COBEM2015

ABCM Associação Brasileira de Engenharia e Ciências Mecânicas

23rd ABCM International Congress of Mechanical Engineering
December 6-11, 2015, Rio de Janeiro, RJ, Brazil

A DISTRIBUTED GAME SYSTEM FOR ROBOTIC TELEREHABILITATION

Leonardo José Consoni
Wilian Miranda dos Santos
Adriano Almeida Gonçalves Siqueira
University of São Paulo
consoni_2519@hotmail.com, wilianmds@yahoo.com.br, siqueira@sc.usp.br

Abstract. *Recent technological advances have allowed experiments with the use of robots in motor rehabilitation therapies for victims of fractures or neurological diseases. These experiments, in turn, open the possibility of using other new techniques, like virtual reality and telecommunication/teleoperation. This paper presents the development of virtual computer games for robotic rehabilitation of lower limbs, as well as of an infrastructure that allows its use in a distributed manner, across a computer network, for interaction, at a distance, between therapist and patient or between multiple patients. All tools used in the implementation of this network communication system for multiplayer games are described. At the end, some collected results of system usability and performance are shown and necessary improvements are discussed.*

Keywords: *rehabilitation robotics, computer network, serious games, telerehabilitation, distributed system*

1. INTRODUCTION

Neurological diseases and orthopedic fractures are among the health problems that most affect people all over the world, causing variable levels of motor impairment. Their consequences involve not only a drop in quality of living for the victim, but also a huge economic cost for society, due to the loss of productive capacity and spending on the provision of medical care and rehabilitation treatments — costs that, combined, were estimated in over US$ 50 million on United States in 2004. The incidence of these problems is higher in older age groups, above 65 years, and as this part of population is becoming more numerous, by the global increase in life expectancy, the quantity of individuals more likely to suffer from them grows (Mackay and Mensah, 2004; Krebs *et al.*, 2008).

Among the neurological diseases, cerebrovascular accident (CVA or stroke), in particular, is one of the three major causes of mortality and disability in the world. Despite the preventive work carried out in some countries, the global number of stroke cases per year - nowadays at 15 million - keeps increasing, due to the worldwide tendency of population ageing, combined with the proliferation of other risk factors, such as high blood pressure, sedentary lifestyle and bad eating habits. Also, even if improvements in first aid and further treatment today allow the survival of about 2 thirds of the victims, there is still no guarantee that this growing number of survivors does not end up suffering from some motor sequel, which occurs for half of them (Mackay and Mensah, 2004).

Consequently, there is an increasing demand for neuromotor therapy, requiring more of healthcare services, particularly public, already quite burdened in many countries. And so, the motivation is created for the search of new forms of treatment to replace or supplement conventional methods, and return the treated person to their usual activities, faster, with fewer resources and in conditions similar to those previous to the disease, minimizing the impacts of these health problems in economic activities (Krebs *et al.*, 2008).

Within this context, the application of robotics — previously seen only as an assistive tecnology — in motor rehabilitation therapies is becoming an increasingly common researh topic in recent times (Krebs *et al.*, 2008; Marchal-Crespo and Reinkensmeyer, 2009; Cao *et al.*, 2014). The use of robots (as end effectors, orthoses or exoskeletons) as a complementary rehabilitation procedure offers theoretical advantages as the lower number of professionals needed in a treatment session, better control of the aid or resistance force applied to the treated person during exercises and greater precision in measuring its performance (muscle force, range of movement, etc.).

In addition to the direct impact on the efficiency of conventional therapy, the presence of a robot that exchanges information — for control and monitoring — with a computer system opens up new possibilities for enhancing treatment, like logging of data from multiple sessions (for more objective evaluation of patient evolution), remote communication, to overcome physical barriers and still allow interaction of therapist with patient (something known as telerehabilitation), and use of virtual reality, the latter possibly in the form of *videogames*, such as in Gonçalves (2013, 2014).

Virtual environments can play the role of presenting the objectives to be achieved by the user in a more intuitive way. Visual and sound elements of a game, for example, besides suggesting actions to the players, can change according to their actual actions, keeping them aware of their performance and focused on the task at hand, possibly entertained, which is important on the rehabilitation case, considering that motivation is a key factor for a good recovery. In addition to the real-time audiovisual response, the patient immersion — and consequent dedication to the exercise — can be raised through a tactile feedback (aiding or resisting the its motion) provided by the robotic device.

These *Serious Games*, classified that way for having goals beyond entertainment, allow an interesting new form of interaction between different patients, for competitive or cooperative gameplay, something already proposed in Andrade (2013); Novak *et al.* (2014) and whose benefits on the motivation of the player has been observed for local matches (individuals located at the same room) (Novak *et al.*, 2014).

This paper shows the development of a new distributed computing system for studying the application of robots and computer games on motor rehabilitation of post-stroke patients. Combined with network (*Ethernet* or *Wi-Fi*) communication technologies, the system adds capabilities of telerehabilitation and remote multiplayer activities, whose effects are also to be studied. These applied methods were tested for robustness when dealing with information transmission delays.

The work presented here extends and utilizes the previous developments made at the Robotic Rehabilitation Laboratory of the School of Engineering of São Carlos (ReRob), which involved the creation of therapeutic robotic devices and its control strategies.

The paper is organized as follows: Section 2 presents the used robotic devices, developed in the ReRob laboratory; Section 3 shows the tools for creation of rehabilitation games; Section 4 describes the system communication structure and its functioning; Section 5 explains how some preliminary performance tests were carried on; Section 6 presents and discusses the results from these tests; and Section 7 briefly exposes future perspectives and improvements to be made.

2. ROBOTIC DEVICES FOR LOWER LIMBS REHABILITATION

2.1 Custom hardware

The rehabilitation devices developed at ReRob, Fig. 1, are designed for different forms of treatment of hemiparesis — one of the common stroke sequelae, that compromises the walking hability. The main common characteristic between them is the use of impedance control, implemented by a combination of series elastic actuators (SEA) — one for each device's degree-of-freedom (DoF) — and their control algorithms, processed by a real-time computer system. The adoption of impedance control allows better handling of interaction forces between the device and the patient, resulting in a safe interface (Jardim *et al.*, 2012).

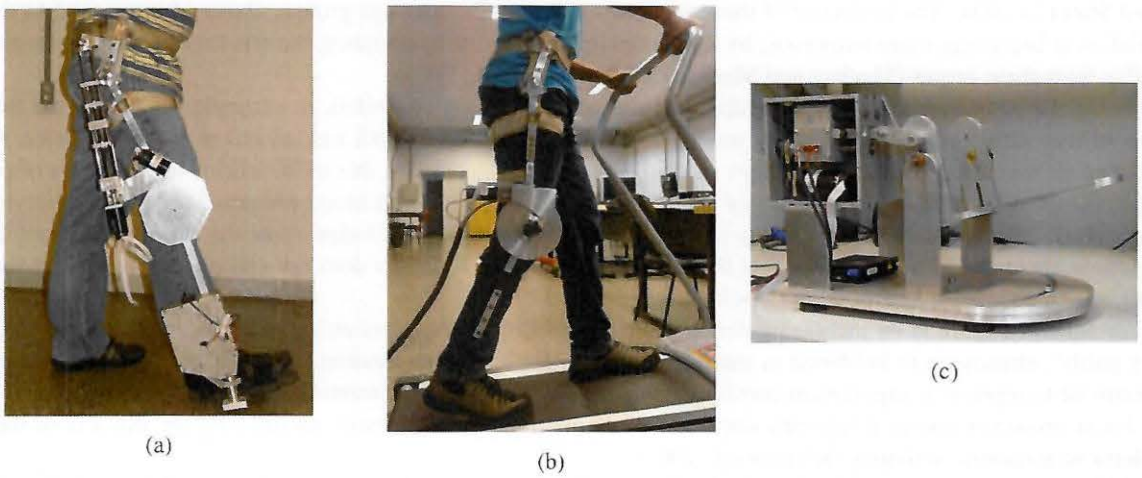
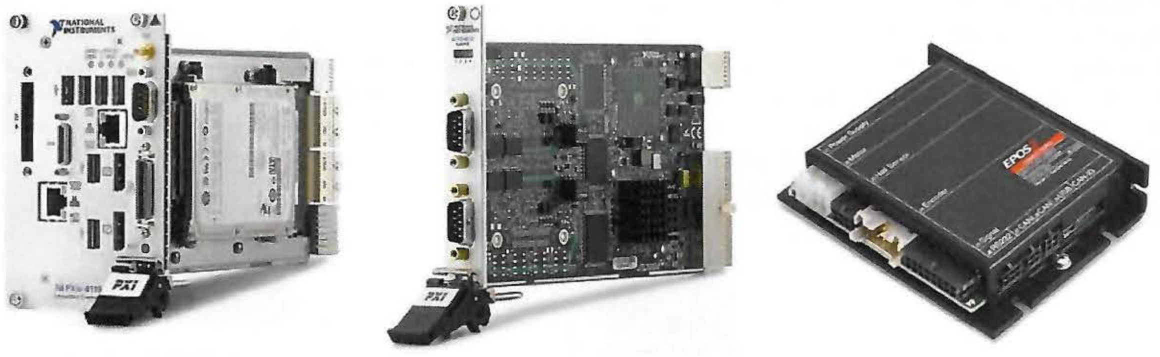


Figure 1: ReRob rehabilitation devices: (a) Exo-Kanguera (Jardim *et al.*, 2012); (b) knee orthosis (Santos and Siqueira, 2014); (c) RePAiR (Robotic Platform for Ankle Rehabilitation) (Gonçalves, 2013, 2014).

2.2 Control Interface

The connection between the controller *hardware* — a *National Instruments PXIe-8115* module, Fig. 2a, running the Phar Lap real-time operating system (RTOS) — and each SEA is made through a CAN network board (*National Instruments PXI-8512/2 HS*, Fig. 2b), which binds to the interfaces (*EPOS Maxon Motor Drive*, Fig. 2c) of each actuator. On the software level, communication is performed with the CANOpen protocol (specification available at: http://www.can-cia.org), using the *NI-XNET* library, included by the board manufacturer on its *LabWindows/CVI RT* development environment, that also provides other libraries — and its own *ANSI C89* compiler — for real-time processing and data aquisition with *National Instruments* equipments. The main data exchange code is based on the work developed in Fernandes (2013).



(a) (b) (c)

Figure 2: Specialized hardware used in the control and data acquisition: (a) NI PXIe-8115 RT module and (b) NI PXI-8512/2 HS CAN board (Source: http://www.ni.com/products/pt); (c) EPOS2 24/5 (5A, 11-24 VDC) (Source: http://www.maxonmotor.com/maxon/view/content/EPOS-Detailsite).

2.3 Control Strategy

The impedance control of the exoskeleton joints follows an assistance-as-needed (AAN) algorithm: the robot doesn't make the entire required movement, encouraging the patient to complete it. The level of aid (torque) to be offered is estimated primarily based on the difference between actual and desired angular positions over time, from which the necessary effector stiffness, damping and velocity are calculated (Santos and Siqueira, 2014). That way, the reference value (setpoint) and the measurements from a movement axis must be updated and provided for each of its control passes. Also, the desired maximum impedance of the robot can be externally set too, making it more or less compliant overall.

3. SERIOUS GAMES FOR REHABILITATION

Being constantly aquired and sent to a computer, the information about patient movements — joints position, velocity, torque, etc. — can be used as commands for a virtual game.

For creating these games, in the present case, the *Unity3D* engine (Official site: http://www.unity3d.org) editor was chosen. By using a game engine, several lower-level features — graphics renderer, shaders, sound, I/O, scripting, cross-platform support — are promptly available, simplifying the game development and allowing a greater focus on improving its suitability for rehabilitation therapies. In addition, the modularity of code components provided by *Unity3D* allows much of the work can be reused in other similar projects, but with different gameplay proposals.

For the game logic programming, the *C#* language was used (*Unity3D*'s editor supports scripts written in C#, *JavaScript* and *Boo*, a *Python* variant), due to its better integration with the engine. The graphical elements rendered in the virtual environment (tridimensional meshes and textures), when not provided by the editor itself, were created in *Blender* (3D modeling program available at: http://www.blender.org) and *Gimp* (image editor available at: http://www.gimp.org) open source softwares.

4. DISTRIBUTED SYSTEM INTERNAL COMMUNICATION

4.1 Components and structure

The RTOS used by the controller is not suited for graphical (non command-line) applications, and it is not supported by *Unity3D*'s projects. Even if a conventional operating system, tweaked for low-latency control (like *Linux RT*), was used, the software and hardware requirements for a 3D game and a low-level controller application would still be considerably different, so it would make sense to keep these components running on different machines.

With the adopted system's distributed architecture design, there was a need for an inter-process communication (IPC) mechanism that works across devices, for information exchange between the involved applications.

A comprehensive solution that allows local (in the same machine) or remote (across different machines) communication is achieved through network sockets. The BSD sockets are a universal interface for communication over IP protocol, originally developed in C for the Berkeley Distribution [Operating] System, and later ported to many other operating systems (*Windows*, *Linux*, *Mac OS*, etc.) and programming languages (like the *C#* used in *Unity3D*). By setting up the addresses for sending and receiving data in each process, a connection can be created between them for transmission of byte sequences (Hall, 2011).

The intent to enable remote interaction between many game instances and online visualization of patient data for the therapist, through a graphical user interface (GUI), adds some complexity that would scale quickly as more applications were plugged in the system, if they all needed to communicate with each other. To alleviate the problem, a client-server model (similar to Andrade (2013)) was adopted, with central applications — master servers — for both orthoses control and multiplayer games handling, to which the other programs could connect to send and receive the necessary data.

Figure 3 shows the resulting structure of data transfer across the system. TCP socket connections — slower but more reliable — are used for information passed only a few times, and TCP clients can swith between the available servers; UDP sockets — faster but with no guarantees on data integrity — are used to transmit values that need to be updated continuously, with clients remaining always plugged in the same server.

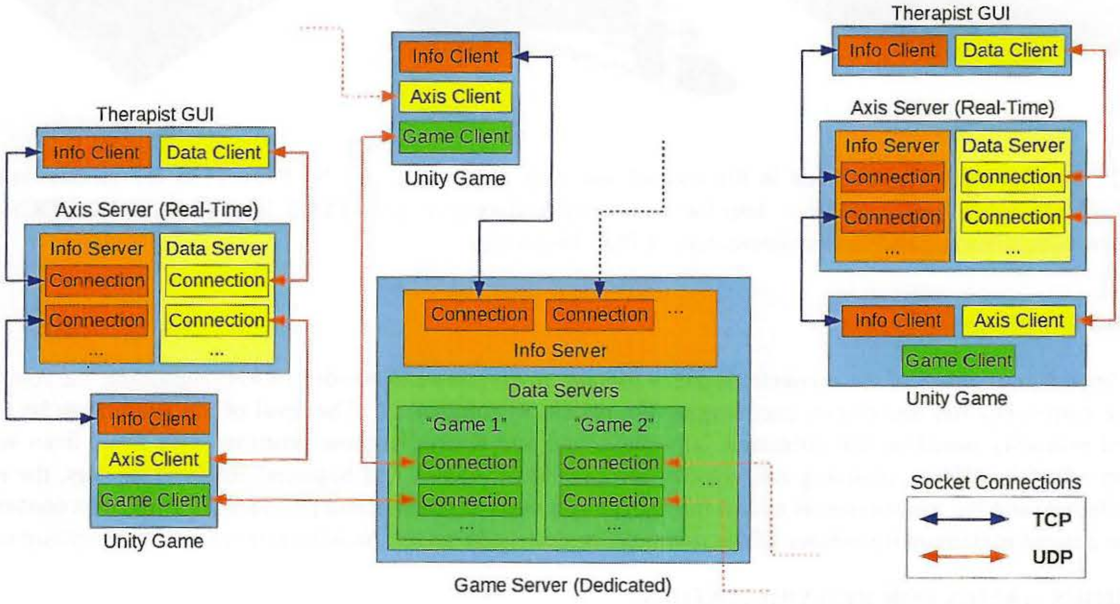


Figure 3: Diagram showing possible communication channels across the distributed system components. *Unity Game* instances connected to the same game data server are on a remote multiplayer match, while the ones not connected like this could be only on single player or local multiplayer activities.

The applications apart from *Unity Game* and *Axis Server* (coupled with the real-time control) were coded in a combination of interpreted and compiled languages. *Python* scripts that perform some simple operations run on an interpreter embedded in a C program. The C code implements the lower level and more complex features (mainly asynchronous socket communication), which are exposed to the scripts through the *CPython* API (Documentation available at: https://docs.python.org/3.4/c-api). Also *Python* Tkinter module (based on the TK toolkit) was used to build the therapist interface (Fig. 4).

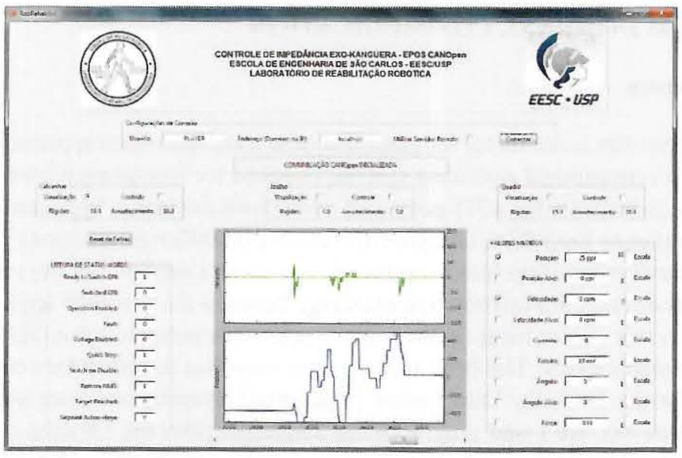

CONTROLE DE IMPEDÂNCIA EXO-KANGUERA - EPOS CANOpen
ESCOLA DE ENGENHARIA DE SÃO CARLOS - EESC/USP
LABORATÓRIO DE REABILITAÇÃO ROBÓTICA
EESC • USP

Figure 4: Therapist GUI, built in TK, showing the available controllers and data visualization graphs and boxes.

4.2 Communication protocol and messages structure

The server applications are supposed to be constantly running on their host machines, available to client requisitions. When a client is instantiated (running game or therapist GUI tries to connect), each server answers it according to its role in the system and to the communication channel used:

- In the event of a TCP connection of a client to *Axis Server*, the request is answered only one time with a list of all axes — motors controlled by the *Axis Server* host — names whose data is available. After that, the same communication channel can be used by the client for operating the axes and by the server to report back their state changes. These informations are passed as string messages, respectively in the format:

$$< Axis\ ID\ [0] >:< Axis\ Name\ String\ [0] >:< Axis\ ID\ [1] >: ...$$
$$< Axis\ ID >:< Axis\ Enable\ Boolean >:< Axis\ Reset\ Boolean >$$
$$< Axis\ ID\ [0] >:< Axis\ Enabled\ Boolean\ [0] >:< Axis\ Failure\ Boolean\ [0] >:< Axis\ ID\ [1] >: ...$$

- The TCP connection of a game with *Game Server* gets back a list with information of multiplayer matches available to the patient. The same channel and message format can be used to request the establishment of a new match:

$$< Match\ Name\ [0] >:< Match\ Port\ [0] >:< Match\ Name\ [1] >: ...$$

- The data exchange of *Axis Server* with its clients through UDP sockets consists of broadcasting axes measurements and receiving the position, velocity and time setpoints from game instances, for impedance control and haptic (force) feedback calculation of particular axes. String messages are built in the form:

$$< Axis\ ID\ [0] >:< Axis\ Position\ [0] >:< Axis\ Velocity\ [0] >:< Axis\ ID\ [1] >: ...$$
$$< Axis\ ID >:< Axis\ Position\ Setpoint >:< Axis\ Velocity\ Setpoint >:< Axis\ Time\ Setpoint >: ...$$

- For its UDP clients, *Game Server* acts only as an intermediate, passing information about elements positions and velocities in one game instance to the others, possibly running in distant computers, connected to the same port:

$$< Element\ ID\ [0] >:< Element\ Position\ [0] >:< Element\ Velocity\ [0] >:< Element\ ID\ [1] >: ...$$

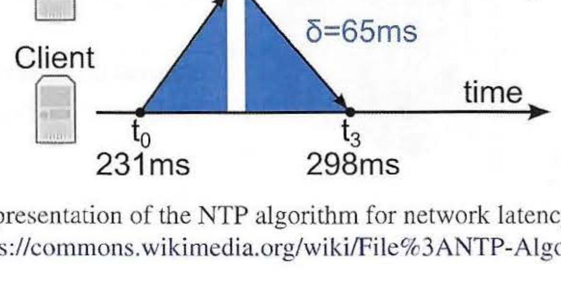
All message sizes are limited to 256 bytes of data, which is small enough to prevent big packet losses (Hall, 2011).

4.3 Delay handling

Every data transfer over IP protocol — on Ethernet or Wi-Fi networks — has an intrinsic and variable lag between information dispatch and arrival times, depending on distance, quality of physical instalation, etc. When combined with control algorithms, these delays can introduce instabilities that make the system unusable or limit its performance. That way, strategies should be implemented to handle them and minimize its effects.

In the case presented, the performance bottleneck happens on UDP communications, as they transfer information that is time related, and with faster update rate.

Proper correction of received values depends on the knowledge of the communication delay amount at the time of message arrival, which can not be measured directly, by the difference between system times of the two machines involved, because the ticks count of their processors does not necessarily occur with the same reference (*epoch*) and frequency (clock). As a solution, the latency θ is continuously estimated by a formula based on the Network Time Protocol (NTP), designed for time synchronization between computers (Mills, 2006), as described in Fig. 5.



$$\delta = (t_3 - t_0) - (t_2 - t_1)$$
$$\theta = \delta/2$$

Figure 5: Representation of the NTP algorithm for network latency estimation
(Source: https://commons.wikimedia.org/wiki/File%3ANTP-Algorithm.svg).

For instantaneous values, updated more often (Δt interval), the time estimate is used to calculate the compensated joint position (P_{new}) and speed (V_{new}), by extrapolating the curve of the last states (P_{last}, V_{last}) received, Eq. 1, in combination with a Kalman filter.

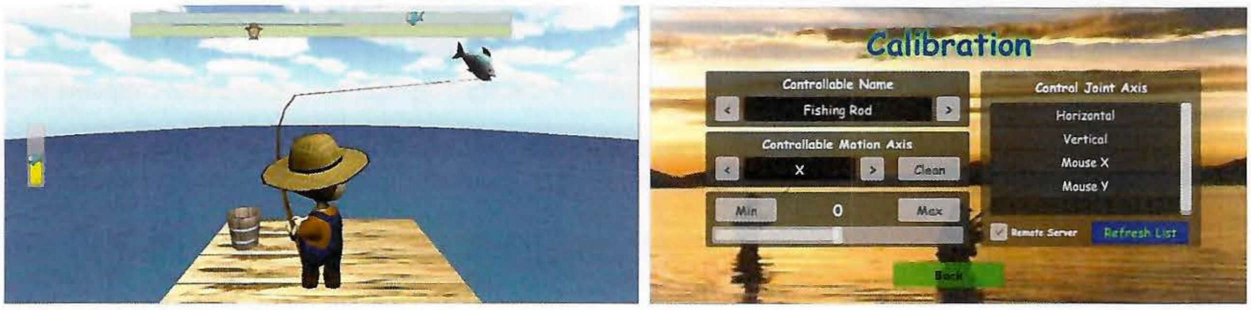
$$V_{new} = \frac{P_{new} - P_{current}}{\Delta t}, \quad with \; P_{new} = kalman(P_{last}, \Delta t) + kalman(V_{last}, \Delta t) \cdot \theta \tag{1}$$

The feedback setpoints ($P_{setpoint}$, $V_{setpoint}$), more spaced out (ΔT update interval) and ahead in time, are interpolated with a more complex function for $P(t)$, Eq. 2.

$$P(t) = \frac{(V_{setpoint} - V_{new})}{2 \cdot (\Delta T - \theta)} \cdot t^2 + V_{new} \cdot t + P_{current}, \quad with \; V_{new} = \frac{P_{setpoint} - P_{current}}{2 \cdot (\Delta T - \theta)} + \frac{V_{current}}{2} \tag{2}$$

5. TESTS METHODOLOGY

For initial system viability tests, The proof of concept game, named *Pescaria* (*The Catch*), was created. In this example game (Fig. 6a), the goal of moving some of the exoskeleton's joints — selected and calibrated in a dedicated screen (Fig. 6b) — is presented to the player as a fishing activity, in which is necessary to follow the movement of the hooked fish, pulling it at the correct times, that are indicated on the screen. The motion of one chosen axis controls the horizontal position of the fisherman's focus, and other controls — only when possible — the line pulling.



(a) (b)

Figure 6: Screens from the example game ("Pescaria"): (a) Gameplay; (b) Calibration

Besides, for this test, only the horizontal movement of the game, more steady, was used. This motion was controlled with the knee orthosis previously presented, which has a rotational SEA.

Figure 7 shows in a simplified way how the system components were arranged for the test.

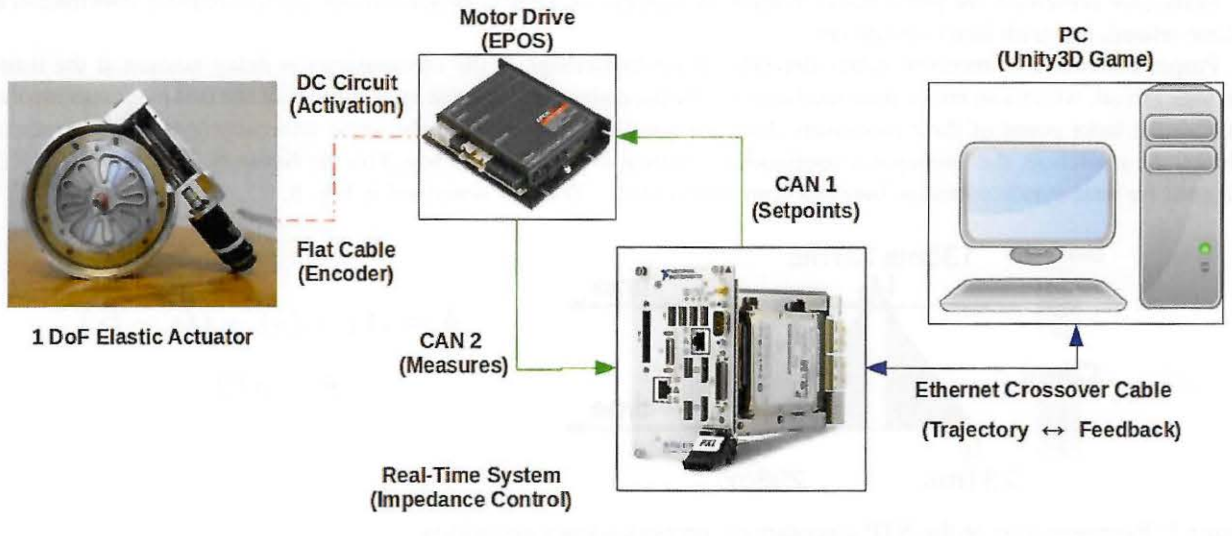


Figure 7: Representation of hardware setup for the performed tests.

To evaluate the performance of the compensation algorithms used, measurements were performed for two tests:

- Compensation for control $\rightarrow$ game communication, with measured instantaneous values of position and speed sent to the game every 5 milliseconds; As the network delays measured with the used hardware — direct connection between machines through *crossover* cable — proved too small to have its effects well observed, an artificial lag of 100 milliseconds was added for every message;
- Reconstruction of the desired trajectory curve (game $\rightarrow$ control communication), for setpoints sent every 50 ms, representing a state also 50 ms in the future.

6. RESULTS AND DISCUSSION

The obtained results were plotted on *Scilab* numerical computation software (Available at: http://www.scilab.org).

On Figure 8, representing the control $\rightarrow$ game communication, the blue line indicates the axis positions in the moment of measurement, which are sent and arrive on the other machine after some time (green line). According to every transmission delay, the correction of received values is performed, simultaneously generating the red curve.

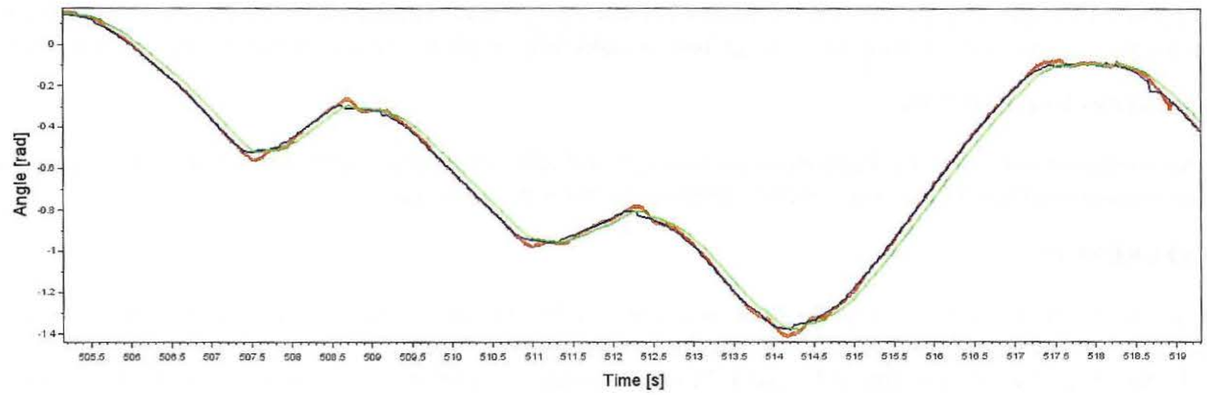


Figure 8: Comparison between real axis position values, on the moment of sending (blue) and arrival (green) and simultaneous game element positions (red), based on predictions to compensate network delays.

As it can be seen, the latency estimation works in making the controlled game element position follow the instantaneous measurements curve more closely than the delayed one, as it is received on the client (game) socket.

In the case of feedback, Fig. 9, the blue curve represents the original values of the reference trajectory, some of which are transmitted and reconstructed (green line) in the receiving machine, based on the latency estimate made. The red line represents the measured position of the robot, which follows the reference using position and impedance control (in this case, with a constant effector stiffness value of 25 N·m).

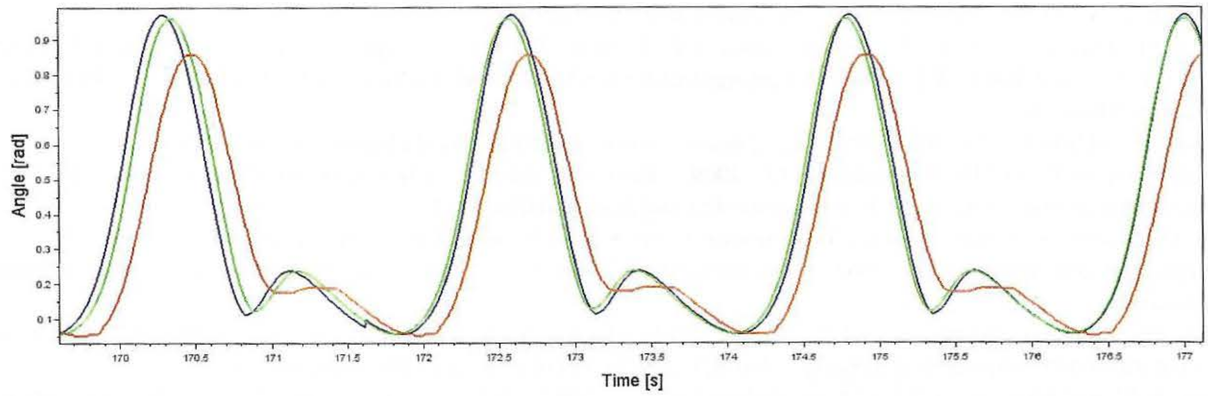


Figure 9: Comparison between control original setpoints curve (blue), spline reconstructed setpoints curve (green), and actual joint positions (red), for an actuator stiffness of 25 N·m.

Due to the delay estimation not being totally accurate, it can be seen that at times the calculated value of θ is larger than the real one, and the reconstructed curve precedes the original. For this test, it had no significant impact, since the delays are small, and the robot control — which doesn't follow the generated reference completely, as the effector is not rigid — itself introduces a delay, staying behind the reference.

7. CONCLUSIONS AND FUTURE WORK

As a research tool for rehabilitation studies, the system presented here proves itself viable and potentially useful. However, its value on proper robotic therapy will only come from the application of an adequate rehabilitation strategy, that potentializes motor learning and recovery on impaired patients, but that also could be achieved using the said system as a test platform. In addition, at the same time these methodologies are developed, the tools could be improved, as they have room for improvement.

For now, patient's effort (that have to be maximized to promote neuromuscular rehabilitation) are only estimated through position error relative to the reference trajectory, but error calculation does not distinguish disability from unwillingness, that would require different levels of robot assistance. As a possible solution, electromyography (EMG) looks promising for measuring muscle activation (that could indicate intention of movement), but, for stroke patients, eletric signals from spasms or voluntary contraction cannot be easily decoupled, so there is still a need to find a reliable way of using it in this cases (Cao *et al.*, 2014).

Considering the crucial problem of communication delays, the current compensation algorithms would benefit from being compared to — or even replaced by or combined with — more well proven concepts, like *wave variables* (Niemeyer and Slotline, 2004), that ensure teleoperation systems passivity and stability even with heavy timing lags.

Aside from that, it would be interesting to enable, in the rehabilitation game, elements to be controlled by a combination of joints, by implementing some sort of direct kinematics (and inverse kinematics for feedback), as it's noticed that motor learnig is more based on endpoint — hand, foot — tasks than on single articulation movements (Hogan, 2014).

8. ACKNOWLEDGEMENTS

This work is supported by São Paulo Research Foundation (FAPESP) under grant 2014/25828-4 and Coordination for the Improvement of Higher Level Personnel (CAPES) in the form of scholarships.

9. REFERENCES

Andrade, K.O., 2013. "Rehabilitation robotics an serious games: An initial achitecture for simultaneous players". In *Biosignals And Biorobotics Conference*. Rio de Janeiro, Brazil.

Cao, J., Xie, S.Q., Das, R. and Zhu, G.L., 2014. "Control strategies for effective robot assisted gait rehabilitation: The state of art and future prospects". *Medical Engineering & Physics*.

Fernandes, G., 2013. *Exploração de ambientes não estruturados através de manipulador robótico implementando controlador de impedância com parâmetros variáveis*. Master's thesis, University of Sao Paulo, Sao Carlos, Brazil.

Gonçalves, A.C.B.F., 2013. "Development and evaluation of a robotic platform for rehabilitation of ankle movements". In *22st International Congress Of Mechanical Engineering*. Ribeirão Preto, Brazil.

Gonçalves, A.C.B.F., 2014. "Serious games for assessment and rehabilitation of ankle movements". In *IEEE 3rd Int. Conf. on Serious Games and Applications for Health*. Niterói, Brazil.

Hall, B., 2011. *Beej's Guide to Network Programming*. Jorgensen Publishing.

Hogan, N., 2014. "Robot-aided neuro-recovery". *Mechanical Engineering*.

Jardim, B., Santos, W.M. and Siqueira, A.A.G., 2012. "Development of an exoskeleton for lower limbs rehabilitation". In *5th Workshop In Applied Robotics And Automation*. Bauru, Brazil.

Krebs, H.I., Dipietro, L., Levy-Tzedek, S., Fasoli, S.E., Rykman-Berland, A., Zipse, J., Fawcett, J.A., Stein, J., Poizner, H., Lo, A.C. and Volpe, B.T., 2008. "A paradigm shift for rehabilitation robotics". *IEEE Engineering in Medicine and Biology Magazine*.

Mackay, J. and Mensah, G., 2004. *The atlas of heart disease and stroke*. World Health Organization.

Marchal-Crespo, L. and Reinkensmeyer, D.J., 2009. "Review of control strategies for robotic movement training after neurologic injury". *Journal of Neuroengineering and Rehabilitation 2009*.

Mills, D.L., 2006. *Computer Network Time Synchronization: The Network Time Protocol*. CRC Press.

Niemeyer, G. and Slotline, J.E., 2004. "Telemanipulation with time delays". *The International Journal of Robotics Research*.

Novak, D., Nagle, A., Keller, U. and Riener, R., 2014. "Increasing motivation in robot-aided arm rehabilitation with competitive and cooperative gameplay". *Journal of Neuroengineering and Rehabilitation 2014*.

Santos, W.M. and Siqueira, A.A.G., 2014. "Robust torque control based on h-infinity criterion of an active knee orthosis". In *5th IEEE RAS&EMBS International Conference on Biomedical Robotics and Biomechatronics*. São Paulo, Brazil.

10. RESPONSIBILITY NOTICE