2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
October 11-14, 2020. Toronto, Canada

ROS-Neuro: implementation of a closed-loop BMI based on motor imagery*

Gloria Beraldo[1‡], Stefano Tortora[1‡], Emanuele Menegatti[1], Luca Tonin[1]

Abstract— **The increasing interest of the research community in the intertwined fields of brain-machine interface (BMI) and robotics has led to the development of a variety of brain-actuated devices, ranging from powered wheelchairs and telepresence robots to wearable exoskeletons. Nevertheless, in most cases, the interaction between the two systems is still rudimentary, allowing only an unidirectional simple communication from the BMI to the robot that acts as a mere passive end-effector. This limitation could be due to the lack of a common research framework, facilitating the integration of these two technologies.**

In this scenario, we proposed ROS-Neuro to overcome the aforementioned limitations by providing a common middleware between BMI and robotics. In this work, we present a working example of the potentialities of ROS-Neuro by describing a full closed-loop implementation of a BMI based on motor imagination. The paper shows the general structure of a closed-loop BMI in ROS-Neuro and describes the specific implementation of the packages related to the proposed motor imagery BMI, already available online with source codes, tutorials and documentations. Furthermore, we show two practical case scenarios where the implemented BMI is used to control a computer game or a telepresence robot with ROS-Neuro. We evaluated the performance of ROS-Neuro by ensuring comparable results with respect to a previous BMI software already validated. Results demonstrated the correct behavior of the provided packages.

I. Introduction

Brain-machine interfaces (BMIs) are systems that allow to communicate with external devices (e.g., computer applications or robotic actuators) using human neural signals. Since the end of the XX century, these technologies have seen an increased interest and a rapid development, firstly as an alternative communication channel for people with severe motor impairments, for example by typing letters on a virtual keyboard [1], [2]. Other applications of BMI include internet browser [3], navigation in virtual realities [4] and games for entertainment [5]. Recently, several studies have demonstrated the feasibility of exploiting BMI to control different kinds of robotic devices. Prototypes have been proposed to control neurorobotic applications: mobile devices for telepresence [6]–[8], semi-autonomous powered wheelchairs [9], and wearable neuroprostheses, like upper- and lower-limb exoskeletons [10], [11].

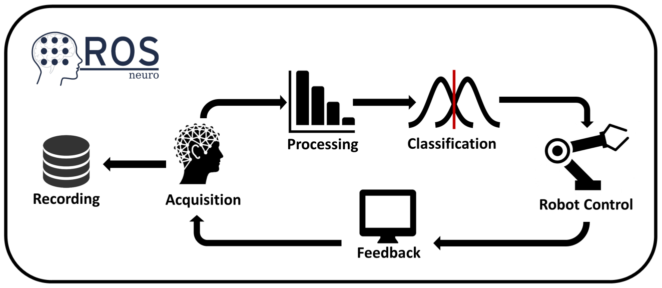

Fig. 1. A general representation of the main elements necessary to implement a BMI closed-loop controlling a robotic device in ROS-Neuro.

However, currently, most of the robotic actuators have the role to implement passively the delivered commands, allowing only a unidirectional communication from BMI to the robot as a mere end-effector. This aspect limits seriously the potentialities of the robotic side in implementing advanced behaviours. Only a few examples can be found in literature in which this limitation has been faced with more advanced control algorithms [12]–[16]. In these cases, the authors showed that a stronger interaction between the intelligence of the robot and the BMI can help the user in controlling the device.

One possible reason for the spreading of BMI systems implementing simple human-robot interaction could be that over the years researchers have been mainly focused on the BMI part by providing common tools and frameworks for signal processing and decoding. Some examples are reported in [17] including BCI2000, OpenViBE, TOBI Common Implementation Platform, BCILAB, BCI++, xBCI and BF++). Indeed, none of those has been explicitly designed to provide optimized solutions for the interaction with the robotic devices. This led not only to restrict the functionalities of the actuators; but also each research group to implement its custom interface for controlling external devices. In the last years, this approach led to the development of heterogeneous solutions and a generalized lack of standards [18].

To overcome the aforementioned limitations, in [19] we proposed ROS-Neuro, a common open-source framework for BMI and robotics research and applications. The aim of ROS-Neuro is to provide a common software framework for the implementation of BMI systems and robotic controllers, boosting their mutual integration. In this regards, ROS-Neuro relies on Robot Operating System (ROS)[1], the middleware that has become the standard *de-facto* for robotic

*Part of this work was supported by MIUR (Italian Minister for Education) under the initiative "Departments of Excellence" (Law 232/2016)

‡ The authors contribute equally to this paper

[1] Intelligent Autonomous System Lab, Department of Information Engineering, University of Padova, Padua, Italy gloria.beraldo@phd.unipd.it, stefano.tortora@phd.unipd.it, emg@dei.unipd.it, luca.tonin@dei.unipd.it

[1]https://www.ros.org/

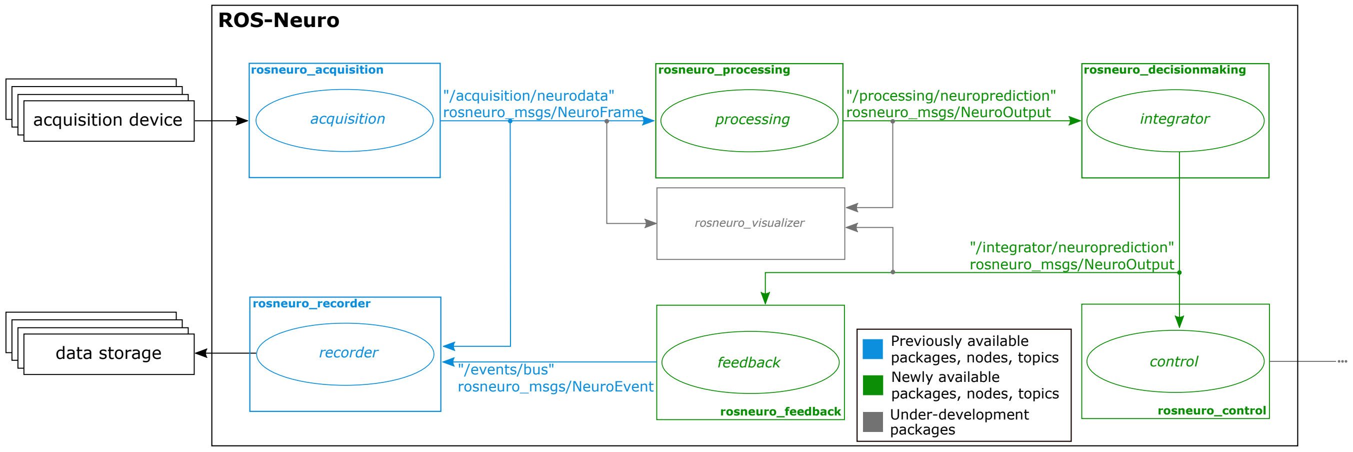

Fig. 2. Schematic representation of the current ROS-Neuro framework. The available packages, nodes, topic and messages are highlighted in green. The underdevelopment packages, tools and connections are reported in grey. The architecture is fully compliant to ROS ecosystem and it allows a transparent bidirectional communication between the BMI loop and the robotic device.

applications. The main advantage of ROS is to be an open-source project developed by a constantly growing community. Furthermore, it provides a modular multi-processing architecture and a reliable communication infrastructure that are fundamental also for BMI systems.

In [18] we have presented the general perspective of the ROS-Neuro framework and its advantages, while in [19] we proposed the first release of the acquisition and recording modules: namely the *rosneuro_acquisition* and *rosneuro_recorder* packages. These packages handle the acquisition and the recording of neural data from different commercial acquisition devices, facilitating their use among the research community.

In this work, we demonstrate the potentials of ROS-Neuro in achieving the typical BMI closed-loop, schematically represented in Fig. 1: from data acquisition and classification to the control of the external device. With this purpose, this paper presents the structure and the effective functioning of ROS-Neuro with an example of BMI closed-loop based on Sensorimotor Rhythms (SMR). We provide the first implementation[2] of the related four modules: the processing, decision making, feedback and control packages.

The paper is organized as follows: the next section will show how the packages communicate within a closed-loop BMI implemented in ROS-Neuro. Section III will be devoted to the explanation of the usage and main functionalities of the new packages implementing a motor imagery BMI. Section IV will present two case scenarios in which ROS-Neuro is used to mentally drive a brain driven system. The performance of the system will be evaluated in Section V. Finally, some conclusive remarks and guidelines for future development will be provided in Section VI.

II. The BMI closed-loop in ROS-Neuro

In this section, we present an overview of the current ROS-Neuro architecture implementing a BMI closed-loop. The system relies on a modular architecture composed of the following main modules: acquisition, recording, processing, decision making, feedback and control.

Exploiting the multi-processing functionalities of ROS, the BMI components can be implemented as stand-alone nodes working on parallel threads. A reliable communication among these nodes is granted in ROS by means of *topics*, *services* and *actions* [18]. Thus, every module of the BMI loop is interconnected with the others through a network of topics sharing different type of information (e.g., EEG data stream, predictions, events) represented by standard messages. Moreover, ROS allows to easily customize the configuration of each module of the BMI loop by setting ROS parameters in the launch file of each node or by creating a list of parameters in a YAML configuration file. ROS parameters are a great way to modify the settings since the operator can easily adapt many parameters of the BMI system according to user's preference or performance without having to re-compile the whole software.

Fig. 2 depicts a schematic representation of the ROS-Neuro framework with the proposed packages and main topics. Since the acquisition and recorder packages have been previously presented [19], herein we describe the other four proposed packages that are necessary to close the BMI loop. Although in this work we provide an example of a SMR BMI implementation, this architecture is designed to be generic and compatible with different BMI paradigms (e.g., P300).

The processing node – implemented in the *rosneuro_processing* package – has access to the stream of EEG data by subscribing to the */acquisition/neurodata* topic. Chunks of new data are received as *NeuroFrame* messages with a frame rate defined in the acquisition node as a ROS parameter [19]. The processing node is in charge to filter the incoming data in temporal, spectral and/or spatial domains. Besides, it applies common machine learning techniques to the extracted features and returns the corresponding prediction as the raw posterior probability distribution over the classes. The prediction is published on the *process-*

[2]https://github.com/rosneuro

TABLE I

DESCRIPTION OF THE NEUROOUTPUT MESSAGE

Field's name	***Datatype***	***Description***
header	Header	Timestamp and incremental identifier
softpredict	float[]	Probability distribution over classes
hardpredict	int32[]	Binary vector with 1 on the predicted class
class_labels	string	Name or label of each class
decoder_type	string	Name of the decoder used
decoder_path	string	Path to the decoder used

ing/neuroprediction topic as a *NeuroOutput* message. The structure of the message is shown in Table I. The design of the *NeuroOutput* datatype has the strength of being flexible enabling to store the output achieved with different BMI decoders (e.g., LDA, SVM, neural networks) by using a uniform representation.

It is also possible to asynchronously process and classify the neural data by calling the service */processing/classify* from other clients, such as the feedback and the control nodes. This on-demand communication method is particularly helpful for the BMI applications that are time-locked, by meaning that they need the corresponding output from the processing node at a specific time (e.g., P300 or SSVEP BMIs).

The output of the processing module can be further processed by the *rosneuro_decisionmaking* module, for instance by integrating the predictions over time, filtering the decoder output, and thus, to achieve a more stable and reliable control signal. The output is provided to the other modules by publishing again a *NeuroOutput* message, while a recognized BMI command is published as a *NeuroEvent* message [19] on the */events/bus* topic. It is worth highlighting that both the processing and the decision making modules are publishing the corresponding predictions on distinct topics through the same type of message (*NeuroOutput*), but containing different information.

The output of the BMI decoding is finally used to drive the external device, passing through a control node. The *rosneuro_control* package provides functionalities to communicate the output of the ROS-Neuro BMI to a brain driven system, implemented either in ROS or in another environment. To receive the BMI outputs, the control node may subscribe to the */events/bus* topic waiting for a command event, if a discrete control is required, or to a */neuroprediction* topic, if the control is continuous. Finally, the *rosneuro_feedback* package is used to close the BMI loop back to the user. The feedback node handles the feedback interface to provide instructions, stimuli or measurements of performance according to the selected protocol and BMI paradigm. To this aim, it receives the output of the BMI decoder by subscribing to a */neuroprediction* topic and updates its elements accordingly.

Both the neural signals and the main events, generated by the feedback node or the controlled applications are recorded by the *rosneuro_recorder* package, already presented in [19], for posterior analysis and evaluation.

III. ROS-NEURO: SENSORIMOTOR RHYTHMS BMI

Herein we will present the development of a closed-loop in ROS-Neuro for BMI applications. In particular, it implements a Sensorimotor Rhythms (SMR) BMI to control for instance a computer game and a mobile robot through the imagination of hands and feet. The aim of the following sections is to provide an explanatory example for the research community implementing their own BMI systems and paradigms within the ROS-Neuro ecosystem.

A. The processing node: usage and functionalities

The processing node has been implemented in C++ and relies on the open-source *Eigen* library[3]. This library is used for optimized linear algebra computation such as matrices, vector and the related algorithm and it is also already supported in ROS. In the current release, we propose a plugin intended for a 2-class SMR BMI. Further details about the kind of processing are presented in [20]. The *rosneuro_processing* module interfaces with a custom C++ library for processing and the FFTW 3.3.8 library[4], from which it exploits some signal processing functionalities.

Once a new *NeuroFrame* message arrives, the processing node stores the new chunk of data (samples x channels) in a ring buffer, initially empty. When the buffer is full, the processing is applied to the buffered data (buffer size x channels). In the beginning, a Laplacian spatial filter is performed. The laplacian matrix (channels x channels) is stored in an external file that can be selected using ROS configuration parameters. The Power Spectral Density (PSD) of each EEG signal is computed via Welch's algorithm with a Hamming sliding window. Specific parameters for this method can be set using ROS parameters. Finally, the online classification is performed according to a previously trained decoder, provided in input with a ROS parameter. This package allows to load and use a custom model trained outside the ROS environment, promoting the integration with other platforms (e.g., Matlab).

B. The decision making node: usage and functionalities

In the case of BMI characterised by high variability and low bit rate such as the SMR BMI, it might be worth further processing the raw predictions. With this purpose, the *rosneuro_decisionmaking* package has been designed. In the current version, the raw probabilities are integrated over the time to provide a more reliable classification. The main aim is to accumulate enough evidence of the user's intention before delivering a BMI command to the application.

In the same way of the *rosneuro_processing*, the package provides a C++ implementation, it depends on the *Eigen* library[3] and it interacts with an external custom library for processing.

[3]http://eigen.tuxfamily.org/index.php?title=Main_Page

[4]http://www.fftw.org/

The integrator node waits for *NeuroOutput* messages, integrates the raw predictions, and communicates the integrated *NeuroOutput* to the other nodes. As soon as the integrated probabilities exceed a pre-defined threshold, indicating the user intention to deliver a new command, it publishes a *NeuroEvent* message in the topic */events/bus*. The *NeuroEvent* specifies the kind of predicted command (e.g, hands or feet motor imagery) through an identifier code (int32 datatype) in the field *event*. In this way, these events can be used to trigger the delivery of the predicted command (e.g., turn left or turn right) by the control node and can be stored by the *rosneuro_recorder* module for posterior analysis.

In the current version, the decision making is based on an exponential integration function (please refer to [13] for more details) and all the configuration settings are set through ROS parameters.

In this specific implementation, once the command is delivered, the integrated probabilities have to be reset to an uniform distribution before the next prediction. The reset operation is requested externally of the integrator node through the service */integrator/reset*, to guarantee that the reset process is properly done after the BMI command has been executed. This means that the reset action is in charge to the client nodes, such as the feedback or the control, managing the translation of the predicted command to the interface and/or to the external device.

C. The feedback node: usage and functionalities

For the proposed SMR BMI, a visual feedback has been adopted. The feedback node relies on the open-source *OpenCV* library[5] available for Python 2.7. This library supports the creation of simple graphical elements–e.g., text, rectangles, circles and other geometrical shapes–and it has been selected since it is an optimized library with focus on real-time applications.

Different graphical elements are shown on a resizable window with black background according to the desired experimental protocol. The graphical elements are represented in Fig. 3 for a two-class BMI. Nevertheless, the protocol and the interface can be automatically scaled up to 5 classes by setting a configuration parameter. The timings of each protocol are provided as ROS parameters. Any change in the graphical interface can be associated with an identifier code and it can be communicated through a *NeuroEvent* message published on the */events/bus* topic. These events enable to trigger updates in the interface and they can be saved by the *rosneuro_recorder* for posterior analysis. For the SMR BMI, three different protocols have been implemented:

1) *calibration*: the protocol is used to acquire the data that are needed for training the classifier. The protocol is organized in a set of trials presenting in sequence a fixation cross (Fig. 3A), a color-coded cue of the motor task the user should imagine (Fig. 3B), and its corresponding bar that fills up with the passage of time (Fig. 3C). The node publishes the *NeuroEvent* related to each part of the protocol;
2) *evaluation*: the protocol is used to evaluate a previously trained classifier. The structure of its trials is similar to the *calibration* trials, but in this case, the bars are filled up according to the outputs of the classifier (*/integrator/neuroprediction* topic);
3) *control*: the protocol has been implemented to provide a visual feedback while the user is controlling the external device with the *rosneuro_control* package. Since the user is autonomously controlling the BMI, no fixation cross nor cue are provided, while the bars are filled up according to the outputs of the classifier, as in the *evaluation* protocol.

[5]https://opencv.org/

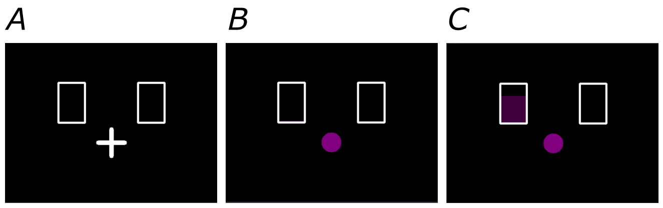

Fig. 3. Graphical elements available in the *rosneuro_feedback* package for a two-class SMR BMI.

D. The control node: usage and functionalities

In the *rosneuro_control* package, we implemented the nodes that are necessary to control the brain driven applications, described in the following sections.

On the one hand, if the controller of the application is implemented under ROS–e.g., the mobile robot–the control node is in charge of converting the BMI outputs into control commands that are understandable by the robot controller (e.g., a direction or a velocity) and simply publishes them in another topic. On the other, if the application is running outside ROS ecosystem, the *rosneuro_control* package might provide some classes implementing common communication sockets–e.g., via UDP. In this case, the stream of commands are sent to the other host, defined by its IP address and a port number, through the selected communication protocol. IP, port and type of communication can be set through ROS parameters in the launch file.

An example of the implemented control strategies to drive a computer game, running on a remote PC, and a telepresence robot, controlled in ROS, is presented in Section IV.

IV. Use cases

A. The Cybathlon BCI race

In this section, we describe how the SMR BMI implemented in ROS-Neuro can be used to control a computer game like in the BCI race [21], one of the disciplines within the Cybathlon event [22]. The "Brain Driver" game consists of four players, or pilots, brain-controlling their avatar in a virtual track. During the race, four control commands are required to drive the avatar, shown in Fig. 4A: *right*, *left*, *light* and *none*.

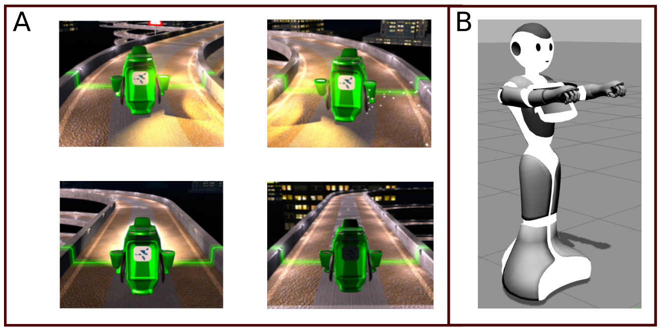

Fig. 4. A: The four commands necessary to control the avatar in the "Brain Driver" game. a) turn right, b) turn left, c) switch on the headlights, d) go straight. B: A telepresence robot executing a turning command.

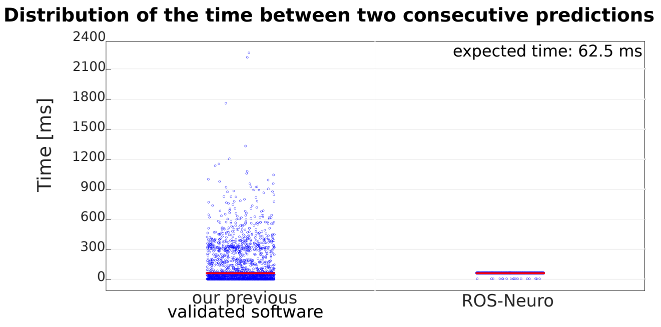

Fig. 5. Results of the test evaluation between ROS-Neuro and our previous validated software in terms of the time elapsed between two consecutive predictions. The horizontal red lines represent the mean values according to the software used.

The communication with the game is achieved with the cybathlon node implemented in the *rosneuro_control* package. This node handles the conversion of the 2-class BMI commands into the four commands necessary to control the avatar, with the following strategy:

- *right*: the avatar turns to the right when a BMI event related to both feet motor imagery occurs;
- *left*: the avatar turns to the left when a BMI event related to both hands motor imagery occurs;
- *light*: this third command is achieved as a combination of the previous two commands. If a sequence of opposite BMI commands (feet-hands or hands-feet) occurs within a certain time window, the second command switches on the headlights;
- *none*: if no BMI commands occur, no control commands are sent to the game and the avatar will go straight.

The length of the time window to send the third command can be adjusted according to the pilot's performance or preference as a ROS parameter in the *rosneuro_control* package, as well as the player number.

B. BCI-driven telepresence robot

In this section, we show how the current release of ROS-Neuro can be easily integrated with the common robotic algorithms for telepresence robots [7], [13], [15]. Typically in the brain driven robotic applications, the devices implement a discrete control strategy according to which the user interacts by sending high level commands (e.g. turn left, turn right) at a certain time. In these cases, the robot behaves as follow: it keeps its current direction as default and when a new input is delivered it changes the orientation accordingly.

Therefore, the control node is in charge to identify among the *NeuroEvent* messages the ones communicating the presence of new commands from the user and to translate them into velocity commands according to a rotation angle and the turning time, pre-defined as ROS parameters. Furthermore, to perform more sophisticated robot navigation including the obstacle avoidance, the planning of the trajectory and eventually the use of the environmental information (e.g., a map), the robotic algorithms, already available in ROS such as the *navigation stack*[6], can be exploited to develop other ROS nodes in order to achieve more intelligent behaviours of the robotic device [23].

It is worth highlighting that, in contrast to our previous implementation [15], [18], any intermediate wrapper in charge of managing the communication between the robot and the BMI loop through UDP packages is no longer necessary, since ROS-Neuro allows a direct communication between these two systems in the same software framework, for example by publishing the control commands in the standard topic */cmd_vel*.

V. Evaluation of the Packages

The entire closed-loop from acquisition to the feedback nodes has been tested to evaluate the correct functioning and to verify possible delays and jitters, especially in the processing module. In this regard, we took our previous custom software as testbed. The custom software is a C++ based library, independent from ROS, that we successfully evaluated during the Cybathlon BCI Series 2019 event. The software has been designed for a 2-class SMR BCI and it implements the same processing functionalities proposed in ROS-Neuro, including the Laplacian spatial filter, the Power Spectral Density, the gaussian classifier and the exponential integration function.

We tested both systems using GDF files, previously recorded with a g.USBamp (16-channels, g.Tec medical engineering GmbH) with a sampling rate of 512 Hz and a frame rate of 16 Hz. We evaluated 20 runs obtained using the three BCI protocols explained in Section IIIC: 2 *calibration*, 1 *evaluation* and 17 *control*.

We reproduced all the acquired files and we processed them using one software at the time keeping the same configuration parameters and the same decoder, in order to replicate the BCI experiment. All the tests were performed on a ASUS TUF Gaming FX505, Intel® Core™ i7-8750H 2.20 GHz, 16 GB RAM, 1TB 5400 rpm SATA HDD, NVIDIA® GeForce® GTX 1050TI, Ubuntu 18.04 LTS operating system and ROS Melodic Morenia.

[6]http://wiki.ros.org/navigation

TABLE II
PERFORMANCE EVALUATION BETWEEN OUR PREVIOUS SOFTWARE AND ROS-NEURO

Time elapsed between two consecutive predictions	
Our previous validated software	***ROS-Neuro***
62.3717 ± 24.5784 ms	62.4943 ± 0.6914 ms

RMSE		
Min	***Max***	***Mean ± SD***
0	0.1501	0.0522 ± 0.0444

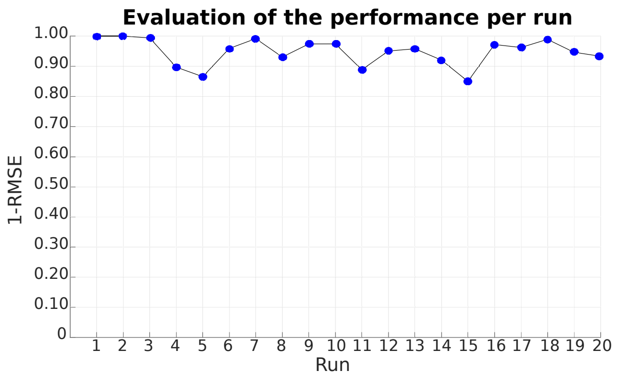

Fig. 6. Results of the test evaluation between ROS-Neuro and our previous validated software in terms of 1-RMSE, computed per run on the predictions given in output from the two systems.

We examined the raw predictions (raw probabilities) returned as output by the two implementations. To verify the presence of delays in both the implementations, we considered the time elapsed between two consecutive predictions. The results are shown in Fig. 5. The prediction is done when a new chunk of data arrives according to the frame rate set, therefore in our case theoretically a new raw output should be available every 62.5 ms (1/framerate Hz = 1/16 Hz).

The achieved mean values ± standard deviation of the processing time estimated on all the runs are shown in Table V and they are highlighted with a horizontal red line in Fig. 5. Both software provide on average the predictions with the expected classification rate. The results are in line with the times achieved in the previous studies, based on a SMR BMI, demonstrating the effectiveness of controlling external devices from graphical interface to robots [13], [15], [21]. Nevertheless, it is worth noticing the stability of ROS-Neuro (SD < 1) with respect to our previous implementation, thanks to the efficient communication infrastructure guaranteed by ROS.

Furthermore, we computed the Root Mean Square Error (RMSE) between the two distributions of probabilities in order to evaluate the accuracy of the predictions returned by ROS-Neuro. The results for each run are shown in Fig. 6, while the mean value and the standard deviation of the RMSE computed on the entire dataset is reported in Table V. The errors between the raw probabilities from the two software are very small (0.0522 on average across all the runs) and they are probably related to slight differences in the windowing used by the PSD. These findings demonstrate the consistency between the output provided by ROS-Neuro with the expected values from a previously validated software, proving the correctness of the processing node.

VI. CONCLUSION

In this work, we proposed the first release of software packages to implement a full BMI closed-loop in ROS-Neuro. As an example, the implementation of a BMI based on SMR to control either a computer game or a mobile robot is described. These packages aim to promote in the community the integration of their BMI control systems in ROS-Neuro to control robotic devices.

Following the guidelines we presented in the previous papers [18], [19], the next development steps will be focused on the implementation of a more modular architecture for neural signal processing and visualization. For instance, the *rosneuro_processing* package should support several common signal processing and machine learning algorithms. As well as, the processing module will also allow to easily concatenate more filtering techniques for processing neural data, in a similar way to the ROS *FilterChain*[7] concept.

Moreover, the *rosneuro_visualizer* package (Fig. 2) will allow the visualization of the neural data at each step of the processing flow, by simply subscribing to the topic containing the desired information (e.g., EEG data, raw or integrated predictions).

It is worth stressing the concept that ROS-Neuro has not be designed as a competitor of other BMI frameworks, like BCI2000 or BCILAB, but rather as an open-source framework to boost the integration between these BMI systems and robotic applications. In this sense, the main advantage of ROS-Neuro is that the BMI loop and robot control would be implemented in the same ecosystem, with the possibility to visualize neural signals and robot's sensors information at the same time. This characteristic may open the frontier for an innovative way of analysing and developing solutions for brain-robot mutual interaction in BMI driven applications.

REFERENCES

[1] N. Birbaumer, N. Ghanayim, T. Hinterberger, I. Iversen, B. Kotchoubey, A. Kübler, J. Perelmouter, E. Taub, and H. Flor, "A spelling device for the paralysed," *Nature*, vol. 398, no. 6725, pp. 297–298, 1999.
[2] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan, "Brain–computer interfaces for communication and control," *Clinical neurophysiology*, vol. 113, no. 6, pp. 767–791, 2002.
[3] A. A. Karim, T. Hinterberger, J. Richter, J. Mellinger, N. Neumann, H. Flor, A. Kübler, and N. Birbaumer, "Neural internet: Web surfing with brain potentials for the completely paralyzed," *Neurorehabilitation and Neural Repair*, vol. 20, no. 4, pp. 508–515, 2006.
[4] R. Leeb, F. Lee, C. Keinrath, R. Scherer, H. Bischof, and G. Pfurtscheller, "Brain–computer communication: motivation, aim, and impact of exploring a virtual apartment," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 15, no. 4, pp. 473–482, 2007.
[5] M. W. Tangermann, M. Krauledat, K. Grzeska, M. Sagebaum, C. Vidaurre, B. Blankertz, and K.-R. Müller, "Playing pinball with non-invasive BCI," in *Proceedings of the 21st International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2008, pp. 1641–1648.

[7]Plugin-based implementation provided by ROS to concatenate several data filters (http://wiki.ros.org/filters)

[6] A. Chella, E. Pagello, E. Menegatti, R. Sorbello, S. M. Anzalone, F. Cinquegrani, L. Tonin, F. Piccione, K. Prifitis, C. Blanda *et al.*, "A BCI teleoperated museum robotic guide," in *2009 International Conference on Complex, Intelligent and Software Intensive Systems*. IEEE, 2009, pp. 783–788.

[7] L. Tonin, T. Carlson, R. Leeb, and J. d. R. Millán, "Brain-controlled telepresence robot by motor-disabled people," in *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2011, pp. 4227–4230.

[8] L. Tonin, F. C. Bauer, and J. d. R. Millán, "The Role of the Control Framework for continuous teleoperation of a brain–machine interface-driven mobile robot," *IEEE Transactions on Robotics*, 2019.

[9] J. d. R. Millán, F. Galán, D. Vanhooydonck, E. Lew, J. Philips, and M. Nuttin, "Asynchronous non-invasive brain-actuated control of an intelligent wheelchair," in *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2009, pp. 3361–3364.

[10] N. A. Bhagat, A. Venkatakrishnan, B. Abibullaev, E. J. Artz, N. Yozbatiran, A. A. Blank, J. French, C. Karmonik, R. G. Grossman, M. K. O'Malley *et al.*, "Design and optimization of an EEG-based brain machine interface (bmi) to an upper-limb exoskeleton for stroke survivors," *Frontiers in neuroscience*, vol. 10, p. 122, 2016.

[11] K. Lee, D. Liu, L. Perroud, R. Chavarriaga, and J. d. R. Millán, "A brain-controlled exoskeleton with cascaded event-related desynchronization classifiers," *Robotics and Autonomous Systems*, vol. 90, pp. 15–23, 2017.

[12] J. R. Millán, F. Renkens, J. Mourino, and W. Gerstner, "Noninvasive brain-actuated control of a mobile robot by human EEG," *IEEE Transactions on biomedical Engineering*, vol. 51, no. 6, pp. 1026–1033, 2004.

[13] L. Tonin, R. Leeb, M. Tavella, S. Perdikis, and J. d. R. Millán, "The role of shared-control in bci-based telepresence," in *2010 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 2010, pp. 1462–1466.

[14] K. Muelling, A. Venkatraman, J.-S. Valois, J. E. Downey, J. Weiss, S. Javdani, M. Hebert, A. B. Schwartz, J. L. Collinger, and J. A. Bagnell, "Autonomy infused teleoperation with application to brain computer interface controlled manipulation," *Autonomous Robots*, vol. 41, no. 6, pp. 1401–1422, 2017.

[15] G. Beraldo, M. Antonello, A. Cimolato, E. Menegatti, and L. Tonin, "Brain-Computer Interface meets ROS: A robotic approach to mentally drive telepresence robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–6.

[16] G. Beraldo, S. Tortora, and E. Menegatti, "Towards a Brain-Robot Interface for children," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE, 2019, pp. 2799–2805.

[17] C. Brunner, G. Andreoni, L. Bianchi, B. Blankertz, C. Breitwieser, S. Kanoh, C. A. Kothe, A. Lécuyer, S. Makeig, J. Mellinger *et al.*, "BCI software platforms," in *Towards Practical Brain-Computer Interfaces*. Springer, 2012, pp. 303–331.

[18] G. Beraldo, N. Castaman, R. Bortoletto, E. Pagello, J. d. R. Millán, L. Tonin, and E. Menegatti, "ROS-Health: An open-source framework for neurorobotics," in *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*. IEEE, 2018, pp. 174–179.

[19] L. Tonin, G. Beraldo, S. Tortora, L. Tagliapietra, J. d. R. Millán, and E. Menegatti, "ROS-Neuro: A common middleware for BMI and robotics. the acquisition and recorder packages," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE, 2019, pp. 2767–2772.

[20] J. del R. Millán, P. W. Ferrez, F. Galán, E. Lew, and R. Chavarriaga, "Non-invasive brain-machine interaction," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 22, no. 05, pp. 959–972, 2008.

[21] S. Perdikis, L. Tonin, S. Saeedi, C. Schneider, and J. d. R. Millán, "The Cybathlon BCI race: Successful longitudinal mutual learning with two tetraplegic users," *PLoS biology*, vol. 16, no. 5, p. e2003787, 2018.

[22] R. Riener, "The Cybathlon promotes the development of assistive technology for people with physical disabilities," *Journal of neuroengineering and rehabilitation*, vol. 13, no. 1, p. 49, 2016.

[23] G. Beraldo, E. Termine, and E. Menegatti, "Shared-Autonomy Navigation for Mobile Robots Driven by a Door Detection Module," in *International Conference of the Italian Association for Artificial Intelligence*. Springer, 2019, pp. 511–527.