View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Repositorio Institucional Universidad de Málaga

Architecture for neurological coordination tests implementation

Michel Velázquez-Mariño[1,2], Miguel Atencia[1], Rodolfo García Bermúdez[3,2], Francisco Sandoval[4], and Daniel Pupo-Ricardo[2]

[1] Universidad de Málaga, Departamento de Matemática Aplicada, Málaga, Spain,
mvquezm@uma.es,matencia@ctima.uma.es
[2] Universidad de Holguín, Grupo de Procesamiento de Datos Biomédicos (GPDB), Holguín, Cuba,
depupor@uho.edu.cu
[3] Universidad Técnica de Manabí, Departamento de Informática y Electrónica, Ecuador,
rodgarberm@gmail.com
[4] Universidad de Málaga, Departamento de Tecnología Electrónica, Málaga, Spain,
fsandoval@uma.es

Abstract. This paper proposes a generic architecture for devising interactive neurological assessment tests, aimed at being implemented on a touchscreen device. The objective is both to provide a set of software primitives that allow the modular implementation of tests, and to contribute to the standardization of test protocols. Although our original goal was the application of machine learning methods to the analysis of test data, it turned out that the construction of such framework was a pre-requisite to collect enough data with the required levels of accuracy and reproducibility. In the proposed architecture, tests are defined by a set of stimuli, responses, feedback information, and execution control procedures. The presented definition has allowed for the implementation of a particular test, the Finger-Nose-Finger, that will allow the exploitation of data with intelligent techniques.

Keywords: Software architecture, touchscreen devices, neurological tests, machine learning, Finger-Nose-Finger test.

1 Introduction

In this paper we propose a framework for the implementation of neurological tests to assess coordination diseases, such as Parkinson's and cerebellar ataxia. The final aim of this research is the assessment of the disease stage, as well as the recommendation of specific test protocols, through the analysis of test results by computational intelligence methods. We have found that the lack of standardized test protocols and software platforms is a severe obstacle for such objective, thus the motivation to undertake the proposed architecture.

Nowadays the presence of touchscreen devices is pervasive in health-related applications such as monitoring personal activity [8], at-home assessment of

health variables [6], and support to diseases diagnosis [12,17]. Touchscreen devices are also introduced in hospitals for patient monitoring, diagnostic evaluations, and interfacing to larger information systems [9]. The most common touchscreen devices are smartphones and tablets and, in particular, devices with Android Operating System (OS) are very popular. Undoubtedly, the reduced cost, size, and power consumption are key factors in this upsurge. Similar reasons can be given for the inclusion of built-in electronic components into different *mHealth* [5,11,18] applications.

Several tests have been developed for or adapted to touchscreen devices, simplifying the communication between users and computing systems. In the coordination assessment area, which plays an important role in the neurological examination, touchscreen devices have been used to record the movement of upper limbs, in evaluation tasks such as circle tracing [12], Archimedean Spiral [10,13], and handwriting/drawing [4]. The obtained accurate data about the disease progression contribute to the design of objective clinical scales with high sensitivity, the evaluation of therapies, and the exploration of early symptoms.

Although several tests have been proposed and validated for ataxia assessment, they are configured with different application protocols, implemented on heterogeneous devices, and based upon disparate techniques for recording the results [1,2,3,7,15]. To the best of our knowledge, no module or architecture has been proposed to standardize the implementation of neurological coordination tests using touchscreen devices and external sensors. This lack of standardization is a severe hindrance to the accurate analysis of collected data, as well as to the development of new coordination tests.

Here we present a modular and portable architecture to implement coordination tests, focused on a main touchscreen device (tablet) that can integrate different external devices (sensors, programmable boards, etc.). The control device manages all tests execution, acquires data from its own sensors (touchscreen, accelerometer, etc.) and from external devices, and integrates and saves data. The main elements that such an architecture must manage are the following:

- Data from touchscreen devices and external sensors.
- Stimuli events (visual, sound and haptic) and feedback information.
- Control logic of the implemented neurological tests.
- Data persistence.

The Finger-Nose-Finger (FNF)—often used in neurological examinations and evaluated in previous work [14,16]—is here implemented using the proposed architecture. All along the paper, this test is used as a proof of concept, by presenting a detailed definition of each one of its elements in the architecture.

In Section 2 we present the basic definitions that shape the implementation of a generic neurological test, resulting in a key module within the whole framework. The interaction between tests, sensors, and graphical user interface is explained in Section 3, by providing a global view on the proposed architecture. Although the framework is intended to be portable, some remarks are pointed out in Section 4 regarding implementation requirements. Finally, in Section 5 the conclusions and directions for future research are discussed.

2 Coordination tests

In this section we propose the general scheme of a neurological coordination test. Our goal is to improve the accuracy of data acquisition by minimizing the dependence on specialists' subjectivity. Many of the neurological coordination tests can be automated, thus we construct the necessary infrastructure; in doing so, our final aim will be to approach a standardization of tests that will be both general and flexible to accommodate a wide range of tests.

From the engineering point of view, we grouped the tests in touch-screen-pure tests and touch-screen-non-pure tests. The former only need the touchscreen device for its execution, whereas the latter need more than one device, thus making its design more complex. Even if a test proper does not use the touchscreen to show stimuli or capture responses, this device is used to show information about the test, to control the execution and to manage data captured. Despite differences among tests, there is a set of common characteristics presented in all of them, namely stimuli, responses, feedback information and execution control:

Stimuli are signals provided by the system to the user, intended to generate a definite response. They can consist of visual, audible or haptic events. They are always present in coordination tests, since at least the start stimulus must exist, if the user's expected actuation can be assumed from the context. In general, more than one stimulus occurs. The general execution of the test, the start, the end, and the rhythm of expected tasks are guided by stimuli, which are the reference to assess the timing and accuracy of responses.

Responses comprise the user's actuations driven by stimuli. They are captured using different devices, such as touchscreens, inertial sensors, keyboard buttons, etc. Responses can be continuously polled at a specific sample rate or consist of discrete events depending on subjects' performance velocity, usually guided by either constant or fixed stimuli, respectively. There may also exist other kind of responses, e.g. the occurrence (or not) of a certain action during a time interval, or other more specific accomplishments of subjects. More complex tests can present all of these responses types.

Feedback information acts as an assessment, just after the response, and is usually aimed at both showing a reference and helping to improve response accuracy. Just like stimuli, it can include visual, audible and/or haptic signals. Feedback information allows to explore different neurological behaviours, for instance improvement due to learning or deterioration caused by fatigue. It can be enabled or disabled in the same test depending on the examination goal, e.g. when trials are intended to be independent.

Execution control defines the flow of test actions, by specifying the sequence of stimuli, response measurements, and feedback signals. In addition, this module may pre-process responses and saves all results. The execution control is the kernel of the tests, and it must guarantee that all functionalities occur exactly on time, avoid delays, and accurately perform data acquisition.

Based on these common characteristics, we can represent the neurological test execution in a block diagram (Figure 1). When a test starts, the execution

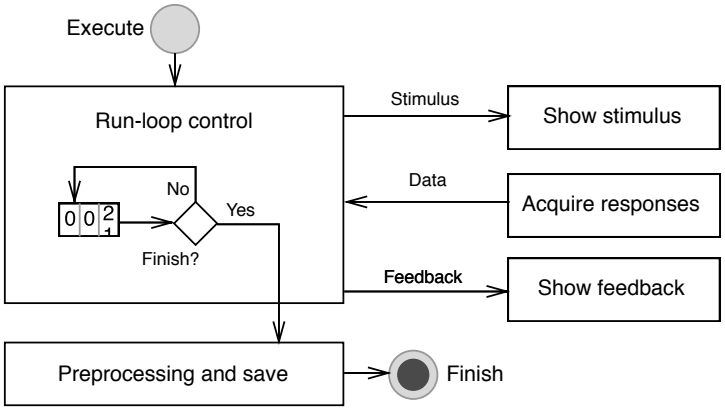

Fig. 1. Execution model of a coordination test.

control block activates a run-loop that owns the precise definitions of the test execution protocol, so as to launch stimuli and feedback information when appropriate. Each neurological test is executed by means of the necessary routines, which can be grouped into five categories: show stimuli, response acquisition, response pre-processing, provide feedback information, and save data. Not always the execution order matches this enumeration, since some complex tests may include repetitions, iterations, and back tracking, so the run-loop is far from linear. Coordination tests start by a stimulus, and finish when some condition is reached. Some common stop conditions include timeout reached, number of responses achieved, number of correct responses achieved, number of wrong responses achieved, and other more specialized conditions for specific tests. All stop conditions are managed by a run-loop controller. The run-loop has a state machine to execute the above mentioned five routines in a coordinated way, as well as to finish the test execution when a stop condition occurs.

In coordination assessment, a session often combines one or several tests into a configuration that we call *evaluation protocol*. For instance, the same test can be configured as a sequence of short repetitive tasks, to evaluate the capacity to learn specific simple movements; or repeated without feedback in long repetitive tasks, in order to train some more complex responses; or a protocol designed to globally evaluate the coordination state could include a combination of different tests comprising non repetitive tasks. The philosophy of defining protocols is intended to guarantee the extensibility of the proposed architecture, by providing constructive blocks that can serve as modules of sophisticated configurations.

As a proof of concept, we have defined all the elements that are necessary to implement the FNF test. A particular protocol, that is often found in clinical practice, is configured as three slightly modified repetitions of the test. The stimulus consists in a red cross drawn on the screen. Driven by the presentation of such visual stimulus, the subject is expected to touch the red cross, so that

the responses include the point actually touched (if any) and the time spent. After the user touches the screen, a yellow circle is drawn on the screen just at the place of the patient's response, providing visual feedback information. The stop condition is defined as ten taps, i.e. each one of the three instances of the test ends when the subject touches the screen ten times. In the first test instance, the stimulus will be placed on the centre of the screen. In the second test occurrence, the stimuli will alternatively be placed at each corner of the screen, moving to the next position after each response, following a clock wise rotation. In the third repetition, the stimulus position will be randomly chosen with a uniform distribution over the screen area. The proposed scheme allows the clinician to easily select a pre-defined protocol, or even to design new protocols by combining and defining different tests.

3 Architecture

In this section we describe the global architecture, which is focused on two critical elements: the execution control and the sensor data acquisition. The execution control manages the stimuli and the feedback information, according to the test definition as described in the previous section. Depending on the kind of stimuli and the feedback information, different low-level routines must be executed, such as simple or complex draw procedures, audio playback, writing to communication ports, and so on. The sensor data acquisition handles the reading of sensors in the main device via *character devices* and of external sensors via communication ports. The punctual execution and coordination of all these tasks, at the instants defined by the test, is crucial for accuracy of results.

The functionality of both execution control and data acquisition is implemented by an architecture that is supported by three levels, namely Graphical User Interface, Protocol Management and Test Execution, as shown in Figure 2.

3.1 Graphical User Interface

The Graphical User Interface (GUI) is a communication layer between users (medical staff and subjects) and the Protocol Management. Routines in this layer are aware of entities defined by the context, which are involved in the neurological evaluation, such as subject and evaluation clinical protocol. This layer implements functionalities to allow the clinician to select those contextual entities and introduce other important information such as hand used, eyes state (open/close), study type (training/evaluation/rehabilitation), study title or required comments. It is the responsibility of this layer to call the correct protocol type and transfer to Protocol Management all information to perform the evaluation.

Following our implemented example, in the Graphical User Interface implementation for the FNF protocol we present on the screen a widget that allows the user to select basic contextual data of the test, namely:

- The hand that will be used in the execution of tests, left or right.

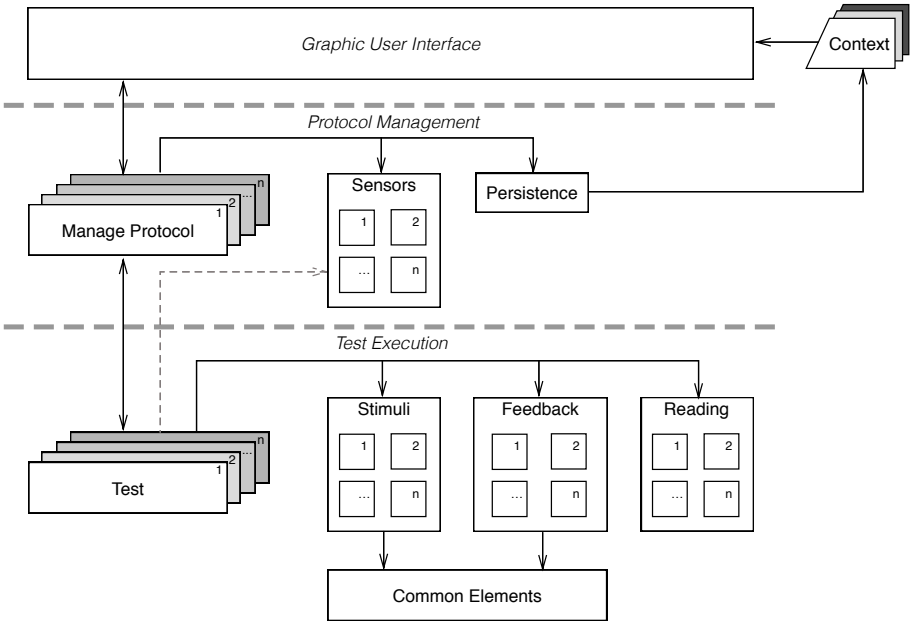

Fig. 2. Architecture diagram.

- Basic personal data of the patient.
- Choice of protocol, as described in the previous section.
- Type of protocol: *evaluation* in this case.

3.2 Protocol Management

The layer Protocol Management implements the mechanisms to control the evaluation protocol execution, manage communication with the main device sensors and external sensors, and save study results. For each evaluation protocol, there exists a block named Manage Protocol$_i$, $i = 1, 2, \ldots, n$, which controls the execution of the protocol by calls to one or more associated tests from the Test Execution layer.

The execution of an evaluation protocol begins with a call from the GUI to the related Manage Protocol block. A test start screen is shown via GUI using the associated test block in the Test Execution layer. The corresponding Manage Protocol routine dispatches the next test to be executed in the Test Execution layer, releases the control and waits for the end of the current running test. At the end of each test, pre-processing routines are called to pack test results and the control is returned to the Manage Protocol routine to dispatch the next test. Finally, when an evaluation protocol is complete, Persistence mechanisms are called to save the results of evaluation protocol performance. Inside this layer, there is an additional Block named Sensors comprising sub-blocks Block$_i$,

$i = 1, 2, \ldots, m$ that group the routines to use the involved sensors. Sensors can be heterogeneous with specific commands and different protocol communications.

The process just described can be better understood with the implemented example. In the case of the FNF test, the test start screen shows summary information about the test and a countdown to prepare the subject and foster their concentration. Then, the Manage Protocol block shows stimuli, measures the subject's responses and shows feedback information when appropriate. The need for result pre-processing is apparent when observing that subjects often lack accuracy touching the target, and then they try to correct their responses with fast taps around the target. Therefore, when the test ends, a pre-processing routine must split the correction touches from the valid response ones. Regarding the usage of sensors by the FNF test, the current implementation is only based upon the touchscreen, but in a more advanced version, we plan to include inertial sensors to assess the tremor and other variables related to movement.

3.3 Test Execution

The Test Execution is the layer that owns the mechanisms to actually perform the tests, by implementing the logic of each test. Tests are represented in the diagram by blocks named Test_i, $i = 1, 2, \ldots, t$, each one consisting of different stimuli, feedback information and response acquisition mechanisms. They also run with different execution logic, by defining a sequence of tasks, an end condition, and related information about the execution, as described in Section 2.

Each Test block interacts with three blocks named Stimuli, Feedback, and Reading, all three in turn comprising the same number t of sub-blocks. Thus, a sub-block Stimuli_i groups the routines that show the stimuli required by the particular i-th test, combining visual, sound and/or haptic signals, either on the touchscreen or through external devices. The Stimuli blocks must account for different screen drawing mechanisms (raster, double-buffer, zone painting, etc.) as well as implement independent threads for read/write operations on communication ports. Stimuli timing is implemented in the Test either by defining them as constant stimuli, or through triggering mechanisms, such as time intervals or subject responses. Each sub-block Feedback_i provides feedback information related to the i-th test, if any is required. When feedback is defined, it is always driven by the subject's responses, thus it can be seen as an stimulus with a fixed trigger mechanism, with the same implementation recommendations. Therefore, the Stimuli and Feedback blocks share a block named Common Elements, which groups basic routines to paint, write information or make calculations used by both blocks. Finally, each sub-block Reading_i contains data acquisition routines, both from the main device or external sensors. Depending on the test type, reading subject's responses can proceed by interruptions or by constant sampling. In order to prevent data loss, reading raw values from the main routine upon every interruption is recommended, whereas in the latter case secondary threads should be implemented to avoid latency in the reading process.

Using the FNF test as example, we can distinguish the following elements. The stimuli consists in painting on the touchscreen a red cross at the position

defined by the configuration of the test. For instance, in the first test iteration explained above, the cross position is at the screen centre. The transition to the next stimulus is triggered by the subject response action on the touchscreen. The feedback information is also a draw action, namely to draw a yellow circle at the position of the subject's response, and hide it after two seconds. Finally, data acquisition is accomplished by reading all events on the touchscreen, getting the position of touch or touches, and recording the timestamp of the event.

4 Implementation

The proposed framework is mostly implementation-independent, so the paper does not aim at a detailed description of software coding and used hardware. However, we include in this section some comments on the requirements of both software and hardware for a successful implementation of the architecture.

4.1 Base classes

Although the proposed architecture was designed to be general and independent of any programming language or technology, we do consider that an object oriented methodology is required to achieve the objectives of modularity and reusability. Consequently, we have coded a number of classes to implement the main functionalities of the architecture. These classes are shown in Figure 3, following the *Unified Modeling Language*™, by using generic references of names and data types to make possible any adaptation.

The **ProtocolExcBase** is the base class to implement the specific protocol execution classes. This class has the *runControl* method to control the execution of a set of tests using a state machine implementation. The enumeration **ProtocolState** defines the states of the state machine and the attributes *mProtocol* and *mCurrentTest* provide information about the order of tests execution. The *start* and *end* methods are used to switch between tests, and *saveResults* allows to access the persistence functionalities. We have implemented a protocol for the FNF test, by constructing a new class FNFProtocolExc that inherits from ProtocolExcBase, and then performs the custom implementations.

The class **GenericSensor** is the base class to implement the attributes and hight level methods needed to use sensors that read and write data. Specific sensors class implementations stem from the extension of this class. Methods in this class are invoked from the protocols execution classes, whereas the TestExcBase class is allowed the control of sensors by means of references. The current implementation of our example case, the FNF test, makes no use of specific sensors, but a planned advanced version will use inertial sensors, thus requiring the construction of a class that will inherit from the GenericSensor class and construct its custom functionalities using the read and write methods of the parent class.

The actual execution of tests will be performed by classes that inherit the basic implementation of the class **TestExcBase**, which comprises a set of attributes and methods according to the theoretical definition of a general test,

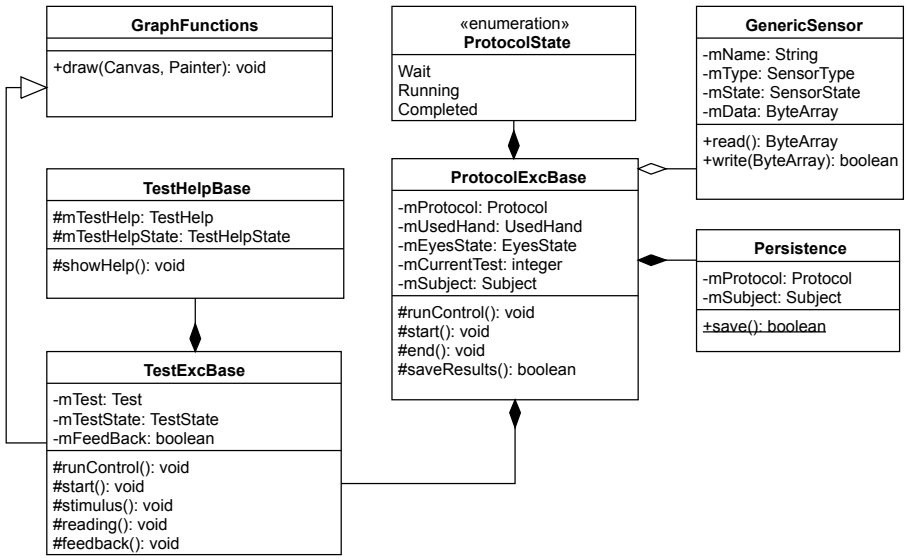

Fig. 3. Base classes diagram.

given in Section 2. We have also included a **TestHelpBase** class to show some explanations and help of the related test, which can be helpful in some complex tests. All draw operations on the touchscreen of the main device are managed by the class **GraphFunctions**. Drawing techniques are strongly dependent on the programming language and the technology, however this class must provide, at least, high level methods to draw basic shapes (circle, cross, square, text, etc.), whereas other drawing complex tasks should be implemented by the specific TestExc class that requires them. In our example, the FNFTestExc class has been built by inheriting from TestExcBase, and then implementing the custom elements described along the paper, such as execution control routines, stimuli routines, and other test elements.

4.2 Acquisition system

As mentioned above, the core device of the architecture is the touchscreen. Sensors may either be integrated in the main device (the touchscreen itself, accelerometer, gyroscope, light sensor, etc.) or be externally developed on programmable boards such as Arduino, Raspberry pi, cc2640, etc. Communication between the main device and external sensors should be wireless, using Bluetooth, Low Energy Bluetooth, or other wireless technology, in order to ease the subject's performance, minimize wires, and isolate the subject from the main device. Also, each associated sensor must have a trust security code to be connected to the main device. The whole hardware setting is shown in Figure 4.

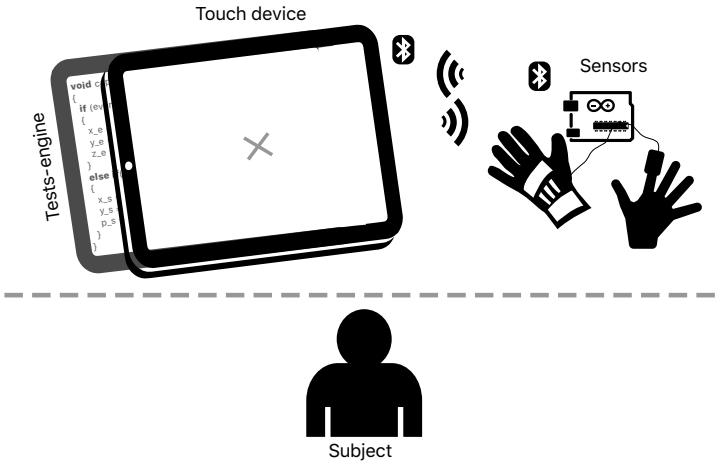

Fig. 4. Architecture environment, showing the main touchscreen device with management acquisition routines, associated sensors and a subject interacting.

In order to achieve the planned functionalities, some requirements are specified on the touchscreen: more than seven screen inches, a capacitive touchscreen with five or more activation points, double core processor, at least one gigabyte of Random Access Memory, and integrated sensors (accelerometer, gyroscope, light sensor, etc.). Additional requirements could be demanded by specific tests. In the main device, the kernel of the OS is responsible for managing interruptions of the integrated sensors. The raw values of sensors are available to the *user space* through *system call interfaces*. For example, when the kernel receives a touchscreen interruption, it writes the raw data (touch position, touch pressure, touch timestamp, etc.) of the event on a *character device*. The routines in the *user space* can then access such character device using *standard libraries*. We propose implementing buffering mechanisms and multi-threads to avoid samples loss. Any OS that includes these mechanisms is, in principle, compatible with the proposed architecture, the most usual being Android, Windows, OS X, iOS and GNU/Linux.

A *beta* version of the end-to-end architecture has been built as a proof of concept, implementing the FNF test. The application was developed in the language C++ using the Qt framework. All described base classes were implemented to provide structures and high level methods to ease the construction of other coordination tests. This implementation is cross-platform and can be ported to Linux, OS X, Windows, and Android OSs. During the actual tests with subjects, only the Android version was used in order to ensure the homogeneity in the experiments. The characteristics of the experiments used to evaluate the implementation and the usefulness of our proposal can be found in [14,16].

5 Conclusions

We have proposed a generic framework for the implementation of neurological tests that are used to assess coordination of movements. The architecture relies on the identification of the key characteristics of a test, as well as the notion of protocol, which is a sequence of one or more tests. The aim of the proposal is twofold: on the one hand, it provides a modular software infrastructure that can be used to improve automation of tests; on the other hand, it is contribution towards the standardization of testing protocols within the clinical community.

The primary goal of our work is the application of computational intelligence techniques to the analysis of test results, but it turned out that this goal was hindered by the lack of both standard clinical protocols and software libraries. This paper is thus a first step towards the integration of machine learning algorithms within neurological tests. In particular, we are currently developing a layer of cloud services that will gather information from the execution of tests in mobile devices. The massive collected data will be processed to recommend specific tests to patients, according to the progression they show at each performed protocol, or even to improve the configuration of protocols.

Acknowledgement

This work has been partially supported by the Universidad de Málaga, as well as the Universidad de Holguín through the joint project titled *"Mejora del equipamiento para la evaluación de la rehabilitación de enfermedades neurológicas de especial prevalencia en el oriente de Cuba"*.

References

1. Gagnon, C., Mathieu, J., Desrosiers, J.: Standardized finger-nose test validity for coordination assessment in an ataxic disorder. The Canadian Journal of Neurological Sciences. Le Journal Canadien Des Sciences Neurologiques 31(4), 484–489 (2004)
2. Gavriel, C., Thomik, A.A.C., Lourencco, P.R., Nageshwaran, S., Athanasopoulos, S., Sylaidi, A., Festenstein, R., Faisal, A.: Kinematic body sensor networks and behaviourmetrics for objective efficacy measurements in neurodegenerative disease drug trials. In: 2015 IEEE 12th International Conference on Wearable and Implantable Body Sensor Networks (BSN). pp. 1–6 (2015)
3. Harish, K., Rao, M.V., Borgohain, R., Sairam, A., Abhilash, P.: Tremor quantification and its measurements on parkinsonian patients. In: International Conference on Biomedical and Pharmaceutical Engineering, 2009. ICBPE '09. pp. 1–3 (2009)
4. Lefebvre, O., Riba, P., Gagnon-Marchand, J., Fournier, C., Fornes, A., Llados, J., Plamondon, R.: Monitoring Neuromotricity On-line: a Cloud Computing Approach. In: Rémi, C., Prévost, L., Anquetil, E. (eds.) Drawing , Handwriting Processing Analysis : New Advances and Challenges. 17th Biennial Conference of the International Graphonomics Society, Pointe-à-Pitre, Guadeloupe (2015)

5. Lennon, T., Bernier, T., Tamayo, D., Goldberg, C., Mankodiya, K.: Multi-sensory system for monitoring dyskinesia in movement disorders. In: 2015 41st Annual Northeast Biomedical Engineering Conference (NEBEC). pp. 1–2. IEEE (2015)
6. Memedi, M., Nyholm, D., Johansson, A., Palhagen, S., Willows, T., Widner, H., Linder, J., Westin, J.: Validity and responsiveness of at-home touch-screen assessments in advanced Parkinson's disease. IEEE journal of biomedical and health informatics (2015)
7. Notermans, N.C., van Dijk, G.W., van der Graaf, Y., van Gijn, J., Wokke, J.H.: Measuring ataxia: quantification based on the standard neurological examination. Journal of Neurology, Neurosurgery, and Psychiatry 57(1), 22–26 (1994)
8. Park, K., Lim, S.: A multipurpose smart activity monitoring system for personalized health services. Information Sciences 314, 240–254 (2015)
9. Paschou, M., Sakkopoulos, E., Tsakalidis, A.: easyHealthApps: e-Health Apps Dynamic Generation for Smartphones & Tablets. Journal of Medical Systems 37(3), 9951 (2013)
10. Pullman, S.L.: Spiral analysis: a new technique for measuring tremor with a digitizing tablet. Movement disorders 13 Suppl 3, 85–89 (1998)
11. Salarian, A., Russmann, H., Wider, C., Burkhard, P.R., Vingerhoets, F.J.G., Aminian, K.: Quantification of tremor and bradykinesia in Parkinson's disease using a novel ambulatory monitoring system. IEEE transactions on bio-medical engineering 54(2), 313–22 (2007)
12. Say, M.J., Jones, R., Scahill, R.I., Dumas, E.M., Coleman, A., Santos, R.C.D., Justo, D., Campbell, J.C., Queller, S., Shores, E.A., Tabrizi, S.J., Stout, J.C., TRACK-HD Investigators: Visuomotor integration deficits precede clinical onset in Huntington's disease. Neuropsychologia 49(2), 264–270 (2011)
13. Surangsrirat, D., Thanawattano, C.: Android application for spiral analysis in Parkinson's Disease. In: 2012 Proceedings of IEEE Southeastcon. pp. 1–6. IEEE (2012)
14. Velázquez-Mariño, M., Atencia, M., García-Bermúdez, R., Pupo-Ricardo, D., Becerra-García, R., Velázquez-Pérez, L.: Contribution analysis of extracted variables from Finger-Nose-Finger test to the classification of SCA2 patients. In: III International Conference on Informatics and Computer Sciences. Havana, Cuba (2016)
15. Velazquez-Perez, L., de la Hoz-Oliveras, J., Perez-Gonzalez, R., Hechavarria, P.R., Herrera-Dominguez, H.: Quantitative evaluation of disorders of coordination in patients with Cuban type 2 spinocerebellar ataxia. Revista De Neurologia 32(7), 601–606 (2001)
16. Velázquez-Mariño, M., Atencia, M., García-Bermúdez, R., Pupo-Ricardo, D., Becerra-García, R., Pérez, L.V., Sandoval, F.: Cluster Analysis of Finger-to-nose Test for Spinocerebellar Ataxia Assessment. In: Rojas, I., Joya, G., Catala, A. (eds.) Advances in Computational Intelligence, pp. 524–535. No. 9095 in Lecture Notes in Computer Science, Springer International Publishing (Jun 2015)
17. Westin, J., Dougherty, M., Nyholm, D., Groth, T.: A home environment test battery for status assessment in patients with advanced Parkinson's disease. Computer Methods and Programs in Biomedicine 98(1), 27–35 (2010)
18. World Health Organization: mHealth: New horizons for health through mobile technologies. In: Second global survey on eHealth. Geneva (2011)