaluated in terms of their ability to predict UPDRS scores. The authors present four approaches: PCA, least squares linear regression, lasso regression and ridge regression. Their results indicate that ridge regression works best with an absolute error of 3.5 UPDRS points. This is followed by lasso regression (i.e. 4.5 UPDRS points) and PCA (i.e. 7 UPDRS points).

Similarly, Kondraske et al. [82] utilize ordinary computer hardware for specialized PD tests. The authors present an initial evaluation of three objective, self-administered and web-based tests (i.e. alternating movement quality, simple visual-based response speed and upper extremity neuromotor channel capacity). Each test has an equivalent version in the real world based on a testing device called “BEP 1”. Twenty-one subjects (i.e. eight healthy controls and thirteen PD patients) enrolled in their evaluation where both lab-based and web-based tests were performed. The results indicate an encouraging well correlation for lab-based and web-based “rapid alternating movement” and “neuromotor channel capacity” tests. The correlation for the “simple visual” test did not show expected results. The authors envision a three-tiered approach that first involves digital, web-based tests then lab-based tests and finally screening by an expert. As suggested by its nature, web-based tests are easily accessible to a broad population. They provide objective measurements within an uncontrolled environment and may provide an initial assessment on whether any signs of PD are apparent. The second tier can then be used for a complementary assessment in a controllable environment. Afterwards a proper clinical screening can be performed by a neurologist if previous results suggested parkinsonian behavior.

An automatic evaluation approach for early detection of PD is presented by Jobbágy et al. [75]. The authors propose and evaluate a set of tests that were specifically designed to highlight features of PD symptoms. They employ a motion tracking system, called precision motion analysis system (PRIMAS), for recording movements patterns. The system uses a combination of infrared (IR) light, passive markers (i.e. small, lightweight reflective disks mounted on body) and cameras in order to track the participants’ movements of their fingers and hands. Jobbágy and colleagues aim at providing tests and measures to indicate the presence of early to moderate PD and subtle changes in its progression. Twenty-nine participants took part in their study (i.e. thirteen young healthy subjects, ten elderly healthy subjects and six subjects afflicted with PD). Three tasks were performed: tapping task, twiddling task as well as a pinching and circling task. The authors describe their analysis of raw movement data from their tracking system and highlight their chosen features (e.g. frequency, symmetry, dexterity, amplitude, etc.). Based on these parameters a score (between zero and one) is proposed in which people with PD achieve higher score-values (as in UPDRS). Their empirical results indicate that their scale does indeed separate PD patients from healthy subjects.

3.2.7 Summary

It is apparent that indication of PD motor symptoms in time series data is clearly not an empty page. Alone in the past decade a great number of publications with a focus on this very topic have been seen. Some of the mentioned authors have published their work on several symptoms (e.g. Cole et al. [36,37], Salarian et al. [142,143] and Zwartjes et al. [182]).

In recent years, accuracy of symptom indication and severity indication have reached percentages well above 90%. However it should be noted that datasets vary greatly in quality and quantity (e.g. from a few minutes to several hours or days of data). The accuracy increases and decreases with the used datasets and employed algorithms. Authors with small datasets or even synthetic datasets tend to achieve higher accuracies than those that utilize medium-sized (De Marchis et al. [98] used data from four patients and achieved a sensitivity of 99% and a positive predictive value (PPV) of 96% in recognizing tremor) or large datasets (Rigas et al. [136] used data from 23 subjects with roughly 20 minutes of data each and achieved an overall accuracy of 74% in detecting tremor) from real people. Another aspect of quality is the task or activity which was performed during recording sessions (i.e. scripted vs. unscripted, constrained vs. unconstrained, etc.). This section gave preference to those publications that were not using standardized motor tasks to identify symptoms (and their severity). Sensitivities in the range of 90% – 95% (sometimes even greater) were achieved with today’s methods, but usually at the cost of a lower specificity.

However, publications are typically limited by the dataset that they utilize. Most datasets contain signals from well below 30 participants. Most publications with smaller datasets achieve acceptable results, but fail to provide sufficient evidence that the same approach would also work in larger datasets (e.g. $50 \geq$ patient) and under “real”-world conditions (i.e. not in a laboratory environment). The limitations of publications with reasonably sized datasets will typically sacrifice either sensitivity and specificity in order to improve on the other measure. These publications also have a tendency towards signals that were obtained under laboratory conditions. They also typically consider only a single symptom (rather than multiple), thus failing to show how their approach performs in the presence of other symptoms. Generally speaking, the presented state-of-the-art can be improved in the following ways: use of larger database (e.g. ≥ 50 patients), considering symptom diversity (i.e. signals should contain multiple symptoms), signals should not have been recorded in laboratory conditions (but rather in “real”-world conditions or close to it) and increasing accuracy (since most publications with sufficiently sized datasets do not achieve sensitivities and specificities of 90%, 95% or above).

Table 3.2 summarizes the papers that were presented in this section. More elaborated tables can be viewed in the appendix (see Table A.3, Table A.4, Table A.5 and Table A.6). They show the presented papers, but also highlight several other relevant publications. Additionally, more details on each publication are shown in those tables.

Despite the reservation of highlighting papers that enable indication of PD motor symptoms and / or assessing their severity while being mobile, a set of publications that do not fit these criteria was presented. Table 3.3 points to several noteworthy publications with a similar focus, but not necessarily with the intent to identify cardinal symptoms. Therefore, they do not necessarily

present the state-of-the-art. Nonetheless, the interested reader is encouraged to read through them.

The presented research (i.e. publications and their approaches in identifying PD symptoms) yields to the conclusion that commonly experienced PD motor symptoms (e.g. bradykinesia, FoG, dyskinesia, tremor at rest) can be identified with ML and DM techniques. The approaches and utilized feature sets have been briefly outlined. Due to the wide variety of PD related symptoms (see Table A.1), the list was narrowed down to the presented symptoms (i.e. commonly experienced motor symptoms). However, one should keep in mind that additional publications on related topics exist (i.e. other symptoms, ON-OFF-state detection, quantification of PD progression, etc.). These have been left out as their inclusion would have exceeded the scope of this section and chapter.

Symptom	Author(s)	Sensor(s)	Algorithm(s)	Result(s)	Comment(s)
T	Salarian et al. [142]	G	thresholds	Sen.: 76.6% Spe.: 98.0%	Sensitivity has been averaged across pitch, roll and yaw axis.
T	Salarian et al. [143]	G	thresholds	Sen.: 99.5% Spe.: 94.2%	
T	Zwartjes et al. [182]	A, G	thresholds	Acc.: 84.7%	Average of rest tremor and kinematic across varying postures.
T	Rigas et al. [136]	A	HMM	Acc.: 87.0%	
T	Cole et al. [36]	A, E	DNN	Sen.: 93.0% Spe.: 95.0%	Average of results for arms and legs.
T	Roy et al. [140]	A, E	DNN	Sen.: 91.2% Spe.: 93.4%	
T	Niazmand et al. [114]	A	thresholds	Sen.: 80.0% Spe.: 98.5%	
B	Cancela et al. [34]	A	KNN, NN, SVM, etc.	Acc.: 86.5%	
B	Pastorino et al. [120]	A, G	SVM	Acc.: 74.4%	Significant for most features ($p \leq 1\%$) Significant for most features ($p \leq 1\%$) Significant for most features ($p \leq 2\%$)
B	Salarian et al. [142]	G	-		
B	Salarian et al. [143]	G	-		
B	Zwartjes et al. [182]	A, G	-		
F	Djurić-Jovičić et al. [51]	A, G	NN, thresholds	Classification error $\leq 16\%$ (84% acc.)	Classification error $\leq 16\%$ (84% acc.) Sen.: 82.9% Spe.: 97.3% Sen.: 88.3% Spe.: 85.3% Sen.: 73.1% Spe.: 81.6% Empirical difference
F	Cole et al. [37]	A, E	DNN, thresholds	Sen.: 82.9% Spe.: 97.3%	
F	Niazmand et al. [115]	A	thresholds	Sen.: 88.3% Spe.: 85.3%	
F	Bächlin et al. [32]	A, G	thresholds	Sen.: 73.1% Spe.: 81.6%	
F	Stamatakis et al. [157]	A	-	Empirical difference	
(continued)					

Symptom	Author(s)	Sensor(s)	Algorithm(s)	Result(s)	Comment(s)
D	Keijsers et al. [78]	A	NN	Acc.: 96.8%	Averaged across arm, trunk and leg accuracies.
D	Tsipouras et al. [166]	A, G	KNN, RF, DT, etc.	Acc.: 93.7%	
D	Cole et al. [36]	A, E	DNN	Sen.: 91.0% Spec.: 93.0%	
D	Roy et al. [140]	A, E	DNN	Sen.: 90.0% Spec.: 93.4%	Averaged across patients and control subjects.
D	Patel et al. [124]	A	EM	Clusters well separable	

Table 3.2: Summarization of state-of-the-art publications on PD symptom indication algorithms. For each symptom (T: tremor, B: bradykinesia, F: FoG, D: dyskinesia) and reference, the employed classification techniques and utilized sensors (A: accelerometer, G: gyroscope, E: EMG sensor) are highlighted. Furthermore, their results are indicated. It should be kept in mind that the results among these papers are not directly comparable due to employment of different datasets.

3.3 Parkinson’s Disease in Research Projects

Considering the aging population throughout Europe, PD is likely to become an increasingly important health issue. As more and more people grow older they are more likely to show parkinsonian symptoms. Thus, several research projects have been spawned within the last two decades. In general, their focus is to build improved management systems or monitoring systems for people with Parkinson’s. On the one hand, some of these projects are focusing more on symptom indication and severity indication based on time series data (coming from sensors on the patient’s body). On the other hand, some of them pursue the goal of building a complete and closed-loop system that does not only indicate symptoms but also actuates upon this information in order to suppress apparent symptoms (e.g. adjust medication intake or support patients with auditory cueing). In latter case, no commercially successful system of this kind has been build or is in wide use. This is mainly due to a lack of objective and reliable monitoring options (i.e. measurement of dopamine levels). Furthermore, the heterogeneous nature of PD and symptomatic fluctuations thwart the building of such systems. Nonetheless, some projects pursue this very goal (e.g. HELP [65] and REMPARK [133]). These and a few other research projects are highlighted in the remainder of this section.

3.3.1 HELP

The primary focus of HELP (Home-based Empowered Living for Parkinson’s disease Patients) [65] is to build a closed-loop personal health system for people with Parkinson’s. Additionally doctors and physicians can use the system for (remote) monitoring of their patients. It is a 36-months European Union (EU) research project with nine participants from Spain, Israel, Italy and Germany. The consortium is a combination of medical professionals, industry and research partners. HELP started in July 2009 and ended in December 2012.

The basic idea is to reduce time spend in OFF state by optimizing drug administration. Instead of pulse-based medication intake (i.e. in form of a pill taken every few hours), an automated adjustment of continuous drug stimulation (i.e. in form of a pump or tooth-implant) is being thrived for. It is thought that continuous medication administration can reduce the overall drug consumption with equal or improved effectiveness when compared to pills (i.e. impulse-based stimulation). However, regular pumps are setup to deliver a specific dosage which typically stays the same for several months. The general idea is that the dosage could be adjusted much more frequently and in accordance with the patient’s needs. E.g. going for a lengthy walk in the park might require a different dosage than sitting and reading a book. Given this assumption, complications and side effects would be decreased and could enable a more effective treatment with an increased quality of life.

Similar to a diabetes system where medication is administered with respect to measured glucose levels, the HELP system dynamically administers one of two distinct medication dosages that are specific to each patient (i.e. “low” and “high”). Here decisions are based on a set of features which correlate with the ON-OFF state (e.g. amount of movement, location, etc.). Additionally, a web-portal enables medical personal to view and override decisions made by the HELP system. A set of patient specific settings (e.g. dosages and thresholds)

Author(s)	Note
Brewer et al. [28]	Application of Modified Regression Techniques to a Quantitative Assessment for the Motor Signs of Parkinson's Disease
Cunningham et al. [43]	Identifying fine movement difficulties in Parkinson's disease using a computer assessment tool
Cunningham et al. [41]	Home-Based Monitoring and Assessment of Parkinson's Disease
Hamilton et al. [63]	Neural networks trained with simulation data for outcome prediction in pallidotomy for Parkinson's disease
Kondraske et al. [82]	Web-based evaluation of Parkinson's Disease subjects: Objective performance capacity measurements and subjective characterization profiles
Wang et al. [173]	A new quantitative evaluation method of Parkinson's disease based on free spiral drawing
Jobbágy et al. [75]	Early detection of Parkinson's disease through automatic movement evaluation

Table 3.3: Lists several related publications. The author(s) and title of their respective publications are highlighted. This list is intended to supplement state-of-the-art publications (shown in Table 3.2) with additional noteworthy and relevant papers.

as well as information (e.g. appointments) are accessible through the web-portal [8].

The main components of the HELP system are: a body sensor and actuator network (BS&AN), a service and network infrastructure and a web-portal. On the hardware level the BS&AN consists of a sensor platform (with accelerometers, magnetic field sensors, gyroscopes and a temperature sensor), a mobile phone and either a subcutaneous pump (for use in more advanced stages of the disease) or an intra-oral device (for use in earlier stages). The sensor platform is worn around the waist and extracts features regarding the presence of motor symptoms. These features are transmitted to the mobile phone where they are forwarded to the main HELP server and web-portal. Here data are processed and a properly adjusted medication dosage may be send back, if necessary (i.e. sends a command to the pump in order to adjust the drug dose).

3.3.2 REM PARK

Comparing to the HELP project, REM PARK (Personal Health Device for the Remote and Autonomous Management of Parkinson's Disease) [133] uses a similar setup. However, medication is not only dynamically administered but FoG events (freezing episodes) are actively tried to overcome. Here research is focusing on building a disease management system for people with Parkinson's Disease. The general idea is to enable neurologists to manage their PD patients more effectively and thus increasing patient's quality of life. Universitat

Catalunya (Spain) coordinates the consortium that combines researchers, medical experts and industrialist partners. REMPARK started in November 2011 and is expected to end in April 2015.

Most of the consortium's effort is dedicated to building a personal health system (PHS) that features a closed-loop detection of PD motor symptoms and other gait related features, has communication abilities and treatment capabilities. A wearable device is utilized to extract gait related features and thus to determine symptoms as well as ON-OFF state. Additionally, a remote monitoring and management portal is setup. Here medical professionals can review their patient's data and give feedback to the patient (e.g. adjustment of medication dosage, phone call to patient, etc.).

A major objective of REMPARK is the indication of symptoms in real time. For this purpose, two wearable sensor platforms are worn (i.e. one on wrist and one around waist). Additionally, medication delivery components as well as a functional electrical stimulation (FES) device or auditory cueing are utilized to prevent and ease motor symptoms. Based on inertial data information regarding motor status, falls and FoG events are derived. The actual process utilizes a set of AI algorithms. Data are forwarded to a centralized server where it is further processed, evaluated and visualized. This is also where a disease management tool, to enable a more efficient treatment of PD patients, resides.

A smart-phone application is intended to serve as a mobile gateway between patients and clinicians (as well as a gateway between devices and a centralized server). The application includes a set of PD related tests for assessing the patient's clinical evolution. Furthermore, a communication component and a questionnaire component enable subjective feedback by the patients (e.g. quality of sleep, etc.). A cueing system is implemented that intends to help patients in overcoming FoG events. Here, two versions (i.e. one auditory and one haptic) are being realized and evaluated. While the auditory cueing system uses sounds to overcome FoG events, FES is going to be used in the haptic version.

3.3.3 PERFORM

The PERFORM (Personal Health Systems for Monitoring and Point-of-Care Diagnostics-Personalized Monitoring) [125] project is also funded by the EU and focuses on improving PD patients' quality of life. Their primary concerns are research and implementation of a system that enables an autonomous and objective assessment of the patient's motor status. PERFORM is led by Universidad Politecnica de Madrid (Spain). In total, fifteen partners from six countries were part of its consortium and contributed their experiences from a variety of fields (e.g. computer science, technology, industry, medicine and research). The duration was 36 months (from February 2008 till January 2011).

The project's objectives are manifold: monitoring of patients in their natural environment, modeling of movements and symptoms in a patient-specific manner and objective assessment of patients' motor status. Furthermore, a decision support tool for physicians and patients is among the objectives. To achieve those goals a flexible wearable monitoring system (capable of recognizing symptoms as well as their severity) was developed. The system can be split into: a remote motor behavior recorder (located in the patient's normal environment) and a point of care platform (central hospital processor). The former aggregates data while the latter processes it.

It should be mentioned, that the consortium does not specifically focus on PD. It rather considers problems that are of a more general nature (e.g. remote health monitoring, motor status assessment and treatment personalization). It is noteworthy that these problems do not necessarily apply to PD patients only but also to patients suffering from other movement disorders or motor neurodegenerative disorders.

A set of wearable sensors, advanced processing techniques and fusion algorithms are employed. They are intended to capture relevant information with regard to the patient’s motor status. Data are stored locally and later forwarded to a centralized server where it is further processed, fused and evaluated. The sensors themselves are located on the patient’s body either in a garment or as an accessory.

A modular architecture is used to facilitate different combinations of measurements. This ranges from tremor with accelerometers and gyroscopes to bradykinesia with standardized motor tasks. All sensing devices communicate in a wireless fashion and can be monitored with a patient-customized monitoring tool. Raw data from the sensor is preprocessed before it is forwarded to a centralized server where medical personal can monitor the patient’s pathological evolution as well as efficiency of their current treatment.

3.3.4 TREMOR

The TREMOR (An ambulatory BCI-driven tremor suppression system based on functional electrical stimulation) [164] project does not specifically focus on people with Parkinson’s Disease either but rather on tremor as a movement disorder on its own. Tremor may be apparent in varying frequencies and may affect a variety of body parts. Patients with PD typically experience a tremor within $4 - 6\text{Hz}$. This research project pursues the goal of mechanically suppressing tremor through means of FES and a brain-to-computer interaction (BCI)-driven analysis. TREMOR was coordinated by Agencia Estatal Consejo Superior de Investigaciones Científicas (Spain; a.k.a. CSIC) and further eight partners from six countries were part of the consortium. The project started in September 2008 and ended in August 2011.

A system for monitoring motor activities and capable of indicating tremor (i.e. involuntary motor activities) was build. The main objective was to evaluate and validate this system on various levels (i.e. technical, clinical, functional and conceptual). It utilizes a multi-modal brain interface (i.e. EMG and electroencephalography (EEG)) and a wearable sensor unit (i.e. inertial measurements) for signifying tremor or tremor-like behavior within the streamed data. A FES system is applied to the patient’s symptomatic limb. Here, an array of FES probes are used to suppress episodes of tremor once they occur.

3.3.5 Summary

In general, it can be noted that quite a bit of research has been done in the direction of automatically indicating PD related motor symptoms in time series data. Some of this research came directly from projects like TREMOR [164] and REMPARK [133] (see Table A.3, Table A.4, etc.). This section has presented four research projects related to PD. All of them have influenced this thesis in one way or another. It is because of these research projects that some of

the previously highlighted publications were published at all. Thus some of the insights and state-of-the-art papers of this chapter would not exist if it wasn't for these projects. Nonetheless, not all publications can be directly attributed with such research projects.

With the exception of REMPARK, none of the presented projects are directly related to this thesis (i.e. apart from the publications that were attributed to them). As will be pointed out in a later chapter (see Chapter 5), this thesis has utilized a database (DB) with recordings from 92 PD patients that belongs to the REMPARK consortium.

By highlighting these research projects, it was the author's intention to broaden the horizon and context in which the previously presented state-of-the-art papers can be seen. All of these projects have contributed to the state-of-the-art and thus it is only fair to credit them. This section has also provided a foothold for interested readers that are looking for publications beyond the scope of this thesis.

Chapter 4

A Framework for Time Series Analysis

This chapter highlights the development process of a framework for time series analysis and its applications. Thus, a solution to the first research question (see Section 1.2) is presented. Several design aspects as well as their applicability are discussed. Furthermore, different ways of extending and maintaining the framework are described in the course of this chapter.

Unlike the development process, “Modules of Framework” (see Section B.3) presents a brief description of individual modules as well as a road-map (see Section B.4) for implementing the algorithms found in Chapter 6. Those readers, that are interested in this part of the framework, may want to have a look at the named chapter.

4.1 Design and Development

The framework, being designed and developed in this chapter, is intended to be as general purpose as the topic of time series allows. Due to the nature of this thesis, an initial selection of filters and algorithms was chosen, that is coined toward an analysis of Parkinson’s Disease (PD) related time series. Apart from this initial choice the framework is by no means tuned toward PD. Instead PD provides an adequate application domain with reasonably complex scenarios, where real people may benefit from analysis of time series data (see Section 4.3.1). However PD is not the only application scenario, other applications include but are not limited to analysis of currency exchange rates (see Section 4.3.2), analysis of network traffic (see Section 4.3.3), quality control of Open Street Map (OSM) data (see Section 4.3.4) and many others.

Many publications of new algorithms do not provide sufficient comparisons against competitive state-of-the-art techniques [21]. In this context, a framework (as described in this chapter) may enable researchers to evaluate their algorithmic approaches in comparison to known techniques (under reproducible conditions) more easily. However, this assumes that a great subset (if not all) of state-of-the-art techniques is already implemented in the framework. Nonetheless, this framework could provide a decent foundation which reduces burden and effort of future algorithm developers. Researchers and practitioners can

benefit from this framework alike. On the one hand, practitioners can utilize these algorithms and apply them to their domain and problems. On the other hand, researchers can easily develop and test their algorithms against a feasible set of (related) techniques and approaches. The framework can also be used in classroom scenarios (i.e. one can quickly swap out modules and one can easily observe the behavior of modules which may be desirable attributes in this type of scenario).

The overall idea is to implement filters, machine learning (ML) techniques and artificial intelligence (AI) algorithms in the form of reusable modules. These modules are then fed with data and their outputs can be utilized to feed further modules. Thus, an entire network of interconnected modules can be created. Keeping this in mind, the framework can be divided into two major parts: data processing and information flow. These parts are described as follows.

- **Data Processing:** Each module in the framework is intended to perform a single (specific) purpose and should be configurable to a certain degree. Configurability increases reusability of modules and supports rapid prototyping, while keeping the total number of modules manageable.
- **Information Flow:** Outputs of modules need to be managed and routed to their corresponding successors. Here it is crucial that not only sequential work-flows are possible, but rather arbitrary complex ones should be producible (e.g. loops, parallel strains of processing, etc.)

Both of these major components are intended to be kept as dynamic and flexible as possible to a degree that is still feasible and does not degrade user experience. The remainder of this section highlights several design challenges and presents solutions to them.

4.1.1 Design Challenges

In the following, a set of design challenges is enumerated. Each of these design challenges covers a particular part of the framework and a short overview of the area that they cover is given for orientation purposes.

- **Structure and Design of Modules:** One of the core elements of the framework is the representation of modules. They represent an “atom”-like element in the framework. Their design and structure are important and affect large parts of the framework. For these reasons, it is imperative to state their role, functionalities, responsibilities and boundaries.
- **Flow In-between Modules:** Part of a module’s nature is reusability. Inter-connectivity between modules allows them to unfold their full potential. The way modules interact with each other and the way modules can be connected with each other requires a careful design as it largely affects the framework’s usability.
- **Data Sources and Sinks:** The framework must provide not only the option to artificially generate data, but also the option to read from and write to predefined places (e.g. files, universal resource locators (URLs), databases (DBs), etc.). In addition, the data format, which is read or written, must be adjustable. The way that the framework deals with this is captured in this design challenge. The realization can greatly influence usability and therefore user experience.

Each of these design challenges cover significant parts within the framework and thus they are discussed individually. Hereafter, each section is devoted to a single design challenge. Within each section, the design challenges are further divided into chunks that are more manageable. These chunks are separately discussed and will typically be illustrated with at least a diagram showing the general implementation idea as well as a diagram with the actual implementation. These diagrams are kept as compact and simple as possible in order to facilitate an easier understanding of the matter in discussion. Thus, it is possible that less relevant implementation details were omitted. This was done for the sake of simplicity as well as to keep the diagrams from cluttering up.

4.1.2 Structure and Design of Modules

Here the structure and overall design aspects of modules are discussed. This design challenge influences large parts of the framework and must be cautiously considered. With respect to the requirements (see Section 1.2), the following points shall be touched upon in the course of this section.

- **Reusability:** The modules of the framework are intended to perform tasks (i.e. reading data from a file, computing moving average, classifying data samples, etc.) such that each module can be utilized multiple times. Configuration of modules should also be constructed in a way that allows reusability.
- **Extensibility:** Adding further modules to the framework must be easily possible (e.g. new data sources, AI algorithms, etc.). Usability of the framework needs to be weighted against the ease of adding new modules.
- **Iterative:** The modules of the framework must be designed such that they are capable of handling large amounts of data. Their structure should not limit the amount of throughput, but rather enable utilization of arbitrarily large datasets (i.e. starting with several thousands of data samples and going to potentially infinite datasets).
- **Scalability:** Similarly to the above requirement, the modules should not only be able to handle infinite data streams, but the framework should also cope with large amounts of utilized modules (i.e. several hundred modules).

Which functionality must a module provide?

The concept of reusable modules is important. Their proper design is crucial to the usability of the framework. It must be clear which responsibilities and functions are taken care of by these modules. Each module shall perform a single task only. The idea is that a single task per module increases reusability, while too abstract and complex tasks or very specialized tasks in a module are likely to decrease reusability. Here, the Strategy design pattern is utilized. It defines a simple interface that allows the framework to pass data into modules and retrieve data from them. This has the benefit that clients can simply utilize the given interface for any module. Thus allowing them to use different modules interchangeably. On the other hand, a naive implementation would likely require a client to tightly couple with utilized modules. Thus making maintenance and reusability difficult or even impossible.

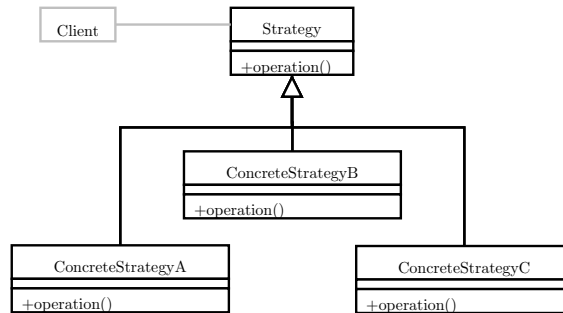


Figure 4.1: Illustrates the structure of the Strategy design pattern as defined by [57, p. 349]. The relationship among *ConcreteStrategy* classes and abstract *Strategy* is highlighted.

The Strategy design pattern is an object-based behavioral design pattern that allows the flexibility of choosing any algorithm from a family of algorithms at run-time. The following paragraphs are based on the description of the Strategy design pattern as shown in [57, p. 349].

Structure: The structure of the Strategy design pattern is illustrated in Figure 4.1. Its general idea is to provide an algorithm’s interface and create a whole family of algorithms by sub-classing or implementing this very same interface.

Participants: The Strategy design pattern has three participants, which are described hereafter.

- **Strategy:** Represents the main class of this design pattern. It defines an interface that all algorithms in a given family have in common. The *Client* object references a *Strategy* instance and utilizes it to call the algorithm defined by it.
- **ConcreteStrategy classes:** Each of them represents a concrete strategy or algorithm within the family. These are the concrete implementations of the *Strategy* interface. At run-time any of them can be chosen.
- **Client:** This is utilized to call *Strategy* classes. It references a *Strategy* instance and may take care of its initialization.

Collaborations: Developers are meant to avoid working with *ConcreteStrategy* classes directly in order to avoid tight coupling. The *Client* and *Strategy* classes work together. The *Client* provides all necessary information to a *Strategy* object (either directly or through an intermediary). In most cases, a *Client* will collaborate with an intermediary *Context* object (e.g. instructing it to use a particular strategy) rather than interacting with it directly.

Implementation: An implementation of the Strategy design pattern is shown in Figure 4.2. No major adjustments of the design pattern were required for the realization in this context.

The *Processor* class defines a simple interface with three methods which also closely resemble the module’s life circle. The *Strategy* interface is represented

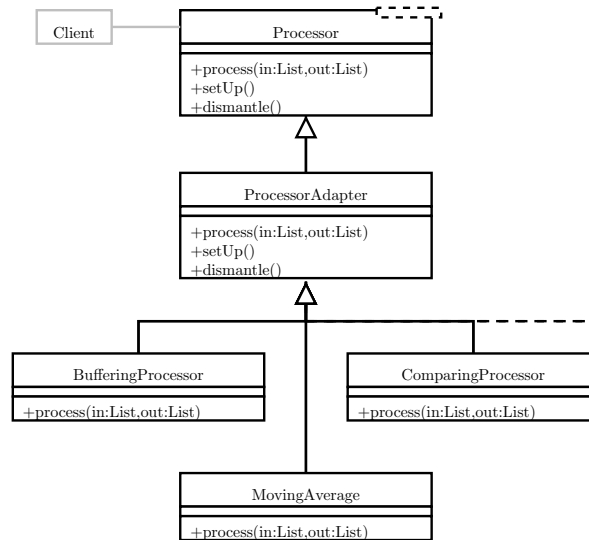


Figure 4.2: Shows the implementation of the Strategy design pattern. Here the *Processor* interface and a few exemplary implementations are highlighted.

by this very same interface. The *process* accepts input values as well as a list for output values. It is the idea that the framework passes a set of input values in (i.e. first parameter) and a second set of values is interpreted as output data by the method (i.e. second parameter). Concrete implementations of the *Processor* interface are meant to read the input data (if appropriate) and provide some output values. However, utilization of the input data or output data are optional. A module does not have to provide output data nor does input data have to be interpreted. In any case, it should be kept in mind that altering the input data can affect other modules as well (e.g. changing order in list or manipulating single entries).

Furthermore, the methods *setUp* and *dismantle* are part of the interface. They are called by the framework to initialize and to finalize a module. The *setUp* method is called by the framework before the *process* method is called for the first time. It is intended to allow developers to perform any configurations, preparations and so on. The *dismantle* method is called by the framework when the module is no longer needed and its allocated resources can be freed.

The overall life circle of a module is illustrated in Figure 4.3. There it also becomes obvious that the *setUp* and *dismantle* methods may be called more than once within the life time of a module.

How can modules be configured?

As described earlier, each module is intended to perform a single task only. However, it is not the author's intention to create a gigantic pool of modules in which each module performs a very specialized task (e.g. calculate average of last five data points, low pass filter with a cut-off frequency at 100 Hz, etc.). Instead modules shall perform tasks with a feasible and sensible layer of abstraction (e.g. calculate moving average of last $n \in \mathbb{N}^+$ data points, low pass filter with

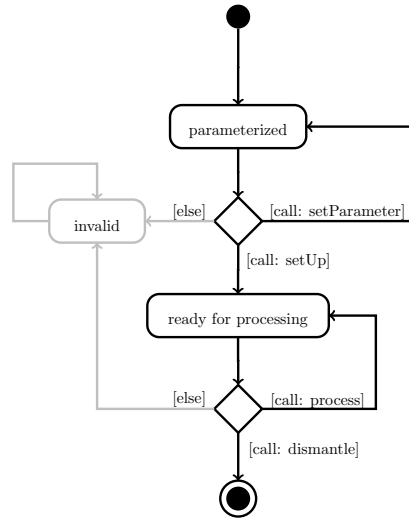


Figure 4.3: Shows the life circle of a module within the framework.

a frequency of $m \in \mathbb{R}^+$ Hz, etc.). Thus, a reasonable level of configuration is desirable. It keeps the number of modules from exploding while still enabling reusability (even fostering it).

A naive realization of such functionality could implement configurable parameters in each module individually (e.g. *getFrequency* and *setFrequency* in case of the low pass filter). However, this approach would enforce a tight coupling between modules and clients using them. Thus clients would no longer be able to utilize the *Processor* interface to its full potential. Instead, they have to remember the interface for each module separately. For these reasons the Strategy design pattern is utilized. A common interface for configuration of parameters is defined. Thus reducing the coupling between modules and clients, as they can again utilize the *Processor* interface.

Design Pattern: The structure and their corresponding participants of the Strategy design pattern have been described earlier in this chapter. For more details on the Strategy design pattern refer to “Which functionality must a module provide?” (see Section 4.1.2).

Implementation: Figure 4.4 highlights the implementation of the Strategy design pattern in this context. There was no need for major adjustments.

The *Strategy* interface is represented by the *Configurable* class. The *Configurable* interface contains three methods for retrieving and manipulating parameters. The interface can then be implemented in various ways and may be reused throughout the framework. It can be utilized to retrieve all available configuration parameters (i.e. *getParameters* method), to retrieve actual configuration parameter values (i.e. *getParameter* returns the corresponding value given a parameter name) and manipulate configuration parameter values (i.e. *setParameter* sets the corresponding value that is associated with a parameter name).

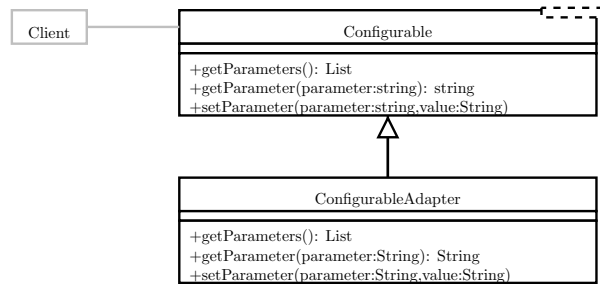


Figure 4.4: Shows the implementation of the Strategy design pattern in the context of configurability. The classes *ConfigurableAdapter* and *Configurable* are shown.

The reason for this interface is that there are multiple places in the framework where such configuration options are useful. They are shown in “Which functionality must a module provide?” (see Section 4.1.2), “How does information travel in-between modules / nodes?” (see Section 4.1.4) and “Can data format and kind of stream vary independently?” (see Section 4.1.3). If it was not for them, the *Configurable* and *Processor* interfaces could have been merged into a single interface.

How can parameters be constrained or conditioned?

With the introduction of configuration parameters, it also becomes necessary to constrain them. This may not be required in all cases but certainly many configuration parameters will need some type of constraint in order to alert about invalid configuration options (i.e. only positive numbers, only integers, only a set of values, etc.).

In order to allow a large diversity of such constraints and reduced coupling, the Strategy design pattern is used once more.

Design Pattern: For more information on the Strategy design pattern refer to “Which functionality must a module provide?” (see Section 4.1.2). There, its structure and participants are described in more detail. The pattern itself is an object-based behavioral design pattern that allows interchangeability of components within a family of algorithms at run-time.

Implementation: Figure 4.5 shows the structure and implementation of the Strategy design pattern in this context. Again, no major adjustments were required.

The *Strategy* class is represented by the *Condition* interface which contains a single method for determining the compliance of a value with a given configuration parameter. The method *complies* is utilized by the framework to ensure that all configuration parameters have valid values. Here, it can be safely assumed that the configuration parameter will always have a non-empty value (i.e. non-null). The idea is that this method is called right before a configuration parameter would be changed. Depending on the result of this method, the new configuration value is either accepted or rejected.

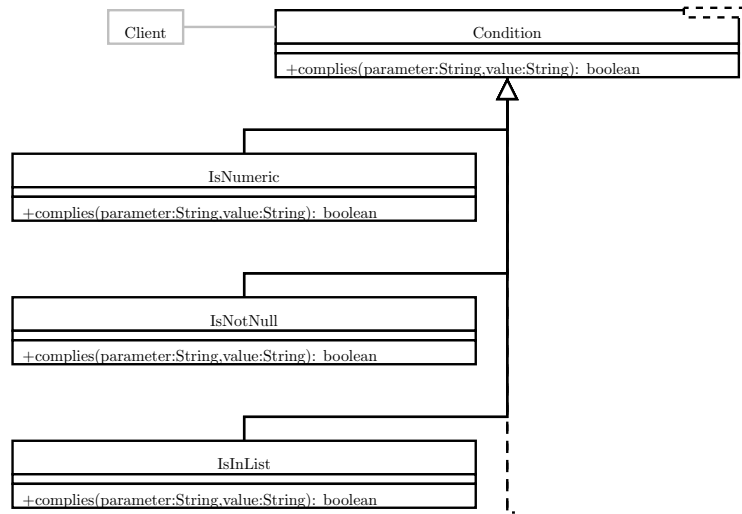


Figure 4.5: Shows the implementation of the Strategy design pattern for constraining configuration parameters. The *Condition* interface and several concrete implementations are highlighted.

How are dependencies and relations between parameters expressed?

With increasingly capable modules, it is likely that the need to express relations among configuration parameters arises. This can be helpful to automatically convert between a set of different units (e.g. °C and °F, seconds and minutes, etc.), but it can also be used to further constrain configuration parameters based on another parameter's value. In short, it synchronizes configuration parameters within a module as long as a relation among them has been (explicitly) defined.

A naive approach might implement such functionality directly in the corresponding module, which would also be a viable solution if the implemented functionality is used only once. However, this is not necessarily the case. A mechanism that allows to keep track of state changes can be useful for various other purposes and not just for automatically converting between units. The Observer design pattern provides a similar functionality but without unnecessary coupling and redundancies. Consequently, this design pattern is utilized.

The Observer design pattern is an object-based behavioral design pattern that defines an one-to-many dependency between objects such that changes in one object are automatically reflected in other objects. The following paragraphs are based on the description of the Observer design pattern by Gamma et al. [57, p. 293].

Structure: The general structure of the Observer design pattern is highlighted in Figure 4.6. There it can be seen how *Subject* and *Observer* classes relate to each other. Generally speaking, *Subject* classes provide methods for attaching, detaching and notifying their observers. The corresponding *Observer* objects are automatically notified every time the state of a *Subject* is changed.

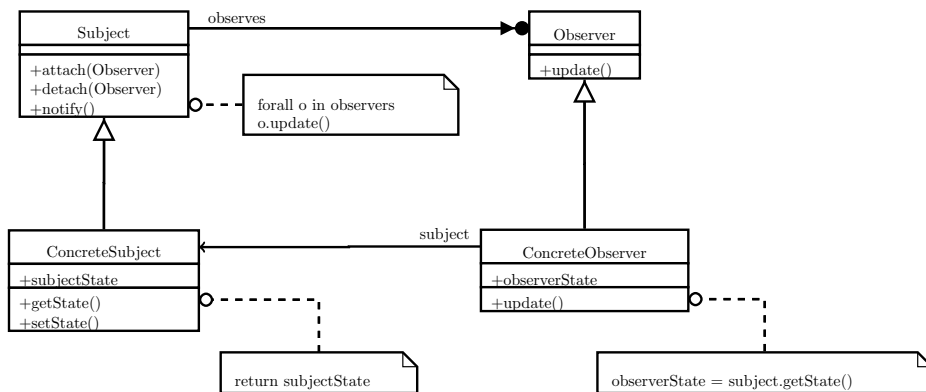


Figure 4.6: Illustrates the structure of the Observer design pattern as defined by [57, p. 293]. The relationship among *ConcreteObserver* class and abstract *Observer* is highlighted as well as their relationship to *ConcreteSubject* and abstract *Subject*.

Participants: The four participants of the Observer design pattern are described hereafter.

- **Subject:** Resembles one of the main interfaces in this design pattern. It provides an interface for attaching, detaching and notifying observers. Any number of observers can express their interest in a particular *Subject* by registering themselves.
- **ConcreteSubject:** Implements the *Subject* interface and notifies registered observers as changes occur. It also contains the state that *ConcreteObserver* objects are interested in.
- **Observer:** The second main interface in this design pattern. It defines methods for updating *Observer* objects as changes to their observed subjects occur.
- **ConcreteObserver:** Implements the *Observer* interface and keeps a reference to all subjects that it observes. Furthermore it duplicates those parts of the subject's state that are relevant to it.

Implementation: The realization of this design pattern is shown in Figure 4.7. No major adjustments were required.

The *Observer* interface requires a single method to be implemented. The *update* method takes the actual *Configurable* component as well as the configuration parameter being manipulated. It can be safely assumed that the *Configurable* object and configuration parameter will always have a non-empty value (i.e. non-null). This method is called to notify observers about changes in configuration parameters, to keep multiple of such parameters synchronized as well as keeping track of their changes. Internally concrete implementations of the *Observer* interface will make sure to adjust other configuration parameters according to a change.

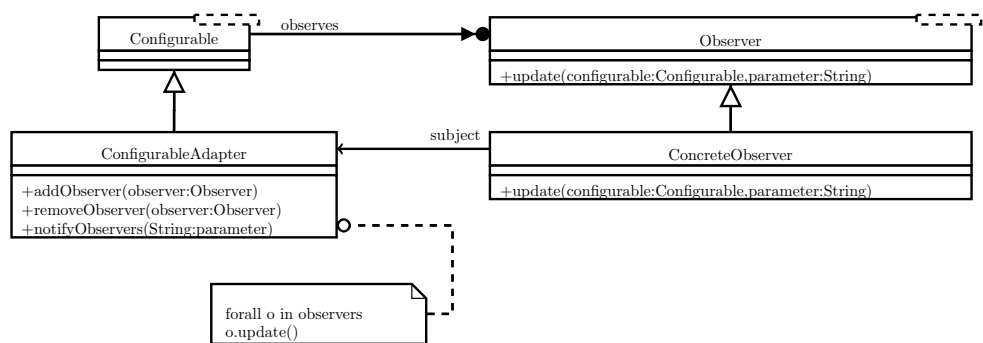


Figure 4.7: Shows the implementation of the Observer design pattern. Here the *Observer* interface and an exemplary implementation are highlighted.

How do these objects (i.e. *Configurable*, *Condition*, *Observer*) work together?

The last few statements were concerned with various aspects of configuration. Here the interaction between those parts of the framework is discussed. It is one thing to provide constraints and express relations among configuration parameters, but it is a very different thing when and where they come into play. The general idea is to provide a default implementation that combines all these aspects.

This can be realized with the Adapter design pattern. It allows to combine several interfaces while still adhering to the *Configurable* interface. Another benefit is that it can be reused as often as required. In contrast, a naive approach would likely result in code redundancies as the same functionality would be implemented in different parts of the framework (e.g. individual modules would utilize the *Condition* interface separately).

The Adapter design pattern is an object-based structural design pattern. It allows multiple otherwise incompatible interfaces to work together and adapts them to an interface that a client expects. The following paragraphs are based on Gamma et al.'s view of this design pattern [57, p. 157].

Alternatively, this could also have been realized with the Facade design pattern [57, p. 208]. However, the result would have been the same (or at least very similar). Due to the simple nature of the would-have-been sub-system, it was decided to utilize the Adapter design pattern instead.

Structure: Figure 4.8 shows the Adapter design pattern. The overall idea is to adapt an interface to an expected interface and thus allowing otherwise incompatible interfaces to work together.

Participants: The four participants of the Adapter design pattern are described hereafter.

- **SomeClass:** This class defines an interface that is intended to be used by clients.
- **Client:** Uses classes of the above type.

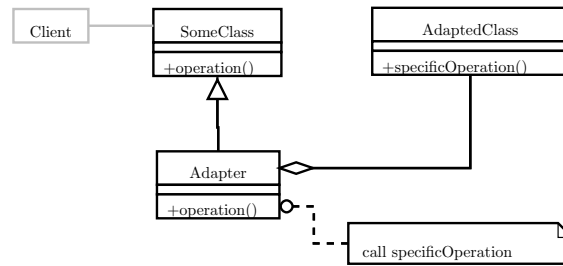


Figure 4.8: Illustrates the structure of the Adapter design pattern as defined by [57, p. 157].

- **AdaptedClass:** This class defines an interface to be utilized by clients. It can also be a set of classes, which should be used by clients.
- **Adapter:** This is the main class of the design pattern. It makes *SomeClass* and *AdaptedClass* compatible.

Collaborations: Clients utilize *Adapter* objects in the same way they would treat *SomeClass* objects. The only difference is that *Adapter* objects call operations on the *AdaptedClass* class in order to complete requests from clients.

Implementation: The implementation of the Adapter design pattern can be reviewed in Figure 4.9. There were no major adjustments necessary.

The *Adapter* class is represented by the *ConfigurableAdapter* class and the *AdaptedClass* class is represented by both *Condition* and *Observer* interfaces. The *Configurable* class represents *SomeClass*.

The *ConfigurableAdapter* class acts as a *Configurable* class, but also allows constraints and relations among configuration parameters. To clients the extra functionality is completely transparent. Only those parts of the framework that need to know about this extra functionality have the ability to manipulate constraints and relations. All other parts will simply be informed if a constraint was not met. Internally, the *ConfigurableAdapter* class queries all related *Condition* implementations right before a configuration parameter would be changed. Only if all conditions for a given parameter accept this change, then the actual change is performed. Otherwise, the change would be rejected. As for the observers, they are notified after a change has occurred.

How can typed parameters (i.e. not just strings) be supported?

The configuration of modules and links (see “Design Challenges” in Section 4.1.1) are coined towards utilization of strings to enhance portability of the framework (to different programming languages and operating systems). However, it is likely to become frustratingly cumbersome to convert non-string parameters back and forth all the time. To ease this burden, the framework employs the Template Method design pattern in order to provide access to parameters in the form of primitive data types. This enables the change of parameters without the hassle of converting them manually.

The Template Method design pattern offers access to parameters in the form of primitive data types (i.e. integer, boolean, floating point numbers,

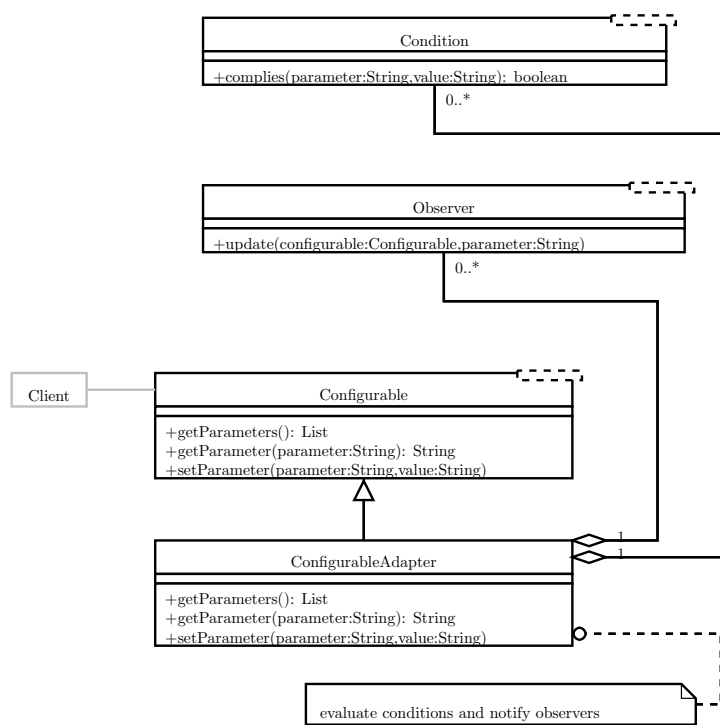


Figure 4.9: Shows the implementation of the Adapter design pattern as part of the framework. Here, the interaction between various configuration related parts of the framework (i.e. *Configuration*, *Condition* and *Observer*) is highlighted.

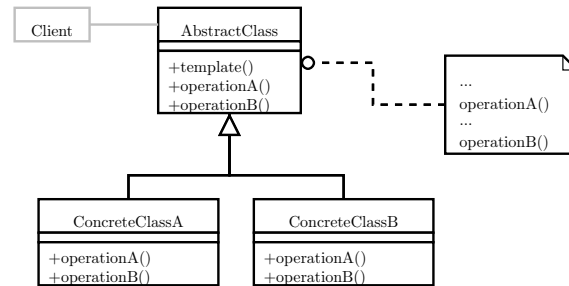


Figure 4.10: Illustrates the structure of the Template Method design pattern as defined by [57, p. 360]. The *AbstractClass* defines a set of template methods as well as a set of function-specific operations. *ConcreteClass* classes are meant to change the behavior of these operations. The template methods are typically based on these operations as well.

etc.). However, it internally only utilizes the original string-based interface. In doing so the Template Method takes care of the conversion to and from strings, whereas a naive implementation would likely store these parameters of different types separately. Thus affecting portability of the framework as it increases the framework's dependency of the chosen programming language and in some cases the operating system as well (i.e. at the very least it makes it more difficult and complex to port).

The following paragraphs are based on the definition of the Template Method design pattern by Gamma et al. [57, p. 360]. It is an object-based behavioral pattern, which defines a skeleton of an algorithm or general functionality and defers steps of it to sub-classes. This enables sub-classes to adapt the algorithm to their needs without changing the algorithm's overall structure.

Structure: The structure of the Template Method pattern is illustrated in Figure 4.10. Its general idea is to provide a unified structure for an algorithm and defer certain parts of that algorithm to sub-classes.

Participants: The Template Method design pattern has two participants, which are described hereafter.

- **AbstractClass:** This is the main class of the design pattern. It defines the overall structure of an algorithm as well as those steps that are subject to variations. The actual template methods then utilize this structure.
- **ConcreteClass classes:** These classes represent variations of the original algorithm, but do not change its structure. Here only variations of the original algorithm are implemented.

Collaborations: The *AbstractClass* class is an interface that defines a method for each step, in which an algorithm is subject to change. It then utilizes these methods in its template methods. The actual realization or specialization of the algorithm's steps is left to *ConcreteClass* classes.

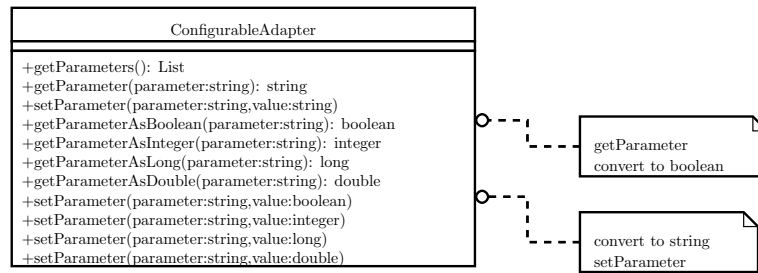


Figure 4.11: Shows the implementation of the Template Method design pattern. The *ConfigurableAdapter* class contains both function-specific methods (i.e. *getParameter*, *setParameter* and *getParameters*) and template methods (e.g. *getBoolean*, *setBoolean*, etc.).

Implementation: The actual implementation of the Template Method design pattern can be reviewed in Figure 4.11. The naming and structure of the interfaces have been adapted to reflect the context, in which they are used, more accurately.

Here the *AbstractClass* class is represented by the class *ConfigurableAdapter*. This class provides three primitive methods that are subject to variations and it defines several template methods that utilize them. The actual primitive methods correspond to the interface defined by *Configurable* (see How can modules be configured? in Section 4.1.2). All template methods direct their calls towards these methods, thus allowing to store and to retrieve configuration parameters of numerous data types (i.e. boolean, integer, etc.).

E.g. storing and retrieving of configuration parameters as a boolean value is realized in the template methods *getBoolean* and *setBoolean*. Both of these methods utilize the *Configurable* interface internally. Thus, they also take care of any necessary conversions between string values and their boolean representations.

One of the most obvious differences is the absence of any concrete sub-classes. This is due to the fact that *ConfigurableAdapter* already provides a default implementation for each primitive method. Nonetheless, it can be sub-classed and the desired parts can be overwritten.

Can a module's functionality be dynamically extended?

Especially during development, it can be useful to dynamically extend a module's responsibilities. This can be leveraged to generate statistics such as processing time that a module requires to compute output values, determining the actual sampling frequency at a given module and many more. Other applications could ensure that parameters of a module are read-only or that certain parameters are ignored.

A naive implementation could realize such suggestions by directly sub-classing a given module and manipulating it in a way that it meets their requirements. However, this approach is quite cumbersome and has several drawbacks. It completely neglects the fact that all of the suggestions above can be realized independently of a concrete module. Any module could be subject to a statistical analysis, but sub-classing all modules to include extra functionalities is neither

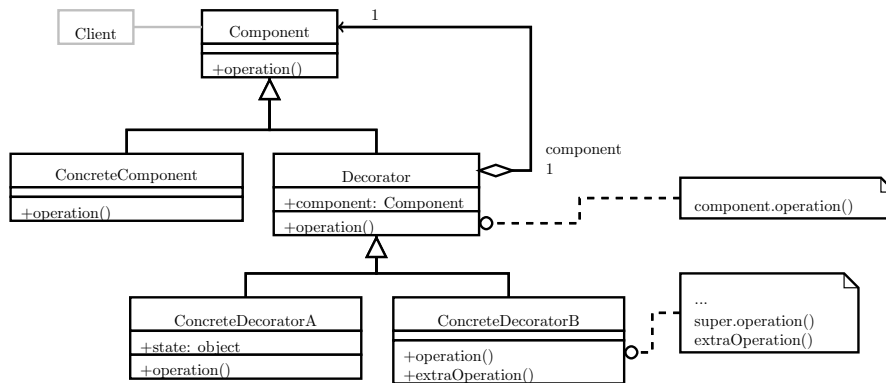


Figure 4.12: Illustrates the structure of the Decorator design pattern as defined by [57, p. 196]. The generic *Component* interface is shown as well as the *Decorator* classes. Their relationship is highlighted.

feasible nor a practical approach. Further drawbacks are code redundancies, increased chance of errors as well as minimal reusability. Instead the Decorator design pattern could be utilized. It allows to dynamically wrap any module and easily allows a module to be extended with new responsibilities, which would not have been easily possible with the naive approach.

The Decorator design pattern is an object-based structural pattern. It allows a class to extend its responsibilities in a dynamic way without sub-classing it. An elaborated review of this design pattern can be found in [57, p. 196].

Structure: The structure of the Decorator pattern is illustrated in Figure 4.12. By referencing a given module, the Decorator can wrap it and extend its responsibilities. Therefore developers can still use the component's interface and treat decorated objects in the same way they would treat regular modules (i.e. without having to make a distinction between them).

Participants: The Decorator design pattern has four participants, which are described hereafter.

- **Component:** Represents the main class of an object-structure. Furthermore, the class declares operations that are common to all components and it provides default implementations where appropriate.
- **ConcreteComponent classes:** They are concrete realizations of the *Component* interface. One or more of these implementations are meant to be extended in terms of functionality and responsibility.
- **Decorator:** Is the main class of the Decorator design pattern. It defines the same interface as a regular *Component*, but manages a reference to an arbitrary component and wraps all interface calls to it. Thus, the decorated component can only be indirectly accessed. All queries and calls have to go through the *Decorator* object.
- **ConcreteDecorator classes:** They extend the *Decorator* class and add any desired functionality to it.

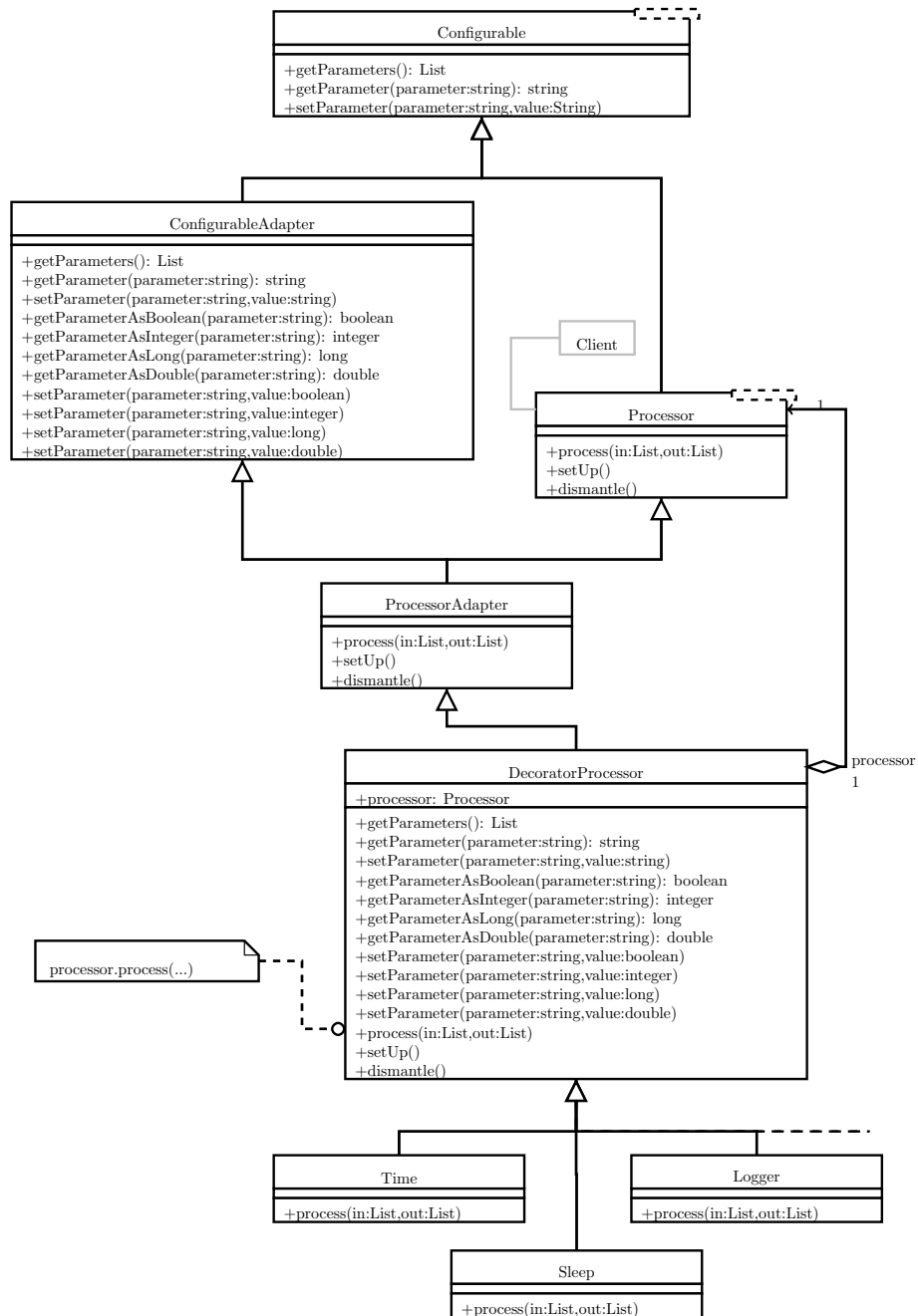


Figure 4.13: Shows the implementation of the Decorator design pattern in the framework. Several variations of the *Decorator* class are shown.

Collaborations: The *Decorator* class forwards any call to the referenced *Component* object and optionally executes additional operations before and after doing so.

Implementation: An implementation of the Decorator design pattern is shown in Figure 4.13. No adjustments of the pattern were required.

The *Component* class is represented by the *Processor* interface, while the design pattern’s main participant *Decorator* is represented by the *DecoratorProcessor* class. The default implementation of the *DecoratorProcessor* class references a single *Processor* instance and wraps every method call to it. When sub-classing *DecoratorProcessor*, the desired functionality can be realized by overwriting one or more methods.

4.1.3 Flow In-between Modules

This section is dedicated to the description of information flow in-between modules. More precisely, it highlights several important aspects associated with inter-module communications. The following requirements are related to this design challenge.

- **Iterative:** The modules in the framework need to be invoked with every data sample of a data stream. This part of the framework must be designed to enable just that.
- **Scalability:** This part of the framework has to make sure that modules are delivered with their data samples. It should not matter whether the stream contains several hundred data samples nor should it matter whether it ever ends.
- **Flexibility:** The data flow between two modules does not need to be designed in a static way. Instead, it should allow filtering (i.e. only passing integers, ignoring string values, etc.) to enable a dynamic flow and flexible junctions between modules.
- **Reusability:** Just as important as the previous design challenge, the components in this part of the framework should be designed in such a way that they can actually be reused throughout the framework.
- **Extensibility:** It must be possible to add further kinds of *Link* components and *Iterator* components. This should be possible without a hassle.

How can compositions and individual objects be treated in a unified way?

Even though each module is intended to stand on its own (i.e. perform a single task without prior assumptions on preceding and succeeding modules), their usefulness in terms of reusability truly shows when being part of a larger structure or graph. Of course, a module can function on its own. However, its full potential can most likely be harvested once data from one or more modules passes through. It is important to note that the chosen representation strategy of connections between modules affects the possible structures that can be formed.

An easy realization would allow modules to have an arbitrary number of successor modules (or “children”) as well as a preceding module (or “parent”).

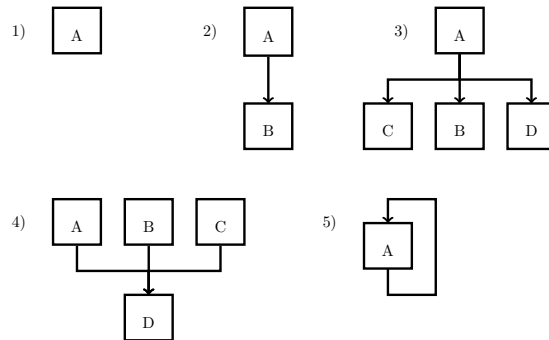


Figure 4.14: Shows possible structures that can be created with tree-like and graph-like representations. Items 4 and 5 demonstrate the main difference between both approaches. In order of appearance, the modeling situations refer to: “single node”, “single child, single parent”, “multiple children, single parent”, “single child, multiple parents” and “cycle”. Each box represents a single module whereas each sub-figure depicts an “atomic” modeling situation. One can replace a box with any other modeling situation and still have a functional structure.

Thus effectively allowing tree-like structures to be formed. In some cases, this may be all that is required. However, there are cases which cannot be expressed in this way (e.g. loops, parallelism, merging of “processing” paths). Given the constraint of having a single preceding module, only tree-like structures with a single root (i.e. first, topmost node in the tree) can be created. This, however, does not directly allow use of multiple data sources at the same time (e.g. reading data from a file and reading complementary data from a second file). However, when relaxing the above constraints, not only tree-like structures can be formed but also graph-like structures become possible (trees only have a single root-node and do not allow cycles while graphs do allow cycles and can have multiple entry points). This can be achieved by allowing the presence of multiple preceding modules (or “parents”). Figure 4.14 indicates possible structures for both approaches.

Furthermore it is not only desirable to allow for multiple data sources, data sinks, and so on, but also to layer graph-like structures themselves. Thus enabling a single node in the graph to hold an entire graph on its own. This can effectively summarize existing graphs within a single node and allows for an easier realization of situations as shown in Figure 4.15.

The framework developed in this thesis allows the construction of graphs based on reusable modules. An application developer may group modules to form a more complex module, which in turn can be grouped with other modules to form a module with even more complexity.

Compositions and individual modules should ideally be treated in the same way, whereas a naive implementation might favor a definition of primitive modules and modules that act as containers. There are several problems with such an approach. Developers would have to handle containers and primitives in different ways even if their interfaces are almost identical (i.e. compositions and individual modules need to be distinguished). The application would gain

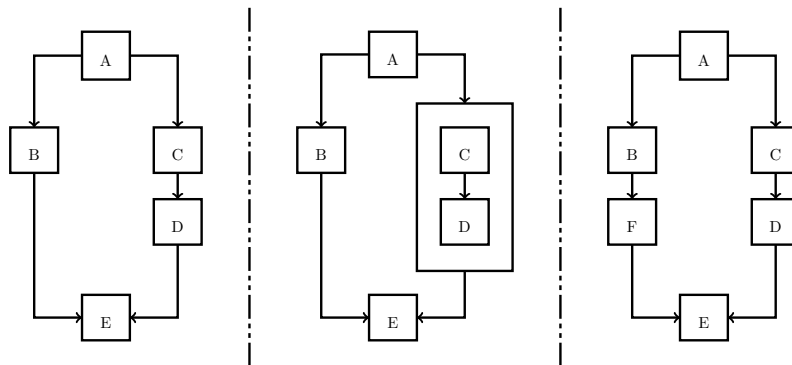


Figure 4.15: Illustrates the concept of layering graphs. Module “A” represents the entry point of the graph (i.e. data source). From there information is passed to module “B” and “C”. Continuing this path the information from “B” is going to reach “E” one iteration before the data from “D” would (left). This is due to the extra module and can be avoided (if wished) by grouping modules “C” and “D” into a single one (middle). Alternatively, an extra “dummy” module could be inserted between “B” and “E” (right).

unnecessary complexity and would increase development efforts as well as maintenance costs. The Composite design pattern [57, p. 163] can be used to avoid the described problems. It allows treating primitive components and compositions uniformly.

The following paragraphs are based on the definition of the Composite design pattern by Gamma et al. [57, p. 163] and describe its implementation in the framework. The Composite design pattern is an object-based structural pattern and can be used to represent part-whole hierarchies.

Structure: The structure of the Composite design pattern is illustrated in Figure 4.16. The basic idea is to create an abstract class that represents both primitive objects and containers and defines all child-related operations. Therefore, developers can use the interface and treat them in the same way without having to make a distinction.

Participants: The Composite design pattern has four participants, which are described hereafter.

- **Component:** Is the main class of this design pattern. The interface for managing and accessing child components is defined by this participant. It optionally declares the interface for accessing its parent component (if present). Furthermore, the class declares operations that are common to all components (compositions and primitive components) and provides default implementations where appropriate.
- **Leaf:** Is a primitive *Component* and defines the behavior of them in a composition. It does not have child components, thus representing a leaf in a tree structure.
- **Composite:** Is a composition of *Component* objects. It may have child

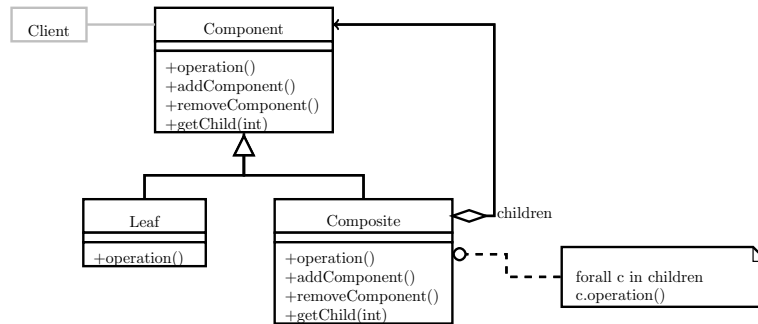


Figure 4.16: Illustrates the structure of the Composite design pattern as defined by [57, p. 163]. The generic *Component* interface is shown as well as *Leaf* and *Composite* elements (including their implementations).

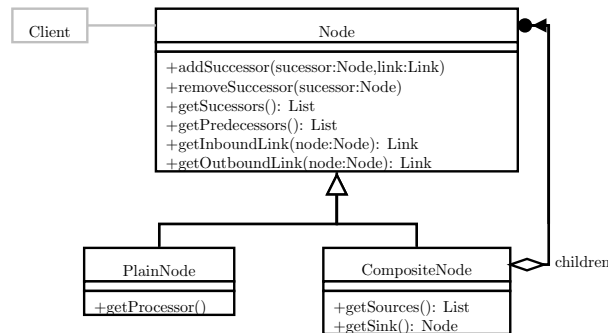


Figure 4.17: Shows the implementation of the Composite design pattern. It supports treating compositions (e.g. class *CompositeNode*) as well as individual components (e.g. class *PlainNode*) in a homogeneous way.

components and represents a container object that provides an implementation for child-related operations.

- **Client:** Accesses and manipulates objects in the composition.

Collaborations: A request is handled directly if the recipient is a *Leaf* and usually forwarded to child components if the recipient is a *Composite*. In the latter case, additional operations may be performed before and after forwarding.

Implementation: An implementation of the Composite design pattern is shown in Figure 4.17. The naming of classes and operations has been modified to reflect the context in which they appear more appropriately.

The *Component*, *Leaf* and *Composite* classes are represented by the classes *Node*, *PlainNode* and *CompositeNode* respectively. The *Node* class defines operations for managing and accessing child components. Furthermore a method for accessing the parent components is provided. No further operation is required with respect to treating compositions and individual components uniformly.

Nonetheless it should be noted that this realization of the Composite design pattern does not only allow for the presence of an arbitrary number of

successor nodes. The conservative interpretation of the Composite design pattern only supports two kinds of basic elements (i.e. leaf elements that do not have successors and composite nodes which do have successors). Depending on its implementation, arbitrary complex graphs are possible. However, the liberal implementation in this framework extends the pattern by adding a further dimension. Thus allowing to layer graphs and structure them in a way that “hides” multiple modules behind a single node in an upper layer.

How can contents of compositions be accessed sequentially?

As previously shown, the framework does enable the construction of complex aggregate structures. However, it is possible that future versions of the framework may utilize a different internal representation of these aggregates. Thus the actual internal representation of compositions and aggregates is subject to change. Even if such changes would occur, it is desirable to have a uniform way of traversing such compositions without exposing their underlying representation (i.e. a unified way of accessing modules within a given structure without actually knowing what the actual graph looks like). Thus allowing clients and large parts of the framework to remain untouched if such an event were to occur.

It should be kept in mind that different traversal strategies may be required for different scenarios (e.g. development vs. production environments, preserving an aggregate’s structure vs. processing their data). Nonetheless, it is certainly desirable for all traversal strategies to adhere to a uniform interface. In addition, it may be necessary to traverse over a composition in parallel (i.e. iterate over a composition multiple times at the same time). A naive implementation could make use of the *Component* interface and manually iterate over the child components. However, this solution produces unnecessary source code because it ignores the fact that the same type of iteration might be used multiple times throughout the framework. Thus, it causes redundancy, it is more likely to be error-prone and it is less flexible.

The Iterator design pattern [57, p. 257] is used to accomplish this requirement. It does not clutter up the components interface. Instead, it provides a unified interface for accessing children of compositions and encapsulates the iteration strategy. An Iterator object has to keep track of visited elements and those that are still to be visited.

Alternatively, the Visitor design pattern [57, p. 331] could have been used to traverse aggregate structures. This pattern is designed to easily facilitate the realization of (multiple) operations on aggregate structures. However the Iterator design pattern is used instead because it favors a loose coupling and allows a seamless integration of new components. In contrast, the Visitor design pattern is less flexible with regard to extensibility of the framework. Adding a new component would imply the modification of the Visitor interface and the adjustment of classes implementing that interface.

The following paragraphs are based on the definition of the Iterator design pattern by Gamma et al. [57, p. 257] and highlight several properties of it. The Iterator design pattern is an object-based behavioral pattern. It provides sequential access to elements within a composition without revealing the underlying representation.

Structure: The structure of the Iterator pattern is illustrated in Figure 4.18. The basic idea is to encapsulate the iteration process and provide a uniform interface.

Participants: The Iterator design pattern has four participants which are described in the following.

- **Iterator:** Is the main class of this design pattern. It declares operations for accessing and enumerating elements in a composition. The interface is used for all traversal strategies.
- **ConcreteIterator:** Is an *Iterator* for a concrete implementation of a traversal strategy. A *ConcreteIterator* holds knowledge on the ordering of the elements being traversed. It is responsible for managing the current position and choosing the next element.
- **Aggregate:** Is an aggregate that is supposed to be traversed. It declares an operation for generating an appropriate *Iterator* object.
- **ConcreteAggregate:** Is an implementation of the *Aggregate* interface. It is responsible for creating appropriate *ConcreteIterator* objects when requested.

Collaborations: A *ConcreteAggregate* class is an *Aggregate* and creates *ConcreteIterator* objects. A *ConcreteIterator* class implements the *Iterator* interface and traverses its *ConcreteAggregate*. It remembers the current position in the aggregate and knows how to determine the next element.

Implementation: An implementation of the Iterator pattern is shown in Figure 4.19. The interfaces of the *Iterator* design pattern have been adapted according to the context of this design challenge. Here, the aggregate structures do not suggest a concrete *Iterator*. This is because the same aggregate may be traversed with various different traversal strategies (depending on the context). They are simply unknown at this point and need to be dynamically instantiated at run-time.

The *Iterator* interface defines three operations. The *hasNext* operation can be used to determine whether there is at least one more element in the underlying aggregate. The next element in the traversal is returned by the *next* operation. The *remove* operation can be used to remove the last element from the underlying aggregate returned by the *next* operation.

InfiniteLevelOrder and *OneShotLevelOrder* are concrete implementations of the *Iterator* interface. The *InfiniteLevelOrder* class follows an iterative deepening approach and will never stop (i.e. *hasNext* will always return true). In the first iteration, it will visit only root elements in the graph (i.e. elements without preceding modules; elements without parents). In the second iteration, it will visit those elements that are on level one and zero (i.e. depth equals zero and one). The third iteration will visit elements on the first three levels (i.e. depth ≤ 3). As for the *OneShotLevelOrder*, it propagates a set of information through the entire graph once (from data source to sink). Both of these approaches are illustrated in Figure 4.20.

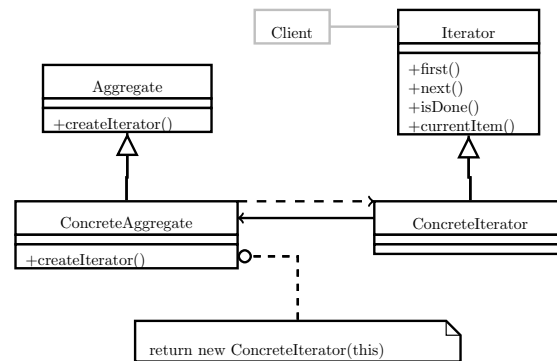


Figure 4.18: Defines the structure of the Iterator design pattern [57, p. 257]. The *Aggregate* and *Iterator* interfaces are shown as well as specializations of both.

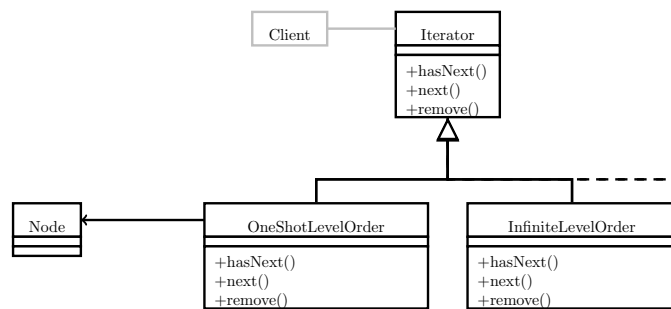


Figure 4.19: Illustrates the implementation of the Iterator design pattern. Furthermore, the adapted *Iterator* and *Node* (former *Aggregate*) interfaces are shown. Concrete implementations are presented as well.

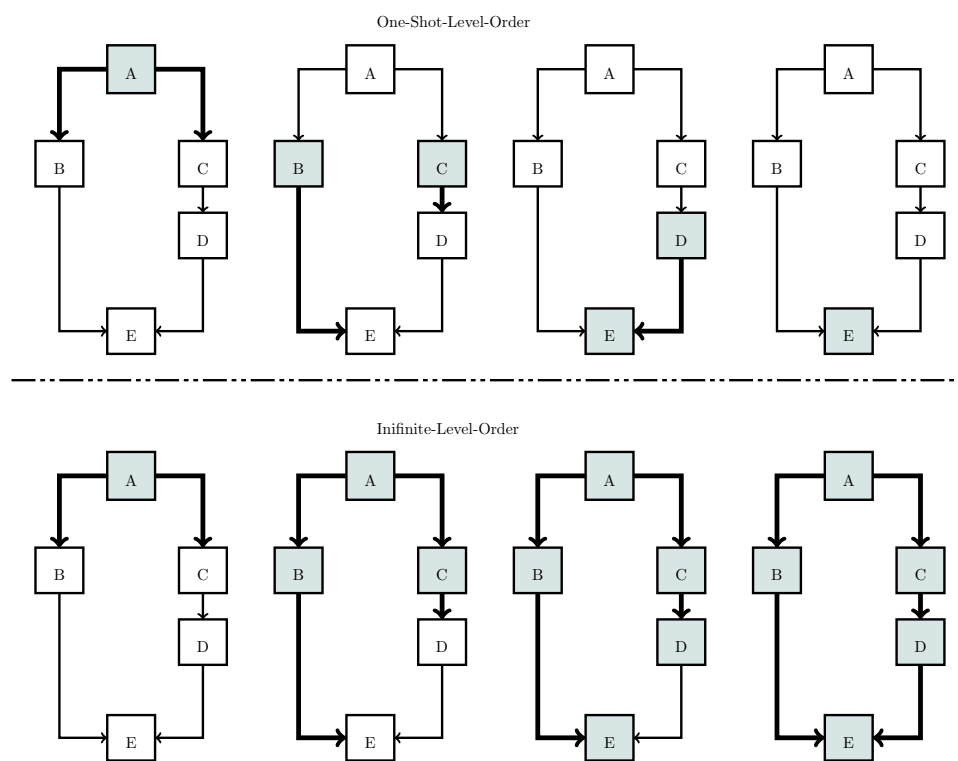


Figure 4.20: Shows realizations of the Iterator design pattern: *InfiniteLevel-Order*(bottom) and *OneShotLevelOrder*(top).

How does information travel in-between modules / nodes?

It has already been mentioned several times, that output data from modules is passed to their corresponding successor modules. However, it was not mentioned how this process is implemented in the framework. An easy and quick solution to this problem would be that the framework passes data to the next module(s) as soon as it is available. While this approach would certainly work in situations where modules form a tree-like structure, it does not work in situations where a graph structure is utilized, because multiple input sources and their synchronization are not taken into account. Furthermore such an approach is likely to be infeasible and does not allow for much variations, if required from users of the framework. On the other hand, an alternative approach would be to buffer data that is in transit between two modules. This approach would go well with the previously described Iterator design pattern because it allows to process each module individually and helps to gather all data from multiple sources before a module is processed. This buffering can easily be realized in a flexible and extensible manner if the Strategy design pattern is employed. It would not just allow transport of data but also to choose which data are forwarded and which are not.

Design Pattern: The Strategy design pattern has already been described earlier in this chapter. For more details on the subject refer to “Which functionality must a module provide?” (see Section 4.1.2).

Implementation: Figure 4.21 shows the implementation of this design pattern. There were no major changes required.

The *Link* interface represents the former *Strategy* class. It contains methods for adding and removing data (i.e. *push* and *poll*) from the buffer as well as a method for checking whether the buffer is empty (*isEmpty*). The *isEmpty* method returns an indication of whether there is no data in the buffer. The *push* method is utilized by the framework to fill the buffer with data. Similarly, the *poll* method will return the next data point in the buffer, if present. Additionally, the *Link* interface is implemented in such a way that it can be configured by implementing the *Configurable* interface previously described (see “How can modules be configured?” in Section 4.1.2). This adds flexibility to the responsibilities of the *Link* implementation.

These *Link* instances are meant to be placed between exactly two modules (thus it represents exactly one connection). Once the results of a module are determined, the framework will attempt to place them in all succeeding *Link* instances (i.e. provided that the *Link* accepts the given data). Later the data are read from all incoming connections and passed into the corresponding module. The output data are again stored in the succeeding *Link* instances.

How are functions across the (entire) graph implemented (i.e. processing of data or serialization of graph)?

The framework defines operations based on a set of modules. These include but are not limited to: saving a graph of modules to a file, analysis of graph for unreachable nodes and other unwanted side effects. Other functions are for invoking individual modules, links, and nodes that make up the actual graph.

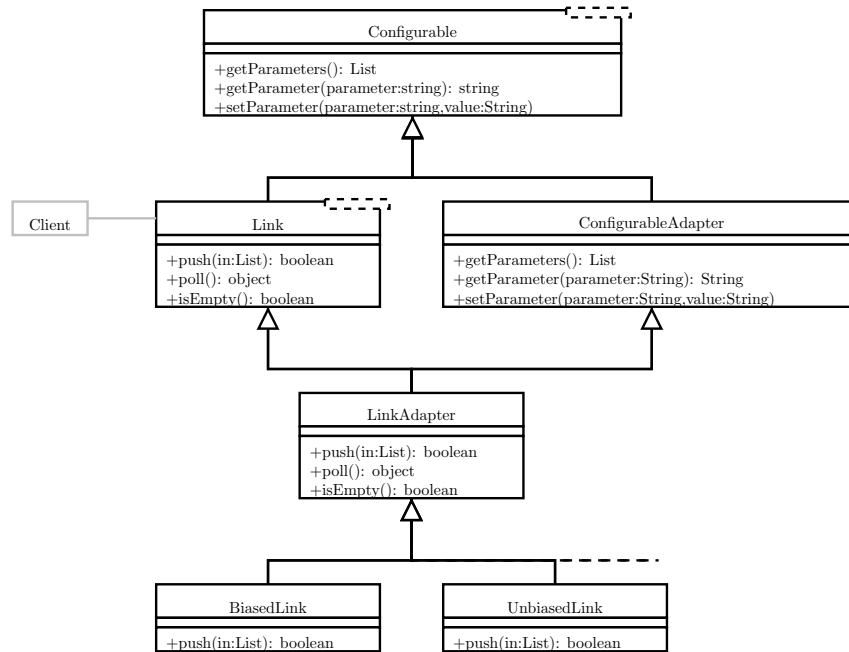


Figure 4.21: Shows the implementation of the Strategy design pattern for linking modules together. It illustrates the relationship among *Link*, *Configurable* and concrete implementations.

However, none of the related classes require knowledge about these functions. In fact, their interfaces would rapidly clutter up if these functions were implemented in a naive way. Additionally, future generations of the framework are very likely to include even more functions of this sort.

It is desirable to encapsulate this functionality in a way that does not affect modules, links and nodes alike. Ideally, new functionalities should be addable without the need for any changes on side of the framework. The same goes the other way around; adding new modules should not affect functionalities in any way. These requirements can be met by utilizing the Visitor design pattern. However, it should be pointed out that adding further node-like classes does have a negative impact on existing functionalities and requires modifications of existing functions that were implemented in this way. However, it will very rarely be necessary (if at all) to add another node-like class to the mix. This structure will practically never change.

The Visitor design pattern represents an object-based behavioral design pattern. It encapsulates an operation that is to be executed over an aggregate or composition. Thus allowing modifying and adding operations without affecting the corresponding aggregate and composition. The definition of this pattern can be reviewed in [57, p. 366].

Structure: The structure of the Visitor pattern is illustrated in Figure 4.22. The general idea is to implement operations over an object-structure of individual objects with an identical interface. The object-structure is not affected when new functions are added.

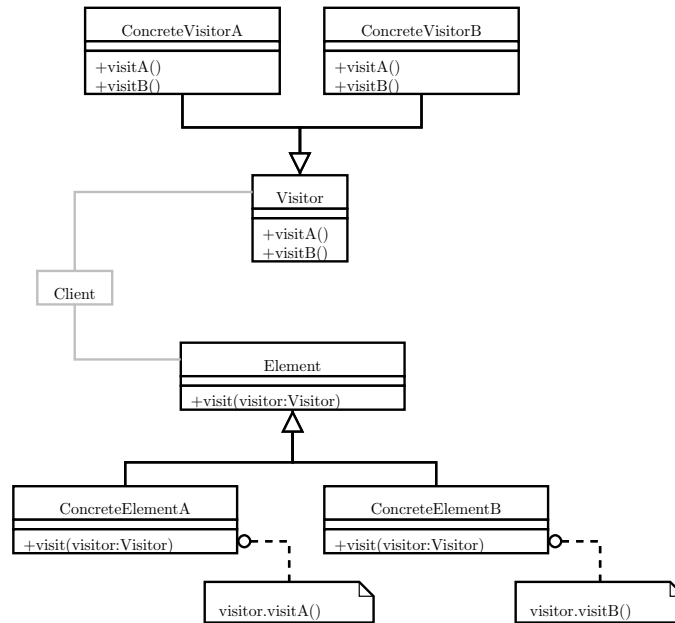


Figure 4.22: Illustrates the structure of the Visitor design pattern as defined by [57, p. 366]. Hierarchies for *Visitor* and *Element* classes as well as their interaction are shown.

Participants: The Visitor design pattern has four participants which are described in the following.

- **Element:** Is the main class of the object-structure. It defines a *visit* operation which takes a *Visitor* as argument.
- **ConcreteElement classes:** These are realizations of the *Element* interface. Thus they implement the *visit* method and call their corresponding “visit”-method of the given *Visitor* instance.
- **Visitor:** Is the main class of the Visitor design pattern. It defines an interface which includes a “visit”-method for each *Element* class. Name and signature of each method are tied to a specific *ConcreteElement* class. This enables a *Visitor* to determine the actual type of objects within the object-structure and allows accessing the *ConcreteElement*’s interface (instead of the *Element*’s interface only).
- **ConcreteVisitor classes:** They implement the *Visitor* interface and provide some functionality based on the object-structure. Here each interface method realizes its part of the overall operation. A *ConcreteVisitor* will likely store some kind of context (i.e. results of preceding *Elements*) which is utilized by the algorithm.

Collaborations: *Visitors* are typically used in conjunction with the previously described Iterator design pattern. Latter guides a *Visitor* through the object-structure in a pre-determined way. The *Visitor* is passed to each *Element* in the object-structure. Each *ConcreteElement* passes itself to the appropriate “visit”-method once visited by a *Visitor*.

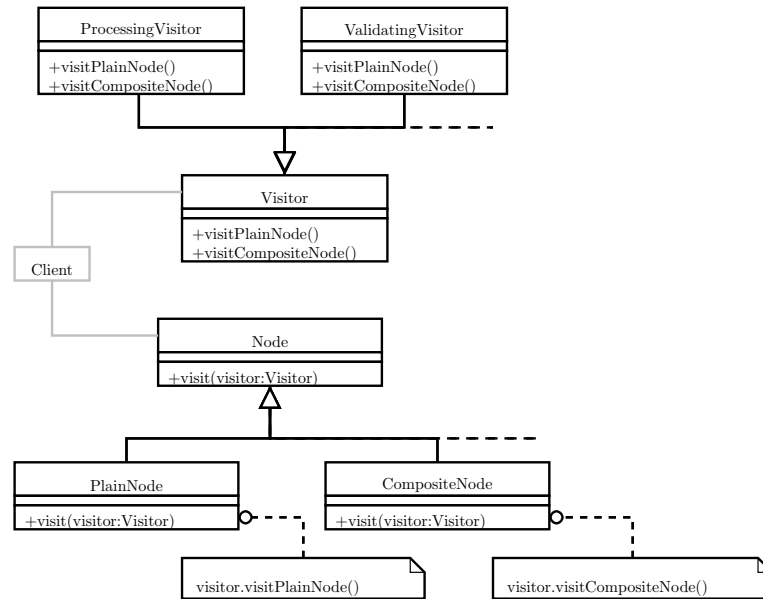


Figure 4.23: Shows the implementation of the Visitor design pattern as part of the framework.

Implementation: The implementation of the Visitor design pattern can be seen in Figure 4.23. No major adoptions were required. However, the naming conventions represented by the *Node* class and its descendants (*PlainNode* and *CompositeNode*) were adapted to reflect the context of their usage. This object-structure is not expected to change, if at all. Thus making a good fit for the Visitor design pattern.

The *Visitor* interface contains two methods to be implemented by any visitor. Both methods *visitPlainNode* and *visitCompositeNode* take a single argument which resembles the actual node that is currently being visited. Developers can safely assume that this parameter will always be non-empty (i.e. non-null). Furthermore, the *Node* class was adjusted to include another method called *visit*, which takes a *Visitor* object as argument and calls the corresponding *visit* method.

Even though, a *Visitor* could also take care of traversing the object-structure, this has been avoided as it would likely create code redundancies. Instead the Iterator design pattern is utilized in conjunction with the Visitor design pattern.

4.1.4 Data Sources and Sinks

This section discusses matters related to the third design challenge (i.e. Data Sources and Sinks). Several related requirements (see Section 1.2) are picked-up in the course of this section. The most involved ones are highlighted hereafter.

- **Stream-based:** This part of the framework must be designed in such a way that it enables a stream-based processing of data samples. The reading and writing mechanisms should work with a single data sample at a time.

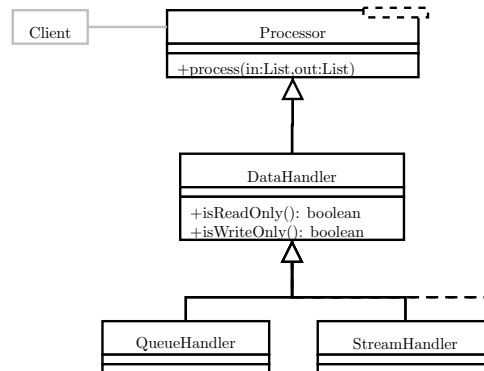


Figure 4.24: Shows the implementation of the Strategy design pattern for reading and writing arbitrary data.

- **Scalability:** These mechanisms must also be able to handle varying sizes of datasets. Smaller datasets (i.e. less than 10^4 data samples) as well as large datasets (i.e. greater than 10^6 data samples) should be supported.
- **Reusability:** The components in this part of the framework should be designed to be reusable. The interfaces and classes for accessing data should be utilized multiple times.
- **Extensibility:** It should be easy to add support for further data formats as well as data storage places.
- **Support for distribution:** The components should be designed to enable inter-process and inter-machine communications. There should be the option to utilize network-based communication protocols.

Where does data come from and go to?

The whole framework would not make much sense without an easy possibility to utilize existing data sources (i.e. files, network, etc.). Thus, the framework provides the means to read and write arbitrary data. It should enable developers to access available data files, network-based data sources and so on. However, there should be no tight coupling between the framework and its data sources or data sinks. Instead, they should be an integral part of it (i.e. they should be a regular module). This way data sources and sinks can be treated uniformly with any other module in the framework. In order to accomplish this, the Strategy design pattern is employed to avoid unnecessary redundancies in the source code.

Design Pattern: The Strategy design pattern has been discussed previously. Refer to “Which functionality must a module provide?” (see Section 4.1.2) for more details on its structure, participants and so on.

Implementation: Figure 4.24 shows the Strategy design pattern and the way it is implemented in the framework. There were no noteworthy adjustments required.

The *Strategy* class is represented by the *DataHandler* class. It resembles a typical module, but already provides commonly utilized configuration parameters. These can be used to make the *DataHandler* read-only, write-only as well as readable and writable. Additionally, it provides a minimal basis for those modules that want to read / write data.

It should be noted that this *DataHandler* class merely provides an abstract and very basic view on the topic in question. In most cases, an existing implementation of the interface might be more preferable than starting from scratch. In particular, refer to “Can data format and kind of stream vary independently?” (see Section 4.1.4) if stream-based resources (i.e. files, network, etc.) are involved.

Can data format and kind of stream vary independently?

The data that is being processed in the framework may stem from a variety of places and may be present in even more data formats. Naturally, the framework should support a set of commonly employed data formats (e.g. CSV, ARFF, JSON, etc.) as well as numerous data storage places (e.g. files, HTTP, FTP, etc.). A naive implementation would create a module for each combination of data format and place of origin. Only these few examples already add up to nine combinations (i.e. CSV+file, CSV+HTTP, CSV+FTP, ARFF+file, ARFF+HTTP, etc.). If compression variants are included (e.g. zip and tar) then this number grows even larger (i.e. 27 different combinations). It is obvious that such a number of combinations cannot be easily maintained if implemented in individual modules. It should also be mentioned that this number is very likely to increase even more as data formats and places grow. Especially for the latter two reasons, the manual and naive way cannot be recommended.

It is desirable to vary data formats and data storage places independently, such that the number of components becomes manageable. In principle data formats are not bound to the way they are stored (i.e. in file, database or network). Ideally, each data format should be usable with any data storage mechanism. This would drastically reduce code redundancies and increase reusability. At run-time the sought combination and place could be created dynamically. It should also be possible to add a new format without changing any data storage or data retrieval mechanism. The same goes the other way around. All of these expectations can be met with the Bridge design pattern.

The Bridge design pattern represents an object-based structural design pattern. The general idea is to decouple an abstraction from its implementation and thus allowing them to vary independently. This pattern can be reviewed in [57, p. 171].

Structure: The structure of the Bridge pattern is illustrated in Figure 4.25. The overall structure separates an abstraction from its implementation into two objects, which can vary independently of each other.

Participants: The Bridge design pattern has four participants, which are highlighted hereafter.

- **Abstraction:** Is one of the two main classes in this design pattern. This class represents an abstraction which references an *Implementation* in-

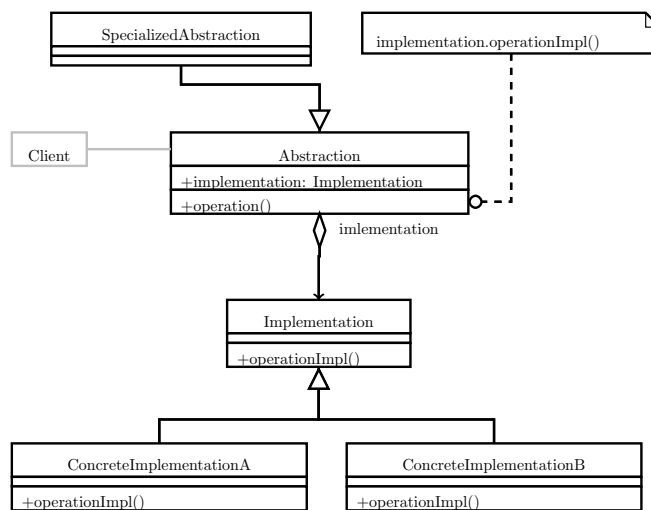


Figure 4.25: Illustrates the structure of the Bridge design pattern as defined by [57, p. 171].

stance. The interface of this class resembles that of the *Implementation* class but usually contains a few more methods. Whenever appropriate the referenced implementation is called.

- **SpecializedAbstraction classes:** These classes are specializations of the *Abstraction* class. They typically specialize one of the additional methods that the *Abstraction* class provides.
- **Implementation:** This is the second main class in the design pattern. It defines an interface for the implementation. This interface, however, is typically somewhat resembled in the *Abstraction* class.
- **ConcreteImplementation classes:** These classes extend or implement the *Implementation* class.

Collaborations: The *Abstraction* class references an instance of the *Implementation* class. The *Implementation* does not know anything about the *Abstraction*. Where appropriate the *Abstraction* class calls the *Implementation* that it references.

Implementation: The implementation of the Bridge design pattern is shown in Figure 4.26. Apart from name changes, no adjustments of the design pattern were required.

The *Abstraction* and *Implementation* classes are represented by *StreamHandler* and *StreamHandlerImpl* respectively. On the one hand, the *StreamHandler* takes care of parsing and generating a particular data format (e.g. XML, JSON, etc.). On the other hand, the *StreamHandlerImpl* defines an interface for accessing some raw data through a stream (e.g. file, web, etc.). This *StreamHandlerImpl* is referenced by *StreamHandler* and called when data are to be read from somewhere or saved to somewhere.

The *StreamHandler* class defines two methods for accessing raw data. The

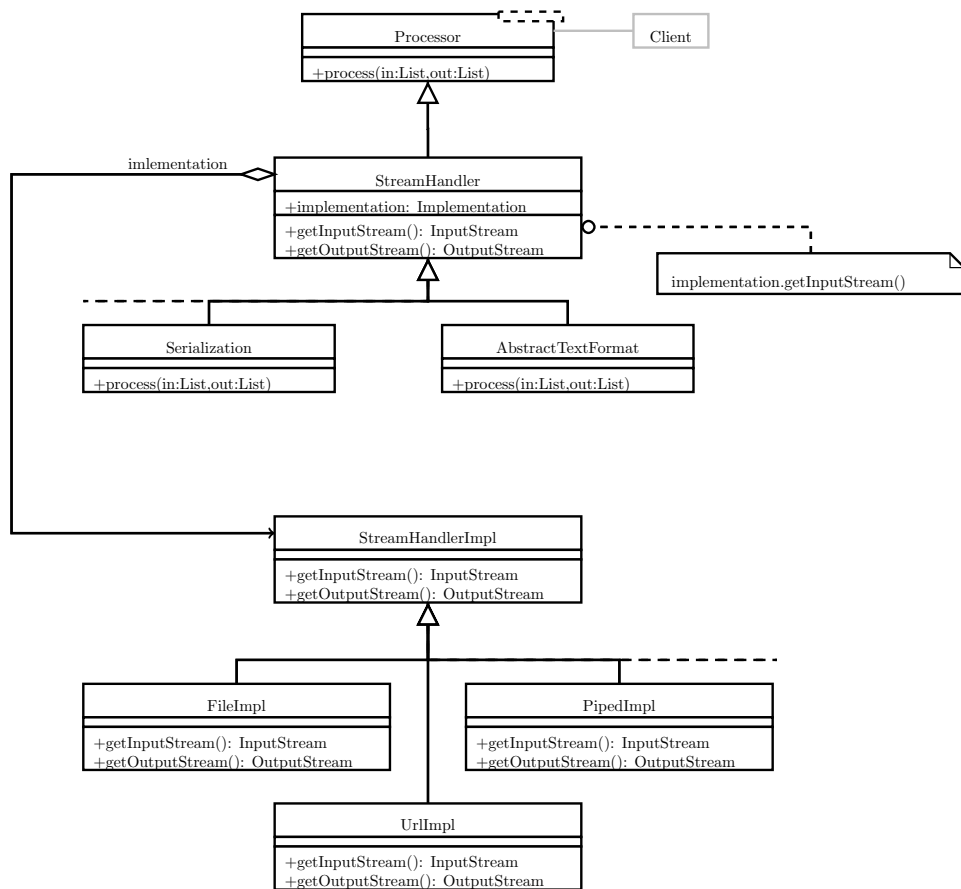


Figure 4.26: Shows the implementation of the Bridge design pattern.

operations *getInputStream* and *getOutputStream* are meant to provide access to a stream-based resource every time they are called. Respectively, *getInputStream* provides read access while *getOutputStream* provides write access to the same resource. Depending on the concrete implementation of *StreamHandlerImpl* the utilized resource will change (i.e. whether it is a file on the local computer or somewhere on the network). As for the *StreamHandler* class, it references a concrete implementation of *StreamHandlerImpl*. It also contains methods *getInputStream* and *getOutputStream* which utilize the methods on the *StreamHandlerImpl* interface. However, the actual streams are buffered to refrain from opening unnecessarily many streams. Specialized abstractions, such as *Serialization* or *AbstractTextFormat*, utilize these methods during parsing (i.e. *process* method). Here input and output streams are utilized to read or write a particular data format.

4.1.5 Summary (of Used Design Patterns)

In the course of this chapter, several aspects have been described which are related to the development of a framework for time series analysis. Numerous design challenges were discussed and their solutions highlighted.

The following list represents design patterns that were utilized or adapted in the framework to solve the problems at hand. Most of these can be reviewed in [57]. Both, design patterns and design challenges they solved, are highlighted hereafter in order of appearance.

- **Strategy:** Defines the basis for a family of algorithms or functions. All descendants in the family can then be treated in a unified way, they are interchangeable. This design pattern is used numerous times in the framework. For details refer to “Which functionality must a module provide?” (see Section 4.1.2), “How can modules be configured?” (see Section 4.1.2), “How can parameters be constrained or conditioned?” (see Section 4.1.2), “How does information travel in-between modules / nodes?” (see Section 4.1.3) and “Where does data come from and go to?” (see Section 4.1.4).
- **Observer:** Defines a one-to-many dependency between objects such that changes in one object are automatically reflected in other objects. This design pattern is primarily used to express relations among configuration parameters and to react to changes in configuration parameters. For details refer to “How are dependencies and relations between parameters expressed?” (see Section 4.1.2).
- **Adapter:** Allows multiple classes to work together, that are otherwise incompatible. The interface is adapted to allow their cooperation. The framework utilizes this design pattern in “How do these objects (i.e. Configurable, Condition, Observer) work together?” (see Section 4.1.2).
- **Template Method:** Defines a structure of an algorithm or function and defers steps of it to sub-classes. This enables sub-classes to change the algorithm’s behavior without changing its overall structure. This design pattern is used in “How can typed parameters (i.e. not just strings) be supported?” (see Section 4.1.2).
- **Decorator:** Enables a dynamic extension of an object’s responsibilities without the need for sub-classing it. The object in question is wrapped.

The framework employs this design pattern in “Can a module’s functionality be dynamically extended?” (see Section 4.1.2).

- **Composite:** Defines a structure for part-whole-hierarchies that allow clients to treat compositions and individual objects in a unified way. The framework utilizes this design pattern in order to avoid unnecessary complexity in graph-like structures of the modules. The interested reader is directed to “How can compositions and individual objects be treated in a unified way?” (see Section 4.1.3) for more details.
- **Iterator:** Provides sequential access to components within a composition without the need for details on the underlying structure. This design pattern is employed for accessing all modules within a graph (i.e. for computation, etc.). For details refer to “How can contents of compositions be accessed sequentially?” (see Section 4.1.3).
- **Visitor:** Encapsulates an operation across a composition of elements in an object. This allows creating new operations without changing the composition’s structure. This design pattern is used in “How are functions across the (entire) graph implemented (i.e. processing of data or serialization of graph)?” (see Section 4.1.3).
- **Bridge:** Allows the independent variation of an abstraction and its implementations. The framework makes use of this design pattern in “Can data format and kind of stream vary independently?” (see Section 4.1.4).

As noted at the beginning of this section, diagrams were kept as compact as possible, thus only necessary operations and interfaces are included. The idea was to provide simple diagrams in order to facilitate an easier understanding of particular design challenges. More details can be found in the framework’s implementation itself [108].

4.2 Extensibility

The framework is not complete as there will be the need to add further functionalities (e.g. ML, AI and data processing algorithms) at some point in time. Thus, providing the means to easily add new modules as well as other components is essential. Developers can make use of this option to implement their domain-specific functionalities and thus contribute to the framework. The remainder of this section describes various ways in which the framework can be extended. A concise summary of these possibilities is given hereafter.

- **Adding Modules:** One of the most obvious and most frequently performed use-cases (with respect to extending the framework) is the addition of modules. This is where filters as well as all sorts of ML and AI algorithms can be implemented.
- **Wrapping and Decorating Modules:** A likely rarely performed use-case is the addition of wrapping and decorating modules. This type of module will be added whenever a new functionality, which could in principle be applied to any other module (e.g. logging of input and output data, measuring frequency of input and output values or measuring the processing times of a module) is added to the framework.
- **Data Sources and Sinks:** Describes a use-case in which a new data format or a new way of retrieving and storing data is added to the framework.

This is expected to be performed occasionally. Support for proprietary data formats and the like can be added here.

- **Links:** Another occasionally performed use-case is the addition of new links (i.e. the means of data transportation between two modules). This can be used to implement specific data buffering behaviors, which are not covered by the default implementation in the framework.
- **Functions Across the Entire Graph:** This represents a use-case that is expected to be moderately often utilized. It allows adding new analysis mechanisms as well as general functionalities or operations that operate on a graph of modules (e.g. determining root or leaf nodes, checking for loops, etc.).
- **Alternative Traversal Methods:** One of the most infrequently performed used options is the addition of new traversal methods to the framework. These traversal methods are usually used in conjunction with methods described in “Functions Across the Entire Graph”. If the default traversal methods do not provide access in the expected way or order, then this is where new orderings can be realized.

4.2.1 Adding Modules

Probably the most obvious way of extending the framework is adding new data processing modules as needed. Here “data processing module” refers to any algorithm or functionality that is capable of generating data, manipulating data, transforming data or absorbing data. This can range from simple moving average computations over function generation to writing data in a DB. It is very likely that the necessity for adding new modules will arise at some point in the future.

In general it is sufficient to implement the *Processor* interface. However, in most cases it will suffice to actually extend *ProcessorAdapter*. Latter suggestion already provides a partial implementation of the *Processor* interface. In particular, parameterization (i.e. getting and setting parameters) is already taken care of. Unless it is necessary to create an alternative implementation for manipulating parameters, latter option (i.e. extending *ProcessorAdapter*) should be preferred. It is the easiest way to add a new module, because it only requires to implement a single method called *process*.

The *process* method does not directly provide a set of return values. Instead, it accepts a set of input values and manipulates a set of output values. The entity that calls the *process* method ensures the availability of input values and an empty set of output values. The module itself has the flexibility to ignore input values and is not required to provide any output values. If a module does choose to ignore input values, it still has the option to generate output values. This can be useful for generating data (e.g. reading from a file, querying a database or generating function values) which can then be used by other modules. If, on the other hand, a module decides to process input values, it should not manipulate them as this can have negative effects on other modules. This also includes adding and removing from the set of input values. However, the module can still decide to provide output values but is not required to do so. Latter case would be useful when writing the data to a file or database. On the other hand, the former case (processing input data and providing output data) would be

considered the normal case. An example for such a behavior is demonstrated in Listing B.2. It highlights a module that forwards its input values, which can most easily be done by extending *ProcessorAdapter*.

Additionally there is always the option to extend any existing module. This is particularly useful if a module already behaves in almost the expected way. Regardless, whether *ProcessorAdapter* or any other existing module is used as template, the *process* method must execute as fast as possible. The method may be called several thousand times per seconds (or more). Thus long running tasks should be avoided.

Related design challenges:

- **Section 4.1.2** Which functionality must a module provide?
- **Section 4.1.2** How do these objects (i.e. Configurable, Condition, Observer) work together?

4.2.2 Wrapping and Decorating Modules

Certain situations may require a dynamic analysis of data passing through a module, while other situations may require dynamic extension of the functionality or responsibility of some module. In all of these cases it is useful to create a module that is capable of wrapping (or decorating) an existing module. Concrete examples are modules that log input and output values or simply ensure the validity of input data before they are being passed into the actual module in order to avoid faulty behavior of the wrapped module. It should be stressed that this kind of module can be combined or can be used with any other module in the framework. This includes other wrapping and decorating modules as well.

The framework already provides the means to create such modules. Although they can be implemented by hand (see “Adding Modules”), the easiest way is to extend the *DecoratorProcessor* class. In doing so, developers are able to perform any extra operations before and after the wrapped module is called. The *DecoratorProcessor* itself implements the *Processor* interface by simply forwarding all interface calls to a wrapped module. Some decorating modules may even want to decide not to call the wrapped module (this only makes sense for a selected few methods).

An example, that demonstrates the addition of a decorating module, is shown in Listing B.3. It can be used to generate statistics on the processing time that a module requires.

Related design challenges:

- **Section 4.1.2** Which functionality must a module provide?
- **Section 4.1.2** Can a module’s functionality be dynamically extended?

4.2.3 Data Sources and Sinks

Support for new data formats, data retrieval and data storage mechanisms can be added to the framework. The framework supports a set of commonly utilized data formats and the like, but further proprietary data formats or domain-specific data storage places can be added.

The *StreamHandler* and *StreamHandlerImpl* classes can be utilized for these purposes. The *StreamHandler* class is meant to enable developers to realize custom data formats. Overwriting the *process* method and accessing the raw data via *getInputStream* and *getOutputStream* methods allows this. On the other hand, the *StreamHandlerImpl* class allows developers to implement new data retrieval and data storage mechanisms. Here, the *getInputStream* and *getOutputStream* methods need to be implemented. They can be utilized to provide access to information on a website, in files and the like.

In cases where no streaming is involved, the *DataHandler* class can be utilized as a template. Here, the *process* method is implemented to read data from some data source (i.e. other than an input stream; e.g. a queue or list) or write to a data sink (i.e. other than an output stream).

An example for implementing raw access to a file is shown in Listing B.4. The realization is done by implementing the *StreamHandlerImpl* interface. Another example is listed in Listing B.5 where the *DataHandler* class is extended to utilize a queue.

Related design challenges:

- **Section 4.1.4** Can data format and kind of stream vary independently?
- **Section 4.1.4** Where does data come from and go to?

4.2.4 Links

Links represent directional connections between pairs of modules, thus modeling the flow of information from one module to the next.

It is their purpose to accept data from one module and provide it to another module when requested. However, there are situations in which certain data must not be transferred from one module to the next. This might be the case when a module produces multiple kinds of output values (e.g. numbers and strings) while an adjacent module can only handle numeric values. Here, a link could take the responsibility of filtering out unwanted types of values. Of course an alternative approach would be to implement this sort of filtering in a separate module (see “Adding Modules”). However, a mechanism that transports data is required nonetheless (i.e. it cannot be done by modules alone). Thus, it does not really matter and the decision is left to the developer’s preferences.

Even though several common types of links are already implemented in the framework, there may be the need to add new ones in the future, which cover special use-cases. If such a need arises then the implementation is most easily done by extending the *LinkAdapter* class or any other concrete *Link* implementation. Of course, it is sufficient to implement the *Link* interface but this requires more effort. In almost all cases, it will suffice to specialize either the *push* or the *poll* method of the *LinkAdapter*. This would be the case if the *Link* is supposed to perform more than simple data validation. Thus requiring the developer to implement the *push* and *poll* methods. They are utilized by the framework to push any given output data from a module into an intermediate buffer and to poll this data from the buffer.

An example for implementing the *push* method is shown in Listing B.6. It demonstrates the case where one would want to filter the flow of data between

two modules, then this is most easily done by extending the *LinkAdapter* class and specializing the *push* behavior.

Related design challenges:

- **Section 4.1.3** How does information travel in-between modules / nodes?
- **Section 4.1.2** How can modules be configured?

4.2.5 Functions Across the Entire Graph

From time to time, it may be beneficial to add an analysis mechanism or function that operates on a graph of modules. Examples of this functionality range from simply saving the graph to a file, determining leaf and root nodes to ensure that all modules are potentially reachable (i.e. they can potentially be fed with data). The framework provides the means to realize such functionalities via the *Visitor* interface.

The general idea is that an *Iterator* is utilized (by the framework) to visit all nodes within a graph in an orderly fashion. For each type of node, the *Visitor* interface provides a corresponding method (e.g. *visitPlainNode* and *visitCompositeNode*). Both of which can be freely implemented. When visiting a node, a method that corresponds to the type of node (i.e. *PlainNode* and *CompositeNode*), is called on the *Visitor* interface. Functions of this sort may have certain assumptions on the ordering of modules. Thus one should consider available *Iterator* implementations and check whether they already fulfill the requirements. If not, it may be necessary to implement a corresponding *Iterator* as well.

For example if one wanted to initialize all modules within a graph, this functionality is easily created by implementing the *Visitor* interface. The Listing B.7 shows a *Visitor* that visits all modules exactly once and calls there *setUp* method.

Related design challenges:

- **Section 4.1.3** How are functions across the (entire) graph implemented (i.e. processing of data or serialization of graph)?
- **Section 4.1.3** How can contents of compositions be accessed sequentially?

4.2.6 Alternative Traversal Methods

Considering the various analysis techniques and endless possibilities of transformations of a module-graph, it becomes apparent that certain techniques require certain traversal methods. Thus, a specific traversal method is usually dependent on the kind of function that is to be performed on the graph. Of course, the same applies the other way around. Functions will likely produce different results depending on the chosen traversal method. Whenever another analysis technique or function is added to the framework, it may be required to add a new traversal method as well. This can be most easily done by implementing the *Iterator* interface or by sub-classing an existing *Iterator* implementation.

The main methods of this interface are *hasNext* and *next*. They are utilized to tell the framework that there are still more items which can be traversed (i.e.

hasNext method) and provides the means to access these items in a sequential fashion (i.e. *next* method).

An example of an *Iterator* implementation is shown in Listing B.8. The shown example provides sequential access to a graph of modules in such a way that the items are returned with respect to their depth within the graph (i.e. shortest distance from a root). It is a level ordered *Iterator*.

Related design challenges:

- **Section 4.1.3** How can contents of compositions be accessed sequentially?
- **Section 4.1.3** How are functions across the (entire) graph implemented (i.e. processing of data or serialization of graph)?

4.3 Applications and Scenarios

Here, a few exemplary applications and use-case scenarios of the proposed framework are briefly outlined.

4.3.1 Recognizing PD Motor Symptoms

The described framework can be used as an ML and AI toolkit for recognition of patterns in time series. Such applications include classifying PD symptoms as well as decision support systems for PD care givers and doctors.

A similar system is described in “Indication of Parkinson’s Disease Motor Symptoms” (see Chapter 6). It details numerous algorithms for recognizing PD motor symptoms (i.e. tremor at rest, dyskinesia and freezing of gait (FoG)). These algorithms can be broken down into smaller pieces which can then be individually implemented in the framework. In such a scenario, modules can be utilized to perform preprocessing of raw sensor data as well as the actual recognition of symptoms. Assuming that recorded signals are organized in separate files for each patient and recording session, then one module could be implemented that looks up all files containing recorded signals and another module could load these files. Once the signals have been loaded into memory, further modules could perform resampling, feature extraction, performance evaluation and other tasks. These modules can then be connected to form a system such as the one described in Chapter 6.

Some examples for such modules are described in the appendix (see Section B.3). This chapter also includes a road-map for implementing the algorithms described in Chapter 6 (see Section B.4).

4.3.2 Generating Trading Decisions

Autonomous trading systems and financial data analysis systems typically utilize time series data on foreign exchange currencies, bonds, stocks, etc. Their goal is to optimize trading strategies in order to maximize profit returns and minimize losses (ideas for such systems can be found in [107, 109]). These systems provide a set of rules, which can be utilized to estimate future movement of some financial instrument with a certain probability and confidence. Such a system can also be realized with the framework described in this chapter.

Some modules can be utilized to read financial data (i.e. from file, web, stock exchange, broker, etc.), which can then be resampled and processed to evaluate trading strategies. Alternatively, it can also be used to predict future price changes and automatically trigger trading actions.

4.3.3 Analysis of Network Traffic

Another application of the framework could be the systematic analysis of network traffic. The framework is not only able to perform passive analysis, but can also be utilized to reproduce (parts of) network traffic or manipulate traffic as it is being processed. The overall idea would be to create an input and output module for the corresponding type of network traffic (i.e. user datagram protocol (UDP), transmission control protocol (TCP), etc.). Each data packet can then be passed as regular input value through the framework. Once that is possible, succeeding modules can analyze and manipulate the information in each data packet. In the end, the (possibly modified) data packet is sent to the intended recipient. This mechanism could be utilized for network traffic from and to a client.

4.3.4 Quality Control of OpenStreetMap-Data

The developed framework can also be utilized on large-scale datasets. Within this category resides OSM data from the entire planet. The data contains, among many other things, all recorded streets, intersections, parks, houses. Here the framework can be used to parse various OSM data formats, analyze and transform the data as well as write it again. An exemplary application could analyze OSM nodes and paths for an overall quality assessment on the utilized key-value pairs.

4.4 Included Modules and Algorithms

For a global overview on the included modules in the framework refer to “Modules of Framework” (see Section B.3) in the appendix. The reader interested may want to have a look at the entire Chapter B or visit the framework’s repository [108].

4.5 Summary

This chapter has provided an answer to the first research question of this thesis. A software framework for time series has been proposed. The original requirements were analyzed and grouped into design challenges which were then evaluated and solved on their own. These design challenges were further sub-divided into manageable parts which were individually solved and implemented. The implementation focused on software design patterns that facilitate reusability and maintainability.

Additionally, several important ways of extending the framework were deduced from the implementation and explicitly highlighted for easy reference. Furthermore, several applications and use-case scenarios were pointed out.

The interested reader may want to obtain a copy of the framework [108]. A short description of individual modules as well as a road-map for implementing the algorithms presented in Chapter 6 can be found in the appendix (see Section B.3 and Section B.4, respectively).

Chapter 5

Database: Patients and Their Symptoms

This chapter describes the acquisition of data as well as the contents of the database (DB). The DB was specifically designed for development of algorithms that are intended to indicate Parkinson’s Disease (PD) motor symptoms in non-laboratory conditions [144]. The summarized data are those that are utilized during the remainder of this thesis. Consequently, the data aided in the development and evaluation of all algorithms that are presented in Chapter 6.

5.1 Data Acquisition and Labeling

Data acquisition was not part of this thesis, instead the data has been provided to the author as part of the REMPARK project [133]. Sensors and their locations on the patient’s body were also fixed prior to this thesis.

5.1.1 Sensors

During data acquisition, three kinds of measurements were recorded and were stored for offline analysis. Two different sensor platforms were placed on the participant’s body (i.e. one on the wrist and one on the waist). Both devices were able to communicate with each other via Bluetooth. The primary objective of the wrist worn sensor platform was to ease the detection of tremor. The waist sensor platform is used to detect other gait related symptoms (e.g. dyskinesia and bradykinesia) [144].

The wrist-mounted platform recorded three-dimensional acceleration data from the left wrist. The position is reflected in Figure 5.1. In addition to the acceleration sensor, the platform contains a microprocessor as well as a communication unit. The components are utilized to capture acceleration data at 80 Hz and send them to the waist platform, where the data are further processed. In the data acquisition phase, data are recorded for offline analysis. During final trials, most of the processing could potentially be performed on the sensor platform itself (e.g. preprocessing for symptom recognition). All in all, the complete device (including the power supply) weights around 50g and is 70x42x13mm³ (i.e. length, width and height) in size [139, 144].

The waist measurement platform recorded three-dimensional acceleration data, three-dimensional orientation data and three-dimensional compass data. The device is placed near the anterior superior illiac spine (ASIS), which is also highlighted in Figure 5.1. Furthermore, a microprocessor, a data storage unit and a communication unit are part of the platform. They are utilized to capture the acceleration, orientation and compass data with a sampling frequency of 200 Hz. The communication unit receives data from the wrist sensor. During data acquisition, the signals are stored for offline analysis. During trials, part of the symptom recognition could be done on the platform itself. The platform is approximately $99 \times 53 \times 19 \text{ mm}^3$ (i.e. length, width and height) in size and weights around 125g (including the power supply) [144].

5.1.2 Criteria and Demography

The database contains recordings from 92 participants (i.e. 36 females and 56 males). The average participant was 68 years (± 7.9 years). Most of them were either married or live with a partner (74 participants). The remainder was either single (5 participants), widowed (8 participants) or separated / divorced (5 participants). All participants have a clinical diagnosis of PD and a score of at least two on the Hoehn & Yahr scale (HYS) (i.e. moderate to severe phase of PD). The median score on the HYS is 3 with an interquartile range (IQR) of 0.5. The median scores of the unified Parkinson's Disease rating scale (UPDRS) for OFF, ON and intermediate are 40 (IQR ± 21.75), 14 (IQR ± 12.5) and 22 (IQR ± 17). Furthermore, 68 participants had predictable OFF states, 54 participants had unpredictable OFF states and 33 participants had sudden OFF states.

The inclusion and exclusion criteria have been summarized in Table 5.1. All participants fulfilled the inclusion criteria, while none of the exclusion criteria was present. The criteria and protocols were approved by the corresponding local ethic committees (Ireland: NUI Galway Research Ethics Committee, Israel: Assuta Hospital Helsinki Committee, Italy: Istituto di Ricovero e Cura a Carattere Scientifico, Spain: Comité Ético de Investigación Clínica de Centro Médico Teknon de Barcelona).

5.1.3 Protocols

In general, the protocols can be divided into two major parts: (1) screening / base-lining and (2) data acquisition. Both of these parts were performed on separate days. All participants were screened before any data acquisition took place. The main purpose of this procedure was to ensure that all inclusion criteria were met and none of the exclusion criteria were observed (see Table 5.1). During data acquisition, various scripted and unconstrained activities were performed in both ON and OFF states of the participant. Both parts of the protocols are touched upon in the remainder of this subsection.

During screening, the interviewer asked questions regarding overall health and sociodemographic nature. Furthermore, the inclusion criteria was verified (e.g. clinical diagnosis of PD and presence of clinical fluctuations). It was also established that none of the exclusion criteria (e.g. treatment with intestinal apomorphine or enrollment in another study) were met by the participant. A special focus was put on freezing of gait (FoG) and dyskinesia.

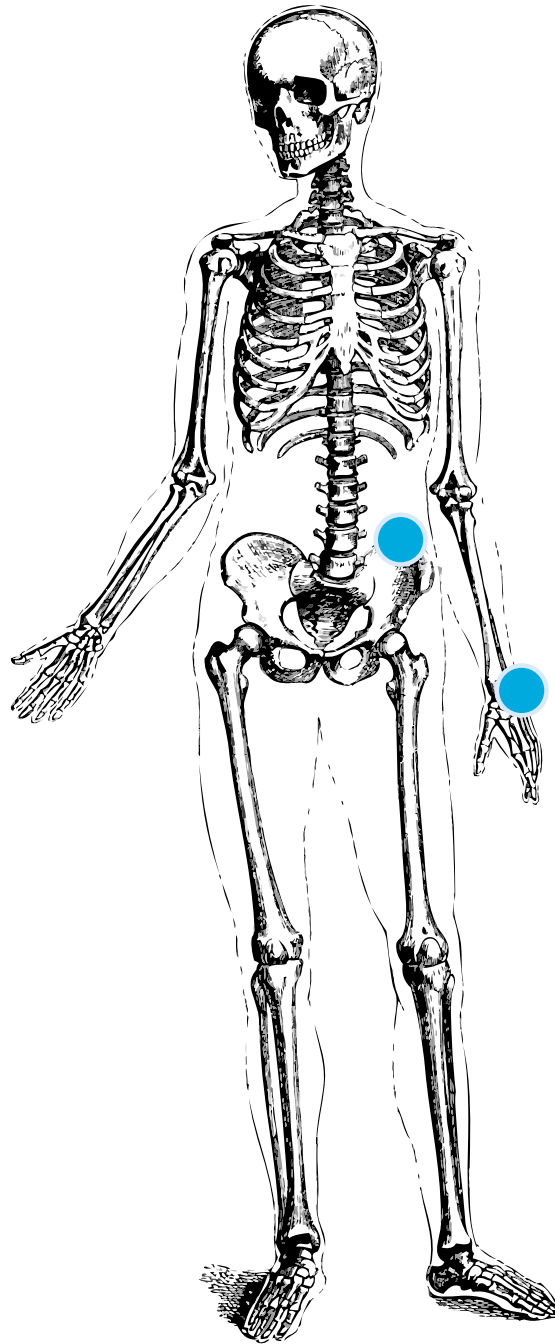


Figure 5.1: Highlights the sensor locations in relationship to a (human) skeleton. The wrist sensor is placed on the left hand. The waist sensor is located near the ASIS. The image has been retrieved from Wikimedia Commons and belongs to the public domain.

Inclusion criteria:

- Signed the consent form
- Clinical diagnosis of PD (see Table A.7)
- Hoehn and Yahr stage of two or above (moderate-severe phase of PD)
- Presence of clinical fluctuations
- Aged between 50 and 75 years

Exclusion criteria:

- Other health problems that impair gait or physical activities (e.g. rheumatologic, neuromuscular, respiratory, cardiologic problems or significant pain)
- Major consumption of alcohol or other drugs
- Wearing a pace maker or other implantable devices
- Receiving treatment with intestinal dopodopa or intestinal apomorphine
- Receiving treatment with deep brain stimulation (DBS)
- Enrollment in another study
- Mini-Mental-Score [55] below 23

Table 5.1: Inclusion and exclusion criteria for participation in data acquisition. The contents have been summarized from parts of the REMPARK project documentation (i.e. case report form for screening and baseline). The original document can be found in the appendix.

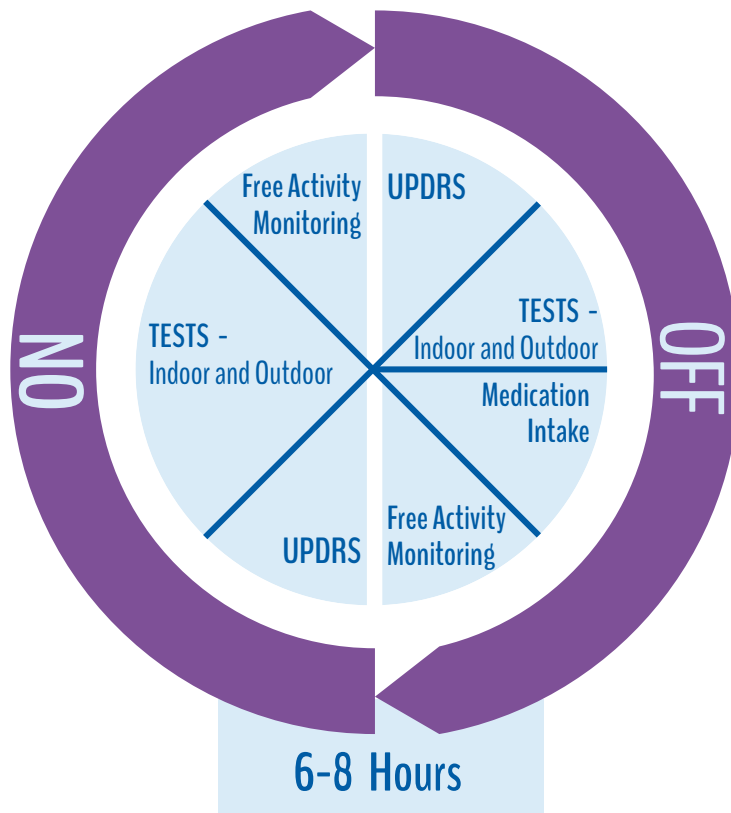


Figure 5.2: Illustrates the order and length of the recording sessions.

During data acquisition, the participant's activities were recorded by means of a wrist-mounted sensor and a waist-mounted sensor. The recording sessions can be largely divided into a recording session in a clinically defined OFF-state and a second recording session in a clinically defined ON-state. Both of these recording sessions were partly videotaped (first part) and directly annotated with tablet computer (second part). The general process is also highlighted in Figure 5.2.

Before the arrival of medical professionals and researchers at the participant's home in the morning, the participant was asked to skip the morning dose of medication. Thus, the participant was expected to be found in an OFF-state. Prior any recordings, the motor state was verified and the motor section of the UPDRS was evaluated. Afterward, the recording devices were switched on and synchronized. Then the sensors were placed on the participant's body (i.e. wrist and waist) and recording started (i.e. recording of sensor data, video recording and tablet annotation). The participant performed a series of tests (i.e. indoors walking test, FoG provocation test, gait test, outdoors walking test and tap test). Then the participant was asked to take his / her medication and performed free activities. Up to the gait test (including) the participant was videotaped. From this point onward, only annotations from the tablet computer are provided. Once the participant has switched to the ON-state, this session

is over. All recording devices are stopped and sensors are removed from the participant.

A similar procedure was repeated for the ON recording session. Again, the participant was recorded while performing a series of tests (i.e. indoors walking test, gait test, dyskinesia test, false positive test for tremor and OFF-state, outdoors walking test as well as tap test). At the beginning of this session, the motor section of the UPDRS is also evaluated. Afterward, the recording devices are switched on, synchronized and sensors are placed on the participant's body. The tests are performed and afterward the participant performs free activities. Then all recording devices are switched off.

At the end of the recording sessions, a few usability related questions are asked. The system usability scale (SUS) and Quebec User Evaluation of Satisfaction with Assistive Technology (QUEST) are answered by the participant.

5.1.4 Tests

The tests that were referred to in the previous subsection are summarized in the following.

- **Indoors Walking Test:** The participant starts by sitting in a chair. Then he / she is asked to stand up and start walking through the house / apartment (e.g. as if they were selling the house / apartment or showing it someone). Having shown several rooms, the participant returns to the chair and sits down again. The idea is to get overall gait related information (in OFF and ON states). The video recording is focused on the feet.
- **FoG Provocation Test:** The participant starts by sitting in a chair. He / she is then asked to get up and walk through an opened door (approximately 2 - 3 meters away), turn around, walk back to the chair and sit down. This test is repeated several times and the door may be gradually closed (narrowed) if FoG episodes could not be provoked. The participant's feet must be visible in the video recording.
- **Gait Test:** The participant is asked to walk for around 15 - 20 meters in a straight line (ideally on a flat and clear street, sidewalk or pavement) and stand still for a few seconds. An odometer is used as a reference for the walked distance. The video recording shows the participant (facing the back of the participant) walking the entire way. This test is repeated. However, this time the steps are counted and a stopwatch is used to measure the time required to bridge the distance.
- **Outdoors Walking Test:** The participant is asked to walk for a period of 10 - 15 minutes in the nearby neighborhood or park. The idea is to capture gait related information (in OFF and ON states).
- **Tap Test:** The participant is asked to perform a series of tapping movements with varying degrees of difficulty. This test is performed on a mobile phone with a touch sensitive screen [46,47].
- **Dyskinesia Test:** The participant starts by sitting in a chair. He / she is then asked to stand up and keep standing for a full minute before sitting down again for another full minute. The recordings show the entire body of the participant. The idea is to capture dyskinesia episodes with the recording devices.

- **False Positive Test For OFF:** The participant starts by sitting in a chair in the kitchen. He / she is then asked to carry a glass of water to another room (ideally the room furthest away from the kitchen). On the way back, the participant is also asked to read out loud a text and sit down in the kitchen. The idea is to perform and capture activities that could potentially be interpreted as signs of an OFF-state (e.g. slowness).
- **False Positive Test For Tremor:** The participant is asked to perform the following activities: brushing teeth, shaking a deodorant, erasing with an eraser, typing on a computer keyboard, cleaning window(s) as well as drying a glass. The idea is to perform and capture a set of activities that could potentially be interpreted as tremor (e.g. rhythmical movements).

5.1.5 Labeling

As previously mentioned, the sessions were videotaped and annotated. A smart-phone was used to record participants (or part thereof) and a tablet computer was used to annotate the participants' actions directly (e.g. walking, standing, rough terrain, etc.). Video recordings are only available for parts of the sensor signals (i.e. first part of session was videotaped while the second part was only annotated). For this reason, proper labeling is also only available where video recordings were available. The actual labeling was performed by medical professionals based on the video recordings. They provided the gold standard.

5.2 Contents of Database

The data that are analyzed in this thesis are comprised of signals from 92 patients and include ≈ 4500 minutes of recordings. This corresponds to those parts that have been video taped. The entire REMPARK DB contains roughly 360 hours of labeled signals [145]. It also contains recordings from patients in varying stages (i.e. ON, OFF, intermediate). This section is intended to give a high-level overview on those parts of the project's DB for which a gold standard (based on video recordings) could be established.

The recordings originate from four countries (i.e. Spain, Italy, Israel and Ireland). In Spain, Dr. Àngels Bayes led the movement signals gathering from Centro Médico Teknon in Barcelona. Dr. Roberta Annicchiarico coordinated the data collection in Rome, Italy. In Israel, signals were gathered under the coordination of Dr. Hadas Lewy in Maccabi Healthcare Services, Tel Aviv. Finally, signals gathered in Ireland were coordinated by Prof. Gearóid ÓLaighin from National University of Ireland, Galway. All participants had a clinical diagnosis of Idiopathic Parkinson's disease according to the United Kingdom (UK) PD Society Brain Bank [71]. Furthermore, all patients gave their signed informed consent before their participation. The experimental protocol was approved by the corresponding local Ethics Review Committee.

The general process was that patients started in the morning in the OFF-state as they were asked to skip their morning dose of medication. The procedure that followed (see Section 5.1.3) included a set of scripted activities (e.g. gait test, walking test, showing the house / apartment, carrying a glass of water) and an unscripted session. The scripted activities were designed to provoke certain motor symptoms (i.e. walk through a narrow doorway to initiate FoG) and

allowed to quantify other symptoms (e.g. dyskinesia and bradykinesia). The free session allowed recording of patients in their own apartment and performing arbitrary activities as they wished. Both recording sessions lasted (in total) several hours. In the afternoon, the recording sessions were repeated, this time in a clinically defined ON-state. During each recording session, participants wore a wrist-mounted accelerometer on the left wrist. Additionally, a sensor platform was attached to the patient's waist, recording acceleration data, gyroscope data and magnetometer data. In parallel, a video recording of all participants was done which was later synchronized to the sensor data and served as a basis for labeling. Also in parallel, a rough annotation was performed with a tablet computer. The gold standard was provided by medical professionals who performed the final labeling (based on the video recordings). At the beginning of the morning and afternoon session, a partial UPDRS (motor section) was determined in order to assess the patient's overall motor status.

The Table 5.2 shows a general overview of the symptoms that are present in the database on a per-patient-basis. This is done regardless of them occurring in ON, OFF or intermediate state.

Patient	Tremor	Dysk.	Brady.	FOG
1	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00
4	0.00	0.22	0.00	0.00
5	0.00	0.00	1.95	0.00
6	7.03	0.00	0.00	0.00
7	0.00	0.00	0.00	0.00
8	0.00	0.00	1.08	0.00
9	0.00	0.32	0.00	0.00
10	2.57	0.94	0.00	0.00
11	0.00	0.37	0.00	4.34
12	0.00	0.33	0.00	0.21
13	0.00	7.02	0.00	4.40
14	0.00	0.00	0.00	8.48
15	0.00	0.00	0.00	3.05
16	1.35	0.00	1.21	1.14
17	0.00	0.00	0.00	4.88
18	0.00	0.00	6.59	4.13
19	0.00	0.00	0.00	1.05
20	0.00	7.93	0.00	2.02
21	0.00	0.00	0.00	0.00
22	0.00	0.00	0.00	0.00
23	0.00	1.79	0.00	0.34
24	1.18	0.00	0.00	2.25
25	0.00	0.00	0.00	0.00
26	0.03	6.73	0.00	2.20
27	2.84	3.73	30.05	4.97
28	0.63	9.08	0.00	3.98
29	0.71	10.77	0.00	0.00
30	4.42	0.00	19.67	0.71
(continued)				

Patient	Tremor	Dysk.	Brady.	FOG
31	1.10	13.38	0.00	5.46
32	0.00	0.28	8.53	0.11
33	14.30	0.00	0.00	0.00
34	0.00	26.64	0.00	3.70
35	1.11	0.00	5.31	0.04
36	4.96	0.00	0.00	3.69
37	0.00	9.07	5.69	0.00
38	0.00	0.00	0.00	0.55
39	0.00	10.82	21.13	5.60
40	16.64	0.00	16.74	0.45
41	0.56	7.56	32.67	9.49
42	14.31	22.77	14.30	0.00
43	0.00	1.22	17.50	9.82
44	5.39	21.63	22.83	6.26
45	0.00	4.34	15.83	4.39
46	0.00	1.98	11.00	0.46
47	0.00	4.57	11.87	0.06
48	0.00	22.72	9.76	0.00
49	0.00	0.00	10.12	0.00
50	0.00	16.79	21.31	0.84
51	0.00	10.74	13.77	0.00
52	0.00	0.00	35.32	6.49
53	0.00	0.00	18.23	1.47
54	0.00	0.00	8.25	0.66
55	0.09	0.00	0.00	0.00
56	0.00	0.00	12.09	1.07
57	0.00	0.00	14.07	1.26
58	12.71	16.30	54.75	0.18
59	0.00	0.00	9.40	0.20
60	5.79	0.00	0.00	0.88
61	0.00	0.00	0.00	0.60
62	9.54	0.00	14.71	0.54
63	37.59	20.68	16.48	3.98
64	0.00	17.26	12.52	5.89
65	68.87	0.00	42.78	1.45
66	0.00	10.19	22.31	2.37
67	0.00	0.00	11.03	0.11
68	0.00	0.00	20.95	1.42
69	0.00	0.00	13.29	1.38
70	0.81	0.00	16.07	2.01
71	1.66	11.19	4.84	0.00
72	0.00	0.00	12.79	0.01
73	0.00	0.98	1.88	2.85
74	0.07	2.41	10.17	0.00
75	0.00	0.00	20.05	0.00
76	0.00	22.63	8.60	0.00
77	0.00	2.08	16.01	0.00
(continued)				

Patient	Tremor	Dysk.	Brady.	FOG
78	0.00	3.05	18.42	12.82
79	0.00	0.00	18.12	0.00
80	0.00	6.99	2.62	0.34
81	0.00	4.92	1.82	0.11
82	46.17	0.00	25.68	0.11
83	0.18	0.00	6.37	0.00
84	1.83	11.04	11.53	0.38
85	0.00	20.87	20.74	1.84
86	0.00	3.81	5.19	1.62
87	0.00	0.00	8.09	0.00
88	12.11	0.00	3.74	0.19
89	0.00	0.00	31.85	1.49
90	4.10	7.81	22.49	7.29
91	0.66	16.14	16.27	1.50
92	12.83	1.76	10.76	0.00

Table 5.2: Lists the amount of time (in minutes) that each patient experienced a particular symptom. Here resting tremor, dyskinesia, bradykinesia and FoG are listed. The values were extracted from those parts of the DB where a gold standard was available.

Label	Location	ON-State	OFF-State	Intermediate	Unknown
Undefined	Waist	633.02	490.59	109.33	1413.38
Undefined	Wrist	633.02	490.59	109.33	1413.28
Without tremor	Waist	763.79	633.24	109.48	60.51
Without tremor	Wrist	763.79	633.25	109.47	60.51
Right hand/arm tremor	Waist	45.11	105.85	0.00	1.68
Right hand/arm tremor	Wrist	45.11	105.85	0.00	1.68
Right foot/leg tremor	Waist	5.13	15.58	0.00	0.07
Right foot/leg tremor	Wrist	5.13	15.58	0.00	0.07
Trunk tremor	Waist	0.00	0.00	0.00	0.00
Trunk tremor	Wrist	0.00	0.00	0.00	0.00
Left hand/arm tremor	Waist	43.55	77.00	9.65	7.71
Left hand/arm tremor	Wrist	43.56	77.00	9.65	7.71
Left foot/leg tremor	Waist	0.18	12.53	0.00	1.78
Left foot/leg tremor	Wrist	0.18	12.53	0.00	1.78

Table 5.3: Lists the length of recordings (in minutes) with respect to resting tremor. The amount of time is given for combinations of sensor location (i.e. wrist or waist), motor state (i.e. ON, OFF and intermediate) as well as varying types of tremor (which can occur simultaneously). These values were extracted from those parts of the REMPARK DB for which a gold standard was available.

Label	Location	ON-State	OFF-State	Intermediate	Unknown
Undefined	Waist	445.71	350.89	74.27	1344.43
Undefined	Wrist	445.69	350.89	74.27	1344.33
Without dysk.	Waist	699.19	938.44	135.05	115.66
Without dysk.	Wrist	699.20	938.45	135.05	115.66
Weak head dysk.	Waist	50.97	0.54	0.00	1.88
Weak head dysk.	Wrist	50.98	0.54	0.00	1.88

(continued)

Label	Location	ON-State	OFF-State	Intermediate	Unknown
Strong head dysk.	Waist	25.17	0.00	0.00	0.00
Strong head dysk.	Wrist	25.18	0.00	0.00	0.00
Weak left hand/arm dysk.	Waist	118.79	4.52	4.56	2.53
Weak left hand/arm dysk.	Wrist	118.80	4.53	4.56	2.53
Strong left hand/arm dysk.	Waist	1.61	0.00	0.00	0.00
Strong left hand/arm dysk.	Wrist	1.61	0.00	0.00	0.00
Weak left foot/leg dysk.	Waist	38.68	22.14	5.76	2.64
Weak left foot/leg dysk.	Wrist	38.68	22.14	5.76	2.64
Strong left foot/leg dysk.	Waist	0.84	0.00	0.00	0.00
Strong left foot/leg dysk.	Wrist	0.84	0.00	0.00	0.00
Weak trunk dysk.	Waist	65.65	4.46	4.59	5.00
Weak trunk dysk.	Wrist	65.66	4.46	4.59	5.00
Strong trunk dysk.	Waist	21.28	0.00	0.00	1.36
Strong trunk dysk.	Wrist	21.28	0.00	0.00	1.36
Weak right hand/arm dysk.	Waist	134.86	0.12	0.00	10.31
Weak right hand/arm dysk.	Wrist	134.86	0.12	0.00	10.31
Strong right hand/arm dysk.	Waist	0.47	0.00	0.00	0.02
Strong right hand/arm dysk.	Wrist	0.47	0.00	0.00	0.02
Weak right foot/leg dysk.	Waist	89.55	1.34	1.76	1.21
Weak right foot/leg dysk.	Wrist	89.55	1.34	1.76	1.21
Strong right foot/leg dysk.	Waist	11.03	0.00	0.00	0.00
Strong right foot/leg dysk.	Wrist	11.03	0.00	0.00	0.00
Choreic	Waist	29.66	0.00	0.00	0.30
Choreic	Wrist	29.66	0.00	0.00	0.30
Dystonic	Waist	5.95	0.00	4.23	0.00
Dystonic	Wrist	5.95	0.00	4.23	0.00

Table 5.4: Lists the length of recordings (in minutes) with respect to dyskinesia. The amount of time is given for combinations of sensor location (i.e. wrist or waist), motor state (i.e. ON, OFF and intermediate) as well as varying types of dyskinesia (which can occur simultaneously). These values were extracted from those parts of the REMPARK DB for which a gold standard was available.

Label	Location	ON-State	OFF-State	Intermediate	Unknown
Undefined	Waist	523.71	232.75	121.05	1349.09
Undefined	Wrist	523.70	232.75	121.05	1348.99
Bradykinesia	Waist	50.11	768.37	25.75	54.91
Bradykinesia	Wrist	50.11	768.39	25.75	54.91
No bradykinesia	Waist	901.72	317.23	81.65	81.09
No bradykinesia	Wrist	901.73	317.23	81.65	81.08

Table 5.5: Lists the length of recordings (in minutes) with respect to bradykinesia. The amount of time is given for combinations of sensor location (i.e. wrist or waist), motor state (i.e. ON, OFF and intermediate) as well as varying types of bradykinesia. These values were extracted from those parts of the REMPARK DB for which a gold standard was available.

Label	Location	ON-State	OFF-State	Intermediate	Unknown
Undefined	Waist	625.41	646.93	126.05	1404.48
Undefined	Wrist	625.40	646.87	126.04	1404.37
Without FoG	Waist	829.22	548.80	94.92	70.13
Without FoG	Wrist	829.23	548.81	94.92	70.13
Start Hesitation FoG	Waist	3.53	17.00	1.28	3.75
Start Hesitation FoG	Wrist	3.54	17.01	1.28	3.75
Straight Line FoG	Waist	1.72	22.44	0.52	0.88
Straight Line FoG	Wrist	1.72	22.44	0.52	0.88
Turning FoG	Waist	7.59	43.09	2.94	3.11
Turning FoG	Wrist	7.59	43.13	2.94	3.11
Tight FoG	Waist	7.82	37.41	2.32	2.68
Tight FoG	Wrist	7.82	37.42	2.33	2.68
Destination FoG	Waist	0.25	2.69	0.43	0.06
Destination FoG	Wrist	0.25	2.69	0.43	0.06

Table 5.6: Lists the length of recordings (in minutes) with respect to FoG. The amount of time is given for combinations of sensor location (i.e. wrist or waist), motor state (i.e. ON, OFF and intermediate) as well as varying types of FoG. These values were extracted from those parts of the REMPARK DB for which a gold standard was available.

The remaining section discusses motor symptoms and their occurrences in the DB.

5.2.1 Tremor at Rest

The database contains 33 patients with tremor at rest. An abstract overview of tremor occurrences in dependence of the participant’s motor state (i.e. ON, OFF or intermediate) is shown in Table 5.3. Furthermore, the different locations of tremor are highlighted. The most predominant location of tremor is “right hand / arm tremor” (judged by absolute numbers). However, as the acceleration sensor was always on the patient’s left wrist only the “left hand / arm tremor” label can be effectively utilized.

Among the occurrences of tremor, the hands / arms are mostly affected. This is in accordance with the literature which states that upper extremities are more likely to be affected than lower extremities such as feet of legs [85, 146]. The database does not contain any recordings of tremor in the trunk, nor does it contain a large percentage of recordings in the intermediate motor state. Most recordings are labeled as either “undefined” or “without tremor”. Nonetheless, the database contains recordings of tremor for more than five hours for each kind of sensor and sensor platform.

The result of a χ^2 -test ($\chi^2 = 56.98$, $\chi^2_{1;99.9\%} = 10.828$, $\phi = 0.18$) suggests a dependency between ON-OFF motor state and presence of tremor. However, the strength of this dependency produces only a decrease of the relative error rate by 18% (see ϕ -coefficient). This indicates that the presence of tremor at rest only contributes a rather small part to the ON-OFF motor state. Thus tremor at rest should not solely be used to estimate the ON-OFF motor state. This is also reflected in the following measures: sensitivity 69%, specificity 55%, positive predictive value (PPV) 25% and negative predictive value (NPV) 89%. Similar results are obtained when testing dependence between arm / hand tremor and motor state as well as the dependence between left hand / arm tremor and motor state.

In most cases, recordings from the waist and wrist sensors are aligned (in terms of length of recorded data). However, there are several occurrences where this is not the case due to mostly technical difficulties or human error. A difference of several minutes in both wrist and waist recordings was observed. The most probable explanation is that the sensor platforms were not turned off at the same time (i.e. wrist first and later waist).

The total amount of time spent in tremor, as shown in Table 5.2, does not necessarily correspond to the numbers found in Table 5.3. This is due to the fact that several types of tremor may be occurring at the same time. Thus artificially increasing the total number of time in tremor.

5.2.2 Dyskinesia

In total, the database contains 45 patients suffering from dyskinesia in the course of their recording sessions. Table 5.4 summarizes the occurrences of dyskinesia in the database. There the overall length of dyskinesia episodes is shown for each sensor platform and motor state. The most predominant types of dyskinesia were observed in the upper extremities (i.e. hand and / or arm). Alone these recordings amount to more than four hours of recorded data for each sensor

(i.e. wrist and waist). As one may expect, the largest amount of dyskinesia was recorded in the ON period. Only negligible differences in recorded data lengths between wrist and waist data were observed.

The result of a χ^2 -test ($\chi^2 = 501.42$, $\chi^2_{1;99.9\%} = 10.828$, $\phi = 0.47$) suggests that ON-OFF motor state and presence of dyskinesia are indeed dependent as the literature proposes (see Chapter 2). The ϕ -coefficient indicates a decrease of the relative error rate by about half (i.e. 47%). The measures sensitivity (95%), specificity (57%), PPV (97%) and NPV (46%) reflect this as well. Combining detection of dyskinesia with further symptoms may produce even lower error rates.

Similarly to the remarks on tremor, several types of dyskinesia may be occurring at the same time. Thus the numbers shown on Table 5.4 are likely to be greater than those found in Table 5.2.

5.2.3 Bradykinesia

The database contains 60 patients suffering from bradykinesia. In Table 5.5, a summary of all occurrences is given. Here the length of recordings (in minutes) is shown against the motor state of the patient. Additionally, the different sensor locations (and thus the source of the recordings) are highlighted. In general, most occurrences of bradykinesia were recorded in the OFF period. In total fifteen hours of data, which exhibits bradykinesia, are contained in the database. An increase in bradykinesia is observed in the OFF-state, which stands in accordance with a typical PD profile (see Chapter 2).

The result of a χ^2 -test ($\chi^2 = 905.7$, $\chi^2_{1;99.9\%} = 10.828$, $\phi = 0.67$) suggests that presence of bradykinesia does seem to be dependent on the ON-OFF motor state. The ϕ -coefficient indicates a decrease of the relative error rate by 67%. The measures sensitivity (94%), specificity (74%), PPV (71%) and NPV (95%) do support this finding. Bradykinesia (on its own) might not be enough to estimate the patient's motor state, but it does contribute. However, if bradykinesia is used in combination with further symptoms lower error rates could be obtained.

The recordings of both sensor locations are mostly aligned (in terms of length of recorded data). Their length of recordings lie within tolerable differences, except for a noteworthy difference in "Undefined" label and "Unknown" motor state. The resulting difference can best be explained by the time difference between turning on / off the waist sensor and turning on / off the wrist sensor as well as possible connection problems between the two sensor platforms.

5.2.4 Freezing of Gait

In total, 62 patients in the database experienced FoG episodes during recording sessions. This is reflected in Table 5.6, where the occurrences of several FoG types relative to the patient's motor status and sensors' location are shown. The most predominant types of FoG were observed to be "turning" and "tight". These two types of FoG amount to nearly to one and a half hours in the OFF period alone, while the total number of recorded data amounts to more than two and a half hours. Start and destination FoG were least observed. Again, an increase of FoG occurrences was observed in the patient's OFF-state.

The result of a χ^2 -test ($\chi^2 = 109.67$, $\chi^2_{1;99.9\%} = 10.828$, $\phi = 0.27$) indicates a dependence between the ON-OFF motor state and presence of FoG. However, the strength of this dependence does only result in a decrease of the relative error rate by 27% (see ϕ -coefficient). As such, the presence or absence of FoG should not be solely used to estimate the motor state. This is underlined by the measures sensitivity (0.85), specificity (0.60), PPV (0.18) and NPV (0.98).

The predominance of “turning” and “tight” FoG could also have occurred due to the structure of recording sessions. They included tests that may have favored the occurrences of these freezing episodes. Recordings of both wrist and waist sensors show nearly identical amounts of data (i.e. differences lie within an acceptable range).

5.2.5 Summary

The data acquisition and labeling procedure for the recordings of 92 patients were outlined. Furthermore, inclusion and exclusion criteria of the REMPARK database were described. The general recording procedure and sequence (including utilized sensors, their locations, and scripted activities) were outlined as well.

The contents of the REMPARK database were (partially) summarized. Occurrences and statistical measures of several PD motor symptoms were presented. This includes the contributions that were made by tremor at rest, dyskinesia, bradykinesia and FoG in terms of distinguishing motor states. The results of the χ^2 -test indicated that bradykinesia contributed most. On its own bradykinesia decreased the relative error rate by 67%. The remaining symptoms (and side effects) reduced the relative error rate by 47% (dyskinesia), 27% (FoG) and 18% (resting tremor). This suggests that combining the contributions from bradykinesia, dyskinesia and FoG may lead to an even greater reduction. Thus allowing an adequate distinction between ON and OFF motor states.

Chapter 6

Indication of Parkinson's Disease Motor Symptoms

Several artificial intelligence (AI)-based approaches for indicating symptoms related to Parkinson's Disease (PD) are discussed. In the course of this chapter, algorithms for recognizing motor symptoms in patients with PD are developed. Furthermore, initial evaluations of these approaches are presented. Thus, most of this chapter is dedicated to answering research question three. With respect to the requirements and related work, a solution is formulated. The actual comparison against the state-of-the-art is done in the next chapter (see Chapter 7).

6.1 Tremor (at Rest)

The effectiveness of therapeutic interventions (e.g. levodopa-based medications) is partially measured through tremor assessment. Consequently, detecting tremor may improve the evaluation of such therapeutic interventions. This section outlines the development of an algorithm for indicating the presence of Parkinsonian tremor in PD patients. The first variation is a naive approach and involves training a support vector machine (SVM) in directly classifying tremor without any major pre- or post-processing. As the section progresses, the (naive) approach is iteratively refined. Each iteration will likely add complexity based on the findings of the previous approach (with the intention of improving results). For reasons of comparability, the training and test datasets stay the same throughout all iterations. Details on the corresponding datasets can be found in Table 6.1.

The overall database (DB) contains 92 patients, while 33 of them experienced tremor during recording sessions. The training and testing datasets are comprised of 9 and 174 recordings (i.e. 89 patients), respectively. None of the presented datasets overlap and recordings are used exactly once.

As it is apparent in Chapter 5 ("Database: Patients and Their Symptoms"), the DB contains labels for several types of tremor (e.g. left hand / arm tremor, right hand / arm tremor, trunk tremor). However only data samples labeled as "left hand / arm tremor" or "no tremor" are used for detecting the symptom. Consequently, Table 6.1 refers to these labels as well. The primary reason for this

restriction is that upper extremities are more likely to be affected (as opposed to lower extremities; see Chapter 2) and that data from the left hand was directly measured and available (i.e. one of the sensor platforms was located around the left wrist). Furthermore, focus (of those performing the labeling) might have been on trunk and left hand tremor rather than other locations such as the legs. They might have been biased by the presence of sensors.

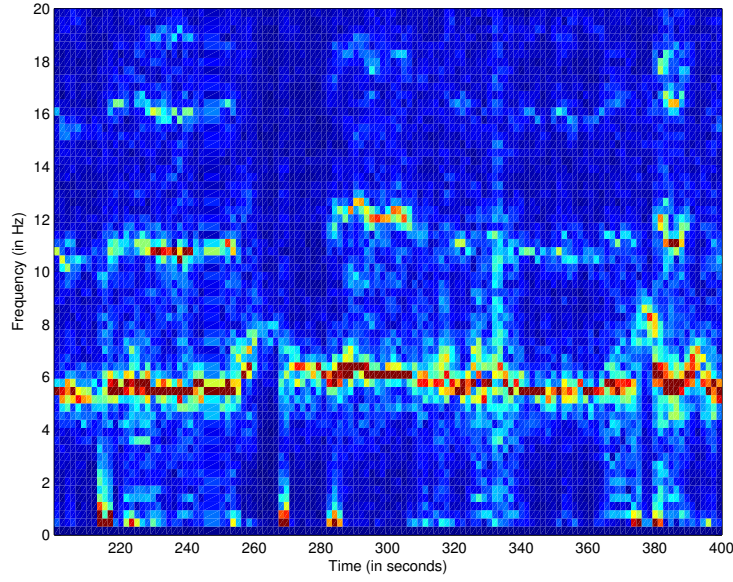
An analysis of the DB yielded to the conclusion that less than 1% of data samples were “corrupted” or missing. Those parts of the recordings were not utilized. As such, this restriction merely represents an additional measure of quality and does not have a major effect on the datasets’ sizes. The test dataset was automatically relabeled. This was done to increase the amount of usable data samples and, at the same time, reduce the number of “undefined” labels. The relabeling was performed in such a way that patients without “left hand / arm tremor” labels or with only “no tremor” labels were automatically relabeled as patients with “no tremor” (i.e. the entire patient is relabeled accordingly). On the other hand, data from patients who experienced tremor during the recorded datasets were constrained to the data with tremor. The remaining data (i.e. everything except the tremor data) is rejected. Thus, sensitivity is measured only from patients with tremor while those patients that did not experience tremor (during the recording sessions) determine specificity.

As described in the related work (see Chapter 3), frequency related features are commonly used to characterize tremor in acceleration signals. The frequency behavior of tremor is illustrated in Figure 6.1, which shows the frequency distribution of signals obtained from a wrist-worn sensor by a patient. It is clearly observable that frequencies from 4 Hz to 6 Hz appear when Parkinsonian tremor is present. In addition, these frequencies are not observed when tremor is absent, which agrees with current literature [118]. Given the main frequency behavior of this sort of tremor, it could theoretically be detected uniquely by means of frequency characteristics. However, many other features have been used in the literature. Thus, in order to measure the impact of non-frequency features in recognizing tremor, two feature sets are proposed. On the one hand, the first set employs only frequency features while, on the other hand, the second set will also utilize additional features that were previously used in the literature. The specific list of features used is given in Table 6.2.

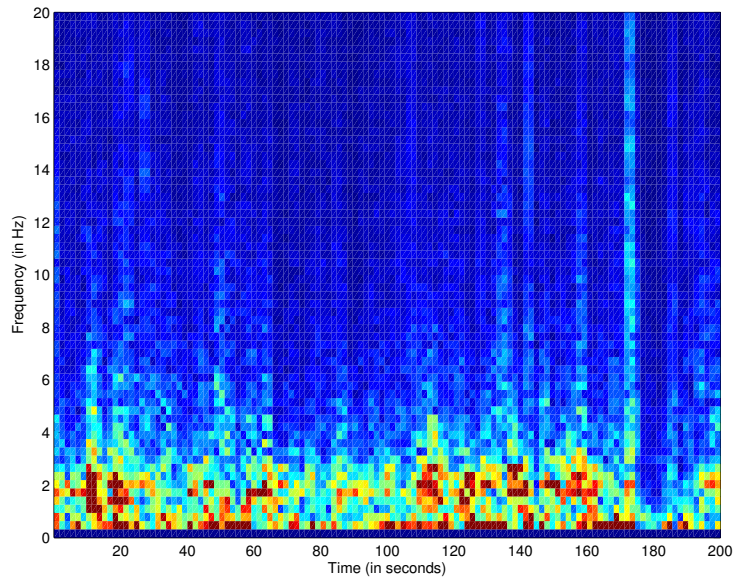
Results show that a real-time implementation of the proposed approach is feasible. However, signals obtained while patients did not wear the system, that is, in the beginning and end of the recordings, were rejected. In order to avoid false positives during these situations in a real-time daily use of the proposed approach, a *courtesy period* of, for instance, 10 minutes after switching on the tremor device could be used. Similarly, the last 10 minutes before switching it off could be rejected. As stated in Chapter 1, the development of the algorithms may be considered to have acceptable results if both sensitivity and specificity are above 80%. For the final algorithms, a threshold of at least 90% is expected to be reached.

6.1.1 Variation 1: Naive Approach

The first approach for signifying PD tremor in acceleration signals shall be a naive one. Here, a sliding window is used to directly distinguish between tremor and normal gait with an SVM.



(a) Frequency spectrum from an individual with tremor.



(b) Frequency spectrum from an individual without tremor.

Figure 6.1: Shows the frequency spectrum of an individual with tremor and another individual without tremor. It can be noted that during episodes of tremor frequencies between 4 - 6 Hz are apparent (as well as their harmonics).

	Training	Test
Number of tremor windows	1589	3209
Number of non-tremor windows	2109	106596
Number of recordings in ON state	3	86
Number of recordings in OFF state	6	88
Overall number of recordings	9	174

Table 6.1: Lists the number of windows (before aggregation) in each dataset that are used for signifying tremor (i.e. left hand tremor).

Indexes	Feature
1	FFT (raw, mean frequency removed)
2	Peak frequency and it's amplitude
3	Median and mean amplitude
4	Sum of every two-adjacent amplitudes
5	Sum of every four-adjacent amplitudes
6	Sum of first, second and third harmonic
7	Histogram (bins: 0,1,2,3,...,15)
8	Correlation between acceleration signals (XY, XZ, YZ, max)
9	Entropy of signal in time domain and frequency domain

Table 6.2: Shows the full set of features used for tremor detection. The reduced feature set is comprised of index 1.

Methodology: The time series from the wrist sensor (i.e. left hand) are resampled to 40 Hz. This sampling rate was chosen to reduce the amount of data samples, but at the same time still retain the frequency characteristics of tremor as well as (ordinary) human movements [11]. The resampled stream is divided into equally sized windows (length denoted as ws) which overlap to 50%. A set of features is then extracted for each individual window (see Table 6.2). These features are used to train an SVM in distinguishing windows with tremor and without tremor.

Frequency from three axes must be obtained, and their amplitudes are summed up without taking into account the amplitude of the zero-frequency harmonic. Thus, dependence on the sensor's orientation is avoided. Given the representation of a window under analysis \mathbf{f}_i , a previously trained SVM determines whether this window corresponds to tremor according to:

$$tremor_i^1 = \begin{cases} 0 & \text{no tremor} & \text{if } f_{SVM} \leq 0 \\ 1 & \text{tremor} & \text{if } f_{SVM} > 0 \end{cases} \quad (6.1)$$

where $f_{svm}(x) = \sum_{i=1}^l y_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b$, $\mathbf{x}_1, \dots, \mathbf{x}_l$ are the support vectors (SVs). y_i and α_i are the corresponding label and lagrange multiplier of each SV. x and b are the classified data sample and the bias [39]. Here, the one in the superscript of $tremor_i^1$ identifies the first variation. Subsequent variations will have a higher index in the superscript.

Model Selection: As previously pointed out, the training dataset does only include “left hand / arm tremor” labels. Furthermore, only windows without missing data samples are utilized with the intention to minimize potential side

effects. The SVM model is trained with the features (extracted from the windows of the training set). During training, varying settings for kernel, weighting, cost and gamma are considered. The weighting parameters are used to balance both classes “left hand tremor” (less than 5% of available labels) and “non-tremor” (at least 95% of available labels). The cost and gamma parameters were systematically evaluated (i.e. $10^x, x \in \{-3, -2, \dots, 2, 3\}$) depending on the chosen kernel (i.e. radial basis function (RBF) kernel or linear kernel). Additionally, a ten-fold cross-validation is performed. However, instead of averaging the accuracy of the training set, the geometric mean of sensitivity and specificity is used (i.e. $\sqrt{\text{sensitivity} * \text{specificity}}$) to identify those parameter combinations with high sensitivity and specificity. Then the maximum geometric mean is used to train the final SVM model and determine its test performance.

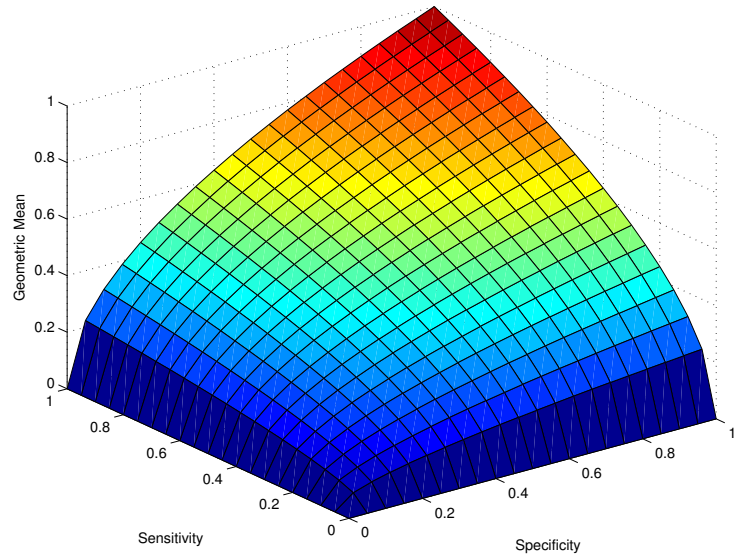
The geometric mean was chosen as it does treat both classes (i.e. “left hand / arm tremor” and “no tremor”) equally as opposed to accuracy which implicitly weights classes. The weighting of latter measure can be problematic if the classes have (very) different priors. Assuming, there is a large amount of “no tremor” labels and comparatively few “left hand / arm tremor” labels, this could lead to a situation in which an SVM that only recognizes “no tremor” windows (and nothing else) achieves a higher accuracy than a properly trained SVM (but with slightly lower specificity). This behavior is illustrated in Figure 6.2.

In total, four conditions were evaluated: two kernels (i.e. RBF and linear) and two feature sets (i.e. frequency only and other commonly employed features). The optimal window size ws is initially determined from the following set of discrete values: $ws \in 2^{\{5, \dots, 10\}}$ samples. For each ws , an SVM is trained with the training dataset and the respective results are taken from the test dataset.

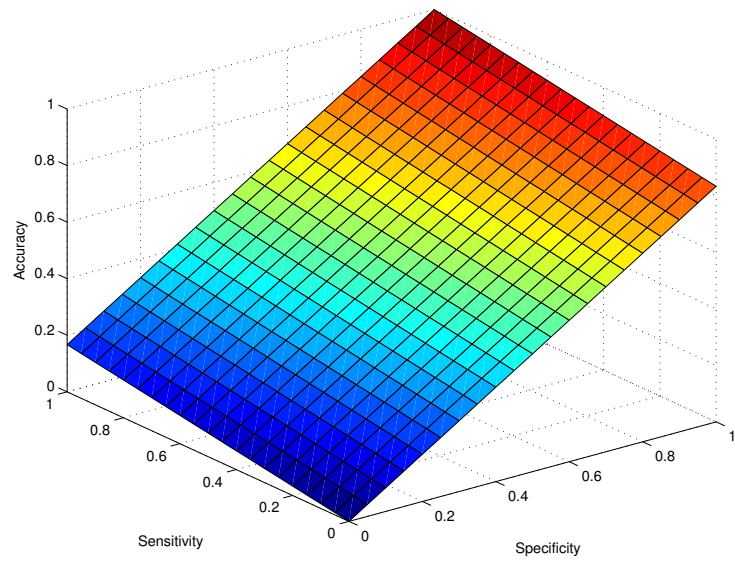
Results: The window size of 128 samples (i.e. 3.2 seconds and 1.6 seconds overlap) was chosen according to the results depicted in Figure 6.3. The figure shows measures of quality (i.e. accuracy, geometric mean, sensitivity and specificity), that were observed on the test dataset. The window size of 128 samples represents a reasonable compromise between the aforementioned measures of quality as well as the window size itself (i.e. length in seconds). Preference is given to a lower window size because lower window sizes require less memory (during operation) and thus they are more easily implemented on computationally constrained devices (such as microprocessors). This has been done despite the fact that a window size of 1024 samples (i.e. 25.6 seconds) achieved the highest accuracy and geometric mean.

The results shown in Table 6.3 contain the geometric mean and accuracy of the above approach on the test dataset. These results represent the groundwork for the upcoming variations of the tremor recognition algorithm. Surprisingly, the best performance on the test dataset was achieved with the simplest condition (i.e. linear kernel and frequency features only). In terms of geometric mean and accuracy, the conditions with a linear SVM kernel outperformed those conditions with an RBF kernel, regardless of the chosen feature set.

Regarding sensitivity, the RBF conditions produced superior results to those with a linear kernel. Furthermore, it is noticeable that the gap between sensitivity and specificity is larger in the RBF conditions than in the linear conditions



(a) Geometric Mean



(b) Accuracy

Figure 6.2: Shows the behavior of geometric mean and accuracy with unbalanced datasets (number of positive and negative labels diverges).

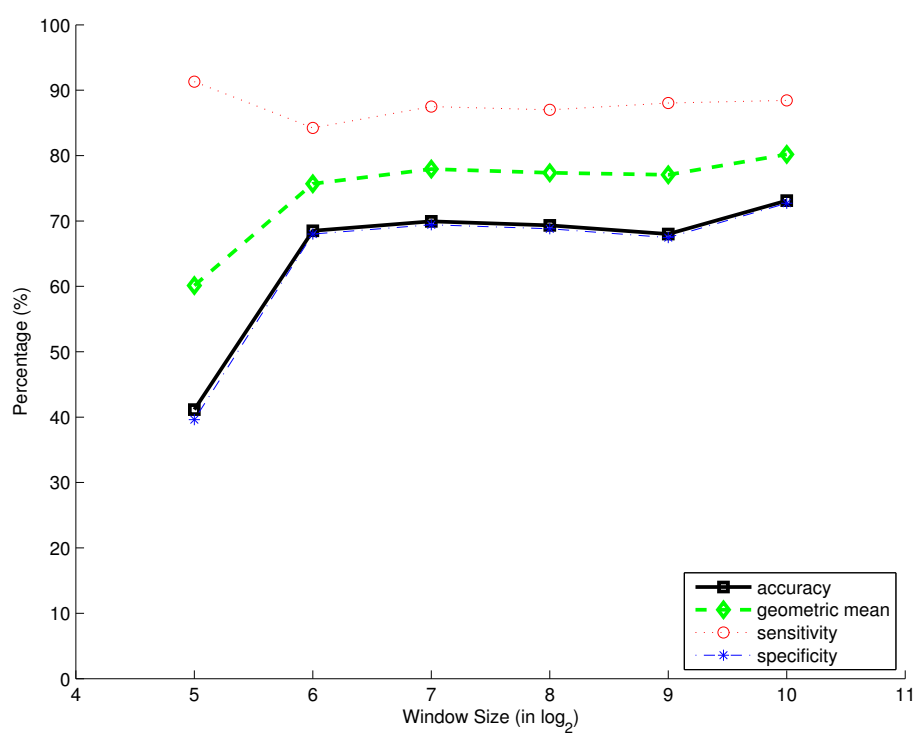


Figure 6.3: Illustrates the results of an evaluation for varying window sizes with respect to Parkinsonian tremor. For each window size, an SVM has been trained on the training dataset and evaluated on the test dataset.

Kernel	RBF	Linear	RBF	Linear
Features	Freq.	Freq.	All	All
Sensitivity (train)	0.910	0.649	0.928	0.847
Specificity (train)	0.568	0.799	0.645	0.742
Data Usage (train)	1.000	1.000	1.000	1.000
Geometric Mean (train)	0.719	0.720	0.774	0.792
Accuracy (train)	0.570	0.798	0.647	0.742
Sensitivity (test)	0.936	0.803	0.921	0.846
Specificity (test)	0.564	0.805	0.652	0.759
Data Usage (test)	1.000	1.000	1.000	1.000
Geometric Mean (test)	0.727	0.804	0.775	0.801
Accuracy (test)	0.576	0.805	0.661	0.762

Table 6.3: Outlines results in recognizing tremor with the naive approach (i.e. variation one). Various measures are listed for both datasets.

where minor differences are observed. The training dataset also yields to an increased sensitivity of conditions with an RBF kernel compared to those with a linear one. The data usage in all conditions is marked as 100% (i.e. regardless of kernel, feature set and dataset) which indicated that all available windows of the corresponding dataset (see Table 6.1) have been considered. This is no surprise, as the naive approach does not discard windows. However, succeeding variations of this approach may be allowed to do so.

In terms of specificity, the linear kernel seems to provide a “better” generalization (quite possibly because of its simplicity) than the RBF kernel, regardless of the chosen feature set and dataset. However, this comes at the cost of a reduced sensitivity when comparing the results of linear conditions to those with an RBF kernel. The opposite is valid for RBF conditions, they achieve a higher sensitivity at the cost of a reduced specificity.

Conclusion: The presented approach does not yield to reasonably good results. However, it can be seen that even this simple and arguably naive approach may already achieve a sensitivity and specificity above 80%. Furthermore, the conditions with an RBF kernel show a promising sensitivity, but a hardly acceptable specificity.

Overall, the naive approach seems inflexible and the results in terms of geometric mean as well as accuracy can still be improved.

6.1.2 Variation 2: Aggregation

In order to minimize the resources needed for detecting tremor, time windows must be short (i.e. few seconds). However, short window sizes are likely to produce false positives (FPs) (also shown in Figure 6.3) since short rhythmical movements may be confused with tremor (e.g. teeth-brushing, shaking deodorant and typing; see Section 5.1.4). A meta-analysis, similarly to the approach by Salarian et al. [143], is added in order to enhance the reliability of the proposed approach by removing isolated segments that were classified to exhibit tremor or tremor-like behavior. This is done to counteract the reduced specificity on the test dataset for variation one (see Table 6.3).

Methodology: The processing is done very similarly to the previous approach (Section 6.1.1). The signals are resampled to 40 Hz then split into windows of 3.2 seconds length ($ws = 128$) and a 50% overlap. Afterwards, features are extracted and an SVM is trained to recognize tremor. However, instead of relying on the SVM’s classification, its outputs are aggregated over a period of time t . Thus, the final classification is less susceptible to noise in a particular window. The classification results from n consecutive windows are aggregated and used to provide a final classification output. If the proportion of windows that were classified to exhibit tremor is above a threshold th then the final classification is also that tremor is present. Otherwise, no tremor is present (see Equation 6.4).

The meta-analysis is applied once the signals have been evaluated by the SVMs. This method considers the algorithm’s outputs in a set of n consecutive windows $tremor_1^1, \dots, tremor_n^1$ that cover a period of t seconds. These outputs are aggregated into a value denoted as c_j that represents the confidence degree of having tremor in the period of t seconds (starting at the j^{th} window):

$$t = \frac{ws(n+1)}{40 * 2} \quad (6.2)$$

$$c_j = \sum_{i=j}^{j+n-1} \frac{tremor_i^1}{n} \quad (6.3)$$

$$tremor_j^2 = \begin{cases} 0 & \text{no tremor} & \text{if } c_j < th \\ 1 & \text{tremor} & \text{if } c_j \geq th \end{cases} \quad (6.4)$$

where $th, c_j \in [0, 1]$; $j, ws, n \in \mathbb{N}$; $j, ws, n > 0$; $t \in \mathbb{R}$; $t > 0$ and $tremor_i^1$ may be found in Equation 6.1.

Model Selection: The training dataset (see Table 6.1) is used to initially train an SVM as well as to optimize the parameters t (i.e. length of aggregated time frame) and th (i.e. minimum threshold for tremor detection). The testing dataset is then used to evaluate the performance of this approach. For both parameters a discrete set of values is used during evaluation (i.e. $t \in \{10, 15, 20, 25, 30, 45, 60\}$ and $th \in \{0, 0.05, \dots, 0.95, 1\}$).

Overall, two feature sets (i.e. one with frequency features only and one with various commonly employed features) and two kernels (i.e. linear and RBF) are evaluated. Instead of solely evaluating the SVM’s predicted labels for each window, consecutive windows are aggregated into blocks of fixed length (e.g. 15, 30, 45 and 60 seconds). The original labels of the test dataset were used to determine the labels for these blocks. If at least one “left hand tremor” label is among the labels, then the aggregated label is set to “left hand tremor” as well. Otherwise the aggregated label is set to “no tremor”. During testing, the predicted labels from the SVM are used to determine the proportion of tremor windows. If the proportion is above a predefined threshold, then the label is considered to be “left hand tremor” and otherwise “no tremor”.

Results: The best performing parameter configurations and results are listed in Table 6.4. There it is apparent that the best performance (judging by the

geometric mean) is achieved when using the full feature set and an RBF kernel. This condition yielded to a sensitivity and specificity of 90.2% and 93.2% respectively. However, the full feature set with the linear kernel yielded to a comparable result (sensitivity: 84.2%, specificity: 94.6%). Nonetheless, the other conditions follow closely in terms of accuracy as well as geometric mean.

In general, it can be noted that conditions with the full feature set outperformed those with the reduced feature set. Thus suggesting an overvalue of the additional features. Levels close to the 90% mark were reached. The same tendency is observed in the training dataset. Furthermore, it is apparent that longer aggregation lengths are favored (i.e. 45 and 60 seconds), which indicated reduced specificities of the underlying SVMs. Here, the specificity for all four conditions is above the 90% threshold, while the sensitivity is constantly above the 80% mark. This applies to the test dataset as well as the training dataset.

Most thresholds th were found to be above the natural border of 50%. This suggests that the number of FPs can actually be reduced in this way. Comparing specificities of the second variation to the first approach, an improvement is clearly observable. They are located in the upper spectrum (i.e. 80% or higher) and only one threshold is at the intuitive border of 50%. The best configuration (i.e. RBF kernel with full feature set) as well as its counterpart (RBF with reduced feature set) have a threshold of 75% and 90%, respectively. This fact suggests that the underlying SVM model produces a relatively large number of false positives (results from Table 6.3 support this interpretation).

The impact of window aggregation t and threshold th are highlighted in Figure 6.4. A value of zero for t means that no aggregation is done. This case provides the worst case, as no meta-analysis is performed to reduce false positives and false negatives. Optimal performance is obtained for any $t \geq 30$ seconds. According to Figure 6.4, the best value for th is shown to be among 40% and 70% (for the linear kernel), which is reasonably close to 50%, meaning that at least half of the windows in the period of t seconds must be considered as tremor in order to accept the complete period to have the symptom. However, Figure 6.4 also shows that an optimal th for the RBF kernel is expected to be between 70% and 90% before the geometric mean ebbs off. Such high thresholds suggest that quite a number of FPs were apparent.

The data usage is still denoted as 100% because none of the windows are being rejected by the proposed algorithm.

Conclusion: The results show an overvalue of the full feature set which had not been quite as clear in the naive variation. Comparing to the naive approach, a definitive improvement is apparent. However, the different conditions do not benefit equally from the aggregation. The conditions with an RBF kernel clearly improve while the linear kernel does not change as much. Nonetheless, the average geometric mean increased by roughly 11.6%. It can also be observed that sensitivity and specificity do not benefit uniformly from the aggregation (sensitivity: -4.8%, specificity: 23.9%). For the most part, the specificity benefited from the aggregation at the cost of a slightly reduced sensitivity.

Those conditions with the RBF kernel have achieved reasonable acceptable results. Despite the clear improvement due to the aggregation, the overall sensitivity and specificity still leave room for improvements.

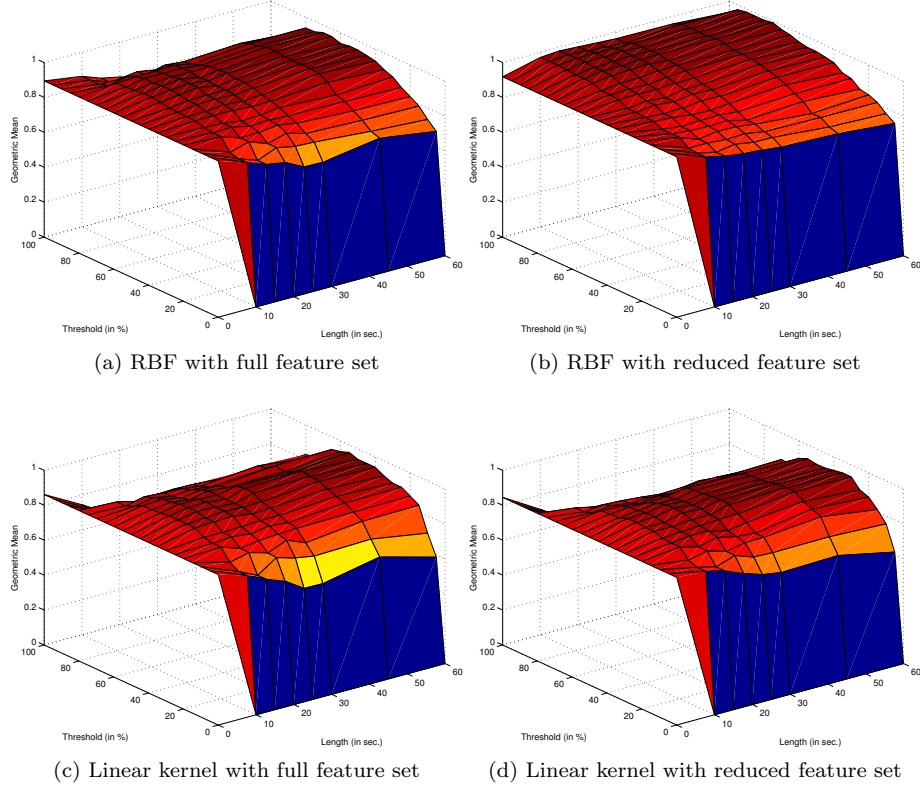


Figure 6.4: Indicates the effect of window aggregation t and threshold th on geometric mean. The results are shown for all four conditions.

Kernel Features	RBF Freq.	Linear Freq.	RBF All	Linear All
t	45	60	60	45
th	0.900	0.500	0.750	0.650
Sensitivity (train)	0.800	0.875	0.875	0.900
Specificity (train)	0.949	0.916	0.926	0.934
Data Usage (train)	1.000	1.000	1.000	1.000
Geometric Mean (train)	0.871	0.895	0.900	0.917
Accuracy (train)	0.946	0.915	0.925	0.934
Sensitivity (test)	0.820	0.848	0.902	0.842
Specificity (test)	0.946	0.911	0.932	0.946
Data Usage (test)	1.000	1.000	1.000	1.000
Geometric Mean (test)	0.881	0.879	0.917	0.892
Accuracy (test)	0.941	0.908	0.931	0.942

Table 6.4: Summarizes results in recognizing tremor with the one-sided approach (i.e. variation 2).

6.1.3 Variation 3: Two-Sided Aggregation

The third variation aggregates the SVM’s outputs as well. However, this variation uses two thresholds (i.e. one for classifying an aggregated window as tremor and a second one for classifying aggregated windows as no tremor) as opposed to a single threshold. As with the second approach, the general assumption is that activities of daily living (ADL) (i.e. teeth-brushing, shaking a deodorant, typing, etc.) can easily be confused with tremor due to their rhythmical components. However, the previous approach may be too rough as it reduced the overall sensitivity. To reduce these issues without further lowering sensitivity, the confidence degree (or proportion of tremor windows) is treated in a different way in order to tune the algorithm to be more specific or sensitive in detecting tremor. Consequently, this approach might not always provide a classification result. It introduces uncertainty by outputting “unknown / undefined” as a result and may ignore analyzed data samples.

Methodology: Again, the time series are resampled to 40 Hz and split into equally sized windows (i.e. 3.2 seconds length or $ws = 128$ samples). The window is slid across the time series in increments of 1.6 seconds (or 64 samples). For each window a set of features is extracted which are then used during training and classification. Afterwards, the SVM’s outputs are aggregated over a time period t and the proportion of windows classified as tremor c_j (within the time period t) is determined (see Equation 6.3). The degree of confidence can then be used to detect the presence of tremor as follows:

$$tremor_j^3 = \begin{cases} 0 & \text{no tremor} & \text{if } c_j < th_l \\ 1 & \text{tremor} & \text{if } c_j \geq th_u \\ -1 & \text{undefined} & \text{if } th_l \leq c_j < th_u \end{cases} \quad (6.5)$$

where $th_l \leq th_u$; $c_j, th_l, th_u \in [0, 1]$; $j \in \mathbb{N}$; $j > 0$

This way, a high sensitivity could be obtained by setting a high th_u or, inversely, a high specificity could be obtained with a low th_l . However, maximizing both performance measurements requires a suitable balance between th_l and th_u . Previously, the effect of not rejecting any meta-analysis period had been tested (i.e. $th_l = th_u$). Here, this rejection is enabled by allowing $th_l \neq th_u$ which, ideally, should allow the method to improve both specificity and sensitivity. However, the larger the gap between these thresholds the more aggregated windows are going to be labeled as “unknown / undefined” because they cannot be labeled as tremor nor as non-tremor. This approach is tested in the next sections.

All in all, the methodology can be summarized as illustrated in Figure 6.5.

Model Selection: The SVM models are still trained based on the training dataset. The parameters t (i.e. length of aggregated time frame), th_l (i.e. maximum threshold for non-tremor detection) and th_u (i.e. minimum threshold for tremor detection) are evaluated and tuned. The final results are based on the test dataset. The following sets of discrete values were evaluated: $t \in \{10, 15, 20, 25, 30, 45, 60\}$, $th_l \in \{0, 0.05, 0.1, \dots, 0.95, 1.0\}$ and $th_u \in (th_u \geq th_l | th_u \in \{0, 0.05, 0.1, \dots, 0.95, 1\})$.

For details on the (re-)labeling of the original labels refer to the model selection of the previous approach (see Section 6.1.2).

Kernel Features	RBF Freq.	Linear Freq.	RBF All	Linear All
t	60	45	60	30
th_l	0.650	0.150	0.500	0.500
th_u	1.000	0.950	1.000	0.800
Sensitivity (train)	1.000	1.000	1.000	0.900
Specificity (train)	0.985	0.997	0.990	0.964
Data Usage (train)	0.783	0.541	0.701	0.858
Geometric Mean (train)	0.992	0.998	0.995	0.932
Accuracy (train)	0.985	0.997	0.990	0.963
Sensitivity (test)	0.910	0.884	0.964	0.884
Specificity (test)	0.979	0.993	0.989	0.972
Data Usage (test)	0.772	0.539	0.713	0.871
Geometric Mean (test)	0.944	0.937	0.976	0.927
Accuracy (test)	0.976	0.989	0.988	0.969

Table 6.5: Summarizes results in detecting tremor with the two-sided approach (i.e. variation 3).

Results: The results indicate that the condition with an RBF kernel and full feature set (see Table 6.5) performs best on the test dataset. Regardless of the chosen feature set, the best performing configurations with the RBF kernel were found to achieve a similar geometric mean as well as accuracy. The linear kernel with the reduced feature set does benefit from the third variation more than the linear kernel with the full feature set, but at the cost of a reduced data usage.

In general it can be noted that the two-sided approach outperforms the one-sided approach (see Table 6.5 and Table 6.4). This can be seen when comparing the geometric mean values for each of the conditions. On average the geometric mean for the two-sided approach is about 5.4% (sensitivity: 4.8%, specificity: 5.8%) higher than its corresponding one-sided condition. The thresholds in the one-sided approach were consistently found to be between the thresholds of its corresponding two-sided configuration.

The specificity for any of the four conditions was found to be above the 90% mark and the sensitivity was found to be above or close to the 90% threshold. On the training dataset, both sensitivity and specificity were consistently above this threshold. In terms of geometric mean, the RBF kernel (regardless of feature set) outperformed the conditions with a linear kernel. However, the condition with linear kernel and reduced feature set yielded to the best accuracy on the test dataset.

As far as data usage is concerned, it seems to be largely dependent on the chosen condition and the difference between the two thresholds (i.e. $th_u - th_l$). The one-sided approach does not allow for a gap between the th_l and th_u thresholds, consequently none of the data are rejected. For the two-sided approach, it can only be said that a larger gap between these thresholds yields to a decreased data usage (i.e. more windows are rejected). The improved geometric mean and accuracy (comparing the one-sided and two-sided approaches) come at the cost of a reduced data usage.

Conclusion: The conditions with an RBF kernel were superior to those with a linear kernel. A sensitivity and specificity above 90% were achieved and indicate an acceptable level of accuracy.

This section has also evaluated the effect of using frequency features against using, additionally, other non-frequency features previously reported in the literature (see Table 6.2 and Chapter 3). The results show that frequency features may be enough to obtain sensitivities and specificities above 90%, although data usage is penalized. However, in situations where only limited computational capabilities are available (e.g. microprocessor) the reduced feature set may already yield to suitable results. Depending on the application, the presented approach can be tuned towards a high sensitivity or specificity by selecting appropriate values for th_l and th_u . Furthermore, the amount of (usable) data can be controlled with these parameters.

At this point, the tremor detection algorithm may be considered finalized as sensitivity and specificity above 90% were achieved. The methodology is summarized in Figure 6.5.

6.1.4 Further Variations

The SVM's classification output in the form of a probability (rather than a class label) of belonging to the tremor class was evaluated. However, using thresholds at this low level did not yield to a significant improvement. This is in alignment with the frequently asked questions (FAQ) of libSVM, where this fact is stated as well.

Another noteworthy approach was aimed at detecting any kind of tremor (as opposed to just left-hand tremor) based on the acceleration data from the waist (as opposed to the previously used wrist sensor). When employing the third variation, the results are listed in the appendix (see Section C.1). The same analysis was also performed for detecting tremor in the upper extremities (i.e. left and right hand / arm) as well as in the lower extremities (i.e. left / right leg / foot). These results can also be found in the appendix.

6.2 Dyskinesia

The development of an algorithm for detecting dyskinesia is presented in the course of this section. Due to the similarity of dyskinesia to tremor (i.e. in terms of frequency characteristics), the same methodology that was used for detecting tremor is also applied here. However, a different set of features and different datasets are employed (i.e. more tuned toward dyskinesia and thus more specific to the problem).

Datasets are not changed throughout the remainder of this section. A training dataset is used for learning to distinguish dyskinesia from none dyskinesia. The same dataset is also used for parameter tuning (if any) and a test dataset is utilized for the final test results. More details on the datasets can be found in Table 6.6.

The training and test datasets are comprised of 13 and 172 recordings, respectively. There are no overlaps among the datasets. Furthermore, only labels with trunk dyskinesia are utilized. The test dataset is automatically relabeled in order to reduce the number of "undefined" labels and consequently increase

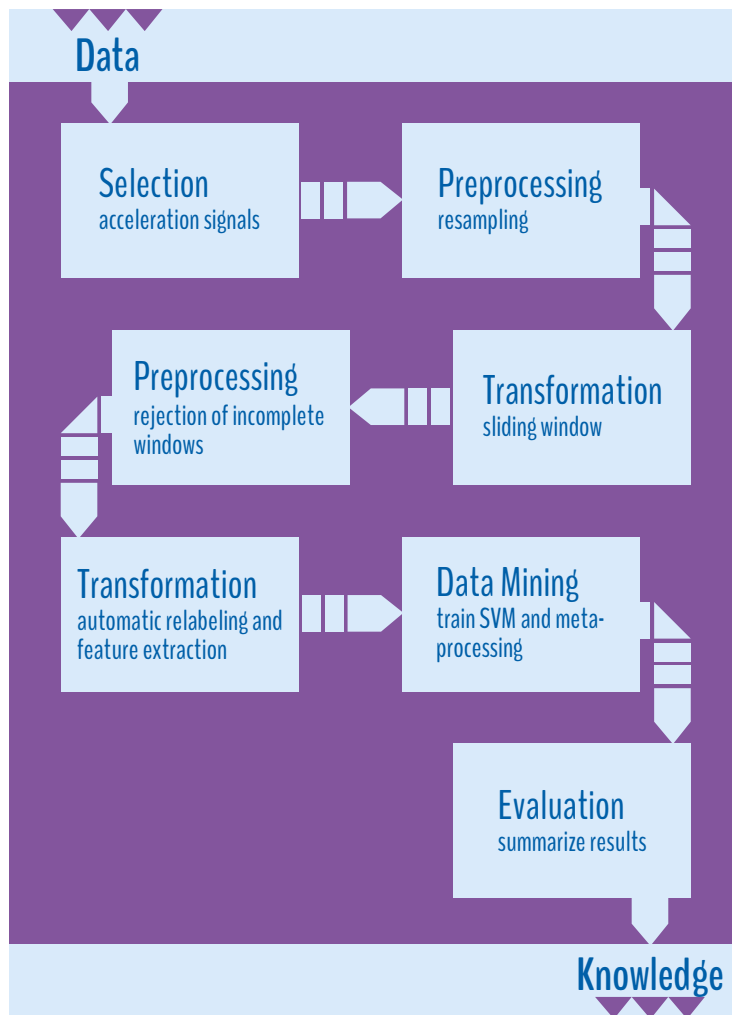


Figure 6.5: Shows the overall methodology for detecting tremor.

	Training	Test
Number of dyskinesia windows	631	2866
Number of non-dyskinesia windows	6528	102300
Number of recordings in ON state	7	82
Number of recordings in OFF state	6	90
Overall number of recordings	13	172

Table 6.6: Lists the number of windows (before aggregation) in each dataset that are used for signifying dyskinesia (i.e. trunk dyskinesia).

Indexes	Feature
1	FFT (raw, no filtering)
2	Mean and standard deviation of amplitude (band: 0.0-20.0 Hz)
3	Entropy of signal in time domain (band: 0.0-20.0 Hz)
4	Peak amplitude and it's frequency (band: 0.0-20.0 Hz)
5	Mean and standard deviation of amplitude (band: 1.0-4.0 Hz)
6	Entropy of signal in time domain (band: 1.0-4.0 Hz)
7	Peak amplitude and it's frequency (band: 1.0-4.0 Hz)
8	Mean and standard deviation of amplitude (band: 4.0-8.0 Hz)
9	Entropy of signal in time domain (band: 4.0-8.0 Hz)
10	Peak amplitude and it's frequency (band: 4.0-8.0 Hz)
11	Mean and standard deviation of amplitude (band: 8.0-20.0 Hz)
12	Entropy of signal in time domain (band: 8.0-20.0 Hz)
13	Peak amplitude and it's frequency (band: 8.0-20.0 Hz)
14	Histogram (bins: 0,1,2,3,...,15)

Table 6.7: Shows the full set of features used for dyskinesia detection. The reduced feature set is solely based on index 1.

the amount of usable data. Only those patients having “no dyskinesia” or not having “trunk dyskinesia”-labels were automatically relabeled as “no dyskinesia” (i.e. entire patient is relabeled accordingly). Only the dyskinesia part from patients suffering from dyskinesia was used (i.e. they provided the basis for true positives and false negatives; sensitivity). The remaining recordings provided the data for true negatives and false positives (i.e. specificity).

The utilized features are listed in Table 6.7. The features are split in the same way that tremor features were split into two sets: reduced and full feature set. Consequently, the reduced feature set is simply the result of an fast Fourier transform (FFT) while the full feature set includes additional features (e.g. peak, amplitude and frequency, mean amplitude, etc.) for several frequency ranges (e.g. 1-4, 4-8, etc.).

The final results (i.e. third variation) indicate that sensitivities and specificities above 80% as well as 90% are possible. However, specificity is generally above sensitivity and data usage is heavily penalized. The meta-analysis does improve results.

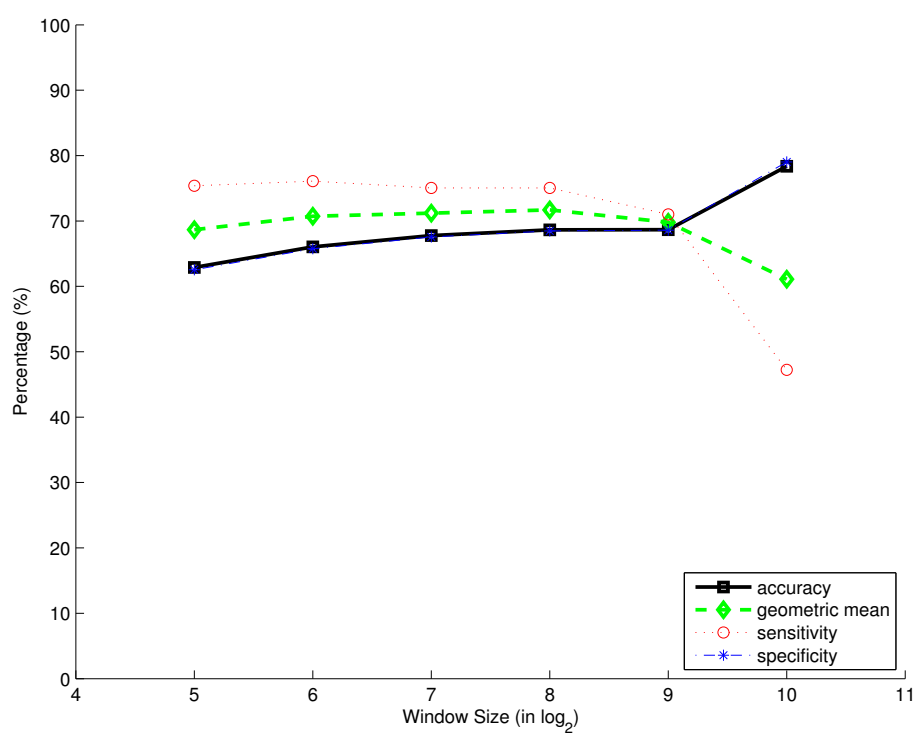


Figure 6.6: Illustrates the results of an evaluation for varying window sizes with respect to dyskinesia. For each window size, an SVM has been trained on the training dataset and evaluated on the test dataset.

6.2.1 Variation 1 to 3:

These variations are identical (in terms of their methodology and model selection) to those found in Section 6.1. Consequently, the methodology and model selection are only briefly outlined here. The most notable differences are that trunk dyskinesia is recognized (as opposed to “left hand / arm tremor”), that a different set of features is used (see Table 6.7) and that the sensor platform on the waist is used (formerly the wrist sensor was used).

Methodology: At first the acceleration signals are resampled to 40 Hz to which a sliding window of length ws is applied (overlap of 50%). For each window \mathbf{f}_i , a set of features is extracted (see Table 6.7), which are then used to train an SVM in distinguishing windows with trunk dyskinesia from those windows without dyskinesia. The classification is done in accordance with:

$$dysk_i^1 = \begin{cases} 0 & \text{no dyskinesia} & \text{if } f_{SVM} \leq 0 \\ 1 & \text{dyskinesia} & \text{if } f_{SVM} > 0 \end{cases} \quad (6.6)$$

where $f_{svm}(x) = \sum_{i=1}^l y_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b$, $\mathbf{x}_1, \dots, \mathbf{x}_l$ are the SVs. y_i and α_i are the corresponding label and lagrange multiplier of each SV. x and b are the classified data sample and the bias [39]. Again, the one in the superscript of $dysk_i^1$ identifies the first variation. Subsequent variations will have a higher index in the superscript.

A second variation aggregates n consecutive classified windows $dysk_1^1, \dots, dysk_n^1$ over a time frame t and determines a degree of confidence c_j . If a threshold th is reached then the aggregated time frame t is considered to exhibit trunk dyskinesia. Otherwise, dyskinesia is considered to be absent. Consequently, $dysk_j^2$ determines whether dyskinesia is present in n consecutive windows $dysk_j^1, \dots, dysk_{j+n-1}^1$ that start at the j^{th} window and cover a time frame t .

$$t = \frac{ws(n+1)}{40 * 2} \quad (6.7)$$

$$c_j = \sum_{i=j}^{j+n-1} \frac{dysk_i^1}{n} \quad (6.8)$$

$$dysk_j^2 = \begin{cases} 0 & \text{no dyskinesia} & \text{if } c_j < th \\ 1 & \text{dyskinesia} & \text{if } c_j \geq th \end{cases} \quad (6.9)$$

where $th, c_j \in [0, 1]; j, ws, n \in \mathbb{N}; j, ws, n > 0; t \in \mathbb{R}$ and $t > 0$.

The third variation utilizes two thresholds th_l and th_u instead of a single threshold th . They are employed as shown by $dysk_j^3$. This approach introduces the option for an “undefined” classification (i.e. neither “trunk dyskinesia” nor “no dyskinesia”), if the confidence value c is between the thresholds th_l and th_u . Starting at the j^{th} window, $dysk_j^3$ determines whether dyskinesia is present in the n following windows (covering a time frame t).

$$dysk_j^3 = \begin{cases} 0 & \text{no dyskinesia} & \text{if } c_j < th_l \\ 1 & \text{dyskinesia} & \text{if } c_j \geq th_u \\ -1 & \text{undefined} & \text{if } th_l \leq c_j < th_u \end{cases} \quad (6.10)$$

where $th_l \leq th_u; c_j, th_l, th_u \in [0, 1]; j \in \mathbb{N}; j > 0$

Model Selection: The optimal window size ws is determined in the same way that it has been determined for tremor (see Section 6.1). The SVM models are trained on the training dataset and the therein contained windows with their respective features. For the second and third variation, the training dataset is also queried in order to optimize the parameters t , th , th_l and th_u . Finally, the test dataset is used to determine the overall performance.

The following set of values were evaluated for the given parameters: $ws \in 2^{\{5, \dots, 10\}}$ samples; $t \in \{10, 15, 20, 25, 30, 45, 60\}$ seconds; $th, th_l \in \{0, 0.05, 0.1, \dots, 0.95, 1.0\}$ and $th_u \in (th_u \geq th_l | th_u \in \{0, 0.05, 0.1, \dots, 0.95, 1\})$. This was done for each of the four conditions (i.e. two kernels and two feature sets).

Results: The optimal window size ws was chosen to be 128 samples (i.e. 3.2 seconds length and 1.6 seconds overlap) as indicated by Figure 6.6. Here, any of the window sizes 2^6 , 2^7 and 2^8 would be expected to work equally well as all of them yielded to similar results. However, one may notice a slight decline in sensitivity as ws increases. At the same time specificity has slightly grown. Consequently, the window size of 128 samples was chosen as a compromise among geometric mean, accuracy and window length. The (low) percentages of geometric mean and accuracy indicate that the feature set may require refinement.

The Table 6.8 shows the results for the first variation (i.e. naive approach, no aggregation). It is apparent that the best performance (in terms of geometric mean and accuracy) is achieved for the linear SVM kernel. In fact it can be noted that the full feature set does provide a minimal overvalue for most of the three variations (see Table 6.8, Table 6.9 and Table 6.10). Their geometric mean and accuracy are slightly higher when compared to their corresponding configuration with the reduced feature set. The same applies to the linear kernel. It's geometric mean and accuracy are slightly better than their competitive configurations with an RBF kernel.

Aggregating the classification outputs over a period of time does improve the overall geometric mean by roughly 9.5% in comparison to the naive approach (see Table 6.8 and Table 6.9). It is notable that all thresholds are fairly close to the intuitive border of 50%. The threshold th of 50% suggests a roughly even number of FPs and false negatives (FNs). Furthermore, configurations of the second variation yield to similar parameter settings t and th for the reduced feature set. Again, the linear kernel achieved the highest geometric mean and accuracy. Adding a second threshold does further increase the overall geometric mean by about 9.0% (see Table 6.9 and Table 6.10).

The naive approach (first variation; see Table 6.8) yields to an average geometric mean of 71.2%. In all cases, the sensitivity is above the respective specificity. This observation suggests that an even greater number of FPs (compared to sensitivity) were raised by the SVM. In turn, this suggests that the selected features also trigger the SVM to classify data samples as dyskinesia where no dyskinesia is apparent or that there is a set of activities which are similar to dyskinesia. On the training dataset, the best results are obtained for the reduced feature set as opposed to the linear kernel on the test dataset. Furthermore, it is apparent that the discrepancy between training and test dataset with respect to sensitivity is much greater than those of specificity (sensitivity: 14.9%, specificity: 3.2%).

In the second approach the classification results are aggregated over time.

Kernel	RBF	Linear	RBF	Linear
Features	Freq.	Freq.	All	All
Sensitivity (train)	0.731	0.529	0.716	0.527
Specificity (train)	0.638	0.666	0.633	0.652
Data Usage (train)	1.000	1.000	1.000	1.000
Geometric Mean (train)	0.683	0.594	0.674	0.586
Accuracy (train)	0.642	0.660	0.637	0.647
Sensitivity (test)	0.739	0.782	0.770	0.811
Specificity (test)	0.657	0.695	0.676	0.688
Data Usage (test)	1.000	1.000	1.000	1.000
Geometric Mean (test)	0.697	0.737	0.721	0.747
Accuracy (test)	0.659	0.697	0.678	0.691

Table 6.8: Outlines results in recognizing dyskinesia with the naive approach (i.e. variation 1). Various measures are listed for both datasets.

Figure 6.7 shows the geometric mean that is observed on the training dataset for all parameter combinations. It is apparent that a threshold around 50%-60% works well for all four configurations. Above the threshold of 50% the geometric mean ebbs off rapidly in case of the linear SVM kernel. For the RBF kernel, the geometric mean peaks above the 50% threshold and then slowly declines. These thresholds are also reflected in Table 6.9, where the results for variation two are listed.

As far as the second approach is concerned, sensitivity did not improve as much as specificity (0.8% and 15.3%, respectively). The accuracy notably increased with specificity. On the training, the RBF kernel outperforms the linear one (as it has been in the naive approach). The overall discrepancy between training and test datasets with respect to sensitivity and specificity did not dramatically change (sensitivity: 14.8%, specificity: 3.7%).

With the third approach, the full feature set with the RBF kernel performs best for most measures in both datasets (i.e. train and test). The discrepancy between training and test datasets with respect to sensitivity and specificity has reduced (sensitivity: 1.7%, specificity: 6.0%). This is most likely due to the reduced data usage on both datasets. All thresholds th_l and th_u were found to encapsulate the thresholds th of variation two. All conditions reached acceptable results in terms of accuracy and geometric mean. However, the data usage is heavily penalized which suggests a high number of FPs and FNs. Nonetheless, the accuracy and specificity were found to be above the 90% mark in all cases. In fact, those conditions with a linear kernel also achieved a sensitivity close to 90% whereas conditions with an RBF kernel reached a sensitivity around 80%.

Conclusion: The full feature set does provide a minimal overvalue. In general, the idea of aggregating classification output does seem to be beneficial. With each variation the average geometric mean rises. The second variation increases the average geometric mean by 9.5% and the third variation further raises the geometric mean by another 9.0%. Despite the low data usage, 90% sensitivity and specificity have been reached. Consequently, the development may be considered as finalized within the framework of this thesis. However, future refinements of this approach should further investigate the low data usage

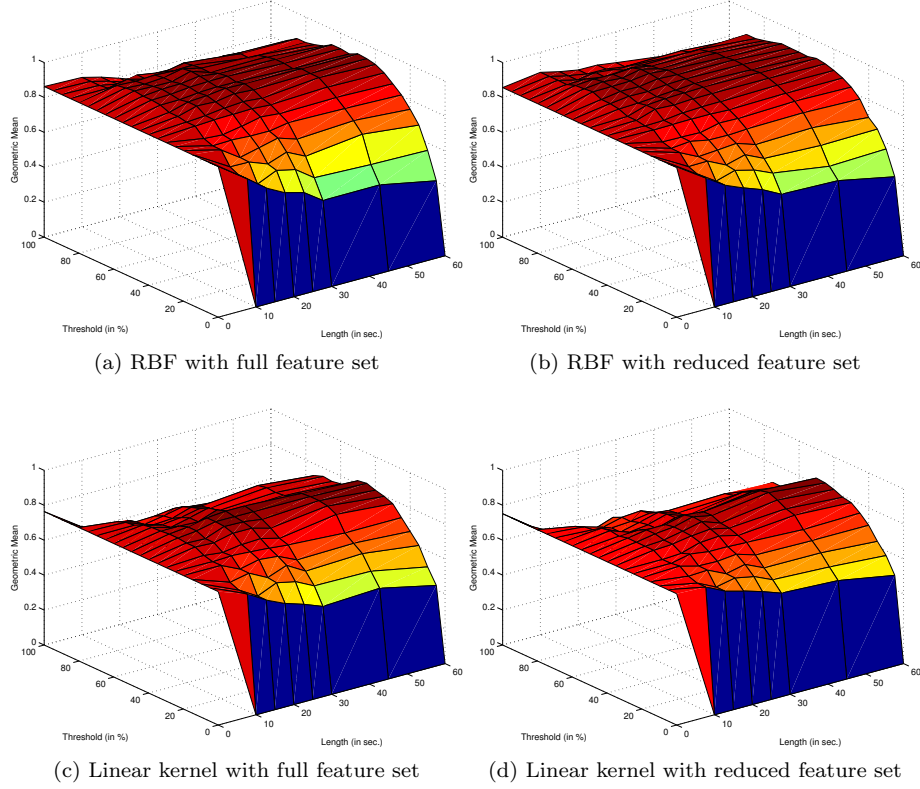


Figure 6.7: Indicates the effect of window aggregation t and threshold th on geometric mean. The results are shown for all four conditions.

Kernel Features	RBF Freq.	Linear Freq.	RBF All	Linear All
t	60	60	45	20
th	0.600	0.650	0.550	0.600
Sensitivity (train)	0.769	0.500	0.735	0.536
Specificity (train)	0.809	0.856	0.746	0.770
Data Usage (train)	1.000	1.000	1.000	1.000
Geometric Mean (train)	0.789	0.654	0.741	0.642
Accuracy (train)	0.806	0.827	0.745	0.756
Sensitivity (test)	0.714	0.747	0.798	0.874
Specificity (test)	0.823	0.880	0.810	0.815
Data Usage (test)	1.000	1.000	1.000	1.000
Geometric Mean (test)	0.767	0.811	0.804	0.844
Accuracy (test)	0.819	0.875	0.810	0.817

Table 6.9: Summarizes results in recognizing dyskinesia with the one-sided approach (i.e. variation 2).

Kernel Features	RBF Freq.	Linear Freq.	RBF All	Linear All
t	60	25	60	30
th_l	0.300	0.100	0.150	0.150
th_u	0.900	1.000	0.950	0.900
Sensitivity (train)	0.909	0.733	1.000	0.692
Specificity (train)	0.893	0.872	0.915	0.865
Data Usage (train)	0.525	0.359	0.272	0.359
Geometric Mean (train)	0.901	0.800	0.956	0.774
Accuracy (train)	0.894	0.864	0.921	0.855
Sensitivity (test)	0.791	0.889	0.815	0.905
Specificity (test)	0.958	0.931	0.965	0.932
Data Usage (test)	0.522	0.341	0.272	0.354
Geometric Mean (test)	0.870	0.909	0.887	0.918
Accuracy (test)	0.953	0.929	0.960	0.931

Table 6.10: Summarizes results in detecting dyskinesia with the two-sided approach (i.e. variation 3).

which indicates that detecting dyskinesia with the selected features is difficult (i.e. a different set of features might be more appropriate). All in all, the methodology is the same as illustrated in Figure 6.5.

6.2.2 Further Variations

Several other variations have also been evaluated and yielded to noteworthy results. These variations include employing the final tremor detection algorithm to recognize varying kinds of dyskinesia from the acceleration signals at the waist. On the one hand, any kind of dyskinesia was recognized, and on the other hand, any kind of trunk dyskinesia as well as (strong) limb / head dyskinesia. The corresponding results can be found in the appendix (see Section C.2) as well as details on the datasets. The datasets themselves did not change, only a different set of labels was used.

6.3 Akinesia / Freezing of Gait

This section outlines the development of an algorithm for indicating freezing episodes in PD patients. The initial approach closely resembles the final algorithm that is used to recognize tremor (see Section 6.1). As such, episodes of freezing of gait (FoG) are first tried to be directly detected by an SVM. This approach is refined to include a meta-analysis for avoiding false positives and false negatives. However, the volatile nature of FoG episodes must be considered during the development. In contrast to tremor (or dyskinesia for that matter) episodes of FoG do not last for prolonged periods of time which may emphasize the importance of the chosen window size. In any case, the contents of the database are split into two datasets (i.e. training and testing) each of which serve the same purpose as described in earlier sections (i.e. training an SVM as well as optimizing additional parameters and testing). As noted earlier, the

datasets stay the same for all approaches within this section. Details of the datasets are listed in Table 6.11.

The entire database contains recordings of 92 patients, while the individual datasets hold 15 and 5 patients for the training and testing dataset, respectively. At this point, it should be noted that not all available recordings of freezers (i.e. patients with FoG episodes) and non-freezers were utilized. Instead, a subset of those recordings is employed. This is largely due to the fact that episodes of FoG were especially difficult to label and verify (mostly due to their volatility). As a consequence only a subset of those recordings is used that could actually be verified. The recordings are identical to those employed by Rodríguez-Martín et al. [137, 138].

As far as the actual labeling is concerned, the presence of any type of freezing (e.g. start, turn, end, etc.) is considered to be an episode of FoG. As such, the labeling, as it is introduced in Chapter 5 is simplified. The detection of individual types of freezing requires additional contextual information which is not contained within the DB. Furthermore, such a fine granularity might not provide much of an overvalue (e.g. to a PD monitoring system). The fact that a freezing episode is happening is more relevant than the actual type of episode.

The same type of automatic relabeling has been employed as it was done in previous sections. Those patients without any freezing episodes were relabeled in such a way that all available recordings could be used rather than just those parts that were specifically labeled as “no freezing”. On the other hand, recordings of freezers were cut to the point where only “FoG” labels remained. This reduced the overall amount of data but ensured that no freezing episodes (which might not have been properly labeled) were used. Consequently, sensitivity is determined by patients with freezing episodes while specificity is determined by none-freezing patients.

Two feature sets are evaluated: a reduced feature set with only the FFT and a full feature set with various additional features. The effect of adding these additional features is quantified during the course of this section. These features are comprised of the freezing index [31] as well as some frequency related features for differing frequency ranges [106]. Details are listed in Table 6.12.

Results of the second and third variation indicate that sensitivities and specificities well above the 90% mark are possible. The meta-analysis does improve results on the test dataset. However whether a one-sided or two-sided threshold is best employed depends on the utilized SVM kernel.

6.3.1 Variation 1 to 3:

For starters, the final approach in detecting tremor in PD patients is employed. However, the feature sets and labels are tuned toward freezing of gait episodes (as opposed to tremor). As the methodology and model selection are almost identical to those found in Section 6.1, they are briefly outlined. After all, the general idea is to start with a familiar approach and then refining that methodology to the point where suitable results are obtained.

Methodology: At first varying window sizes ws are evaluated such that freezing episodes can be captured as good as possible. The comparison of different window sizes is done on an episode level (rather than a data sample or window level, which was the case in the previous approaches). An episode of FoG is

detected when at least one window within an actual FoG episode is classified as such. As far as non-freezing episodes are concerned, an aggregation of windows over a period of time that corresponds to the average length of a FoG episode (plus twice the standard deviation) is performed. The data are resampled to 40Hz and split into unisized chunks of data with a certain length ws (and 50% overlap). These windows are then used to extract features which in turn are fed to an SVM for training and classification. Consequently, the following equation determines whether FoG is apparent in a window \mathbf{f}_i . This represents the first and naive variation.

$$fog_i^1 = \begin{cases} 0 & \text{no freezing} & \text{if } f_{SVM} \leq 0 \\ 1 & \text{freezing} & \text{if } f_{SVM} > 0 \end{cases} \quad (6.11)$$

where $f_{svm}(x) = \sum_{i=1}^l y_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b$, $\mathbf{x}_1, \dots, \mathbf{x}_l$ are the SVs. y_i and α_i are the corresponding label and lagrange multiplier of each SV. x and b are the classified data sample and the bias [39]. Again, the one in the superscript of fog_i^1 identifies the first variation. Subsequent variations will have a higher index in the superscript.

The second variation aggregates the SVMs' outputs fog_j^1 over a time period t and calculates the degree of confidence c_j . If the confidence value exceeds a threshold th then the aggregated time frame t is considered to be an episode of FoG, otherwise not. The result of fog_j^2 determines whether a freezing event occurred in the time frame t starting at the j^{th} window.

$$t = \frac{ws(n+1)}{40 * 2} \quad (6.12)$$

$$c_j = \sum_{i=j}^{j+n-1} \frac{fog_i^1}{n} \quad (6.13)$$

$$fog_j^2 = \begin{cases} 0 & \text{no freezing} & \text{if } c_j < th \\ 1 & \text{freezing} & \text{if } c_j \geq th \end{cases} \quad (6.14)$$

where $th, c_j \in [0, 1]; j, ws, n \in \mathbb{N}; j, ws, n > 0; t \in \mathbb{R}$ and $t > 0$.

The third variation introduces a second threshold. The two thresholds th_l and th_u can then be used to tune sensitivity and specificity separately. One threshold (th_l) sets the maximum confidence value for "no freezing" periods and the second threshold (th_u) sets the minimum confidence value for freezing episodes. By not requiring that these thresholds need to be equal, the final output of the algorithm may indicate the presence of freezing as well as "undefined" (when the confidence value is between the two thresholds). Consequently, some aggregated windows may be ignored and data usage is lowered. The equation fog_j^3 defines when a freezing event is considered to be present in a time frame t starting at the j^{th} window.

$$fog_j^3 = \begin{cases} 0 & \text{no freezing} & \text{if } c_j < th_l \\ 1 & \text{freezing} & \text{if } c_j \geq th_u \\ -1 & \text{undefined} & \text{if } th_l \leq c_j < th_u \end{cases} \quad (6.15)$$

where $th_l \leq th_u; c_j, th_l, th_u \in [0, 1]; j \in \mathbb{N}; j > 0$

Model Selection: The individual SVM models are trained with the features that were extracted from the training dataset. For the second and third variation, the individual parameters t , th , th_l and th_u are also optimized on the training dataset. The final results are then obtained from the testing dataset.

The following discrete values have been evaluated: $ws \in 2^{\{5,6,7,8\}}$ samples, $t \in \{10, 15, 20, 25, 30, 45, 60\}$ seconds, $th = th_l \in \{0, 0.05, 0.1, \dots, 0.95, 1.0\}$ and $th_u \in (th_u \geq th_l | th_u \in \{0, 0.05, 0.1, \dots, 0.95, 1\})$. The appropriate values and parameters were evaluated for each of the four conditions (two kernels and two feature sets).

Results: The average length of a FoG episode is 3.48 seconds (std ± 3.29 seconds). Figure 6.8 shows several measures for varying window sizes (i.e. sensitivity, specificity, geometric mean and accuracy). There it can be seen that the best values for those measures are achieved with a window size of 128 samples (i.e. 2^7 samples). Accuracy and geometric mean are closest at this level. This window size is also closest to the average length of a freezing episode. Consequently, this window size is utilized during the remainder of this section.

On the training dataset, the reduced and full feature sets yield to a similar geometric mean regardless of the employed SVM (see Table 6.13). This, however, diverges on the test dataset. The RBF kernel seems to benefit from the reduced feature set while the linear kernel favors the full feature set. Acceptable levels of specificity are consistently achieved on the test dataset, while sensitivity is reduced by FNs. Latter may be counteracted when windows are aggregated. Nonetheless, accuracies above 90% were consistently reached.

The impact of window aggregation t and threshold th are highlighted in Figure 6.9. For all conditions, the geometric mean of all parameter combinations are shown. The sub-figures indicate that a threshold close to 50% works best in all cases, suggesting an even distribution of FPs and FNs. Furthermore, the greater the aggregation level the greater the geometric mean.

After adding the meta-analysis in variation two (see Table 6.14), the geometric mean of the test dataset does increase by 9.4% (on average). This suggests that the procedure is in fact beneficial for detecting FoG episodes. Furthermore, all conditions yielded to a threshold close to the intuitive border of 50%, which is consistent with the observations in Figure 6.9. Also the aggregation period t is the same across all four conditions.

Having optimized parameters t and th on the training dataset, the results on the testing dataset are rather well. All conditions achieve a high specificity of 98% or greater and most conditions also reached a sensitivity of 90% or above for an aggregation period of 60 seconds. Regardless of the kernel and feature set reasonable results were achieved.

The results in Table 6.15 are those of the third variation. Here most conditions still favor an aggregation level of 60 seconds. The lower and upper thresholds th_l and th_u were consistently found to enclose the previously found thresholds th in the second approach (see Table 6.14). Allowing for two thresholds increased sensitivity and specificity values on the test dataset for the linear kernel. However, the RBF kernel did not benefit from this approach (in terms of geometric mean). Now all conditions yielded to a sensitivity of roughly 90% and a specificity well above the 90% mark. This comes at the cost of a slightly reduced data usage although it is still above 90% for the most part.

	Training	Test
Number of freezing windows	93	45
Number of non-freezing windows	3883	2312
Number of recordings in ON state	0	0
Number of recordings in OFF state	15	5
Overall number of recordings	15	5

Table 6.11: Lists the number of windows (before aggregation) in each dataset that are used for detecting FoG.

Indexes	Feature
1	FFT (raw, no filtering)
2	Mean and standard deviation of amplitude (band: 0.5-3.0 Hz)
3	Entropy of signal in time domain (band: 0.5-3.0 Hz)
4	Peak amplitude and it's frequency (band: 0.5-3.0 Hz)
5	Mean and standard deviation of amplitude (band: 3.0-8.0 Hz)
6	Entropy of signal in time domain (band: 3.0-8.0 Hz)
7	Peak amplitude and it's frequency (band: 3.0-8.0 Hz)
8	Freezing Index

Table 6.12: Shows the full set of features used for FoG detection. The reduced feature set is comprised of index 1.

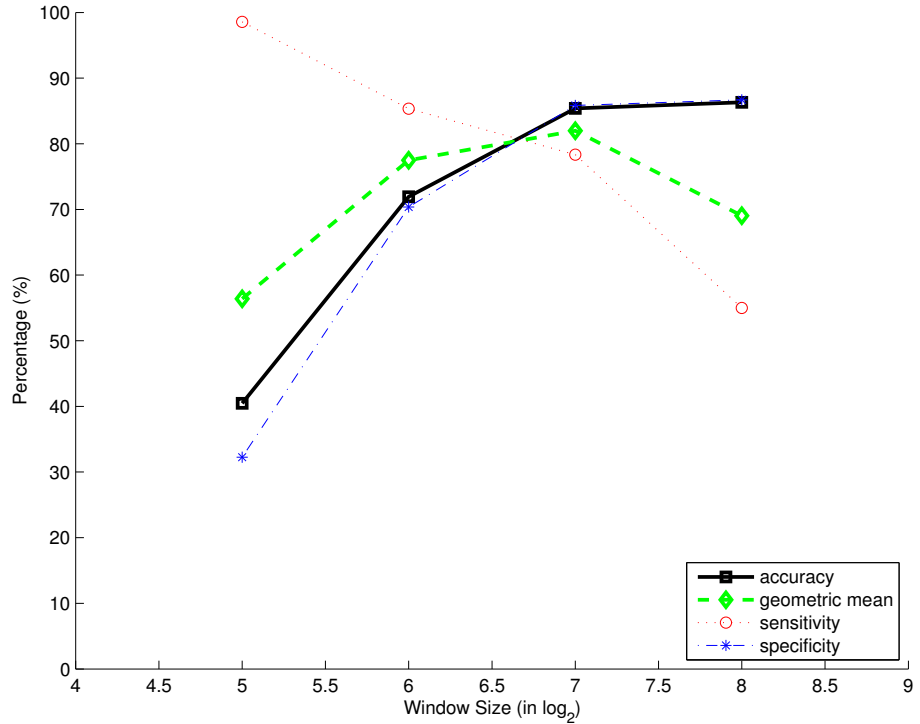


Figure 6.8: Illustrates the results of an evaluation for varying window sizes with respect to freezing episodes. For each window size, an SVM has been trained on the training dataset and evaluated on the test dataset.

Kernel Features	RBF Freq.	Linear Freq.	RBF All	Linear All
Sensitivity (train)	0.946	0.903	0.946	0.925
Specificity (train)	0.860	0.903	0.901	0.932
Data Usage (train)	1.000	1.000	1.000	1.000
Geometric Mean (train)	0.902	0.903	0.924	0.928
Accuracy (train)	0.862	0.903	0.902	0.932
Sensitivity (test)	0.822	0.667	0.711	0.822
Specificity (test)	0.928	0.949	0.946	0.956
Data Usage (test)	1.000	1.000	1.000	1.000
Geometric Mean (test)	0.873	0.796	0.820	0.887
Accuracy (test)	0.926	0.944	0.942	0.954

Table 6.13: Outlines results in recognizing FoG with the naive approach (i.e. variation 1). Various measures are listed for both datasets.

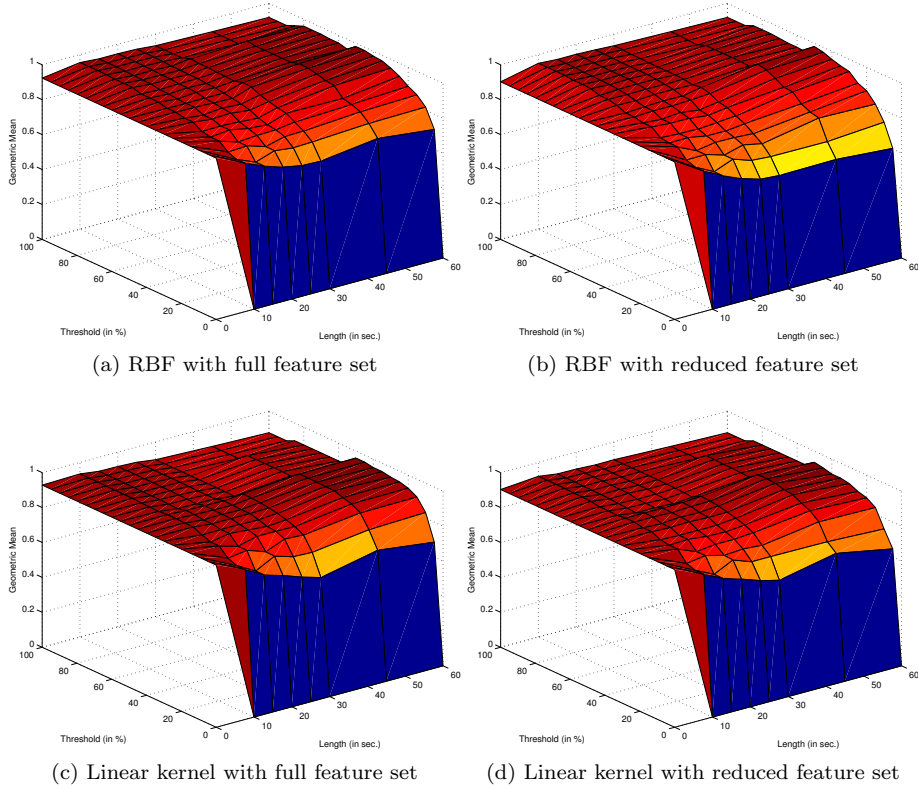


Figure 6.9: Indicates the effect of window aggregation t and threshold th on geometric mean. The results are shown for all four conditions.

Kernel Features	RBF Freq.	Linear Freq.	RBF All	Linear All
t	60	60	60	60
th	0.400	0.500	0.400	0.450
Sensitivity (train)	1.000	0.885	1.000	0.923
Specificity (train)	0.911	0.991	0.946	1.000
Data Usage (train)	1.000	1.000	1.000	1.000
Geometric Mean (train)	0.954	0.936	0.973	0.961
Accuracy (train)	0.928	0.971	0.957	0.986
Sensitivity (test)	0.923	0.769	0.923	0.923
Specificity (test)	1.000	0.985	1.000	1.000
Data Usage (test)	1.000	1.000	1.000	1.000
Geometric Mean (test)	0.961	0.870	0.961	0.961
Accuracy (test)	0.987	0.949	0.987	0.987

Table 6.14: Summarizes results in recognizing FoG with the one-sided approach (i.e. variation 2).

Kernel Features	RBF Freq.	Linear Freq.	RBF All	Linear All
t	60	45	60	60
th_l	0.150	0.250	0.300	0.350
th_u	0.900	0.800	0.800	0.500
Sensitivity (train)	1.000	0.923	1.000	0.923
Specificity (train)	1.000	1.000	1.000	1.000
Data Usage (train)	0.696	0.891	0.906	0.986
Geometric Mean (train)	1.000	0.961	1.000	0.961
Accuracy (train)	1.000	0.987	1.000	0.985
Sensitivity (test)	0.900	0.889	0.900	0.923
Specificity (test)	1.000	1.000	1.000	1.000
Data Usage (test)	0.823	0.919	0.949	0.987
Geometric Mean (test)	0.949	0.943	0.949	0.961
Accuracy (test)	0.985	0.989	0.987	0.987

Table 6.15: Summarizes results in detecting FoG with the two-sided approach (i.e. variation 3).

Conclusion: In general, it can be noted that the second and third approach yielded to satisfactory results. Although the linear and RBF kernel do not benefit equally from the third approach, combining the results from both variations (i.e. second and third variation) shows promising results. While the RBF kernel achieved a geometric mean greater than 95.0% and an accuracy greater than 98% with the second approach, the linear kernel reached similar levels (close to 95.0% geometric mean and 98% accuracy) with the third approach. However, in latter case the data usage is slightly penalized. The findings suggest that the full feature set is not required for satisfactory results. Instead, even a linear kernel that has been trained with a result of an FFT can already be enough to accurately detect FoG episodes within periods of 45 seconds.

The overall methodology is very similar to the methodology of the tremor algorithm and it is summarized in Figure 6.5.

6.4 Summary

The final algorithms, that have been employed for detecting Parkinsonian tremor, dyskinesia as well as FoG episodes, are almost identical. All of them consider two SVM kernels (i.e. RBF and linear) and two feature sets (i.e. reduced and full feature set). Only the full feature set differed in the approaches (and of course the utilized datasets are different). The initial and naive approach of directly detecting PD motor symptoms was iteratively refined to a point where sensitivities and specificities above 90% were reached. Improved results were observed after aggregating classification outputs of the naive approach. The results were further improved once a second threshold was introduced. Although, the second threshold did increase geometric mean as well as accuracy, it did also penalize data usage by rejecting data samples, that could not clearly be identified as a motor symptom but neither could be identified as no motor symptom.

The proposed algorithms penalize data usage to achieve high sensitivity and specificity. However, they are configurable in such a way that they consider more data but at the cost of decreasing sensitivity or specificity. The algorithm is flexible enough to provide a tradeoff and parameters can be tailored to the needs of a particular application. Consequently, a balance between high sensitivity, high specificity and high data usage can be configured. If necessary, the algorithm's aggregation level can also be adjusted to control the degree of real-time with which the algorithm produces classification results.

Chapter 7

Benchmark of Symptom Detecting Algorithms

Chapter 3 and Chapter 6 introduced several algorithms for detecting Parkinson's Disease (PD) motor symptoms. This chapter compares the performance of these algorithms. Thus, most of this chapter is devoted to answering the fourth research question (see Section 1.2). The benchmarks are first described. Afterwards, their results are highlighted and discussed.

The following sections are devoted to the proposed algorithms from the preceding chapter (see Chapter 6). The results for each of the symptoms are compared to those found in the related work (see Chapter 3). However, only the final variations of the proposed approaches are compared to the state-of-the-art techniques in terms of sensors, number of patients, sensitivity, specificity and accuracy whenever possible. Nonetheless, results for the full and reduced feature set as well as the linear and radial basis function (RBF) kernel are outlined.

7.1 Tremor (at Rest)

A new methodology to detect tremor in PD patients through a single wrist-worn accelerometer has been proposed. The approach has been tested on a database of signals obtained, for the first time, from a relevant sample of 89 patients while, furthermore, they were at home. The presented approach consists in a meta-analysis applied to the output of a support vector machine (SVM) that detects tremor in a sliding time window. It provides slightly better performances than previous methods presented in the literature. However, these results were obtained on a database (DB) that is much larger (i.e. 4 to 8 times as many participants) than the DBs used in related works. Table 7.1 provides a summary of tremor algorithms. It shows related works as well as the newly proposed algorithm. In contrast to previous works, this approach is tested with signals that were obtained in a non-laboratory environment and originate from a large cohort of PD patients.

The accuracies by Zwartjes et al. [182] and Rigas et al. [136] were clearly outperformed by all four configurations (see Table 7.1). This is also true for the one-sided approach that has been previously described (see Table 6.4). Zwartjes et al. [182] use four inertial sensors on 13 subjects (6 PD patients and 7 controls).

Variation / Author(s)	Acc.	Sens.	Spec.	D.U.	Subjects
Salarian et al. [142]		0.766	0.980		20
Salarian et al. [143]		0.995	0.942		20
Zwartjes et al. [182]	0.847				13
Rigas et al. [136]	0.870				23
Cole et al. [36]		0.930	0.950		12
Roy et al. [140]		0.912	0.934		23
Niazmand et al. [114]		0.800	0.985		12
RBF+red.	0.976	0.910	0.979	0.772	89
linear+red.	0.989	0.884	0.993	0.539	89
RBF+full	0.988	0.964	0.989	0.713	89
linear+full	0.969	0.884	0.972	0.871	89

Table 7.1: Summarizes results from various algorithms for detecting Parkinsonian tremor. Results found in Chapter 3 and Chapter 6 are briefly summarized. “D.U.” is used as a shorthand for “data usage”. More details can be found in Table A.4.

Their results indicate 84.5% accuracy in detecting resting tremor (in the arm) while subjects are in a sitting position (79.1% in the thigh). In contrast, 94.1% accuracy is achieved for resting tremor in the arms while subjects were standing (90.1% in the thigh). All of these results are outperformed.

In contrast to this work, Rigas et al. [136] use 6 accelerometers and 2 hidden markov models (HMMs) to classify resting tremor. Their results are based on 23 subjects (18 PD patients and 5 control subjects) while the proposed work makes use of 89 patients. One HMM classifies action / posture tremor while the second HMM assesses tremor. The output of the latter HMM yields to 87% accuracy in classifying four tremor severities (i.e. *S-0* and *other*, *S-1*, *S-2* and *S-3*). When combining the outputs of these HMMs to form a unified result, the accuracy drops to 74% (in discriminating resting tremor from action / posture tremor with their respective severities). However, Rigas et al. [136] achieve a high specificity for *S-1* (95%) with respect to other activities and symptoms (i.e. dyskinesia, bradykinesia and freezing of gait (FoG)). For *S-2* and *S-3*, their methodology does separate tremor perfectly from those other activities and symptoms.

In the approach by Salarian et al. [142] (i.e. without meta-analysis), their sensitivity is exceeded by all four configurations while the specificity is very similar in all cases. After adding the meta-analysis [143], their specificity is consistently outperformed while sensitivity of the new approach stays below their results (max. sensitivity 96.4% for RBF+full and 99.5% for Salarian et al. [143]). The approach by Salarian et al. [142] relies on 20 subjects (10 PD patients and 10 control subjects) and a single tri-axial gyroscope. Their results indicate an overall specificity of 98% (min. 94% and max. 100%) and an overall sensitivity of 86.6% (i.e. 70.8% pitch, 84.8% roll and 74.1% yaw). A similar level of specificity is achieved. However their sensitivity is consistently exceeded. In a later study by Salarian et al. [143] on the same dataset, they achieve a much higher sensitivity. As in their previous study, the authors use 3 second windows. Their results indicate a specificity of 94.2% for all axes and 99.5% sensitivity. These results are outrivaled in terms of specificity.

The results by Cole et al. [36] are exceeded by the RBF kernel with the full feature set. However, all other configurations yielded to a higher specificity even though the sensitivity is not exceeded. In contrast to an SVM, Cole et al. [36] use a dynamic neural network (DNN) in a study with 12 subjects (8 PD patients and 4 control subjects). Several surface electromyograph (EMG) sensors and accelerometers are used to detect tremor in 1-second windows. On the contrary, the proposed approach (of this work) makes use of higher timeframes (e.g. 60 seconds), but requires only a single tri-axial accelerometer. Additionally, when the approach by Cole et al. [36] is applied to their test dataset (i.e. 2 PD patients with 5 minutes of tremor each) then a 89.5% sensitivity and 92.5% specificity are achieved. These results are outperformed in most cases except for the sensitivity of the conditions with a linear kernel.

Sensitivity and specificity by Roy et al. [140] and Niazmand et al. [114] did not consistently exceed the new approach (in case of the RBF kernel). The algorithm by Niazmand et al. [114] does yield to a similar level of specificity, but their sensitivity is lower when compared to the new approach. The authors use a pullover with 8 integrated accelerometers to assess the severity of postural and resting tremor. They make use of data from 12 subjects (10 PD patients and 2 controls). Their sensitivity (i.e. resting tremor: 71%, postural tremor: 89%) is almost always exceeded while their specificity for postural tremor (97%) is consistently outperformed. However, their specificity for resting tremor (100%) is not reached by the proposed approach (in this work). Nonetheless, their dataset are not quite as comprehensive in terms of size as in this work.

Roy et al. [140] achieved a better sensitivity than those configurations with a linear kernel. However, their specificity is always exceeded by the new approach. The results by Roy et al. [140] are based on four surface EMG sensors as well as four tri-axial accelerometers. They use data from 23 subjects (i.e. 19 PD and 4 control subjects). They classify at a 1 second resolution, but they use less data recordings for testing (i.e. 4 controls and 8 PD patients) and more sensors. Here, DNNs are employed to classify motor symptoms as well as rudimentary actions / postures which are then combined to a final outcome. On their test dataset, Roy et al. [140] achieve 93.8% sensitivity and 91.9% specificity for the sensors on the arms. The specificity is consistently outperformed whereas the sensitivity is only exceeded by the RBF kernel with the full feature set.

To sum up, the presented approaches slightly outperform those found in the literature although, qualitatively, results are comparable since roughly similar accuracies or geometric means are provided. However, results presented in this thesis have not only been attained with a larger dataset (i.e. 89 patients rather than 12 to 23 patients) but also the data have been recorded in a real-world environment (i.e. at the patients' home / apartment). Furthermore, the proposed approach is patient-independent, only requires a single tri-axial accelerometer and can be configured to attain a higher sensitivity or specificity.

7.2 Dyskinesia

The same methodology that has been used to detect tremor at rest in PD patients can also be used to recognize dyskinesia events. The approach has been verified with 172 recordings (trained with 13 recordings) of a single waist-mounted tri-axial accelerometer. Recording sessions took place in the patient's

Variation / Author(s)	Acc.	Sens.	Spec.	D.U.	Subjects
Keijsers et al. [78]	0.968				13
Tsipouras et al. [166]	0.937				10
Cole et al. [36]		0.910	0.930		12
Roy et al. [140]		0.900	0.934		23
RBF+red.	0.953	0.791	0.958	0.522	90
linear+red.	0.929	0.889	0.931	0.341	90
RBF+full	0.960	0.815	0.965	0.272	90
linear+full	0.931	0.905	0.932	0.354	90

Table 7.2: Summarizes results from various algorithms for detecting dyskinesia. Results found in Chapter 3 and Chapter 6 are briefly summarized. “D.U.” is used as a shorthand for “data usage”. More details can be found in Table A.6.

home / apartment. Table 7.2 lists relevant publications from the related work (see Chapter 3). It summarizes their results and shows the results of the proposed methodology.

The accuracy obtained by Keijsers et al. [78] does exceed the results by the proposed approach. However, the RBF kernel with the full feature set achieves a similar result. The authors use an neural network (NN) to recognize dyskinesia in segments of 15 minutes. In doing so they achieve an accuracy of 93.7%, 99.7% and 97.0% for the arms, trunk and legs, respectively. Six tri-axial accelerometers were used in their study with 13 subjects (all PD patients with dyskinesia). For 1 minute segments (rather than 15 minute segments), an accuracy of 77.0%, 83.0% and 76.9% were measured (i.e. arms, trunk and legs, respectively). This segmentation length is in fact more comparable to the proposed approach as both use a similar length. Comparing these results, the proposed methodology outperforms their approach not just in terms of accuracy but also in terms of number of sensors.

The approach by Tsipouras et al. [166] is outperformed by the RBF kernel (regardless of feature set). The linear kernel does achieve slightly lower results. The authors used recordings from 10 subjects (i.e. 4 PD patients with dyskinesia, 3 PD patients without dyskinesia and 3 healthy control subject) to detect dyskinesia in 2 second windows. They employed a set of 6 tri-axial accelerometers and 2 tri-axial gyroscopes. Considering their results from the left wrist and waist acceleration signals, an accuracy of 92.7% and 93.1% (respectively) was achieved.

The sensitivity achieved by Cole et al. [36] is consistently above the sensitivity achieved by the proposed approach. However, specificity of the proposed approach exceeded their specificity. They utilized a dataset that is comprised of 12 subjects (i.e. 8 PD patients and 4 control subjects). A surface EMG and acceleration sensor were attached to the subjects wrist and their measurements were recorded while the subjects performed unscripted activities. Applying their approach to their test dataset (2 PD patients with 5 minutes of dyskinesia each), an average sensitivity of 94.6% and specificity of 94.3% is achieved. Contrary to their work, a single tri-axial accelerometer (on the waist) is used to detect dyskinesia by the proposed methodology.

The specificity by the approach from Roy et al. [140] is below the specificity of the proposed approach with the RBF kernel. As for the linear kernel, sim-

Variation / Author(s)	Acc.	Sens.	Spec.	D.U.	Subjects
Djurić-Jovičić et al. [51]	0.840				4
Cole et al. [37]		0.829	0.973		12
Niazmand et al. [115]		0.883	0.853		6
Bächlin et al. [32]		0.731	0.816		10
RBF+red.	0.985	0.900	1.000	0.823	20
linear+red.	0.989	0.889	1.000	0.919	20
RBF+full	0.987	0.900	1.000	0.949	20
linear+full	0.987	0.923	1.000	0.987	20

Table 7.3: Summarizes results from various algorithms for detecting FoG. Results found in Chapter 3 and Chapter 6 are briefly summarized. “D.U.” is used as a shorthand for “data usage”. More details can be found in Table A.5.

ilar specificities are achieved. Sensitivity by Roy et al. [140] does exceed the results by the proposed approach. These authors also use a surface EMG and an acceleration sensor on the patients arm. Here a DB of 23 subjects (i.e. 19 PD patients and 4 control subjects) were used to train (11 patients) and test (8 patients and 4 controls) a DNN. The DNN was used in combination with the Integrated Processing and Understanding of Signals (IPUS) framework to determine motor and mobility outcomes. An evaluation by Roy et al. [140] indicates that mild and severe dyskinesia can be detected with 93.9% sensitivity and 95.5% specificity (mild dyskinesia) as well as 95.0% sensitivity and 98.6% specificity. Almost all of these measures exceed those by the proposed methodology.

In general, the newly proposed approach achieves similar results and does not exceed those found in the literature. However, data usage is heavily penalized in order to achieve these results. Nonetheless, it has been verified with a large cohort of patients and only requires a single tri-axial accelerometer. The methodology is patient-independent and can be configured towards sensitivity or specificity.

7.3 Freezing of Gait

The Chapter 6 has introduced a new approach in detecting FoG events by means of a single waist-mounted tri-axial accelerometer. This algorithm has been trained and verified with signals from 20 PD patients (i.e. 209 freezing events), that have been recorded at the patient’s home. an SVM is used to recognize freezing episodes in a sliding window. This classification is then aggregated as part of a meta-analysis. The results indicate an excellent accuracy compared to current works (see Table 7.3). The overall approach is patient-independent and can be configured toward a high sensitivity / specificity.

Bächlin et al. [32] yielded to 73.1% sensitivity and 81.6% specificity which are clearly outperformed by the new approach (i.e. regardless of kernel and feature set). The same is true for Niazmand et al. [115], their results are also clearly exceeded. Here, five accelerometers on 6 PD patients are used. Even though the results by Bächlin et al. [32] are exceeded, they captured 237 freezing events in 10 PD patients based on three accelerometers and three gyroscopes. In

contrast to this work, Bächlin et al. [32] use 0.5 second time frames. However, the authors also point out that their results could be improved by using patient-specific settings as opposed to an patient-independent configuration.

Djurić-Jovičić et al. [51] achieved a mean error rate of 16% (84% accuracy) in detecting akinesia. Their results are based on 4 PD patients, six accelerometers and six gyroscopes that are mounted on the legs. Their results are also clearly exceeded.

The results by Cole et al. [37] are consistently exceeded. Both sensitivity and specificity of the proposed approach are above their results. A DNN is used by Cole et al. [37] to detect episodes of freezing. They captured 87 freezing events in 4 PD patients (and 0 freezing events from 2 control subjects) with three tri-axial accelerometers and a surface EMG sensor. Their results are exceeded in terms of sensitivity and specificity, but also in terms of number of subjects. After adding a simple meta-analysis, the results improve to 99% specificity and 82% sensitivity, which are still outperformed by the proposed approach (in this work).

The presented approach outperformed the highlighted works. The main advantages include: (1) patient independence, (2) use of a single tri-axial accelerometer, (3) configurability (i.e. the algorithm can be tuned toward high sensitivity or high specificity as well as high / low data usage) and (4) the optimal window size is determined by evaluation at episode level (as opposed to window level, which increases specificity). Even though these results are obtained for time intervals of 45 to 60 seconds, the methodology may be useful for FoG prevention systems (and not just for FoG monitoring systems). In these systems, a freezing event is not necessarily required to be detected immediately. Instead an auditory cueing system could activate (even with a minute delay) and still help patients to regain their rhythm and thus possibly prevent further freezing events from happening.

7.4 Discussion

All proposed approaches yield to accuracies above 95% for most of the results (except for the linear kernel in detecting dyskinesia at the waist) and the same is true for specificity. Results for dyskinesia show that data usage is heavily penalized and that sensitivity is found to be close to or above 80%. The low data usage can be explained by quite a few instances in which the underlying SVM incorrectly classified dyskinesia (see Table 6.8). To counteract this behavior, the upper and lower thresholds were adjusted accordingly (see Table 6.10). But they were required to be set so far away from the intuitive boundary of 50% (e.g. 10% for the lower threshold and 90% for the upper threshold) that large portions of data were classified as “unknown” and thus ignored. The most probable cause for the incorrect classification of dyskinesia is that the utilized features were not capable of capturing the subtle differences between dyskinesia and other activities with a rhythmical component (such as walking). Nonetheless similar results to those of state-of-the-art techniques were achieved.

In case of resting tremor and FoG, sensitivity is close to or above 90%. For these symptoms, the data usage is also penalized but still within an acceptable range. Both algorithms also compare reasonably well with related works. The results for tremor at rest slightly outperform state-of-the-art techniques. In

case of freezing events, the results indicate good sensitivity and specificity in comparison to state-of-the-art techniques. Furthermore, the proposed methodology only utilizes a single tri-axial accelerometer whereas most state-of-the-art techniques employ a set of various sensors at various body locations. Also the methodology is patient-independent and can be configured. All in all, the proposed methodology works well in at least two of the three presented scenarios and utilized a much larger database.

Chapter 8

Conclusions

This chapter summarizes conclusions drawn from the work and results that have been presented in the course of this thesis. Furthermore, a summary of contributions made to the research field and possible directions for future work are outlined.

8.1 Conclusions

The thesis is driven by four research questions (see Section 1.2), each of which is answered in the course of this work. For reasons of simplicity, the questions are repeated in the following.

1. How can a time series be represented? What is an adequate software-architecture for a data mining (DM) and time series analysis framework?
2. Which Parkinson's Disease symptoms can be detected and how can they be detected?
3. How can published state-of-the-art techniques for detecting motor symptoms of Parkinson's Disease be improved?
4. How well do the new / improved approaches perform when compared to state-of-the-art techniques?

The first research question has been answered in Chapter 4 ('A Framework for Time Series Analysis'). The design and development process of a software-architecture for a DM and time series analysis framework is shown from beginning to end. Requirements are individually discussed and broken down into manageable pieces. The resulting framework is a general-purpose data processing environment that is built around the principles of modularity, reusability and extensibility. It can handle arbitrary data and model non-linear processes (e.g. graph structures and cyclic processing).

Section 3.2 focuses on Parkinson's Disease (PD) motor symptoms with respect to research question two. Due to the rather large number of motor and non motor symptoms of PD (see Table A.1 and Table A.2), the list of symptoms was narrowed down to commonly experienced motor symptoms. A set of publications, which detail algorithms and their utilized features, is shown for relevant

symptoms. Identifiable symptoms are outlined by means of publications that detect them in time series data.

Proposals for improving state-of-the-art techniques are developed in Chapter 6. Here algorithms for detecting a subset of the previously outlined symptoms and side effects (i.e. tremor at rest, freezing episodes and dyskinesia) are presented. A flexible, configurable and patient-independent methodology is developed around a support vector machine (SVM) and a meta-analysis. This approach (with minor differences) is then applied to these symptoms and side effects.

The forth research question is answered in Chapter 7. The previously presented methodology is compared to state-of-the-art techniques in terms of sensors, dataset, accuracy, sensitivity and specificity where possible. The proposed methodology is shown to outperform related works in case of resting tremor and freezing of gait (FoG). In case of dyskinesia, the results do not exceed those of state-of-the-art techniques but yield to similar results.

8.2 Contributions

The following publications were created during the work on this thesis (in reverse order of appearance). Those that are yet to be published are marked as such.

- C. Ahlrichs, A. Samà, J. Cabestany, M. Lawo, C. Pérez-López, D. Rodríguez-Martín, S. Alcaine, B. Mestre, P. Quispe, A. Costa, I. Mazzú, H. Lewy, A. Bayés, T. Counihan, and A. Rodríguez-Molinero. Real-world Continuous Monitoring of Tremor in Parkinson’s Disease: A Study with 92 Patients Wearing a Wrist-Worn Accelerometer. *IEEE Journal of Biomedical and Health Informatics. Special Issue: Enabling Technologies in Parkinson’s Disease Management*, 2015. **Submitted for publication. Awaiting acceptance.**
- C. Ahlrichs, A. Samà, M. Lawo, J. Cabestany, D. Rodríguez-Martín, C. Pérez-López, D. Sweeney, L. Quinlan, G. Ó Laighin, T. Counihan, P. Browne, L. Hadas, G. Vainstein, A. Costa, R. Annicchiarico, S. Alcaine, B. Mestre, P. Quispe, A. Bayés, and A. Rodríguez-Molinero. Detecting Freezing of Gait with a Tri-Axial Accelerometer in Parkinson’s Disease Patients. *Medical & Biological Engineering & Computing*, 2014. **Submitted for publication. Awaiting acceptance.**
- C. Ahlrichs and A. Samà. Is “Frequency Distribution” Enough to Detect Tremor in PD Patients Using a Wrist Worn Accelerometer? In *Proceedings of the 8th International Conference on Pervasive Computing Technologies for Healthcare*, PervasiveHealth ’14, pages 65–71, ICST, Brussels, Belgium, Belgium, 2014. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering)
- C. Ahlrichs and M. Lawo. MOSIS: An Open Source Framework for Signal Processing and Machine Learning. *Journal of Machine Learning Research (JMLR)*, 2014. **Submitted for publication. Awaiting acceptance.**
- C. Ahlrichs and M. Lawo. Parkinson’s Disease Motor Symptoms in Machine Learning: A Review. *Health Informatics: An International Journal (HIJ)*, 2(3), November 2013

- C. Ahlrichs and M. Lawo. Workshop Paper: Indicating Motor Symptoms in PD Patients Using AI-based Algorithms. In *ICT meets Medicine and Health (ICTMH)*, March 2013
- C. Ahlrichs. Poster Abstract: Development and Evaluation of AI-based Parkinson’s Disease Related Motor Symptom Detection Algorithms. In *KI 2012: Advances in Artificial Intelligence - 35th Annual German Conference on AI, Saarbrücken, Germany, September 24-27, 2012. Proceedings*, September 2012

These publications as well as unpublished contributions are briefly summarized as done hereafter (grouped by topic).

- **Framework:** A framework for DM applications as well as processing and analyzing time series is presented. It focuses on online processing rather than delayed or offline processing and employs a strictly modularized structure. It counteracts the drawbacks of other related frameworks. A copy can be obtained at [108].
- **Algorithm(s) for tremor at rest:** A new methodology to detect Parkinsonian tremor was proposed. The approach has been trained and verified, for the first time, on a database with signals from 183 recordings of PD patients. These recordings were obtained in the patient’s home. The results show that previous works were slightly outperformed.
- **Algorithm(s) for dyskinesia:** The same methodology (as it has been used for tremor) can be successfully applied to recognize dyskinesia. Results are verified with a large cohort of patients (i.e. 172 recordings), but they do not outperform related work. Nonetheless, a reasonable accuracy is achieved even though data usage is heavily penalized.
- **Algorithm(s) for FoG:** Applying the presented methodology to FoG, the approach showed excellent accuracy compared to current works. The main advantages include: (1) patient independence, (2) use of a single tri-axial accelerometer, (3) configurability and (4) use of recordings that were obtained in a non-laboratory setting.
- **Review of publications:** A large set of publications (i.e. > 40) related to PD motor symptoms, PD side effects and software frameworks were reviewed. These were used to compile a comprehensive list of publications for several motor symptoms and to present them in a unified and comparable way (see Table A.3, Table A.4, Table A.5 and Table A.6). A similar process yielded to a comparison of several related frameworks (see Table 3.1).

To sum up, several journal articles, conference and workshop papers have been published as part of this thesis. Additionally, this work still contains large parts of (previously) unpublished materials.

8.3 Future Work

The results of this work may be presented in a coherent and self-contained way, but there remains a lot of work to be done. This section outlines possible directions of future work.

The same methodology (as it has been presented in Section 6.1) may also be useful in detecting festination, bradykinesia and other symptoms or side effects of PD. This has not been tested nor evaluated as part of this thesis and remains a promising path for future exploration.

The current methodology is applied to two sensors (i.e. wrist sensor for tremor detection and waist sensor for detecting freezing events and dyskinesia). However, it may be enough to just use a single sensor at the wrist to detect all these symptoms. Consequently, another line of research could concentrate on evaluating accuracies of a wrist mounted device in detecting freezing events as well as dyskinesia. Using a single sensor platform (rather than two sensor platforms) may increase usability and end-user acceptance of a monitoring system. In the same line of thought, the methodology could be applied to detect resting tremor, FoG and dyskinesia at the same time rather than having three separate SVMs for detection. A single SVM that classifies multiple symptoms may already be enough. This has not been evaluated and may resolve (potential) performance issues on computationally constrained devices such as a micro-processor.

Detecting dyskinesia deserves more attention than it has received during the work on this thesis. Especially the low data usage suggests that refinements are an appropriate path for future research. Differentiation between varying severities of dyskinesia or other (common) activities that provoke false positives (FPs) are just two ways which may lead to improved results.

The proposed framework still offers room for improvements in terms of number of algorithms, number of processing modules, supported data formats and so on. In the future, it may be of interest to keep on implementing new algorithms as they emerge and maintain “old” ones. With respect to the proposed methodology of this work, a proof-of-concept implementation of several modules and processing units was done. However, this is not yet complete and definitely needs refinements. Furthermore, most processing in this work was done by means of Matlab. In order to change this, a proof-of-concept implementation was realized and a road-map of required modules was deduced. Those interested in pursuing this goal may want to have a look at Section B.4.

This work utilized a traditional supervised learning approach (i.e. SVM) to recognize motor symptoms and side effects. It would be interesting to evaluate Hoeffding Trees [52], D-Stream [167], count-min [38] and related algorithms. These approaches maintain their state over time (i.e. as the stream of sensor signals evolves) rather than classifying on a per-sample-basis. To the knowledge of the author, this type of algorithm has yet to be applied in this context and the author is not aware of any preceding publication that may have done so.

Bibliography

- [1] C. Ahlrichs. Poster Abstract: Development and Evaluation of AI-based Parkinson's Disease Related Motor Symptom Detection Algorithms. In *KI 2012: Advances in Artificial Intelligence - 35th Annual German Conference on AI, Saarbrücken, Germany, September 24-27, 2012. Proceedings*, September 2012.
- [2] C. Ahlrichs and M. Lawo. Parkinson's Disease Motor Symptoms in Machine Learning: A Review. *Health Informatics: An International Journal (HIJ)*, 2(3), November 2013.
- [3] C. Ahlrichs and M. Lawo. Workshop Paper: Indicating Motor Symptoms in PD Patients Using AI-based Algorithms. In *ICT meets Medicine and Health (ICTMH)*, March 2013.
- [4] C. Ahlrichs and M. Lawo. MOSIS: An Open Source Framework for Signal Processing and Machine Learning. *Journal of Machine Learning Research (JMLR)*, 2014. **Submitted for publication. Awaiting acceptance.**
- [5] C. Ahlrichs and A. Samà. Is “Frequency Distribution” Enough to Detect Tremor in PD Patients Using a Wrist Worn Accelerometer? In *Proceedings of the 8th International Conference on Pervasive Computing Technologies for Healthcare*, PervasiveHealth '14, pages 65–71, ICST, Brussels, Belgium, Belgium, 2014. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
- [6] C. Ahlrichs, A. Samà, J. Cabestany, M. Lawo, C. Pérez-López, D. Rodríguez-Martín, S. Alcaine, B. Mestre, P. Quispe, A. Costa, I. Mazzú, H. Lewy, A. Bayés, T. Counihan, and A. Rodríguez-Molinero. Real-world Continuous Monitoring of Tremor in Parkinson's Disease: A Study with 92 Patients Wearing a Wrist-Worn Accelerometer. *IEEE Journal of Biomedical and Health Informatics. Special Issue: Enabling Technologies in Parkinson's Disease Management*, 2015. **Submitted for publication. Awaiting acceptance.**
- [7] C. Ahlrichs, A. Samà, M. Lawo, J. Cabestany, D. Rodríguez-Martín, C. Pérez-López, D. Sweeney, L. Quinlan, G. Ó Laighin, T. Counihan, P. Browne, L. Hadas, G. Vainstein, A. Costa, R. Annicchiarico, S. Alcaine, B. Mestre, P. Quispe, A. Bayés, and A. Rodríguez-Molinero. Detecting Freezing of Gait with a Tri-Axial Accelerometer in Parkinson's Disease Patients. *Medical & Biological Engineering & Computing*, 2014. **Submitted for publication. Awaiting acceptance.**

- [8] C. Ahlrichs, A. Samà, J. R. Simon, S. Herrlich, and A. Rodríguez-Molinero. HELP: Optimizing Treatment of Parkinson’s Disease Patients. In *3rd International Conference on the Elderly and New Technologies*, Castellón, Spain, Apr. 2012.
- [9] P. Andlin-Sobocki, B. Jönsson, H.-U. Wittchen, and J. Olesen. Cost of Disorders of the Brain in Europe. *European Journal of Neurology*, 12:1–27, 2005.
- [10] F. Antonelli and A. P. Strafella. Behavioral Disorders in Parkinson’s Disease: The Role of Dopamine. *Parkinsonism & Related Disorders*, 20, Supplement 1(0):S10 – S12, 2014. Proceedings of XX World Congress on Parkinson’s Disease and Related Disorders.
- [11] E. K. Antonsson and R. W. Mann. The Frequency Content of Gait. *Journal of Biomechanics*, 18(1):39–47, 1985.
- [12] R. A. Armstrong. Visual Signs and Symptoms of Parkinson’s Disease. *Clinical and Experimental Optometry*, 91(2):129–138, 2008.
- [13] R. Arvind, B. Karthik, N. Sriraam, and J. K. Kannan. Automated Detection of PD Resting Tremor Using PSD with Recurrent Neural Network Classifier. In *2010 International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom)*, pages 414–417, October. 2010.
- [14] M. Asgari and I. Shafran. Predicting Severity of Parkinson’s Disease from Speech. In *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 5201–5204, 31 2010–September 4 2010.
- [15] K. Bache and M. Lichman. UCI Machine Learning Repository, 2013.
- [16] A. Bagnall, C. A. Ratanamahatana, E. Keogh, S. Lonardi, and G. Janacek. A Bit Level Representation for Time Series Data Mining with Shape Based Similarity. *Data Mining and Knowledge Discovery*, 13(1):11–40, 2006.
- [17] Z. A. Bakar, N. M. Tahir, and I. M. Yassin. Classification of Parkinson’s Disease Based on Multilayer Perceptrons Neural Network. In *2010 6th International Colloquium on Signal Processing and Its Applications (CSPA)*, pages 1–4, May 2010.
- [18] E. Bakstein, K. Warwick, J. Burgess, O. Staudahl, and T. Aziz. Features for Detection of Parkinson’s Disease Tremor from Local Field Potentials of the Subthalamic Nucleus. In *2010 IEEE 9th International Conference on Cybernetic Intelligent Systems (CIS)*, pages 1–6, September 2010.
- [19] A. Beric, P. J. Kelly, A. Rezai, D. Sterio, A. Mogilner, M. Zonenshayn, and B. Kopell. Complications of Deep Brain Stimulation Surgery. *Stereotactic And Functional Neurosurgery*, 77(1-4):73–78, 2001.
- [20] A. Bifet, E. Frank, G. Holmes, and B. Pfahringer. Ensembles of Restricted Hoeffding Trees. *ACM Transactions on Intelligent Systems and Technology*, 3(2), 2012.

- [21] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl. MOA: Massive Online Analysis, a Framework for Stream Classification and Clustering. *Journal of Machine Learning Research - Proceedings Track*, 11:44–50, 2010.
- [22] C. Bockermann and H. Blom. The Streams Framework. Technical Report 5, TU Dortmund University, 12 2012.
- [23] V. Bonifati. Genetics of Parkinson’s Disease - State of the Art, 2013. *Parkinsonism & Related Disorders*, 20, Supplement 1(0):S23 – S28, 2014. Proceedings of XX World Congress on Parkinson’s Disease and Related Disorders.
- [24] R. R. Bouckaert, E. Frank, M. A. Hall, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. WEKA–Experiences with a Java Open-Source Project. *Journal of Machine Learning Research*, 11:2533–2541, 2010.
- [25] H. Braak, J. R. Bohl, C. M. Müller, U. Rüb, R. A. I. de Vos, and K. Del Tredici. Stanley Fahn Lecture 2005: The Staging Procedure for the Inclusion Body Pathology Associated with Sporadic Parkinson’s Disease Reconsidered. *Movement Disorders*, 21(12):2042–2051, 2006.
- [26] H. Braak, D. Sandmann-Keil, W. Gai, and E. Braak. Extensive Axonal Lewy Neurites in Parkinson’s Disease: A Novel Pathological Feature Revealed By α -synuclein Immunocytochemistry. *Neuroscience Letters*, 265(1):67–69, 1999.
- [27] H. Braak, K. D. Tredici, U. Rüb, R. A. I. de Vos, E. N. H. J. Steur, and E. Braak. Staging of Brain Pathology Related to Sporadic Parkinson’s Disease. *Neurobiology of Aging*, 24(2):197–211, 2003.
- [28] B. R. Brewer, S. Pradhan, G. Carvell, and A. Delitto. Application of Modified Regression Techniques to a Quantitative Assessment for the Motor Signs of Parkinson’s Disease. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 17(6):568–575, December 2009.
- [29] K. J. Burchiel. Thalamotomy for Movement Disorders. *Neurosurgery Clinics Of North America*, 6(1):55–71, 1995.
- [30] J. P. Burg. *Maximum Entropy Spectral Analysis*. PhD thesis, Stanford University, 1975.
- [31] M. Bächlin, M. Plotnik, D. Roggen, I. Maidan, J. M. Hausdorff, N. Giladi, and G. Troster. Wearable Assistant for Parkinson’s Disease Patients with the Freezing of Gait Symptom. *IEEE Transactions on Information Technology in Biomedicine*, 14(2):436–446, March 2010.
- [32] M. Bächlin, D. Roggen, G. Troster, M. Plotnik, N. Inbar, I. Meidan, T. Herman, M. Brozgol, E. Shaviv, N. Giladi, and J. M. Hausdorff. Potentials of Enhanced Context Awareness in Wearable Assistants for Parkinson’s Disease Patients with the Freezing of Gait Syndrome. In *2009 International Symposium on Wearable Computers (ISWC)*, pages 123–130, September 2009.

- [33] A. P. L. Bó, P. Poignet, and C. Geny. Pathological Tremor and Voluntary Motion Modeling and Online Estimation for Active Compensation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 19(2):177–185, April 2011.
- [34] J. Cancela, M. Pansera, M. T. Arredondo, J. J. Estrada, M. Pastorino, L. Pastor-Sanz, and J. L. Villalar. A Comprehensive Motor Symptom Monitoring and Management System: The Bradykinesia Case. In *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1008–1011, 31 2010–September 4 2010.
- [35] L. Chen and M. T. Özsu. Multi-Scale Histograms for Answering Queries Over Time Series Data. In Z. M. Özsoyoglu and S. B. Zdonik, editors, *ICDE*, page 838. IEEE Computer Society, 2004.
- [36] B. T. Cole, S. H. Roy, C. J. De Luca, and S. H. Nawab. Dynamic Neural Network Detection of Tremor and Dyskinesia from Wearable Sensor Data. In *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 6062–6065, 31 2010–September 4 2010.
- [37] B. T. Cole, S. H. Roy, and S. H. Nawab. Detecting Freezing-of-gait During Unscripted and Unconstrained Activity. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 5649–5652, 30 2011–September 3 2011.
- [38] G. Cormode and S. Muthukrishnan. An Improved Data Stream Summary: The Count-min Sketch and Its Applications. *Journal of Algorithms*, 55(1):58–75, April 2005.
- [39] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, New York, NY, USA, 2000.
- [40] N. Crosby, K. H. Deane, and C. E. Clarke. Amantadine in Parkinson’s Disease. *Cochrane Database Syst Rev*, 1:CD003468, 2003.
- [41] L. Cunningham, S. Mason, C. Nugent, G. Moore, D. Finlay, and D. Craig. Home-Based Monitoring and Assessment of Parkinson’s Disease. *IEEE Transactions on Information Technology in Biomedicine*, 15(1):47–53, January 2011.
- [42] L. Cunningham, C. Nugent, G. Moore, D. Finlay, and D. Craig. Computer-Based Assessment of Bradykinesia, Akinesia and Rigidity in Parkinson’s Disease. In M. Mokhtari, I. Khalil, J. Bauchet, D. Zhang, and C. Nugent, editors, *Ambient Assistive Health and Wellness Management in the Heart of the City*, volume 5597 of *Lecture Notes in Computer Science*, pages 1–8. Springer Berlin Heidelberg, 2009.
- [43] L. Cunningham, C. Nugent, G. Moore, D. Finlay, and D. Craig. Identifying Fine Movement Difficulties in Parkinson’s Disease Using a Computer Assessment Tool. In *9th International Conference on Information Technology and Applications in Biomedicine*, pages 1–4, November 2009.

- [44] W. Dauer and S. Przedborski. Parkinson's Disease: Mechanisms and Models. *Neuron*, 39(6):889–909, 2003.
- [45] C. A. Davie. A Review of Parkinson's Disease. *British Medical Bulletin*, 86(1):109–127, 2008.
- [46] A. de Barros, J. a. Cevada, A. Bayés, S. Alcaine, and B. Mestre. Design and Evaluation of a Medication Application for People with Parkinson's Disease. In G. Memmi and U. Blanke, editors, *Mobile Computing, Applications, and Services*, volume 130 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 273–276. Springer International Publishing, 2014.
- [47] A. C. de Barros, J. a. Cevada, A. Bayés, S. Alcaine, and B. Mestre. User-centred Design of a Mobile Self-management Solution for Parkinson's Disease. In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*, MUM '13, pages 23:1–23:10, New York, NY, USA, 2013. ACM.
- [48] C. De Marchis, S. Conforto, G. Severini, M. Schmid, and T. D'Alessio. Detection of Tremor Bursts from the SEMG Signal: An Optimization Procedure for Different Detection Methods. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 7508–7511, 30 2011–September 3 2011.
- [49] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, 51(1):107–113, 2008.
- [50] J. L. Dideriksen, F. Gianfelici, L. Z. P. Maneski, and D. Farina. EMG-Based Characterization of Pathological Tremor Using the Iterated Hilbert Transform. *IEEE Transactions on Biomedical Engineering*, 58(10):2911–2921, October. 2011.
- [51] M. Djurić-Jovičić, N. S. Jovičić, I. Milovanović, S. Radovanović, N. Kresojević, and M. B. Popović. Classification of Walking Patterns in Parkinson's Disease Patients Based on Inertial Sensor Data. In *2010 10th Symposium on Neural Network Applications in Electrical Engineering (NEUREL)*, pages 3–6, September 2010.
- [52] P. Domingos and G. Hulten. Mining High-speed Data Streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pages 71–80, New York, NY, USA, 2000. ACM.
- [53] S. Fahn and R. Elton. *Recent Developments in Parkinson's Disease*, chapter Unified Parkinson's disease rating scale, pages 153–163. Macmillan Healthcare Information, Florham Park, NJ, 1987.
- [54] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(3):37–54, 1996.
- [55] M. F. Folstein, S. E. Folstein, and P. R. McHugh. Mini-mental State: A Practical Method for Grading the Cognitive State of Patients for the Clinician. *Journal of Psychiatric Research*, 12(3):189–198, 1975.

- [56] E. Frachtenberg, F. Petrini, J. Fernandez, and S. Pakin. STORM: Scalable Resource Management for Large-Scale Parallel Computers. *IEEE Trans. Comput.*, 55(12):1572–1587, December 2006.
- [57] E. Gamma, R. Helm, R. Johnson, and J. M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994.
- [58] D. J. Gelb, E. Oliver, and S. Gilman. Diagnostic Criteria for Parkinson Disease. *Arch Neurol*, 56(1):33–39, 1999.
- [59] C. G. Goetz, W. Poewe, O. Rascol, C. Sampaio, G. T. Stebbins, C. Counsell, N. Giladi, R. G. Holloway, C. G. Moore, G. K. Wenning, M. D. Yahr, and L. Seidl. Movement Disorder Society Task Force Report on the Hoehn and Yahr Staging Scale: Status and Recommendations. *Movement Disorders*, 19(9):1020–1028, 2004.
- [60] C. G. Goetz, B. C. Tilley, S. R. Shaftman, G. T. Stebbins, S. Fahn, P. Martinez-Martin, W. Poewe, C. Sampaio, M. B. Stern, R. Dodel, B. Dubois, R. Holloway, J. Jankovic, J. Kulisevsky, A. E. Lang, A. Lees, S. Leurgans, P. A. LeWitt, D. Nyenhuis, C. W. Olanow, O. Rascol, A. Schrag, J. A. Teresi, J. J. van Hilten, and N. LaPelle. Movement Disorder Society-sponsored Revision of the Unified Parkinson’s Disease Rating Scale (MDS-UPDRS): Scale Presentation and Clinimetric Testing Results. *Movement Disorders*, 23(15):2129–2170, 2008.
- [61] A. Gustavsson, M. Svensson, F. Jacobi, C. Allgulander, J. Alonso, E. Beghi, R. Dodel, M. Ekman, C. Faravelli, L. Fratiglioni, B. Gannon, D. H. Jones, P. Jennum, A. Jordanova, L. Jönsson, K. Karampampa, M. Knapp, G. Kobelt, T. Kurth, R. Lieb, M. Linde, C. Ljungcrantz, A. Maercker, B. Melin, M. Moscarelli, A. Musayev, F. Norwood, M. Preisig, M. Pugliatti, J. Rehm, L. Salvador-Carulla, B. Schlehofer, R. Simon, H.-C. Steinhausen, L. J. Stovner, J.-M. Vallat, P. V. den Bergh, J. van Os, P. Vos, W. Xu, H.-U. Wittchen, B. Jönsson, and J. Olesen. Cost of Disorders of the Brain in Europe 2010. *European Neuropsychopharmacology*, 21(10):718–779, 2011.
- [62] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.
- [63] J. L. Hamilton, E. Micheli-Tzanakou, and R. M. Lehman. Neural Networks Trained with Simulation Data for Outcome Prediction in Pallidotomy for Parkinson’s Disease. In *Proceedings of the 22nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 1, pages 1–4 vol.1, 2000.
- [64] J. H. Han, W. J. Lee, T. B. Ahn, B. S. Jeon, and K. S. Park. Gait Analysis for Freezing Detection in Patients with Movement Disorder Using Three Dimensional Acceleration System. In *Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 2, pages 1863 – 1865 Vol.2, September 2003.

- [65] HELP. Home-based Empowered Living for Parkinson's Disease Patients, February 2013. <http://help-parkinson-aal-project.tid.es>.
- [66] H. Hochheiser and B. Shneiderman. Interactive Exploration of Time Series Data. In K. Jantke and A. Shinohara, editors, *Discovery Science*, volume 2226 of *Lecture Notes in Computer Science*, pages 441–446. Springer Berlin Heidelberg, 2001.
- [67] M. M. Hoehn. Parkinsonism: Onset, Progression, and Mortality. *Neurology*, 17:427–442, 1967.
- [68] C. N. Homann, K. Wenzel, K. Suppan, G. Ivanic, R. Crevenna, and E. Ott. Sleep Attacks—facts and Fiction: A Critical Review. *Adv Neurol*, 91:335–41, 2003.
- [69] J.-G. G. Hou and E. C. Lai. Non-motor Symptoms of Parkinson's Disease. *International Journal of Gerontology*, 1(2):53–64, 2007.
- [70] L. Huang, D. Milne, E. Frank, and I. H. Witten. Learning a Concept-based Document Similarity Measure. *Journal of the American Society for Information Science and Technology*, 2012.
- [71] A. J. Hughes, S. E. Daniel, L. Kilford, and A. J. Lees. Accuracy of Clinical Diagnosis of Idiopathic Parkinson's Disease: A Clinico-pathological Study of 100 Cases. *Journal of Neurology, Neurosurgery & Psychiatry*, 55(3):181–184, 1992.
- [72] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: Distributed Data-parallel Programs from Sequential Building Blocks. *SIGOPS Oper. Syst. Rev.*, 41(3):59–72, March 2007.
- [73] J. Jankovic. Parkinson's Disease: Clinical Features and Diagnosis. *Journal of Neurology, Neurosurgery & Psychiatry*, 79(4):368–376, 2008.
- [74] H. Jiancheng, W. Wenwu, and D. Hongjuan. Development of the Parkinson's Disease System for Chinese Medicine Diagnosis Based on Database. In *2008 IEEE International Symposium on IT in Medicine and Education (ITME)*, pages 749–752, December 2008.
- [75] A. Jobbágy, E. Furnee, P. Harcos, and M. Tarczy. Early Detection of Parkinson's Disease Through Automatic Movement Evaluation. *Engineering in Medicine and Biology Magazine, IEEE*, 17(2):81–88, March/April 1998.
- [76] W. H. Jost. Autonomic Dysfunctions in Idiopathic Parkinson's Disease. *Journal of Neurology*, 250:–, 2003. 10.1007/s00415-003-1105-z.
- [77] E. Jovanov, E. Wang, L. Verhagen, M. Fredrickson, and R. Fratangelo. deFOG - a Real Time System for Detection and Unfreezing of Gait of Parkinson's Patients. In *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 5151–5154, September 2009.

- [78] N. L. W. Keijsers, M. W. I. M. Horstink, and S. C. A. M. Gielen. Automatic Assessment of Levodopa-induced Dyskinesias in Daily Life By Neural Networks. *Movement Disorders*, 18(1):70–80, 2003.
- [79] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, SIGMOD '01, pages 151–162, New York, NY, USA, 2001. ACM.
- [80] E. J. Keogh, K. Chakrabarti, M. J. Pazzani, and S. Mehrotra. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Knowl. Inf. Syst.*, 3(3):263–286, 2001.
- [81] E. J. Keogh and M. J. Pazzani. Derivative Dynamic Time Warping. *Science*, pages 1–11, 2001.
- [82] G. V. Kondraske and R. M. Stewart. Web-based Evaluation of Parkinson's Disease Subjects: Objective Performance Capacity Measurements and Subjective Characterization Profiles. In *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*, pages 799–802, August 2008.
- [83] A. D. Korczyn. Parkinson's Disease. In E. in Chief: Kris Heggenhougen, editor, *International Encyclopedia of Public Health*, pages 10–17. Academic Press, Oxford, 2008.
- [84] A. D. Korczyn and T. Gurevich. Parkinson's Disease: Before the Motor Symptoms and Beyond. *Journal of the Neurological Sciences*, 289(1-2):2–6, 2010.
- [85] A. Krenz. *The Pathological Role of Synphilin-1 and the Therapeutic Potential of Hsp70 in Models of Parkinson's Disease Using Viral Vectors*. PhD thesis, Universität Tübingen, Wilhelmstr. 32, 72074 Tübingen, 2010.
- [86] A. Kupryjanow, B. Kunka, and B. Kostek. UPDRS Tests for Diagnosis of Parkinson's Disease Employing Virtual-Touchpad. In *2010 Workshop on Database and Expert Systems Applications (DEXA)*, pages 132–136, 30 2010–September 3 2010.
- [87] L. V. Laitinen, A. T. Bergenheim, and M. I. Hariz. Leksell's Posteroven-tral Pallidotomy in the Treatment of Parkinson's Disease. *Journal of Neurosurgery*, 76(1):53–61, 1992. PMID: 1727169.
- [88] V. R. Lesser, S. Nawab, and F. I. Klassner. IPUS: An Architecture for the Integrated Processing and Understanding of Signals. *Artificial Intelligence*, 77(1):129–171, 1995.
- [89] J. Lin, E. Keogh, S. Lonardi, J. P. Lankford, and D. M. Nystrom. Visually Mining and Monitoring Massive Time Series. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 460–469. ACM, 2004.

- [90] J. Lin, E. J. Keogh, S. Lonardi, and B. Y. Chiu. A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In M. J. Zaki and C. C. Aggarwal, editors, *DMKD*, pages 2–11. ACM, 2003.
- [91] M. A. Little, P. E. McSharry, E. J. Hunter, J. Spielman, and L. O. Ramig. Suitability of Dysphonia Measurements for Telemonitoring of Parkinson’s Disease. *IEEE Transactions on Biomedical Engineering*, 56(4):1015–1022, April 2009.
- [92] X. Liu, C. B. Carroll, S.-Y. Wang, J. Zajicek, and P. G. Bain. Quantifying Drug-induced Dyskinesias in the Arms Using Digitised Spiral-drawing Tasks. *Journal of Neuroscience Methods*, 144(1):47–52, 2005.
- [93] G. Lo, A. R. Suresh, L. Stocco, S. González-Valenzuela, and V. C. M. Leung. A Wireless Sensor System for Motion Analysis of Parkinson’s Disease Patients. In *2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 372–375, March 2011.
- [94] M. G. Longstaff, P. R. Mahant, M. A. Stacy, A. W. A. Van Gemmert, B. C. Leis, and G. E. Stelmach. Discrete and Dynamic Scaling of the Size of Continuous Graphic Movements of Parkinsonian Patients and Elderly Controls. *Journal of Neurology, Neurosurgery & Psychiatry*, 74(3):299–304, 2003.
- [95] A. M. Lozano. Surgery for Parkinson’s Disease, the Five W’s: Why, Who, What, Where, and When. *Adv Neurol*, 91, 2003.
- [96] K. C. Luk and V. M.-Y. Lee. Modeling Lewy Pathology Propagation in Parkinson’s Disease. *Parkinsonism & Related Disorders*, 20, Supplement 1(0):S85 – S87, 2014. Proceedings of XX World Congress on Parkinson’s Disease and Related Disorders.
- [97] M. R. Luquin, O. Scipioni, J. Vaamonde, O. Gershanik, and J. A. Obeso. Levodopa-induced Dyskinesias in Parkinson’s Disease: Clinical and Pharmacological Classification. *Movement Disorders*, 7(2):117–124, 1992.
- [98] C. D. Marchis, M. Schmid, and S. Conforto. An Optimized Method for Tremor Detection and Temporal Tracking Through Repeated Second Order Moment Calculations on the Surface EMG Signal. *Medical Engineering & Physics*, 34(9):1268–1277, 2012.
- [99] P. Martínez-Martín, T. del Ser Quijano, R. Piñeiro, M. Andrés, et al. A New Clinical Tool for Gait Evaluation in Parkinson’s Disease. *Clinical neuropharmacology*, 20(3):183–194, 1997.
- [100] P. Martínez-Martín, C. Rodríguez-Blázquez, M. J. Forjaz, and J. de Pedro. The Clinical Impression of Severity Index for Parkinson’s Disease: International Validation Study. *Movement Disorders*, 24(2):211–217, 2009.
- [101] C. Mathers, D. M. Fat, J. T. Boerma, and World Health Organization (WHO). *the Global Burden of Disease : 2004 Update*. World Health Organization, Geneva, Switzerland, 2008.

- [102] C. McRae, G. Diem, A. Vo, C. O'Brien, and L. Seeberger. Reliability of Measurements of Patient Health Status: A Comparison of Physician, Patient, and Caregiver Ratings. *Parkinsonism & Related Disorders*, 8(3):187–192, 2002.
- [103] J. Mekyska, I. Rektorova, and Z. Smekal. Selection of Optimal Parameters for Automatic Analysis of Speech Disorders in Parkinson's Disease. In *2011 34th International Conference on Telecommunications and Signal Processing (TSP)*, pages 408–412, August 2011.
- [104] F. Miralles, S. Tarongi, and A. Espino. Quantification of the Drawing of An Archimedes Spiral Through the Analysis of Its Digitized Picture. *Journal of Neuroscience Methods*, 152(1 - 2):18–31, 2006.
- [105] T. Mitsa. *Temporal Data Mining*. Chapman & Hall/CRC data mining and knowledge discovery series. CRC Press, Boca Raton, Fla. [u.a.], 2010. XXII, 373 S. ; 25 cm : graph. Darst.
- [106] S. T. Moore, H. G. MacDougall, and W. G. Ondo. Ambulatory Monitoring of Freezing of Gait in Parkinson's Disease. *Journal of Neuroscience Methods*, 167(2):340–348, 2008.
- [107] G. Morris. *Candlestick Charting Explained: Timeless Techniques for Trading Stocks and Futures*. Mcgraw-Hill, 3rd edition, May 2006.
- [108] MOSIS. Modularized Signal Processing and Analysis, January 2015. <http://www.github.com/claasahl/MOSIS>.
- [109] J. Murphy. *Technische Analyse Der Finanzmärkte: Grundlagen, Methoden, Strategien, Anwendungen*. Börse-online-Edition. Finanzbuch Verlag GmbH, 2000.
- [110] L. J. Myers and C. D. MacKinnon. Quantification of Movement Regularity During Internally Generated and Externally Cued Repetitive Movements in Patients with Parkinson's Disease. In *2005 2nd International IEEE EMBS Conference on Neural Engineering*, pages 281–284, March 2005.
- [111] K. Niazmand, A. Kalaras, H. Dai, and T. C. Lueth. Comparison of Methods for Tremor Frequency Analysis for Patients with Parkinson's Disease. In *2011 4th International Conference on Biomedical Engineering and Informatics (BMEI)*, volume 2, pages 693–697, October. 2011.
- [112] K. Niazmand, I. Somlai, S. Louizi, and T. C. Lueth. Proof of the Accuracy of Measuring Pants to Evaluate the Activity of the Hip and Legs in Everyday Life. In J. C. Lin, K. S. Nikita, O. Akan, P. Bellavista, J. Cao, F. Dressler, D. Ferrari, M. Gerla, H. Kobayashi, S. Palazzo, S. Sahni, X. S. Shen, M. Stan, J. Xiaohua, A. ZoMaya, and G. Coulson, editors, *Wireless Mobile Communication and Healthcare*, volume 55 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 235–244. Springer Berlin Heidelberg, 2011. 10.1007/978-3-642-20865-2_30.

- [113] K. Niazmand, K. Tonn, A. Kalaras, U. M. Fietzek, J. H. Mehrkens, and T. C. Lueth. Quantitative Evaluation of Parkinson's Disease Using Sensor Based Smart Glove. In *2011 24th International Symposium on Computer-Based Medical Systems (CBMS)*, pages 1–8, June 2011.
- [114] K. Niazmand, K. Tonn, A. Kalaras, S. Kammermeier, K. Boetzel, J. H. Mehrkens, and T. C. Lueth. A Measurement Device for Motion Analysis of Patients with Parkinson's Disease Using Sensor Based Smart Clothes. In *2011 5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)*, pages 9–16, May 2011.
- [115] K. Niazmand, K. Tonn, Y. Zhao, U. M. Fietzek, F. Schroeteler, K. Ziegler, A. O. Ceballos-Baumann, and T. C. Lueth. Freezing of Gait Detection in Parkinson's Disease Using Accelerometer Based Smart Clothes. In *2011 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 201–204, November 2011.
- [116] J. G. Nutt. Motor Fluctuations and Dyskinesia in Parkinson's Disease. *Parkinsonism & Related Disorders*, 8(2):101–108, 2001.
- [117] C. W. Olanow, C. G. Goetz, J. H. Kordower, A. J. Stoessl, V. Sossi, M. F. Brin, K. M. Shannon, G. M. Nauert, D. P. Perl, J. Godbold, and T. B. Freeman. A Double-blind Controlled Trial of Bilateral Fetal Nigral Transplantation in Parkinson's Disease. *Annals of Neurology*, 54:403–414, 2003.
- [118] P. E. O'Suilleabhain and J. Y. Matsumoto. Time-frequency Analysis of Tremors. *Brain*, 121(11):2127–2134, 1998.
- [119] J. Parkinson. An Essay on the Shaking Palsy. 1817. *The Journal of neuropsychiatry and clinical neurosciences*, 14(2):223–236; discussion 222, 2002.
- [120] M. Pastorino, J. Cancela, M. T. Arredondo, M. Pansera, L. Pastor-Sanz, F. Villagra, M. A. Pastor, and J. A. Martín. Assessment of Bradykinesia in Parkinson's Disease Patients Through a Multi-parametric System. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1810–1813, 30 2011–September 3 2011.
- [121] S. Patel, T. Hester, R. Hughes, N. Huggins, D. Standaert, Alice, A. Flaherty, and P. Bonato. Using Wearable Sensors to Enhance DBS Parameter Adjustment for Parkinson's Disease Patients Through Measures of Motor Response. In *2006 3rd IEEE/EMBS International Summer School on Medical Devices and Biosensors*, pages 141–144, September 2006.
- [122] S. Patel, K. Lorincz, R. Hughes, N. Huggins, J. Growdon, D. Standaert, M. Akay, J. Dy, M. Welsh, and P. Bonato. Monitoring Motor Fluctuations in Patients with Parkinson's Disease Using Wearable Sensors. *IEEE Transactions on Information Technology in Biomedicine*, 13(6):864–873, November 2009.
- [123] S. Patel, K. Lorincz, R. Hughes, N. Huggins, J. H. Growdon, M. Welsh, and P. Bonato. Analysis of Feature Space for Monitoring Persons with

- Parkinson's Disease with Application to a Wireless Wearable Sensor System. In *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*, pages 6290–6293, August 2007.
- [124] S. Patel, D. Sherrill, R. Hughes, T. Hester, N. Huggins, T. Lie-Nemeth, D. Standaert, and P. Bonato. Analysis of the Severity of Dyskinesia in Patients with Parkinson's Disease via Wearable Sensors. In *2006 International Workshop on Wearable and Implantable Body Sensor Networks (BSN)*, pages 4 pp. –126, April 2006.
 - [125] PERFORM. Personal Health Systems for Monitoring and Point-of-Care Diagnostics-Personalized Monitoring, October 2012. <http://www.perform-project.eu>.
 - [126] M. Pouloupoulos, O. A. Levy, and R. N. Alcalay. The Neuropathology of Genetic Parkinson's Disease. *Movement Disorders*, 27(7):831–842, 2012.
 - [127] S. D. Pradhan, B. R. Brewer, G. E. Carvell, P. J. Sparto, A. Delitto, and Y. Matsuoka. Relation Between Ability to Track Force During Dual Tasking and Function in Individuals with Parkinson's Disease. In *IEEE International Conference on Rehabilitation Robotics, 2009. ICORR 2009*, pages 885–892, June 2009.
 - [128] N. P. Quinn, A. E. Lang, W. C. Koller, and C. D. Marsden. PAINFUL PARKINSON'S DISEASE. *The Lancet*, 327(8494):1366–1369, 1986. Originally published as Volume 1, Issue 8494.
 - [129] L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.
 - [130] G. Rao, L. Fisch, S. Srinivasan, F. D'Amico, T. Okada, C. Eaton, and C. Robbins. Does This Patient Have Parkinson Disease? *JAMA: The Journal of the American Medical Association*, 289(3):347–353, 2003.
 - [131] C. A. Ratanamahatana, E. J. Keogh, A. J. Bagnall, and S. Lonardi. A Novel Bit Level Time Series Representation with Implications for Similarity Search and Clustering. In T. B. Ho, D. W.-L. Cheung, and H. Liu, editors, *PAKDD*, volume 3518 of *Lecture Notes in Computer Science*, pages 771–777. Springer, 2005.
 - [132] J. Read, A. Bifet, B. Pfahringer, and G. Holmes. Batch-Incremental Versus Instance-Incremental Learning in Dynamic and Evolving Data. In *Proceedings Symposium on Advances in Intelligent Data Analysis*, pages 313–323. SPRINGER, 2012.
 - [133] REMPARK. Personal Health Device for the Remote and Autonomous Management of Parkinson's Disease, June 2012. <http://www.rempark.eu>.
 - [134] K. Revett, F. Gorunescu, and A.-B. M. Salem. Feature Selection in Parkinson's Disease: A Rough Sets Approach. In *2009 International Multiconference on Computer Science and Information Technology (IMCSIT)*, pages 425–428, October. 2009.

- [135] P. Riederer, J. Sian, and M. Gerlach. Is There Neuroprotection in Parkinson Syndrome? *Journal of Neurology*, 247:IV8–IV11, 2000. 10.1007/PL00007777.
- [136] G. Rigas, A. Tzallas, M. Tsipouras, P. Bougia, E. Tripoliti, D. Baga, D. Fotiadis, S. Tsouli, and S. Konitsiotis. Assessment of Tremor Activity in the Parkinson’s Disease Using a Set of Wearable Sensors. *IEEE Transactions on Information Technology in Biomedicine*, PP(99):1, 2012.
- [137] D. Rodríguez-Martín, A. Samà, C. Pérez-López, J. Cabestany, A. Català, and A. Rodríguez-Molinero. Enhancing FoG Detection By Means of Postural Context Using a Waist Accelerometer. *First International Freezing of Gait Congress (IFOG 2014)*, Februray 2014.
- [138] D. Rodríguez-Martín, A. Samà, C. Pérez-López, J. Cabestany, A. Català, and A. Rodríguez-Molinero. Posture Transition Identification on PD Patients Through a SVM-based Technique and a Single Waist-worn Accelerometer. Accepted for publication in *Neurocomputing*, 2015.
- [139] D. Rodríguez-Martín, C. Pérez-López, A. Samà, J. Cabestany, and A. Català. A Wearable Inertial Measurement Unit for Long-Term Monitoring in the Dependency Care Area. *Sensors*, 13(10):14079–14104, 2013.
- [140] S. H. Roy, B. T. Cole, L. D. Gilmore, C. J. De Luca, and S. H. Nawab. Resolving Signal Complexities for Ambulatory Monitoring of Motor Function in Parkinson’s Disease. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 4832–4835, 30 2011–September 3 2011.
- [141] D. Rye. Excessive Daytime Sleepiness and Unintended Sleep in Parkinson’s Disease. *Current Neurology and Neuroscience Reports*, 6:169–176, 2006. 10.1007/s11910-996-0041-8.
- [142] A. Salarian, H. Russmann, F. J. G. Vingerhoets, P. R. Burkhard, Y. Blanc, C. Dehollain, and K. Aminian. An Ambulatory System to Quantify Bradykinesia and Tremor in Parkinson’s Disease. In *2003 4th International IEEE EMBS Special Topic Conference on Information Technology Applications in Biomedicine*, pages 35–38, April 2003.
- [143] A. Salarian, H. Russmann, C. Wider, P. R. Burkhard, F. J. G. Vingerhoets, and K. Aminian. Quantification of Tremor and Bradykinesia in Parkinson’s Disease Using a Novel Ambulatory Monitoring System. *IEEE Transactions on Biomedical Engineering*, 54(2):313–322, February 2007.
- [144] A. Samà, C. Peréz, D. Rodríguez-Martin, J. Cabestany, J. M. Moreno Aróstegui, and A. Rodríguez-Molinero. A Heterogeneous Database for Movement Knowledge Extraction in Parkinson’s Disease. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2013.
- [145] A. Samà, C. Pérez-López, D. Rodríguez-Martín, J. M. Moreno-Aróstegui, J. Rovira, C. Ahlrichs, R. Castro, J. a. Cevada, R. Graça, V. Guimarães,

- B. Pina, T. Counihan, H. Lewy, R. Annicchiarico, A. Bayés, A. Rodríguez-Molinero, and J. Cabestany. A Double Closed Loop to Enhance the Quality of Life of Parkinson's Disease Patients: REMPARK System. *Studies in health technology and informatics*, 207:115–24, 2014.
- [146] A. Samii, J. G. Nutt, and B. R. Ransom. Parkinson's Disease. *The Lancet*, 363(9423):1783–1793, 2004.
- [147] J. D. Schaafsma, Y. Balash, T. Gurevich, A. L. Bartels, J. M. Hausdorff, and N. Giladi. Characterization of Freezing of Gait Subtypes and the Response of Each to Levodopa in Parkinson's Disease. *European Journal of Neurology*, 10(4):391–398, 2003.
- [148] A. Schrag and N. Quinn. Dyskinesias and Motor Fluctuations in Parkinson's Disease. *Brain*, 123(11):2297–2305, 2000.
- [149] P. R. Schuurman, D. A. Bosch, P. M. M. Bossuyt, G. J. Bonsel, E. J. W. van Someren, R. M. A. de Bie, M. P. Merkus, and J. D. Speelman. A Comparison of Continuous Thalamic Stimulation and Thalamotomy for Suppression of Severe Tremor. *New England Journal of Medicine*, 342(7):461–468, 2000.
- [150] R. S. Schwab and A. C. England. Projection Technique for Evaluating Surgery in Parkinson's Disease. In *Third symposium on Parkinson's disease*. Edinburgh: Livingstone, pages 152–157, 1969.
- [151] C. Shearer. The CRISP-DM Model: The New Blueprint for Data Mining. *Journal of Data Warehousing*, 5(4), 2000.
- [152] D. M. Sherrill, R. Hughes, S. S. Salles, T. Lie-Nemeth, M. Akay, D. G. Standaert, and P. Bonato. Advanced Analysis of Wearable Sensor Data to Adjust Medication Intake in Patients with Parkinson's Disease. In *2005 2nd International IEEE EMBS Conference on Neural Engineering*, pages 202–205, March 2005.
- [153] J. Shieh and E. Keogh. iSAX: Indexing and Mining Terabyte Sized Time Series. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 623–631, New York, NY, USA, 2008. ACM.
- [154] J. Sian, M. Gerlach, M. B. H. Youdim, and P. Riederer. Parkinson's Disease: A Major Hypokinetic Basal Ganglia Disorder. *Journal of Neural Transmission*, 106:443–476, 1999. 10.1007/s007020050171.
- [155] S. L. Smith and K. Shannon. Vector-based Analysis of Motor Activities in Patients with Parkinson's Disease. In *EUROMICRO 97. 'New Frontiers of Information Technology'. Short Contributions., Proceedings of the 23rd Euromicro Conference*, pages 50–55, September 1997.
- [156] SPARK. Lightning-Fast Cluster Computing, January 2014. <http://www.spark-project.org>.

- [157] J. Stamatakis, J. Crémers, D. Maquet, B. Macq, and G. Garraux. Gait Feature Extraction in Parkinson’s Disease Using Low-cost Accelerometers. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 7900–7903, 30 2011-September 3 2011.
- [158] K. Steece-Collier, E. Maries, and J. H. Kordower. Etiology of Parkinson’s Disease: Genetics and Environment Revisited. *Proceedings of the National Academy of Sciences*, 99(22):13972–13974, 2002.
- [159] STORM. Scalable Tool for Resource Management, January 2014. <http://www.storm-project.net>.
- [160] I. Strauss, S. K. Kalia, and A. M. Lozano. Where Are We with Surgical Therapies for Parkinson’s Disease? *Parkinsonism & Related Disorders*, 20, Supplement 1(0):S187 – S191, 2014. Proceedings of XX World Congress on Parkinson’s Disease and Related Disorders.
- [161] Y. Su, C. R. Allen, D. Geng, D. Burn, U. Brechany, G. D. Bell, and R. Rowland. 3-D Motion System (“data-gloves”): Application for Parkinson’s Disease. *IEEE Transactions on Instrumentation and Measurement*, 52(3):662–674, June 2003.
- [162] T. Sugiura, N. Sugiura, K. Sugiyama, and T. Yokoyama. Chaotic Approach to the Quantitative Analysis of Parkinson’s Disease. In *Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 3, pages 1583 –1586 vol.3, October-1 November 1998.
- [163] E. Tolosa, G. Wenning, and W. Poewe. The Diagnosis of Parkinson’s Disease. *Lancet Neurol*, 5(1):75–86, 2006.
- [164] TREMOR. An Ambulatory BCI-driven Tremor Suppression System Based on Functional Electrical Stimulation, July 2011. <http://www.iai.csic.es/tremor>.
- [165] M. G. Tsipouras, A. T. Tzallas, D. I. Fotiadis, and S. Konitsiotis. On Automated Assessment of Levodopa-induced Dyskinesia in Parkinson’s Disease. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2679–2682, 30 2011-September 3 2011.
- [166] M. G. Tsipouras, A. T. Tzallas, G. Rigas, P. Bougia, D. I. Fotiadis, and S. Konitsiotis. Automated Levodopa-induced Dyskinesia Assessment. In *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2411–2414, 31 2010-September 4 2010.
- [167] L. Tu and Y. Chen. Stream Data Clustering Based on Grid Density and Attraction. *ACM Trans. Knowl. Discov. Data*, 3(3):12:1–12:27, July 2009.
- [168] UIMA. Unstructured Information Management Architecture, January 2014. <http://uima.apache.org>.

- [169] A. Umemura, J. L. Jaggi, H. I. Hurtig, A. D. Siderowf, A. Colcher, M. B. Stern, and G. H. Baltuch. Deep Brain Stimulation for Movement Disorders: Morbidity and Mortality in 109 Patients. *Journal Of Neurosurgery*, 98(4):779–784, 2003.
- [170] S. von Campenhausen, B. Bornschein, R. Wick, K. Bötzel, C. Sampaio, W. Poewe, W. Oertel, U. Siebert, K. Berger, and R. Dodel. Prevalence and Incidence of Parkinson’s Disease in Europe. *Eur Neuropsychopharmacol*, 15(4):473–90, 2005.
- [171] O. Šprdlík, Z. Hurák, M. Hoskovicová, and E. Růžicka. Tremor Analysis By Decomposition of Acceleration Into Gravity and Inertial Acceleration Using Inertial Measurement Unit. In *2009 9th International Conference on Information Technology and Applications in Biomedicine (ITAB)*, pages 1–4, November 2009.
- [172] E. A. Wan. Discrete Time Neural Networks. *Applied Intelligence*, 3:91–105, 1993. 10.1007/BF00871724.
- [173] M. Wang, B. Wang, J. Zou, L. Chen, F. Shima, and M. Nakamura. A New Quantitative Evaluation Method of Parkinson’s Disease Based on Free Spiral Drawing. In *2010 3rd International Conference on Biomedical Engineering and Informatics (BMEI)*, volume 2, pages 694–698, October. 2010.
- [174] C. Watson, M. Kirkcaldie, and G. Paxinos. *The Brain: An Introduction to Functional Neuroanatomy*. Elsevier/Academic, Amsterdam, 1st ed edition, 2010.
- [175] I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning, Tools and Techniques*. Morgan Kaufmann series in data management systems. Elsevier/Morgan Kaufmann, Amsterdam [u.a.], 3. ed. edition, 2011. XXXIII, 629 S. : Ill., graph. Darst.
- [176] E. Wolters and C. Baumann, editors. *Parkinson Disease and Other Movement Disorders: Motor Behavioural Disorders and Behavioural Motor Disorders*. VU University Press, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands, January 2014. ISBN: 978 90 8659 666 9.
- [177] C. Xiuming, S. Jinjie, and Z. Caipo. Diagnose Model of Parkinson’s Disease Based on Principal Component Analysis and Sugeno Integral. In *2011 30th Chinese Control Conference (CCC)*, pages 2830–2834, July 2011.
- [178] H.-c. Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker. Map-reduce-merge: Simplified Relational Data Processing on Large Clusters. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1029–1040. ACM, 2007.
- [179] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, NSDI’12*, pages 2–2, Berkeley, CA, USA, 2012. USENIX Association.

- [180] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster Computing with Working Sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 10–10, Berkeley, CA, USA, 2010. USENIX Association.
- [181] K. Ziegler, F. Schroeteler, A. O. Ceballos-Baumann, and U. M. Fietzek. A New Rating Instrument to Assess Festination and Freezing Gait in Parkinsonian Patients. *Movement Disorders*, 25(8):1012–1018, 2010.
- [182] D. G. M. Zwartjes, T. Heida, J. P. P. van Vugt, J. A. G. Geelen, and P. H. Veltink. Ambulatory Monitoring of Activities and Motor Symptoms in Parkinson's Disease. *IEEE Transactions on Biomedical Engineering*, 57(11):2778–2786, November 2010.

Glossary

a.k.a. also known as.

ADL activities of daily living.

AE analysis engine.

AI artificial intelligence.

ALv2 Apache License, version 2.0.

ANOVA analysis of variance.

APCA adaptive piecewise constant approximation.

API application programming interface.

ARMA autoregressive moving average.

ASIS anterior superior illiac spine.

BCI brain-to-computer interaction.

BS&AN body sensor and actuator network.

CAS common analysis structure.

CISI-PD clinical impression of severity index.

CMA central moving average.

COMT catechol-O-methyltransferase.

CT computed tomography.

DAPHNet Dynamic Analysis of Physiological Networks.

DB database.

DBS deep brain stimulation.

DFT discrete Fourier transform.

DM data mining.

DNN dynamic neural network.

DT decision tree.

DTW dynamic time warping.

DWT discrete wavelet transform.

e.g. *exempli gratia* (for example).

EEG electroencephalography.

EM expectation maximization.

EMA exponential moving average.

EMG electromyograph.

EU European Union.

FAQ frequently asked questions.

FES functional electrical stimulation.

FFT fast Fourier transform.

FI freezing index.

FIR finite response.

FitForAge Bayerischer Forschungsverbund FitForAge.

FN false negative.

FoG freezing of gait.

FP false positive.

FT Fourier transform.

GBD Global Burden of Disease.

GNU AGPL GNU Affero General Public License.

GNU GPL GNU General Public License.

GPS global positioning system.

HELP Home-based Empowered Living for Parkinson's disease Patients.

HMM hidden markov model.

HYS Hoehn & Yahr scale.

i.e. *id est* (that is).

IICDR inter-intra class distance ratio.

IPUS Integrated Processing and Understanding of Signals.

IQR interquartile range.

IR infrared.

iSAX indexable symbolic aggregate approximation.

KDD knowledge discovery in databases.

KNN k-nearest neighbors.

LB Lewy body.

LCSS longest common subsequence.

LM Levenberg-Marquardt.

LN Lewy neurite.

MA moving average.

MAO-B monoamine oxidase B.

MDS Movement Disorder Society.

ML machine learning.

MM machine manager.

MOA massive online analysis.

MRI magnetic resonance imaging.

mRMR minimum redundancy maximum relevance.

MSE mean squared error.

NINDS National Institute of Neurological Disorders and Stroke.

NM node manager.

NN neural network.

NPV negative predictive value.

OS operating system.

OSM Open Street Map.

PAA piecewise aggregate approximation.

PCA principle component analysis.

PD Parkinson's Disease.

PERFORM Personal Health Systems for Monitoring and Point-of-Care Diagnostics-
Personalized Monitoring.

PHS personal health system.

PL program launcher.

PPV positive predictive value.

PRIMAS precision motion analysis system.

QUEST Quebec User Evaluation of Satisfaction with Assistive Technology.

RA rolling average.

RBF radial basis function.

RC remote controlled.

RDD resilient distributed dataset.

REM rapid eye movement.

REMPARK Personal Health Device for the Remote and Autonomous Management of Parkinson's Disease.

RF random forest.

RM resource management.

RMS root mean square.

RSGE rating scale for gait evaluation.

SAX symbolic aggregate approximation.

SCG Scaled Conjugate Gradient.

SES Schwab & England scale.

SOFA subject of analysis.

SPA signal processing algorithm.

STORM scalable tool for resource management.

SUS system usability scale.

SV support vector.

SVD singular value decomposition.

SVM support vector machine.

TCP transmission control protocol.

TN true negative.

TP true positive.

TREMOR An ambulatory BCI-driven tremor suppression system based on functional electrical stimulation.

UCI University of California Irvine.

UDP user datagram protocol.

UIMA Unstructured Information Management Architecture.

UK United Kingdom.

UPDRS unified Parkinson's Disease rating scale.

URL universal resource locator.

VM virtual machine.

VTP Virtual-Touchpad.

WEKA Waikato Environment for Knowledge Analysis.

WHO World Health Organization.

Appendix A

Parkinson's Disease

A.1 Symptoms

-
- | | |
|--|-----------------------------------|
| • tremor | • micrographia |
| • bradykinesia | • cutting food |
| • rigidity | • feeding |
| • postural instability | • hygiene |
| • hypominia | • slow activities of daily living |
| • dysarthria | • glabellar reflex |
| • dysphagia | • blepharospasm |
| • sialorrhoea | • dystonia |
| • decreased arm swing | • striatal deformity |
| • shuffling gait | • scoliosis |
| • festination difficulty arising from
chair | • camptocormia |
| • turning in bed | |
-

Table A.1: Lists a collection of motor symptoms that are associated with Parkinson's Disease (PD). This table has been constructed based on data found in a publication by Jankovic [73].

-
- | | |
|---|--|
| <ul style="list-style-type: none"> • depression • apathy • anxiety • compulsive-obsessive behavior • repetitive behavior • attention deficit • hallucinations • illusions • delusions • delirium • anxiety and panic attacks • dementia • Rapid eye movement (REM) sleep behavior disorder • REM loss of atonia • Non-REM-sleep related movement disorders • insomnia • excessive daytime somnolence • restless legs • periodic limb movements • nightmares • vivid dreams • sleep disordered breathing • cardiovascular system: • orthostatic hypotension • falls related to orthostatic hypotension • bradycardia or arrhythmia | <ul style="list-style-type: none"> • gastrointestinal system: • sialorrhea • dysphagia and choking • reflux • vomiting • nausea • fecal constipation • fecal incontinence • urinary system: • bladder disturbances • urgency and frequency • nocturia • incontinence • reproductive system: • sexual dysfunction • erectile impotence • hypersexuality • thermoregulation: • sweating • dry eyes • heat or cold intolerance • pain • paraesthesia • olfactory disturbance • fatigue • weight changes |
|---|--|
-

Table A.2: Shows a set of non motor symptoms known to be associated with PD. The contents of this table have been gathered from publications by Jankovic [73] and Hou et al. [69].

Authors	Project	State of the Art	Multiple Symptoms	Accelerometer	Gyroscope	Electromyography	Other Sensor(s)	Real Data	Synthetic Data	Length	Patients	Controls	Results
Cancela et al. [34]	PERFORM	✓	✗	✓	✗	✗	✗	✓	✗	-	20	0	Acc.: 86.5%
Myers et al. [110]	-	✗	✗	✓	✗	✓	✗	✓	✗	10m	3	0	-
Niazmand et al. [113]	FitForAge	✗	✓	✓	✗	✗	✓	✓	✗	-	6	3	Empirical difference
Pastorino et al. [120]	PERFORM	✓	✗	✓	✓	✗	✗	✓	✗	1w	30	0	Acc.: 74.4%
Patel et al. [121]	-	✗	✗	✓	✗	✓	✗	✓	✗	-	3	0	High classification rate
Patel et al. [122]	-	✗	✓	✓	✗	✗	✗	✓	✗	5m	12	0	2.2% estimation error
Patel et al. [123]	-	✗	✓	✓	✗	✗	✗	✓	✗	5m	12	0	feature ranking and window length
Salarian et al. [142]	-	✓	✓	✗	✓	✗	✗	✓	✗	40m	10	10	Significant for most features ($p \leq 1\%$)
Salarian et al. [143]	-	✓	✓	✗	✓	✗	✗	✓	✗	5h	11	0	Significant for most features ($p \leq 1\%$)
Sherrill et al. [152]	-	✗	✓	✓	✗	✗	✗	✓	✗	-	6	0	Recognition of motor state (with wearable sensors) is feasible
Zwartjes et al. [182]	-	✓	✓	✓	✓	✗	✗	✓	✗	10m	6	7	Significant for most features ($p \leq 2\%$)

Table A-3: Lists a series of publications related to automatic indication algorithms for bradykinesia. For each paper a set of algorithm-related points are highlighted. Various insights on the used datasets are also given (i.e. length, number of participants, kind of activities / tasks). It should be noted that some values (i.e. length of dataset and results) were estimated in order to keep the rows somewhat comparable. Whether or not the publication was referenced in chapter “Related Work Regarding Symptom Detection” is indicated by the “State of the Art” column. Fields marked by “_” indicated that no information was given in the corresponding paper.

Authors	Project	State of the Art	Multiple Symptoms	Accelerometer	Gyroscope	Electromyography	Other Sensor(s)	Real Data	Synthetic Data	Length	Patients	Controls	Results
Arvind et al. [13]	-	X	X	X	X	✓	X	✓	X	30m	-	-	Acc.: 95.6%
Cole et al. [36]	-	✓	✓	✓	X	✓	X	✓	X	4h	8	4	Sen.: 93% Spe.: 95%
De Marchis et al. [48]	TREMOR	X	X	X	X	✓	X	✓	✓	-	1	0	Sen.: 96% PPV: 96%
Dideriksen et al. [50]	TREMOR	X	X	X	✓	✓	X	✓	✓	1m	4	0	-
De Marchis et al. [98]	TREMOR	X	X	✓	✓	✓	✓	✓	✓	5m	4	0	Sen.:99% PPV: 96%
Niazmand et al. [111]	FitForAge	X	X	✓	X	X	X	✓	X	2m	1	0	frequency estimation error 1±0.88 Hz
Niazmand et al. [113]	FitForAge	X	✓	✓	X	X	✓	✓	X	-	4	1	Sen.: 100% Spe.: 83%
Niazmand et al. [113]	FitForAge	X	✓	✓	X	X	✓	✓	X	-	5	1	Sen.: 95% Spe.: 96%
Niazmand et al. [114]	FitForAge	✓	X	✓	X	X	X	✓	X	-	10	2	Sen.: 71% Spe.: 100%
Patel et al. [122]	-	X	✓	✓	X	X	X	✓	X	5m	12	0	3.4% estimation error
Rigas et al. [136]	-	✓	X	✓	X	X	X	✓	X	20m	10	13	Acc.: 74%
Roy et al. [140]	-	✓	✓	✓	X	✓	X	✓	X	4h	19	4	Sen.: 91.2% Spe.: 93.4% (mean)
Salarian et al. [142]	-	✓	✓	X	✓	X	X	✓	X	40m	10	10	Sen.: 76.6% Spe.: 98% (mean)
Salarian et al. [143]	-	✓	✓	X	✓	X	X	✓	X	45m	10	10	Sen.: 99.5% Spe.: 94.2%
Spirdllk et al. [171]	-	X	X	✓	✓	X	✓	✓	X	10m	28	25	Empirical difference
Zwartjes et al. [182]	-	✓	✓	✓	✓	X	X	✓	X	10m	6	7	Acc.: 84.7% (mean)

Table A.4: Lists a series of publications related to automatic indication algorithms for tremor. For more details on the column descriptions, please refer to Table A.3.

Authors	Project	State of the Art	Multiple Symptoms	Accelerometer	Gyroscope	Electromyography	Other Sensor(s)	Real Data	Synthetic Data	Length	Patients	Controls	Results
Bächlin et al. [32]	DAPHNet	✓	X	✓	✓	X	X	✓	X	50m	8	2	Sen.: 73.1% Spe.: 81.6%
Bächlin et al. [31]	DAPHNet	X	X	✓	X	X	X	✓	X	50m	8	2	Sen.: 73.1% Spe.: 81.6%
Cole et al. [37]	-	✓	X	✓	X	✓	X	✓	X	4h	10	2	Sen.: 82.9% Spe.: 97.3%
Djurić-Jovićić et al. [51]	-	✓	X	✓	✓	X	X	✓	X	6m	4	0	Classification error $\leq 16\%$
Han et al. [64]	-	X	X	✓	X	X	X	✓	X	-	2	5	Empirical difference
Niazmand et al. [115]	-	✓	X	✓	X	X	X	✓	X	-	6	0	Sen.: 88.3% Spe.: 85.3%
Stamatakis et al. [157]	-	✓	X	✓	X	X	X	✓	X	1m	1	1	Feature ranking

Table A.5: Lists a series of publications related to automatic indication algorithms for freezing of gait (FoG) events. For more details on the column descriptions, please refer to Table A.3.

Authors	Project	State of the Art	Multiple Symptoms	Accelerometer	Gyroscope	Electromyography	Other Sensor(s)	Real Data	Synthetic Data	Length	Patients	Controls	Results
Cole et al. [36]	-	✓	✓	✓	X	✓	X	✓	X	4h	8	4	Sen.: 91% Spe.: 93%
Keijsers et al. [78]	-	✓	X	✓	X	X	X	✓	X	2.5h	13	0	Acc.: 96.8% (mean)
Lo et al. [93]	-	X	X	✓	✓	X	X	✓	X	-	0	1	-
Patel et al. [122]	-	X	✓	✓	X	X	X	✓	X	5m	12	0	3.2% estimation error
Patel et al. [123]	-	X	✓	✓	X	X	X	✓	X	5m	12	0	feature ranking and window length
Patel et al. [124]	-	✓	X	✓	X	X	X	✓	X	5m	12	0	Clusters well separable
Roy et al. [140]	-	✓	✓	✓	X	✓	X	✓	X	4h	19	4	Sen.: 90% Spe.: 93.4% (mean)
Sherrill et al. [152]	-	X	✓	✓	X	X	X	✓	X	-	6	0	-
Tsipouras et al. [166]	PERFORM	✓	X	✓	✓	X	X	✓	X	20m	4	6	Acc.: 93.7%
Tsipouras et al. [165]	PERFORM	X	X	✓	✓	X	X	✓	X	10m	10	19	Acc.: 84.3%

Table A.6: Lists a series of publications related to automatic indication algorithms for dyskinesia. For more details on the column descriptions, please refer to Table A.3.

A.2 Diagnosis and Treatment

Step 1:

- Bradykinesia
- At least one of the following criteria:
 - Rigidity
 - 4-6 Hz rest tremor
 - Postural instability not caused by primary visual, vestibular, cerebellar or proprioceptive dysfunction

Step 2:

- Exclude other causes of parkinsonism

Step 3:

- At least three of the following supportive (prospective) criteria:
 - Unilateral onset
 - Rest tremor
 - Progressive disorder
 - Persistent asymmetry primarily affecting side of onset
 - Excellent response (70-100%) to levodopa
 - Severe levodopa induced chorea (dyskinesia)
 - Levodopa response for five years or more
 - Clinical course of ten years or more

Table A.7: United Kingdom (UK) PD Society Brain Bank's clinical criteria for the diagnosis of probable Parkinson's Disease. The contents have been reproduced based on a publication by Jankovic [73].

Group A features (characteristic of PD):

- Resting tremor
- Bradykinesia
- Rigidity
- Asymmetric onset

Group B features (suggestive of alternative diagnoses):

- Features unusual early in the clinical course
- Prominent postural instability in the first three years after symptom onset
- Freezing phenomenon in the first three years
- Hallucinations unrelated to medications in the first three years
- Dementia preceding motor symptoms or in the first year
- Supranuclear gaze palsy (other than restriction of upward gaze) or slowing of vertical saccades
- Severe, symptomatic dysautonomia unrelated to medications
- Documentation of condition known to produce parkinsonism and plausibly connected to the patient's symptoms (such as suitably located focal brain lesions or neuroleptic use within the past six months)

Criteria for definite PD:

- All criteria for probable Parkinson's are met and
- Histopathological confirmation of the diagnosis is obtained at autopsy

Criteria for probable PD:

- At least three of the four features in group A are present and
- None of the features in group B is present (note: symptom duration \geq three years is necessary to meet this requirement) and
- Substantial and sustained response to levodopa or a dopamine agonist has been documented

Criteria for possible PD:

- At least two of the four features in group A are present; at least one of these is tremor or bradykinesia and
- Either none of the features in group B is present or symptoms have been present \leq three years and none of the features in group B is present and
- Either substantial and sustained response to levodopa or dopamine agonist have been documented or the patient has not had an adequate trial of levodopa or dopamine agonist

Table A.8: National Institute of Neurological Disorders and Stroke (NINDS) diagnostic criteria for PD. The contents have been reproduced based on a publication by Jankovic [73] and Gelb et al. [58].

Clinically possible (at least one of):

- Asymmetric resting tremor
- Asymmetric rigidity
- Asymmetric bradykinesia

Clinically probable (any two of):

- Asymmetric resting tremor
- Asymmetric rigidity
- Asymmetric bradykinesia

Clinically definite:

- Criteria for clinically probable
- Definitive response to anti Parkinson drugs

Exclusion criteria:

- Exposure to drugs that can cause parkinsonism such as neuroleptics, some antiemetic drugs, tetrabenazine, and reserpine, flunarizine, and cinnarizine
- Cerebellar signs
- Corticospinal tract signs
- Eye movement abnormalities other than slight limitations of upward gaze
- Severe dysautonomia
- Early moderate to severe gait disturbance or dementia
- History of encephalitis, recurrent head injury (such as seen in boxers), or family history of PD in two or more family members
- Evidence of severe subcortical white-matter disease, hydrocephalus, or other structural lesions on magnetic resonance imaging (MRI) that may account for parkinsonism

Table A.9: Clinical diagnostic criteria for idiopathic PD. The contents have been reproduced based on a publication by Samii et al. [146].

Inclusion criteria:

- Clinically definite PD
- Hoehn and Yahr stage 2-4 (moderate to severe bilateral disease, but still ambulatory when ON)
- Levodopa response with clearly defined ON and OFF periods
- Persistent disabling motor fluctuations despite best drug treatment with some combination of
 - At least three hours of OFF period daily
 - Unpredictable OFF periods
 - Disabling dyskinesia
- Intact cognition as measured by neuropsychological testing and no active psychiatric disturbances
- Strong social support system and commitment from patient and family members to keep follow-up appointments

Exclusion criteria:

- Parkinson-plus syndromes
- Atypical parkinsonism (e.g. vascular parkinsonism)
- Drug-induced parkinsonism
- Medical contraindications to surgery or stimulation (serious comorbid medical disorders, chronic anticoagulation with warfarin, cardiac pace-makers, etc.)
- Dementia or psychiatric issues (untreated depression, psychosis, etc.)
- Intracranial abnormalities that would contraindicate surgery (e.g. stroke, tumor, vascular abnormality affecting the target area)
- Severe brain atrophy on imaging (makes target localization difficult)
- Serious doubt about patient's commitment to return to follow-up visits (several no-shows in the past, poor compliance record, etc.)

Table A.10: Proposed inclusion and exclusion criteria for deep brain stimulation (DBS). The contents have been reproduced based on a publication by Samii et al. [146].

Step 1: Diagnosis of parkinsonian syndrome

- Bradykinesia (slowness of initiation of voluntary movement with progressive reduction in speed and amplitude of repetitive actions and
- At least one of the following:
 - Muscular rigidity
 - 4-6 Hz rest tremor
 - Postural instability not caused by primary visual vestibular, cerebellar or proprioceptive dysfunction

Step 2: Exclusion criteria

- History of repeated strokes with stepwise progression of parkinsonian features
- History of repeated head injury
- History of definite encephalitis
- Oculogyric crises
- Neuroleptic treatment at the onset of symptoms
- More than one affected relative
- Sustained remission
- Strictly unilateral features after three years
- Supranuclear gaze palsy
- Cerebellar signs
- Early severe autonomic involvement
- Early severe dementia with disturbances of memory, language and praxis
- Babinski's sign
- Presence of cerebral tumor or communicating hydrocephalus on computed tomography (CT) scan
- Negative response to large doses of levodopa (if malabsorption excluded)

Step 3: Supportive criteria for PD (three or more required for diagnosis of definite PD)

- Unilateral onset
- Rest tremor present
- Progressive disorder
- Persistent asymmetry affecting side of onset
- Excellent response (70-100%) to levodopa
- Severe levodopa-induced chorea
- Levodopa response for five years or more
- Clinical course of ten years or more

Table A.11: Lists UK PD Society Brain Bank's criteria for diagnosis of parkinsonian syndrome. The contents have been reproduced based on a publication by Davie et al. [45].

-
- Substantial nerve cell depletion with accompanying gliosis in the substantia nigra.
 - At least 1 Lewy body in the substantia nigra or in the locus ceruleus
 - No pathologic evidence of other diseases that produce parkinsonism
-

Table A.12: Proposed diagnostic criteria for histopathologic confirmation of PD. The contents have been reproduced based on a publication by Gelb et al. [58].

Appendix B

Implementation Resources

B.1 Algorithm Samples

Listing B.1: Demonstrates the implementation of *mean*-binning using the Awk programming language.

```
1 # Assumptions:
2 # - Input data in separated by ";" (e.g. body-temperature
   .csv)
3 # - Second column contains the actual data (e.g. body-
   temperature.csv)
4 BEGIN{
5     FS=";" ;
6     SIZE=4;
7     CNT=0;
8 }
9
10 NF>=2{
11     VAL=$2 ;
12
13     # Put data in BIN
14     if (CNT!=SIZE) {
15         BIN[CNT++]=VAL;
16     }
17
18     # Process BIN, once it is full
19     if (CNT==SIZE) {
20         for ( i=0; i<SIZE; i++){
21             SUM+=BIN[ i ];
22         }
23         for ( i=0; i<SIZE; i++){
24             print SUM/SIZE;
25         }
26         CNT=0;
27         SUM=0;
28     }
```

29 }

B.2 Framework Samples

Listing B.2: An example of a *Processor* implementation. The shown example forwards its input values.

```
1 package de.claas.mosis.processing.debug;
2
3 import de.claas.mosis.annotation.Category;
4 import de.claas.mosis.annotation.Documentation;
5 import de.claas.mosis.model.ProcessorAdapter;
6
7 import java.util.List;
8
9 /**
10  * The class {@link de.claas.mosis.processing.debug.
11    * Forward}. It is intended for
12    * debugging purposes. This {@link de.claas.mosis.model.
13      * Processor}
14    * implementation returns the input values directly. It
15    * performs no operation
16    * other than forwarding the received input values.
17    *
18    * @param <I> type of data. See {@link de.claas.mosis.
19      * model.Processor} for
20      * details.
21    * @author Claas Ahlrichs (claasahl@tzi.de)
22    */
23 @Documentation(
24     supportMultipleInputs = true,
25     canHandleMissingData = true,
26     category = Category.Other,
27     author = {"Claas Ahlrichs"},
28     description = "This module is mostly intended for
29       debugging purposes. It forwards all input
30       data as output data. This can be useful when
31       two or more paths within the graph of modules
32       need to be normalized (i.e. their length must
33       be identical before they can be merged).",
34     purpose = "This implementation is intended for
35       debugging purposes.")
36 public class Forward<I> extends ProcessorAdapter<I, I> {
37
38     @Override
39     public void process(List<I> in, List<I> out) {
40         out.addAll(in);
41     }
42 }
```


Listing B.3: An example of a *DecoratorProcessor* implementation. The shown example generates statistics on the processing time that a module requires.

```

1 package de.claas.mosis.processing.debug;
2
3 import de.claas.mosis.annotation.Category;
4 import de.claas.mosis.annotation.Documentation;
5 import de.claas.mosis.annotation.Parameter;
6 import de.claas.mosis.model.Condition;
7 import de.claas.mosis.model.DecoratorProcessor;
8
9 import java.util.List;
10
11 /**
12  * The class {@link de.claas.mosis.processing.debug.Time
13  *   }. It is intended for
14  * debugging purposes. This {@link de.claas.mosis.model.
15  *   DecoratorProcessor}
16  * implementation measures the time (measured in
17  *   milliseconds) required to
18  * execute {@link de.claas.mosis.model.Processor#process(
19  *   java.util.List,
20  *   java.util.List)} of the wrapped {@link de.claas.mosis.
21  *   model.Processor}
22  * object. Furthermore it provides the time that the
23  * wrapped {@link
24  * de.claas.mosis.model.Processor} object was first
25  * called and the time it was
26  * last called (measured in milliseconds, between the
27  * current time and midnight,
28  * January 1, 1970 UTC).
29  *
30  * @author Claas Ahlrichs (claasahl@tzi.de)
31  */
32 @Documentation(
33     category = Category.Decorator,
34     author = {"Claas Ahlrichs"},
35     description = "This is a realization of a
36         DecoratorProcessor that is mostly intended for
37         debugging purposes. This implementation
38         measures the number of milliseconds that the
39         decorated module requires for processing. It
40         can be used to analyze processing time of
41         modules and optimize time efficiency.",
42     purpose = "This implementation is intended for
43         debugging purposes.")
44 public class Time extends DecoratorProcessor<Object,
45     Object> {
46
47     @Parameter("Number of milliseconds the last

```

```

        invocation of the process-method took.")
32     public static final String TIME = "time";
33     @Parameter("Total number of milliseconds spend in
        process-method (since first call of process-method
        and not just last invocation).")
34     public static final String TOTAL_TIME = "total time";
35     @Parameter("Timestamp of first invocation of process-
        method.")
36     public static final String FIRST_CALL = "first call";
37     @Parameter("Timestamp of last invocation of process-
        method.")
38     public static final String LAST_CALL = "last call";
39
40     /**
41      * Initializes the class with default values.
42      */
43     public Time() {
44         setParameter(LOCAL + TIME, "");
45         addCondition(LOCAL + TIME, new Condition.
            IsInteger());
46         addCondition(LOCAL + TIME, new Condition.
            IsGreaterOrEqual(0d));
47         setParameter(LOCAL + TOTAL_TIME, 0);
48         addCondition(LOCAL + TOTAL_TIME, new Condition.
            IsInteger());
49         addCondition(LOCAL + TOTAL_TIME, new Condition.
            IsGreaterOrEqual(0d));
50         setParameter(LOCAL + FIRST_CALL, "");
51         addCondition(LOCAL + FIRST_CALL, new Condition.
            IsInteger());
52         addCondition(LOCAL + FIRST_CALL, new Condition.
            IsGreaterOrEqual(0d));
53         setParameter(LOCAL + LAST_CALL, "");
54         addCondition(LOCAL + LAST_CALL, new Condition.
            IsInteger());
55         addCondition(LOCAL + LAST_CALL, new Condition.
            IsGreaterOrEqual(0d));
56     }
57
58     @Override
59     public void process(List<Object> in, List<Object> out
        ) {
60         long started = System.currentTimeMillis();
61         super.process(in, out);
62         long ended = System.currentTimeMillis();
63         long total = getParameterAsLong(TOTAL_TIME);
64         if (getParameter(FIRST_CALL).isEmpty()) {
65             setParameter(FIRST_CALL, started);
66         }
67         setParameter(LAST_CALL, started);

```

```

68         setParameter(TIME, ended - started);
69         setParameter(TOTALTIME, total + ended - started)
70     };
71 }
72 }

```

Listing B.4: An example of a *StreamHandlerImpl* implementation. The shown example provides raw access to a file.

```

1  package de.claas.mosis.io;
2
3  import de.claas.mosis.annotation.Parameter;
4  import de.claas.mosis.model.Condition;
5
6  import java.io.*;
7
8  /**
9   * The class {@link de.claas.mosis.io.FileImpl}. It is
10   * intended to provide
11   * access to files, such that {@link de.claas.mosis.io.
12   *   StreamHandler}
13   * implementations can process them.
14   *
15   * @author Claas Ahlrichs (claasahl@tzi.de)
16   */
17  public class FileImpl extends StreamHandlerImpl {
18
19      @Parameter("The file that is to be accessed /
20      processed.")
21      public static final String FILE = "filename";
22      public static final String APPEND = "append to file";
23
24      /**
25       * Initializes the class with default values.
26       */
27      public FileImpl() {
28          addCondition(FILE, new Condition.IsNotNull());
29          setParameter(FILE, "values.ser");
30          addCondition(APPEND, new Condition.IsBoolean());
31          setParameter(APPEND, false);
32      }
33
34      @Override
35      public InputStream getInputStream() throws
36          IOException {
37          return new FileInputStream(getParameter(FILE));
38      }
39
40      @Override

```

```

37     public OutputStream getOutputStream() throws
        IOException {
38         return new FileOutputStream(getParameter(FILE) ,
39                                     getParameterAsBoolean(APPEND));
40     }
41
42 }

```

Listing B.5: An example of a *DataHandler* implementation. The shown example provides access to queue.

```

1  package de.claas.mosis.io;
2
3  import de.claas.mosis.annotation.Category;
4  import de.claas.mosis.annotation.Documentation;
5  import de.claas.mosis.annotation.Parameter;
6  import de.claas.mosis.model.Condition;
7  import de.claas.mosis.util.Utills;
8
9  import java.util.LinkedList;
10 import java.util.List;
11 import java.util.Queue;
12
13 /**
14  * The class {@link de.claas.mosis.io.QueueHandler}. It
15  * is intended to enable
16  * communication with external entities through a {@link
17  * java.util.Queue}. This
18  * {@link de.claas.mosis.io.DataHandler} provide an
19  * alternative to the otherwise
20  * (mostly) stream-based communication. This will
21  * typically be utilized in
22  * scenarios where an external entity needs to push {
23  * @link java.lang.Object}s
24  * directly into the framework (and not through other
25  * means such as files or
26  * databases).
27  *
28  * @param <T> type of (incoming and outgoing) data. See {
29  * @link
30  * de.claas.mosis.model.Processor} for details
31  * .
32  *
33  * @author Claas Ahlrichs (claasahl@tzi.de)
34  */
35 @Documentation(
36     category = Category.InputOutput ,
37     author = {"Claas Ahlrichs"},
38     description = "This is a realization of a
39                   DataHandler which allows external programs to
40                   programatically feed data into the framework

```

```

    and programatically read data from the
    framework. This module is intended to create a
    simple way of interfacing external programs
    with the framework. The actual implementation
    of the utilized queue can be configured (by
    default a LinkedList is used). In contrast to
    the BlockingQueueHandler, this module expects
    external entities to provide their data
    samples in a timely manner. The framework will
    process all data samples in the queue until
    it is empty. This assumes that the external
    entity can provide data samples with at least
    the same timing that the framework requires to
    process them. Once the queue is empty (either
    because all samples were processed or the
    external entity could not keep up) processing
    will stop.",
30     purpose = "To allow storage and retrieval of
        objects within a queue.")
31 public class QueueHandler<T> extends DataHandler<T> {
32
33     @Parameter("Name of class from queue. An instance of
        this class backs this handler. Any class,
        implementing java.util.Queue, can be used.")
34     public static final String CLASS = "class (queue)";
35     private Queue<T> _Queue;
36
37     /**
38      * Initializes the class with default values.
39      */
40     public QueueHandler() {
41         addCondition(CLASS, new Condition.ClassExists());
42         setParameter(CLASS, LinkedList.class.getName());
43     }
44
45     /**
46      * Returns the (input / output) {@link java.util.
        Queue}. When in "reading"
47      * mode (see {@link #isReadOnly(List)}), then the
        head of the queue is
48      * removed and returned every time {@link #process(
        java.util.List,
49      * java.util.List)} is called. When in "writing" mode
        (see {@link
50      * #isWriteOnly(List)}), then all incoming elements
        are appended to the
51      * queue every time {@link #process(java.util.List,
        java.util.List)} is
52      * called.
53      *

```

```

54      * @return the (input / output) {@link java.util.
      Queue}
55      */
56      public Queue<T> getQueue() {
57          return _Queue;
58      }
59
60      @SuppressWarnings("unchecked")
61      @Override
62      public void setUp() {
63          super.setUp();
64          try {
65              _Queue = (Queue) Utils.instance(Class.forName
66                  (getParameter(CLASS)));
67          } catch (Exception e) {
68              e.printStackTrace();
69          }
70
71      @Override
72      public void dismantle() {
73          super.dismantle();
74          _Queue = null;
75      }
76
77      @Override
78      public void process(List<T> in, List<T> out) {
79          if (isReadOnly(in)) {
80              if (!getQueue().isEmpty()) {
81                  out.add(getQueue().poll());
82              }
83          } else {
84              for (T obj : in) {
85                  getQueue().offer(obj);
86              }
87              if (shouldForward()) {
88                  out.addAll(in);
89              }
90          }
91      }
92  }

```

Listing B.6: An example of a *Link* implementation. The shown example demonstrates the case where one would want to filter the flow of data between two modules.

```

1  package de.claas.mosis.flow;
2
3  import de.claas.mosis.model.Condition;
4  import de.claas.mosis.model.Configurable;

```

```

5 import de.claas.mosis.model.Observer;
6
7 import java.util.ArrayList;
8 import java.util.List;
9
10 /**
11  * The class {@link de.claas.mosis.flow.BiasedLink}. It
12  * is intended to act as a
13  * link between two {@link de.claas.mosis.flow.Node}
14  * objects that only allows
15  * objects of a certain type to pass through. Other
16  * objects are discarded.
17  *
18  * @author Claas Ahlrichs (claasahl@tzi.de)
19  */
20 public class BiasedLink extends LinkAdapter implements
21     Observer {
22
23     public static final String CLASS = "class";
24     private final List<Object> _Accepted;
25     private Class<?> _Class;
26
27     /**
28      * Initializes the class with default values.
29      */
30     public BiasedLink() {
31         _Accepted = new ArrayList<>();
32         addCondition(CLASS, new Condition.IsNotNull());
33         addObserver(this);
34         setParameter(CLASS, Object.class.getName());
35     }
36
37     @Override
38     public boolean push(List<Object> in) {
39         _Accepted.clear();
40         for (Object o : in) {
41             if (o == null || !_Class.isAssignableFrom(o.
42                 getClass())) {
43                 _Accepted.add(o);
44             }
45         }
46         return super.push(_Accepted);
47     }
48
49     @Override
50     public void update(Configurable configurable, String
51         parameter) {
52         if (CLASS.equals(parameter)) {
53             try {
54                 _Class = Class.forName(getParameter(CLASS

```

```

    ));
49         } catch (ClassNotFoundException e) {
50             e.printStackTrace();
51         }
52     }
53 }
54 }

```

Listing B.7: An example of a *Visitor* implementation. The shown example initializes all modules within a graph by calling the *setUp* method.

```

1  package de.claas.mosis.flow.visitor;
2
3  import de.claas.mosis.flow.CompositeNode;
4  import de.claas.mosis.flow.PlainNode;
5  import de.claas.mosis.flow.Visitor;
6  import de.claas.mosis.flow.iterator.OneShotLevelOrder;
7  import de.claas.mosis.model.Processor;
8
9  import java.util.HashSet;
10 import java.util.Set;
11
12 /**
13  * The class {@link de.claas.mosis.flow.visitor.
    SettingUpVisitor}. It is an
14  * implementation of the {@link de.claas.mosis.flow.
    Visitor} interface. It is
15  * intended to initialize all processing modules within a
    graph.
16  *
17  * @author Claas Ahlrichs (claasahl@tzi.de)
18  */
19 public class SettingUpVisitor implements Visitor {
20
21     private final Set<Processor> setup = new HashSet<>();
22
23     @Override
24     public boolean visitPlainNode(PlainNode node) {
25         Processor p = node.getProcessor();
26         if (!setup.contains(p)) {
27             p.setUp();
28             setup.add(p);
29         }
30         return true;
31     }
32
33     @Override
34     public boolean visitCompositeNode(CompositeNode node)
35     {
        OneShotLevelOrder levelOrder = new

```



```

36         OneShotLevelOrder(node.getSources());
37         while (levelOrder.hasNext()) {
38             if (!levelOrder.next().visit(this)) {
39                 break;
40             }
41         }
42         return true;
43     }
44 }

```

Listing B.8: An example of an *Iterator* implementation. The shown example provides sequential access to a graph of modules in such a way that the items are returned with respect to their depth within the graph (i.e. shortest distance from a root). It is a level ordered *Iterator*.

```

1 package de.claas.mosis.flow.iterator;
2
3 import de.claas.mosis.flow.Node;
4
5 import java.util.*;
6
7
8 /**
9  * The class {@link de.claas.mosis.flow.iterator.
10   * OneShotLevelOrder}. It is
11   * intended to provide sequential access to all {@link de
12   * .claas.mosis.flow.Node}
13   * objects in a {@link de.claas.mosis.flow.Graph}. This {
14   * @link
15   * java.util.Iterator} is meant for a single iteration
16   * across all objects in a
17   * {@link de.claas.mosis.flow.Graph}. The objects are
18   * returned based on their
19   * shortest distance to a root (or data source). Data
20   * sources are returned first
21   * (i.e. level 0), then their successors (i.e. level 1),
22   * then the successors'
23   * successors (i.e. level 2), and so on (i.e. level
24   * 3,4,...). This is repeated
25   * until the deepest {@link de.claas.mosis.flow.Node} ({
26   * @link
27   * de.claas.mosis.flow.Node} with longest distance to a
28   * root) was returned.
29   * There are no ordering constraints for {@link de.claas.
30   * mosis.flow.Node}
31   * objects that have the same depth. They are essentially
32   * randomly returned.
33   *
34   * <p/>
35   * <ol> <li>iteration: level 0 (data sources)</li> <li>

```

```

        iteration: level 1</li>
23 * <li>iteration: level 2</li> <li>iteration: level 3</li>
    > <li>...</li> </ol>
24 *
25 * @author Claas Ahlrichs (c.ahlrichs@neusta.de)
26 */
27 public class OneShotLevelOrder implements Iterator<Node>
    {
28
29     private final Set<Node> _Visited;
30     private final Queue<Node> _Nodes;
31
32     /**
33      * Initializes the class with the given parameter.
34      *
35      * @param sources the data sources to start with
36      */
37     public OneShotLevelOrder(Set<Node> sources) {
38         _Visited = new HashSet<>(sources);
39         _Nodes = new LinkedList<>();
40         for (Node src : sources) {
41             _Nodes.add(src);
42         }
43     }
44
45     @Override
46     public boolean hasNext() {
47         return !_Nodes.isEmpty();
48     }
49
50     @Override
51     public Node next() {
52         Node next = _Nodes.poll();
53         for (Node successor : next.getSuccessors()) {
54             if (!_Visited.contains(successor)) {
55                 _Visited.add(successor);
56                 _Nodes.add(successor);
57             }
58         }
59         return next;
60     }
61
62     @Override
63     public void remove() {
64         throw new UnsupportedOperationException();
65     }
66
67 }

```

B.3 Modules of Framework

Main Modules

- **Processor:** This represents the interface to which all modules within the framework adhere. Here all basic functionalities that a module must implement are listed and described. Most notably, it provides a common interface for processing time series data as well as handling of parameters. When adding (new) modules to the framework, this interface must be implemented either directly or indirectly by another (partial) implementation (e.g. *ProcessorAdapter*, *BufferingProcessor*, *ComparingProcessor*, etc.). It is encouraged to use these modules as reference implementations.
- **ProcessorAdapter:** This is a partial implementation of the main interface (i.e. *Processor*). It is intended to provide a unified way of handling parameters. Most modules within the framework use this implementation for getting, setting and constraining parameters. When a new module is created, then this implementation is likely to be referenced. It provides functionality that is expected to be common to almost all modules. Those wanting to implement new modules may also want to use *BufferingProcessor* or *Linear* as reference implementations.
- **DecoratorProcessor:** This implementation is meant to be overridden in one way or another. It is intended to wrap another module and to forward all method calls to it. Subclasses may want to override some (or all methods) to add functionality and behavior to existing modules. Most decorators within the framework use this default implementation to realize their specific functionality. When new decorators are added to the framework, then this module is likely to provide all the required default behavior. It is encouraged to use concrete decorators as reference implementations (e.g. *Time* or *Logger*).
- **DataHandler:** This represents a partial implementation for modules that intend to act as data sources and data sinks. Here, a common set of functions and configuration options are defined. Most modules within the framework, that are capable of reading data from some external source and capable of writing data to some external sink, will make use of this module as their basis. When creating new data sources and data sinks, one is encouraged to use modules that built on this module as reference (e.g. *StreamHandler* or *QueueHandler*). This partial implementation allows setting the mode of operation (i.e. read-only, write-only or read-and-write).
- **StreamHandler:** This is a partial implementation of a *DataHandler* which allows reading and writing of stream-based resources. This implementation manages input and output streams. It represents the middle piece between implementations of data formats and data storage options. This implementation will read and write to the standard input and output by default. But it can also be configured to any stream-based resource (see implementations of *StreamHandlerImpl* such as *FileImpl* or *UrlImpl*). This module is most useful when creating a new data format which is also why most data formats in this framework utilize this module. One may want to use concrete implementations such as *PlainText* or *TransmissionControlProtocolImpl* as reference.

Input and Output

- **FileHandler:** This is a realization of a *DataHandler* which accesses file systems entries. It is used to recursively list files and directories. By default, it will list all files of the root file system (e.g. “/” on unix-based operating systems and “C:” on Windows-based operating systems). However, it can be configured to point to any directory. During reading operations, the module will list all files and directories. During writing operations, the module will create the files and directories that are passed into it.
- **QueueHandler:** This is a realization of a *DataHandler* which allows external programs to programatically feed data into the framework and programatically read data from the framework. This module is intended to create a simple way of interfacing external programs with the framework. The actual implementation of the utilized queue can be configured (by default a *LinkedList* is used). In contrast to the *BlockingQueueHandler*, this module expects external entities to provide their data samples in a timely manner. The framework will process all data samples in the queue until it is empty. This assumes that the external entity can provide data samples with at least the same timing that the framework requires to process them. Once the queue is empty (either because all samples were processed or the external entity could not keep up) processing will stop.
- **BlockingQueueHandler:** This is a realization of a *DataHandler* which allows external programs to communicate with the framework. It allows external entities to push and pull data into / from the framework thus circumventing the otherwise (mostly) stream-based interface options. The actual implementation of the utilized queue can be configured to any *BlockingQueue* (by default a *LinkedBlockingQueue* is used). In contrast to the *QueueHandler*, this module will block during reading and writing operations when necessary. This can be especially useful when the external entity wants to provide data samples at its own pace rather than adhering to the timing requirements of the framework.
- **UserDatagramProtocolHandler:** This is a realization of a *DataHandler* which provides access to UDP-based services. The module can be used to retrieve and transmit UDP-based datagrams. Depending on the mode, this implementation can either retrieve (read-only), send (write-only) or both (read-and-write). The server and port with which the communication is established are configurable. The buffer used for reading datagrams can also be configured (by default 2048 bytes are allocated).
- **Linear:** This implementation acts as a data source for a predefined (and linear) sequence of numbers. The generated numbers follow the pattern $y = m * x + b$, where y corresponds to the returned number, m defines the slope, b represents the offset (for $x = 0$) and x sets the starting point. Any variable on the right hand-side of the expression can be configured. Every call to this module will increase the value of x by one (default value) and return the y -value. However, the step width (for x) between two successive calls can also be configured to any real number.
- **Random:** This implementation acts as a data source for pseudo random numbers. By default, it outputs a random number between zero (inclusive) and one (exclusive). The upper and lower boundary can be configured.

The seed that is used to generate pseudo random numbers can also be configured.

- **Time:** This implementation acts as a data source. By default it outputs the current time in milliseconds since January first 1970. The implementation will do so regardless of the input data, if any. The module can also be configured to return relative time values. In this mode, the first call to this module will output zero and all succeeding calls will output the time (in milliseconds) that elapsed since the first call.
- **Null:** This module is mostly intended for debugging purposes. Regardless of the input data, it will always return exactly one “null”-value. This is basically an empty generator.
- **Classes:** This implementation iterates all accessible classes within the current class path and outputs their fully-qualified name. The returned classes are in no particular order. The class path can be configured to consider only classes within a particular directory or jar-file. By default, the entire Java class path is searched and returned (i.e. including all classes within the Java runtime environment).

Data Formats

- **AbstractTextFormat:** This is a partial realization of a *StreamHandler* which provides common functionality for reading and writing text-based data formats. Here, the character set, size of buffer for reading and writing operations as well as the line separator can be configured. When creating new text-based data formats, one may want to use *PlainText* or *CommaSeparatedValues* as reference implementations.
- **CommaSeparatedValues:** This is a realization of an *AbstractTextFormat*. It allows the framework to read and write data samples in the form of comma separated values (CSV). In addition to the configuration options from *AbstractTextFormat* (i.e. buffer size, line separator and character set) and *StreamHandler* (i.e. whether data is read from files, websites, etc.), this module can also be configured to include a header line. The separator between fields can be configured as well.
- **PlainText:** This is a realization of an *AbstractTextFormat*. It allows the framework to read and write plain text. The module can be configured to enforce a line separator after writing a data sample. It can also be configured to prepend a prefix before any writing operation.
- **Serialization:** This is a realization of a *StreamHandler*. It allows the framework to read and write Java objects directly from files, network streams, etc. It utilizes the serialization mechanism that are part of the Java runtime environment.

Decorators

- **BreakOut:** This is a realization of a *DecoratorProcessor* that is mostly intended for debugging purposes. This implementation provides access to the most recent set of input and output data that were passed to the *process*-method. It also counts the number of calls to various methods (e.g. *setUp*, *process* and *dismantle*).

- **Counter:** This is a realization of a *DecoratorProcessor* that is mostly intended for debugging purposes. This implementation counts the number of times that the *process*-method of the decorated module was called. Here, the count can be retrieved as a parameter.
- **Logger:** This is a realization of a *DecoratorProcessor* that is mostly intended for debugging purposes. This implementation logs all method calls including their parameters and return values. It utilizes Java's built-in logging mechanisms and it can be configured to use any available logger. This implementation is most likely to be used when detailed access logs are required.
- **Sleep:** This is a realization of a *DecoratorProcessor* that is mostly intended for debugging purposes. This implementation waits a defined number of milliseconds before calling the decorated module. It is useful for slowing down the processing of data within the framework. The wait time could be configured to a length that processing can be observed as it happens (e.g. log messages can be read by a human as processing happens).
- **SystemOut:** This is a realization of a *DecoratorProcessor* that is mostly intended for debugging purposes. This implementation outputs the input and output data of the decorated module. A name can be configured which would also be outputted.
- **Time:** This is a realization of a *DecoratorProcessor* that is mostly intended for debugging purposes. This implementation measures the number of milliseconds that the decorated module requires for processing. It can be used to analyze processing time of modules and optimize time efficiency.

Miscellaneous

- **BufferingProcessor:** This is a partial implementation of a module which provides a sliding window. It is intended to buffer incoming data (in an ordered fashion) and provide access to it. The size of the sliding window can be configured to any positive integer (including zero). Setting the length to zero will disable buffering and all input data are directly accessible.
- **ComparingProcessor:** This is a partial implementation of the main interface (i.e. *Processor*) which buffers a single sample. The idea is to provide the ability to compare an incoming sample to the previous sample without having to use a *BufferingProcessor* module. By default it compares input data from the first port (i.e. if multiple modules output their data into this implementation then data from the first one is used only). However, this number can be configured to any inbound module.
- **MovingAverage:** This is a realization of the *BufferingProcessor* which is used to calculate a moving average for its input data. The number of data samples that are considered for the moving average can be configured. By setting the mode, it can also be configured how the moving average is calculated in the initial phase where the buffer is only partially filled (i.e. the number of data samples in the buffer is smaller than the actual size of the buffer). In general, the options include: waiting for the buffer to

fill up, using the current size of the buffer or using the actual size of the buffer.

- **Forward:** This module is mostly intended for debugging purposes. It forwards all input data as output data. This can be useful when two or more paths within the graph of modules need to be normalized (i.e. their length must be identical before they can be merged).
- **NoOperation:** This module is intended for debugging purposes. This implementation does nothing (i.e. no forwarding, no processing, etc.) and basically represents a dead end in the graph of modules.
- **ToString:** This implementation is mostly intended for debugging purposes. It converts all input values into String-objects. Null values are simply forwarded. This is especially useful when a sequence of data needs to be dumped in a plain text file or written to the standard input-output (i.e. terminal or console).
- **Delay:** This is a realization of the *BufferingProcessor* which allows the delayed forwarding of input data. The delay can be configured to any positive integer (including zero). Setting the delay to zero forwards input data without a delay. This implementation can be used (similarly to *Forward*) to avoid misalignments when multiple paths of varying lengths within a graph of modules need to be merged.
- **Distance:** This is a realization of the *ComparingProcessor* which determines the distance between two successive input data. The module can be configured to use any port (i.e. if multiple modules output their data into this module then any one of them can be used). By default the first module is used.
- **Function:** This is a realization of a *ProcessingAdapter* and it returns a predefined sequence of numbers (similarly to *Linear*). This implementation returns one by default. However, the mathematical expression that is used to generate the sequence can be configured. The input data from one or multiple inbound modules are used as input values for the expression. This module is especially powerful when used in combination with *Linear*. Not just linear expressions, but also functions (e.g. *sin*, *cos*, etc.) as well as exponential expressions can be utilized.

B.4 Road-Map

This section describes those modules that are required to implement the algorithms presented in Chapter 6. For each of the modules, their purpose is outlined and special attention is given to their function within the implemented algorithm. Figure B.1 shows a general overview of how the modules can be organized to implement the proposed algorithms.

- **FileHandler:** Assuming that recorded signals are stored in separate files for each patient and recording session (see Chapter 5), then a module is required to find and list these files. This module would be configured to point at the root directory of the database. The module would then recursively find all files and output them (e.g. the recordings could be organized by medical site and patient).

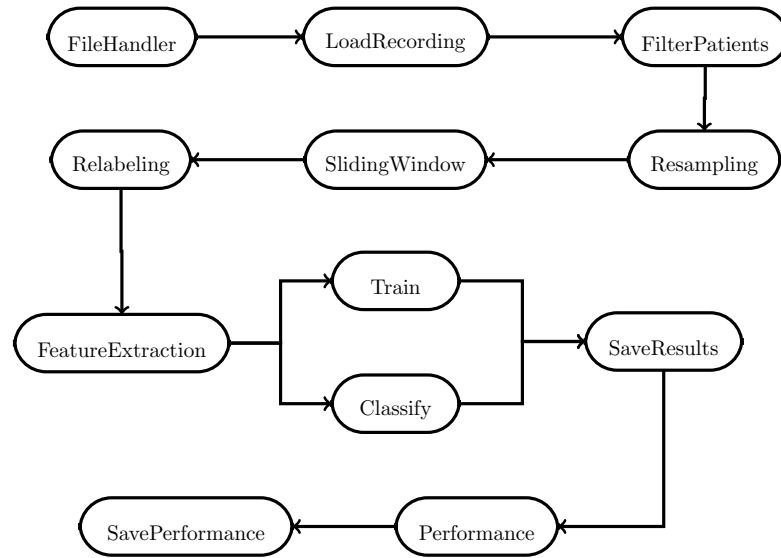


Figure B.1: Illustrates a possible way of implementing the algorithms presented in Chapter 6.

- **LoadRecording:** This module would be fed with the files found by *FileHandler* and it will have to figure out whether a particular file corresponds to recorded signals or some other arbitrary file (e.g. patient consent form, summary file, etc.). This could be done by inspecting the file ending or analyzing the filename. Once the module has identified a file with recorded signals, the contents need to be loaded into memory. Here, the module should be configurable to only load data from the wrist sensor or only data from the waist sensor platform. Furthermore, the actual sensor should be specifiable (i.e. accelerometer, magnetometer or gyroscope) and whether the recorded signals should be trimmed (i.e. whether those parts of the recording that correspond to signals before / after the sensor was attached to the patient should be removed).
- **FilterPatient:** The loaded recordings may need to be filtered. Training of a support vector machine (SVM) requires only patients from the training dataset while testing expects only patients from the testing dataset. The configuration of this module needs to specify which patients are required for the current goal or action (i.e. training or testing). Depending on the structure of database, this might already be possible without loading the actual signals.
- **Resampling:** This module resamples the raw sensor data. It is intended to reduce the overall data and thus ease the processing within successive modules. Here, the module only requires a sampling frequency to be configured (e.g. 40Hz compared to the original sampling rate of 80Hz and 200Hz of the wrist and waist sensor platform).
- **SlidingWindow:** This module would extract the starting and ending point for each window within the recorded signals. Each window would be expected to have the same size and successive windows should always

have the same overlap (i.e. they should start at regular intervals). Both values, length and step, need to be configurable.

- **Relabeling:** The labels for each window are analyzed and unified for further processing. This module needs to be supplied with details on how positive labels, negative labels and “undefined” labels can be identified. This module is responsible for aggregating the original (and resampled) labels into a single label for each window. Depending on whether an SVM is currently being trained or whether an SVM is being tested, the module may offer the option to automatically relabel signals (i.e. when a patient only has negative labels or “undefined” labels but not a single positive label, then the entire recording could be relabeled with negative labels).
- **FeatureExtraction:** This module would also utilize the starting and ending points for the windows to extract a set of features. Only the data within each window are used to extract features that are related to the symptom being trained or tested. The module should have at least an option to specify whether the full or reduced feature set should be used (see Chapter 6).
- **Train:** This module performs the actual training of an SVM. It requires a kernel function (e.g. linear or radial basis function (RBF)), cost and gamma values. Furthermore, the number of folds used during cross validation needs to be specified as well as a place where the trained SVM model should be stored (including any normalization parameters).
- **Classify:** This module uses the extracted features to classify each window and determine whether a particular symptom is detected (or not). This module needs to be told where the SVM model resides such that it can load the model, normalize the data and start recognizing the symptom.
- **SaveResults:** This module is used to persist or save any results that were obtained up to this point. This is mainly to have something for later reference and for verifying the correct functioning of preceding modules. Here, only a path for storing the results and whether they should be appended to existing results are required.
- **Performance:** This module uses the classification results to determine the actual performance. Here, the module may be configured to perform another aggregation in order to apply a meta-analysis. The level of aggregation should be specified (i.e. if any aggregation is to be performed). In case of any aggregation, the lower and upper thresholds (see Chapter 6) need to be specified as well.
- **SavePerformance:** This module is much like the *SaveResults* module, but instead of persisting the features and classification results, the actual performance in terms of true positives (TPs), true negatives (TNs), false positives (FPs) and false negatives (FNs) are stored. This module also only requires a place to store the data and whether it should append the data to existing performance results.

Appendix C

Additional Results in Signifying Motor Symptoms

C.1 Tremor (at Rest)

C.1.1 Any Tremor at Waist

See Table C.1, Table C.2, Table C.3 and Table C.4.

C.1.2 Lower Tremor at Waist

See Table C.5, Table C.6, Table C.7 and Table C.8.

C.1.3 Upper Tremor at Waist

See Table C.9, Table C.10, Table C.11 and Table C.12.

C.2 Dyskinesia

C.2.1 Any Dyskinesia at Waist

See Table C.13, Table C.14, Table C.15 and Table C.16.

	Training	Test
Number of tremor windows	1689	8591
Number of non-tremor windows	1077	97244
Number of recordings in ON state	3	86
Number of recordings in OFF state	6	90
Overall number of recordings	9	176

Table C.1: Lists the number of windows (before aggregation) that are used for signifying tremor (i.e. any tremor).

Kernel	RBF	Linear	RBF	Linear
Features	Freq.	Freq.	All	All
Sensitivity (train)	0.673	0.531	0.756	0.780
Specificity (train)	0.492	0.525	0.411	0.499
Data Usage (train)	1.000	1.000	1.000	1.000
Geometric Mean (train)	0.575	0.528	0.557	0.624
Accuracy (train)	0.495	0.525	0.418	0.504
Sensitivity (test)	0.646	0.600	0.752	0.646
Specificity (test)	0.533	0.559	0.408	0.504
Data Usage (test)	1.000	1.000	1.000	1.000
Geometric Mean (test)	0.587	0.579	0.554	0.571
Accuracy (test)	0.543	0.563	0.438	0.517

Table C.2: Results (any-tremor-waist) for detecting tremor using the naive approach.

Kernel	RBF	Linear	RBF	Linear
Features	Freq.	Freq.	All	All
t	60	45	60	15
th	0.750	0.550	0.800	0.900
Sensitivity (train)	0.533	0.500	0.600	0.750
Specificity (train)	0.780	0.625	0.770	0.815
Data Usage (train)	1.000	1.000	1.000	1.000
Geometric Mean (train)	0.645	0.559	0.679	0.782
Accuracy (train)	0.768	0.619	0.762	0.813
Sensitivity (test)	0.434	0.579	0.565	0.349
Specificity (test)	0.827	0.636	0.695	0.808
Data Usage (test)	1.000	1.000	1.000	1.000
Geometric Mean (test)	0.599	0.607	0.626	0.531
Accuracy (test)	0.785	0.630	0.681	0.765

Table C.3: Results (any-tremor-waist) for detecting tremor using the one-sided approach.

Kernel Features	RBF Freq.	Linear Freq.	RBF All	Linear All
t	25	20	60	45
th_l	0.150	0.100	0.350	0.400
th_h	1.000	0.950	1.000	1.000
Sensitivity (train)	0.777	0.750	1.000	0.769
Specificity (train)	0.807	0.715	0.871	0.880
Data Usage (train)	0.218	0.215	0.206	0.452
Geometric Mean (train)	0.792	0.732	0.933	0.823
Accuracy (train)	0.806	0.715	0.878	0.873
Sensitivity (test)	0.645	0.670	0.681	0.480
Specificity (test)	0.811	0.755	0.790	0.821
Data Usage (test)	0.261	0.257	0.296	0.508
Geometric Mean (test)	0.723	0.711	0.733	0.628
Accuracy (test)	0.793	0.747	0.774	0.787

Table C.4: Results (any-tremor-waist) for detecting tremor using the two-sided approach.

	Training	Test
Number of tremor windows	44	1204
Number of non-tremor windows	4071	112747
Number of recordings in ON state	3	86
Number of recordings in OFF state	6	87
Overall number of recordings	9	173

Table C.5: Lists the number of windows (before aggregation) that are used for signifying tremor (i.e. lower tremor).

Kernel Features	RBF Freq.	Linear Freq.	RBF All	Linear All
Sensitivity (train)	0.000	0.107	0.160	0.178
Specificity (train)	0.999	0.996	0.996	0.995
Data Usage (train)	1.000	1.000	1.000	1.000
Geometric Mean (train)	0.000	0.326	0.400	0.421
Accuracy (train)	0.994	0.993	0.993	0.992
Sensitivity (test)	0.040	0.042	0.064	0.041
Specificity (test)	0.998	0.994	0.991	0.991
Data Usage (test)	1.000	1.000	1.000	1.000
Geometric Mean (test)	0.202	0.206	0.252	0.203
Accuracy (test)	0.987	0.984	0.981	0.980

Table C.6: Results (lower-tremor-waist) for detecting tremor using the naive approach.

Kernel Features	RBF Freq.	Linear Freq.	RBF All	Linear All
t	45	25	25	25
th	0.300	0.100	0.200	0.250
Sensitivity (train)	0.000	0.400	0.400	0.400
Specificity (train)	1.000	0.990	0.996	0.995
Data Usage (train)	1.000	1.000	1.000	1.000
Geometric Mean (train)	0.000	0.629	0.631	0.631
Accuracy (train)	0.992	0.986	0.993	0.992
Sensitivity (test)	0.089	0.112	0.146	0.101
Specificity (test)	1.000	0.981	0.985	0.989
Data Usage (test)	1.000	1.000	1.000	1.000
Geometric Mean (test)	0.298	0.332	0.379	0.316
Accuracy (test)	0.986	0.970	0.974	0.977

Table C.7: Results (lower-tremor-waist) for detecting tremor using the one-sided approach.

Kernel Features	RBF Freq.	Linear Freq.	RBF All	Linear All
t	10	25	25	25
th_l	0.100	0.100	0.250	0.250
th_h	0.400	0.300	0.350	0.300
Sensitivity (train)	0.000	0.400	0.400	0.400
Specificity (train)	0.999	1.000	1.000	1.000
Data Usage (train)	0.996	0.990	0.997	0.995
Geometric Mean (train)	0.000	0.632	0.632	0.632
Accuracy (train)	0.994	0.996	0.996	0.996
Sensitivity (test)	0.053	0.092	0.113	0.090
Specificity (test)	0.999	0.996	0.994	0.992
Data Usage (test)	0.994	0.984	0.992	0.996
Geometric Mean (test)	0.230	0.302	0.336	0.300
Accuracy (test)	0.988	0.984	0.983	0.980

Table C.8: Results (lower-tremor-waist) for detecting tremor using the two-sided approach.

	Training	Test
Number of tremor windows	1645	7455
Number of non-tremor windows	1077	98864
Number of recordings in ON state	3	86
Number of recordings in OFF state	6	90
Overall number of recordings	9	176

Table C.9: Lists the number of windows (before aggregation) that are used for signifying tremor (i.e. upper tremor).

Kernel Features	RBF Freq.	Linear Freq.	RBF All	Linear All
Sensitivity (train)	0.590	0.536	0.751	0.926
Specificity (train)	0.482	0.474	0.405	0.493
Data Usage (train)	1.000	1.000	1.000	1.000
Geometric Mean (train)	0.533	0.504	0.551	0.676
Accuracy (train)	0.483	0.474	0.409	0.499
Sensitivity (test)	0.665	0.668	0.751	0.621
Specificity (test)	0.528	0.515	0.404	0.499
Data Usage (test)	1.000	1.000	1.000	1.000
Geometric Mean (test)	0.593	0.587	0.551	0.556
Accuracy (test)	0.539	0.527	0.431	0.508

Table C.10: Results (upper-tremor-waist) for detecting tremor using the naive approach.

Kernel Features	RBF Freq.	Linear Freq.	RBF All	Linear All
t	30	20	15	15
th	0.700	0.650	0.850	0.900
Sensitivity (train)	0.555	0.523	0.642	0.892
Specificity (train)	0.679	0.620	0.717	0.812
Data Usage (train)	1.000	1.000	1.000	1.000
Geometric Mean (train)	0.614	0.569	0.679	0.851
Accuracy (train)	0.676	0.617	0.716	0.814
Sensitivity (test)	0.509	0.599	0.539	0.305
Specificity (test)	0.741	0.658	0.675	0.804
Data Usage (test)	1.000	1.000	1.000	1.000
Geometric Mean (test)	0.614	0.628	0.603	0.495
Accuracy (test)	0.721	0.653	0.664	0.763

Table C.11: Results (upper-tremor-waist) for detecting tremor using the one-sided approach.

Kernel Features	RBF Freq.	Linear Freq.	RBF All	Linear All
t	45	45	60	60
th_l	0.200	0.150	0.250	0.400
th_h	0.950	1.000	1.000	1.000
Sensitivity (train)	0.666	1.000	1.000	1.000
Specificity (train)	0.791	0.853	0.755	0.858
Data Usage (train)	0.213	0.095	0.157	0.421
Geometric Mean (train)	0.726	0.923	0.868	0.926
Accuracy (train)	0.787	0.857	0.773	0.866
Sensitivity (test)	0.696	0.787	0.777	0.410
Specificity (test)	0.836	0.850	0.714	0.812
Data Usage (test)	0.248	0.157	0.228	0.488
Geometric Mean (test)	0.763	0.818	0.745	0.577
Accuracy (test)	0.823	0.845	0.723	0.778

Table C.12: Results (upper-tremor-waist) for detecting tremor using the two-sided approach.

	Training	Test
Number of dyskinesia windows	2825	10639
Number of non-dyskinesia windows	5721	84031
Number of recordings in ON state	7	82
Number of recordings in OFF state	6	90
Overall number of recordings	13	172

Table C.13: Lists the number of windows (before aggregation) that are used for signifying dyskinesia (i.e. trunk dyskinesia).

Kernel Features	RBF Freq.	Linear Freq.	RBF All	Linear All
Sensitivity (train)	0.747	0.814	0.787	0.788
Specificity (train)	0.598	0.568	0.559	0.513
Data Usage (train)	1.000	1.000	1.000	1.000
Geometric Mean (train)	0.668	0.680	0.663	0.636
Accuracy (train)	0.622	0.609	0.597	0.559
Sensitivity (test)	0.678	0.672	0.676	0.732
Specificity (test)	0.573	0.545	0.540	0.519
Data Usage (test)	1.000	1.000	1.000	1.000
Geometric Mean (test)	0.623	0.605	0.604	0.616
Accuracy (test)	0.584	0.559	0.555	0.542

Table C.14: Results (any-dyskinesia-waist) for detecting dyskinesia using the naive approach.

Kernel Features	RBF Freq.	Linear Freq.	RBF All	Linear All
t	60	60	60	60
th	0.600	0.750	0.750	0.700
Sensitivity (train)	0.753	0.726	0.685	0.685
Specificity (train)	0.759	0.844	0.808	0.826
Data Usage (train)	1.000	1.000	1.000	1.000
Geometric Mean (train)	0.756	0.783	0.744	0.752
Accuracy (train)	0.758	0.815	0.778	0.791
Sensitivity (test)	0.660	0.514	0.584	0.541
Specificity (test)	0.739	0.835	0.846	0.809
Data Usage (test)	1.000	1.000	1.000	1.000
Geometric Mean (test)	0.698	0.655	0.703	0.661
Accuracy (test)	0.727	0.788	0.807	0.769

Table C.15: Results (any-dyskinesia-waist) for detecting dyskinesia using the one-sided approach.

Kernel Features	RBF Freq.	Linear Freq.	RBF All	Linear All
t	30	45	60	60
th_l	0.100	0.100	0.300	0.250
th_h	1.000	1.000	0.900	0.900
Sensitivity (train)	0.830	0.900	0.833	0.886
Specificity (train)	0.938	0.979	0.914	0.937
Data Usage (train)	0.200	0.229	0.357	0.330
Geometric Mean (train)	0.882	0.939	0.873	0.911
Accuracy (train)	0.892	0.943	0.887	0.918
Sensitivity (test)	0.787	0.717	0.770	0.733
Specificity (test)	0.845	0.818	0.860	0.841
Data Usage (test)	0.160	0.161	0.300	0.271
Geometric Mean (test)	0.815	0.766	0.813	0.785
Accuracy (test)	0.833	0.792	0.842	0.820

Table C.16: Results (any-dyskinesia-waist) for detecting dyskinesia using two-sided approach.

	Training	Test
Number of dyskinesia windows	660	4111
Number of non-dyskinesia windows	5721	99144
Number of recordings in ON state	7	82
Number of recordings in OFF state	6	90
Overall number of recordings	13	172

Table C.17: Lists the number of windows (before aggregation) that are used for signifying dyskinesia (i.e. trunk dyskinesia).

Kernel Features	RBF Freq.	Linear Freq.	RBF All	Linear All
Sensitivity (train)	0.761	0.562	0.738	0.558
Specificity (train)	0.632	0.672	0.632	0.655
Data Usage (train)	1.000	1.000	1.000	1.000
Geometric Mean (train)	0.694	0.614	0.682	0.605
Accuracy (train)	0.639	0.666	0.637	0.651
Sensitivity (test)	0.671	0.618	0.673	0.636
Specificity (test)	0.659	0.704	0.682	0.696
Data Usage (test)	1.000	1.000	1.000	1.000
Geometric Mean (test)	0.665	0.660	0.678	0.666
Accuracy (test)	0.660	0.701	0.682	0.694

Table C.18: Results (trunk-and-strong-limb-dyskinesia-waist) for detecting dyskinesia using the naive approach.

C.2.2 Trunk + Strong Limb Dyskinesia at Waist

See Table C.17, Table C.18, Table C.19 and Table C.20.

Kernel Features	RBF Freq.	Linear Freq.	RBF All	Linear All
t	45	60	45	60
th	0.550	0.550	0.550	0.450
Sensitivity (train)	0.778	0.593	0.750	0.667
Specificity (train)	0.744	0.795	0.741	0.671
Data Usage (train)	1.000	1.000	1.000	1.000
Geometric Mean (train)	0.760	0.687	0.745	0.669
Accuracy (train)	0.747	0.779	0.742	0.671
Sensitivity (test)	0.677	0.593	0.706	0.721
Specificity (test)	0.780	0.824	0.820	0.747
Data Usage (test)	1.000	1.000	1.000	1.000
Geometric Mean (test)	0.726	0.699	0.761	0.734
Accuracy (test)	0.775	0.812	0.814	0.746



Table C.19: Results (trunk-and-strong-limb-dyskinesia-waist) for detecting dyskinesia using the one-sided approach.

Kernel Features	RBF Freq.	Linear Freq.	RBF All	Linear All
t	60	25	60	30
th_l	0.150	0.100	0.150	0.150
th_h	0.950	0.900	0.950	0.900
Sensitivity (train)	1.000	0.778	1.000	0.706
Specificity (train)	0.881	0.835	0.915	0.871
Data Usage (train)	0.286	0.385	0.277	0.366
Geometric Mean (train)	0.939	0.806	0.956	0.784
Accuracy (train)	0.893	0.832	0.922	0.858
Sensitivity (test)	0.737	0.692	0.718	0.620
Specificity (test)	0.962	0.893	0.973	0.946
Data Usage (test)	0.262	0.369	0.273	0.354
Geometric Mean (test)	0.842	0.786	0.836	0.766
Accuracy (test)	0.950	0.883	0.959	0.932

Table C.20: Results (trunk-and-strong-limb-dyskinesia-waist) for detecting dyskinesia using the two-sided approach.

Appendix D

Questionnaires

	Personal Health Device for the Remote and Autonomous Management of Parkinson's Disease	
Documento	CRF_ Screening and baseline REMPARK	Revisió: 1 Data: 1/10/2012 Pàg.:

PARTNER

QUESTIONNAIRE NUMBER:.....

TEKNON.....	1
NUIG.....	2
FSL.....	3
NEVET.....	4

Date:

REMPARK Project Screening visit and baseline visit

NAME: _____
 SURNAME: _____
 PHONE NUMBER: _____
 ADDRESS: _____

IDENTIFYING LABEL

INFORMATION:

The object of study is to create a technological device that automatically recognizes different motor symptoms in Parkinson's patients. To do so, you will be visited by researchers at home twice. The first day will be asked questions about your Parkinson's disease and your overall health. The second day will be placed two devices at the waist and wrist and will be asked to perform a series of activities and movements both in ON phase and OFF phase. To induce the OFF state, you may be asked to discontinue your Parkinson's pills for 12 hours prior to the experiment (as you know, your neurologist is aware of the experiment and the possible discontinuation of treatment for 12 hours). The second day will also be asked to perform their normal activity for a few hours, taking positions devices. During those hours will be accompanied at all times by a researcher will collect data about your symptoms and their movements, which are important to develop the device. Some of the test will be video recorded.

We appreciate very much your cooperation. Please read carefully the information sheet we deliver. We are always available to answer your questions.

INTERVIEWER

- ☐ I have informed the patient of the study procedures..... 1
- ☐ I have left a copy of the patient information sheet..... 2

INCLUSION CRITERIA:

I.1 - Did he/she sign the consent form?

- No..... 1
- Yes..... 2

A. SOCIODEMOGRAPHIC

A.1 Sex:

- Male 1
- Female 2

A.2 Age: years old

A.3 Marital Status (*INT, read*).

Single	1
Married	2
Living with partner	3
Separated	4
Widowed	5
<i>(Don't read)</i> NO ANSWER	9

A.4 Until what age studied?

Age: years old

B. HUGHES et al DIAGNOSIS CRITERIA FOR PARKINSON'S DISEASE.

B. 1. DIAGNOSIS OF PARKINSONIAN SYNDROME

B. 1. 1 Bradykinesia

(Slowness of initiation of voluntary movement with progressive reduction in speed and amplitude of repetitive actions)

- ☐ No1
☐ Yes2

B. 1. 2 And at least one of the following:

- 1.- Muscular rigidity
☐ No..... 1
☐ Yes 2
- 2.- 4-6 Hz rest tremor
☐ No..... 1
☐ Yes 2
- 3.- Postural instability not caused by primary visual, vestibular, cerebellar, or proprioceptive dysfunction.
☐ No..... 1
☐ Yes 2

B. 2. EXCLUSION CRITERIA FOR PARKINSON'S DISEASE:

- 1.- History of repeated strokes with stepwise progression of parkinsonian features
☐ No1
☐ Yes2
- 2.- History of repeated head injury
☐ No1
☐ Yes2
- 3.- History of definite encephalitis
☐ No1
☐ Yes2
- 4.- Oculogyric crises
☐ No1
☐ Yes2
- 5.- Neuroleptic treatment at onset of symptoms
☐ No1
☐ Yes2
- 6.- More than one affected relative
☐ No1
☐ Yes2
- 7.- Sustained remission
☐ No1
☐ Yes2
- 8.- Strictly unilateral features after 3 years
☐ No1
☐ Yes2

- 9.- Supranuclear gaze palsy
☐ No..... 1
☐ Yes 2
- 10.- Cerebellar signs
☐ No..... 1
☐ Yes 2
- 11.- Early severe autonomic involvement
☐ No..... 1
☐ Yes 2
- 12.- Early severe dementia with disturbances of memory, language, and praxis
☐ No..... 1
☐ Yes 2
- 13.- Babinski sign
☐ No..... 1
☐ Yes 2
- 14.- Presence of cerebral tumour or communicating hydrocephalus on CT scan
☐ No..... 1
☐ Yes 2
- 15.- Negative response to large doses of Levodopa (if malabsorption excluded)
☐ No..... 1
☐ Yes 2
- 16.- MPTP exposure
☐ No..... 1
☐ Yes 2

INCLUSION CRITERIA:

I.2 - ¿Does the patient fulfill the Parkinson's Disease criteria?

- ☐ No1
☐ Yes.....2

C. MODIFIED HOEHN AND YAHR STAGING

(Mark the appropriate option).

No signs of disease.....	0
Unilateral disease	1
Unilateral plus axial involvement	1,5
Bilateral disease, without impairment of balance	2
Mild bilateral disease, with recovery on pull test.....	2,5
Mild to moderate bilateral disease; some postural instability; physically independent	3
Severe disability; still able to walk or stand unassisted.....	4
Wheelchair bound or bedridden unless aided	5

INCLUSION CRITERIA:

1.3 - In the best state (ON state)... does he/she have a score on the Hoehn and Yahr staging greater than 2?

- No..... 1
- Yes..... 2

NOTE: Patients with Hoehn and Yahr of 2 could be included if it is expressly authorized by the recruitment coordinator.

D. CLINICAL FLUCTUATIONS

1. Are "off" periods predictable?

- No0
- Yes1

2. Are "off" periods unpredictable?

- No 0
- Yes1

3. Do "off" periods come on suddenly, within a few seconds?

- No0
- Yes 1

4. What proportion of the waking day is the patient "off" on average?

- None..... 0
- 1-25% of day 1
- 26-50% of day..... 2
- 51-75% of day..... 3
- 76-100% of day..... 4

INCLUSION CRITERIA:

1.4a - Does he/she have clinical fluctuations?

(Any positive answer to the 4 previous questions)

- No..... 1
- Yes..... 2

INCLUSION CRITERIA:

1.4b –Can you walk without human or technical (cane, walker, ...) assistance when you are "off"?

- No..... 1
- Yes..... 2

INCLUSION CRITERIA:**I.5 - Is he/she aged between 50 and 75 years old?**

- No 1
- Yes..... 2

NOTE: in some cases the recruitment coordinator may authorized the inclusion of patients ageing up to 80.

Does he meet all the criteria for inclusion?

- No 1 (END OFF QUESTIONNAIRE)
- Yes..... 2

E. FREEZING OF GAIT:

All answers, except in response to item 3, should be based on the experience over the last week.

This questionnaire should be completed by the researcher after asking and demonstrating freezing phenomenon, if necessary.

1.- During your worst state – do you walk:

- normally 0
- almost normally ... somewhat slow.. 1
- slow but fully independent..... 2
- need assistance or walking aid..... 3
- unable to walk 4

2.- Are your gait difficulties affecting your daily activities and independence?

- not at all..... 0
- mildly 1
- moderately 2
- severely 3
- unable to walk 4

3.- Do you feel that your feet get glued to the floor while walking, making a turn or when trying to initiate walking (freezing)?

- never 0
- very rarely: about once a month 1
- rarely: about once a week 2
- often: about once a day 3
- always: whenever walking 4

4.- How long is your longest freezing episode?

- never happened 0
- 1 – 2 seconds 1
- 3 – 10 seconds..... 2
- 11 – 30 seconds..... 3
- unable to walk for more than 30 seconds..... 4

5.- How long is your typical start hesitation episode (freezing when initiating the first step)?

- none 0
- takes longer than 1 second to start walking..... 1
- takes longer than 3 seconds to start walking..... 2
- takes longer than 10 seconds to start walking..... 3
- takes longer than 30 seconds to start walking..... 4

6.- How long is your typical turning hesitation: (freezing when turning)

- none 0
- resume turning in 1 to 2 seconds..... 1
- resume turning in 3 to 10 seconds..... 2
- resume turning in 11 to 30 seconds 3
- unable to resume turning for more than 30 seconds 4

F. DYSKINESIAS

1.- Duration: What proportion of the waking day are dyskinesias present?
(Historical information.)

- None.....0
- 1-25% of day.....1
- 26-50% of day.....2
- 51-75% of day.....3
- 76-100% of day.....4

2.- Disability: How disabling are the dyskinesias?
(Historical information; may be modified by office examination.)

- Not disabling.....0
- Mildly disabling.....1
- Moderately disabling.....2
- Severely disabling.....3
- Completely disabled.....4

3.- Painful Dyskinesias: How painful are the dyskinesias?

- No painful dyskinesias 0
- Slight 1
- Moderate..... 2
- Severe 3
- Marked..... 4

4.- Presence of Early Morning Dystonia
(Historical information)

- No..... 0
- Yes..... 1

G. CLINICAL EVALUATION

G. 1. Past medical history

DISEASE	YES	NO	DK		DISEASE	YES	NO	DK
1. High blood pressure	1	2	9		11. Urinary incontinence	1	2	9
2. Heart infarct	1	2	9		12. High cholesterol	1	2	9
3. Other heart conditions	1	2	9		13. Depression	1	2	9
4. Arthritis, osteoarthritis or rheumatic conditions	1	2	9		14. Anxiety disorder (diagnosed by a doctor)	1	2	9
5. Back ache (cervical)	1	2	9		15. Stroke, cerebral embolism, cerebral infarct or cerebral bleeding in the past.	1	2	9
6. Back ache (lumbar)	1	2	9		16. Cancer (malignant tumors)	1	2	9
7. Asthma	1	2	9		17. Osteoporosis	1	2	9
8. Chronic bronchitis	1	2	9		18. Dementia (diagnosed by a doctor)	1	2	9
9. Diabetes	1	2	9		19. Thyroid disease	1	2	9
10. Pacemaker or other implantable devices	1	2	9		20. Drugs or alcohol abuse	1	2	9

G. 2. Other past medical history

G2. 1 _____

G2. 2 _____

G3. 3 _____

G. 3. Parkinson's disease

G. 3.1 Year of onset of symptoms:

G. 3.2 Year of diagnosis:

EXCLUSION CRITERIA:

E.1 - Does he / she have other health problems that impair gait or physical activity?

- No..... 1
- Yes..... 2

EXCLUSION CRITERIA:

E.2 - Is a major consumer of drugs or alcohol?

- No..... 1
- Yes..... 2

EXCLUSION CRITERIA:

E.3 - Does he/she wear a pacemaker or other implantable devices?

- No..... 1
- Yes..... 2

G. 4. Current drug regimen

G4.a - Drug 1 name

G4.b - Dose and timing of drug 1

G4.c - Drug 2 name

G4.d - Dose and timing of drug 2

G4.e - Drug 3 name

G4.f - Dose and timing of drug 3

G4.g - Drug 4 name

G4.h - Dose and timing of drug 4

G4.i - Drug 5 name

G4.j - Dose and timing of drug 5

G4.k - Drug 6 name

G4.l - Dose and timing of drug 6

G4.m - Drug 7 name

G4.n - Dose and timing of drug 7

G4.ó - Drug 8 name

G4.p - Dose and timing of drug 8

H. CLINICAL FLUCTUATIONS REPORTED BY THE PATIENT:

Mark in the box below each dose of L-dopa (milligrams) and its relationship to the ON and OFF periods. Mark also sleeping hours (see example below).

	5h	6h	7h	8h	9h	10h	11h	12h	13h	14h	15h	16h	17h	18h	19h	20h	21h	22h	23h	24h	1h	2h	3h	4h
L-DOPA																								
ON																								
OFF																								
Sleep																								

Example:

	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	1	2	3	4
L-DOPA			mg					mg				mg							mg					
ON				X	X	X	X	X	X	X				X	X	X	X	X	X	X				
OFF			X								X	X	X											
Sleep	X	X																	X	X	X	X	X	

EXCLUSION CRITERIA:

E4 - Is he/she under treatment with dopodopa (ump)?

- ☐ No 1
☐ Yes 2

EXCLUSION CRITERIA:

E.5 Does he/she have apomorphine pump?

- ☐ No 1
☐ Yes 2

EXCLUSION CRITERIA:

E.6 - Is he/she on deep brain stimulation treatment?

- ☐ No 1
☐ Yes 2

EXCLUSION CRITERIA:

E7 - Are you enrolled in another clinical trial?

- ☐ No 1
☐ Yes 2

I. Mini-Mental:

Folstein Mini-Mental State Exam		
I. ORIENTATION (Ask the following questions; correct = <input checked="" type="checkbox"/>)	Record Each Answer:	(Maximum Score = 10)
What is today's date?	Date (eg, May 21)	1 <input type="checkbox"/>
What is today's year?	Year	1 <input type="checkbox"/>
What is the month?	Month	1 <input type="checkbox"/>
What day is today?	Day (eg, Monday)	1 <input type="checkbox"/>
Can you also tell me what season it is?	Season	1 <input type="checkbox"/>
Can you also tell me the name of this hospital/clinic?	Hospital/Clinic	1 <input type="checkbox"/>
What floor are we on?	Floor	1 <input type="checkbox"/>
What city are we in?	City	1 <input type="checkbox"/>
What county are we in?	County	1 <input type="checkbox"/>
What state are we in?	State	1 <input type="checkbox"/>
II. IMMEDIATE RECALL	(correct = <input checked="" type="checkbox"/>)	(Maximum Score = 3)
Ask the subject if you may test his/her memory. Say "ball," "flag," "tree" clearly and slowly, about on second for each. Then ask the subject to repeat them. Check the box at right for each correct response. The first repetition determines the score. If he/she does not repeat all three correctly, keep saying them up to six tries until he/she can repeat them	Ball	1 <input type="checkbox"/>
	Flag	1 <input type="checkbox"/>
	Tree	1 <input type="checkbox"/>
	NUMBER OF TRIALS: _____	
III. ATTENTION AND CALCULATION		
A. Counting Backwards Test	(Record each response, correct = <input checked="" type="checkbox"/>)	(Maximum Score = 5)
Ask the subject to begin with 100 and count backwards by 7. Record each response. Check one box at right for each correct response. Any response 7 or less than the previous response is a correct response. The score is the number of correct subtractions. For example, 93, 87, 80, 72, 66 is a score of 4; 93, 86, 78, 70, 62, is 2; 92, 87, 78, 70, 65 is 0.	93	1 <input type="checkbox"/>
	86	1 <input type="checkbox"/>
	79	1 <input type="checkbox"/>
	72	1 <input type="checkbox"/>
	65	1 <input type="checkbox"/>
B. Spelling Backwards Test		
Ask the subject to spell the word "WORLD" backwards. Record each response. Use the instructions to determine which are correct responses, and check one box at right for each correct response.	D	1 <input type="checkbox"/>
	L	1 <input type="checkbox"/>
	R	1 <input type="checkbox"/>
C. Final Score	O	1 <input type="checkbox"/>
Compare the scores of the Counting Backwards and Spelling Backwards tests. Write the greater of the two scores in the box labeled FINAL SCORE at right, and use it in deriving the TOTAL SCORE.	W	1 <input type="checkbox"/>
	FINAL SCORE _____ (Max of 5 or Greater of the two Scores)	
IV. RECALL	(correct = <input checked="" type="checkbox"/>)	(Maximum Score = 3)
Ask the subject to recall the three words you previously asked him/her to remember. Check the Box at right for each correct response.	Ball	1 <input type="checkbox"/>
	Flag	1 <input type="checkbox"/>
	Tree	1 <input type="checkbox"/>

V. Language	(correct = <input checked="" type="checkbox"/>)	(Maximum Score = 9)
Naming	Watch	1 <input type="checkbox"/>
Show the subject a wrist watch and ask him/her what it is. Repeat for a pencil.	Pencil	1 <input type="checkbox"/>
Repetition		
Ask the subject to repeat "No, ifs, ands, or buts."	Repetition	1 <input type="checkbox"/>
Three -Stage Command		
Establish the subject's dominant hand. Give the subject a sheet of blank paper and say, "Take the paper in your right/left hand, fold it in half and put it on the floor."	Takes paper in hand	1 <input type="checkbox"/>
	Folds paper in half	1 <input type="checkbox"/>
	Puts paper on floor	1 <input type="checkbox"/>
Reading		
Hold up the card that reads, "Close your eyes." So the subject can see it clearly. Ask him/her to read it and do what it says. Check the box at right only if he/she actually closes his/her eyes.	Closes eyes	1 <input type="checkbox"/>
Writing		
Give the subject a sheet of blank paper and ask him/her to write a sentence. It is to be written spontaneously. If the sentence contains a subject and a verb, and is sensible, check the box at right. Correct grammar and punctuation are not necessary.	Writes sentence	1 <input type="checkbox"/>
Copying		
Show the subject the drawing of the intersecting pentagons. Ask him/her to draw the pentagons (about one inch each side) on the paper provided. If ten angles are present and two intersect, check the box at right. Ignore tremor and rotation.	Copies pentagons	1 <input type="checkbox"/>
DERIVING THE TOTAL SCORE		
Add the number of correct responses. The maximum is 30.		TOTAL SCORE _____

EXCLUSION CRITERIA:

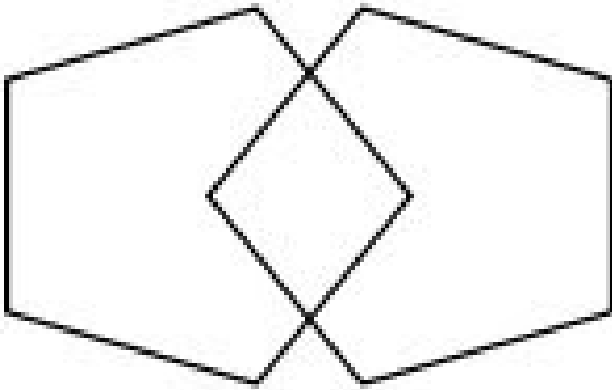
¿Total score under 23?

- No 1
- Yes 2

Does the patient meet any of the exclusion criteria?

- No, he/she does not meet exclusion criteria 2
- Yes, he /she meet any of the exclusion criteria 1 (END OFF QUESTIONNAIRE)

CLOSE YOUR EYES



J. USUAL ACTIVITIES IN THE EXPERIMENTAL VISIT:

Make a date with the participant for the experimental sessions. Explain that the recruiting coordinator must authorize further study with him, so that the date is tentative.

- Date of experimental visit _____

The participant will describe what usually he/she does in a day like that the scheduled day for the experimental visit

Morning

- 1.- _____
 - 2.- _____
 - 3.- _____
 - 4.- _____
 - 5.- _____
-
-
-
-
-
-

Afternoon



- 6.- _____
 - 7.- _____
 - 8.- _____
 - 9.- _____
 - 10.- _____
-
-
-
-
-
-

K. CONTACT THE RECRUITMENT COORDINATOR:

- Patient refused1
- Patient accepted2

L. DRUGS DISCONTINUATION: SCHEDULE A REMINDER PHONE CALL

- Date of phone call _____

	Personal Health Device for the Remote and Autonomous Management of Parkinson's Disease	
Documento	CRF_ Experimental REMPARK	Revisió: 1 Data: 1/10/2012 Pàg.:

PARTNER

TEKNON.....	1
NUIG.....	2
FSL	3
NEVET.....	4

QUESTIONNAIRE NUMBER:.....

Date:

REMPARK Project Experimental session

NAME: _____ IDENTIFYING LABEL
SURNAME: _____
PHONE NUMBER: _____
ADDRESS: _____

INTERVIEWERS'S NAME: _____
INTERVIEWERS'S SURNAME: _____

OFF PERIOD

S.1 UPDRS (OFF PERIOD)

0.1 In terms of mobility are you now:

- At your worst moment (OFF)..... 0
- At your best moment (ON)..... 1
- In an intermediate situation 2

0.2 Last L-Dopa dose time and date:

____ h / ____ m date ____/____/____

0.3 UPDRS time:

____ h / ____ m

NOTE: do not continue until the patient enters an OFF period (or is really close to it).

0.4 Describe how the patient describes his clinical status: _____

Mark the option that corresponds to the patient

1. Speech

- Normal..... 0
- Slight loss of expression, diction and/or volume..... 1
- Monotone, slurred but understandable; moderately impaired 2
- Marked impairment, difficult to understand 3
- Unintelligible 4

2. Facial Expression

- Normal..... 0
- Minimal hypomimia, could be normal "Poker Face" 1
- Slight but definitely abnormal diminution of facial expression 2
- Moderate hypomimia; lips parted some of the time..... 3
- Masked or fixed facies with severe or complete loss of facial expression; lips parted 1/4 inch or more..... 4

3. Tremor at rest (lower extremities)

- Absent 0
- Slight and infrequently present 1
- Mild in amplitude and persistent. Or moderate in amplitude, but only intermittently present 2
- Moderate in amplitude and present most of the time 3
- Marked in amplitude and present most of the time 4

4. Tremor at rest (face, lips, chin)

- Absent 0
- Slight and infrequently present 1
- Mild in amplitude and persistent. Or moderate in amplitude, but only intermittently present 2
- Moderate in amplitude and present most of the time 3
- Marked in amplitude and present most of the time 4

5. Tremor at rest (upper extremities)

- Absent 0
- Slight and infrequently present 1
- Mild in amplitude and persistent. Or moderate in amplitude, but only intermittently present 2
- Moderate in amplitude and present most of the time 3
- Marked in amplitude and present most of the time 4

6. Action or Postural Tremor of hands

- Absent 0
- Slight; present with action 1
- Moderate in amplitude, present with action 2
- Moderate in amplitude with posture holding as well as action 3
- Marked in amplitude; interferes with feeding 4

7. Axial rigidity (Judged on passive movement of the neck with patient relaxed in sitting position. Cogwheeling to be ignored)

- Absent 0
- Slight or detectable only when activated by mirror or other movements 1
- Mild to moderate 2
- Marked, but full range of motion easily achieved 3
- Severe, range of motion achieved with difficulty 4

8. Right upper extremity rigidity (Judged on passive movement of major joints with patient relaxed in sitting position. Cogwheeling to be ignored)

- Absent 0
- Slight or detectable only when activated by mirror or other movements 1
- Mild to moderate 2
- Marked, but full range of motion easily achieved 3
- Severe, range of motion achieved with difficulty 4

9. Left upper extremity rigidity (Judged on passive movement of major joints with patient relaxed in sitting position. Cogwheeling to be ignored)

- Absent 0
- Slight or detectable only when activated by mirror or other movements 1
- Mild to moderate 2
- Marked, but full range of motion easily achieved 3
- Severe, range of motion achieved with difficulty 4

10. Right lower extremity rigidity (Judged on passive movement of major joints with patient relaxed in sitting position. Cogwheeling to be ignored)

- Absent 0
- Slight or detectable only when activated by mirror or other movements 1
- Mild to moderate 2
- Marked, but full range of motion easily achieved 3
- Severe, range of motion achieved with difficulty 4

11. Left lower extremity rigidity (Judged on passive movement of major joints with patient relaxed in sitting position. Cogwheeling to be ignored.)

- Absent 0
- Slight or detectable only when activated by mirror or other movements 1
- Mild to moderate 2
- Marked, but full range of motion easily achieved 3
- Severe, range of motion achieved with difficulty 4

12. Finger Taps (Patient taps thumb with index finger in rapid succession) (Right)

- Normal..... 0
- Mild slowing and/or reduction in amplitude 1
- Moderately impaired. Definite and early fatiguing.
May have occasional arrests in movement 2
- Severely impaired. Frequent hesitation in initiating
movements or arrests in ongoing movement..... 3
- Can barely perform the task 4

13. Finger Taps (Patient taps thumb with index finger in rapid succession) (Left)

- Normal..... 0
- Mild slowing and/or reduction in amplitude 1
- Moderately impaired. Definite and early fatiguing.
May have occasional arrests in movement 2
- Severely impaired. Frequent hesitation in initiating
movements or arrests in ongoing movement..... 3
- Can barely perform the task 4

14. Hand grips (Patient opens and closes hands in rapid succession) (Right)

- Normal..... 0
- Mild slowing and/or reduction in amplitude 1
- Moderately impaired. Definite and early fatiguing.
May have occasional arrests in movement 2
- Severely impaired. Frequent hesitation in initiating
movements or arrests in ongoing movement..... 3
- Can barely perform the task 4

15. Hand grips (Patient opens and closes hands in rapid succession) (Left)

- Normal..... 0
- Mild slowing and/or reduction in amplitude 1
- Moderately impaired. Definite and early fatiguing.
May have occasional arrests in movement 2
- Severely impaired. Frequent hesitation in initiating
movements or arrests in ongoing movement..... 3
- Can barely perform the task 4

16. Rapid Alternating Movements of Hands (Pronation-supination movements of hands, vertically and horizontally, with as large an amplitude as possible, both hands simultaneously) (Right)

- Normal..... 0
- Mild slowing and/or reduction in amplitude..... 1
- Moderately impaired. Definite and early fatiguing.
May have occasional arrests in movement..... 2
- Severely impaired. Frequent hesitation in initiating
movements or arrests in ongoing movement..... 3
- Can barely perform the task 4

17. Rapid Alternating Movements of Hands (Pronation-supination movements of hands, vertically and horizontally, with as large an amplitude as possible, both hands simultaneously) (Left)

- Normal..... 0
- Mild slowing and/or reduction in amplitude..... 1
- Moderately impaired. Definite and early fatiguing.
May have occasional arrests in movement..... 2
- Severely impaired. Frequent hesitation in initiating
movements or arrests in ongoing movement..... 3
- Can barely perform the task 4

18. Leg Agility (Patient taps heel on the ground in rapid succession picking up entire leg. Amplitude should be at least 3 inches.) (Right)

- Normal..... 0
- Mild slowing and/or reduction in amplitude..... 1
- Moderately impaired. Definite and early fatiguing.
May have occasional arrests in movement..... 2
- Severely impaired. Frequent hesitation in initiating
movements or arrests in ongoing movement..... 3
- Can barely perform the task 4

19. Leg Agility (Patient taps heel on the ground in rapid succession picking up entire leg. Amplitude should be at least 3 inches.) (Left)

- Normal..... 0
- Mild slowing and/or reduction in amplitude..... 1
- Moderately impaired. Definite and early fatiguing.
May have occasional arrests in movement..... 2
- Severely impaired. Frequent hesitation in initiating
movements or arrests in ongoing movement..... 3
- Can barely perform the task 4

20. Arising from Chair (Patient attempts to rise from a straight-backed chair, with arms folded across chest.)

- Normal..... 0
- Slow; or may need more than one attempt 1
- Pushes self up from arms of seat..... 2
- Tends to fall back and may have to try more than one time, but can get up without help 3
- Unable to arise without help 4

21. Posture

- Normal erect 0
- Not quite erect, slightly stooped posture; could be normal for older person 1
- Moderately stooped posture, definitely abnormal; can be slightly leaning to one side 2
- Severely stooped posture with kyphosis; can be moderately leaning to one side 3
- Marked flexion with extreme abnormality of posture 4

22. Gait

- Normal..... 0
- Walks slowly, may shuffle with short steps, but no festination (hastening steps) or propulsion..... 1
- Walks with difficulty, but requires little or no assistance; may have some festination, short steps, or propulsion..... 2
- Severe disturbance of gait, requiring assistance..... 3
- Cannot walk at all, even with assistance 4

23. Postural Stability (Response to sudden, strong posterior displacement produced by pull on shoulders while patient erect with eyes open and feet slightly apart. Patient is prepared)

- Normal..... 0
- Retropulsion, but recovers unaided 1
- Absence of postural response; would fall if not caught by examiner 2
- Very unstable, tends to lose balance spontaneously 3
- Unable to stand without assistance 4

24. Body Bradykinesia and Hypokinesia (Combining slowness, hesitancy, decreased arm swing, small amplitude, and poverty of movement in general)

- None..... 0
- Minimal slowness, giving movement a deliberate character; could be normal for some persons. Possibly reduced amplitude 1
- Mild degree of slowness and poverty of movement which is definitely abnormal. Alternatively, some reduced amplitude..... 2
- Moderate slowness, poverty or small amplitude of movement..... 3
- Marked slowness, poverty or small amplitude of movement..... 4

S.2. SYNCHRONIZATION

1. Sync phone with Tablet:
 - a. Connect the tablet to the mobile phone by the USB cable
 - b. Run the App 'Synchronization tablet-smartphone' in the tablet
 - c. Disconnect cable
2. Start recording data with sensors:
 - a. Turn on the sensors
 - b. Sync the tablet and the sensors with the app 'Switch On Sensors'
 - c. If everything is correct the waist sensor must blink in green and blue and the wrist sensor must blink in blue
3. Start video recording in the mobile phone (app 'REMPARK Video'). The name of the video must follow the convention 'PARTNER patient_nº 1', where 1 means OFF (in S15 the motor state is changed to 2=ON).
4. Make visual synchronization while the mobile phone is recording it:
 - a. Put the waist sensor standing on the table
 - b. Leave it still during 10sec,
 - c. Drop it
 - d. Leave it lying over the table during 10 sec
5. Place the sensors to the patient and videotape it

S.3. RECORDING OF THE OFF PHASE

Equipment required:
Mobile Phone Video

Procedure:

Patient will be asked to walk at the preferred speed and following the preferred trajectory for 20 seconds. Turns are allowed (no matter if the patient walks indoors or outdoors).

S.4. INDOORS WALKING TEST IN OFF PHASE

Equipment required:
Mobile Phone Video

Procedure:

In the indoors walking test the patient will start sitting on a chair in the living room. They will stand up and show their house to the researchers, showing each room and explaining what the room is for, just like if they were trying to sell the house. After the whole visit, the patient will return to the chair in the living room and will sit down again and the test will be over.

1. Start: Hour____ / minute____ (Tablet PC time)
2. End: Hour____ / minute____ (Tablet PC time)

NOTE: If the visit lasts less than two minutes, the test will be performed twice. If the test has lasted more than two minutes, move to the next section.

1. Start: Hour____ / minute____ (Tablet PC time)
2. End: Hour____ / minute____ (Tablet PC time)

S.5. FOG PROVOCATION TEST

Equipment required:

Mobile Phone Video

Procedure:

The test will start with the patient sitting on a chair. The patient will first stand up, walk through the door (2-3 meters away), turn around and walk through the door again, and sit down in the same chair. He/she will repeat this movement up to 10 times. If FOG phenomenon does not appear, then the door can be partially shut to get a narrower path. The test will be more successful if the place where the patient turns before returning through the door is a narrow site, like a bathroom or a corner of the kitchen.

During the testing, the patient's feet shall be videotaped at all times.

CAUTION: Falls could happen during this test, the interviewers will be especially careful when performing this test. One of the interviewers shall stay close to the patient when FOG appears, and when the patient turns around.

1. Start: Hour____ / minute____ (Tablet PC time)
2. End: Hour____ / minute____ (Tablet PC time)

S.6a. GAIT TEST (OFF)

Equipment required:

Mobile Phone Video y odometer

Procedure:

The patient will walk 15 meters on a flat, bare street. First the camera is placed on a tripod behind the patient; the camera can be on the floor, or on street furniture (bench, trash ...). The patient is placed standing in front of the camera (backwards to the camera) so that his/her feet appear in the image from the beginning to the end of the ride. The interviewer will be placed next to the patient and will set to zero the odometer. Both will walk a minimum distance of 15 meters, straight and then will stop.

1. Start: Hour____ / minute____ (Tablet PC time)
 2. End: Hour____ / minute____ (Tablet PC time)
 3. Distance (shown in odometer at the end of the walk): ____ meters and ____ cm
 4. The patient does not want or cannot perform the test. ☐
 5. Explain why the patient did not perform the test: _____
-

S.6b. Gait test (OFF)

Equipment required:

Odometer and chronometer

Procedure:

The patient will need to walk a minimum of 15 meters in a flat and clear street

The test will start with the patient standing still in a convenient street of their neighbourhood. Then he will start walking straight forward a minimum distance of 15 meters. One of the researchers will operate an odometer to measure the distance during the walk. The other will operate a chronometer to time the walk. Both will count the total number of steps of the walk. After a minimum of 15 meters the patient will stop and stay still.

1. Start: Hour____ / minute____ (Tablet PC time)
 2. End: Hour____ / minute____ (Tablet PC time)
 3. Distance (shown in odometer at the end of the walk): ____ meters and ____ cm
 4. Number of steps _____
 5. Time (chronometer): Minutes____ / Seconds____ / Tenths____
 6. The patient does not want or cannot perform the test. ☐
 7. Explain why the patient did not perform the test: _____
-

S.7. SHUT DOWN THE VIDEO AND SENSORS

1. Take out the sensors from the patient
2. Make visual synchronization while the mobile phone is recording it:
 - a. Put the waist sensor standing on the table
 - b. Leave it still during 10sec,
 - c. Drop it
 - d. Leave it lying over the table during 10 sec
3. Switch the sensors off by pressing the central button in the waist sensor while the mobile is recording
4. Stop video recording and check the integrity of the video (check that the video is listed in the application and its date is correct).
5. Sync phone with Tablet:
 - a. Connect cable and video sync
 - b. Run the App 'Synchronization tablet-smartphone' in the tablet
 - c. Disconnect cable

S.8. START SENSORS AND TABLET APP

1. Start recording data with sensors:
 - a. Turn on the sensors
 - b. Sync the tablet and the sensors with the app 'Switch On Sensors'
 - c. If everything is correct the waist sensor must blink in green and blue and the wrist sensor must blink in blue
2. Launch the monitoring application in the Tablet PC ('Tablet annotations')

S.9. OUTDOORS WALKING TEST IN OFF PHASE

Equipment required:

Tablet PC

Procedure:

The patient will go for a 10 to 15 minute walk in the neighbourhood (he/she can follow the preferred route). The entire patient's movements, the slope of the path, and the ground characteristics will be continuously recorded by the researchers on the tablet PC. Please, press the 'Outdoors walking test' in the tablet annotations application to mark that this activity is being performed.

1. Start: Hour____ / minute____ (Tablet PC time)
2. End: Hour____ / minute____ (Tablet PC time)

S.10. TAP

Start TAP in the mobile phone (app 'REMPARK Prompts').

Equipment required:

Mobile Phone

Procedure:

Mobile phone will be placed on a table and the TAP test will be performed.

Please, press the 'TAP test' in the tablet annotations application to mark that this activity is being performed

S.11. THE PATIENT CAN TAKE THEIR USUAL MEDICATIONS IN ORDER TO RETURN TO THE ON STATE

Depending on the time, the instructions of the neurologist and patient's preferences, the patient can take the medication which was not taken in morning, or he/she can move to the next dose. If the patient prefers to use a rescue by apomorphine injection or other procedure can do it too.

If in doubt call: _____ (phone number)

S.12. FREE ACTIVITY MONITORING DURING OFF PHASE

Equipment required:

Tablet PC

Procedure:

The remaining time in OFF state, the patient will freely choose what they want to do, when and where, until he/she switches to an evident ON state. As patients may change their usual activity due to the observation process, researchers will encourage the patient to perform their usual activity, as recorded in the basal visit. Nevertheless, the patient will be free to change the day activity if they desire.

During the free-activity monitoring period, one researcher will accompany the patient recording in the tablet PC all the movements, postures, terrain conditions, symptoms and drugs doses taken.

Press the "rest of the observer" button if you cannot follow the activity of the patient, if the patient goes to the bathroom or if the observer needs to rest.

The unforeseen postures and activities can be reported as "other" 1,2,3 or "other" a, b, c, respectively.

Whether the patient is an ON, OFF or intermediate state will be reviewed and recorded periodically. The test is over when the patient unequivocally declares to be in the ON state (moves to the next section of the questionnaire).

UNFORESEEN POSTURES: describe the postures that has been assigned to each of the labels "other" ("otros")

Other 1: _____

Other 2: _____

Other 3: _____

UNFORESEEN ACTIVITIES: describe the activity that has been assigned to each of the labels "other" ("otros")

Other a: _____

Other b: _____

Other c: _____

Case Report Form for the free activity monitoring session: describe every relevant event

1. Start: Hour____ / minute____ (Tablet PC time)

Hour____ / minute____ (Tablet PC time)

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

2. End: Hour____ / minute____ (Tablet PC time)

S.13. SHUT DOWN THE SENSORS

1. Take out the sensors from the patient
2. Switch the sensors off by pressing the central button of the waist sensor
3. Close the 'Tablet annotations' app in the Tablet

ON PERIOD

S.14 UPDRS (ON PERIOD)

0.1 In terms of mobility, you are:

- In the optimum state 0
- In an acceptable state 1

Mark the option that corresponds to the patient.

1. Speech

- Normal..... 0
- Slight loss of expression, diction and/or volume..... 1
- Monotone, slurred but understandable; moderately impaired 2
- Marked impairment, difficult to understand 3
- Unintelligible 4

2. Facial Expression

- Normal..... 0
- Minimal hypomimia, could be normal "Poker Face" 1
- Slight but definitely abnormal diminution of facial expression 2
- Moderate hypomimia; lips parted some of the time..... 3
- Masked or fixed facies with severe or complete loss of facial expression; lips parted 1/4 inch or more 4

3. Tremor at rest (lower extremities)

- Absent 0
- Slight and infrequently present..... 1
- Mild in amplitude and persistent. Or moderate in amplitude, but only intermittently present..... 2
- Moderate in amplitude and present most of the time 3
- Marked in amplitude and present most of the time 4

4. Tremor at rest (face, lips, chin)

- Absent 0
- Slight and infrequently present..... 1
- Mild in amplitude and persistent. Or moderate in amplitude, but only intermittently present..... 2
- Moderate in amplitude and present most of the time 3
- Marked in amplitude and present most of the time 4

0.2 UPDRS time:

____h / ____m (Tablet PC time)

5. Tremor at rest (upper extremities)

- Absent 0
- Slight and infrequently present 1
- Mild in amplitude and persistent. Or moderate in amplitude, but only intermittently present..... 2
- Moderate in amplitude and present most of the time 3
- Marked in amplitude and present most of the time 4

6. Action or Postural Tremor of hands

- Absent 0
- Slight; present with action..... 1
- Moderate in amplitude, present with action 2
- Moderate in amplitude with posture holding as well as action 3
- Marked in amplitude; interferes with feeding 4

7. Axial rigidity (Judged on passive movement of the neck with patient relaxed in sitting position. Cogwheeling to be ignored)

- Absent 0
- Slight or detectable only when activated by mirror or other movements 1
- Mild to moderate 2
- Marked, but full range of motion easily achieved 3
- Severe, range of motion achieved with difficulty 4

8. Right upper extremity rigidity (Judged on passive movement of major joints with patient relaxed in sitting position. Cogwheeling to be ignored)

- Absent 0
- Slight or detectable only when activated by mirror or other movements 1
- Mild to moderate 2
- Marked, but full range of motion easily achieved 3
- Severe, range of motion achieved with difficulty 4

9. Left upper extremity rigidity (Judged on passive movement of major joints with patient relaxed in sitting position. Cogwheeling to be ignored)

- Absent 0
- Slight or detectable only when activated by mirror or other movements 1
- Mild to moderate 2
- Marked, but full range of motion easily achieved 3
- Severe, range of motion achieved with difficulty 4

10. Right lower extremity rigidity (Judged on passive movement of major joints with patient relaxed in sitting position. Cogwheeling to be ignored)

- Absent 0
- Slight or detectable only when activated by mirror or other movements 1
- Mild to moderate 2
- Marked, but full range of motion easily achieved 3
- Severe, range of motion achieved with difficulty 4

11. Left lower extremity rigidity (Judged on passive movement of major joints with patient relaxed in sitting position. Cogwheeling to be ignored.)

- Absent 0
- Slight or detectable only when activated by mirror or other movements 1
- Mild to moderate 2
- Marked, but full range of motion easily achieved 3
- Severe, range of motion achieved with difficulty 4

12. Finger Taps (Patient taps thumb with index finger in rapid succession) (Right)

- Normal 0
- Mild slowing and/or reduction in amplitude 1
- Moderately impaired. Definite and early fatiguing. May have occasional arrests in movement 2
- Severely impaired. Frequent hesitation in initiating movements or arrests in ongoing movement 3
- Can barely perform the task 4

13. Finger Taps (Patient taps thumb with index finger in rapid succession) (Left)

- Normal 0
- Mild slowing and/or reduction in amplitude 1
- Moderately impaired. Definite and early fatiguing. May have occasional arrests in movement 2
- Severely impaired. Frequent hesitation in initiating movements or arrests in ongoing movement 3

• Can barely perform the task 4
14. Hand grips (Patient opens and closes hands in rapid succession) (Right)

- Normal 0
- Mild slowing and/or reduction in amplitude 1
- Moderately impaired. Definite and early fatiguing. May have occasional arrests in movement 2
- Severely impaired. Frequent hesitation in initiating movements or arrests in ongoing movement 3
- Can barely perform the task 4

15. Hand grips (Patient opens and closes hands in rapid succession) (Left)

- Normal 0
- Mild slowing and/or reduction in amplitude 1
- Moderately impaired. Definite and early fatiguing. May have occasional arrests in movement 2
- Severely impaired. Frequent hesitation in initiating movements or arrests in ongoing movement 3
- Can barely perform the task 4

16. Rapid Alternating Movements of Hands (Pronation-supination movements of hands, vertically and horizontally, with as large an amplitude as possible, both hands simultaneously) (Right)

- Normal 0
- Mild slowing and/or reduction in amplitude 1
- Moderately impaired. Definite and early fatiguing. May have occasional arrests in movement 2
- Severely impaired. Frequent hesitation in initiating movements or arrests in ongoing movement 3
- Can barely perform the task 4

17. Rapid Alternating Movements of Hands (Pronation-supination movements of hands, vertically and horizontally, with as large an amplitude as possible, both hands simultaneously) (Left)

- Normal 0
- Mild slowing and/or reduction in amplitude 1
- Moderately impaired. Definite and early fatiguing. May have occasional arrests in movement 2
- Severely impaired. Frequent hesitation in initiating movements or arrests in ongoing movement 3
- Can barely perform the task 4

18. Leg Agility (Patient taps heel on the ground in rapid succession picking up entire leg. Amplitude should be at least 3 inches.) (Right)

- Normal..... 0
- Mild slowing and/or reduction in amplitude 1
- Moderately impaired. Definite and early fatiguing. May have occasional arrests in movement 2
- Severely impaired. Frequent hesitation in initiating movements or arrests in ongoing movement..... 3
- Can barely perform the task 4

19. Leg Agility (Patient taps heel on the ground in rapid succession picking up entire leg. Amplitude should be at least 3 inches.) (Left)

- Normal..... 0
- Mild slowing and/or reduction in amplitude 1
- Moderately impaired. Definite and early fatiguing. May have occasional arrests in movement 2
- Severely impaired. Frequent hesitation in initiating movements or arrests in ongoing movement..... 3
- Can barely perform the task 4

20. Arising from Chair (Patient attempts to rise from a straight-backed chair, with arms folded across chest.)

- Normal..... 0
- Slow; or may need more than one attempt 1
- Pushes self up from arms of seat..... 2
- Tends to fall back and may have to try more than one time, but can get up without help 3
- Unable to arise without help 4

21. Posture

- Normal erect 0
- Not quite erect, slightly stooped posture; could be normal for older person 1
- Moderately stooped posture, definitely abnormal; can be slightly leaning to one side 2
- Severely stooped posture with kyphosis; can be moderately leaning to one side 3
- Marked flexion with extreme abnormality of posture 4

22. Gait

- Normal..... 0
- Walks slowly, may shuffle with short steps, but no festination (hastening steps) or propulsion..... 1
- Walks with difficulty, but requires little or no assistance; may have some festination, short steps, or propulsion..... 2
- Severe disturbance of gait, requiring assistance..... 3
- Cannot walk at all, even with assistance..... 4

23. Postural Stability (Response to sudden, strong posterior displacement produced by pull on shoulders while patient erect with eyes open and feet slightly apart. Patient is prepared)

- Normal..... 0
- Retropulsion, but recovers unaided 1
- Absence of postural response; would fall if not caught by examiner 2
- Very unstable, tends to lose balance spontaneously 3
- Unable to stand without assistance 4

24. Body Bradykinesia and Hypokinesia (Combining slowness, hesitancy, decreased armswing, small amplitude, and poverty of movement in general)

- None..... 0
- Minimal slowness, giving movement a deliberate character; could be normal for some persons. Possibly reduced amplitude 1
- Mild degree of slowness and poverty of movement which is definitely abnormal. Alternatively, some reduced amplitude..... 2
- Moderate slowness, poverty or small amplitude of movement..... 3
- Marked slowness, poverty or small amplitude of movement..... 4

S.15. SYNCHRONIZATION

1. Sync phone with Tablet:
 - a. Connect the tablet to the mobile phone by the USB cable
 - b. Run the App 'Synchronization tablet-smartphone' in the tablet
 - c. Disconnect cable
2. Start recording data with sensors:
 - a. Turn on the sensors
 - b. Sync the tablet and the sensors with the app 'Switch On Sensors'
 - c. If everything is correct the waist sensor must blink in green and blue and the wrist sensor must blink in blue
3. Start video recording in the mobile (app 'REMPARK Video'). The name of the video must follow the convention 'PARTNER patient_nº 2' (2 means ON).
4. Make Visual synchronization:
 - a. Put the waist sensor standing on the table
 - b. Leave it still during 10sec,
 - c. Drop it
 - d. Leave it lying over the table during 10 sec
5. Place the sensors to the patient and videotape it

S.16. RECORDING OF THE ON PHASE

Equipment required:
Mobile Phone Video

Procedure:

Patient will be asked to walk at the preferred speed and following the preferred trajectory for 20 seconds. Turns are allowed (no matter if the patient walks indoors or outdoors).

S.17. INDOORS WALKING TEST IN ON PHASE

Equipment required:
Mobile Phone Video

Procedure:

In the indoors walking test the patient will start sitting on a chair in the living room. They will stand up and show their house to the researchers, showing each room and explaining what the room is for, just like if they were trying to sell the house. After the whole visit, the patient will return to the chair in the living room and will sit down again and the test will be over.

1. Start: Hour____ / minute____ (Tablet PC time)
2. End: Hour____ / minute____ (Tablet PC time)

NOTE: If the visit lasts less than two minutes, the test will be performed twice. If the test has lasted more than two minutes, move to the next section.

1. Start: Hour____ / minute____ (Tablet PC time)
2. End: Hour____ / minute____ (Tablet PC time)

S.17a. GAIT TEST (ON)

Equipment required:
Mobile Phone Video y odometer

Procedure:

The patient will walk 15 meters on a flat, bare street. First the camera is placed on a tripod behind the patient; the camera can be on the floor, or on street furniture (bench, trash ...). The patient is placed standing in front of the camera (backwards to the camera) so that his/her feet appear in the image from the beginning to the end of the ride. The interviewer will be placed next to the patient and will set to zero the odometer. Both will walk a minimum distance of 15 meters, straight and then will stop.

1. Start: Hour____ / minute____ (Tablet PC time)
2. End: Hour____ / minute____ (Tablet PC time)
3. Distance (shown in odometer at the end of the walk): ____ meters and ____ cm
4. The patient does not want or cannot perform the test. ☐
5. Explain why the patient did not perform the test:

S.17b. GAIT TEST (ON)

Equipment required:

Odometer and chronometer

Procedure:

The patient will need to walk a minimum of **20 meters** in a flat and clear street
The test will start with the patient standing still in a convenient street of their neighbourhood. The he will start walking straight forward a minimum distance of 20 meters. One of the researchers will operate an odometer to measure the distance during the walk. The other will operate a chronometer to time the walk. Both will count the total number of steps of the walk. After a minimum of 20 meters the patient will stop and stay still.

1. Start: Hour____ / minute____ (Tablet PC time)
2. End: Hour____ / minute____ (Tablet PC time)
3. Distance (shown in odometer at the end of the walk): ____ meters and ____ cm
4. Number of steps _____
5. Time (chronometer): Minutes____ / Seconds____ / Tenths____

S.18. DYSKINESIA TEST (when the patient presents dyskinesia)

Equipment required:

Mobile Phone Video

Procedure:

The dyskinesia test starts with the patient sitting on a chair, then he/she stands up, and stays still for 1 minute, afterwards he/she sits down again and stays at rest for another minute. Then the test is over. The test will be video recorded by a researcher. Start and end timings will be registered.

1. Start: Hour____ / minute____ (Tablet PC time)
2. End: Hour____ / minute____ (Tablet PC time)

3. Did the patient have tremor during this test?

- Yes.....0
- No.....1

4. Where?

- Right leg1
- Left leg2
- Right arm3
- Left arm4
- Head or jaw5

S.19. OFF FALSE POSITIVE TEST & DUAL TASK TEST

Equipment required:

Mobile Phone Video. Glass of water. Text to read.

Procedure:

The test will start with the patient sitting on a chair in the kitchen. The patient will be invited to stand up, and to walk from the kitchen to the furthest room in the house, carrying a full glass of water. The patient will return from the room walking while they read a given text in loud voice. Once they are again in the kitchen, they will sit down again.

La prueba comienza con el paciente sentado en una silla en la cocina. El paciente se levantará y caminará de la cocina a la habitación más alejada de la casa llevando un vaso lleno de agua. El paciente retornará de la habitación leyendo un texto en voz alta. Cuando llegue a la cocina se volverá a sentar en la silla.

1. Start: Hour____ / minute____ (Tablet PC time)
2. End: Hour____ / minute____ (Tablet PC time)

S.20. TREMOR FALSE POSITIVE TEST

Equipment required:

Mobile Phone Video.

Procedure:

The patient will perform the following activities

- Brushing teeth (with both hands)
- Shake a deodorant (with both hands)
- Erase with a rubber (with both hands)
- Type on the computer
- Clean the window glass or the furniture (with both hands)
- Draying a glass

1. Start: Hour____ / minute____ (Tablet PC time)
2. End: Hour____ / minute____ (Tablet PC time)

S.21. SHUT DOWN THE VIDEO AND SENSORS

1. Make Visual synchronization
 - a. Put the waist sensor standing on the table
 - b. Leave it still during 10sec,
 - c. Drop it
 - d. Leave it lying over the table during 10 sec
2. Switch the sensors off by pressing the central button in the waist sensor while the mobile is recording
3. Stop video recording and check the integrity of the video (check that the video is listed in the application and its date is correct).
4. Sync phone with Tablet:
 - a. Connect cable and video sync
 - b. Run the App 'Synchronization tablet-smartphone' in the tablet
 - c. Disconnect cable

S.22. START SENSORS AND TABLET APP

1. Start recording data with sensors:
 - a. Turn on the sensors
 - b. Sync the tablet and the sensors with the app 'Switch On Sensors'
 - c. If everything is correct the waist sensor must blink in green and blue and the wrist sensor must blink in blue
2. Launch the monitoring application in the Tablet PC ('Tablet annotations')

S.23. OUTDOORS WALKING TEST IN ON PHASE

Equipment required:

Tablet PC

Procedure:

The patient will go for a 10 to 15 minute walk in the neighbourhood (he/she can follow the preferred route). The entire patient's movements, the slope of the path, and the ground characteristics will be continuously recorded by the researchers on the tablet PC.

Please, press the 'Outdoors walking test' in the tablet annotations application to mark that this activity is being performed

1. Start: Hour____ / minute____ (Tablet PC time)
2. End: Hour____ / minute____ (Tablet PC time)

S.24. TAP

Equipment required:

Mobile Phone

Procedure:

Mobile phone will be placed on a table and the TAP test will be performed.

Please, press the 'TAP test' in the tablet annotations application to mark that this activity is being performed

S.25. FREE ACTIVITY MONITORING DURING ON PHASE

Equipment required:

Tablet PC

Procedure:

The remaining time of the experimental session (which last about 6 hours since the first contact with the patient in the morning), the patient will freely choose what they want to do, when and where. As patients may change their usual activity due to the observation process, researchers will encourage the patient to perform their usual activity, as recorded in the basal visit. Nevertheless, the patient will be free to change the day activity if they desire.

During the free-activity monitoring period, one researcher will accompany the patient recording in the tablet PC all the movements, postures, terrain conditions, symptoms and drugs doses taken.

Press the "rest of the observer" button if you cannot follow the activity of the patient, if the patient goes to the bathroom or if the observer needs to rest.

The unforeseen postures and activities can be reported as "other" 1,2,3 or "other" a, b, c, respectively.

Whether the patient is an ON, OFF or intermediate state will be reviewed and recorded periodically.

Do not forget to perform dyskinesia test at least once when the patient presents with dyskinesia (next section of the questionnaire)

UNFORESEEN POSTURES: describe the postures that has been assigned to each of the labels "other"
("otros")

Other 1: _____
Other 2: _____
Other 3: _____

UNFORESEEN ACTIVITIES: describe the activity that has been assigned to each of the labels "other"
("otros")

Other a: _____
Other b: _____
Other c: _____

Case Report Form for the free activity monitoring session: describe every relevant event

1. Start: Hour_____ / minute_____ (Tablet PC time)

Hour_____ / minute_____ (Tablet PC time)

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour_____ / minute_____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

Hour____ / minute____

2. End: Hour____ / minute____

(Tablet PC time)

S.26. SHUT DOWN THE SENSORS

1. Switch the sensors off by pressing the central button in the waist sensor.
2. Close the 'Tablet annotations' app in the Tablet

S.27. USABILITY

S27.1 The system usability scale (SUS)

	Strongly disagree							Strongly agree
1. I think that I would like to use this system frequently	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
	1	2	3	4	5			
2. I found the system unnecessarily complex	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
	1	2	3	4	5			
3. I thought the system was easy to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
	1	2	3	4	5			
4. I think that I would need the support of a technical person to be able to use this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
	1	2	3	4	5			
5. I found the various functions in this system were well integrated	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
	1	2	3	4	5			
6. I thought there was too much inconsistency in this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
	1	2	3	4	5			
7. I would imagine that most people would learn to use this system very quickly	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
	1	2	3	4	5			
8. I found the system very cumbersome to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
	1	2	3	4	5			
9. I felt very confident using the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
	1	2	3	4	5			
10. I needed to learn a lot of things before I could get going with this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
	1	2	3	4	5			

S27.2 The Quebec User Evaluation of Satisfaction with Assistive Technology

1	2	3	4	5
Not satisfied at all	Not very satisfied	More or less satisfied	Quite satisfied	Very satisfied

How satisfied are you with,

1. the dimensions (size, height, length, width) of your assistive device? Comments:	1 2 3 4 5
2. the weight of your assistive device? Comments:	1 2 3 4 5
3. the ease in adjusting (fixing, fastening) the parts of your assistive device? Comments:	1 2 3 4 5
4. how safe and secure your assistive device is? Comments:	1 2 3 4 5
5. the ease in using your assistive device? Comments:	1 2 3 4 5
6. the comfort of your assistive device? Comments:	1 2 3 4 5
7. What is your overall satisfaction with the assistive device? Comments:	1 2 3 4 5