An Assistive Shared Control Architecture for a Robotic Arm Using EEG-Based BCI with Motor Imagery

Giuseppe Gillini, Paolo Di Lillo, Filippo Arrichiello

***Abstract*— The paper presents a shared control architecture for robotic systems commanded through a motor imagery based Brain-Computer Interface (BCI). The overall system is aimed at assisting people to perform teleoperated manipulation tasks, and it is structured so as to leave different levels of autonomy to the user depending on the actual stage of the task execution. The low-level part of the shared control architecture is also in charge of taking into account safety and operational tasks, such as to avoid collisions or to manage robot joint limits. The overall architecture has been realized by integrating control and perception software modules developed within the ROS environment, with the OpenVibe framework used to operate the BCI device. The effectiveness of the proposed architecture has been validated through experiments where a healthy user, wearing a Unicorn g.tec BCI, performs an assisted task through motor imagery sessions, with a 7 Degrees of Freedoms Kinova Jaco2 robotic arm.**

I. Introduction

For people with severe motor diseases, daily life activities, such as drinking or getting fed, might be difficult or even impossible to accomplish; such conditions might limit one's independence and lower the quality of life, thus requiring a constant presence of caregivers for assistance. In this context, the development of assistive robotics architectures aimed at providing some sort of autonomy to the user in the execution of basic life tasks [1] could represent a possible help.

From a robotic system control perspective, two main operational modes can be envisioned for assistive operations: shared and supervisory control strategies. Such modes are strictly related to the level of involvement of the user into the task, to the level of autonomy required to the robotic system, and to the Human-Machine Interfaces (HMIs) used to provide operational commands, that could rely on different kinds of devices, as e.g. joysticks or haptic interfaces, and signals, as e.g. ElectroOcoluGraphic (EOG), ElectroEncephaloGraphic (EEG) and ElectroMyoGraphic (EMG) signals [2]. Here we focus on the use of non invasive Brain-Computer Interfaces (BCIs) [3], i.e. headsets with a series of electrodes to place on the scalp of the user to measure the EEG signals. In BCIs' field, supervisory control applications might be obtained relying on the P300 component of the Event Related Potentials (ERPs) [4], i.e. fluctuations in the EEG signals generated by the electrophysiological response to a significant sensorial stimulus or event, or through Steady State Visually Evoked Potential (SSVEP) [5], i.e. a resonance phenomenon in the EEG signal which can be observed when a subject looks at a light source flickering at a constant frequency. For example, in [6] and in [7] we presented a control solution where a P300-based BCI was used to generate high-level directives for different robotic systems. On the other hand, shared control application can be obtained relying on motor imagery [8], i.e. a cognitive process in which a user imagines to perform a movement without actually performing it. Different approaches have been adopted to control robots with two class motor imagery, in which the classifier has to discriminate between two possible intended movements (usually right hand and left hand). In [9] motor imagery signals are used for controlling the lifting and the dropping of an object carried by a dual-arm robot. In order to obtain a wider variety of possible actions to perform, the two classes could be translated in a set of commands by defining a specific paradigm as in [10]. In particular, the authors use the two motor imagery commands to define a posture-dependent control architecture where, depending on the motor command and the robot position, the BCI user is able to send four motion commands to a mobile robot. To execute more complex tasks, two motor imagery classes could be inadequate. Thus, a larger class number could be considered as in [11], where a four class motor imagery session is adopted to control two different kind of mobile robots movements in the 3D space. In [12] a motor imagery EEG-based continuous teleoperation robot control system to grasp objects in the 2D planar coordinates with tactile feedback is presented. In all these works, the four motor imagery classes have been used to control only the mobile robot movements or few DOFs (Degrees Of Freedom) of the manipulator end-effector, limiting the range of possible tasks to be performed.

In this work we present a shared control architecture aimed at performing assisted teleoperation tasks using a motor imagery based BCI. The architecture is developed so as to leave different levels of autonomy to the user depending on the actual stage of the task execution. Indeed, depending on the task stage, the control architecture automatically switches among control modes that commands the robot to autonomously execute specific operations, and control modes that might allow the user to provide to the robot either four or two different commands. During all the different stages, safety constraints tasks are handled through a set-based task-priority inverse kinematic control algorithm; i.e., a specific motion control strategy able to manage both set-based and

Authors are with the Department of Electrical and Information Engineering of the University of Cassino and Southern Lazio, Via G. Di Biasio 43, 03043 Cassino (FR), Italy {giuseppe.gillini, pa.dilillo, filippo.arrichiello}@unicas.it

This work has been supported by the MIUR program "Dipartimenti di Eccellenza 2018-2022" granted to Department DIEI of the University of Cassino and Southern Lazio.

equality-based tasks organized in a hierarchy [13], [14].

The contribution of the paper is two-fold: 1) the extension of the work [15], in which the authors developed a filter to continuously send commands reducing the false positives rate for managing two motor imagery classes, to the use of four motor imagery classes; 2) the design of a shared control architecture that assists the user in the control of a robotic arm in 3D space to execute manipulation tasks, relying on a maximum of four available commands.

The effectiveness of the technological approach used in the proposed architecture has been validated through experiments where a healthy user (the first author of the present paper), wearing a BCI, performs assisted teleoperation tasks to pick and place an object using a 7 DOFs robotic arm.

II. Assistive Shared Control Architecture

In the proposed architecture the user is able to send up to a maximum of four commands for the robot end-effector control, limiting the possibility to accomplish complex tasks. Thus, we propose a shared control strategy that assists the user by understanding his/her intended task, helping in the execution of some operation and autonomously performing some of the phases needed for the task accomplishment. This is handled structuring the envisioned shared control strategy into three possible control modes: *user 2-D control*, *user 1-D control* and *robot control*. The switching among these control modes is automatically handled by the architecture depending on the operational conditions.

In the *user 2-D control* mode, the motor imagery BCI allows the user to discern among four classes (RIGHT, LEFT, FORWARD, BACKWARD), and the output of the BCI classifier is used to modify the desired robot end-effector position in the 2-D space (e.g. the horizontal plane).

In *user 1-D control* mode, the motor imagery BCI allows the user to discern among two classes (FORWARD, BACKWARD), and the output of the BCI classifier is used to modify the advancement of the robot in a task execution (e.g. move forward or backward in the grasping direction of a target object).

In *robot control* mode, the output of the BCI is neglected and the robot automatically executes a preassigned operation.

A. BCI based Motor Imagery

When the system is in a *user control* mode, through a motor imagery session, the user can use 2 or 4 commands to teleoperate the robotic arm. In particular, in *user 1-D control* mode, the user imagines right or left hand movements to respectively go ahead and back in the task execution, while for the *user 2-D control* mode he imagines right and left hand movements to control the robot in the right/left directions, and both hands and tongue movements to control the robot in the forward and backward directions. All the aspects related to the acquisition, processing and classification of the brain signals are handled by resorting to the open source framework OpenVibe.

In order to improve the classification reliability and continuity, it is worthwhile introducing an additional filtering layer as presented in [15], where the authors developed a filter suitable for mitigating the above-mentioned issues for 2 motor imagery classes. In this paper we rely on the solution presented in [15] for generating commands in the *user 1-D control* mode; furthermore we have extended this formulation in order to handle also sessions in which there are 4 possible classes to classify, as for the *user 2-D control* mode.

1) Motor imagery classes for user 1-D control mode

Given a manifold of two possible classes $\mathcal{M} = \{C_{\text{forward}}, C_{\text{backward}}\}$, defining x_t^{forward} as the probability of C_{forward} coming from the classifier output at time t, and denoting as y_t^{forward} the filter output at time t, then the filter output increment at time t, Δy_t^{left}, is obtained by

$$\Delta y_t^{\text{forward}} = \chi \cdot [\phi \cdot F_{\text{free}}(y_{t-1}^{\text{forward}}) + (1-\phi) \cdot F_{\text{BCI}}(x_t^{\text{forward}})] \quad (1)$$

where y_{t-1}^{forward} is the filter output at time $t-1$, F_{free} is the *free force* exerted by the filter, which is dependent from its past outputs, and F_{BCI} is the *force* exerted by the BCI depending on the current input. These two forces are then weighted for a confidence factor ϕ, and multiplied by a filter gain χ. The filter works in the following way: the current probability x_t^{forward} coming from the BCI classifier is taken as input by the filter, together with its output at the previous time step. x_t^{forward} is then weighted by F_{BCI}, while the filter output at previous instant is used to compute the F_{free}. The F_{free} is defined as:

$$F_{\text{free}} = \begin{cases} -\sin(\frac{\pi}{0.5-\omega} \cdot y), & \text{if } y \in [0, 0.5-\omega) \\ -\psi \sin[\frac{\pi}{\omega} \cdot (y-0.5)], & \text{if } y \in [0.5-\omega, 0.5+\omega] \\ \sin[\frac{\pi}{0.5-\omega} \cdot (y-0.5-\omega)], & \text{if } y \in (0.5+\omega, 1] \end{cases} \quad (2)$$

where ψ is the sin wave amplitude, and ω the range used to discriminate the filter output. The F_{BCI} is defined as:

$$F_{\text{BCI}} = \alpha_1 \cdot (x-0.5)^3 + \alpha_2 \cdot (x-0.5). \quad (3)$$

where α_1 and α_2 are parameters to be properly tuned depending on the EEG data. The F_{free} is designed in order

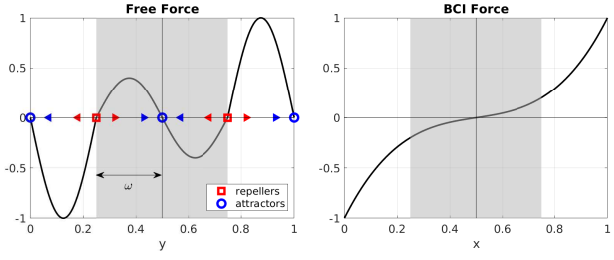

Fig. 1: Left: evolution of the free force F_{free} with $\psi = 0.4$, $\omega = 0.3$. Right: BCI force F_{BCI} with $\alpha_1 = 6.4$ and $\alpha_2 = 0.4$.

to exert a conservative force when the current state of the system is close to 0.5, 0.0 and 1.0 and a pushing force otherwise. These points can be considered as *attractors* for the filter output, while the points located at $0.5-\omega$ and $0.5+\omega$ are *repellers* that represent undesirable intermediate points in which the filter output cannot be determined due to the too low probability belonging to the current class. This

helps the system to be less sensitive to random perturbations reducing the false positives rate, and allows managing the relaxed state, i.e. a state in which the user does not want to generate any command. On the other hand, the F_{BCI} has been designed to reduce or enhance the impact of BCI responses with low or high confidence. Then, the two forces are combined using ϕ as parameter to influence how much to trust the BCI output versus the filter evolution. Finally, the class in output from the filter is:

$$C^{\text{out}} = \begin{cases} C_{\text{forward}}, & \text{if } y_t^{\text{forward}} >= 0.5 + \omega \\ C_{\text{backward}}, & \text{if } y_t^{\text{forward}} <= 0.5 - \omega \end{cases} \quad (4)$$

being C_{forward} and C_{backward} mutually exclusive, while the filter does not give any output if $y_t^{\text{forward}} \in (0.5 - \omega, 0.5 + \omega)$. The evolution of the two forces is shown in Fig. 1.

2) Motor imagery classes for user 2-D control mode

The described method, designed for binary classifications, cannot be directly used in cases in which there are 4 possible classes. Therefore, given the manifold of the four possible classes $\mathcal{M} = \{C_{\text{left}}, C_{\text{right}}, C_{\text{forward}}, C_{\text{backward}}\}$, here we propose to use two filters in cascade, one for discriminating the direction of the intended movement, and the other one for discriminating the sense. In particular the first one works on a manifold composed by other two manifolds:

$$\mathcal{M}_{\text{direction}} = \{\mathcal{M}_{\text{horizontal}} = \{C_{\text{left}}, C_{\text{right}}\}, \quad (5)$$

$$\mathcal{M}_{\text{vertical}} = \{C_{\text{forward}}, C_{\text{backward}}\}\}. \quad (6)$$

In this way, the first filter takes in input the probability $x_t^{\text{horizontal}} = x_t^{\text{left}} + x_t^{\text{right}}$ and it discriminates the direction of the intended movement. Its output is the manifold:

$$\mathcal{M}_{\text{direction}}^{\text{out}} = \begin{cases} \mathcal{M}_{\text{horizontal}}, & \text{if } y_t^{\text{horizontal}} > 0.5 + \omega \\ \mathcal{M}_{\text{vertical}}, & \text{if } y_t^{\text{horizontal}} < 0.5 - \omega \end{cases} \quad (7)$$

The manifold in output from the first filter is used by the second one that discriminates the sense of the movement. Assuming that $\mathcal{M}_{\text{horizontal}}$ is the output of the first filter, the input of the second one would be the normalized probability of the LEFT class $\bar{x}_t^{\text{left}} = \frac{x_t^{\text{left}}}{x_t^{\text{left}} + x_t^{\text{right}}}$, with output class:

$$C^{\text{out}} = \begin{cases} C_{\text{left}}, & \text{if } y_t^{\text{left}} > 0.5 + \omega \\ C_{\text{right}}, & \text{if } y_t^{\text{left}} < 0.5 - \omega \end{cases} \quad (8)$$

Assuming that $\mathcal{M}_{\text{vertical}}$ is the output of the first filter, the input of the second one would be the normalized probability of the FORWARD class $\bar{x}_t^{\text{forward}} = \frac{x_t^{\text{forward}}}{x_t^{\text{forward}} + x_t^{\text{backward}}}$, with output class:

$$C^{\text{out}} = \begin{cases} C_{\text{forward}}, & \text{if } y_t^{\text{forward}} > 0.5 + \omega \\ C_{\text{backward}}, & \text{if } y_t^{\text{forward}} < 0.5 - \omega \end{cases} \quad (9)$$

B. Robot end-effector control strategy

The switching among the control modes is related to the operational conditions and it follows a Finite State Machine related to the configured achievable tasks. Starting from a certain initial position of the end-effector $P_{\text{ee,ini}}$, the initial control mode is assumed to be *user 2-D control*, thus the user can control the planar position of the end-effector. The procedure is initialized taking into account the end-effector initial position $P_{\text{ee,ini}}$ and the position of the considered target p_{target}, while *user 2-D control*is considered as the initial control mode. The control strategy updates the robot end-effector position depending on the current control mode and, if the latter is not set on *robot control*, on the BCI command received. More in detail, when the control mode is *user 2-D control*, the user could move the end-effector position on a plane, so as to reach to the desired object. When he/she brings the end-effector position close enough to a certain target (inside a sphere of radius smaller or equal than a certain threshold Δ centered in the target), the control mode switches in *robot control*. At this point the robot autonomously executes a movement that changes the position/orientation of the end-effector, in order to initialize a task that depends on the type of the considered target (e.g. it aligns the end-effector to an object to be grasped). Once that the autonomous movement is completed, the control mode is changed to *user 1-D control*, and the control returns to the user. Then, he/she can choose to abort the task (i.e. moving the end-effector away from the target), or to go ahead until its completion (e.g. moving the end-effector toward an object until it can be autonomously grasped). In both cases the control mode switches to *robot*, that once again makes the robot autonomously move to a predefined location, eventually switching the control mode back to *user 2-D control*.

When the system is in *user 1-D control* or *user 2-D control* mode, the class C^{out} computed by the filters at each time step goes in input to a dedicated software node that computes a point of the reference trajectory for the end-effector to be followed by the robot.

In particular, when in *user 2-D control* mode, each command is associated to a desired velocity at timestep k expressed in the arm base frame:

$$\boldsymbol{v}_{\text{RL}}(k) = \begin{bmatrix} \{V_{\text{des}}, -V_{\text{des}}\} \\ 0 \end{bmatrix} \quad \boldsymbol{v}_{\text{FB}}(k) = \begin{bmatrix} 0 \\ \{V_{\text{des}}, -V_{\text{des}}\} \end{bmatrix} \quad (10)$$

where $V_{\text{des}} > 0$ is a parameter that expresses the desired end-effector velocity amplitude, $\boldsymbol{v}_{\text{RL}}$ and $\boldsymbol{v}_{\text{FB}}$ are the velocity associated with the (RIGHT, LEFT) and (FORWARD, BACKWARD) commands respectively, assuming RIGHT and FORWARD as positive directions and LEFT and BACKWARD as negative directions. These velocity commands get then integrated over time, obtaining a reference position on the xy plane to be sent to the robot.

$$\boldsymbol{p}_{\text{des}}^{\text{xy}}(k+1) = \boldsymbol{p}_{\text{des}}^{\text{xy}}(k) + \begin{bmatrix} v_{\text{RL},x}(k) \\ v_{\text{FB},y}(k) \end{bmatrix} T_s \ , \quad (11)$$

where T_{s} is the sampling time, while the z_{des} coordinate is kept constant at the current value, obtaining the 3D desired position as $\boldsymbol{p}_{\text{des}}(k) = [\boldsymbol{p}_{\text{des}}^{xy}(k)\ z_{\text{des}}(k)]^T$. When system enters in the *user 1-D control* mode, a linear path associated to the specific operation to achieve is defined, e.g. a linear movement to perform the grasping of an object from a pregrasp configuration, or to move toward a release location

when the grabbed object is close to it. In this mode, the user can generate only two commands (FORWARD, BACKWARD) to go ahead or back in the task execution. In this case the reference position moves along a segment, which is expressed as:

$$\boldsymbol{p}_{\text{des}}(s(k)) = (1 - s(k))\, \boldsymbol{p}_{\text{ini}} + s(k)\, \boldsymbol{p}_{\text{fin}} \,, \tag{12}$$

where $\boldsymbol{p}_{\text{ini}}$ is an initial 3D position (e.g. the pregrasp position when the task is to grasp an object), $\boldsymbol{p}_{\text{fin}}$ is the a final 3D position (e.g. the object position) and $s(k) \in [0, 1]$ is a parameter. The command given by the user is translated in a velocity v_{AB} set as V_{des} for the FORWARD command and $-V_{\text{des}}$ for the BACKWARD one, that gets integrated over time to increase or decrease the parameter s allowing to go forward with the task execution until its completion ($s(k) = 1$) or to go back until the task is aborted ($s(k) = 0$):

$$s(k+1) = s(k) + v_{\text{AB}} T_{\text{s}} \,. \tag{13}$$

In *robot control* mode, the robot autonomously adjusts its end-effector 3D position and orientation according to the actual stage of the task, e.g. it brings the end-effector to $\boldsymbol{p}_{\text{ini}}$ when a task is started, or it moves the end-effector in predefined configurations once a task is completed. In sum, the reference pose for the end-effector at each time step is composed of a desired position $\boldsymbol{p}_{\text{des}}$ and orientation $\boldsymbol{\mathcal{Q}}_{\text{des}}$ expressed in quaternion:

$$\boldsymbol{\sigma}_{\text{pose, des}} = \begin{bmatrix} \boldsymbol{p}_{\text{des}} \\ \boldsymbol{\mathcal{Q}}_{\text{des}} \end{bmatrix} \in \mathbb{R}^7 \tag{14}$$

The computed reference pose goes in input to an inverse kinematics algorithm that computes the joint velocities needed to move the end-effector toward it. For more details about the joint control algorithm the reader is referred to [6], [16].

III. Experimental Results

A. System Architecture

The overall architecture has been implemented and experimentally tested using a non-invasive g.tec Unicorn BCI, a 7 DOF Kinova Jaco2 robotic arm, and an Intel RealSense RGB-D sensor for object detection and localization. Figure 2 shows a scheme of the overall software architecture. The RGB-D based perception module recognizes the available objects in the scene through an object recognition algorithm based on YOLO, that locates them in the 2D image plane.

B. Filters cascade effect on the command generation

The first experimental tests[1] were aimed at evaluating the effectiveness of the implemented filters cascade on the command generation. In particular, we focus the attention on the *user 2-D control* mode. The user is asked to generate each one of the four possible commands in the following sequence: LEFT, FORWARD, RIGHT, BACKWARD. Each

[1]The experiments were taken in accordance with the Declaration of Helsinki, the protocol has been approved by the Research Ethics Committee of the University of Cassino and Southern Lazio and the participant gave informed consent.

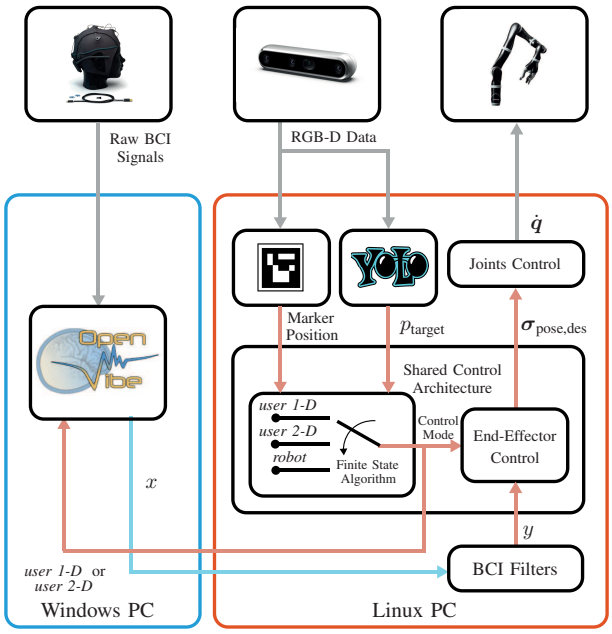

Fig. 2: Main components of the proposed architecture.

command is intended to be generated for a given amount of time (10 s). The goal of the experiment is to evaluate the differences on the commands generated by the BCI when using the filter cascade and when considering as output the raw command with the highest probability among the four possible choices. The parameters used for Filter 1 and Filter 2 in this experiment are: $\omega_1 = \omega_2 = 0.3$, $\psi_1 = 0.2$, $\psi_2 = 0.1$, $\chi_1 = 2$, $\chi_2 = 1$, $\phi_1 = 0.6$, $\phi_2 = 0.7$, $\alpha_{1,1} = \alpha_{1,2} = 6.4$, $\alpha_{2,1} = \alpha_{2,2} = 0.4$.

Figure 3 shows the input-output of the two filters over time during one of the experiments, while the top plot in Figure 4 shows the desired sequence over time, the mid plot shows the classified commands without using the filters, which are the ones with the highest probability, and the bottom plot shows the classified commands using the filters, given the same input as in the mid plot. It is possible to see that, without using the filters, between $0-10$s and $20-30$s there are several time intervals in which the command with the highest probability is not the desired one, leading to misclassification. Using the filters, instead, this kind of problem is avoided. Indeed, the attractor/repeller system prevents the change of the output caused by random perturbations in the input, letting the filter successfully classify the correct command even if the probability in input tries to push the output toward 0.5 (peaks right before 10s and after 20s in the first filter output). Moreover, it is worth noticing that, for the first seconds of the experiment, the filters correctly do not give any output (white background in the bottom plot of Fig. 4) because the output of the second filter is close to 0.5, most likely because the user was in a relaxed state and needed some time to focus on the task when the test started.

C. Experimental Validation

Following the general architecture described in Section II, pick and place operations were considered for the experi-

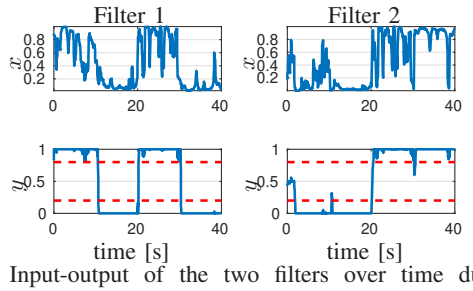

Fig. 3: Input-output of the two filters over time during the experiment.

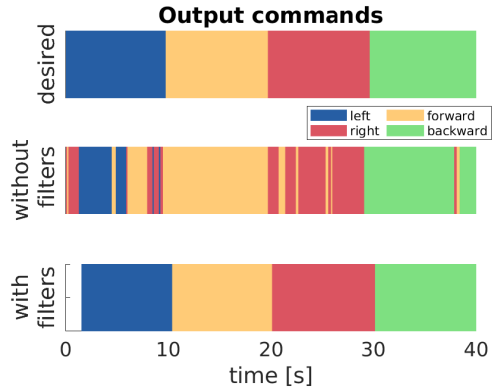

Fig. 4: BCI filter effect. Starting from the top plot: the desired commands sequence over time; the classified commands without using the filters; the classified commands using the filter.

mental tests, in which the user could teleoperate the robot to grasp one of the bottles located on a table and release it in one of the pre-assigned Release Object Areas (ROAs). In this section we show results of a set of experiments aimed at validating the proposed technological approach, during which, a healthy user (the first author of the present paper) is asked to perform 10 times with the filters and 10 times without the filters, the following sequence of tasks: grasp a bottle, place it on a first ROA, regrasp it and place it on the second ROA. Far all the repetitions of the experiment, the end-effector started from a predefined configuration and the bottle was placed in the same initial configuration. Note that the bottle location on the table could be any, but we have chosen to start each experimental run with the bottle placed on the same initial configuration in order to perform a fair comparison between the two cases in terms of percentage of failures and execution time.

The defined sequence initially requires the user to control the robot end-effector position on the xy plane relying on the *user 2-D control* mode. When the end-effector is close enough to the bottle, the shared architecture switches in *robot control* mode and the robot autonomously reaches a pregrasp configuration aligned with the object. At that point the control goes back to the user, this time in *user 1-D control* mode, and he/she can control the end-effector to go forward or backward along a segment that connects the pregrasp position with the bottle. Once the task is completed ($s = 1.0$ in Eq. 12), the architecture switches again in *robot control* mode and the robot grasps it and moves the end-effector in a predefined configuration. From that point on, the user, being in *user 2-D control* mode, can control again

TABLE I: Pick and place experiments results

Run	Execution time [s]	
	With filters	Without filters
1	80	110
2	85	115
3	87	FAIL
4	82	108
5	79	FAIL
6	77	112
7	80	113
8	77	FAIL
9	75	117
10	78	FAIL
Mean [s]		
	80	113
Standard deviation		
	0.04	0.03

the planar position of the end-effector. The same happens when the user teleoperates the end-effector with an object grasped close enough to a ROA. If, during a grasping or releasing task, the user moves backward to $s = 0.0$, this is interpreted as the user intention to give up to the selected task, bring back system to the *user 2-D control* mode to move the hand towards a different location. The Finite State Machine diagram of the pick and place experiment is shown in Fig. 5.

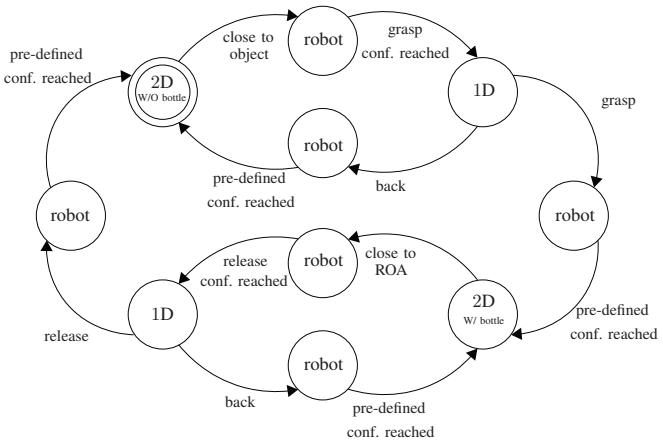

Fig. 5: Finite State Machine diagram of the proposed shared control architecture, particularized for the pick and place experiment.

The desired orientation is automatically handled by the shared architecture. We have chosen three possible predefined orientations useful as a visual feedback to the user for understanding when the shared architecture switches among the different control modes. When in *user 2-D control* mode, the desired orientation, expressed as unit quaternion, is $\mathcal{Q}_{\text{des}} = [-0.5\ 0.5\ 0.5\ -0.5]^T$. When in *user 1-D control* mode, the orientation depends on the specific task to be accomplished: in case of a grasping task, the desired orientation is set to $\mathcal{Q}_{\text{des}} = [-0.7071\ 0\ 0.7071\ 0]^T$; in case of a releasing task on a ROA, the orientation is $\mathcal{Q}_{\text{des}} = [-0.5\ 0.5\ 0.5\ 0.5]^T$.

Table I shows the results of the different runs, where we consider a run failed when the execution time to complete the experiment was higher than 120s. Even ignoring the

failed runs, the average execution time using the filters (80s) is anyway lower than without using them (113s). This demonstrates that the implemented filters cascade allows to control the end-effector more precisely. Figure 6 shows some snapshots of the experiment execution, while a complete video can be found at the following link[2]. Figure 7 shows

Fig. 6: Screenshots of the video taken during an experiment.

the path followed by the end-effector in one of the runs executed using the filters. In the figure are depicted the initial end-effector configuration (in black), the two ROAs (red and white circles) and the bottle in the initial position and on the two ROAs. The color of the 3D path of the end-effector highlights the state of the shared control architecture: green for *robot control* mode; blue for *user 2-D control* mode; red for *user 1-control* mode. It is clear that the control mode switches several times during the execution of the sequence, effectively allowing to perform the requested tasks using only a maximum of four commands.

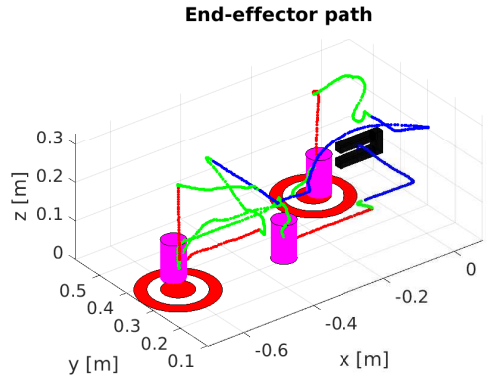

Fig. 7: Path followed by the end-effector in one of the experiment runs. In the plot are highlighted the initial end-effector configuration (in black), the two ROAs (red and white circles) and the bottle in positions of interest.

IV. Conclusions and Future Work

In this paper, we presented a shared control architecture for robotic systems aimed at assisting people with severe motion disabilities to perform teleoperated manipulation tasks through a motor imagery based BCI. Experimental results showed the effectiveness of the implemented filter cascade and the technological approach used in the proposed architecture.

Future works will include: i) an extensive experimental campaign, to be performed in collaboration with healthcare organizations, on multiple healthy and unhealthy users to evaluate the actual usability of the system, which represents a key point for the evaluation of the whole architecture in terms of robustness, accuracy and perceived ease-of-use; ii) the integration of other possible manipulation tasks to assist the user in wider use case scenarios.

[2]https://www.youtube.com/watch?v=EnztvCKG2z4

References

[1] N. C. M. Nickelsen, "Imagining and tinkering with assistive robotics in care for the disabled," *Paladyn, Journal of Behavioral Robotics*, vol. 10, no. 1, pp. 128 – 139, 2019. [Online]. Available: https://www.degruyter.com/view/journals/pjbr/10/1/article-p128.xml

[2] G. Quere, A. Hagengruber, M. Iskandar, S. Bustamante, D. Leidner, F. Stulp, and J. Vogel, "Shared control templates for assistive robotics," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1956–1962.

[3] S. Liyanage and C. Bhatt, "Wearable electroencephalography technologies for brain–computer interfacing," in *Wearable and Implantable Medical Devices*. Elsevier, 2020, pp. 55–78.

[4] C. Mulert, O. Pogarell, G. Juckel, D. Rujescu, I. Giegling, D. Rupp, P. Mavrogiorgou, P. Bussfeld, J. Gallinat, H. Möller *et al.*, "The neural basis of the p300 potential," *European archives of psychiatry and clinical neuroscience*, vol. 254, no. 3, pp. 190–198, 2004.

[5] D. Zhu, J. Bieger, G. Garcia Molina, and R. M. Aarts, "A survey of stimulation methods used in ssvep-based bcis," *Computational intelligence and neuroscience*, vol. 2010, 2010.

[6] P. Di Lillo, F. Arrichiello, D. Di Vito, and G. Antonelli, "BCI-controlled assistive manipulator: developed architecture and experimental results," *IEEE Transactions on Cognitive and Developmental Systems*, 2020.

[7] G. Gillini, P. Di Lillo, F. Arrichiello, D. Di Vito, A. Marino, G. Antonelli, and S. Chiaverini, "A dual-arm mobile robot system performing assistive tasks operated via p300-based brain computer interface," *Industrial Robot*, 2020.

[8] 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.

[9] Y. Liu, W. Su, Z. Li, G. Shi, X. Chu, Y. Kang, and W. Shang, "Motor-imagery-based teleoperation of a dual-arm robot performing manipulation tasks," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 11, no. 3, pp. 414–424, 2018.

[10] M. Aljalal, R. Djemal, and S. Ibrahim, "Robot navigation using a brain computer interface based on motor imagery," *Journal of Medical and Biological Engineering*, vol. 39, no. 4, pp. 508–522, 2019.

[11] A. M. Batula, Y. E. Kim, and H. Ayaz, "Virtual and actual humanoid robot control with four-class motor-imagery-based optical brain-computer interface," *BioMed research international*, vol. 2017, 2017.

[12] B. Xu, W. Li, X. He, Z. Wei, D. Zhang, C. Wu, and A. Song, "Motor imagery based continuous teleoperation robot control with tactile feedback," *Electronics*, vol. 9, no. 1, p. 174, 2020.

[13] S. Moe, G. Antonelli, A. Teel, K. Pettersen, and J. Schrimpf, "Set-based tasks within the singularity-robust multiple task-priority inverse kinematics framework: General formulation, stability analysis and experimental results," *Frontiers in Robotics and AI*, vol. 3, p. 16, 2016.

[14] P. Di Lillo, E. Simetti, F. Wanderlingh, G. Casalino, and G. Antonelli, "Underwater intervention with remote supervision via satellite communication: Developed control architecture and experimental results within the dexrov project," *IEEE Transactions on Control Systems Technology*, 2020.

[15] 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*, vol. 36, no. 1, pp. 78–91, 2019.

[16] E. Cataldi, F. Real, A. Suarez, P. Di Lillo, F. Pierri, G. Antonelli, F. Caccavale, G. Heredia, and A. Ollero, "Set-based inverse kinematics control of an anthropomorphic dual arm aerial manipulator," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 2960–2966.