MASTER EN SCIENCES INFORMATIQUES

Detection of neuropsychiatric disorders based on motion capture and machine

Von Haus, Steven

Award date:
2015

Awarding institution:
Universite de Namur

Link to publication

University of Namur
Faculty of Computer Science
Academic Year 2014–2015

Detection of neuropsychiatric disorders based on motion capture and machine learning

Steven VON HAUS

Internship mentor: Martin DESSEILLES

Supervisor: ______________________ (Signed for Release Approval - Study Rules art. 40)
Vincent ENGLEBERT

A thesis submitted in the partial fulfillment of the requirements
for the degree of Master of Computer Science at the University of Namur

Abstract

Résumé

Ces dernières années, la capture de mouvement s'est popularisée dans de nombreux domaines et s'est même installée dans nos salons, avec des capteurs comme la Kinect. Dans le secteur médical, les systèmes informatisés fournissant une assistance pour le diagnostic sont en pleine émergence tout comme les applications capables de détecter des comportements anormaux. Le principal objectif de ce mémoire est d'étudier la faisabilité de détecter un trouble dépressif mais également d'autres troubles neuropsychiatriques, à l'aide de captures de mouvement et d'algorithmes d'apprentissage automatique, dans le but de diagnostiquer, prévenir et intervenir. À cette fin, de larges ensembles de données ont été récoltés parmi différents sujets en utilisant la deuxième version de la Kinect. Ces données reprennent la position et l'orientation des articulations du corps, des mesures dans le temps mais également d'autres variables significatives. Par la suite, ces données seront envoyées à un algorithme d'apprentissage automatique, dans le but de générer des forêts d'arbres décisionnels capables de détecter des troubles neuropsychiatriques avec de nouveaux sujets qui n'ont jamais été observés auparavant.

Mots clés : Trouble Neuropsychiatrique, Dépression, Aide au Diagnostic, Capture de Mouvement, Kinect, Suivi du Squelette, Apprentissage automatique, Forêt d'Arbres Décisionnels.

Abstract

In the last few years, motion capture has been popularized in many areas, and has even appeared in the living rooms, with sensors as the Kinect. In the medical sector, computer-based systems providing diagnostic support are emerging, but also applications able to detect abnormal behaviors. The main objective of this master thesis is to study the feasibility of detecting signs of depression or other neuropsychiatric disorders by using motion capture and learning algorithms, in order to diagnose, prevent and intervene. To this end, large data sets of movements among different subjects have been collected by using the second version of the Kinect sensor. These data include the position and the orientation of the joints of the body, measurements in time, and other meaningful variables. Then, this collection of data is sent to a machine learning algorithm in order to generate random forests able to detect neuropsychiatric disorders with new subjects who have never been seen before.

Keywords: Neuropsychiatric Disorder, Depressive Disorders, Diagnostic Aid, Motion Capture, Kinect, Skeleton Tracking, Machine learning, Random Forest.

Foreword

I would first of all like to thank the Professor M. Desseilles, my internship mentor who is at the origin of the idea of this master thesis, for his regular monitoring and precious advices. Thanks to him for his great contribution.

I would also like to thank the Professor V. Englebert, my supervisor, who has accepted to cooperate in the realization of this project. Thanks to him for his careful reading and his useful remarks and advices concerning the writing of this master thesis.

Thanks to the Professor B. Frénay, who helped us for the machine learning part of this work. We thank him for his useful advices.

I would like to thank to the Professor C. Pal, who has accepted to take me in his laboratory, and who has provided useful tools and papers.

Writing in English was a real challenge, especially in the beginning. Indeed, it was the first time that I wrote such a large document. I would like to thank Mister F. Brogniez, who has accepted to read carefully this thesis.

I would also want to thank all my family and friends for all their supports during my studies, especially my parents, who gives me the opportunity to realize this cursus. I also thank particularly G. Gruson, for her unfailing support.

Contents

List of Figures

Introduction

Computer-based systems for psychological assessments brought real beneficial assets. Such a system allows to score the subjects and to control the assessments in a completely normalized way, regardless of patients, specialists and even institutions. Similarly, the stimuli can be presented in a standardized manner. These stimuli can also be randomized in order to easily create new assessments. Moreover, a computer system can collect accurate spatial-temporal data in large quantities and explore them with recognized disciplines as motion capture, database engineering and machine learning.

The main objective of this work is to realize a system able to detect signs of depression or other signs of neuropsychiatric disorders by using motion capture and learning algorithms, in order to diagnose, prevent and intervene. So, this system aims to recognize specific neuropsychiatric disorders for a patient. In addition, the condition of the patients could be monitored over time in order to indicate improvements due to a treatment or simply degradations. Furthermore, domain knowledge could be acquired by recognizing similarities in the data. For the purpose of this work, short predefined tasks will be used for classification. For example, we aim to discriminate tasks which have been executed by depressive subjects and healthy subjects. In the future, a more global classification of the patients could be managed by using the results given by the tasks.

Such a system may seem like a complete utopia. However, spectacular progress in motion capture, and more generally in computer vision, have been made in recent years. In addition, machine learning, a subfield of artificial intelligence, provides algorithms that can automatically learn to recognize similarities in large data sets in order to take actions such as classification. As a result, these two related disciplines give some impressive results like a system which aims to evaluate the effects of a treatment for Parkinson by analyzing the gait and the movements. Other systems attempt to prevent falls from people in hospital by monitoring their gait and their changes in posture. Even more incredible, the self-driving car has been created thanks to computer vision and machine learning.

We will begin this thesis by analyzing the motor impairments and the cognitive impairments caused by neuropsychiatric disorders in Chapter 1. In order to detect and diagnose signs of neuropsychiatric disorders with motion capture, we need to ensure that these disorders have measurable effects on the motor and the cognitive components of a patient. The other variables having similar effects as the treatments will also be analyzed. As the main objective is to discriminate patients among different disorders, we will also describe all the differences in psychomotor retardation. In addition, we aim to indicate our contribution. So, we will discussed about the beneficial aspects of using a computer-based system to collect, but also to explore data. Moreover, some existing methods to measure psychomotor retardation will also be introduced.

A large significant amounts of data will be collected during this work. Naturally, millions of records cannot be manually explored. However, in machine learning, it exists algorithms that are able to learn automatically to take actions by exploring large data sets. Therefore, the next chapter will be devoted to machine learning. We aim to introduce what machine learning is. We will also describe the decision tree and the random forest algorithms which will be used to classify the tasks. Then, some methods evaluating the quality of a model will be introduced in order to measure the accuracy of generated models, but also to improve them.

Chapter 3 aims to describe the motion capture process. We will begin by introducing some examples of applications using motion capture in the medical area. Then, we will present the second version of the Kinect used for motion capture. Moreover, we will explain how the Kinect works. Then, we will talk about the different data that can be obtained from the different sources. So, we will focus on the skeleton and the face. The accuracy and the relevance of data that can be obtained during the process will be criticized.

According to Chapter 1 and Chapter 3, we will define all the data which will be captured for a patient. A set of important neuropsychiatric features will be recorded as the neuropsychiatric diseases which already have been diagnosed. Then, we will specify all the data captured by the Kinect during the motion capture process. We will also present two different methods to recognize a gesture in order to a get a normalized scoring and a normalized presentation of a stimulus.

Chapter 5 aims to introduce the prototype which was built to collect all the data. In this context, all the short predefined tasks created for collecting the data will be introduced. Then, all the different approaches for evaluating the prototype will be elicited as a lot of testing is required before making evaluations in a real patient population. So, we explained how we recorded 14 subjects from a general population before and after alcohol consumption. Moreover, different simulations of impairments have been executed in order to evaluate the ability of the models to recognize similarities in the captured data.

Once the data collected, Chapter 6 will specify how short predefined tasks will be used to detect psychomotor and cognitive impairments. Moreover, we aim to indicate how the features have been computed from large data sets in order to be used in a learning algorithm. Then, we will analyze and select the more valuable features according to the collected data.

Chapter 7 will explain how the random forest implementation of Weka, a popular suite of machine learning tools, will be used in order to generate models able to recognize a task among different groups from the collected data and the best features selected. Next, we will define the different meta-parameters that can be set up to improve the learning phase. Then, we will describe how the different forests were built for each task. All the generated forests will be evaluated and criticized.

Finally, we will finish up this thesis by providing several options for future work. We will discuss the classification of the patients according to the set of tasks that they have executed. In addition, we would like to provide guidance for the use of other sensors than the Kinect.

Part I

State of art

Chapter 1

Psychomotor retardation

Since we are trying to detect signs of neuropsychiatric disorders by using motion capture, it is necessary to examine if it is possible to diagnose a patient by just observing his movements. Indeed, if neuropsychiatric diseases have no effect on the motor components of a patient, it would be impossible to find a correlation between the captured movements and a neuropsychiatric disorder. In addition, one of the main objectives of the system is to discriminate different neuropsychiatric disorders. So, the goal of this section is to establish a state of the art from scientific articles which had analyzed the impacts of neuropsychiatric disorders and their effects on the motor system in order to prove that building such a system is possible. In addition, the motor components is affected by a considerable amount of variables which will be examined. Finally, we aim to indicate our contribution by analyzing what would be the added value of this work.

1.1 Influences of neuropsychiatric disorders on the motor system

According to psychiatric authors [14] [23] [53], even the earliest, **psychomotor retardation**, which includes motor and cognitive impairments, is one of the most fundamental features of major depressive disorder (MDD). Psychomotor retardation, which is also known as psychomotor impairment involves decreased movements, cognitive impairments and disturbances in speech [21]. For example, daily tasks which would normally require little thought and effort, become difficult to perform as simple household chores. The **motor impairments** [45] refers to the partial or the total loss of control, of a body part, during movements. For example, movements as standing, sitting and walking. Moreover, they also include the reflex movements or the gestural expressions. While, the **cognitive impairments** [4] encompass all the obstacles to the thought processes including the attention, the memory and the computation.

More generally, **psychomotor disturbance**, which includes psychomotor retardation, is a classic feature for diagnosing major depressive disorder. Unfortunately, according to Sobin and Sackeim [66], there is no agreement on the definition of psychomotor disturbances. Therefore, this term is rather ambiguous and refers to both psychomotor retardation and psychomotor agitation which occur during a depressed state. **Psychomotor agitation** [50][10] refers to a restless state or an increase of muscular activities, associated to mental tensions occurring with depression or anxiety. It results in involuntary movements as tapping fingers or shaking legs. However, retardation and agitation are often inseparable as most studies are measuring psychomotor disturbances without distinctions [66].

Bipolar disorder [65][18] are characterized by serious shifts between manic states and depressed states with normal states in between. Moreover, mood changes are really intense and can last for days. The psychomotor symptoms in general are common in bipolar disorder. So, both agitation and retardation [66] indicators are associated with bipolar disorder. The depressed bipolar patients differ from the healthy group, the unipolar group and the schizophrenic group in activity levels. The bipolar patients show low activity levels during depressed states with an increased level in manic phases. So in a 24-hour activity, bipolar patients can be discriminated from the other groups. They also differ from schizophrenic patients with more frequent and smaller movements of the head, shorter eye contact with the stimuli and they also tend to touch most frequently their body. There are also a lot of other differences. However, the main objective of this chapter is to prove that a discrimination among the neuropsychiatric groups is possible and not to list all the differences.

Melancholic depression is systematically the most described subtype of major depressive disorder. Melancholic patients [32] suffer from cognitive impairments which results in poor concentration, but also in retardation and agitation affecting body parts, the speech and the facial expressions. The melancholic patients [32] are also characterized by a lack of pleasure and energy in any activities. Nevertheless, many authors, as Pier in [49] or Rush and Weissenburger in [57], but also Sobin and Sackeim in [66], assert that "the presence of clinical psychomotor retardation would be sufficient to give a diagnostic among depressive subtypes or at least, it would be one of the strongest indicators of the melancholic subtype". Indeed, melancholic depression is considered as divergent to non-melancholic depression in terms of cognitive and motor impairments. Moreover, findings suggest that unlike non-melancholic patients, melancholic patients exhibit motor slowing [48] [54] [55] and particular difficulties initiating movements in the absence of external indications.

Schizophrenia [6] is a disorder defined by an altered perception of the reality. For example, some schizophrenic patients become paranoid and can even believe that others are conspiring against them. Moreover, some of them, see or hear things that does not exist. So, the symptoms are not always similar. Such disorders act as barriers in daily activities, but also in work. Schizophrenia and depression have both a slowing in motor and mental activities denoted as "psychomotor poverty" in schizophrenia and as "psychomotor retardation" in depression. Van Hoof has proved [70] that it is possible to observe a slowing in both disorders. But most of all, a different pattern of slowing could be identified which would allow to classify patients between the two groups. To achieve this outcome, different measurements have been realized during the performance a *Digit Symbol Test* (DST) [70]. So, the psychomotor speed, the rate of good answers, but also the computerized analysis of writing movements recorded have been analyzed. An example of DST can be seen in Figure 1.1, the subject receives a table where the digits are replaced by symbols. For example, a "1" becomes a "#" and a "3" becomes a "!". Then the subject must translate correctly, as fast as possible, a sequence of digits by the corresponding symbols given.

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
#	&	!	{	$	%	=	€	/

9	**2**	**3**	**8**	**5**	**2**	**8**	**7**	**1**

Figure 1.1: Example of a digit symbol test.

A lot of similarities were reported in the studies of Schrijvers [61] who has compared major depressive disorder (MDD) and chronic fatigue syndrome (CFS). But some differences between the both disorders have also been denoted. To clarify the both terms, **major depressive disorder** [1], also called "unipolar disorder" differs from bipolar disorder with only one pole which is the depressed mood. There is no manic pole as in bipolar disorder. **Chronic fatigue syndrome** [2] is characterized by a chronicle fatigue, which does not improve with rest. CSS patients also suffer from other symptoms as cognitive impairments, inattention and a slowing in motor activity. This fatigue state is not provoked by exercising or by a medical treatment and interfere in daily activities, but also in work. Analyses [61] have been led by digitally recording output with two copying tasks: complex figures as a wave pattern or a diamond and simple shapes as a circle or a line. As a result, both the MDD and CFS patients showed motor slowing. Moreover, the motor component in the MDD patients was more affected than in the CFS patients. Nevertheless, for simple shapes, MDD patients needed substantially more time than the CFS patients to reproduce a single line. While, for complex figures, both patient groups, but also the healthy group were slower even if the effects were less pronounced in the healthy group. However, the motor processes involved in simple drawing movements were more severely disturbed in MDD than in CFS patients. It should be noted that the results showed an absence of differences in cognitive performance. In short, both groups demonstrated a motor slowing, which was more pronounced in the MDD group.

1.2 Influences of treatments on the motor system

Now that we have seen the influences of some neuropsychological pathologies on the motor system. We found interesting to explore the influences in case of treatments. Indeed, as already mentioned, psychomotor disturbance may have many different causes. So, we must examine if the impact on the motor system is not just a side effect caused by medication or other treatments. But most of all, we are interested in analyzing clinical improvements for patients.

Volkers analysed in [71], the clinical effects of imipramine and fluvoxamine for depressed inpatients. The study shows that patients treated with imipramine demonstrated an increase in motor activity during the awake state while showing more fragmented motor activity during sleep. However, patients treated with fluvoxamine demonstrated no variations in motor pattern activity during the evaluation.

The use of other antidepressants such as **venlafaxine**, **mirtazapine**, **reboxetine** and **moclobemide** in the treatments of MDD has also increased since the last decade [62]. Different effects of all these types of antidepressants on psychomotor performance have been revealed. But, for most antidepressants, beneficial psychomotor effects appear only in the long term. All these types of drugs have different curves of progress. So, the prediction of psychomotor symptoms relies on the type of disturbance: retardation or agitation. But also, on the type of antidepressant [62].

1.3 Influences of gender

Clinical studies about the influences of the gender on psychomotor performances and cognitive impairments have provided divergent results. Indeed, several authors [30] did not find any impact of the gender on the motor system in case of psychomotor retardation while other studies [39] reported that psychomotor retardation is more pronounced in women than men. But it may be due to sample sizes, differences in methodologies and other variables as the background of a person. However, as there is no consensus on the influence of the gender, it is always recommended to consider gender as a variable to analyse.

1.4 Influences of age

Psychomotor disturbances in 70 year old depressed patients may be different from middle-aged patients of 40 years old because aging already causes slowing. In [37], they show that the age and the depression can be related in the case of a depression which has lasted some years, which results in a more pronounced psychomotor retardation. In his empirical study, Barkley [12] confirms this hypothesis by stating that the age and the gender are significantly related to the activity level. Thereby, this information should always be recorded.

1.5 Influences of alcohol consumption

We also investigated the effects of the alcohol on the cognitive impairments for two main reasons. Firstly, alcohol is the source of many movement disorders as tics, tremor and parkinsonism [29]. Secondly, before using a prototype in groups of patients, it is definitely necessary to make sure that the prototype is really capable of detecting differences in terms of motor impairments between different groups. This proof of concept can be realized by simple simulations. For examples, slow movements and quick movements can be executed and then we will observe if the system is able to discriminate the two groups. But for testing in more rational situations, another solution was necessary. One of this solution, is the cognitive impairments caused by an alcohol consumption. We will discussed about the ethical aspects of the data collection, using volunteers in Chapter 5.

Obviously, alcohol has many effects on the motor system in terms of slowing, attention, but also in reaction time. For this reason, alcohol is one of the largest causes of accidents. Williamson [72] argued that a 0.5% *blood alcohol concentration* (BAC) introduces a slowing of 10% in reaction time tests with a score of 534 ms while the score was 486 ms without alcohol. The **reaction time** [34] is the elapsed time between the presentation of a stimuli and the behavioral response which is typically a button to press. So, the reaction time is an indicator of efficiency used to measure the speed of processing. Only two glasses of standard wine (12 % vol. and 144 ml) are enough to reach a BAC of 0.5%. Moreover the number of misses in the test goes from 0.36 to 1.17. The results become rapidly worse with a BAC of 1%. The reaction time and the number of errors are equal to 566 ms and 2.81 respectively. Williamson also tested the subjects with the symbol digit test, represented in Figure 1.1. With a 0% BAC, they took 2233 ms to perform the test with an accuracy of

99% while the same subjects with 0.5% BAC show respectively 2415 ms and 97.83%. With 1% BAC they exhibited a time of 2656 ms and an accuracy of 94.52%. So, the performance in time has decreased until 16 % and the accuracy has declined of 5% with only 1% BAC. Only reasonable quantity of alcohol have been tested, but the performance was already decreasing really rapidly with the increase of the BAC.

But very surprisingly, Hess [29] argued that alcohol with correct doses can also have beneficial effects for improving tremor. Indeed, some subjects in [29] has demonstrated improvements with little doses. However, Hess also considers such methods as not viable as they are generally uncontrolled. Moreover, a repeated intake demands more and more doses for the same effect and can create an addiction while causing other problems. So, alcohol has definitely an impact on the cognitive process and on the motor component. In this context, alcohol is an interesting feature to take into account.

1.6 Influences of sleep deprivation

Another solution for testing the detection of motor impairments, in real situations, is the sleep deprivation. Williamson also [72] compared the effects of alcohol and the sleep deprivation with the same subjects during a different day. After 18 hours of sleep deprivation, the results of the tests are very similar than a subject with a 0.5% BAC. The effects are rather constant during the first 18 hours of sleep deprivation. However the performances decrease exponentially with 22 hours of sleep deprivation and more hours will certainly make the situation worse. So after 22 hours, patients show approximately the same result than a 1% BAC. The subjects accomplished the reaction time test with a score of 554 ms and a number of errors equals to 3.10. The results in the symbol digit test was not much famous with a time of 2577 ms to perform the test and an accuracy of 97.41%. In this way, the sleep deprivation can also be an effective method.

1.7 Influences of other variables

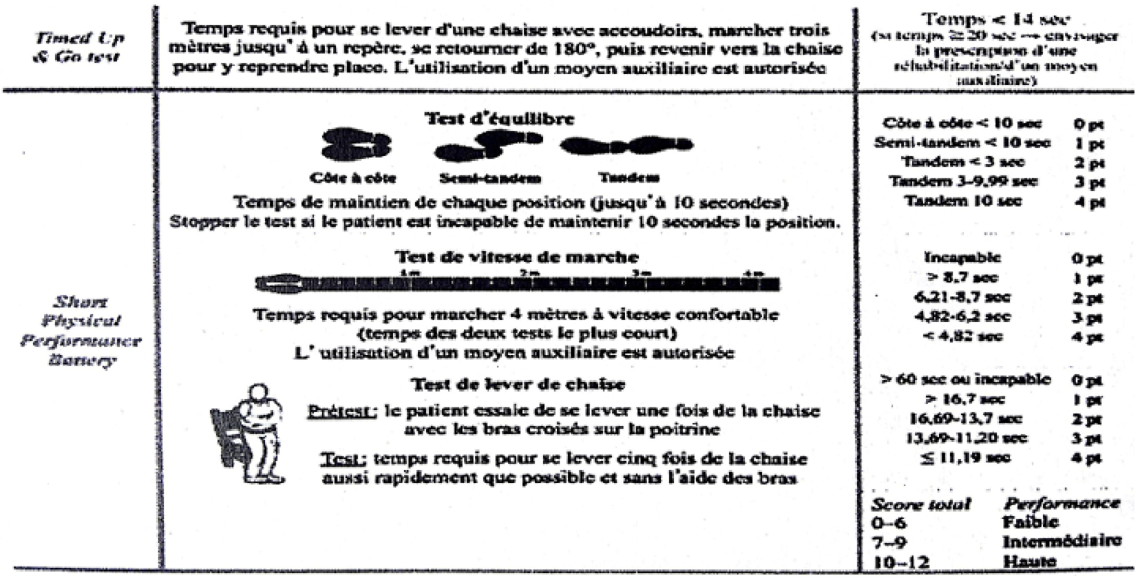
The *Body Mass Index* (BMI) could be an important variable. Indeed, Chirico [22] claims that “reports of physical activity agree that obese persons are less active than persons of normal weight”. So, the BMI is a feature to record even if a more relevant measure would be the percentage of body fat. However, this variable can be more difficult to acquire.

In reality, there are countless variables which could engender cognitive and motor impairments. Indeed, some substance abuses other than alcohol will affect the motor activity as sleep disorder, but also eating disorder or any illness in general. Obviously, they cannot all be considered. However, the main objective in this work, is to focus on the diseases and the medications already discussed.

1.8 Methods used to measure effect on the motor system

In addition to the symbol digit tests already mentioned, many methods exist to measure the psychomotor disturbances as the drawing tasks. In the study of Sabbe [58], a special design pen, a graphics tablet and a computer were used to record the kinematics of drawing movements in depressive groups. The detailed analysis of the data gathered allowed to determine precisely the patterns of the motor slowing. Moreover, the study highlights a severe limitation in the speed of execution, but also some difficulties to initiate a movement during the first part of the tasks without external cues. So, it would be interesting to sample the data and focus on the first part of the movement and then compare it to the others [58].

In [27], the authors have employed the daily interactions with a keyboard as a means to quantify psychomotor impairments. Many pattern recognition algorithms have been proposed to certify the identity of a person typing. These algorithms, also known as keystroke algorithm, can achieve an excellent accuracy where the identification rate can be higher than 95 %. Most of these algorithms are mostly specialized for biometric identification and are not built up to identify health related variations. Indeed, they are especially adapted for recognizing typing patterns with known pre-defined text. All kinds of algorithms have been used for this purpose: Bayesian analysis, hidden Markov models, artificial neural networks. While physical or psychological variations can weaken the accuracy for biometric applications, these features have been used to infer the psychomotor status of the person. Significant changes have been detected between a group of subject rested and the same group with the condition of sleep inertia.



Timed Up & Go test	Temps requis pour se lever d'une chaise avec accoudoirs, marcher trois mètres jusqu' à un repère, se retourner de 180°, puis revenir vers la chaise pour y reprendre place. L'utilisation d'un moyen auxiliaire est autorisée	Temps < 14 sec (si temps ≥ 20 sec → envisager la prescription d'une réhabilitation/d'un moyen auxiliaire)	
Short Physical Performance Battery	**Test d'équilibre** Côte à côte Semi-tandem Tandem Temps de maintien de chaque position (jusqu'à 10 secondes) Stopper le test si le patient est incapable de maintenir 10 secondes la position.	Côte à côte < 10 sec Semi-tandem < 10 sec Tandem < 3 sec Tandem 3-9,99 sec Tandem 10 sec	0 pt 1 pt 2 pt 3 pt 4 pt
	Test de vitesse de marche Temps requis pour marcher 4 mètres à vitesse confortable (temps des deux tests le plus court) L' utilisation d'un moyen auxiliaire est autorisée	Incapable > 8,7 sec 6,21-8,7 sec 4,82-6,2 sec < 4,82 sec	0 pt 1 pt 2 pt 3 pt 4 pt
	Test de lever de chaise Prétest: le patient essaie de se lever une fois de la chaise avec les bras croisés sur la poitrine Test: temps requis pour se lever cinq fois de la chaise aussi rapidement que possible et sans l'aide des bras	> 60 sec ou incapable > 16,7 sec 16,69-13,7 sec 13,69-11,20 sec ≤ 11,19 sec	0 pt 1 pt 2 pt 3 pt 4 pt
		Score total 0–6 7–9 10–12	*Performance* Faible Intermédiaire Haute

Figure 1.2: An example of test evaluating the gait, the balance and the changes in posture [3].

Figure 1.2 shows short predefined tasks that are used by specialists in order to evaluate the gait, the balance and the changes in posture. This test is used at CHU Dinant Godinne UCL Namur but also in other hospitals. The first task consists in maintaining balance during 10 seconds. The patient get the highest score if he is able to maintain balance during 10 seconds with his foot in front of the other. The score decreases if he stops before 10 seconds or if his feet are adjacent. During the second task, the patient must walk 4 meters. The more faster he walks and the more the score will be high. For the last task, the patient must get up and sit down 5 times in a row as quick as possible without using the arms. For example, in [67], they used the gait, the balance and the changes in posture during motion capture to detect impairments in order to evaluate the efficiency of a treatment for Parkinson. We will return at this example, in the Chapter 3.1 devoted to motion capture.

1.9 The added value of the project

As seen in this chapter, all these studies claim that it is possible to diagnose a neuropsychiatric disorder by analyzing the motor components of a patient and so just by observing the movements of a person through time. As a reminder, there are differences, in terms of psychomotor retardation and cognitive impairments, that can be measured. Actually, the challenge is to build a system capable of reproducing the statements made by these specialists. So, the system aims to distinguish an healthy person from a person suffering from neuropsychiatric disorders. More precisely, the algorithm intents to discriminate between the different classes of disorders.

Furthermore, as already discussed, the medications have an impact on the motor system. The antidepressants in general affect the psychomotor performances. It would be really interesting to measure the efficiency of medications by comparing the results of a patient through time. For example, it would be possible to measure if the reaction time has changed after a treatment of two months. More generally, it would allow a specialist to follow the efficiency of the treatment, but also to make sure that everything is going as expected.

Some methods measuring psychomotor disturbances already exist such as the drawing tasks or the symbol digit test. At first glance, a computerized system seems not necessary. However, during a task, it would be a high bias, if the measurements were done manually with a stopwatch, started and ended by a specialist. Indeed, the measurements can vary a lot for a same patient between different doctors and even for a same doctor. For example, if the specialist is exhausted, he can be slower of hundreds milliseconds to stop the stopwatch whereas little differences can be very important to measure the reaction time. However, a lot of studies are still performed with papers and stopwatches [74]. Stuart studies in [74] the beneficial effects of using computer-based neuropsychological assessment. A computer-based system allows a standardized presentation of a stimulus. So, the presentation remains the same for every subject and every specialist. In addition, the stimuli can be randomized to prevent from routine. For instance, the order of the tasks could be changed. The scoring is also standardized and controlled. It is particularly important when the evaluations are done by different specialists. Moreover, a computer is able to compute precisely measures in time, but also the speed or the acceleration during a movement. These measures are precisely quantified and can then be compared mathematically. Therefore, the added value of this project is to build a system, invariant to the noise. In other words, for each patient and for each specialist, we aim to acquire data as much as precise and invariant as possible. In this way, it is possible to have comparable results.

For example, the balance test shown in Figure 1.2 uses very precise measures in time in order to give notes to the subject. The more the patient has notes and the more he is considered vigorous. The task consisting in walking 4 meters is until the subject cross a line is using intervals from 6.21 to 8.7 seconds to give 2 points or from 4.82 to 6.2 seconds to grant 3 points. However, it is clear that men cannot measure in milliseconds precisely because of the reaction time. Jensen [35] leads a study on reaction time and movement time with 39 girls of 14.7 years old on average. He measures an average reaction time varying from 0.310 to 0.350 seconds, which represents approximately one quarter of the 4.82-6.2 interval. Yet, we have seen in this chapter that this average increases with age, fatigue and other variables.

Furthermore, with a computer-based system, the data can be captured in large quantities that cannot be directly analyzed manually. Even if this was the case, it would be impossible to process all these data in a reasonable time to get accurate results. For example, the Kinect can capture hundreds of thousands of information per second from a single body. It's obvious that all these data can only be explored by a computer. In computer sciences, there are many mature disciplines such as database engineering, statistics, data mining and machine learning which are perfectly able to process that amount of information.

Lately, medical applications using computer vision and machine learning are emerging. We will introduce some examples in the next chapter. For instance, the applications to prevent falls from a patient either an elderly person or a person suffering, are the most widespread. However, the applications diagnosing neuropsychiatric diseases from motion capture are not singular except for the Parkinson disease. They will be introduced in Chapter 3.1 related to motion capture. There are many methods that are used to evaluate the patients, but they are not always computerized. So, the project aims to bring an innovation to the crossroads between computer science and neuropsychiatry.

In the first place, the system would be a diagnostic aid, available for the specialists, which would allow them to follow their patients through time and to record all the tasks in the same location. The information acts as an indicator to bring a reflection among the specialists. For example, if the expert believed that the patient is healthy while the system says the opposite, it would be interesting to check if the program is right or not. So, the specialist who is not sure, may have to perform further tests.

Finally the dream would be to have such a system at home, which would allow anyone to perform a test anywhere and at any time. The goal is not to replace the specialists, but to allow the patients to monitor its condition on a regular basis. In this way, the data will be gathered more easily, through time, in a larger quantity. Naturally, it is not advisable that the person relies on the results produced by a computer. Instead, this larger amount of data will be interpreted by a specialist. This analysis will be a great complement to his own observations. So, the main objective, would be to encourage the patient to consult a specialist in case of suspicion of a disorder. Moreover, the environment will be different from others as almost all studies that have measured activity have been on inpatients [12], what bring a bias that should not be disregarded. Indeed, the patients were evaluated in hospitals or in clinics but rarely in their natural environments.

Chapter 2

Machine Learning

During motion capture a large amount of data will be stored. Generally, in such systems thousands of information can be recorded in only one second. These data cannot manually be explored. So, machine learning is a subfield of computer sciences frequently used to give diagnostics but also to acquire knowledge in the medical area. More generally, machine learning is used to discover similarities in large amount of data in order to take actions such as classification [15]. So, this chapter will describe what machine learning is. Some methods to evaluate the quality of a model will also be introduced. Moreover, random forests and decision trees will be used in this work for detecting cognitive and motor impairments. For recognizing the gestures, random forests and AdaBoost models will be used. For this chapter we were inspired by the course of Machine Learning from Coursera provided by the Professor Andrew Ng [44] but must of all by the course of the Professor Benoît Frénay [26] at the University of Namur.

2.1 Introduction to machine learning

The big artificial intelligence dream of someday building truly intelligent machines [44], has been one great motivation for machine learning. The brain can learn to recognize digits, pictures, musics, but also all sorts of amazing things. However, these tasks could seem difficult for a computer using a preprogrammed algorithm. For example, the task of recognizing hand-written digits can seem difficult to implement with a preprogrammed algorithm because the number of possible ways to write a digit by hand is infinite as little variations will be introduced. In this context, an algorithm able to identify all the possible digits does not exist. However, such a task seems really easy for men.

Therefore, an algorithm able to learn with generalization like the brain is required to solve this problem. A remarkable result is that simple algorithms, as the decision trees, (section 2.7) using records from past experiences are generally more effective than sophisticated algorithms for this kind of tasks. In addition, these algorithms are not specific for a unique problem and can be reused in many areas. So, the use of massive amounts of information and relatively simple learning algorithms resolved many problems which seemed inaccessible. A lot of other examples as the text recognition (OCR), the speech recognition, the recommendation of products on a e-commerce platform and the spam detection are also unresolved problems which demand a learning approach.

Naturally, some algorithms are more efficient in certain problems than others. Finally, with these examples, we can noticed that machine learning is used everywhere and in everyday life.

2.2 Definition of machine learning

Even among machine learning experts, there is no well accepted definition of what machine learning is. Nevertheless, there are enough known attempts which will be explored.

Definition of Machine Learning (Arthur Samuel 1959): "Field of study that gives computers the ability to learn without being explicitly programmed".

Samuel's reputation comes from the 1950s, when he wrote a program that can play chess [60]. The algorithm had the special ability to be able to learn the positions which most often led to victory. So, the program learned over time, what were the best moves to play and those that were not. In this way, the computer even becomes better than Arthur Samuel. The most amazing result is that, unlike a human, the machine has the patience to play thousands or millions of games without being bored. Therefore, the computer can acquire many more gaming experiences than a human.

However, this definition can seem quite informal. There is a more recent and more formal definition which was proposed by Tom Mitchell who defines what is machine learning and what is a well-posed learning problem.

Definition of machine Learning by Tom Mitchell [42]: " Addresses the question of how to build computer programs that improve their performance at some task through experience ".

Definition of a well-posed learning problem by Tom Mitchell [42]: " A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E ".

This definition can seem quite formal and so it will be illustrated. For example, for a program which want to distinguish persons who suffer from melancholic depression from a general population. Task T would consist in classifying people as suffering from melancholic depression or not. Experience E would be to observe an expert who marks a person as suffering or not. Finally, performance P would be the rate of person correctly classified.

Definition of pattern recognition analysis by Bishop [15]: "Field, within the area of machine learning, which is concerned with the automatic discovery of regularities in data through the use of computer algorithms to take actions such as classification"

2.3 Supervised learning

The term **supervised learning** refers to the idea that an expert will teach the computer to do something thanks to his knowledge. The learning algorithm receives a set of **labelled data** from experts in order to get the ability to make predictions for unseen data in the future. A data is labeled when the answer is given. For example, a labeled data consists of a features about the person which represent the data and the label which is “melancholic depression” or “not melancholic depression”. In supervised learning, the data used for training are always labeled. More formally, we have a data set which is called a **training set** and the objective of a learning algorithm is to learn from this training set in order to make prediction over unseen records.

In order to describe machine learning algorithms, some notations need to be defined. So, lower case m will be used to denote the number of training examples in a training set. While lowercase x will represent a single record which contains a set of features. A feature is an input variable which contains information about a record. For example, a record could represent a patient with two features: the reaction time and the speed during the digit symbol test. The variable y is the answer according to x. For example, “melancholic” or “healthy. We will use (x, y) to denote a single training example. Therefore, the training set is represented by a matrix X where a single row corresponds to a single training example, we will use (x_i, y_i) to denote the i^{th} training example.

2.4 Definition of models in machine learning

In machine learning, only the data are observed as the process is not directly accessible. To illustrate this statement, it is like guessing the score of a tennis match. The process would be to play the match and have the result at the end while an approximation of the score can be built, by gathering data as the performances of the player during the year, their previous results against each other, the past injuries or the favorite field of the players. In this context, learning involves the development of a good and useful approximation.

Definition of models [26]: “A model provides an approximation to the process which generates data”.

Predictive models help to make prediction about unseen objects. For example, the model can be used to determine what are the disorders of a patient. While, in **descriptive models**, data are observed in order to gain domain knowledge. For instance, this kind of model may help to discover what are the best characteristics, to figure out psychomotor retardations.

2.5 Linear regression (Supervised Learning)

The linear regression is a type of supervised algorithm which will try to predict a real value as an output. This technique is one of the most intuitive learning algorithm and will be used in this section to introduce supervised learning. So, a polynomial representation will be used to describe the error in a model. Then, the learning phase will also be discussed by using calculus as the derivatives. Moreover, in future sections, this representation will be quite appreciated to introduce the concepts of underfitting and overfitting (section 2.8).

In the next part, we will focus on classification rather than regression by describing the decision tree and the random forest algorithms which will be used to detect motor and cognitive impairments.

In linear regression, the model generated is represented by a mapping function from x to y called the hypothesis. In our case, x is a simple real value and θ_0 and θ_1 are the parameters of the function. These two parameters are contained in a vector θ. So, the hypothesis function is represented as:

$$h_\theta(x) = \theta_0 + \theta_1 x \tag{2.1}$$

The parameters θ_0 and θ_1 must be chosen in order to make the best predictions from the x value. For example let suppose that we have $\theta_0 = 1$ and $\theta_1 = 2$:

$$h_\theta(x) = 1 + 2x \tag{2.2}$$

In this context, the hypothesis function will looks like a straight line as seen in Figure 2.1. The main objective of the hypothesis function is to fit the data as much as possible.

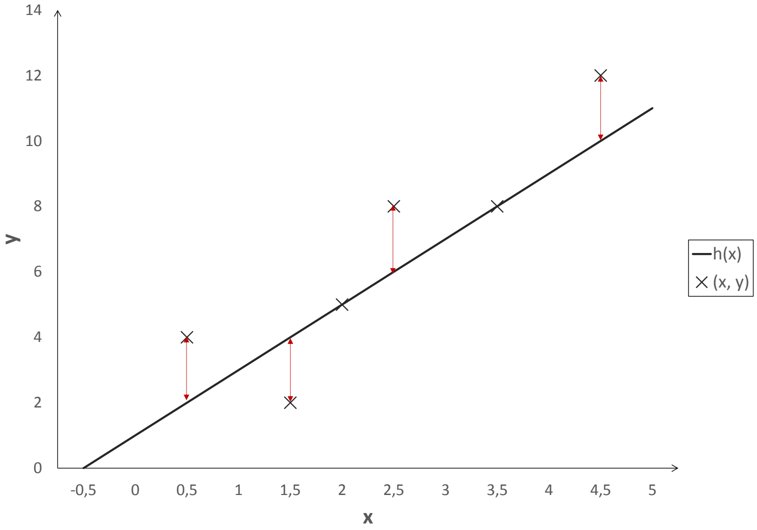


Figure 2.1: Distance between the hypothesis and the training set.

So, the parameters θ_0 and θ_1 are chosen in order to make reasonably accurate predictions for the y values. More formally we want the difference between $h(x)$ and y to be small. So θ is chosen in order to minimize this error. So, the main objective is to solve a minimization problem. Therefore, the goal is to minimize the squared difference between y and the hypothesis over the training set. The cost function J, which will be minimized as much as possible, can be written as the following:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})) \tag{2.3}$$

In Figure 2.1, the distance between the hypothesis and the training set is represented by the arrow between each pair of (x_i, y_i) and the hypothesis. To get the average error, the sum is divided by m.

2.6 Gradient descent

Intuitively, the gradient descent algorithm corresponds to the learning phase. At each iteration, a model will be created and then updated to reduce the error until an acceptable error has been attained. More formally, in order to minimize the error of the cost function, θ_1 and θ_1 must be updated until a satisfying minimum is reached. In this case θ is the vector which includes θ_1 and θ_1. So, the vectorized representation looks like:

```
repeat
    $\theta := \theta - \alpha \frac{\partial}{\partial\theta} J(\theta)$
until convergence
```

The **until loop** is made until convergence. It's up to the user to define what he considers as convergence. For example if the user discovers that after 1000 loops, a good global minimum is already reached, the algorithm can stop.

The α is a important meta-parameter called the **learning rate**. Basically, at each iteration, a step is taken to reach the global minimum. What α controls is how big the step will be.

The second term is the partial derivative term $\frac{\partial}{\partial\theta} J(\theta)$. For the purpose of this work, the complete formal demonstration is not needed. However the intuition behind this derivative term will be described. Let suppose that the cost function looks like the x^2 function in Figure 2.2. For $\theta = 1$, the slope at the tangent point is positive. So this straight line is positive and the derivative term is also positive. As the learning rate is always positive, θ will be update as θ minus a positive number. Thereby θ, which was equal to 1, will decrease. Similarly, for $\theta = -1$, as the slope is negative, the derivative term will be negative. So, θ will be update as θ minus a negative number and so θ will increase. As a result, θ will be updated in both cases in order to decrease the cost function. Finally, once the global optimum will be reached, the slope of the tangent line at the global optimum will tend to zero which means that theta will remain unchanged.

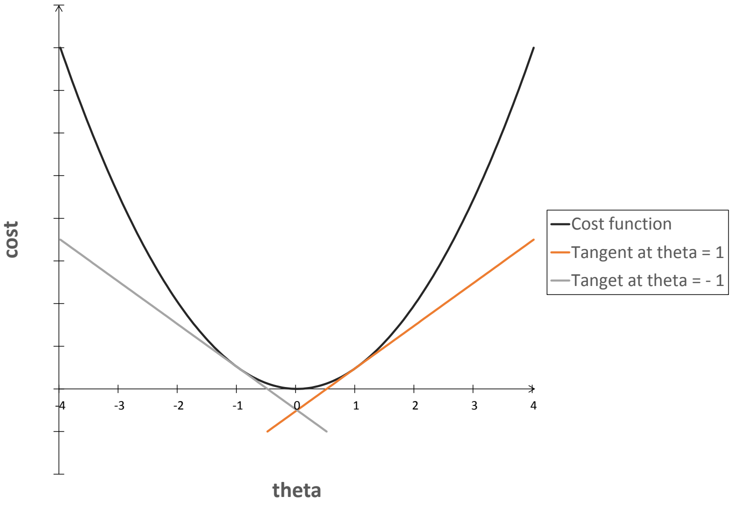


Figure 2.2: Intuition about the derivative term of the minimizing function.

The linear regression was a good algorithm to introduce supervised learning or more generally machine learning. However in the case of this work, we are facing a classification problem where the prediction of a class is expected.

2.7 Decision Trees

A decision tree is a predictive model generally used for classification. The model is represented by a tree composed of nodes: the root node, the internal nodes and the leaves. When new unseen records need to be evaluated by the model, the predictive algorithm always starts in the root node. This root node and the internal nodes involves one and only one feature and have two outgoing edges representing a binary split. So, a new entry just follows a series of very simple tests until it reaches a leaf node which gives the distribution for the classes which are predicted.

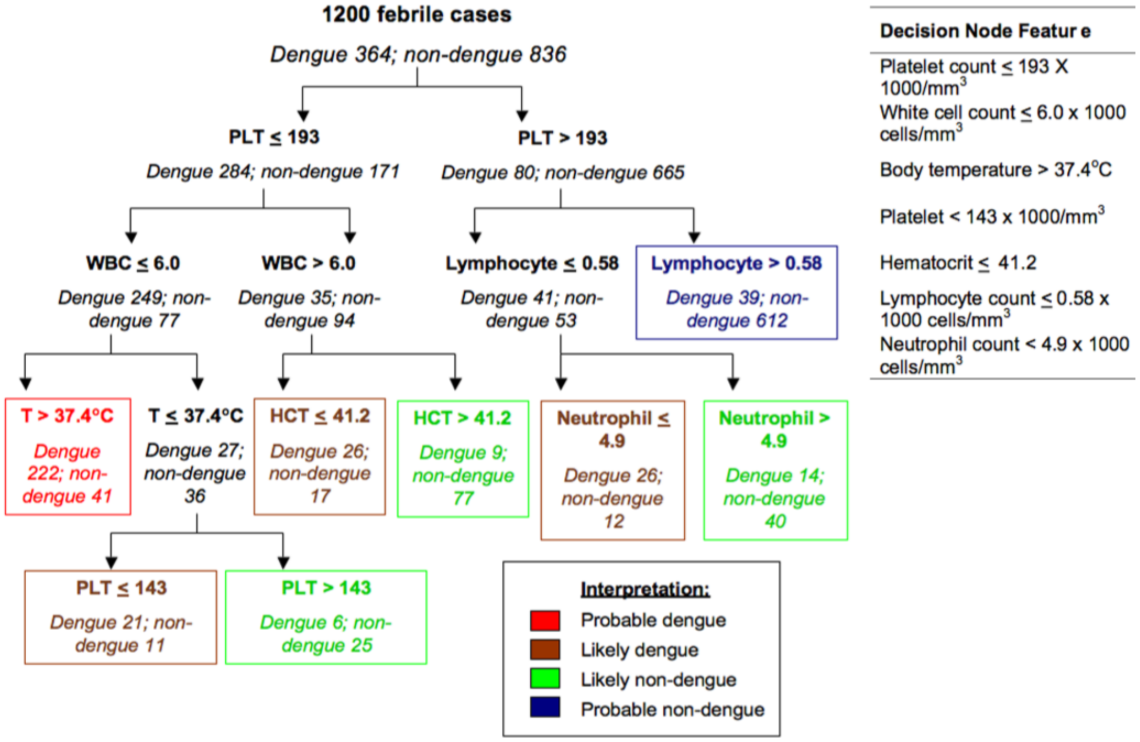


Figure 2.3: Decision tree model for dengue diagnosis reproduced from [68].

Figure 2.3 shows an example of a decision tree, from [68], which aims to diagnosis the probability of suffering from dengue which is a fever from the tropical world. For each node, a feature, but also a threshold discriminating the instances of the training set are selected. The leaf nodes represent a probability of suffering from dengue. For instance, if the platelet count exceeds 193 and that the lymphocyte count is superior than 0.56 then, the subject is probably not suffering from dengue with a probability of 0.936 %.

It is fair to say that the a decision tree is a very intuitive model and easy to understand even for people who are not experts in computer sciences. Indeed, the model is quite descriptive and can express the knowledge easily. By enumerating all the possible paths, a set of rules represented by conjunctions can be gathered. Then, they can directly be used by experts to predict a value even without browsing the tree. However, with a large number of features, the decision tree can quickly become difficult to read, but also to build by hand, although this is not really a problem for a computer.

As the project will be mainly based on random forests which are using decision tree, it is useful to present briefly the ID3 algorithm, one of the algorithm, which allows to build a decision tree. **Algorithm 1** is recursive. A node is created and is labeled with the most frequent label in the training set. If all instances in the data set are labeled with the same class c or if there is no more feature to consider, then the node can be returned as a leaf. Otherwise, the best discriminating features A according to the records in X becomes the decision feature of the node.

For each condition on the feature A, called a, if it exists at least one record in X where the value for this feature respect the condition a: the algorithm iterates with the records of the training set which respects the condition, but also with the set of features minus the feature A which has just been used.

Otherwise, a leaf node is simply added to the current node and the for-each loop is continued. We can see that the algorithm is very simple to implement, but also that it can learn efficiently. It should be noted that there exists many variations of this algorithm. In our case, we tried to present the more abstract one. For example, it exists implementations where the features are not removed after being used. In Figure 2.3, the platelet count is used at the beginning, but also at the ending. Moreover, some implementations are using, more than two edges for the split. All these variations can be used to optimize the algorithm.

Algorithm 1 function ID3(X : a training set, F: a set of features)

```
if ($X$.isEmpty()) then
    Error()
end if

$node$ = createNode();
$node$.label = getMostFrequentLabelInX($X$)

if ($X$.isAllInstancesLabelledWithTheSameClass()) then
    return $node$
end if

if ($F$.isEmpty()) then
    return $node$;
end if

$A$ := getBestFeaturesToClassifyX($F$, $X$)
$node$.decisionFeature = $A$

for each (a in A) do

    $Xf = \{x_i \in X | x_{ij} = a\}$

    if ($Xf$ != $\emptyset$) then
        $tree$ = ID3($Xf$, $F \backslash \{A\}$)
        $node$.addChild($tree$)
    else
        $leaf$ = createNode();
        $leaf$.label = getMostFrequentLabelInX($X$);
        $node$.addChild($leaf$)
    end if
end for

return $node$
```

Only one part of the algorithm remains uncertain: how to determine the best feature? Indeed, ID3 provides the freedom of choice. There is different algorithms which are used to determine the best split at each node. At a split, the main objective is to choose the variables and the values for a split which will maximize the information gain. In other words, the most discriminative value is chosen to make the split at in order to separate the classes while going down. This choice is really important as it will determine the quality of the model [26].

Despite all the benefits provided, the decision tree has also many negative aspects. A decision tree tends to overfit (see section 2.8) what can give poor results with test data. Moreover, the algorithm to construct the tree is a greedy heuristic and must browse all the features and the training set to determine the best feature. However, a simple solution exists: many decisions trees in a forest. So, the random forests will be seen in section 2.11.

2.8 Overfitting and underfitting

Overfitting is a recurrent problem in machine learning where a model acts with poor generalisation. A model with poor generalisation is a model giving accurate predictions over the training set but which acts very poorly over unseen data. In other words, this is a model conceived specifically for the training set and which is unusable in other cases. Needless to say that such a model is completely useless and that overfitting must be avoided.

Definition of generalisation: "Ability of a model to make prediction for new, unseen objects".

However, it is also important to avoid underfitting. Indeed, a model not enough complex will be too general over the training set, but also over the unseen records. In a few words, the model is unable to model the dataset. So, the right balance must be found between the two extremes as the main objective is to choose the model complexity which gives the best generalisation.

The complexity of the model is determined by the meta-parameters used during the generation of the model. These meta-parameters are chosen before the learning phase. So, the main issue is to find the right complexity by choosing correctly the meta-parameters. These meta-parameters are specific to the algorithm used and have the main objectives of optimizing the algorithm.

For example, in polynomial regression, the polynomial order of the hypothesis function is a meta-parameter. Figure 2.4 shows that it is not always possible to approximate accurately the records of the training set in a polynomial regression with a polynomial order equals to 1, even with the best algorithm and the best data. In this case, the model is too much general and underfit. While the algorithm performs already better with a polynomial order of 2. Maybe, the accuracy on the training set is not perfect but the generalization is really satisfying. However with a polynomial degree of 9, the accuracy on the training set is nearly perfect but the model seems poorly general. So, by overfitting, the model will certainly make poor predictions on unseen records.

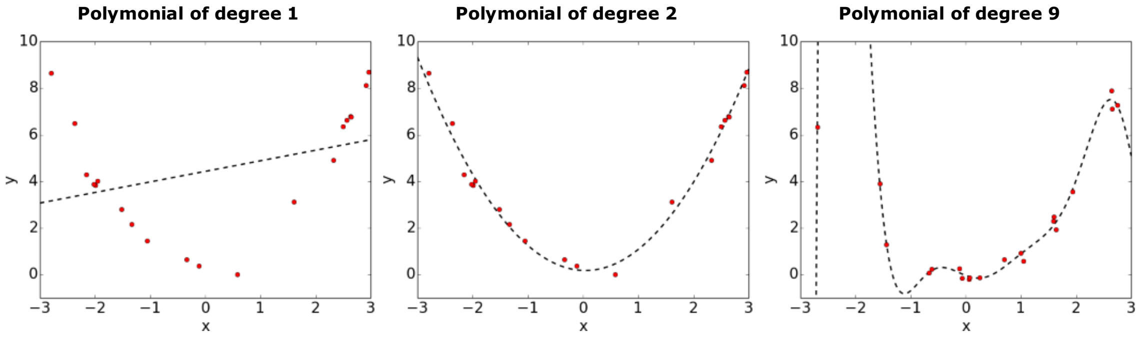


Figure 2.4: The order p of the polynomial is a meta-paramer influencing the model generated by the learning algorithm (inspired from [26]).

Choosing the right complexity is a difficult task as the meta-parameters depend on the data set characteristic in terms of size and quality. Moreover, the number of features influences the meta-parameters.

2.9 Model testing

One may wonder how to select the right complexity. The solution is to test the result over a training set and a validation set. Indeed, the meta-parameters are selected specifically for the training set, but also for the parameters in order to generate an accurate model. Therefore, unseen data are needed to test the ability of the model to generalize. The idea is to split the data set into a training set and a validation set. The validation set is composed of unseen data which has never be seen by the model. Generally, two thirds are used as a training set and the remaining third is used as a validation set. This **simple validation** [26] is really easy and fast but it is reliable only in the case of a number of records which approaches an infinity. Indeed, the method is unreliable as the results can give good results with luck, but also give disastrous results with an other repartition.

An other solution is to do a **cross validation with k folds** [26]. In this approach, the data set is split into k folds, and the validation is repeated k times. At each repetition the training set and the validation set contain $k-1$ folds and 1 fold respectively. Each fold is only used once as a validation set. So, this method is more reliable because of the repetitions. Moreover, each record is also used, at least once in the validation set. Nevertheless, the algorithm needs more computation as k models must be trained and tested instead of one.

For example, we will suppose that the training set has 100 records and that a cross validation with 5 folds will be performed. In this context, each fold has 20 records and will be used as a validation set only once. This procedure is repeated for each fold where the 4 remaining folds are used to train a model which will evaluate the validation set. So, 5 different models using 80 records for training will be generated and each record will be evaluated only once.

One may think to use a leave-one-out cross-validation where k is equal to the number of instances m in the training set. In this context, each record is evaluated with a model trained with $m-1$ records. However, such a validation tends to be too optimistic because each model generated are just slightly different from the others. Moreover, it can be very time consuming to generate a large number of models.

2.10 Evaluate a model

Machine learning cannot be used without asking important questions. Even, the best tools cannot do everything. When a model is created, its ability to make prediction on unseen data must be evaluated.

The most basic information to evaluate a model is the accuracy, which is to the rate of correctly classified instances.

$$\text{Accuracy} = \frac{\text{true positives} + \text{true negatives}}{\text{total instances}} \tag{2.4}$$

However, even in admitting that cross validation was used, the results are not always sufficient. Let's assume that the training set is composed of 95% healthy patients and 5% of melancholic patients, which represents the positive class. If the model returns always healthy whatever the record, it will have an impressive accuracy but an awful predictive ability.

In this way, other measurements will be introduced. The precision is a measure which evaluate the number of false positives predicted by the system while the recall highlights the number of false negatives predicted. For instance, if we always predict melancholic, the recall will be equal to 100% while the precision will be equal 5%.

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{2.5}$$

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{2.6}$$

The F-score tends to be a more reliable measure as both the recall and the precisions are taken into account. However these measures are not sufficient to guarantee the quality of a model as a model completely overfitting will get great scores but will act very poorly in terms of generalization.

$$\text{F-score} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{2.7}$$

As the random forests will be used for future classifications, the out-of-bag error will be presented in the next section. This measure aims to approximate the generalization error.

2.11 Random Forest

Many medical systems have thousands of features but only few instances in the training set. In such cases, it is quite difficult to explore all the features with a single tree. So, a decision tree turns out to not be the best classifier in our context. However, by gathering many simple classifiers together, it becomes possible to get a very powerful classifier. This technique is also called "**bagging**". In this context, the random forest will be used in Chapter 7 to generate models able to detect cognitive and motor impairments. Moreover, the random forest algorithm is used by the Kinect in order to recognize the joints of a body but also to recognize a gesture. This point will be discussed later in Chapter 3.

A random forest is a combination of decision trees. For each tree, a random subset of the training set is selected in order to train a tree. Generally, this subset represents two thirds of the entire training set. Moreover, at each node, a subset of features is selected randomly. Then, a feature is chosen in this subset in order to make the best split. More formally, Leo Breiman [17], the author of random forest, gives the following definition in the detailed implementation:

Definition: "A random forest is a classifier consisting of a collection of tree-structured classifiers where each classifier are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x".

The classifier are said to be distributed as each tree can be grown independently from a subset of the training set. This property is particularly interesting. Indeed, if a training set contains billions of records, it would not be possible to load in the memory of a single computer every record. But a tree can be easily built with a smaller subset. So, to improve the accuracy of the classifier 10000 trees can concurrently be generated from smaller subsets on different computers. The objective is to explore a maximum of different instances of the training set.

One of the greatest advantages of random forest is that they are computationally inexpensive, in terms of complexity, because of the randomness. It will be particularly interesting in our case as we will have a lot of features.

2.11.1 Generalization error

The random forests have a specific measure, called the **out-of-bag error** to estimate the generalization error. For each tree, a subset of the entire training set T is selected with replacement in order to train the tree and the remaining records are the out-of-bag data. So, if k trees are generated, k training sets have also been created. Then, for each labeled record (x, y) from the original training set where x is a record and y the label, each tree that has not used (x, y) for training will vote to predict y from x. Then the most popular class will be te prediction for y. So, a out-of-bag classifier is created by repeating this procedure over the entire training set. As a result each record will have a prediction. Finally the generalization will be the error rate of the out-of-bag classifiers on the training set.

A very remarkable result has been stated by, Breiman [17], the creator of random forests: "According to empirical evidence, the out-of-bag estimate is as accurate as using a test set of the same size as the training set.". Therefore, by using this measure, the need of using a test set apart from the training set is no longer needed [17]. However being really careful and to confirm this statement, we will still use a validation set, but also k-fold cross validation for our classifiers.

2.12 AdaBoost

AdaBoost will be the last algorithm presented in this section. This algorithm is used in Visual Gesture Builder, a software of Microsoft which aims to recognize a gesture in real time. So, for a better understanding of the Chapter 4 we find useful to give a brief introduction about AdaBoost.

As the random forest algorithm, AdaBoost [25] is also using the bagging technique. In this context, AdaBoost gathers many very simple classifiers together, in order to get a more accurate classifier. However, the method used to train a model differs from random forests. For generating a set of T weak classifiers, AdaBoost uses the best simple prediction rules which minimize the error to generate a weak classifier. Then, the weight of the samples are updated in order to give more importance to the misclassified samples in the error. The procedure is repeated until T classifiers are generated. In this context, each classifier is based on the error of the previous weak classifier. At the beginning, each instance has the same weight and the most difficult instances to classify will get a much more pronounced error. In this context, the algorithm will focus on these instances to choose the simple prediction rules.

So, AdaBoost [25] is a fast algorithm quite simple to program which does not require to configure meta-parameters to prevent underfitting and overfitting except the number of weak classifiers generated. However, the algorithm has also important cons. Indeed, empirical evidences showed that AdaBoost is especially vulnerable to noise in the training set. For example, if instances have been wrongly labeled. Moreover, AdaBoost tends to overfit more than random forests.

Chapter 3

Motion capture with the Kinect

Motion capture is one of the key topics in this work. So, the first objective of this chapter is to give a brief introduction about what is motion capture. Then, some examples of application from different areas such as analysis, interface control and surveillance will be presented. Then we aim to explain why we opted for the Kinect sensor, to record the movements in order to detect psychomotor retardation. So, we will evaluate the accuracy of the sensor for detecting motor and cognitive impairments. For that purpose, the underlying algorithms of the Kinect used to track the body will be introduced. It is important to know how the Kinect works in order to determine the choice of the camera, but also for understanding why some parts of the Kinect are less precise than others. In addition, some points of the algorithm, as the introduction of variability in the training set could be interesting to integrate in our own training set. But that's not all, this section aims to introduce all the data that can be recorded by the Kinect during motion capture.

As both versions of the Kinect are discussed in the realization of this work, we define the following terminologies:

- Kinect v1: Refers to the Kinect which has been released on Xbox 360 and in 2012 for Windows.
- Kinect v2: Refer to the Kinect which has been released on Xbox One and in 2014 for Windows. Ironically, the original name of this Kinect is “Kinect One” which can be really ambiguous. That's why it is important to define these terminologies.
- Kinect : Both Kinect have a lot of similarities. So, this terminology will be used, when we want to refer to the both versions.

3.1 Introduction to motion capture

According to Taylor [69], motion capture, also known as **mocap**, is the “process of recording the movement of a person as a time series of physical or virtual points on the body in a 3-dimensional Cartesian coordinate system”. There are two general approaches. The former is the **marker-based** approach, also called the **active sensing** where the systems use several synchronized cameras in order to capture the location of *real* physical markers attached to the subject. And the latter, is the **markerless** approach or **passive sensing**, where machine learning is used in order to predict the *virtual* location of the joints of a body [69]. The last approach is not intrusive at all compared to the active sensing but is generally less precise. So, passive sensing is mainly used in surveillance and in some control applications, but also in the case where placing devices on a subject is not possible. Moreover, a markerless approach allows motion capture in natural environments such as outdoors or in a living room. In addition to being less precise, this approach generally requires more computation, but also an algorithm which has previously been constructed from data collected in a marker-based approach [43]. This fact explains why the markerless approach is less precise. Indeed, in such approach, the system tries to predict the positions of the joint according to the data observed in the marker-based approach.

Over the years, the analysis of human actions by a computer became one of the most active research topics in computer vision. Its popularity comes from its wide range of applications in many areas such as clinical analysis, surveillance, interface control, analysis of athletic performance, etc [36]. This process, also known as human motion capture, is generally used in three major kinds of application [43]: surveillance, control, detailed analysis of the captured data.

3.1.1 Surveillance

The surveillance applications aim to track one or more persons through time or to monitor special actions [43]. For instance, Parajuli (in [46]) introduces a system using a Kinect sensor to detect when a person is likely to fall. Indeed, the injuries caused by falling is one of the biggest preoccupation in hospital as the people most affected are the elderly and the patients which suffer from motor impairments. In addition, all people weakened because of an extensive treatment or a disease are subject to the risks. A fall may cause serious injuries, but can also affect the mental health. So, the system analyses the gait cycle to detect an abnormal gait in order to prevent a fall. The changes in postures are also evaluated, especially from sitting to standing with intent to recognize the postures which led to a fall. Another system which is really similar is described in [51]. The purpose of the system is to detect and prevent falls in hospital rooms by also using a Kinect because of its low cost.

3.1.2 Control

Control applications [43] focus on the human machine interactions where the inputs are the motions of a body. These kinds of applications are generally popular in video games where the movements of the player replace the usual controller. But it also exists applications which are facing major issues. For instance, in a hospital, the operating room must be a cleansed and sterilized environment to prevent an infection for the patients. So the contacts with a mouse or a keyboard are prohibited. That is how the gesture user interfaces are a real concern. In [56], Ruppert describes how they implemented a hand tracking system, for the surgeons, which allows to browse among images and information about patients with a Kinect device. They even argued that the "Kinect showed to be very efficient and enabled a low-cost and accurate control by just using hand gestures" [56].

3.1.3 Detailed analysis of the captured data

While the first two areas are generally using real time recognition. The detailed analysis of the captured motion data [43] is carried out retrospectively. They are generally used in clinical studies for diagnostics, but also to help the athletes to improve their performances. For example, in [67], they built a system which aims to evaluate the effect of a treatment (*Deep Brain Stimulation*) among Parkinson's disease patients by analyzing the movements during two different tasks. The first one consisted in simply analyzing gait abnormality while the second one was using a pull test. During this test, the patient is pulled back and its ability to recover is tested.

3.2 Kinect's History

At first, the Kinect v1 is a device which was intended to be used with the Xbox 360 gaming console. So, this device was designed with the unique purpose to be a sort of controller for the player. This device was launched on November 4 in the United Stated for the cheap price of $150. Of course, such a cheap price have contributed to its success. Indeed, the Guinness Books recorded it as the fastest-selling gaming peripheral of all time from 4 November 2010 to 3 January 2011 with an average of 133,333 units per day in its first 60 days of release [52].

At the beginning, Microsoft only considers the Kinect for gaming. Although the Kinect was compatible with a computer, Microsoft did not plan to publish their software development toolkit (SDK) or their drivers. However, they had not anticipated the fact that the sensor would be noticed by many creative developers and researchers because of its cheap price. Indeed, at that time, a sensor could cost thousands of dollars. Moreover, the capabilities of the sensor, are far away from being limited to the game. Indeed, we have already referred to some great applications using the Kinect. In this context, PrimeSENSE, the company which has developed the hardware part of the Kinect v1 publish its own drivers and SDK, called "OpenNI" compatible for Windows, Mac and Linux, well before the release of the official Windows version. So, a few years ago, a large open-source community was collaborating with PrimeSENSE. However, since the purchase of PrimeSENSE by Apple, the past activities of PrimeSENSE has been stopped and OpenNI has been almost forgotten.

On February 2012, the Kinect v1 for Windows has been released. This version was officially supported by Windows 7 but was not supported on the Xbox 360 although it is almost the same device. Thanks to the success of the first Kinect. Microsoft released on November 2013 a second version of the Kinect for its Xbox One but without the help of PrimeSENSE. Microsoft did not make the same mistake and released the Windows version on June 2014 because of the success of the first Kinect among developers and researchers. They also released an adapter in October 2014 that allows to use directly the Xbox One version with a computer.

3.3 Motivation for choosing the Kinect V2

Actually, until some years ago, sensing was not available for widespread use. Generally, this kind of sensor was at a high cost. That's why the Kinect has really been a major innovation in computer vision by offering a large more competitive pricing. It was suggested [19] that the Kinect has encouraged the development of low cost depth sensors. More specifically, the topics related to the Kinect, includes object tracking and recognition, human activity analysis, hand gesture recognition, and indoor 3-dimensional mapping. This broad diversity of topics clearly shows the potential impact of the Kinect [28] to solve fundamental problems in computer vision. Moreover, just two years after that the Kinect v1 was released, a large number of scientific papers as well as technical demonstrations have already appeared in diverse vision conferences or journals. The Kinect V2 has followed this trend by proposing a sensor at approximately the same price. However, this version provides more accurate depth measurements thanks to the **Time-of-Flight** (ToF) technology, but also a greater resolution

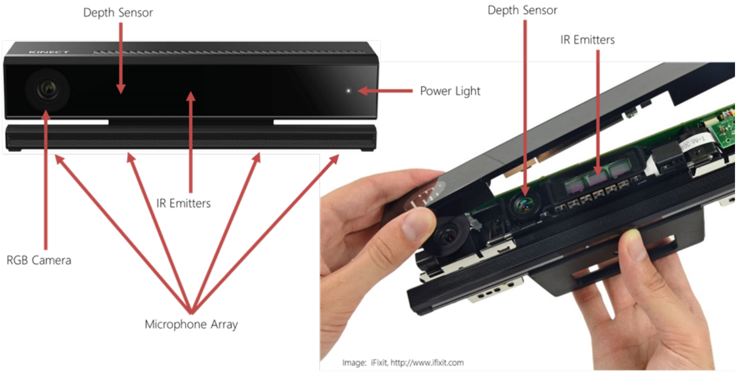


Figure 3.1: Depth sensor, IR emitters and RGB camera of the Kinect v2[1].

Indeed, the Kinect V2 is a ToF camera. These kinds of camera projects infrared light (IR) into the scene and then measures the phase-delay of reflected infrared light [40]. Knowing the speed of light, the phase delay of the signal is directly proportional to the distance between the camera and the object being measured. This measurement is performed independently for each pixel. Thereby the camera obtains a complete 3D image of the measured object. With such precision, it's possible for the camera to differentiate light reflecting from objects in a room and the light from the surrounding environment. This accurate depth estimation allows to compute the shape of those objects [38], but also to get more reliable results. In [40], they compared both versions of the Kinect and stated that the depth measurement precision appeared to be more accurate compared to the first version. In [9], they argued that the Kinect V2 show significative improvements concerning the face tracking. Especially for the detection of the face rotation and an area of detection more larger. It should be noted that the sensor act normally even in the absence of light.

The Kinect provides all data streams required in computer vision to deal with motion capture as a color video, a depth video and an infrared video. Indeed, Kinect resolves many motion capture problems and allows people who are not expert in focus problem to focus directly on their concerns. Indeed, Microsoft provides a powerful SDK with its Kinect. So, the Kinect allows anyone to make abstraction of the computer vision's problem by already providing, the joints position and orientation, predicted from depth data. Furthermore, a microphone array is available to deal with speech recognition.

But the great particularity of the Kinect is its ability to track the motion of an entire human body in real time without the person holding any object or any clothes and so on. It means that there is no need to use a motion capture suit or blobs. In other words, the Kinect is able to track a body, whatever the outfit or the equipments of the person. In this context, a person standing in front of the camera can directly start to execute movements. So, the Kinect is fully independent and doesn't require any other sensors. Indeed, generally,

[1] Reproduced from `https://www.ifixit.com/Device/Xbox_One_Kinect`

a markerless sensor only needs a computer to process the data. It is particulary interesting if a system at home as described in section 1.9, is desired.

However, as seen in Section 3.1, markerless sensors as the Kinect are not accurate as a marker-based system. By the way, the algorithm of the Kinect, which is described in the Section 3.5, has been trained with a marker-based system. So, the points predicted by the Kinect remain approximations even if the predictions are accurate. Moreover, tremors are really difficult to capture with the Kinect. By contrast, a markerless system is more affordable for the general public. In addition, such a system is less intrusive and can be used in natural environments, as a living room, more easily because of its flexible installation.

Finally, as already discussed in Section 3.1, many applications using the Kinect, even in the medical area already exist. A touchless interface has been developed in [56]. The interface allows the surgeon to interact with a computer during an operation without contamination. The Kinect [46] is also used in hospital for detecting the falls of patients and elderly people. For all of these reasons, we thought that the Kinect was the ideal choice.

3.4 Initialization

The **initialization** is the first stage during motion capture before tracking. According to Moeslund [43] “initialization covers the actions needed to ensure that a system begins its operation with a correct interpretation of the current scene”. In a nutshell, initialization mostly relates to camera calibration, adaptation to the scene and model initialization. There are many approaches used to resolve the model initialization problem which consists in building a 3-dimensional mesh of a subject whith the help of multiple cameras, mocap suits or stereo reconstruction. Zheng and Suezaki [75] proposes an offline approach where the operator specifies some key poses and then the system interpolates automatically the poses between the frames. In the online case, Wren [73] proposes an approach where a model is created in real time. So, an initial model is built from the subject’s initial pose before being refined as more information are captured by the system. Some systems do not create a new model for each subject and used a model which is the average of many subjects [13]. However, we will see in Section 3.12 that it is not completely true for the face tracking where a mesh of the face can be built. But it is not mandatory.

Finally, some systems are using the same algorithm for initialization and during tracking which means that no temporal information is used and that these systems are not considered as using initialization. The Kinect belongs to this last category. So, the Kinect does not need any initialization or calibration. The approach used to resolve the initialization problem is the same as for tracking which means that no temporal information is used and that these systems are not considered as using initialization. There is no need to create a model based on the subject. As already mentioned, in the advantages of the Kinect, a person standing in front of the camera can directly start to use it.

3.5 Tracking

All this section is using the article on the origin of the Kinect. This article is a work for about 8 years from the machine learning and perception group in Cambridge and especially

Jamie Shotton [64]. A part is also inspired by an interesting talk by Christopher Bishop about the Kinect [2]. This section aims to explain how the Kinect can tracked a body without interruptions, but also why this sensor does not need calibration. Moreover, this section will help to understand why some parts of the body give better results than the others in terms of accuracy. Finally, the idea of generating synthetic data over an initial training set to prevent overfitting, but also to get more training records can inspired us.

The standard solution for tracking, is based on the idea that at a particular frame, the position and the velocity of the skeleton is known. So it's possible to measure and predict where the skeleton will be on the next frame. As seen earlier, it's the standard approach to tracking. It requires that the system knows the initial configuration. For instance, the operator must ask somebody to stand with a T shape configuration, to get them lock and then the system has the initial configuration. But there are 2 major disadvantages. Firstly, we need to ask to somebody to stand for 2 seconds in T-shape. Secondly, the tracking is also problematic. Indeed, if the movements are slow, the system can make good predictions and remain locked but with quick movements the system can lose track and ask again to repeat the initialization. And it is really problematic with console, where you have people moving fast. In one second a part of the body can move from one side of the image to the other.

Even if the system is good enough to predict the next frame correctly with 99.9 % of reliability. It means that after just a minute, there is a 83 percent probability of failure and so the person, must go back to the initial configuration. Of course, this is not desirable, especially in the case of a reliable system is desired. But the approach used by the Kinect is completely different. Indeed, the goal for the developers of the Kinect was to build a system which uses 3D video to track the human body without initializations. Moreover, they wanted the system to "never fail" which means that the user must be perceived continually. In other words, a system which never stops working. Therefore, there is no initialization because at each frame the Kinect is determining the location in 3D space of some key points on the body, the joints. And by simply connecting them together, we got the skeleton. It must 'never failed' that means the user must perceive that it continually works so that it does not lose track. It also has to deal with a huge range of human poses with different sizes, shapes, clothing and whatever else. All has to be done with a computer budget using less than 10 % of the power of the Xbox 360. Therefore the computing power available was overly tight.

The approach was to not treat this like a tracking problem but instead as a view recognition problem. So the machine learning and perception group in Cambridge Shire have been working for about 8 or 9 years on object recognition recognizing objects in natural scenes. So, they were inspired by recent object recognition work that divides objects into parts [64] and they apply the idea developed in that context to the problem of Kinect. So the idea was to treat each frame of the depth video as a separate recognition problem where the body are fragmented into parts as an object (see Figure 3.2). Even if one frame is wrong for some reason, the next frame gets a fresh start. So, at each frame, the location in 3D space of some key points on the body, the joints, are determined. Then, all these joints simply need to be connected together in order to get the complete skeleton. The goal is to take every pixel in every frame of the video and decides at which class it belongs to. A classification problem needs to be solved. For each pixel in each frame

[2] `http://videolectures.net/ecmlpkdd2011_bishop_embracing`

we consider window of depth data around the pixel. Therefore the main objective is to assign pixel to one of these 31 classes, the window of depth data. So there is a probability distribution over those classes and every pixel must be assigned in one of the classes, for every frame.

According to Shotton [64], simulating human pose data is an unsolved problem. Indeed, the number of possible positions of a human joints is exponential and so it is impossible to record all possible poses. So, an important task was to collect labeled training data to learn that conditional distribution. Training data have been collected with a motion capture suit and a person which was moving around being viewed by multiple cameras from multiple directions. The cameras match up the corresponding blob in the different videos and a little bit of software computes the 3D locations of each of those blobs. For example, animation studios as Pixar, employs this a lot for animating characters with real human motion. In this context, the idea is the same, a person will play a sort of game and the Kinect sensor is pointing at them to use the motion capture data to get labeled ground truth. In this manner, 500,000 frames were captured.

Introducing variability by generating synthetic data

Unfortunately, this approach is not enough complete as it has too much variability in pose, size, etc. Moreover, in real situations, depth cameras can exhibit noise due to many reasons as non-IR reflecting materials, ambient illumination. In this way, with only perfect clean data, the model could overfit. To respond to this problem, they generate synthetic data which means that they took the data from the motion capture and they introduce synthetic variability it in order to prevent overfitting. So, they artificially corrupt depth images by varying resolution, by introducing pixel noise, or by removing pixels, by reducing the quality of the video, by trimming out frames or by introducing occlusions and so on. In order to introduce the variations that are expected to occur in the real world. It was expected that the classifier learns a large degree of invariance. For example, they want to exclude correlations such as thin people always wear a hat [64]. In this manner, they generate more than one million examples of body positions and with all the variability labeled. So they were able to build a predictive model.

- 3D models of 15 varied base characters has been used, with varying ages, from thin to fat, from short to tall and both female and male.
- Redundant poses which were too similar were discarded, which only leavers 100,000 frames.
- Poses were mirrored left-right.
- The characters were rotated at random about the vertical axis and translated in the scene in order to obtain frames where the character is only partly in.
- The characters were added mesh model of several items of clothing, but also hair styles.
- The height and the weight from their characters were varied by 10 percent.
- The camera height, pitch and roll were randomly chosen within a range, which was illustrative of a camera placed in a living room.

- Artificial noises were added to the depth image to simulate real depth cameras that can exhibit noise due to non reflecting materials.

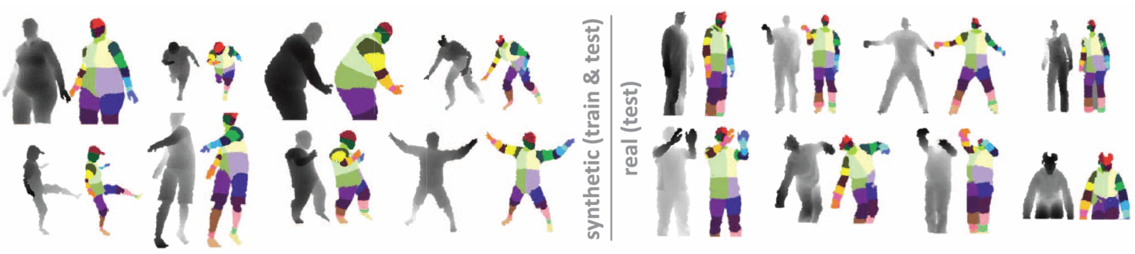


Figure 3.2: Synthetic data are used as test and training data while real data are only used for tests (reproduced from [64]).

Those synthetic training images have the ability to boost easily a supervised learning algorithm by generating ground truth labels almost for free.

Body part classification

The Body Part Classification (BPC) algorithm is used to densely predict discrete body part labels at each frame and then for predicting joint positions. Its implementation is based on random forests. Each of these labels are spatially localized near skeletal joints of interest as you can see in the Figure 3.2. So BPC aims to predict a body part label for each pixel of the body. In this way, for each pixel u, the pixel run through the tree until it reaches a leaf node $l = l(u)$, which give a distribution $p_l(c)$. Then the distributions for each three are averaged in the forest in order to get the final classification as

$$p(c|u) = \frac{1}{T} \sum p_l(c) \tag{3.1}$$

Now that, each pixel has a body part, the joint positions in 3D can be mapped from these body parts by looking at the centroid.

Offset joint regression

The Offset Joint Regression algorithm goes further than the BPR algorithm by estimating the joints position directly in 3D. Before the training phase, the joint positions were labeled in a 3-dimensional vector which were trivially recorded manually during the elaboration of a mesh. Moreover, the use of an intermediate algorithm such as the Body Part Classification was a key contribution. So, the joint position must be find in the specific body part already labeled. Finally, the mean shift algorithm is used to give the final set of 3D body joint proposals [64].

3.6 Body frame source

Thanks to these algorithms the Kinect can detect up to 6 bodies simultaneously in a range from 0.5 to 4.5 meters. As seen in Figure 3.3, for each body in front of the camera, the Kinect can recognize 25 joints at 30 frames per second.

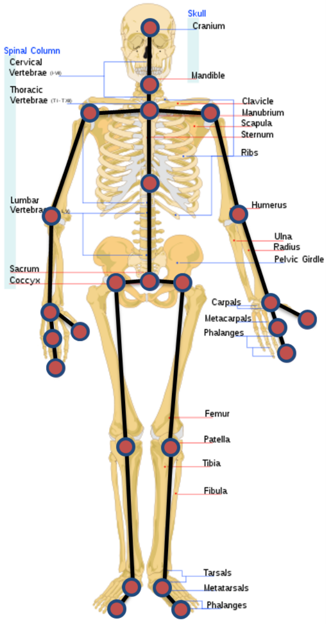


Figure 3.3: Overview of the 25 joints detected by the Kinect V2[3].

3.6.1 Joints position

For each joint, three different **tracking states** are possible. In this case, "tracked" means that the Kinect considers the joint as tracked accurately. A joint can also be "infered" which means that the Kinect is not able to track accurately the joint, but the algorithm is doing its best to estimate the position of the joint. Unfortunately, joints that are inferred tend to have temporary spike noises as they are less accurate. This may be due to several reasons: a part of the body can be out of its range of view or it can simply be an error [5].

One of the most important and remarkable information provided is of course the **position** of the joint. This position is represented by a 3-dimensional vector as visible in Figure 3.4. Intuitively, the x position represents the movements on the vertical axis, if we suppose that the subject is facing the camera. The y position represents the movement on the horizontal axis and the z-axis represents the distance from the camera's perspective. So, the more the subject is close to the camera, the more the value is high.

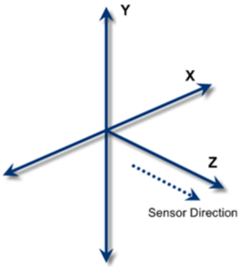


Figure 3.4: Vector xyz of the Kinect in a 3-dimensional Cartesian space[4].

[3] https://www.microsoft.com/en-us/kinectforwindows/develop/learn.aspx (Visited on 01/08/2015)

[4] https://msdn.microsoft.com/en-us/library/hh973078.aspx (Visited on 01/08/2015)

These data cannot be analyzed directly as they are not viewed-invariant. Indeed, the results depend on the camera's perceptive, the height and the weight of the subject. An important problem is to suppress irrelevant source of variation in the representation of motion. So, the objective is to make a representation that is invariant from the point of view of the camera. But it is possible to compute invariant measures about the movements between two frames. For instance, the translation or the distance will allow us to compute the speed of a movement. A translation moves all the points of a geometric object by the same distance in the same direction according to a vector [33]. The distance between two vectors can be computed as:

$$s_i = \sqrt{((x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \tag{3.2}$$

However, this vector alone does not allow to capture all the movements as a joint can stay at the exact same position while rotating. For example, while turning a key, the position of the wrist is not moving that much although a movement is executed.

3.6.2 Joints orientation

Thankfully, in addition to the position of the joints, the sensor provides a **rotation** for each joint. This rotation is expressed in a four-dimensional vector Q, called a quaternion (X, Y, Z, W). A quaternion describes the rotation as the vector perpendicular to the attached bones in the joint hierarchy. This hierarchy is elicited in table 3.6.2. Moreover, this hierarchy gives the connections between the joints and are also described as bones. It seems obvious that unlike the position, the orientation is invariant from the scene. Theoretically, the values remains the same from any perspective, but also for any person. Indeed, we are measuring the angles between two joints in a 3-dimensional perspective.

As a reminder, a rotation is a circular movement of an object around a center. So in a 3-dimensional plan, an object always rotates around an imaginary line called a rotation axis. In the case of a entire body, we can describe the rotation as the motion of a rigid body around a fixed point which is generally the center of gravity or the torso. So, the body is said to rotate upon itself [33]. In our context, the fixed points are the joint positions. As the result, the position of the joints and the rotation of the joints coupled, are able to describe any movement.

Parent joint	**Child joints**
SpineBase	SpineBase, SpineShoulder, SpineMid, HipLeft, HipRight
SpineShoulder	Neck, ShoulderLeft, ShoulderRight
Neck	Head
ShoulderLeft	ElbowLeft
ElbowLeft	WristLeft
WristLeft	HandLeft
HandLeft	HandTipLeft, ThumbLeft
ShoulderRight	ElbowRight
ElbowRight	WristRight
WristRight	HandRight
HandRight	HandTipRight, ThumbRight
HipLeft	KneeLeft
KneeLeft	AnkleLeft
AnkleLeft	FootLeft
HipRight	KneeRight
KneeRight	AnkleRight
AnkleRight	FootRight

Table 3.1: The orientation of a child joint is define according to its parent joint.

One may wonder why the rotation is not represented in a 3-dimensional vector as the position. Indeed, in the area of aviation, but also in computer vision, one of the most common representation is the Euler angles, expressed in radians, called the yaw (Z), the pitch (Y) and the roll (X). Indeed, this representation of the rotation of an object is quite intuitive to understand as seen in the Figure 3.5. However, this representation has a major issue, called the "gimbal lock", which has been a real problem for years for aeronauts before the space age [31]. But, the main objective of this document is not to describe in details what is a gimbal lock as the subject is quite complicated and require complex algebra. In addition, the complete demonstration will not help to understand this document itself. The important information, in our context, is to be aware that the Kinect provides quaternion and that they are converted into a more intuitive representation without any loss of precision: the Euler angles.

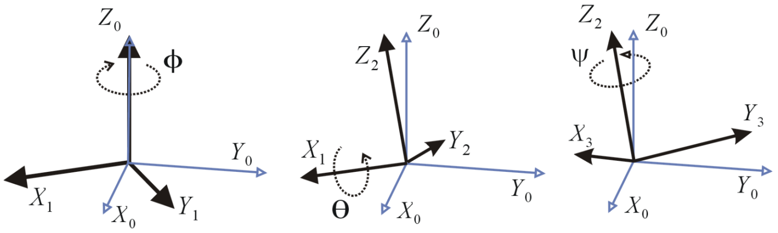


Figure 3.5: Rotation with euler angles: yaw pitch and roll.

To put it in simple terms, a rotation with Euler angles is a sequence of rotations of the 3 coordinate axis where the order is important. For example, in a ZXY rotation sequence (Figure 3.5), the Z angle is rotated then the X and finally the Y. So, Euler angles are not commutative: a different order in the rotation sequence with the same values will give a different result.

The “gimbal lock” happens when two gimbals are superposed and describes the same axes. The gimbal is represented by the circle around the axis in the Figure 3.5. The gimbals are said to be locked as further rotations will have disastrous consequences. For instance, the two gimbals will be stuck together during a rotation. Actually, one of the most used solution is to add an axis. Indeed, the quaternions are notorious for being compact and efficient. In addition, they never suffer from gimbal locks [31].

Diebel argued in [63] that “converting between representations is sometimes necessary to avoid gimbal lock”. Moreover, many papers [63][24][11] gives the equations to convert the quaternion in Euler angles:

$$Z = \mathrm{atan2}(2Q_Y * Q_W - 2Q_X * Q_Z, 1 - 2Q_Y^2 - 2Q_Z^2) \tag{3.3}$$

$$Y = \mathrm{asin}(2Q_X * Q_Y + 2Q_Z * Q_W) \tag{3.4}$$

$$X = \mathrm{atan2}(2Q_X * Q_W - 2Q_Y * Q_Z, 1 - 2Q_X^2 - 2Q_Z^2) \tag{3.5}$$

However if $Q_X * Q_Y + Q_Z * Q_W = 0.5$ then Z and X a replaced by:

$$Z = 2 * \mathrm{atan2}(Q_X, Q_W) \tag{3.6}$$

$$X = 0 \tag{3.7}$$

Else if $Q_X * Q_Y + Q_Z * Q_W = -0.5$ then Z and X are equal to:

$$Z = -2 * \mathrm{atan2}(Q_X, Q_W) \tag{3.8}$$

$$X = 0 \tag{3.9}$$

The results of the implementation of these equations in our prototype can be seen in Figure 3.7. So, we figured out that the representation in Euler angles is quite representative of the gestures executed. The blue line represents the Z axis, the red one is the Y axis and the green one is the X axis. As already mentioned, the orientation of a joint is dependent from its parent. For example, if the elbow is rotated then the same orientation is applied to the wrist, but also to the hand. Naturally, the inverse is not true. At the beginning, the original orientation of the “spine base” is computed according to the camera perspective in 3D. Then, the orientation of the children is computed depending on their parent. The procedure is repeated until the last children. In the future of this work, it would be interesting to focus on the quaternion. However, as the positions of the joints are the most accurate results of the Kinect, we preferred to focus on the position.

3.6.3 State of the hands

It’s also possible to determine the state of the people’s hands. For example, to determine if a person is interacting with an object. The states which can be returned are:

- Open

- Closed
- Lasso
- NotTracked
- Unknow

Although the open state and the closed state are pretty easy to guess, the lasso hand state needs some precisions. This state is represented as a closed hand with the middle and index fingers both up like a pointer. It is like the peace sign, but the two fingers are together, instead of separated. This state is likely to be used to interact with an interface. For instance, for selecting an area by using a circular motion or for drawing on the screen. In fact, two fingers are required in order to create something large enough in depth data to be recognized accurately. In this way, a finger large enough or close enough, will likely work as well. This information, can sound quite anecdotal, but it enlightens an important constraint of the Kinect. So, to be accurately recognizable, a part of the body must have a surface *large enough* or *close enough*. This explains why the Kinect is getting more accurate results with certain parts compared to the others. Like the joints, the SDK also provides a **confidence** about the prediction of the hand state from low (0) to high (1).

3.7 Color frame source

The color camera delivers high quality images with a resolution of 1920 x 1080 pixels for a frame rate of 15 or 30 frames per second according to the lighting conditions. It should be noted that the image is mirrored. Indeed, the image of the left hand is on the left and the right hand is on the right. In a nutshell, it's just the reverse of what a camera normally sees. As a result, the image provided by the Kinect is a much more intuitive interface, especially for video games or tasks where gestures must be executed while being visualized [16]. Color image data are available in different formats which determines the encoding of colors:

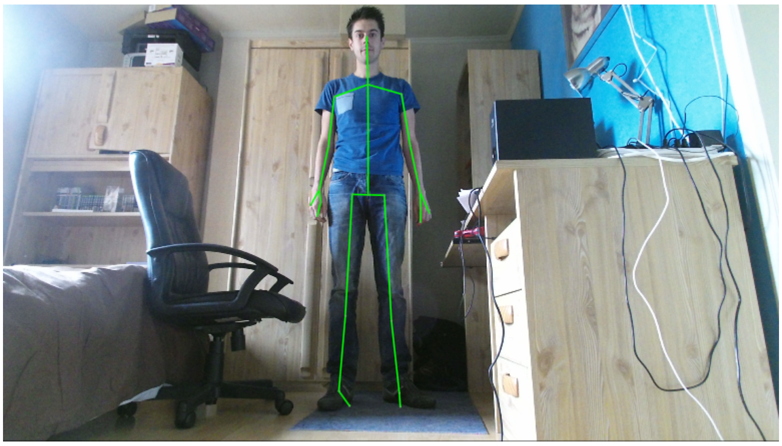


Figure 3.6: BGRA representation of the color frame source

- RGBA (Red Green Blue Alpha): It's simply the same model as RGB but with extra information. The alpha is an integral value varying from 0 to 100 where 100 gives a fully opaque pixel and 0 a fully transparent pixel.

- BGRA (Blue Green Red Alpha) : Mainly the same as Bgra but the bytes are stored in a different order.

- YUV : This color model is defined in 3 components. Y as one luma (the brightness) and UV as two chrominance (color) components. This model is used in the PAL and SECAM standards. In a nutshell, Y provides a black and white image while U and V add the colors. But this model is really pertinent for interfacing with analog television or specific analog devices.

- YUY2 : A 8-bit YUV format that are recommended for video rendering.

- Bayer : A Bayer image is a filtered image. A colour filter is placed over the camera sensor, so that each region of the sensor receives light of only one of the three primary colours. 50 percent of the sensor receives green light, 25 percent red and 25 percent blue. So we need to be aware of which filter colour corresponds to each pixel. To produce the final colour image, an algorithm called demosaicing, derives the missing colour components of each pixel from the neighbouring pixels [59].

3.8 Depth frame source

So, the depth stream gives the depth value of every pixel in the visible area. The depth value is the distance of objects according to the sensor with a range from 0.5 to 4.5 meters. These values are usually represented by using black color for the furthest distances while using white color for the nearest distances. The resolution of this image is 512 x 424 pixels with a frame rate of 30 frames per second. A representation can be found in Figure 3.7. Its infrared sensor allows to identify objects in a dark room. In this way, the Kinect can recognize people and track bodies, even without any light visible to the naked eye.

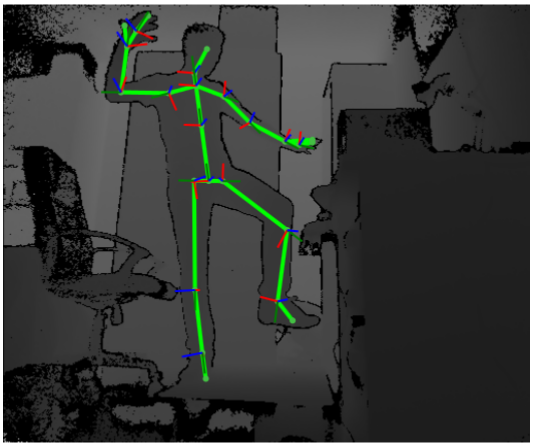


Figure 3.7: Representation of the depth frame source with the position and the orientation of the skeleton joint, each in a 3-dimensional vector.

3.9 Body frame source

Moreover, Kinect v2 provides a data stream which deals with the problem of background detection. This data stream is very similar to the depth stream. However, only the persons standing in front of the camera are visible. So, for each pixel of a 512 x 424 pixels images, the value equals to 1 in body region else 0 in background region. Indeed the algorithm is using a technique of “background removal”.

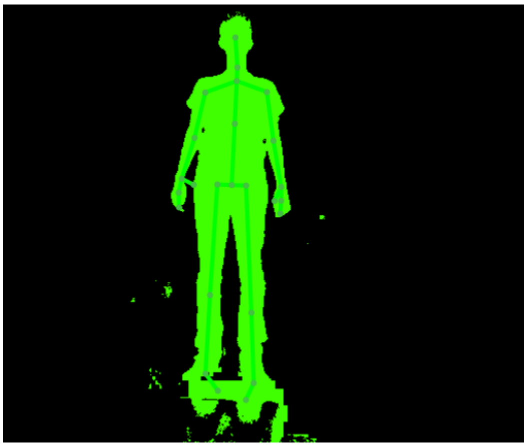


Figure 3.8: Representation of the body frame source similar to the depth but with only the body.

The background removal focus on the problem to distinguish people from their background. Background subtraction attempts to detect an object from the difference between the current frame and a reference frame, pixel by pixel. The reference frame is a model representation of the scene without the objects and it must be updated regularly in order to prevent varying luminance conditions and other settings [47]. There are several methods of background subtraction but all of these mainly lies in the type of the background model, but also in the procedure used to update the background model.

This frame source can be quite useful in the case where we would want to inspect parts of the body other than the joints position and orientation already provided by Microsoft. For example, each finger of a hand close enough. In this manner, this frame source would be coupled with the depth frame source. The body frame would give the location of the body and the depth frame would provide the depth map.

3.10 Infrared frame source

The Kinect is able also able to see in complete darkness as seen in Figure 3.9. This kind of data is generally used in computer vision algorithms which want to perform facial recognition. They are also quite effective while tracking reflective markers. As body frame source, this source can be quite useful in the case where we would investigate other parts to recognize, in this case the face. This source can also be coupled with other sources as the depth, in real time, to get a maximum of information. Naturally, this multitude of data source is a great strength for the Kinect to get accurate results as it is possible to have a lot of information for each pixel observed by the camera.

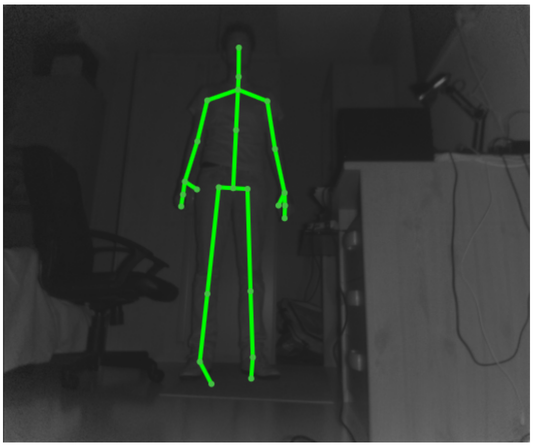


Figure 3.9: Representation of the infrared frame source.

3.11 Face frame source

Luckily, the SDK already provides the face recognition with two different sources. The first one, called the face frame source, is only providing some information about the face but does not allow a complete face tracking. So, unlike the body, a 3D meshed model cannot be created from the face data. This source allows to get a **bouding box** around the face. Moreover, this box is delimiting the face in the color space, described in Section 3.7. In this way, the right, left, bottom and up boundary location is given in the 1920 x 1080 pixels. In addition, this source also provides the orientation of the face with the yaw, the pitch and the roll (see Section 3.6.2) where the pivot point is the head joint of the body.

In addition, it also provides a lot of different features called “animation unit”. An animation unit is a result about an event which has occurred. The result can take 4 values. “Unknown”, means that the algorithm has not enough information to make a prediction. Typically, when the face is not tracked. “Maybe” means that the algorithm thinks that the event has occurred but is not sure. It can also be the result of a too small movement for realizing the event. “Yes” and “No” are quite explicit and mean that the algorithm is certain. The list below contains the animation unit relevant in our case:

- Happy: “Yes” when the subject is smiling.
- Left/right eye closed: The eyes are a great source of information concerning the attention.
- Looking away: “Yes” when the user is not facing the camera. However the Kinect is not able to track the pupil or more generally the movements of the eyes. So, it relies only on the position and orientation of the face with respect to the camera. It should be noted that this measure might be quite useful to measure the attention.
- Engaged: “Yes” when the subject is looking at the camera with the eyes opened. This measure, pretty useful for measuring the attention is combining the “looking away” and the “left/right” eye closed.
- Mouth moved: Measures if the mouth is moving. However, by testing, we noticed that the measure was not accurate or reliable at all.
- Mouth opened: As its name suggests.

3.12 High definition frame source

The Kinect also provides a more efficient way to get information about the face, which includes a stage of capture before the tracking. The API will tell how the subject must face in the camera in order to build the model. So a sufficient quantity of frames from different perspectives must be gathered to generate a 3D meshed model of 2340 triangles representing the face. During the tracking, the model will be deformed by the parameters. However, this capture is not mandatory. In this case, an "average face" will be used to represent the face.

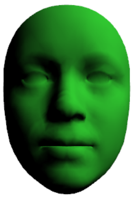


Figure 3.10: Representation of the face mesh model, deformed during tracking.

The tracking in high definition also incorporates the features from the simple API but with further details. So, the animation unit, are also available, but the results have not the same representation. In this case, the result is a value varying from 0 to 1 which gives the confidence of the prediction. Moreover, the unknown value is replaced by a *null* value. This representation is quite appreciated, as it brings more details than simply yes or no. In addition, ten more animations are available which concerns the cheeks, the lips or the eyebrows. However, by some testing with our prototype, we noticed that the results of these ten animation units were very inaccurate. The most reliable are the "looking away" and the "mouth opened". In our context, the eyes are pretty important. The tracking is quite good, but the eyes need to be close enough from the camera. So, the value for the eyes is not always available. In this way, we choose to only consider these four animations. This difficulty to track correctly some animation units can be explained by the underlying algorithm of the Kinect which need a sufficient depth area in order to give confident predictions. Indeed, if the eyes of the subject are closer to the camera, then the depth are is larger. Some features are also provided as the hair color, the skin color but we are not interested in.

Actually, the mesh model will not be used to capture the movements of the face because there are already a lot of opportunities to exploit. However, this model could be used in the future for face recognition in order to get more data than the simple animation unit.

Part II

Detecting neuropsychiatric disorders by motor pattern detection through motion capture and learning algorithms

Chapter 4

Data to collect

At first this chapter aims to indicate which data from the Chapter 1 will be recorded. So, we will describe how the diseases, the medication but also other meaningful clinical information will be organized. Then we will discuss which data from the Kinect will be recorded during motion capture for future analyses. In this context, a large quantities of data will be recorded for each patient, through a period of time, before applying statistics or machine learning algorithms. In short, the purpose of this chapter is to elicit all the data which will be recorded by the system for further analyses but also to describe all the reasons, which led to these choices. Moreover, the recognitions of the tasks implemented to score the subjects and to control the assessments in a completely normalized way will be detailed.

4.1 Data about a patient

At first basic data about the patient. The *birth date* is an important variable to store according to the Section 1.4 as aging already causes slowing. Moreover the *gender* is also an meaningful data since there is no consensus about the impact of this variable on the motor system as mentioned in Section 1.3. Of course, other data such as the *last name* and the *first name* are registered in order to identify the patient although they are not significant for further analysis.

In order to determine neuropsychiatric disorders by a machine learning model, it is important to know if a patient has already been diagnosed by a specialist. Indeed, data could be used to **train** the algorithm in order to make better predictions in the future. Moreover, for a **prediction** purpose, it is interesting to indicate to the algorithm that this person has already been diagnosed. So, the pathology is stored as the *beginning date* and eventually an *ending date*, if it exists. It should be noted that some pathologies are incurable. A scale from 1 to 10 is also available for the specialist to give an appreciation on the *stage* of the pathology. Obviously all these data have also an informative purpose. In the same way, a comment is also available to add a note.

Likewise, it is important to know with which *medications*, the patients has been treated. Indeed, all these medications have different effects and sometimes they are not perceptible before a month. Undoubtedly, the impact is completely different depending on the dose and the frequency. Therefore, the frequency is represented by an interval in number of days with a number of intakes per interval. While the dose is represented by a unit as milligram or milliliter and a real value. As beneficial psychomotor effects, may only appear on the long term, the *beginning date* and the *ending date* are certainly decisive data.

An abstraction of the data collected for a patient can be seen in Figure 4.1. As the goal is to gather a maximum of information about the patient, even if some data are missing, most of the attributes are not mandatory. Of course, the heritage can be used but the goal of the figure is only to give a clear perspective. Moreover, as all the important information are not mandatory for 'PatientMedications', 'PatientInformations' and 'PathientPathologies', an artificial attribute 'entry' has been added. This attribute is incremented by the number of time a patient has been referenced with the pathologies.

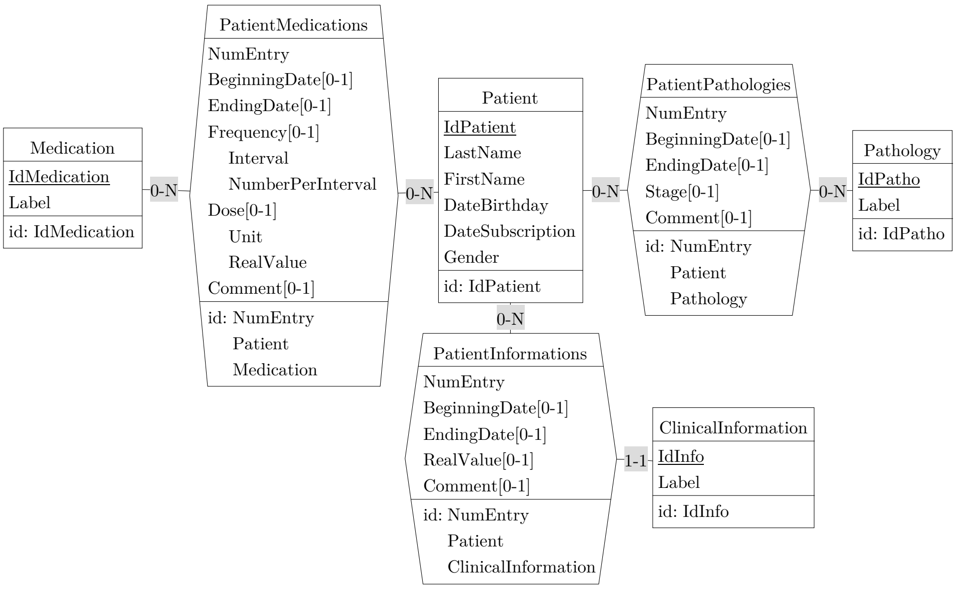


Figure 4.1: Data stored about a patient

4.2 Data about the motion of a patient

During motion capture, a lot of data will be gathered 30 frames per second in order to train models for a classification problems. However, we cannot ask a patient to start moving in front of the camera as he wants. Indeed, it would be extremely difficult to train a model as the data would be difficult to compare. Indeed, some people will just not move while others will move the arms or the legs. So, for some periods of time, several parts of the body would be more stimulated than others depending on the type of the task. For example, for a drawing task, as seen in Section 1.8, the arms will certainly be the most used part of the body. In the same way, some records will last 1 hour with 55 minutes of inactivity while others will last 15 minutes of intense activity because the person was exercising. Obviously, a lot of bias would be introduce if nothing is structured. However, one of the objective of this work is to produce a computer-base system which could score and control the tasks in a normalized way regardless of the specialist or the institution.

As seen in section 1.8, all the techniques used to measure the psychomotor disturbances were employing short and predefined tasks. For examples, the symbol digit test, the reaction test, the balance test or the drawing tasks. Moreover, we also analyzed, in Section

3.1, a system evaluating the effect of a treatment for Parkinson by using motion capture with short tasks. In this context, we decided to employ a similar approach by using a set of simple short predefined tasks for structuring the data. Of course these tasks will be the same for everyone which will allow to perform a standardized comparison.

4.3 Recognize the gestures

The computer-based systems for psychological assessments brought real beneficial assets. Such a system makes it possible to score the subjects and to control the assessments in a completely normalized way, regardless of the patients, the specialists and even the institutions. Similarly the stimuli can be presented in a standardized manner. Moreover, a computer system can collect accurate spatial-temporal data in large quantities and then explore them with recognized disciplines as motion capture, database engineering and machine learning. Moreover the primary use of Kinect in video games is to recognize movements. So, the Kinect will be used to recognize the short predefined tasks.

The problem of recognition is a general problem in motion capture [43]. The recognition is usually the activity of classifying the captured motion as one of several types of actions where the objective is to recognize actions such as walking, carrying objects, removing and placing objects, pointing, gestures for control and even very specific tasks as ballet dance step. Traditionally two different paradigms exist, the static recognition and the dynamic recognition.

The **static recognition** [43] is concerned with spatial data but one frame at a time. Indeed in this approach, we usually compare previously stored information with the current frame. So, the main goal of static recognition is to recognize different postures as standing, sitting, putting the hands over the head. Unfortunately, it's not convenient in the case as we want to detect a continuous gesture.

The **dynamic recognition** [43] uses temporal characteristics during the recognition. The objective is to recognize continuous actions such as walking, moving an object, waving and even very specific tasks as ballet dance steps. Low-level dynamic recognition is usually based on spatio-temporal data without much processing. It consists mainly in raw data as the depth map provided by the Kinect. Whereas more high-level dynamic recognition is usually based on the limbs with the joints position and orientation. Obviously, we will work with high-level dynamic recognition.

4.3.1 Heuristic recognition of the gestures

In the heuristic recognition approach, a gesture is considered as a coding problem based on condition. For example, a task which consists to put the hands over the head: “If the y axis of the left hand and the y axis of the right hand are simultaneously above the y axis of the head”. If the condition is satisfied, the gesture is considered as being correctly executed. So, it is really easy and quick to create simple gestures. However, to be more precise at this stage, this is not really gestures that are recognized but rather **posture** as the recognition is only static and not dynamic.

But this approach can also be extended to recognize much complex gesture. So, a **gesture** is viewed as a series of simple postures in which a final state must be reached. The concept of state machine, from the formal language theory, can be used to represent this approach [41]. As a reminder, a state machine, is a theoretical model which is able to recognize one and only one language. By recognizing a language, we mean that the automate will determine for any word written with the symbols from the alphabet of the language, if this word is part of the language.

- The **alphabet** $\sum$ is a finite set of postures.
- The gesture can be compared to a **word** x defined over an alphabet $\sum$, which is an ordered sequence, finite and potentially empty, of symbols from $\sum$

Our automate is therefore, composed of :

- An input alphabet $\sum$
- A set of finite states Q
- A state $q_0 \in Q$ which is the initial state of the automate
- A set of states $F \subseteq Q$ which are described as final states
- A list of possible transitions Δ which specifies, for each state, the possible transitions to other states or to themselves. Each transition is associated with one or many symbols from the input alphabet and the empty symbol.

A finite automate accepts a word x, if and only if, a path c can be found as :

- The automate starts in the initial state q_0.
- The automate stops in one of the final state from the set F.
- By concatenating, all the transitions taken by c, the word x can be retrieved.

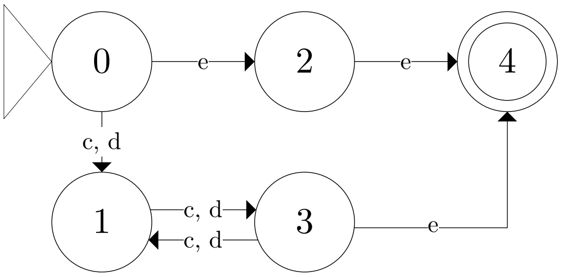


Figure 4.2: Example of a finite automate which represents a gesture.

Figure 4.2 shows a task which includes three different postures: c, d, e. The automate begins in the initial state 0. To move from one state to another, the transition must be respected. Otherwise, the gesture is considered as failed. For example, from the state 0 to 1, only the postures c and d are allowed. If another posture e is executed the gesture is considered as failed. In the end, the automate must be in the final state 4. If the automate ends in the state 0, 1, 2 or 3, the gesture is considered as failed.

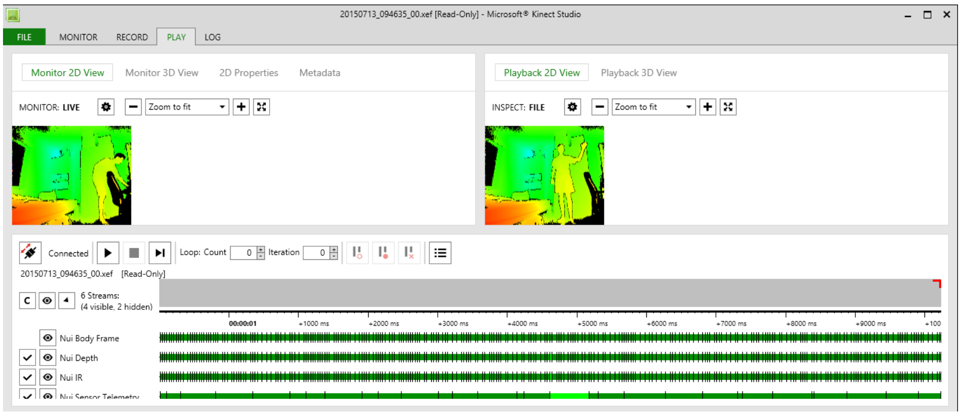
In a more practical way, at each state the system is waiting few seconds for the next one. A transition carried out means that a posture has been correctly performed and the system waits for the next one. If it was a final state, then the gesture will be considered successful. However if after few seconds, the next posture is not executed, the gesture is consider as failed as it is stuck in a non-finite sate. Moreover, if a posture, not authorized by a transition, is executed, the gesture is also consider as failed. Concretely, a transition is not authorized if the posture diverge completely from the current one and the next ones from the transition.

Section 5.1 shows concrete tasks implemented with this method. The prototype used a simplified alternative of this state machine where only one-to-one transitions are allowed. It looks more like a series of postures that must be executed in the right order. In the future, it would be interesting to implement the complete automate. However, for the tasks that we have implemented, this approach was sufficient.

4.3.2 Machine learning recognition of the gestures

However, a same movement can be performed in so many ways. For example, a golf swing. Such a task would be very difficult to code such as all possible manners must be taken into account. So, a more generic approach is needed. Luckily, there is another approach frequently used for recognizing gestures which is the machine leaning approach. In a few words, this technique allows the computer itself to determine what is a task successfully executed. To do this, a large amount of data samples should be stored in order to generate a model which will be able to determine what is the right way to perform the task. Informally, for a given task, more the data are close to the data samples and more there are chances that the task is recognized as properly executed. Though, false positives and false negatives may occur as in all machine learning model.

Nonetheless, Microsoft provides in the SDK of the Kinect, two softwares which allow to recognize gesture by machine learningdans .



20150713_094635_00.xef [Read-Only] - Microsoft® Kinect Studio

MONITOR: LIVE

INSPECT: FILE

Loop: Count 0 Iteration 0

20150713_094635_00.xef [Read-Only]

6 Streams:
(4 visible, 2 hidden)

Figure 4.3: Kinect Studio is a tool, provided by Microsoft, which records the data streams of the Kinect v2.

The former, a software called '**Kinect Studio (KS)**', was built up in order to record and replay scenes that were captured from a Kinect v2. So, the various previously presented data streams can be recorded and then they will be read again in the future for different purposes such as debugging, testing, but also to generate data samples for recognizing movements. To be more precise KS is recording these data streams:

- Infrared stream;
- Depth stream;
- Body frame;
- Audio stream (optionnal).

The latter, a powerful tool called '**Visual Gesture Builder (VGB)**', can recognize movements perfomed in different ways whatever the clothes, the weight, the height and so on. But, as this approach are using **machine learning**, a lot of data must be collected in order to train a model capable of recognizing a movement. Two different of algorithms are used by this software. The former is using *AdaBoost* to recognize a posture while the latter is using the *Random Forest* algorithm to recognize a continuous gesture. Both algorithms are using meta-parameters. These meta-parameters are generally specific to the algorithm used, although some of them are common as:

Input	**Type**	**Description**
Used hands data	Boolean	If true, the algorithm uses the different state of the hands seen in Section 3.6. However even if false, the hands are used but without their states.
Ignore left arm	Boolean	If true, the features of the left arm are not used. So the elbow left, the wrist left and the hand left are omitted.
Ignore right arm	Boolean	If true, the features of the right arm are not used. So the elbow right, the wrist right and the hand right are omitted.
Ignore lower body	Boolean	If true, the features of the lower body are ignored. So the knees, the ankles and the feet are omitted.
Duplicate and mirror data during training	Boolean	Can be useful in the case of a where the side of the gesture doesn't matter. For example, for a punch which can be executed with the right arm, but also with the left arm. So, the variable can be set to true. However if the side is important, the option must be set to false.

Table 4.1: Meta-parameters used by both of algorithms to generate a model.

Obviously, these meta-parameters try to optimize the algorithm. In the case of the meta-parameters below, certain parts of the body can be ignored in order to focus only on the important parts. For example, if a gesture is recognized only by using the arms, it would be prejudicial to introduce bias by observing also the legs. Indeed, more there are features and more records are required to prevent overfitting. For example, some false negative could be introduced by some patient moving an unimportant part during the gesture.

Therefore, this tool is able to recognize postures by using static recognition with the AdaBoost algorithm [8]. In this manner, some data about the movement has been captured by Kinect Studio and then they are used as training data to build a model able to recognize the posture. For generating a model, the algorithm takes as inputs:

Input	Type	Description
Accuracy level	Real	Range between 0 and 1. The value precises the accuracy over the training set which must be reached.
Number of classifiers of the model	Integer	The number of the strongest classifiers which will be used by the model.
Filter results	Boolean	Simple sliding window of N frames. These N frames are sums up and then compare against a threshold value. The user can use the filter already provides by indicating true. Otherwise, the user must set manually the meta-parameters.

Table 4.2: General meta-parameters either in automatic or manual mode.

The other meta-parameters are already parametrized automatically to provide the best meta-parameters. So, Microsoft provides a filter which is a simple sliding window of N frames. These N frames are sums up and then compare against a threshold value. Then the user, can let the algorithm find automatically the best parameters minimizing the false positives and false negatives:

Input	Type	Description
Auto find best filtering params	Boolean	If true, the algorithm find automatically the best parameters minimizing the false positives and false negatives. Otherwise, the user must specified manually the number of frames to filter and the threshold.
Weight of false positives during auto find	Real	By default false positive and false negative have a weight of 0.5. The ratio can be changed by updating this value.

Table 4.3: Meta-parameters specific to the automatic mode.

Moreover, it is also possible to configure manually the filter by choosing the number of frames to sum up and set a treshold where lower value bring a risk of increasing false positives:

Input	Type	Description
Number of frames to filter	Integer	Number of frames over which to filter.
Threshold	Real	Range Between 0 and 1. Lower values could increase true positives while higher value could decrease false positives. A right balance must be found to manage false positives and false negatives.

Table 4.4: Meta-parameters specific to the manual mode.

Then, when the time comes to train the data, the user must just add all the records in relation with the gesture in the training set. After that, each frame where the person is executing the right posture, must be labeled as true. By default all the other frames are considered as false. When all the frames have been labeled, the model can be generated in a file which can only be used by the SDK of Microsoft.

An example with the gesture " both hands over the head", can be observed in Figure 4.4. The green arrow is showing a record. As a reminder, the training can have as many records as desired. The bottom of the figure is showing, the labeling which has been made on all the frames. The frames with a blue bar on the top are true while the frames with a blue bar on the bottom are false. From this, our model can be generated and then be used to recognize that the hands are effectively above the head.

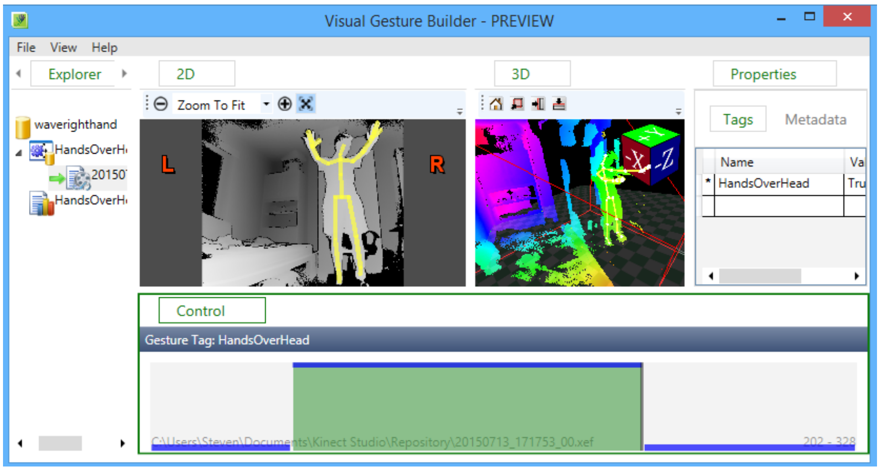


Figure 4.4: Visual Gesture Builder is a software using records from Kinect Studio to create model able to recognize a gesture.

As in an heuristic approach, it is also possible to recognize complex **gestures** which are a sequence of postures. For this purpose, the *random forest* algorithm must be used. Compared to AdaBoost, the meta-parameters are essentially different:

Input	**Type**	**Description**
Number of trees	Integer	Determines the number of trees in a forest. More the value is large and more the model give better results but at the cost of more memory, more CPU and more disk space. Further details are given in Section 2.11.
Maximum tree depth	Integer	Maximal complexity of decision trees in the forest. Complex gestures may need larger value to be more precise.
Cluster threshold	Real	Lower value means a more localized search which can be more accurate for precise and small movements but can be awful if no result is found in this local neighborhood. According to the gesture, larger value may be more appropriate.
Weight factor	Real	The weight of the result of the previous frame. This frame will affect the correct one.
Use rotation variant feature	Real	If true use the orientation described in Section 3.4 else try to ignore these features as much as possible..

Table 4.5: Meta-parameters for the random forest en Visual Gesture Builder.

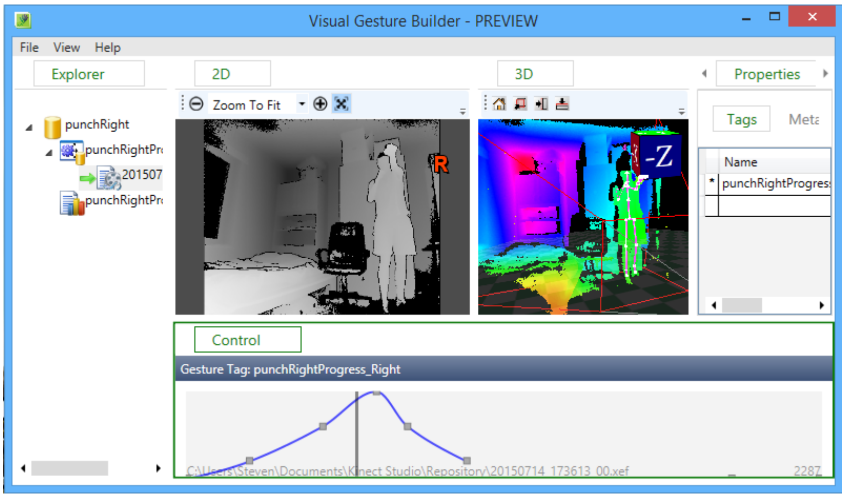

Visual Gesture Builder - PREVIEW
File
View
Help
Explorer
2D
3D
Properties
Zoom To Fit
punchRight
punchRightPr
201507
punchRightPr
R
-Z
Tags
Meta
Name
punchRightProgres
Control
Gesture Tag: punchRightProgress_Right
C:\Users\Steven\Documents\Kinect Studio\Repository\20150714_173613_00.xef
2287

Figure 4.5: Visual Gesture Builder can be used to classify continuous gesture as a punch.

In this case, the indicator concerning the success of the gestures is a real value from 0 to 1 which shows the progress of the gesture while for the postures, the indicator was a simple boolean. Figure 4.5 shows the creation of a punch with the right hand. Each point of the curve represents a posture. In this way, the first two postures must be performed to get the higher value of the curve which is the third point. Then the user gets back to a normal state. It should be noted that a value between 0 and 1 is given at any time, even if the person is not doing the gesture as the system always gives its confidence about the prediction. So, Microsoft recommends, to use AdaBoost to recognize the first posture of the complex gesture and then activate the continuous recognition. Once the gesture has been executed, the continuous recognition can be deactivated for the benefit of AdaBoost.

4.3.3 Heuristic recognition vs. machine learning recognition

At this point, one may wonder what is the best recognition between the best heuristic recognition and the machine learning recognition. We decided to use both approaches according to the complexity of gestures.

Even if the machine learning seems more powerful, the heuristic can be quite useful for very simple task. Actually, it is not necessary to gather a lot of data to train a model in order to prevent bias when the task consists to put the hands over the head. Indeed, the problem is really easy to code. In other cases, some gestures may be too complex to code with an heuristic approach as there are many ways to perform them. So, the machine learning approach is the perfect answer in this case.

By a lot of testing, we noticed that the random forest implemented to recognize complex gestures have some difficulties with gesture which repeats a same posture more than once. For example, the gesture “wave the hand right” met this difficulties as the gesture consisted in repeating three times a same movement because the same posture will have as result 0.33 then 0.66 and finally 1. So the algorithm gets confused and is not able to detect at which stage the gesture is. Otherwise the recognition is pretty accurate.

By the way, we imagine a solution to bypass this problem. The solution is quite natural seeing the different elements:

- The heuristic recognition is able to recognize the postures really easily.
- The machine learning recognition of VGB using AdaBoost is pretty accurate for the postures and can be invariant to the subject. However it requires training.
- The behavior of the heuristic approach can be extended by using the concept of finite automate for recognizing complex gestures. In reality, a complex gesture is a sequence of simple posture. However, this method is not enough invariant.
- Complex gestures can also be recognized by machine learning. It is recommended, to activate the continuous recognition after that the first posture has been recognized by AdaBoost. However, the algorithm performs poorly with repeated postures in a same gesture.

So, the idea is to mix both approaches by partitioning the complex gesture as in the heuristic recognition. The idea is to use a finite automate, where each state will be a random forest recognizing a part of the gesture. The first posture will be detected by AdaBoost in order to activate the continuous recognition relative to the part of the gesture being executed. Obviously, simple parts of a gesture, are far easier to recognize accurately than the complete gesture. However this solution have two major problems. Firstly, a lot of machine learning models must be trained. For instance, a gesture with three parts will be composed of three AdaBoost models and three random forest models which is a lot of work. But to balance this problem, the solution does not need more training samples. They must just be partitioned. Secondly, for recognizing a gesture, all models must perform accurately. If one of the model gives a false positive or a false negative, the complete gesture is biased. However a too complex gesture implemented by only one model will perform poorly. So, a system capable of recognizing gestures with an heuristic approach and with a machine learning approach was built.

4.4 Data about a session

As a patient could execute some gestures in a row, we decided to give the possibility to the user to create a session which will be composed of a series of tasks. The order of the gestures can be chosen.

4.5 Data about a task

Without great surprises, a label and a description are stored for the task. Moreover, a user can precise which joints, must be observed by the system during the realization of the task. For example, one may decides that is not useful to store the data about the legs during a drawing task. We can also easily get some relevant variables for the entire task because of our recognition which labels different measurements in date and time when:

1. The task is started;
2. The first posture of the successful task has been performed;
3. The task has been completely executed (if failed, we measure 2 again);
4. The tasks is stopped.

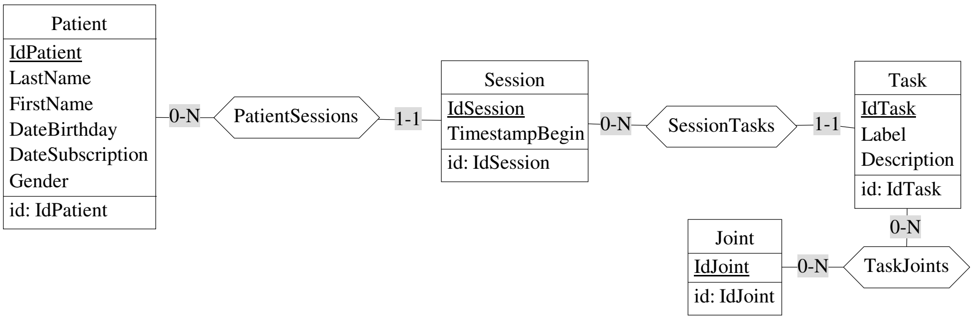


Figure 4.6: General representation of a session

4.6 Data about a frame

As seen in Section 3.3, the most amazing Kinect capability is its ability to track joints of the body. So, instead of working with raw data, it will be much easier and efficient to work with the skeleton data than trying to realize a similar system to detect the body. As a reminder, the realization of such a system took years and an effort of training 1 million records. Therefore when a patient is observed by the Kinect, a new entry is created for each frame and for each joint. As a reminder a frame can be measured in time with a precision of 100 nanoseconds (1 nanosecond = 10^{-6} milliseconds) and each joint can be identified easily with a *name* in order to retrieve the *tracking state* and the *position* of the joint in 3D, but also the *rotation* in a quaternion.

Moreover, at each frame, an array of some events which has been selected as pertinent are registered. As a reminder, we choose to record both of the events concerning the eyes closure and the jaw opening. We also record the measure which evaluates if the subject is looking at the camera. In this context, pertinent means that the results given by these events are appropriate to detect neuropsychomotor retardation, but also that they are the more reliable. This reliability can be explained by the fact that these events are focusing on larger surfaces than the others. As discussed in Section 3.6.3, a part of the body must have a surface *large enough* or *close enough* to be recognized more accurately. So, for each frame, and for each of these four events, the threshold given by the Kinect is recorded. As a reminder, this threshold is a value varying from 0 to 1 which represents the progress of an event. For example, an eye completely open will get a result of 0 while an eye completely closed will have a result of 1. The recognition is continuous, so an eye half closed will have a value of approximately 0.5

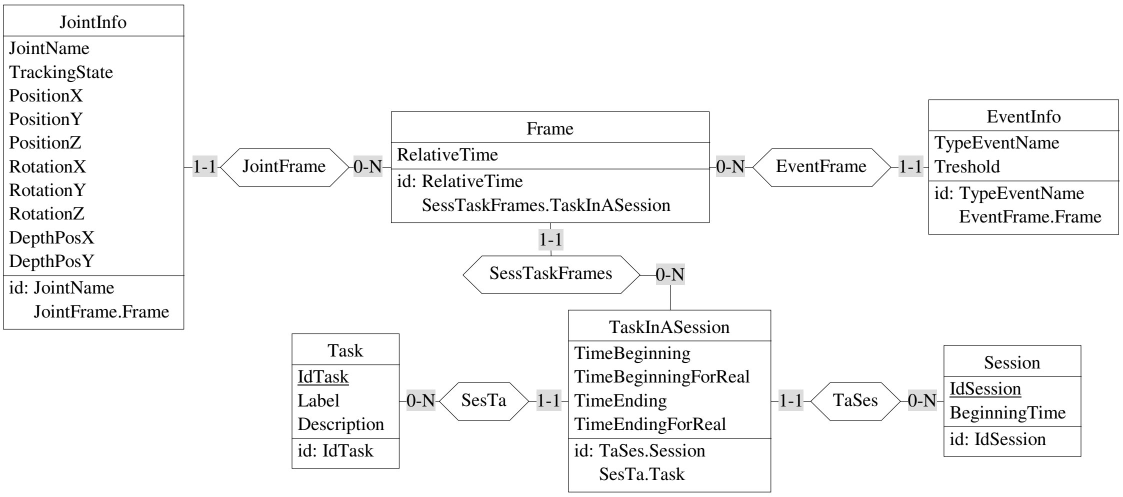


Figure 4.7: Representation of the data recorded by the system during a session.

A representation of the data recorded during a session can be visualized in Figure 4.7. This representation summarizes the different concepts discussed in sections: 4.4, 4.5 and 4.6. Only a part of the schema is represented but in the real database, this schema is merged with all the others presented but also with more technical aspects. However, the complete physical schema is not relevant to expose in the context of this work.

Chapter 5

Experiments

This chapter aims to explain how the experiments have been conducted for collecting the data. Then, all the several approaches for testing our prototype will be explained as a lot of testing are required before using real patients. We described how the different simulations of slowing have been executed, but also how we recorded 14 subjects from a general population before and after alcohol consumption. Then we will also speak about the approval of this study by an ethics committee.

5.1 Prototype for data collection

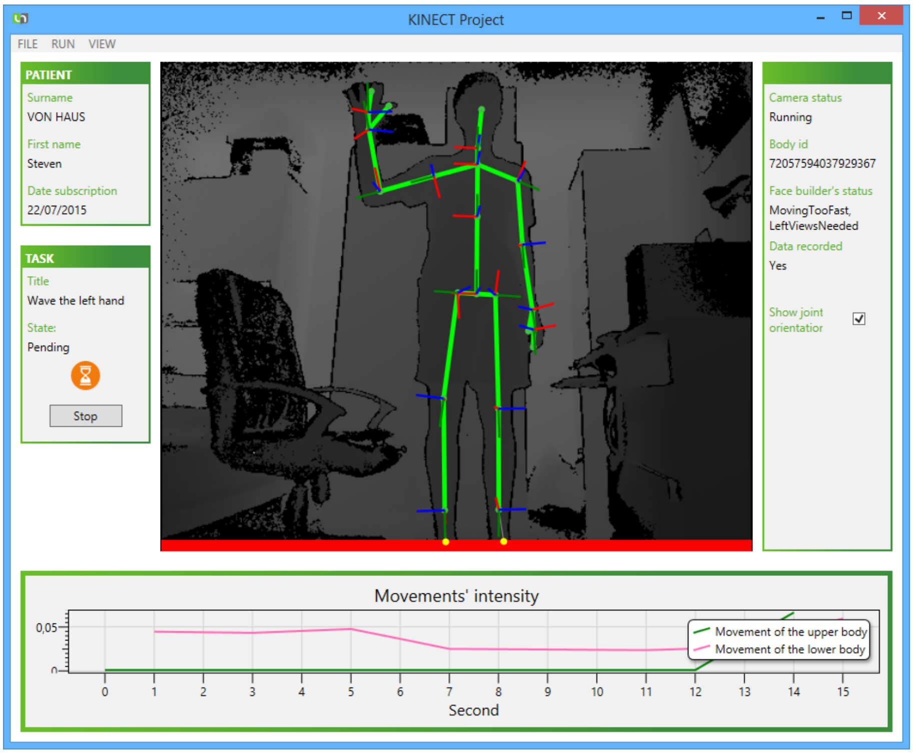
Many details of implementation were already described in Chapter 4. In this section, some relevant details of the implementation will be described in order to understand the methodology. Then, the next chapters will also narrate how we computed and chose the best features among the data collected. Then we will describe how we built our classifier and finally we will give the results. However, the main objective of this document is rather do describe the reflexion and the methodology than the software implementation. Nonetheless, further details that we recommend to consult, are also available in the appendixes.

5.1.1 Monitor the information

The prototype used the BGRA representation. At each frame, an array of 8294400 bytes is used to draw the image seen in Figure 3.6. Indeed, each pixel of the image is represented by 4 bytes, with a resolution of 1920 x 1080. Then, this image is refreshed at each frame (approximately 30 times per second) which gives a real time video. We also provided the depth frame source, the infrared frame source and the body depth frame source that can be showed at any moment. The displayed video can be chosen. Further details are available in the Chapter 3. These videos are not recorded. However, they allow to be aware of what the camera sees in order to avoid a problem during the capture. In this context, the specialist could ask the patient to position correctly before starting a session.

As seen in Figure 5.1, we chose to represent the body by superposing an image over the first video. The body is also refreshed, 30 times per second. The system draw the well tracked joints in green with limbs in green to indicate to the specialist that the body is tracked with a good confidence. Otherwise, as visible in Figure 5.1 the system draws the joints in yellow and the limbs in gray, to indicate that the data actually captured are not completely reliable. We also draw red line along the edge of the screen if a body part exceeds the limits of the camera's view.

The user can choose to show the joint orientation by checking the right box. In this case, the quaternion provided by the Kinect to represent a joint orientation is converted into Euler angles as discussed in section 3.6.2. This more intuitive representation is displayed on the skeleton, where the pivot point is the skeleton joint concerned. The green line express the y-axis The blue line represents the z-axis and the red line is the x-axis.



PATIENT

Surname
VON HAUS

First name
Steven

Date subscription
22/07/2015

TASK

Title
Wave the left hand

State:
Pending

Stop

Camera status
Running

Body id
72057594037929367

Face builder's status
MovingTooFast,
LeftViewsNeeded

Data recorded
Yes

Show joint orientatior ☑

Figure 5.1: Representation of a skeleton with inferred points.

The prototype provides all the tools to register a new patient on the database. All the data recorded for a patient were already introduced in Chapter 4. Then, this patient can be loaded in order to execute a session including of a series of tasks. The composition of the session is represented in Figure 5.1 where the order of the tasks in the session can be chosen. So, on the left side of the screen, appears the title of the task but also its state and a logo which represents the state. At the beginning of the task the state is “Pending” and the logo represents an hourglass in orange. Then, when the task has been executed, the state changes in “Succeeded” and the logo looks like a “V” in green. So the user can stop to execute movements. After 5 seconds of observation, the specialist can choose to record with a comment but also to repeat the task if a problem has occurred or simply cancel it.

As explained in the Chapter 3.12, describing the face tracking in high definition, it is possible to get more accurate results by generating a mesh model in 3D of the face. Otherwise, an average face is used. So, it is possible for a patient to generate a model in real time. As visible on the right side of the Figure 5.1, the label “Face builder’s status” indicates which image from the face need to be recorded. For example, if the person must look down to the right, the left, the bottom or the top. Once the model is generated, it will be recorded for future sessions.

5.1.2 Gestures implemented

In Section 4.3, two approaches for recognizing the gesture have been described. They will be used to implement the gestures of our prototype collecting the data. The gestures implemented were not chosen at random. The most relevant for detecting the neuropsychiatric disorders have been selected.

Sitting down

The first gesture implemented consists in **sitting down**. We were inspired by the balance test presented in Section 1.9 where the patients get up and sit down, 5 times in a row. As discussed in Section 3.1 Parajuli (in [46]), analyses the changes in posture for predicting a fall. Moreover, this kind of tasks required almost all the joints to be performed Therefore, we think that this task is a great opportunity for analyzing the balance. The task has been implemented by using the machine learning recognition and the idea of the finite automate. Two different postures has been trained with AdaBoost. The first one detects when the subject is seated and the second one detects if the patient is standing. The finite automate representing the gesture begins with the first posture and waits for the second posture to reach a final state. To train the first posture, the meta-parameters were configured to consider all the joints but also to mirror the data in order to generate synthetic training examples because there is no need to focus on a particular side of the body (left or right). We labeled 2952 records with 1792 positive instances and 1160 negative instances in the training set. The learning algorithm generated 7938 weak classifiers and only the 1000 stronger one were kept. The other best meta-parameters were automatically configured by the algorithm. For the purpose of testing the model, a validation set including 552 results was evaluated. As a reminder, these 552 instances have never been seen by the model. Each record has also been labeled in order to compute the error after the evaluation. As a result, the 552 instances were classified correctly. As, the model is in a proprietary format, it is not possible to consider other measures or a k-fold validation.

Getting up

The second posture consists in standing up. The records are the same than in the first posture. However, they were labeled differently. The same meta-parameters were also used. We labeled 2820 records with 962 positive instances and 1858 negative instances in the training set. A validation set including 552 instances gives an accuracy of 100%. Moreover, a second task which consists in **getting up** can be created almost for free by switching the two postures. We can emit some criticisms as the model was trained and tested with one subject. However, the model has shown accurate predictions at runtime during the data collection with 14 different subjects. The machine learning recognition was used for these two tasks because it does not exist a unique way to sit down or to stand up. Therefore, the variations introduced by these models allow to generalize the postures.

Lift the left/right leg

For the next gesture, the subject has to **lift the leg** and maintain balance during 10 seconds. This task is also inspired by the balance test like the previous gestures. In [56], they evaluated the effect of a treatment for Parkinson in analyzing the ability of the patient to maintain balance. It is not exactly the same task. However, it also consists in maintaining balance. The heuristic recognition was used for this task. The implementation is really simple. For accessing, the initial state, the algorithm checks if the value of the foot on the y-axis is higher than the value of the opposite ankle on the y-axis. Then the subject must keep this position during 10 seconds to reach a final state. One default of the heuristic recognition is that the subject can sit down and make the gesture. In this case, the objective of the task is lost. However, the algorithm has worked very well with the 14 different subjects during our experiments. The both approaches can be discussed but the heuristic was implemented because it does not require training examples. If the subject fails, he must retry the gesture. We are aware that this task cannot be accessible for everyone. Therefore, in the future, we would like to score the patient according to the time he has maintained balance. As a result, two task can be created to propose an alternative: “ lift the right leg ” and “lift the left leg “.

Wave the left/right hand

Another gesture was implemented with the heuristic recognition. It consists in repeating large movements from the next to the right with the hand. This gesture is called “ **wave the hand** ”. A better representation is provided by the Figure 5.2 where the gesture is composed of two postures repeated 3 times in a row. For accessing, the next part of the gesture, some conditions must be respected. The simplified implementation looks like:

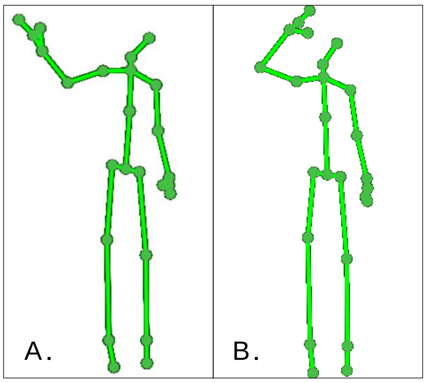


Figure 5.2: The “wave left” task. Step A and B are repeated 3 times in a row.

```
public class WaveRightSegment1 : IStateOfFiniteAutomate
{
  public GesturePartResult CheckGesture(Body body)
  {
    var handRight = body.Joints[JointType.HandRight].Position;
    var elbowRight = body.Joints[JointType.ElbowRight].Position;

    if (handRight.Y > elbowRight.Y + TRESHOLD1)
    {
      if (handRight.X > elbowRight.X + TRESHOLD2)
        return GesturePartResult.Succeed;

      return GesturePartResult.Pausing;
    }

    return GesturePartResult.Fail;
  }
}
```

The first part of the gesture checks if the hand is above the elbow by comparing the value on the y-axis. Otherwise, the gesture fails and returns in the initial state. If the value of the hand is higher than the value of the elbow on the x-axis then the automate goes into the next state else the algorithm waits for the next posture. The user has a limited time that can be parameterized before that the gesture is considered as failed. The implementation of the second part of the gesture looks alike except that the condition for the hand checks if the value of the hand with the threshold added is smallest than the elbow. The implementation above, has presented the gesture for the right arm. Naturally, the same implementation with little changes can be used for the left arm. Therefore, two gestures were implemented. These two gestures aim to evaluate the speed and the amplitude of the movements. For example, we want to test if a melancholic patient shows more slowing than an healthy subject.

Task for tracking the eyes

Section 3.12 shows that the Kinect needs a sufficient depth area in order to make accurate predictions and that it could be a problem for detecting the state of the eye. As a reminder, the state of the eye is represented by a value varying from 0 to 1. More the value tends to 1 and more the eye is closed. During the elaboration of the prototype, we noticed that the Kinect had some difficulties to record the state of the eyes. During the tasks already introduced, the eyes were generally not even detected. Indeed, for detecting the entire body, a distance of 2 meters were appropriate. This distance seems already too high for tracking the eyes. It is necessary to get closer in order to provide a large enough depth area to the Kinect. Therefore, the most convenient way to get the entire body but also the eyes, is to sit down in front of the camera with the face facing it. So, we have added a task where no gestures are expected. The system only tracks the eyes during a period of time. However, facing a camera can seem not very natural and could bring bias because the subject is aware that its eyes are tracked. For example, the Kinect could be fixed near a computer screen or strategically placed in a doctor office to make people forget that they are observed. Since the eyes are not tracked efficiently during the other tasks, they will only be evaluated for this task in this work.

Task for evaluating the orientation

As already discussed in Section 3.6.2, the position is not sufficient to detect all the different movements. Indeed, it is possible for the arm to rotate without any changes in position. For example, while turning a key in a lock. In this context, we also wanted to evaluate the orientation with the prototype. For this purpose, a task where only the arm rotates without any changes in position was created. However, the rotation is really less accurate than the position. Indeed, the values are often oscillating even with no movements. Nevertheless, the value remains a fair approximation in general. For this reason, it was very difficult to create a recognizable task. We tried the two approaches to implement the gesture but without success. The problem is that during 1 second, 25 frames can give good results while 5 frames will give bad predictions. So, our 2 approaches need to be updated to take into account the "bad frames". However, at this stage, there are already a lot of variables and tasks to evaluate. Therefore, we created a task which can be manually started and stopped even if we prefer a system which allows a standardized scoring (see Section 1.9).

To come back to the task, the first posture consists in positioning the elbow, the wrist and the hand parallel to both shoulders with the palm of the hand pointing to the ground. In the second posture, the hand must point to the sky while remaining parallel. The two postures must be repeated 3 times in a row to complete the gesture. With this task, there is almost no changes in position.

5.2 Data collection

Naturally, the prototype must be tested and the proof of concept must be done before using real patients. In this context, for testing in real situations, other solutions was necessary. So, we have looked for other possibilities to test the prototype. In a first time we did experiments in general population, covered by a formal ethic procedure. We also have seen that one of the solution, is to measure the cognitive impairments caused by an alcohol consumption and by the sleep deprivation.

5.2.1 Sleep inertia

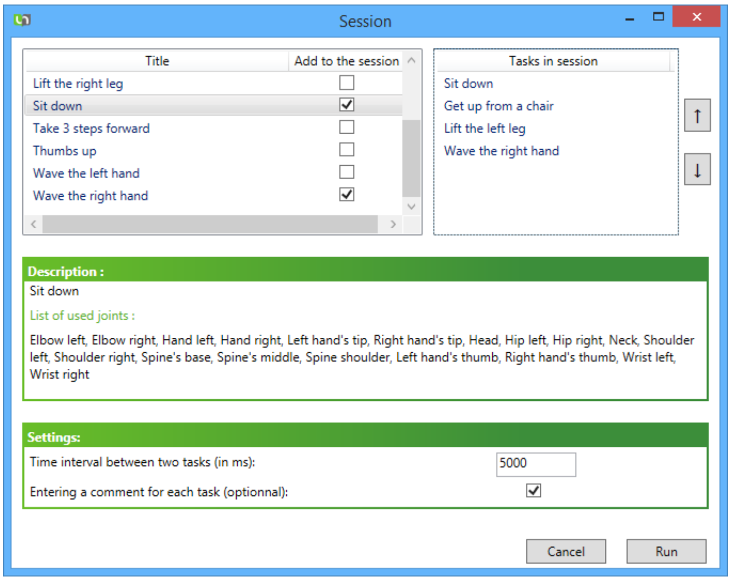
Another idea, to test efficiently the detection of psychomotor impairments by the system is to perform tasks with a condition known as "**sleep inertia**" as a proof of principle. This idea has been used by the Massachusetts Institute of Technology in collaboration with the Harvard Medical School in [27] as a proof of concept for psychomotor impairment detection via finger interactions. This psychomotor effect implies a state of groginess, impaired cognition, reduce motor dexterity but also disorientation. This effect is maximal after being waken up during deep sleep phases. It's also comparable to be sleep deprived for 24 hours.

In order to train an accurate model, a lot of training instances must be recorded in order to explore all the possibilities and so prevent overfitting. More the number of features is large and more we need training instances for inspecting all the features. It is quite obvious that it would be really difficult to measure sleep deprivation as we need a lot of subjects accepting to be awaken abruptly in order to execute our tasks. However, this idea can be kept in mind for the future. Indeed, it exists research centers that conducts sleep research as the Cyclotron Research Center at the University of Liege.

5.2.2 Alcohol consumption

We studied a sample from general population at the beginning and at the end of a dinner, where several persons planned previously to drink alcohol. In addition, the data are only processed statistically and not individually. Furthermore, the ingestion of alcoholic beverages has never been supported. In this context, some subjects did not drink any alcohol and their movements was only classified as healthy. The test has gathered 14 subjects including 9 women and 5 men with an average age of 24 years. Four different tasks where chosen in order to test different parts of the body and the changes in poses. The participation was completely optional and voluntary. All the participants remain free to leave the study at any time without giving any justification. All the results remain confidential.

As we wanted to test the changes in postures, the first two tasks asked to sit and then to get up. Moreover, we aim to test the balance as this is a recurrent exercise during psychomotor tests. Hence, one task consisted in lifting the left leg while keeping balance during 10 seconds. For the last task, it was asked to wave the left hand in order to test the speed of the upper body. In this way, we tried to test a different variety of movements. These tasks has been accomplished before any alcohol absorption at 7pm. Then at 2am, the same tasks was repeated with the same participants. The quantity absorbed was varying from nothing to large quantities. So, we considered as intoxicated a subject with at least a 0.5 % blood alcohol concentration for future classifications. As a result, we recorded 93 tasks.



Title	Add to the session
Lift the right leg	□
Sit down	☑
Take 3 steps forward	□
Thumbs up	□
Wave the left hand	□
Wave the right hand	☑

Tasks in session
Sit down
Get up from a chair
Lift the left leg
Wave the right hand

Description :
Sit down
List of used joints :
Elbow left, Elbow right, Hand left, Hand right, Left hand's tip, Right hand's tip, Head, Hip left, Hip right, Neck, Shoulder left, Shoulder right, Spine's base, Spine's middle, Spine shoulder, Left hand's thumb, Right hand's thumb, Wrist left, Wrist right

Settings:
Time interval between two tasks (in ms): 5000
Entering a comment for each task (optionnal): ☑

Figure 5.3: Session created for the data collection.

During the capture, we got quite unexpected bias. Some subjects considered the tasks as a challenge and try to do their best. For instance, during the task testing the balance, some tried to raise the leg as high as possible. In addition, after alcohol absorption, the effect was not always clearly visible as few subjects was able to focus for a few seconds. However, we have seen in the first chapter that the differences was not high and so it is not always easy to perceive the changes. Another bias, completely unexpected was due to clothing. Indeed, some girls were wearing skirts or dresses and were afraid of making ample movements. For example, during the balance test with the left leg, they were careful and barely raised their legs with slow movements. In the same way, during the tasks asking to sit and to get up, some subjects made an additional movements by crossing or uncrossing their legs. However, we also see correlations. During the task with the left leg, some subjects, after alcohol absorption have to try several times before succeeding.

5.2.3 Simulation of impairments

The last solution was to simply simulate impairments and then test if the algorithm was able to automatically detect them, according to the features given. So, all the gestures described in Section 5.1, have been performed in different ways.

For example, for the task of “waving the left hand”, the main objective was to test the ability of a learned model to predict the right class. So, 45 instances were recorded in four different groups in order to diversify the impairments. In this context, the speed of the movements and the reaction time were varied during motion capture:

a) 12 times with a reaction time particularly *short* and *fast* motions

b) 11 times with a reaction time particularly *short* but with *slow* motions

c) 11 times with a reaction time particularly *long* but with *fast* motions

d) 11 times with a reaction time particularly *long* and *slow* motions.

The other simulations will directly be introduced in Chapter 7 during the evaluation of the different tasks.

Future tests with real patients will be required in order to ensure the performances of the system to learn to recognize the neuropsychiatric diseases. However, this data collection will already allow us to test the ability of the features to highlight the differences between some groups.

Chapter 6

Classification of tasks

This chapter will mainly focus on the classification of tasks but the main objective aims at detecting psychomotor retardations. Therefore, the approach is completely different from the gesture recognition in real time that we have discussed in chapter 4. In the first place, the choice of classifying the different tasks before the patients will be argued. Afterwards, the whole process before creating a model capable of classifying a task will be described as the computation of the features. Then, with the data already collected, the best features which give a representative representation of the movements will be selected. The classes which will be used for classification will also be defined.

6.1 Classify tasks before classifying a patient

The main objective of the project is to classify patients. However, a patient may have a lot of data recorded and these data are not always easily comparable. Indeed, even in one second, thousands of data are recorded. So a patient which has already used the system for few minutes will have approximately some millions of data. Yet these data are arduous to compare between different patients as each patient may have performed different tasks with various length of time. We are not saying that this is impossible but maybe a preliminary stage would be very interesting.

6.2 Task classes

As specific models will be trained to classify tasks, the classes predicted must be defined. A lot of time were spent in thinking about using the diagnosed diseases and the medications as features. But, we were afraid that the predictive algorithm takes a shortcut by always predicting the disease that were diagnosed. For example, if we train a decision tree with 1000 records including 900 healthy patients and 100 patients suffering from melancholic depression. When the algorithm will try to get the best feature, the most discriminative one might be the disease itself. Moreover, at this stage, we are only trying to classify a task.

Instinctively, the first idea is to used the **diseases and the medications as classes** for the tasks. For example, if we want to discriminate an healthy group from a melancholic depression group, two classes would be created. Then, a task will be labeled with either “melancholic depression” or “healthy” during the training by a specialist. Obviously, the classes would be quite intuitive and always pertinent for the classification of the patients according to their disorders. However, creating a class for each disorder can be dangerous. Indeed, as seen in Chapter 1, there are differences among the neuropsychiatric diseases but also similarities. Therefore, some tasks could gave exactly the same result for two classes without real errors. For instance, in the Chapter 1, we have seen that the major depressive disorder group and the chronic fatigue syndrome group showed similarities in [61].

On the other side, **general classes** could be created. These classes could represent variables which aim to measure psychomotor retardation as the cognitive and motor impairments. For examples, a task could have the following classes:

1. Reaction time particularly *short* and *fast* motions (class one) ;
2. Reaction time particularly *short* but with *slow* motions (class two) ;
3. Reaction time particularly *long* but with *fast* motions (class three) ;
4. Reaction time particularly *long* and *slow* motions (class four).

The great particularity of using general classes would be that machine learning algorithms themselves would determined the class related to a disease. For example, melancholic patients would have great chances of belonging to the class four. So, there is no need to create a class for each disorder. Moreover, the classes created could be specific to a task. There will also be less classes due to the generalization. So, the training set will directly give a better distribution among the classes. However, as the classes will be used for classification of the patients, the choice is particularly important for an accurate result. Therefore, a poor choice of classes, will led to a poor classification of the patients. Moreover, they are less representative of the diseases.

We can say that the both approaches have their qualities and their defects. We think that it is necessary to gather a lot of training records among real patients in order to evaluate the accuracy of the approach using diseases and the medication as classes. At this stage, only the second approach can be evaluated. Indeed, during the collection of the data, discussed in Chapter 5, the tasks were executed by a general population in order to test the prototype. Naturally, a proof-of-concept is needed before gathering data among real patients. Therefore, in the following sections of this document, the classification of the tasks will be evaluated with general classes.

6.3 Using Visual Gesture Builder for classifying the tasks

One may think that Visual Gesture Builder can simply be used to classify the tasks. For instance, VGB would be use to resolve the example of classifying a “get up” gesture. However, other variables than those determined by Microsoft cannot be taken into account. For instance, the time taken to initiate a gesture as the time to complete the gesture, are really important variables for determining psychomotor disturbances. Furthermore, considering the different kinds of motion capture, seen in Section 3.1. VGB is more oriented on surveillance for real-time recognition. While our classification will analyses the entire record after the execution.

Indeed, the tool is obviously designed to recognize a task in real time with variations. These variations in speed, height, weight and so on, are applied on purpose. So, these variations prevents from distinguishing the psychomotor retardation as generalization is applied on purpose to prevent overfitting. For instance, VBG will try to recognize a 'get up' gesture with variations and will not detect a slow 'get up' from a quick 'get up' as the objective of the tool is only to estimate its confidence about the realization of the gesture. Indeed, basically, VGB is specialized for the purposes of gaming or for the interactions with an interface as the Kinect for the Xbox One. In conclusion, VGB is not adequate for the purpose of our classification. So, specifics models must be created for detecting psychomotor retardations. However, VGB can be used to recognize the task in real time, as described in Section 4.3.2 for further analyses of the data recorded with our specific models.

6.4 Features for tasks' classification

Even before choosing the machine learning algorithm to use, an important stage is to analyze and define the features which will be used. We know that the Kinect is able to provide accurately the positions and the orientation of 25 joints. It would be interesting to work with these positions at first before exploring the depth image or the infrared image which are more complex to use.

The characteristics which will allow to determine if a patient is suffering from psychomotor retardations must be defined. Obviously, assuming that the features have enough information about psychomotor retardations, is necessary to get accurate results. A very basic method recommended by Andrew Ng [44] to test this hypothesis is to ask the following question: "Can human experts look at the features x and confidently predict the value of y?". Indeed, such hypothesis demands an understanding of what is the sort of problem and how to resolve it. So, each feature will be defined by asking this simple question.

6.5 First approach for measuring the movements

Before implementing, a complete and complex method, we wanted to test the performance of the camera for measuring the intensity of movements. In this context, an approach as simple as possible has been built to test the ability of the Kinect the intensity of a movement according to its speed and its amplitude. So, for each joint, the movement between two frames is computed. In this case, a movement means the translation from the vector at the frame n to the vector at the frame $n + 1$. For the simple approach, the euclidean distance is computed as follow:

$$\sqrt{((x_0 - x_1)^2 + (y_0 - y_1)^2 + (z_0 - z_1)^2} \tag{6.1}$$

Therefore, for each joint and for each frame, an euclidean distance is computed. Then, the standard deviation is used to compute the dispersion among the mean of the data set. So, the small values will be clustered around the average while larger values mean that a movement with a larger intensity has been performed. This measure can be simply computed as follow, where x_i is represented by an euclidian distance between a joint at a frame i and the frame $i - 1$:

$$\sigma = \sqrt{\sum_{i=0}^{n} \frac{(x_i - \overline{x})^2}{n}} \tag{6.2}$$

where

$$\overline{x} = \sum_{i=0}^{n} \frac{x_i}{n} \tag{6.3}$$

However, at this stage, the method computes the intensity for the entire record. But this measure can easily sampled per second. So, approximately 30 values will be used for each sample, which make possible to observe the evolution through time.

In order to verify the relevance of this approach, some tests have been executed from a prototype. By calibrating these data over time on a graph, it is possible to observe easily the intensity of the movements. Figure 6.1 shows the result of the prototype with the right hand. During, the record of the data, this joint has moved with variations in speed and in amplitude. From the 4th second to the 8th second, no movement was made. Then from the 8 th to the 12nd second, a slow movement has been executed. While during the 12nd and the 13th a very quick movement was performed. And so on.

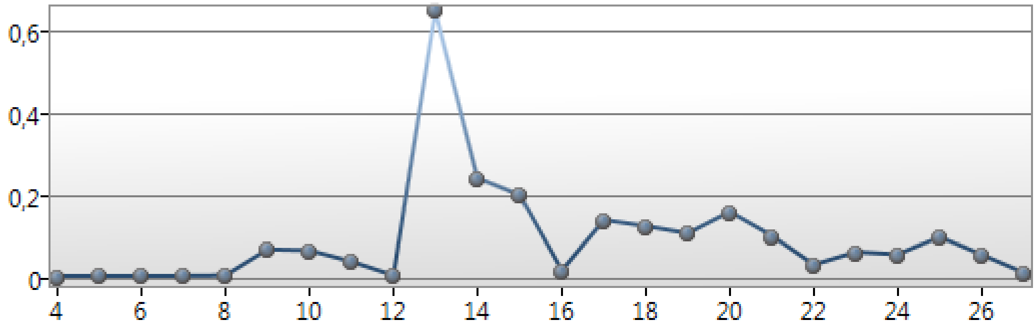


Figure 6.1: Result with our first prototype quantifying the movements. Vertical axis shows the standard deviation while the horizontal axis shows the time elapsed in second.

The results observed are pretty close to the reality. We could say that this approach offer an effective way for computing the quantification. However, only the position is taken into account and not the orientation. So if a person execute many rotating movements, as twisting his arm, no movement will be detected as the system only detect the translations in space. The approach shows that regrouping frames allow the use of statistical functions as the standard deviation. Furthermore, the *dispersion around the mean* gives a quite good approximation about the intensity of movements. However, the solution is still far from being complete. The main objective of this method was fulfilled as the goal was, to get quickly, representative results of some movements performed in front of the camera, even if the result was not precise. Indeed, this method is a good indicator to demonstrate that the camera is completely able to quantify movements.

6.6 Joint positions for building features

As a reminder, the **position** was stored in a 3-dimensional vector for each frame. In addition, we have seen that the euclidean distance can be used to measure the intensity of the movements. So, *for each frame*, except the first one, the speed is computed *for each axis*, by subtracting the distance of the previous frame from the current frame. As a reminder, the **speed** is the distance moved per unit of time. The unit of time will be the duration between two frames which is the more precise scale given by the Kinect. Therefore, the speed is expressed in distance per frame. The *current frame* is denoted by i. The speed can be computed as follows:

$$s_i = \sqrt{((x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 + (z_i - z_{i-1})^2} \tag{6.4}$$

On the other hand, the **velocity** has two components: the magnitude (speed) and the direction. The magnitude considers only the length of a trait between two points. A unit vector is used to define the direction from an origin point. The unit vector aims to normalize any vector in a vector with a length of 1 and can be computed as follows:

$$\widehat{v} = \frac{v}{\text{length}(v)} \tag{6.5}$$

In our case, v is the translation vector given by:

$$(x_i - x_{i-1}, y_i - y_{i-1}, z_i - z_{i-1}) \tag{6.6}$$

For example, if $v_0 = (1, 2, 3)$ and $v_1 = (3, 2, 1)$ then the translation vector $v = (2, 0, -2)$ and $s = 2.83$. Then, the translation vector is normalized and give the vector $(\frac{2}{2.83}, 0, \frac{-2}{2.83})$ which is the direction. This method aims to provide a standardized way to reach a point from any origin point by giving two components isolated which represents the speed and the direction.

Then the **acceleration** which is the rate of change of speed is computed. The result is a positive or a negative value. For example, if a car reach 25 m/s in 5 seconds, its acceleration is equal to 5 m/s^2 because each second, the car speeds up of 5 meters per second per second. In our context, the unit time remains a frame and the acceleration is given by subtracting the speed at the frame i from the speed at the frame $i - 1$. Therefore, in this context the acceleration is expressed in distance per frame per frame.

$$a_i = s_i - s_{i-1} \tag{6.7}$$

The method was also *extended for each axis*. So the speed and the acceleration will be computed for x, y and z. We don't find the velocity useful in this case, as the direction will always be 1 or -1 on a unique axis.

6.7 Joint orientation as features

As already mentioned, the position alone does not allow to capture all the movements as a joint can stay at the exact some position from the camera's perspective while rotating. That is why the orientation will also be used as a feature for each joint. The orientation has the great particularity of being invariant to the subject and to the position in the scene. Indeed, the subject will have the same orientation values regardless of the camera's view and the subject morphology.

As a reminder, during the recording, a quaternion is provided to represent the joint orientation. However, we choose to work with Euler angles, as mentioned in Section 3.6 because the values are much more intuitive to understand. Moreover, as a dimension is dropped, the number of features will be less consequent what is quite satisfying. As discussed in Chapter 2, too many features bring a serious risk of overfitting and underfitting.

We adopted approximately the same approach as for the joint position. So, we also computed the speed, the velocity and the acceleration for each frame.

6.8 Summarize the features

At this point, one record of a task is represented by a $m * n$ matrix X where m is the number of frames of the record and n is the number of features computed with the position and the orientation.

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & \dots & x_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m,1} & x_{m,2} & \dots & x_{m,n} \end{bmatrix} \tag{6.8}$$

However, it is impossible to predict in how many frames a task will be executed. For instance, a task performed in only 10 seconds has approximately 300 frames for 25 joints which would give 7500 multiply by the number of features. So, the dimension of a matrix X will be different for each record because two same tasks can have a different number of frames as the task can be performed in 5, 10, 20 or an infinity of seconds.

But a machine learning model has generally a specific number of features. Indeed, a model with 25 features will be trained completely differently compared to a model with 50 features even if the 25 features are among them. Therefore, the record will be preprocessed to have a fixed length before being sent to a machine learning algorithm. The entire matrix X will be converted into a row vector with the help of statistic functions as the *average* (V_A), the *standard deviation*(V_S) and the *median* (V_M). Each of these statistical functions are applied on each column of the matrix X and will give a vector of size n.

The vector for the average:

$$V_{A_j} = \frac{1}{m} \sum_{i=0}^{m} X_{i,j} \tag{6.9}$$

The vector for the standard deviation:

$$V_{S_j} = \frac{1}{m} \sum_{i=0}^{m} \sqrt{(X_{i,j} - V_{A_j})^2} \tag{6.10}$$

The vector for the median:

$$V_{M_j} = \text{if } (m \bmod 2 \neq 0) \text{ then } (S_{m/2}) \text{ else } (\frac{1}{2} * (S_{m/2} + (S_{(m/2)-1})) \tag{6.11}$$

where S is the sorted vector representing the column j of the matrix X.

The result is a vector F which is the concatenation of the other vectors:

$$F = V_A \oplus V_S \oplus V_M \tag{6.12}$$

This method can be compared to the approach used in [46] to prevent elderly people to fall by using a Kinect. They reduce the data dimension by storing only a percentage of change which represents the shift between the current position and the position from the previous frame in order to consider each frame and to become independent of data's dimension. Indeed, the fact that the speed, the velocity and the acceleration are always dependent from the previous frame has to be enlightened. Each frame is relative to the previous one. Even the result of the last frame is altered by the first frame.

Moreover they [46] also stated that ignoring the z-axis was interesting in order to reduce de number of features as their movements do not rely on the depth of field. However, in the future, some tasks may require to move forward or backward a body part. Additionally, in many tasks, some movements may go a little backward or forward and we think that detecting psychomotor disturbance and all the nuances to discriminate among the neuropsychiatric disease is an exercise which requires a considerable accuracy. So, ignoring completely an axis among the three is maybe not recommended.

6.9 Data sampling

However by applying these functions, only a single value remains for each feature, for the whole task. Such a loss of details is not acceptable. So the records will be sampled in different parts. As the recording includes the motion capture *before the beginning of the execution* of the task. In this context, this part before the task, will be isolated in a single sample. In the same way, the motion capture continues five seconds *after the execution* of the task. The part will also be isolated in a sample. So the most important part containing only the execution of the movement is completely detached. In addition, the sampling can also be employed on this last part. As a consequence, more details will be available for further analyses. But as already mentioned, too many features can be terrible. So, it is more desirable to have a limited number of sample. In consequence, the right balance between the details and the number of samples generated must be found.

As already discussed, Sabbe [58] observes a severe limitation in the speed of execution and in the capacity to accelerate the action during drawing tasks. More generally, melancholic patients show particular difficulties initiating movements particularly. For these reasons, a number of four samples has been chosen. However, the system can be parameterized at any moment if we find out that an other number of sample is more valuable.

6.10 Feature normalization

The feature normalization is a recurrent step in machine learning. It consists in rescaling the features so that they have the same distribution around a center. Indeed, the value of two different features can have completely disparate results. We found out that the collected values were varying from 10^{-5} to 10^{5}. However, some optimization algorithms perform poorly with features completely disparate. A feature with high values can dominate an other feature, with the same importance, because of very smaller values. Without going into the technicalities, the objective functions and the learning will operate more efficiently because of the normalization [7]. The formula for normalizing a feature is quite simple:

$$\frac{f_i - \bar{f}}{\sqrt{\sum_{j=0}^{n} \frac{(f_j - \bar{f})^2}{n}}} \tag{6.13}$$

Where f is a n-dimensional vector representing all the values for a feature. In a nutshell, the average value of the feature is subtracted from the i^{th} value of the feature and then divided by the standard deviation.

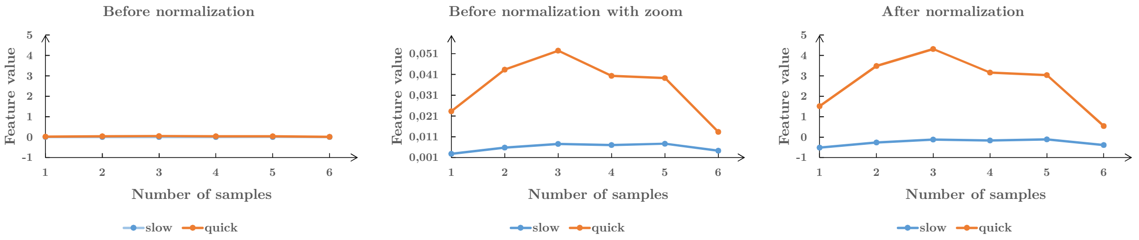


Figure 6.2: Representation of a feature before and after being normalized.

After such transformations, we find useful to evaluate some of the features generated by the system in order to ensure that they stay pertinent. Moreover, at this point, we have yet too many features. It would be interesting to remove the least helpful. For this purpose, the task of "waving the left hand" with impairments simulated has been used (see Section 5.2.3). In this case, only two classes are considered: slow motion (22 instances) and fast motion (23 instances). In Figure 6.2, the curves are impossible to distinguish before a zoom with the interval varying from 0 to 0.05. After normalization, we can see that the trend of the curve remains exactly the same.

6.11 Feature evaluation

Except feature normalization, other transformations has been applied on the training set to obtain normalized features. So, it would be really interesting to investigate all the features generated in order to find the most representative of the reality. For instance, we could reduce the number of features if we discover that some features are not pertinent or that others are more appropriate to describe the same evidences in order to improve the future classifiers. So, the same 45 tasks will be used to evaluate the features.

6.11.1 Joint position evaluation

For recall, the position of the joint is stored a 3-dimensional vector. Section 6.6 showed us how to compute the changes between two frames and then resume all the frames in a vector.

The **average of the position** follows the same trend for the three axis. So, the differences are less visible even if the interval values are not the same between the both kind of tasks because the movements was larger. However this difference can also be explained by the fact that the values are changing according to the camera's perspective or with the morphology of the subject. These feature are not the most rational to use as they are not invariant. A movement can be exactly the same and give different results. So, we decided to remove this feature.

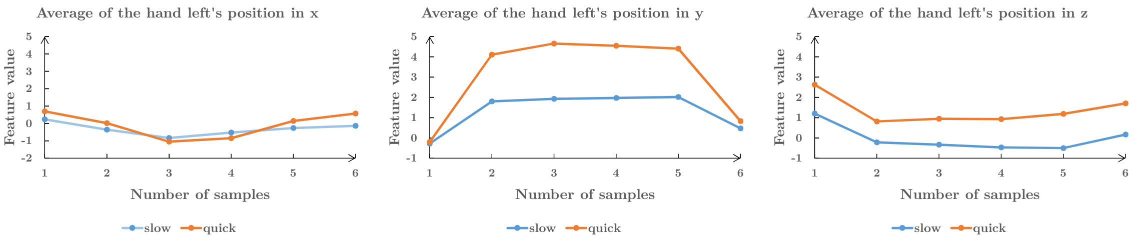


Figure 6.3: Average position values for the left hand during the task "Wave right' performed in two different ways.

The **dispersion of the position** (Figure 6.4), already gives better results but they are not really satisfying. On the x axis, we can see that the dispersion is higher during the quick tasks but not that much. The dispersion is even higher in the slow task during the 5^{th} part. The other axis gives approximately the same results as these axis are less stimulated during the task except for the last part, because the hand gets back to the normal position. However, little differences are visible although the movement has been done in two different ways. So, if we want to select only the best feature maybe it is not the best choice.

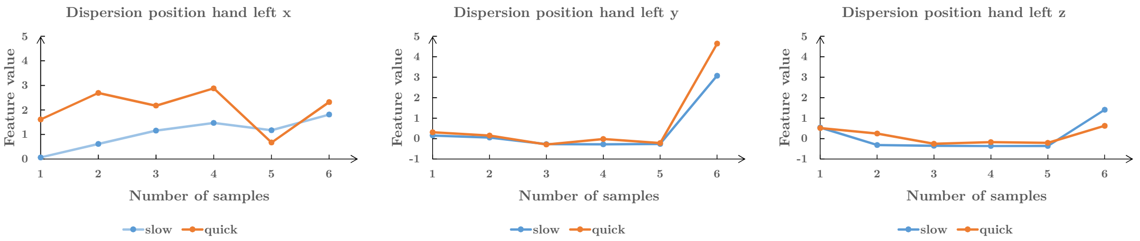


Figure 6.4: Dispersion for the position values of the left hand during the task "Wave right' performed in two different ways.

Figure 6.5 shows the **average speed** for the three different axis of the left hand during the task. The "quick" tasks have graphs which are fluctuating as the movements are really prompt. While, in the "slow" tasks, the curve remains the same as the movement are actually continuous with a constant speed. As expected, the "quick" tasks have a speed higher. Particularly for the x axis, where the speed can grow until 5. This is due to the fact that, the gesture consists in moving the hand from the right to the left 3 times. The effect is less pronounced with the y axis as the hand is not moving that much from bottom to up except at the beginning of the gesture. Indeed, for the first sample, the value is higher because for initiating the movement, the hand must goes approximately at the height of the shoulders. The z axis is barely varying, as the hand are not really getting closer to the camera. So, we can conclude that the feature is really representative of the reality.

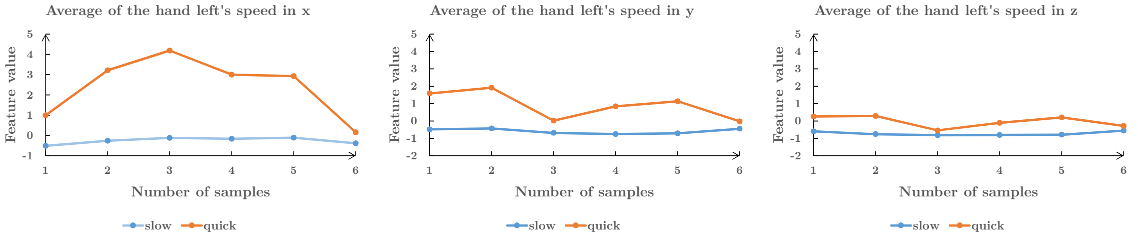


Figure 6.5: Average speed values for the left hand during the task “Wave right’ performed in two different ways.

The **average acceleration**, shown in Figure 6.6 follows approximately the same trend as the speed, except that the changes are less pronounced. In this context, the speed seemed more representative. For example, the changes in speed with y are more visible even if less pronounced than in x. However, the acceleration does not represent the same information even if it is correlated with the speed. For example, an acceleration has been revealed in the last part of the gesture. So, this feature is not to exclude even if in a first time, it would be more appropriate to only use the speed to classify this movement as the curve of the speed already allow to make a prediction. Of course, further tests with the classifiers will verify this hypothesis.

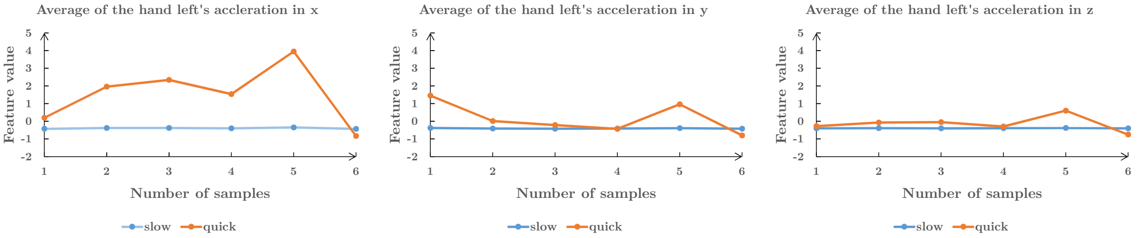


Figure 6.6: Average acceleration values for the left hand during the task “Wave right’ performed in two different ways.

We did not have the time to consider the other features computed in Section 6.6. However, all the gestures implemented (see Section 5.1) have also been evaluated with the average speed and gave the same observations. Since this feature seems representative of the reality, we preferred to focus on the randoms forests generated from collected data in Chapter 7.

6.11.2 Joint orientation evaluation

However, the joint position is not enough to represent the changes during a task. For example, a joint can stay at the exact same position while rotating. As discussed in Section 3.6.2, the position of the joints and the rotation of the joints coupled, are able to describe any movement. So, the features from the orientation will also be evaluated like the position.

Concerning the average speed for the orientation of the left hand, the results shows that the angle changes are more important on the x axis. The differences seems less pronounced than with the position.

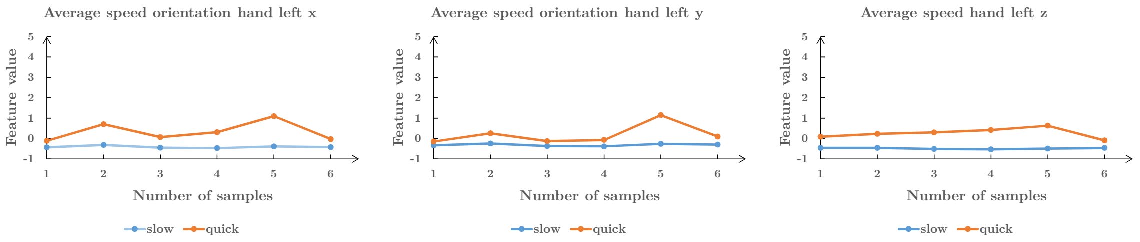


Figure 6.7: Average speed orientation for the left hand during the task "Wave right' performed in two different ways.

To ensure that the results clearly represent the changes in the orientation. We choose to also illustrate the variations with the elbow left in Figure 6.8. Indeed, during the gestures, the orientation of the elbow does not change as the elbow or the shoulder does not rotate during the gesture. As expected, the curve remains the same for quick and slow movements.

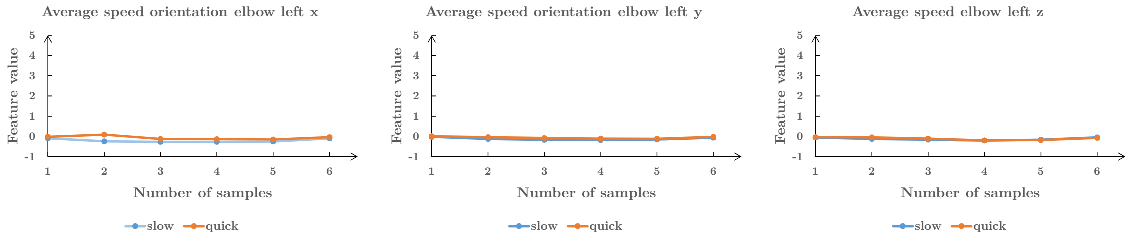


Figure 6.8: Average speed orientation the left elbow during the task "Wave right' performed in two different ways.

6.12 Neuropsychiatric features

In many tests attempting to measure the psychomotor retardations and the cognitive impairments, the time to complete the gesture is almost always recorded, as seen in Chapter 1 with the digit symbol test and the reaction time test. In the same way, the reaction time is also an important variable to measure. These relevant variables can be obtained easily for the entire task because of the recognition, we have implemented. In this manner, 4 measurements in data and time are labeled:

1. The task is started;
2. The first posture of the successful task has been performed;
3. The task has been completely executed (if failed, we measure 2 again);
4. The tasks is stopped.

Then, it is easy to compute the time elapsed between the four measurements. We are particularly interested in the time required to initiate the movement. The result 1 was subtracted from the result 2. Moreover, we were also interested in the time required to execute the movement. So, we subtract the result 2 from the result 3. The measure 4 is actually not use, but a trace was kept because the time after the task could be change with a different value than 5 seconds.

As discussed in Section 1.3, there are no consensus on the influence of the **gender** on the psychomotor retardation. So, it is recommended to consider gender for future classifications. Moreover, during the data collection, we have noticed some biases because of the embarrassment caused by the clothes. The **age** is also an important parameter to consider during classification. As aging already causes slowing. However, the data has been collected with a population with an age varying from 20 years to 27. In a first time, there is no real need to consider the feature. However for further tests with real patients, the parameter will be used.

Chapter 7

Evaluation of the random forests generated for classifying tasks

At this point, data have been collected, features have been computed and each record has been converted into a normalized vector in order to build a training set. We have also evaluated the best features and the classes have been defined. So, the classifier will be defined in order to predict the class of a task. We will describe the Weka implementation of random forests. Then we will discuss the meta-parameters which influence the learning phases. Next, some forests will be created and evaluated from our experiments.

7.1 Implementation of a random forest

As the features and the training data are available, a supervised machine learning algorithm can be built to classify the different tasks among the classes. At the beginning, we implemented a neural network by our own in C#. The operation revealed to be very complex as the implementation of a neural network is quite hard to understand. However, the algorithm could take up to 5 minutes before converging to a minimized error on the training set. Moreover, we discovered in Chapter 2.11 that the random forest will be a more appropriate algorithm. So, we managed to test a tool, called "Weka" to test a random forest. The algorithm takes a couple hundredths of a second to make 100% prediction over the training set. Naturally, the training error is not sufficient to estimate the quality of a model, but the generation time is quite impressive with random forests.

As there are already many toolboxes in Java, Python and other languages, it is much more relevant to use existing tools. In this way, we avoid being stuck with one or two models that would take time to implement. Furthermore, there is a difference between good and bad implementations of an algorithm. Naturally, there are algorithms much more powerful with fewer errors.

7.1.1 Weka

Waikato Environment for Knowledge Analysis [1] (Weka), is a well-known suite of machine learning tools in Java which includes a huge number of algorithms. These algorithms can perform data pre-processing, filtering, classification, regression, clustering, association rules, and visualization. Weka has been built over the years by experts in Machine Learning at the Department of Computer Science in the University of Waikato in New Zealand.

[1] https://www.cs.auckland.ac.nz/courses/compsci367s1c/tutorials/IntroductionToWeka.pdf

Although, the application using the Kinect is in C# and Weka is in Java, we did not find any particular problem. Indeed, a webservice has been created as a sort of wrapper to use directly the API. Even if Weka already provides a command line interface, we found out that it was much more flexible to use directly the API.

7.1.2 Meta-parameters of random forests

During the training of the model, some meta-parameters can be adjusted by using Weka. As a reminder, the complexity of the model is determined by the meta-parameters used during the learning phase. So, the meta-parameters must be chosen carefully to prevent underfitting but also overfitting (Section 2.8). Like other learning algorithms, the random forests have specific meta-parameters.

- The number of threes t in the forest determines the number of weak classifiers generated in the entire forest.
- The number of features f to be used in the random selection. For each tree in the forest and for each node split, f features are selected. Then the best feature among f is chosen to make the split.
- The maximum depth of the tree in the random forest ;
- The seed for the random number generator ;
- The number of threads used in parallelism which generate the tree. However, this last parameter affects only the performance in time during the learning and not the quality of the random forest.

7.2 Evaluation of the random forests with simulated tasks

At first, we chose to evaluate the forests generated with simulated tasks in order to be sure that the algorithm gives representative results.

7.2.1 Simulations of the "wave the hand" task

The "wave hand left" task has been used for testing. As seen in Figure 5.2, the task consists in repeating the step A and the step B, three times in a row. For recall, the gestures have been executed in four different ways:

1. With a reaction time particularly *short* and *fast* motions (class one) ;
2. With a reaction time particularly *short* but with *slow* motions (class two) ;
3. With a reaction time particularly *long* but with *fast* motions (class three) ;
4. With a reaction time particularly *long* and *slow* motions (class four).

For the first random forest created, only some features intuitively relevant for the task have been chosen. So, concerning the position of the joints, we chose to represent the movements by only using the average speed, which has been revealed as one of the best intuitive feature to identify the changes in the movements (Section 6.11). Furthermore, the tasks only required to move the left arm. So, only the left hand, the left wrist, the left elbow and the left shoulder were considered. The task has been fragmented in 6 parts: 1 part before the task, 4 parts during the task and 1 part after the task (5 seconds of observation after the task). In addition, to measure the reaction time we also used the time required to begin the gesture after the stimuli. In a short the following features have been computed:

$$(4 \text{ joints } * 3 \text{ axis } * 6 \text{ parts }) + 2 \text{ time measurements } = 74 \text{ features} \tag{7.1}$$

Validation with out-of-bag error: In order to test the quality of the model according to the parameters and the meta-parameters, the out-of-bag error will be used. With intuition, for the first random forest, the maximum depth of the trees was set at 10 because the model is only using 74 features and a decision tree with a large depth can overfit because of a poor generalization. In the same logic, the number of features was set at 10. Moreover, we preferred to generate a large number of weak classifiers. So the number of trees was fixed at 100. The seed was equal to 1 by default and the value was not changed. Indeed, with such number of trees, a lot of variability is already introduced.

As seen in Section 2.11, the out-of-bag error is an accurate measure to estimate the generalization error. This measure is especially recommended when using bagging. Breiman [17], even argued that using cross validation or a split validation was no longer needed. So, the model was trained with the entire training set. All the 45 instances were correctly classified. But of course, the evaluation was only make predictions about data already seen. Therefore, the out-of-bag error was evaluated at 0.0222 which means that in terms of generalization, we will have likely 2% of bad predictions. The result is quite impressive as we did not try to optimize the meta-parameters at this stage. However, this statement must be handled very carefully as only few labeled records are available and it is just an approximation. But, we can say that the results are certainly not provoked by accident.

Then, to test the ability of the out-of-bag error to evaluate the generalization error, we created a random forest completely overfitting. For this purpose, we decided to only create 3 trees with a maximum depth equals to 100. The other meta-parameters and parameters were not changed. The accuracy remains te same with 45 instances correctly classified. However, the out of out-of-bag error was disastrous with a result of 0.38. This result means that the algorithm will perform poorly on unseen data. The measures estimates that the model will fail to predict the right class approximately 4 times out of 10. This result proves that the accuracy on the training set alone cannot be trusted and that the statement of Breiman revealed to be true in this case. In this case, the model is overfitting because of a very low and uncommon number of trees. As already discussed, a random forest tends to be immunized against overfitting and are generally built with much more trees.

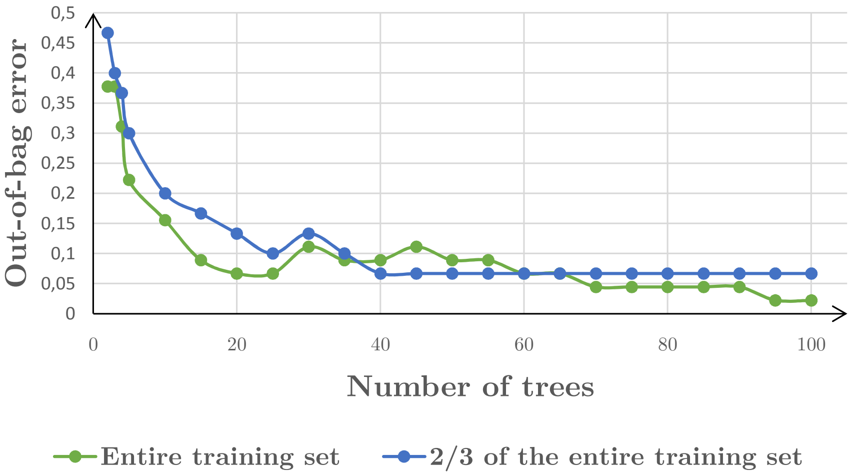


Figure 7.1: Out-of-bag error according to the number of trees.

We noticed that the performance on generalization was not really affected by the maximum depth of the tree or the number of features to select randomly at each split, as long as, they are not exaggerated. Furthermore, Breiman [17] focus much more on the number of trees in his paper describing the random forests. In further optimization, we will focus on the impact of the number of trees. In order to select the best number of trees for the entire training set, a lot of random forests have been generated as seen in Figure 7.1. The results are completely unsatisfactory between two and four trees. However, the out-of-bag error are promptly converging towards a minimum. From 20 trees, the results begin to be acceptable. With 80 trees, the error has already been successfully reduced. Then, from 95 trees, the error has converged to a minimum. It is only with 509 trees that the error continues to decrease. Therefore, we think that 100 is an acceptable number of trees. Indeed, at some point, the benefit from learning more trees will be lower than the cost in computation time. Moreover, we also risk overfitting by using very specific values as meta-parameters. However, much more records would be required to confirm this number.

Validation with separate training set and validation set: In order, to be very careful about the results. The evaluation was repeated but with a validation set completely independent from the training set even if, according to Breiman (Section 2.11.1), the out-of-bag error is a good estimation of the generalization error. For comparing, the both evaluations, the same meta-parameters were kept. One may say that, by doing so, the entire training set has influenced the model but 40, 60 or 80 trees (see Figure 7.1) gave exactly the same result. For this purpose, the entire training set was shuffled randomly before being split. Therefore, 30 records were included in the training set and 15 in the validation set. All the 15 instances has been correctly classified but as already mentioned, this measure alone does not assert the quality of a classifier. Once again, the out-of-bag error was computed. As represented in figure 7.1, with two third of the training set, the error is higher in the beginning but also until 35 trees. However, the value converge more rapidly towards a minimum but this value remains higher. As they are less data, the confidence over the generalization is smallest. Moreover, a lot of records can require more trees to give accurate classifications. It can explained why the error converges more rapidly as there are less records.

Evaluation of the model generated: In summary, the accuracy may not be the most reliable measure as it is not always good to detect a poor generalization. The precision, the recall and the F1 score have not been used as all the instances were correctly classified. However, as stated by Breiman, the out-of-bag error seems to be a great measure to estimate the generalization error. The error tends to decrease with more weak classifiers and training records. The evaluations of the out-of-bag error made with the entire training set and with a separate validation set showed great results. However, caution should be exercised, as there are only 45 records in the record set while thousands would be required. Moreover, another criticism can be addressed as the tasks were simulated. But variations were already introduced as two gestures cannot be exactly executed in the same way two times. In addition, no changes were done after the recording for boosting (wrongly) the classifier.

7.2.2 Simulation of the "Sit down" task

The evaluation of a single task is not enough to ensure the quality of our approach. We propose an other task which consists in sitting down. As a reminder this kind of tasks allow to test the balance but also to consider almost all the joints (see Section 4.3). The task has been executed 35 times:

a) 15 instances with *fast* motions (class one) ;

b) 15 instances with slow motions (class two) ;

c) 5 instances with motions neither slow nor fast (class three).

Unlike the "wave the hand tasks", the reaction time and the execution time are not used in order to prevent the algorithm to take a shortcut for detecting a slowing by only using the measures in time. We want that the algorithm detect the differences by investigating the motion. We also wanted to test a different distribution of instances with less training examples for a class.

The average speed of the changes in position will be computed for all the joints. In addition, only the parts (4 out of 6) where the gesture (see Section 6.9) is executed are considered, which gives:

$$(25 \text{ joints } * 3 \text{ axis } * 4 \text{ parts }) = 300 \text{ features} \tag{7.2}$$

The number of features selected at random was configured at 15 because a lot of features are available. For the maximum depth of each tree, a number of 10 was kept because a decision tree with a large depth can overfit. It is preferable to generate a large number of weak classifiers to have a better generalization (see Section 2.11).

As seen in Figure 7.2, the out-of-bag error converges very rapidly. A very good result of 0.029 is already obtained with only 10 trees. The error even reaches zero between 45 and 55 trees. So, a number of 50 trees seems to be a good choice. However, the error goes back to 0.029 with 60 trees and even with 100 trees. It would not be judicious to believe that the real generalization is equal to zero. As a reminder, the out-of-bag error is just an estimation of the generalization error (see Section 2.11)

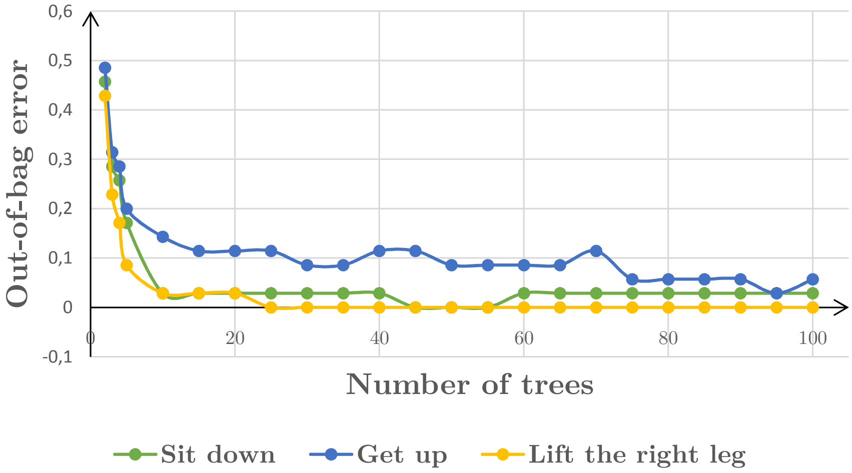


Figure 7.2: Out-of-bag error according to the number of trees.

7.2.3 Simulation of the “get up” task

Similarly, the “get up” task was evaluated with the same 3 classes. As the previous one, this task allows to test the balance during the changes in posture and uses almost all the joints. The same meta-parameters and parameters were chosen. In summary, 300 features were used (see Equation 7.2) with 10 features selected at random and a maximum depth of 15.

Figure 7.2 shows that from 80 trees the error converge towards 0.0571. The error is lower with 0.0286 at 95 trees but the error returns to 0.0571 with 100 trees. In addition, even with 1000 trees the error stays the same. In addition, even with 1000 trees the error stays the same. Therefore, picking a number around 95 seems to be a good idea but it is more careful to consider that the real estimation of the error is around 0.0571.

Some k-fold validations were done to give an other evaluation of the error independent from the out-of-bag error. A validation with 5 folds has been performed with 95 trees and gives an accuracy of 91.43%. This result can seem different from the prediction of the out-of-bag error. It can be explained by the fact that with less training instances, more trees are needed to train an accurate classifier as seen in Figure 7.1 with the first classifier evaluated. So, the evaluation was repeated with 150 trees. The accuracy error was equal to 0.0571 and to the out-of-bag error. In this validation, 1 instance with fast motion and 1 instance with slow motion were classified as normal motion. With a different number of trees, the instances misclassified remains the same. Therefore, it is likely that they are also the instances which increase the out-of-bag-error. The k-fold validation were only described for this task as the error was higher than in the others. Moreover, it has revealed correlations with the out-of-bag-error.

7.2.4 Simulation of the “lift the right leg” task

For the last simulation focusing on the changes in position for the joints, the “lift the right leg” task was evaluated with 3 different classes. In the first class, the subject has difficulties to maintain balance and make many movements to correct his position. For the second class, the subject has less difficulties and make only few movements to maintain balance. In the last class, the subject has a perfect balance and does not move.

Figure 7.2 shows that with only 10 trees, the out-of-bag error was already acceptable with a value of 0.0286 and is even equal to zero from 25 trees. Once again, it would be inappropriate to believe that the model does not have any error. Indeed, they are only few instances in the training set.

7.2.5 Simulation of the "rotate the right arm" task

Until now, the models generated have used the changes in position or different measures in time to make predictions. We also wanted to explore the ability to learn of the algorithm with the orientation instead of the position. Only the features representative of the orientation during the gesture (4 parts out of 6) were kept. Therefore, a task where the right arm does not move in space but only rotates has been analyzed. We choose to focus on the arm because it is the easiest part of the body to rotate. So, the hand, the wrist and the elbow were used. As, the average speed showed great results in Section 6.11, it will be used to compute the features. In short, the following features are available:

$$(3 \text{ joints } * 3 \text{ axis } * 4 \text{ parts }) = 36 \text{ features} \tag{7.3}$$

The minimized out-of-bag error is around 0.05 with 45 trees, 5 features used in the random selection and a max depth for the trees of 10. It is possible to have a smallest error but the errors seems to converge towards 0.05. A cross validation with 5 folds have also evaluated the model. As a result, the accuracy was equal to 0.05 with 1 slow rotation misclassified as a fast rotation.

7.2.6 Blink simulation

As described in Section 5.1, a specific task has been created for the eyes, as the Kinect needs specific conditions to get results from the eyes. The subject must be close to the camera while facing it. So, we made 4 different kinds of records which lasts approximately 10 seconds:

a) 7 instances where the subject was blinking with both eyes (class one)

b) 8 instances where the subject was blinking with only the left eye (class two)

c) 7 instances where the subject was blinking with only the right eye (class three)

d) 8 instances where the subject was not blinking (class four).

A large number of trees was needed to converge towards the minimized out-of-bag error. To get an error of 0.2, the algorithm has generated 850 trees. It should be noted that a leave-one-out-cross-validation (see Section 2.9), gives an accuracy error of 0.2. However, this validation may be too optimistic, as each model generated is just slightly different from the model built with the entire training set. So, a validation with 5 folds was done and gives an accuracy of 70%. The confusions matrix is given below to represent the misclassified instances:

a	b	c	d	
4	3	0	0	a = class one
2	6	0	0	b = class two
0	0	5	2	c = class three
0	1	1	6	d = class four

A row represents all the instances which really belong to a class in the training set and a column represents how the instances were classified by the model. We can see that the class one have the most false positive instances with a recall (see Section 2.10) of 0.571 ($\frac{4}{4+3}$). The precision for the class two is also not very good with a result of 0.6 ($\frac{6}{6+4}$) where 4 false negatives were misclassified in the class one. It seems that there are confusions between the class one and the class two. The same problem can also be found within the classes three and four. There are many possible reasons explaining the error. Indeed, there are only few instances, it would be interesting to have more records in order to allow the algorithm to learn the differences. Moreover, as already mentioned in Section 3.12, the predictions for the eyes are less accurate than the predictions for the joints. The subject must face the camera and be close enough. Some noises can also be induced by the lightning of the room. Perhaps that the actual formulas applied to the features concerning the eyes are less appropriate than for the joints. Maybe that counting the number of blinks and the duration of each closing and opening would be interesting to investigate. However, the error is quite reasonable but it would require a large training set to assert this hypothesis.

7.2.7 Critical analysis for the results of the simulated tasks

As already mentioned, even if most of the models are giving good results, caution should be exercised. Indeed, for each model only some records has been used in the data set while thousands would be required. Furthermore, from now, all the tasks evaluated were simulated. However, we try to build models with multiple classes in order to make the classification more difficult. Moreover, all the records are different as two gestures cannot be exactly executed in the same way especially when the records must be done in four different ways.

The models showed better results with the features concerning the changes in position and the measurements in time. We think that it is certainly due to the fact that the Kinect gives better prediction for the position than for the orientation and eye tracking.

7.3 Evaluation of the random forests according to alcohol consumption

As discussed in Chapter 4, during the data collection, 14 subjects including 9 women and 5 men with an average age of 24 years have tested 4 distinct tasks in order to test different parts of the body and the changes in poses. So, 67 tasks were executed without alcohol consumption and 25 with alcohol consumption. The alcohol consumption will be the positive class.

The first task described is quite similar than the previous one. It consists in "waving the right hand". So, only the right hand, the right wrist, the right elbow were considered. In the same vein, the task has been fragmented in 6 part. Both the reaction and the time required to begin the gesture have been recorded and we only use 3 joints.

$$(3 \text{ joints } * 3 \text{ axis } * 6 \text{ parts }) + 2 \text{ time measurements } = 56 \text{ features} \tag{7.4}$$

Unfortunately, the generalization performs very poorly whatever the meta-parameters and the parameters. Indeed, we tried with different features than above. So the acceleration, the orientation and other compositions were used but we never do better than 0.3043 for the out-of-bag error. We directly noticed that this result was equal to $\frac{7}{23}$ where 23 is the number of instances and 7 the number of instances with alcohol. In order, to verify what was the problem, a k-fold validation with $k = 5$ was done. In result, all the instances with alcohol were misclassified and only 1 from the healthy class. Therefore, we can say that the algorithm has not learned to differentiate the two classes as all instances were considered as healthy except the one false positive.

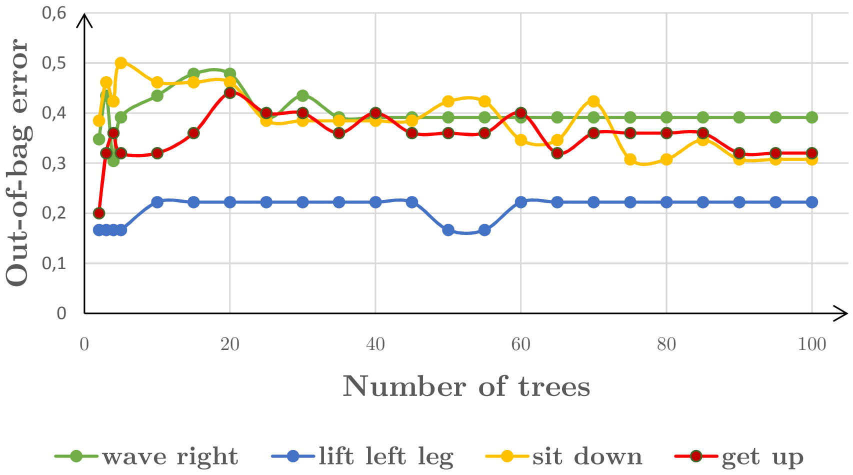


Figure 7.3: Out-of-bag error according to the number of trees in the evaluation of the alcohol consumption.

Despite of these results, a model has been generated for the 3 other tasks. However, the outputs were exactly the same as seen in Figure 7.3. So, for each task, one of the best configuration that could be find gives an out-of-bag error approximately equal to the rate of positive instances. The second tasks consisted in raising the leg foot above the right ankle during 10 seconds while keeping balance. The rate of positive instances was equal to

$\frac{3}{18}$. Effectively, the best out-bag-error is equal to 0.17 and is located between 45 and 55 trees. The third task was to sit down and gives the same output as the first two with a rate equal to $\frac{8}{26}$ and an out-of-bag error of 0.31. The last tasks was to get up from a chair. The rate was equal to $\frac{7}{25}$ and the out-of-bag error was 0.28.

In summary, the classifiers have not succeeded in discriminating the two classes as almost all the instances were classified as healthy in the k-fold cross validation and certainly in the out-of-bag classifier (see Section 2.11). Moreover, different compositions of parameters and meta-parameters has not changed the approximation of the generalization error. Therefore, we analyzed what the problems were. At first, even if we had 92 tasks, there was 4 different tasks with only a maximum of 26 instances per task. It would be interesting to get more much more records. In addition the rate of positive instances was smaller than the rate of negative instances.

Moreover from our point of view, we did not notice particular differences during the experiments except for 2 or 3 records among the 98. We think that it can be explained by the fact that the subject knew that they were evaluated. Some of them wondered how to do the task correctly. In this context, some gestures did not seem natural. However, we expected that the camera would highlight some differences that were not easily visible to the naked eye.

In addition, the vulnerabilities due to the alcohol, are different from one individual to another. We are not going to extend this fact but it is well-known that the genetic, the robustness and the age of a person influence the effects of alcohol, even if we have seen that differences on the motor and cognitive components were visible in [72] (Section 1.5). However, it was an average of the entire group. So, maybe that a wider sample will provide more accurate results but it will be really difficult to get rid of the false negatives. Indeed, we are only trying to detect psychomotor or cognitive impairments and not to predict the alcohol consumption. All these facts can explain, why the classifiers has performed poorly.

In the future, it would be interesting that the subjects forget that they are evaluated. One idea would be to create a kind of video game with a lot of mini games randomly proposed. Moreover, we think that a more rigorous experiment where each cognitive and motor impairments is rigorously controlled, would be required to give significant results.

Conclusion

During this master thesis, we focused on the detection of neuropsychiatric disorders by using motion capture and learning algorithms with short predefined tasks. So, large datasets of movements have been collected by using the second version of the Kinect.

In Chapter 1, the psychomotor retardation, including motor impairments and cognitive impairments, has been described as one of the most common features to diagnose neuropsychiatric disorders or at least depressive subtypes. Moreover, differences in terms of motor and cognitive impairments have been found among the different disorders. Furthermore, we have also described in this document, the use of a motion capture system to evaluate the efficiency of a treatment for Parkinson [67]. These findings suggest that the use of motion capture is a viable approach to collect data representative of neuropsychiatric disorders. So, regularities in the data could be highlighted by machine learning algorithms in order to make predictions for a disorder.

We wanted a computer-based system able to present a stimulus and to score patients in a normalized way. For that purpose, an heuristic recognition of gestures based on an finite automate and conditions has been implemented in our prototype to recognize short predefined tasks. Moreover, a machine learning recognition was also used to build AdaBoost and random forest models for recognizing postures and gestures respectively. Moreover, the prototype allows to record sessions composed of tasks which are automatically recognized. During these tasks, the position of 25 joints of the body has been recorded in 3D, but also the orientation of these joints in a quaternion. The state of each eye, which indicates if the eye is opened or closed, has also been recorded. Many tasks (sitting, get up, wave the hand, lift the leg during 10 seconds, rotate the arm) have been implemented to test the balance, the speed of the movements and the changes in postures.

The position of the joints depends from the camera perspective but also from the morphology of the subject. In order to get invariant features, for each frame, the speed and the acceleration were computed for the positions but also for the rotations. At this stage, the records of a task are represented in matrices with different sizes because each record can have a specific number of frames. In this context, each matrix has been converted into a normalized vector including the same features for each specific task in order to be use in a training set. For that purpose, the average and the standard deviation were applied. Furthermore, to keep a maximum amount of information, each record has been split into k parts. Then, we evaluated the features with a “wave hand” gesture performed 23 times quickly and 22 times slowly. We found out that the average speed of the position was the most discriminating features for this task among the features evaluated. The reaction time and the time to realize the gestures have also been recorded. Neuropsychiatric features as motor or cognitive impairments were used as classes.

The prototype must be tested and the proof of concept must be done before using real patients. Initially, this proof of concept has been realized by simple simulations. So, simulations of psychomotor retardation have been recorded in different ways, with most of the tasks implemented, by varying the speed of the movements but also the initiation time. However, for testing in more rational situations, another solution was necessary. One of this solution, is the cognitive impairments caused by an alcohol consumption, studied in Chapter 1. Indeed, a 0.05% alcohol blood concentration causes cognitive and motor impairments.

First, the simulated tasks have been evaluated. So, the first random forest was trained by using the "wave the left hand" task. The task has been performed in 4 different ways with slow and quick movements combined with short and long reaction times. The generalization error has been approximated with the out-of-bag-error. As a result, we got an impressive out-of-bag-error of only 2.2%. Three random forests were created with 3 others different tasks. These models also showed great outputs for "sit down", "get up" and "lift the leg" tasks. An "eye tracking" task was also investigated but give less accurate results with a generalization error of 20%. Despite the impressive results obtained, the method can be criticized. Indeed, it is only with thousands or even millions of training records that the quality of a machine learning model can really be ensured. It would be inappropriate to believe that the model does not have any error. In this context, it would be interesting to evaluate the models with a more theoretical approach or with an empirical approach including a large data set of millions of records and a large variety of tasks, but also a large mixture of groups.

Then, we generated random forests to detect the motor and cognitive impairments caused by the consumption of alcohol in the general population. However, the classifiers have not succeeded in discriminating the two classes as almost all the instances were classified as "without alcohol consumption". Moreover, different compositions of parameters and meta-parameters did not change the approximation of the generalization error. However, during the experiments, we did not highlight particular differences. We think that it can be explained by the fact that the subject knew that they were evaluated. Moreover, the vulnerabilities due to alcohol are different from one individual to another. In this context, we would like to propose some perspectives for future work, but also for future evaluations.

Perspectives

At this point, different tasks have been classified according to motor and cognitive impairments. But we would also want to classify patients in groups. For example, these groups could represent diseases. In this context, the major idea is to use the results given by the classification of the tasks as features for learning a model capable of making a diagnosis for a patient. Moreover, other variables discussed in Chapter 1 would be combined with these features.

Other sensors than the Kinect should also be considered. In Section 5.1, we discussed about the difficulties of the Kinect to give a proper result concerning the eye tracking. The subject must face the camera and be close enough. Moreover, the model generated for detecting the blinks (see Section 7.2) have some difficulties to give the right predictions. Furthermore, the Kinect is unable to detect the pupils.

Actually, only the animation units, described in Section 3.12, are used in face tracking. However, the Kinect is able to provide a 3D mesh model of the face deformed by parameters to represent facial expressions. However, we must be careful, as the Kinect has difficulties with most of the animation units, especially for little areas as the lips or the eyebrows. In the same context, we would like to analyze in further details the orientation of the joints. Indeed, we mainly used the position because it is the most representative and accurate features provided by the Kinect.

In order, to make an evaluation in a real situation, we think that we need a more controlled procedure. Instead of using alcohol consumption, we could use the sleep inertia state as a proof of principle. This idea has been used by the Massachusetts Institute of Technology in collaboration with the Harvard Medical School in [27] as a proof of concept for detecting psychomotor impairments through the interactions with a keyboard. Moreover, it exists research centers in Belgium that conduct sleep research as the Cyclotron Research Center at the University of Liege.

Finally, it would be interesting that the patients forget that they are evaluated. Indeed, we think that is was a real problem during the data collection in a general population. In this context, some gestures did not seem natural. One idea would be to create a kind of video game with a lot of mini games randomly proposed. Moreover, a video game would allow to prevent boredom by varying the different tasks. Brown depicts (in [20]) a method based on a video game. Such a system would allow a standardized presentation of the stimuli but also a regulated scoring. Moreover, the cost of treatments could be decreased by reducing the number of therapies with a specialist. Indeed, the patient could use the system at home and so execute the sessions in a completely natural environment. So, such a system is absolutely in accordance with the objectives of this work (see Chapter 1).

Bibliography

[1] All About Depression: Diagnosis. http://www.allaboutdepression.com/dia_03.html. (Visited on 27/07/2015).

[2] Chronic Fatigue Syndrome (CFS) Case Definition. http://www.cdc.gov/cfs/case-definition/index.html. (Visited on 27/07/2015).

[3] CHU Dinant Godinne — UCL Namur. http://www.uclmontgodinne.be/. (Visited on 01/08/2015).

[4] Cognitive Impairment - Symptoms, Causes, Treatments. http://www.healthgrades.com/symptoms/cognitive-impairment. (Visited on 26/07/2015).

[5] Kinect for Windows SDK 2.0 Programming Guide Body tracking. https://msdn.microsoft.com/en-us/library/dn799273.aspx. (Visited on 21/01/2015).

[6] Schizophrenia: Symptoms, types, causes, and early warning signs. http://www.helpguide.org/articles/schizophrenia/schizophrenia-signs-types-and-causes.htm. (Visited on 27/07/2015).

[7] sklearn.preprocessing.standardscaler — feature normalization. http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html. (Visited on 31/07/2015).

[8] Visual Gesture Builder Detection Technologies: AdaBoostTrigger. https://msdn.microsoft.com/en-us/library/dn785522.aspx. (Visited on 14/07/2015).

[9] Clemens Amon, Ferdinand Fuhrmann, and Franz Graf. Evaluation of the spatial resolution accuracy of the face tracking system for kinect for windows v1 and v2. In *Proceedings of the 6th Congress of the Alps Adria Acoustics Association*, 2014.

[10] Sheryl Ankrom. What Is Psychomotor Agitation? http://panicdisorder.about.com/od/glossaryip/g/PsychMtrAgita.htm. (Visited on 27/07/2015).

[11] Martin Baker. Conversion Quaternion to Euler. http://www.euclideanspace.com/maths/geometry/rotations/conversions/quaternionToEuler/. (Visited on 03/08/2015).

[12] Thomas J Barkley and Warren W Tryon. Psychomotor retardation found in college students seeking counseling. *Behaviour research and therapy*, 33(8):977–984, 1995.

[13] AM Baumberg and DC Hogg. An efficient method for contour tracking using active shape models. In *Motion of Non-Rigid and Articulated Objects, 1994., Proceedings of the 1994 IEEE Workshop on*, pages 194–199. IEEE, 1994.

[14] Djamila Bennabi, Pierre Vandel, Charalambos Papaxanthis, Thierry Pozzo, and Emmanuel Haffen. Psychomotor retardation in depression: A systematic review of diagnostic, pathophysiologic, and therapeutic implications. *BioMed research international*, 2013, 2013.

[15] Christopher M Bishop. *Pattern recognition and machine learning.* springer, 2006.

[16] Greg Borenstein. *Making Things See: 3D vision with Kinect, Processing, Arduino, and MakerBot.* " O'Reilly Media, Inc.", 2012.

[17] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[18] Steve Bressert.

[19] Timo Breuer, Christoph Bodensteiner, and Michael Arens. Low-cost commodity depth sensor comparison and accuracy analysis. In *SPIE Security+ Defence*, pages 92500G–92500G. International Society for Optics and Photonics, 2014.

[20] Stephen J Brown. Method for treating medical conditions using a microprocessor-based video game, 1999. US Patent 5,918,603.

[21] Jeylan S Buyukdura, Shawn M McClintock, and Paul E Croarkin. Psychomotor retardation in depression: biological underpinnings, measurement, and treatment. *Progress in Neuro-Psychopharmacology and Biological Psychiatry*, 35(2):395–409, 2011.

[22] Anna-Marie Chirico and Albert J Stunkard. Physical activity and human obesity. *New England Journal of Medicine*, 263(19):935–940, 1960.

[23] Nicolas Dantchev and DJ Widlöcher. The measurement of retardation in depression. *The Journal of clinical psychiatry*, 59:19–25, 1997.

[24] James Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58:15–16, 2006.

[25] Yoav Freund, Robert Schapire, and N Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[26] Benoît Frénay. Machine learning : des réseaux de neurones aux big data. `http://directory.unamur.be/teaching/courses/INFOM444`.

[27] L Giancardo, A Sánchez-Ferro, I Butterworth, CS Mendoza, and JM Hooker. Psychomotor impairment detection via finger interactions with a computer keyboard during natural typing. *Scientific reports*, 5, 2015.

[28] Jungong Han, Ling Shao, Dong Xu, and Jamie Shotton. Enhanced computer vision with microsoft kinect sensor: A review. 2013.

[29] Christopher W Hess and Rachel Saunders-Pullman. Review: Movement disorders and alcohol misuse. *Addiction biology*, 11(2):117–125, 2006.

[30] MG Hildebrandt, Kurt Bjerregaard Stage, and Per Kragh-Soerensen. Gender and depression: a study of severity and symptomatology of depressive disorders (icd-10) in general practice. *Acta Psychiatrica Scandinavica*, 107(3):197–202, 2003.

[31] Noel H Hughes. Quaternion to euler angle conversion for arbitrary rotation sequence using geometric methods. *Accessed online*, 2, 2008.

[32] Black Dog Institute. Melancholic depression - melancholic depression - our model of depression - depression - black dog institute. `http://www.blackdoginstitute.org.au/healthprofessionals/depression/ourmodelofdepression/melancholicdepression.cfm`. (Visited on 27/07/2015).

[33] Jared St Jean. *Kinect Hacks: Tips & Tools for Motion and Pattern Detection.* " O'Reilly Media, Inc.", 2012.

[34] Arthur R Jensen. Clocking the mind: Mental chronometry and individual differences. pages 11, 235.

[35] Arthur R Jensen and Ella Munro. Reaction time, movement time, and intelligence. *Intelligence*, 3(2):121–126, 1979.

[36] Xiaofei Ji and Honghai Liu. Advances in view-invariant human motion analysis: A review. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40(1):13–24, 2010.

[37] Lewis L Judd, Mark Hyman Rapaport, Martin P Paulus, and John L Brown. Subsyndromal symptomatic depression: a new mood disorder? *Journal of Clinical Psychiatry*, 1994.

[38] Rob Knies. Collaboration, expertise produce enhanced sensing in Xbox One - The Official Microsoft Blog. `http://blogs.microsoft.com/blog/2013/10/02/collaboration-expertise-produce-enhanced-sensing-in-xbox-one/`. (Visited on 23/01/2015).

[39] Susan G Kornstein, Alan F Schatzberg, Michael E Thase, Kimberly A Yonkers, James P McCullough, Gabor I Keitner, Alan J Gelenberg, CE Ryan, AL Hess, Wilma Harrison, et al. Gender differences in chronic major and double depression. *Journal of Affective disorders*, 60(1):1–11, 2000.

[40] E Lachat, H Macher, MA Mittet, T Landes, and P Grussenmeyer. First experiences with kinect v2 sensor for close range 3d modelling. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS)*, 2015.

[41] Michaël Marcozzi. *Théorie des Langages de Programmation : Syntaxe et Sémantique : Eléments théoriques et exercices.* Presses Universitaires de Namur, 2014.

[42] Tom M Mitchell. Machine learning. wcb, 1997.

[43] Thomas B Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268, 2001.

[44] Andrew Ng. Coursera Standford – Machine learning. `https://class.coursera.org/ml-007/lecture`. (Visited on 01/12/2014).

[45] International Encyclopedia of Rehabilitation. Pathophysiology of Motor Impairments. `http://cirrie.buffalo.edu/encyclopedia/en/article/360/`. (Visited on 26/07/2015).

[46] Monish Parajuli, Dat Tran, Wanli Ma, and Divya Sharma. Senior health monitoring using kinect. In *Communications and Electronics (ICCE), 2012 Fourth International Conference on*, pages 309–312. IEEE, 2012.

[47] Massimo Piccardi. Background subtraction techniques: a review. In *Systems, man and cybernetics, 2004 IEEE international conference on*, volume 4, pages 3099–3104. IEEE, 2004.

[48] MPBI Pier, W Hulstijn, and BGC Sabbe. Differential patterns of psychomotor functioning in unmedicated melancholic and nonmelancholic depressed patients. *Journal of psychiatric research*, 38(4):425–435, 2004.

[49] MPBI Pier, W Hulstijn, and BGC Sabbe. Psychomotor retardation in elderly depressed patients. *Journal of affective disorders*, 81(1):73–77, 2004.

[50] Marcia Purse. Psychomotor agitation - definition. `http://bipolar.about.com/od/glossaryp/g/gl_psymotoragit.htm`. (Visited on 27/07/2015).

[51] Marilyn J Rantz, Ms Tanvi S Banerjee, Ms Erin Cattoor, Ms Susan D Scott, Marjorie Skubic, and Mihail Popescu. Automated fall detection with quality improvement "rewind" to reduce falls in hospital rooms. *Journal of gerontological nursing*, 40(1):13, 2014.

[52] Guinness World Records. Fastest-selling gaming peripheral. http://www.guinnessworldrecords.com/records-9000/fastest-selling-gaming-peripheral/.

[53] D Rogers, AJ Lees, EILEEN SMITH, M Trimble, and GM Stern. Bradyphrenia in parkinson's disease and psychomotor retardation in depressive illness an experimental study. *Brain*, 110(3):761–776, 1987.

[54] MA Rogers, JL Bradshaw, JG Phillips, and E Chiu. Reliance on external cues during serial sequential movement in major depression. *Journal of Neurology, Neurosurgery & Psychiatry*, 69(2):237–239, 2000.

[55] MA Rogers, JL Bradshaw, JG Phillips, E Chiu, K Vaddadi, I Presnel, and C Mileshkin. Parkinsonian motor characteristics in unipolar major depression. *Journal of clinical and experimental neuropsychology*, 22(2):232–244, 2000.

[56] Guilherme Cesar Soares Ruppert, Leonardo Oliveira Reis, Paulo Henrique Junqueira Amorim, Thiago Franco de Moraes, and Jorge Vicente Lopes da Silva. Touchless gesture user interface for interactive image visualization in urological surgery. *World journal of urology*, 30(5):687–691, 2012.

[57] A John Rush and Jan E Weissenburger. Melancholic symptom features and dsm-iv. *American Journal of Psychiatry*, 151(4):489–498, 1994.

[58] Bernard Sabbe, Wouter Hulstijn, Jacques van Hoof, HG Tuynman-Qua, and Frans Zitman. Retardation in depression: assessment by means of simple motor tasks. *Journal of Affective Disorders*, 55(1):39–44, 1999.

[59] Nicola Salmoria.

[60] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

[61] D Schrijvers, F Van Den Eede, Y Maas, P Cosyns, W Hulstijn, and BGC Sabbe. Psychomotor functioning in chronic fatigue syndrome and major depressive disorder: A comparative study. *Journal of affective disorders*, 115(1):46–53, 2009.

[62] Didier Schrijvers, Wouter Hulstijn, and Bernard GC Sabbe. Psychomotor symptoms in depression: a diagnostic, pathophysiological and therapeutic tool. *Journal of affective disorders*, 109(1):1–20, 2008.

[63] Ken Shoemake. ompact, efficient and numerically stable. *Graphics gems IV*, pages 222–229, 1994.

[64] Jamie Shotton, Ross Girshick, Andrew Fitzgibbon, Toby Sharp, Mat Cook, Mark Finocchio, Richard Moore, Pushmeet Kohli, Antonio Criminisi, Alex Kipman, et al. Efficient human pose estimation from single depth images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(12):2821–2840, 2013.

[65] Melinda Smith and Jeanne Segal. Bipolar Disorder Signs & Symptoms: Mania & Bipolar Depression. `http://www.helpguide.org/articles/bipolar-disorder/bipolar-disorder-signs-and-symptoms.htm`. (Visited on 27/07/2015).

[66] Christina Sobin and Harold A Sackeim. Psychomotor symptoms of depression. *American Journal of Psychiatry*, 154(1):4–17, 1997.

[67] Adam Świtoński, Magdalena Stawarz, Magdalena Boczarska-Jedynak, Aleksander Sieroń, Andrzej Polański, and Konrad Wojciechowski. The effectiveness of applied treatment in parkinson disease based on feature selection of motion activities. *Przeglkad Elektrotechniczny*, 88(12b):103–106, 2012.

[68] Lukas Tanner, Mark Schreiber, JG Low, Adrian Ong, Thomas Tolfvenstam, Yee Ling Lai, Lee Ching Ng, Yee Sin Leo, Le Thi Puong, Subhash G Vasudevan, et al. Decision tree algorithms predict the diagnosis and outcome of dengue fever in the early phase of illness. *PLoS Negl Trop Dis*, 2(3):e196, 2008.

[69] Graham William Taylor. *Composable, distributed-state models for high-dimensional time series.* PhD thesis, University of Toronto, 2009.

[70] JJM Van Hoof, BJM Jogems-Kosterman, BGC Sabbe, FG Zitman, and W Hulstijn. Differentiation of cognitive and motor slowing in the digit symbol test (dst): differences between depression and schizophrenia. *Journal of Psychiatric Research*, 32(2):99–103, 1998.

[71] Anita C Volkers, Joke HM Tulen, Walter W Van Den Broek, Jan A Bruijn, Jan Passchier, and Lolke Pepplinkhuizen. 24-hour motor activity after treatment with imipramine or fluvoxamine in major depressive disorder. *European neuropsychopharmacology*, 12(4):273–278, 2002.

[72] Ann M Williamson and Anne-Marie Feyer. Moderate sleep deprivation produces impairments in cognitive and motor performance equivalent to legally prescribed levels of alcohol intoxication. *Occupational and environmental medicine*, 57(10):649–655, 2000.

[73] Christopher Richard Wren, Ali Azarbayejani, Trevor Darrell, and Alex Paul Pentland. Pfinder: Real-time tracking of the human body. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):780–785, 1997.

[74] Stuart C Yudofsky and H Florence Kim. *Neuropsychiatric assessment*, volume 22. 2008.

[75] Jiang Yu Zheng and Shigeru Suezaki. A model based approach in extracting and generating human motion. In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, volume 2, pages 1201–1205. IEEE, 1998.

Part III

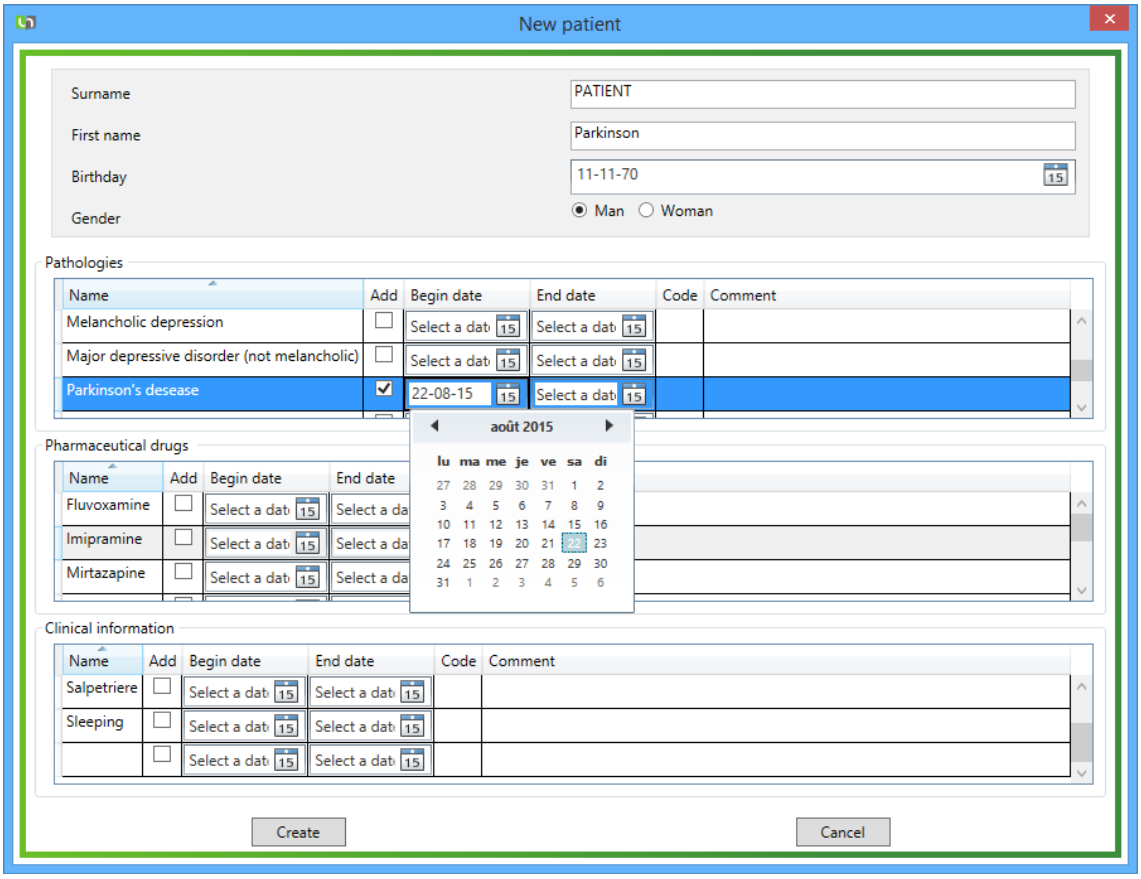
Appendix

Appendix A

Software implementation

This appendix aims to give a global view of the software implementation which has been done for the purpose of this work. Many details of implementation have already been given throughout this document. This section will introduce the most interesting interfaces which have not yet been presented.

Figure A.1 shows the implementation of **Chapter 4**. All the data about the patient are recorded with this interface. However, it is a simplified version of the schema proposed in **Figure 4.1**. Indeed, we preferred to focus on the classification of the tasks. A patient can be loaded, but also updated or deleted.



Surname	PATIENT
First name	Parkinson
Birthday	11-11-70
Gender	● Man ○ Woman

Pathologies

Name	Add	Begin date	End date	Code	Comment
Melancholic depression	□	Select a dat	Select a dat		
Major depressive disorder (not melancholic)	□	Select a dat	Select a dat		
Parkinson's desease	☑	22-08-15	Select a dat		

août 2015

lu	ma	me	je	ve	sa	di
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

Pharmaceutical drugs

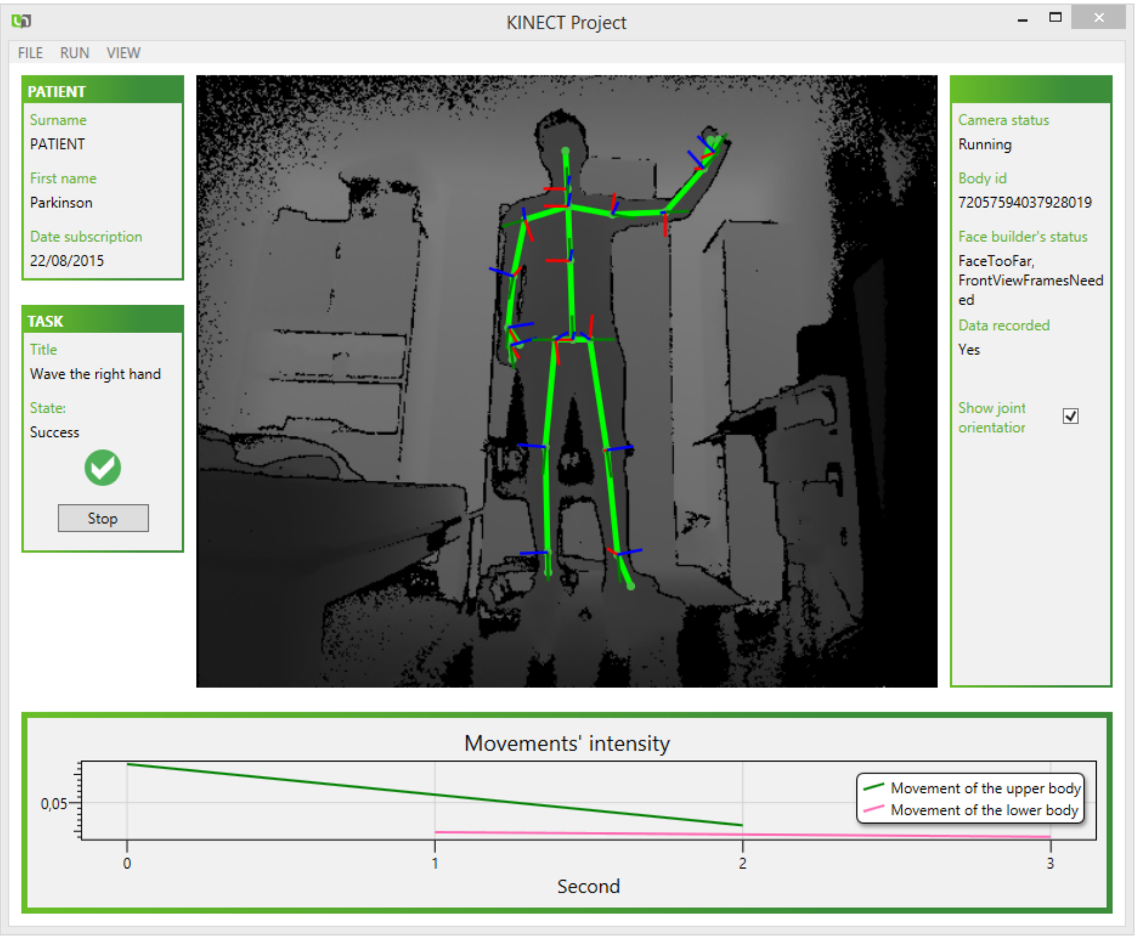
Name	Add	Begin date	End date
Fluvoxamine	□	Select a dat	Select a da
Imipramine	□	Select a dat	Select a da
Mirtazapine	□	Select a dat	Select a da

Clinical information

Name	Add	Begin date	End date	Code	Comment
Salpetriere	□	Select a dat	Select a dat		
Sleeping	□	Select a dat	Select a dat		
	□	Select a dat	Select a dat		

Figure A.1: A new patient with Parkinson is created.

Before recording a session composed of tasks, a patient must be loaded. The creation of a session has already been discussed in **Chapter 5** and represented in **Figure 5.3**. Once the session is started, the system are waiting for a gesture from the patient as already shown in **Figure 5.1**. When the current task has been automatically recognized, the systems informed the patient of the success. If the system is used by a specialist, he can introduce a comment concerning the realization of each task. However, this step is not mandatory (see Figure 5.3) in order to execute all the tasks of the session in a row without interacting with a mouse or a keyboard. In this way, all the tasks presented in Chapter 5 can be executed.



Surname: PATIENT

First name: Parkinson

Date subscription: 22/08/2015

Title: Wave the right hand

State: Success

Camera status: Running

Body id: 72057594037928019

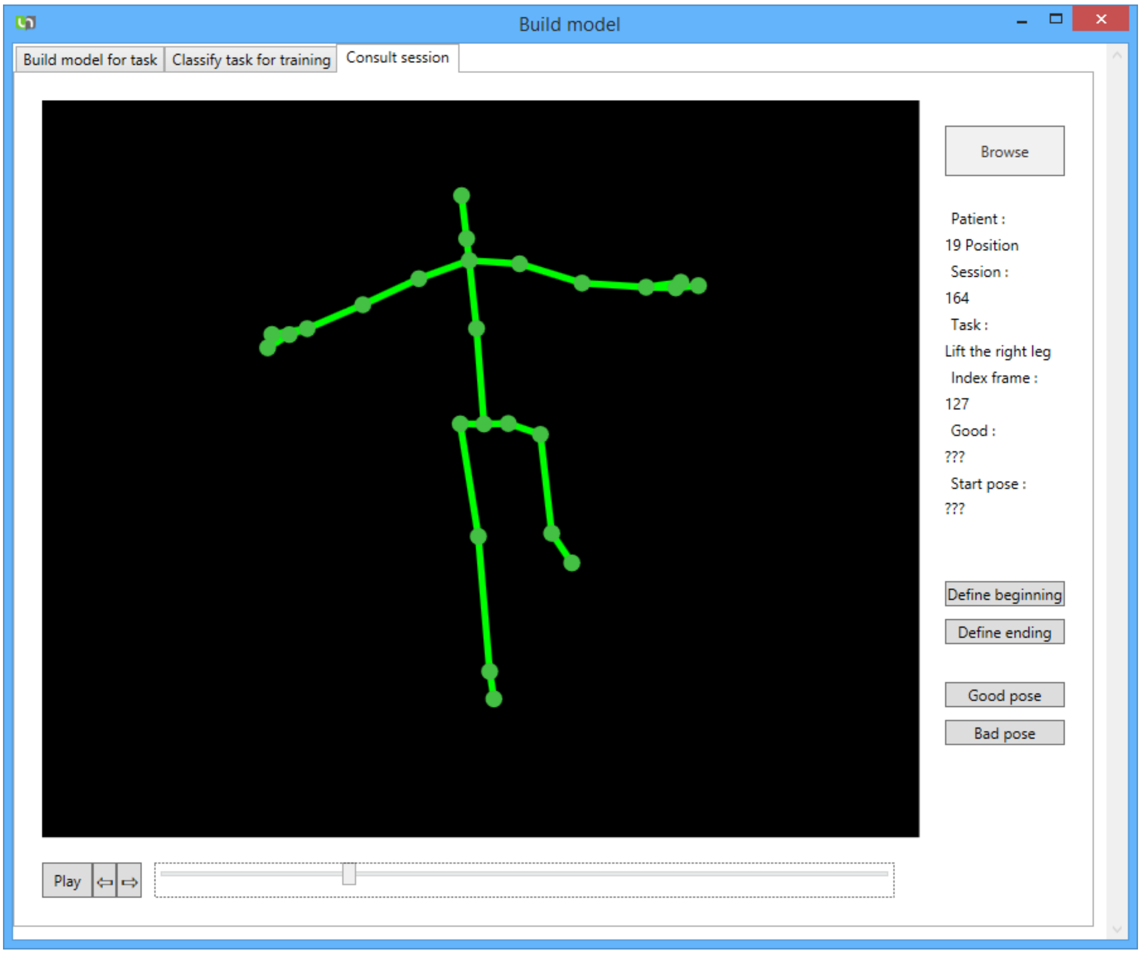
Face builder's status: FaceTooFar, FrontViewFramesNeeded

Data recorded: Yes

Show joint orientatior ☑

Figure A.2: The good execution of the task is automatically detected and the subject is informed of the success.

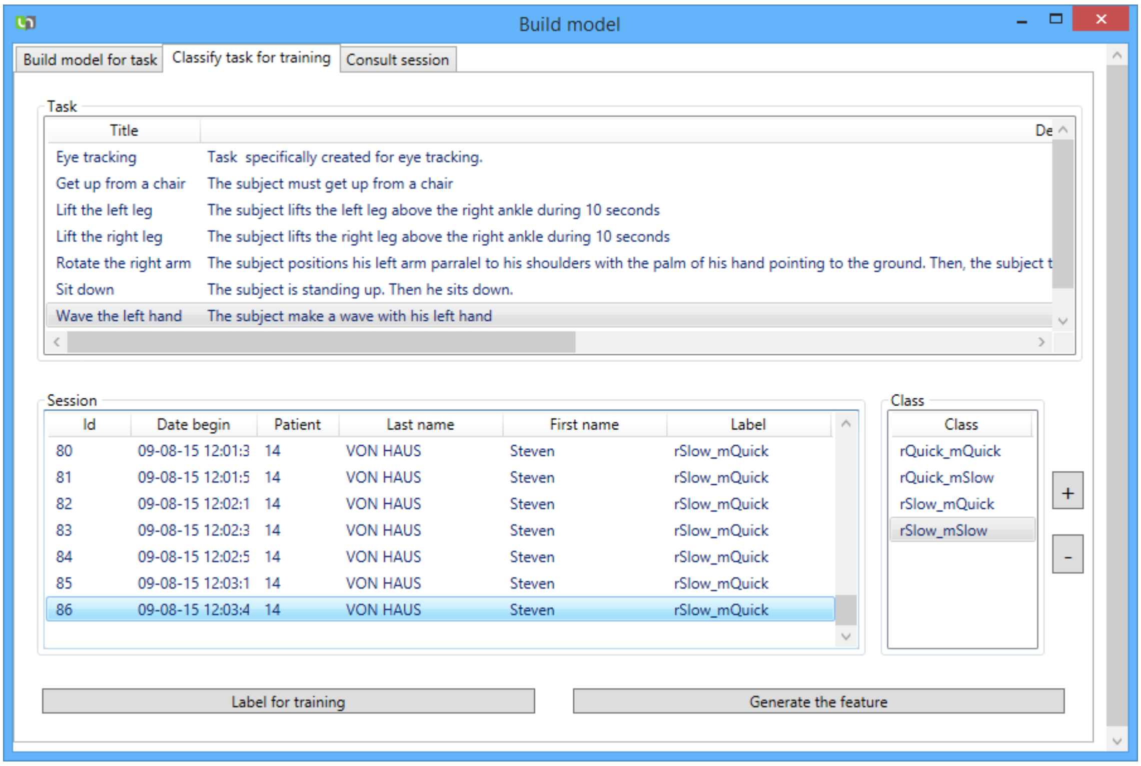
Once the data collected, the user can select the different records and visualize it as seen in Figure 2. For the eye tracking and the rotation with the right arm, there are no recognitions. The reasons are explained in **Chapter 5**. In this context, the user can delimit himself the beginning and the end of the task. However, for the other tasks, the measurements in time are automatically recorded.



Patient :
19 Position
Session :
164
Task :
Lift the right leg
Index frame :
127
Good :
???
Start pose :
???

Figure A.3: Any task can be replayed. The specialist can change the beginning and the end of the task

Figure A.4 shows the interface which allows to prepare the records for the learning phase. So, the user must select the records which will be used in the training set to generate a model able to recognize a class for a task. For each task, the user can create the different classes that must be recognized. Then, the user must labeled the records with the appropriate class and generate the features.



Task

Title	De
Eye tracking	Task specifically created for eye tracking.
Get up from a chair	The subject must get up from a chair
Lift the left leg	The subject lifts the left leg above the right ankle during 10 seconds
Lift the right leg	The subject lifts the right leg above the right ankle during 10 seconds
Rotate the right arm	The subject positions his left arm parralel to his shoulders with the palm of his hand pointing to the ground. Then, the subject t
Sit down	The subject is standing up. Then he sits down.
Wave the left hand	The subject make a wave with his left hand

Session

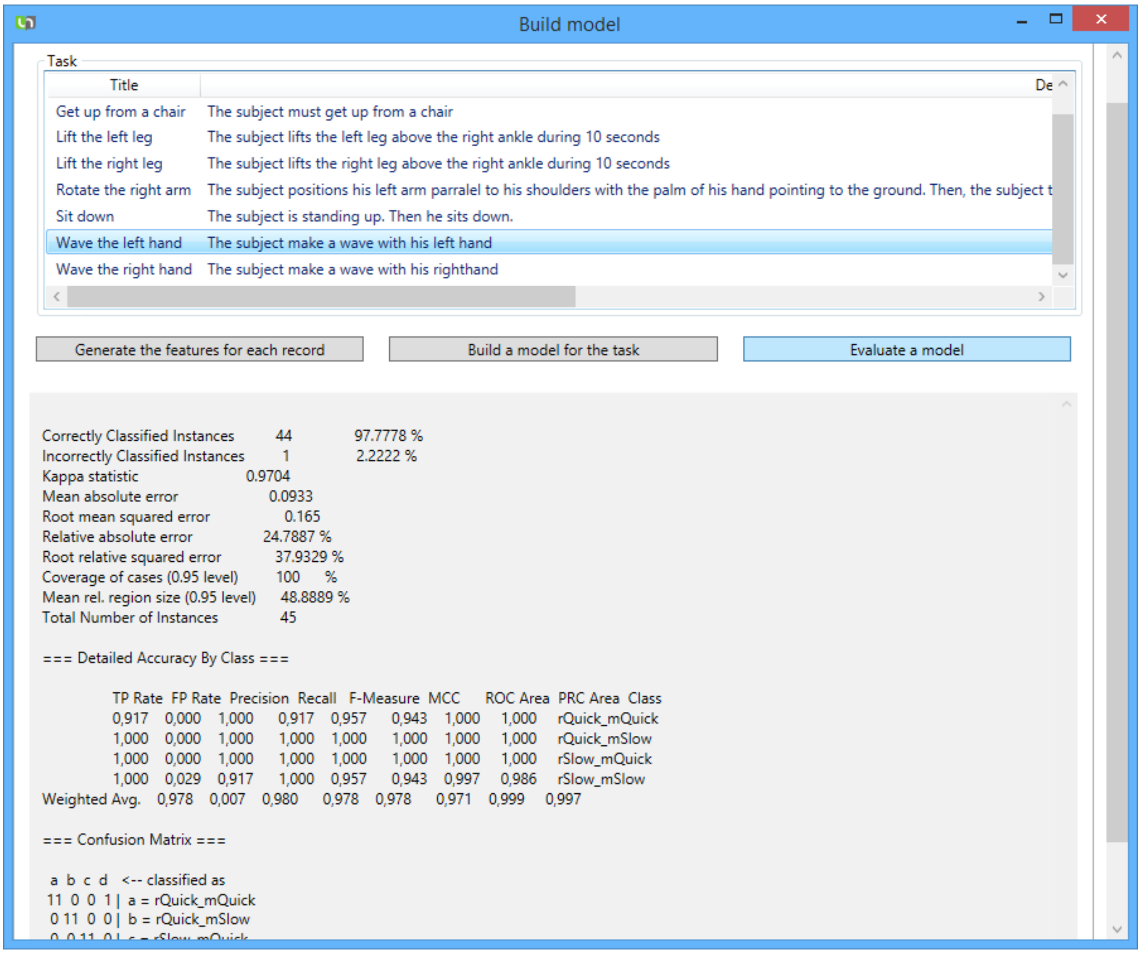
Id	Date begin	Patient	Last name	First name	Label
80	09-08-15 12:01:3	14	VON HAUS	Steven	rSlow_mQuick
81	09-08-15 12:01:5	14	VON HAUS	Steven	rSlow_mQuick
82	09-08-15 12:02:1	14	VON HAUS	Steven	rSlow_mQuick
83	09-08-15 12:02:3	14	VON HAUS	Steven	rSlow_mQuick
84	09-08-15 12:02:5	14	VON HAUS	Steven	rSlow_mQuick
85	09-08-15 12:03:1	14	VON HAUS	Steven	rSlow_mQuick
86	09-08-15 12:03:4	14	VON HAUS	Steven	rSlow_mQuick

Class

Class
rQuick_mQuick
rQuick_mSlow
rSlow_mQuick
rSlow_mSlow

Figure A.4: The tasks executed can be labeled with the appropriate class before training.

When the preparation for the learning phase is over, the model recognizing a task, can be generated. In this case, the objective is to detect psychomotor retardations and not to recognize gestures in real time. So, the approach is completely different. **Chapter 6** gives further details about the features computed but also about the classes that will be predicted by the model. The interface shown in Figure A.5 allows to generate all the features for each record of a task. Moreover, a model can be built. The user selects the meta-parameters to use during the learning phase an generate the model with the labeled records. Finally, the model can also be evaluated with the application. Actually, the results are corresponding to the output given by Weka during the evaluation.

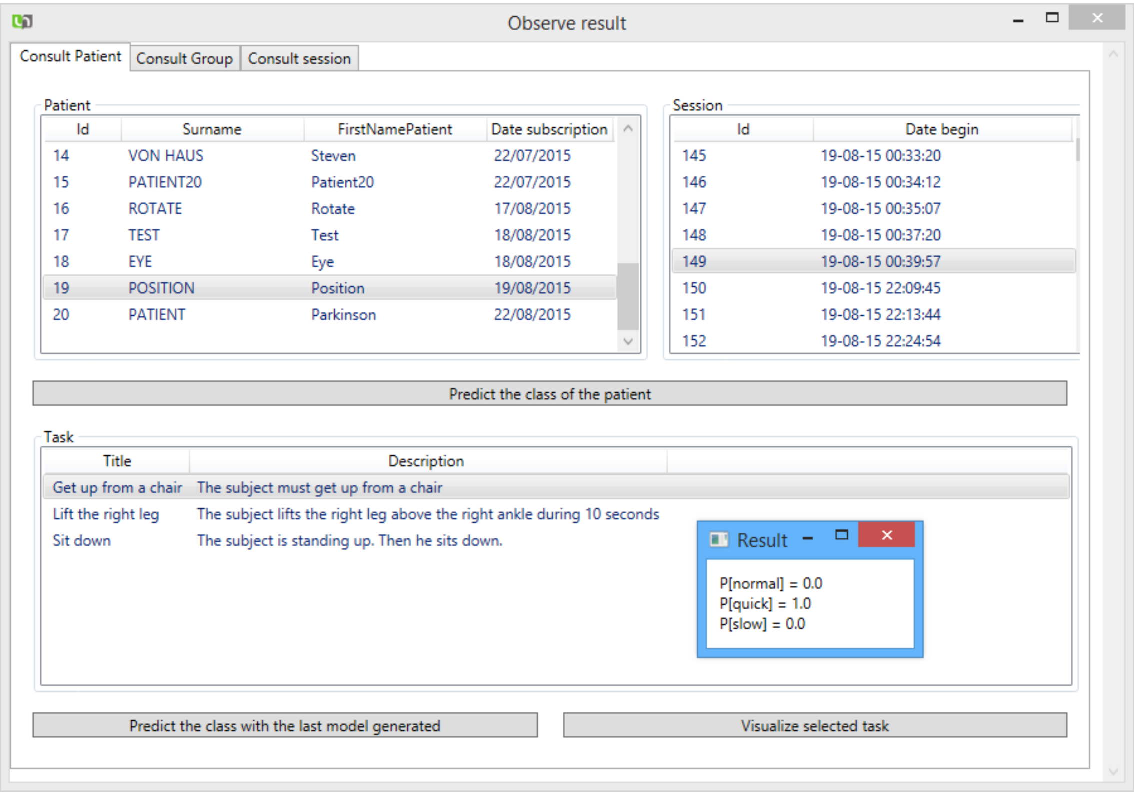

Build model
Task
Generate the features for each record
Build a model for the task
Evaluate a model

Title	De
Get up from a chair	The subject must get up from a chair
Lift the left leg	The subject lifts the left leg above the right ankle during 10 seconds
Lift the right leg	The subject lifts the right leg above the right ankle during 10 seconds
Rotate the right arm	The subject positions his left arm parralel to his shoulders with the palm of his hand pointing to the ground. Then, the subject t
Sit down	The subject is standing up. Then he sits down.
Wave the left hand	The subject make a wave with his left hand
Wave the right hand	The subject make a wave with his righthand

```
Correctly Classified Instances        44           97.7778 %
Incorrectly Classified Instances       1            2.2222 %
Kappa statistic                     0.9704
Mean absolute error                   0.0933
Root mean squared error                0.165
Relative absolute error              24.7887 %
Root relative squared error           37.9329 %
Coverage of cases (0.95 level)        100      %
Mean rel. region size (0.95 level)     48.8889 %
Total Number of Instances             45

=== Detailed Accuracy By Class ===

              TP Rate  FP Rate  Precision  Recall  F-Measure  MCC     ROC Area  PRC Area  Class
              0,917    0,000    1,000      0,917   0,957      0,943   1,000     1,000     rQuick_mQuick
              1,000    0,000    1,000      1,000   1,000      1,000   1,000     1,000     rQuick_mSlow
              1,000    0,000    1,000      1,000   1,000      1,000   1,000     1,000     rSlow_mQuick
              1,000    0,029    0,917      1,000   0,957      0,943   0,997     0,986     rSlow_mSlow
Weighted Avg.  0,978    0,007    0,980      0,978   0,978      0,971   0,999     0,997

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
 11  0  0  1 |  a = rQuick_mQuick
  0 11  0  0 |  b = rQuick_mSlow
  0  0 11  0 |  c = rSlow_mQuick
```

Figure A.5: All the features of the records can be generated for a task. Then the task can be built and evaluated.

Figure A.6 shows the prediction made by the model generated for a specific task with a new unseen data. When a patient is selected, all his sessions are displayed. Then the users selects a task in the session. This task is sent to the last learned model generated for this specific task. As a result, a window showing the probability distribution appears. In this case, we can see that the task has been classified as “quick”. In the future, the main objective would be to classify the patients from the results given by the classification of the tasks, but also with the other variables discussed in **Chapter 1**.



Id	Surname	FirstNamePatient	Date subscription
14	VON HAUS	Steven	22/07/2015
15	PATIENT20	Patient20	22/07/2015
16	ROTATE	Rotate	17/08/2015
17	TEST	Test	18/08/2015
18	EYE	Eye	18/08/2015
19	POSITION	Position	19/08/2015
20	PATIENT	Parkinson	22/08/2015

Id	Date begin
145	19-08-15 00:33:20
146	19-08-15 00:34:12
147	19-08-15 00:35:07
148	19-08-15 00:37:20
149	19-08-15 00:39:57
150	19-08-15 22:09:45
151	19-08-15 22:13:44
152	19-08-15 22:24:54

Title	Description
Get up from a chair	The subject must get up from a chair
Lift the right leg	The subject lifts the right leg above the right ankle during 10 seconds
Sit down	The subject is standing up. Then he sits down.

P[normal] = 0.0
P[quick] = 1.0
P[slow] = 0.0

Figure A.6: The class of an unseen record is predicted by the model specifically generated for the task.