Hindawi
Journal of Robotics
Volume 2022, Article ID 6178917, 12 pages
https://doi.org/10.1155/2022/6178917

Research Article

A Brain-Computer Interface for Teleoperation of a Semiautonomous Mobile Robotic Assistive System Using SLAM

Vidya Nandikolla, Bryan Ghoslin, Kevin Matsuno, and Daniel A. Medina Portilla

Department of Mechanical Engineering, College of Engineering and Computer Science, California State University, Northridge, Los Angeles, CA, USA

Correspondence should be addressed to Vidya Nandikolla; vidya.nandikolla@csun.edu

Received 18 November 2021; Revised 7 February 2022; Accepted 9 February 2022; Published 26 February 2022

Academic Editor: Kaijian Xia

The proposed assistive hybrid brain-computer interface (BCI) semiautonomous mobile robotic arm demonstrates a design that is (1) adaptable by observing environmental changes with sensors and deploying alternate solutions and (2) versatile by receiving commands from the user's brainwave signals through a noninvasive electroencephalogram cap. Composed of three integrated subsystems, a hybrid BCI controller, an omnidirectional mobile base, and a robotic arm, the proposed robot has commands mapped to the user's brainwaves related to a set of specific physical or mental tasks. The implementation of sensors and the camera systems enable both the mobile base and the arm to be semiautonomous. The mobile base's SLAM algorithm has obstacle avoidance capability and path planning to assist the robot maneuver safely. The robot arm calculates and deploys the necessary joint movement to pick up or drop off a desired object selected by the user via a brainwave controlled cursor on a camera feed. Validation, testing, and implementation of the subsystems were conducted using Gazebo. Communication between the BCI controller and the subsystems is tested independently. A loop of prerecorded brainwave data related to each specific task is used to ensure that the mobile base command is executed; the same prerecorded file is used to move the robot arm cursor and initiate a pick-up or drop-off action. A final system test is conducted where the BCI controller input moves the cursor and selects a goal point. Successful virtual demonstrations of the assistive robotic arm show the feasibility of restoring movement capability and autonomy for a disabled user.

1. Introduction

In the fields of robotics and biomedical engineering, there is a growing trend in designing assistive robots to aid users in demanding tasks. The design parameters for these robots involve a detailed understanding of the operating environment, intended task, and user-control inputs. Sensors, cameras, and other instrumentation feedbacks have propelled engineers to create "smart" controllers that allow assistive robots to detect changes in the working environment, plan a work around, and execute sensitivity and accuracy. In addition to improving the assistive robot's robustness and resilience, researchers are developing versatile controller inputs that allow users, regardless of their severe physical or neurological limitations, to interface with the assistive robot. In most cases, the severity of the user's condition increases the controller's complexity and the robot's level of autonomy.

This paper focuses on the implementation of an assistive hybrid brain-computer interface (BCI) semiautonomous mobile robotic arm for users with debilitating paralysis or progressive nervous system diseases. While these users have impaired brain lobes that prohibit motor movement and/or speech, their cognitive and sensory functions are often uncompromised. Brain signals from these functional lobes are obtained from a noninvasive electroencephalogram (EEG) cap. As a result, a set of specific physical or mental tasks (i.e., jaw clench and imagined left/right hand squeeze) are mapped to robot commands with some user training. Receiving these commands, a mobile robotic arm is a

solution for everyday object manipulation and transportation in household and occupation environments. To keep the number of inputs low and prevent users from getting fatigued, the proposed robot is semiautonomous and goal position oriented. The omnidirectional mobile robot base has obstacle avoidance capability and can plan paths through known free spaces. Likewise, the robot arm uses a vision system where the user specifies a goal point and the controller calculates the necessary joint movement to grab or release the recognized object.

The three integrated subsystems, BCI controller, input, mobile base, and arm, are developed separately to ensure that the individual criteria are met before the system level testing. For the BCI controller input, the active electrodes for the specified tasks are identified, a proper machine learning algorithm is selected, and the final controller configuration is created. For the robot base, rapid prototyping is used to generate the SLAM algorithm that localizes the robot within its environment while building a map of obstacles and free space from sensor data. For the robot arm, the incoming camera stream cloud data is calibrated to link kinematic algorithms and physical markers. To verify the subsystems, the software communicates and operates cohesively; virtual testing is performed in a simulation environment called Gazebo. The final version of the BCI controller input is paired with the robot base and arm. A simulated brainwave stream is used to run through every command and demonstrate the successful action. The following sections of the paper summarize the development of the subsystems: modeling of the proposed robot and the simulated environment, simulation and testing verification, results, and conclusions.

2. Motivation and Literature Review for Semiautonomous Mobile Robotic Assistive Systems

2.1. Motivation of Semiautonomous Mobile Robotic Assistive System. In 2011, the World Health Organization (WHO) reported that over a billion people (or about 15% of the world's population) were living with a disability [1]. This significant number of people worldwide shows that access to education and work opportunities, even performing activities of daily living (ADLs) at home, is unreachable by themselves and require caregiving personnel [2]. The additional impact of COVID-19 and epidemiological measures (i.e., stay-at-home orders) has added to the burden of caregivers and care recipients; one journal points that about one out of five care recipients had difficulty in obtaining care from outside the household and states the need for intervention [3]. While health policymakers and social organizations can certainly improve aid to the disabled, the deployment of semiautonomous mobile robotic assistive systems can empower one of the more severe handicap conditions, a quadriplegic, to perform daily tasks at home that normally require personnel in stay-at-home scenarios. In the journal, improving the autonomy for people suffering from paralysis in all four limbs is the primary focus.

2.2. Literature Review for Semiautonomous Mobile Robotic Assistive Systems. Semiautonomous mobile robotic assistive systems are in development in the forms of smart wheelchairs and robotic arms. For quadriplegics, or people suffering from amyotrophic lateral sclerosis (ALS), controlling the advance devices requires input from the user generating specific signals from the brain. Some researchers utilized the user's brain signal that mimicked the frequency of a specific flashing light on a screen [4, 5]. However, this approach is not optimal for operating a mobile robotic device due to the following: (1) driving a robot or operating a robotic hand requires the user's undivided attention of the surroundings and (2) waiting on the brain to mimic one of the flashing frequencies may take longer time to implement and become ineffective if the environment suddenly changes. As a result, the control chosen for this study is an asynchronous approach, where the user has to perform mental tasks and clench his/her jaw. Studies that favor this approach show that the action implemented by the user is quicker and accurate as if the training is conducted in real time [6–8]. In terms of hardware, combined smart wheelchairs and robotic arms are developed by universities and companies [9, 10]. While these have greatly increased the mobility and functional ability of the disabled, the end user is still required to operate either the wheelchair or the arm by either the joystick or the eye tracker. The assistive robotic system presented in this paper is designed to operate solely on the user's brain signals. In addition, since this proposed mobile robotic arm is separate from a wheelchair, the end user can retrieve objects without having the hassle of leaving his/her current position.

3. BCI Input Signal

3.1. 6-Class BCI Mobile Arm Controller: Schematic. Similar to a truck crane, the user operates the 6-class BCI mobile arm's base and arm separately (Figure 1). First, the robot base is moved to a desired location; then, the user switches to operating the arm's focus cursor to select an object that will be picked up. Likewise, to drop off the held object, the user navigates using the robot base to the new location and then switches to the robot arm once again to select the location to place the item. Since these two operations are isolated, the same mental tasks can be assigned multiple commands.

In this case, the asynchronous signals of the imagined right foot, left foot, right hand, and left hand correspond to the robot base moving forwards, backwards, right, and left; the robot arm cursor moves up, down, right, and left on a camera screen with the same mental tasks. An electromyogram signal of the user's jaw clench is used for transitioning between the robot base and arm and initiating a pick-up or drop-off action. When switching from the robot arm to the base, the jaw clench first confirms the desired object selected by the cursor. A built-in software calculates the distance between the robot and the object and moves the proper arm motors to pick up. Likewise, when dropping off the object, the user first guides the cursor and clenches his/her jaw to confirm the location point; the smart robot arm

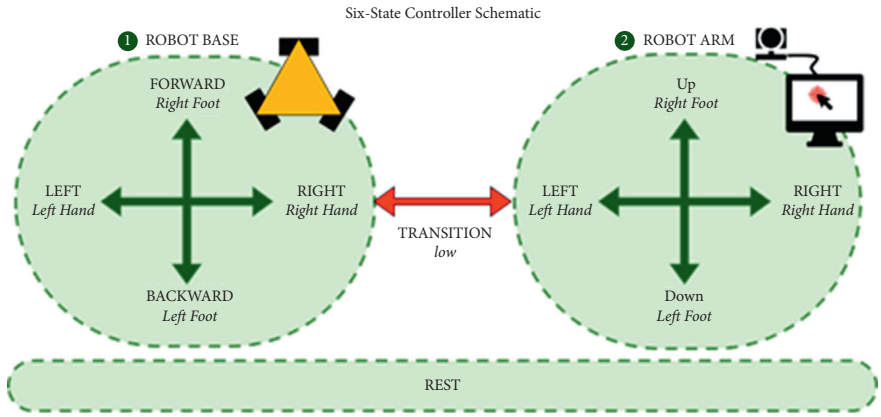


Figure 1: A 6-class BCI mobile arm controller with commands.

software then performs the action, freeing the user from the complexity of controlling every mechanical joint like the fingers and wrist. Additional sensors on the mobile base help detect obstacles or walls and avoid collisions while the user is giving movement commands.

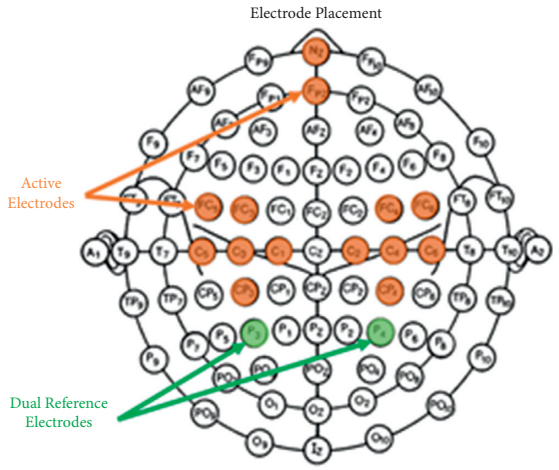
3.2. 6-Class BCI Mobile Arm Controller: Development. Development of the 6-class BCI controller involved identifying responsive electrodes on the scalp and selecting the appropriate machine learning algorithm for the targeted mental tasks. Three male subjects participated in a series of experiments where their brainwaves were recorded in response to physical and imaginary tasks (i.e., hand squeezes, foot taps, and jaw clenches). These recordings were preprocessed and analyzed using event-related potential plots and topographical maps [11, 12]. The results for each subject showed that the twelve electrode locations with respect to the 10–20 international system, FC3, FC4, FC5, FC6, C1, C2, C3, C4, C5, C6, CP4, and CP4, were capable of recognizing the unique brain signal characteristics of each task; as a result, this electrode layout is finalized and incorporated into the BCI controller headset (Figure 2(a)). To overcome the nonstationary nature and incoming clusters of other brainwaves, the next step is to downselect the appropriate machine learning algorithm between linear discriminant analysis (LDA) and relevance vector machines (RVM) (Figure 2(b)). Using the same preprocessed recordings, the error rate percentages for each algorithm were calculated and compared for two-, three-, five-, and six-class controllers. The results showed that RVM had a higher accuracy than LDA on multiclass controllers among all three subjects. As a result, RVM is selected for the final 6-class BCI controller configuration.

3.3. 6-Class BCI Mobile Arm Controller: Virtual Testing Architecture. With the 6-class BCI controller fully functional, the software architecture needed to support virtual testing is developed. Referencing to Figure 3, the block diagram starts with a compiled set of brainwave recordings that are looped to simulate a twelve-electrode channel EEG input stream from a user. The combined brainwave file of one male subject contains fifteen consecutive examples of each mental task and is placed in nearly sequential order: right foot (RF), left foot (LF), right hand (RH), left hand (LH), and jaw. By systematically running through the sequence of tasks, any discrepancy or deviation from this order is ruled as an error.

The "rest" mental task did not follow the order since it was embedded in the five other examples. Having more examples of the "rest" task is strategically done to bias the controller to keep the robot idle if it is unable to classify the user's brainwaves; this increases the controller's overall safety and reliability. With a complete runtime of 6 minutes and 25 seconds, the simulated brainwave stream is fed to the lab streaming layer (LSL). LSL is an open source multisignal acquisition system that collects unified measurement time series data and allows compatible programs to retrieve or add to the data [13]. This system's built-in code has libraries and wrappers that allow the transaction of data to occur across platforms with different languages (i.e., C/C++, MATLAB, Java, and Python). BCILAB, an open source MATLAB plug-in that contains the 6-class RVM BCI controller, is linked to LSL and retrieves the incoming stream of brainwaves. The controller actively compares the sequenced brainwaves to its trained examples and outputs a stream of integers that are uniquely associated with each mental task (Table 1); note that this integer represents the controller's highest calculated probability out of the six mental tasks. BCILAB then adds the controller's command stream of integers back to LSL for the virtual robot code to retrieve. Since both the robot mobile base and the arm subsystem's control center use Python, LSL converts the stream of integers into a compatible data type in Python. These control centers receive additional feedback from its semiautonomous simulated sensors before performing the intended action. In summary, the virtual testing block diagram starts with a raw stream of brain signals and ends with a robotic action dictated by a command stream.

4. Robot System

The robot system consists of 3 degrees of freedom (DoF) omnidirectional mobile base, to which a 3 DoF robotic arm



(a)

Machine Learning Comparison

Error Rate Percentage		subject 501		subject 502		subject 503	
		LDA	RVM	LDA	RVM	LDA	RVM
Hands	2-Class	26.7	26.7	30	43.3	10	10
Feet	2-Class	30	35	26.7	16.7	23.3	30
jaw/roof	2-Class	-	-	-	-	3.3	33
Hands	3-Class	41.7	40	28.3	20	16.7	20
Feet	3-Class	43.3	40	38.3	31.7	23.3	25
jaw/roof	3-Class	-	-	-	-	11.7	10
	5-Class	52.5	45	35.8	30.8	33.3	29.2
	6-Class	-	-	-	-	34.5	34.5

(b)

Figure 2: (a) Finalized electrode layout [1] and (b) LDA vs. RVM performance comparison.

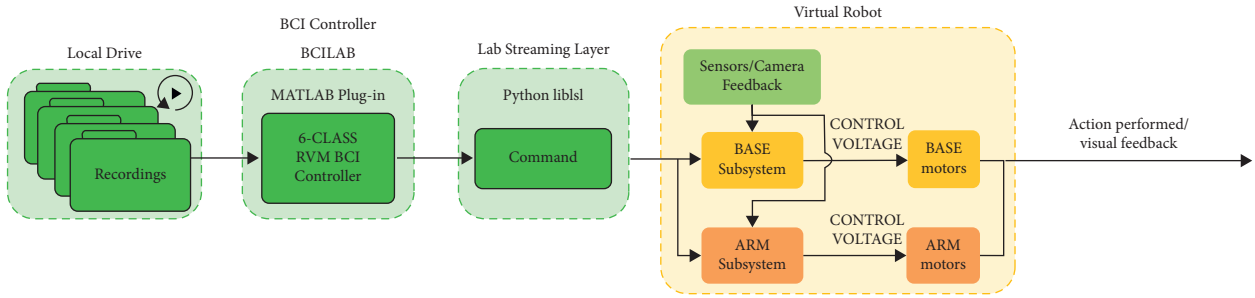


Figure 3: A detailed block diagram showing the virtual testing of the 6-class BCI controller.

Table 1: 6-class BCI controller reference showing the mental task input and associated robot action.

Input brainwave mental tasks	BCILAB output integers	Robotic base actions	Robotic arm cursor actions
RF	1	FORWARD	DOWN
LF	2	BACKWARD	U
RH	3	RIGHT	RIGHT
LH	4	LEFT	LEFT
JAW	5	N/A	GOAL POINT
REST	6	IDLE	IDLE

with a 4 DoF end-effector is attached. The total system together consists of 10 DoF which is controlled together. The principle idea of an omnidirectional wheel is that the central wheel is surrounded by rollers which are placed at an angle of either 45° or 90° around the periphery of the wheel. The rollers introduce a linear slip perpendicular to the face of the wheel, which does not exist in a fixed standard wheel [14]. Combined with the rotational motion of the wheel and the rotational twist at the contact point of the ground, omnidirectional wheels can achieve 3 DoF. As such, a mobile system which utilizes omnidirectional wheels is also capable of 3 DoF and can travel through an environment under any orientation. To achieve this, each omnidirectional wheel requires its own motor and controller. Referencing to Figure 4, the overall motion of the mobile robot is dictated by the rate of rotation of each wheel [15].

The wheels of a mobile robot must rotate at appropriate speeds to achieve a desired chassis speed, which includes both the linear and the rotational velocities. To determine the required speeds of each wheel, the kinematics of the mobile robot are studied. The kinematic calculations were derived from the works of [16] and were modified to the specific notations as shown in Figure 5. The work was also simplified by using the body twist V_b, which does not rely on the robot's orientation relatively to the work environment but rather the angular velocity of the chassis:

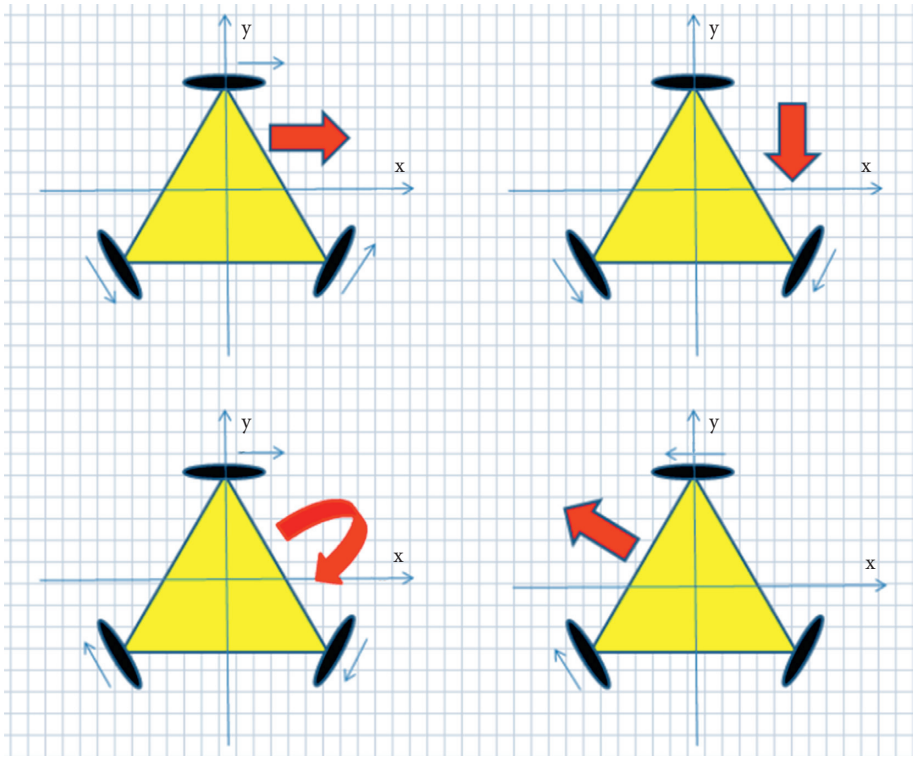


Figure 4: Robot body motion and required wheel rotation.

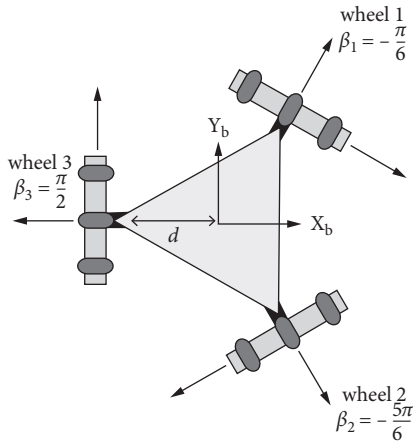


Figure 5: Kinematic model of the robot [6].

$$V_b = \begin{bmatrix} \omega_b \\ v_x \\ v_y \end{bmatrix}. \tag{1}$$

$$u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = H(0)V_b = \frac{1}{r}\begin{bmatrix} -d & \cos\left(-\frac{\pi}{6}\right) & \sin\left(-\frac{\pi}{6}\right) \\ -d & \cos\left(-\frac{5\pi}{6}\right) & \sin\left(-\frac{5\pi}{6}\right) \\ -d & 0 & 1 \end{bmatrix}\begin{bmatrix} \omega_b \\ v_x \\ v_y \end{bmatrix}. \tag{2}$$

As referenced in Figure 5, d is the distance from the center of the robot to the contact point of each wheel and β_i is the driving angle of each omnidirectional wheel relative to the X_b-axis of the body reference frame. Thefollowing equation is the solution of the kinematic model and expresses the wheel rotational velocities with respect to the body twist:

Letting $d = 0.1554$ m and the wheel radius $r = 0.05$ m, the wheel velocity can be expressed as

$$\begin{aligned} u_1 &= \frac{1}{0.05}\left(-0.1554 * \omega_b + 0.866 * v_x - 0.5 * v_y\right), \\ u_2 &= \frac{1}{0.05}\left(-0.1554 * \omega_b - 0.866 * v_x - 0.5 * v_y\right), \\ u_3 &= \frac{1}{0.05}\left(-0.1554 * \omega_b + v_y\right). \end{aligned} \tag{3}$$

The 3 Dof robot arm consists of an elbow, shoulder, and wrist joint. The arm is rigidly attached and has a planar workspace parallel to wheel 2 as shown in Figure 6. The end-effector used is the Brunel Hand 2.0 and is attached to the wrist joint of the arm. The kinematics of the manipulator is solved using the mobile base as the fixed reference. For the end-effector to reach a goal target, the inverse kinematics first solves for a goal location of the mobile base that places the goal target in the arms workspace. This goal location is solved based on a desired arm pose when grabbing an object.

The goal target is the input from a user interface that uses BCI commands to select a point from a camera feed. The camera feed provides the point cloud data used to measure

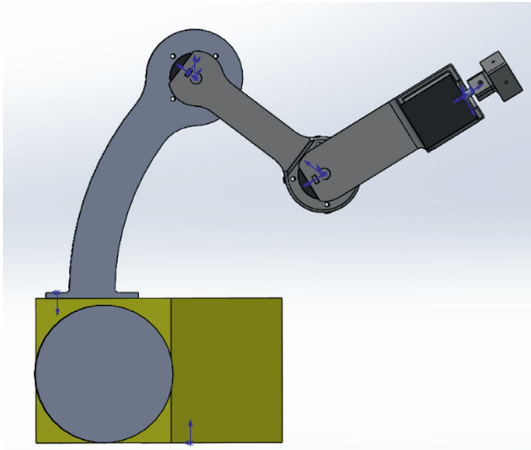


Figure 6: Robot arm 3D model.

distance from the robot. Using end-effector control allows the robot to automatically solve for the movement of the entire system [17]. This semiautonomous mode uses the camera system to facilitate the user input while providing the more complex robot movements required for object manipulation [18]. Once the robot arm achieves the desired pose, the Brunel Hand initiates a grabbing command which has been physically tested on objects [19].

5. Virtual Software Interface

Due to the restrictions caused by COVID-19, a physical build of the robot was not possible, and the project was moved to a virtual environment. Robot Operating System (ROS) and Gazebo are used to visualize and simulate the entire assistive robot system. The user interface is created using RViz, a visualization tool available in ROS, and is shown in Figure 7.

The user interface consists of a visualization display on the left and a camera feed for user input on the right. During operation, a cursor is bounded within the right camera feed. The cursor moves and can select points using BCI commands. The camera cannot move since it is not connected to the mobile robot; therefore, visualization markers are created to show the intended motion of the mobile base and appear in the visualization display. The motion is also simulated in the Gazebo environment, which offers the use of joint torque limits and collision physics. The simulation environment uses the robot's collision and inertial models alongside joint torque and velocity limits to simulate the results of input commands. PID controllers are used for each joint and the outputs of these controllers are sent to the Gazebo environment.

Figure 8 shows a portion of the block diagram for the robot system. After including the torque and velocity limits of the selected motors to the robot definition, the PID controller of each joint is tuned. The arm uses position controllers and is tuned to reach different grabbing poses with no steady state error. The controllers for the omnidirectional wheels are designed with the assistance of the Joint Effort Interface. This ROS controller converts the desired joint speed into a torque or force required for the joint to achieve the goal velocity. The resulting speed is recorded by the sensors and compared to the desired speed to determine the error of the system.

6. SLAM

The ability of accurately estimating the position of a mobile robot as it travels through an environment is a computationally challenging problem, as it is difficult to calculate the instantaneous speed of a robot. Without an accurate measure of the speed, it is hard to estimate how far it has traveled after a period. To produce an estimate, a robot must perform odometry by identifying obstacles within its immediate vicinity. Tracking obstacles and calculating the change of the robot's distance between them can help estimate the speed at which the robot is traveling. The collected odometry data can be used to achieve any high level of autonomous control. The inclusion of a map of the environment can assist in localizing the robot around known obstacles and completing path planning calculations. A robot which can also produce or update a map of the environment while localizing itself to the map can achieve even higher levels of autonomous control as it can account for known obstacles. This computational capability is what is known as Simultaneous Localization and Mapping (SLAM). There are many different SLAM algorithms which can achieve the same purposes [19–22]; most just rely on differing sensors employed in the system. The SLAM algorithm employed in this project is developed with the use of ROS nodes which rely on the sensor data of the entire system.

The localization node runs a function called Adaptive Monte Carlo Localization (AMCL) [23]. AMCL parses the data from laser scans and cameras to determine the distance from the robot to obstacles. The obstacles are compared to those identified on the map to create an approximation of the robot's current location. The AMCL algorithm compares to the robot odometry which is calculated by the data recorded by encoders on the motor which estimates how fast the robot is moving and therefore how far it traveled. This cross-reference is used to verify the measurements from the sensor data and improve the accuracy in the localization approximation.

The mapping node, known as Move Base [24], is responsible for updating costmaps by marking obstacles and free space as well as running the trajectory planning algorithm. This project is developed with the use of a prebuilt map of the environment, which is provided to the robot upon deployment. Move Base is responsible for creating two separate costmaps which are built on top of the existing map. In this setup, the boundaries of the environment remain consistent, but the changes of obstacles within the environment are updated as they change. One of the costmaps is a global costmap, which will mark and record obstacles even once they are out of range of the robot's sensors and no longer an immediate obstacle. The other costmap is a local costmap, which only keeps track of obstacles observed within a defined distance from the robot. Both costmaps are

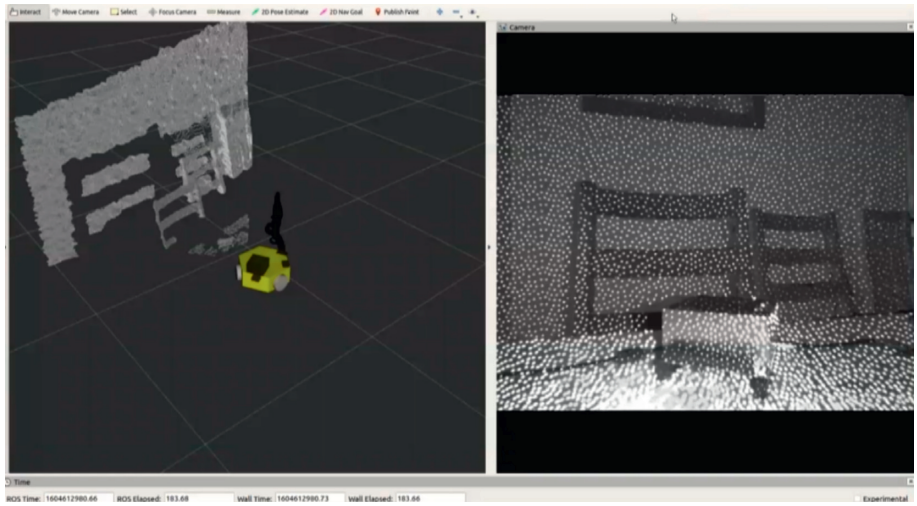


Figure 7: User interface using RViz.

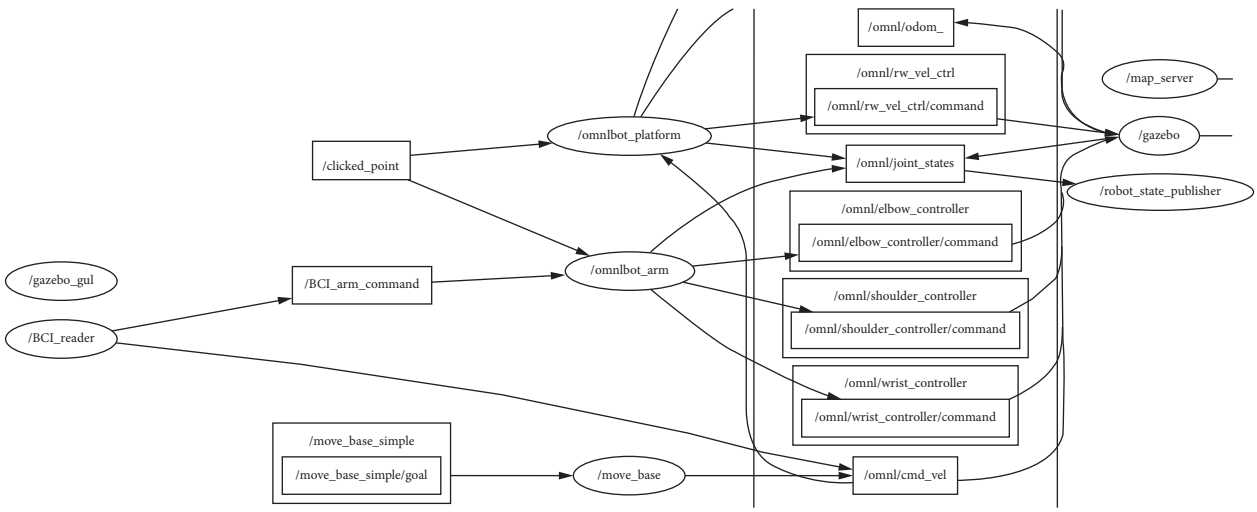


Figure 8: Block diagram of the robotic system.

updated at the same time; the only difference is how long each one records the obstacles.

Move Base handles trajectory planning with the Dynamic Window Approach (DWA) planner [25]. Given a goal destination, a global path is set within the global costmap. It follows the free space and avoids as many obstacles as possible, while still following the shortest path. The DWA planner then performs a sampling of forward simulations, by calculating the outcomes of running the robot a short distance along the global plan at varying speeds. Each forward simulation is scored by weighted criteria which determine the safest and shortest route possible for the robot. Forward simulations which result in collisions are immediately discredited. The trajectory of the best forward simulation is sent to the robot system and the DWA planner performs another round of simulations based on the change of location of the robot since the last simulation. The deployment of this SLAM algorithm permits the robot to safely navigate from one goal to the next. The routine checks accurately account for the robot and allow for precise remote BCI control. They also keep the robot from crashing into any obstacles which may have been moved and also account for any delays in the data stream.

7. Methods

7.1. Experimental Setup. To ensure that the BCI controller, the robotic base, and the arm subsystems all work cohesively and a virtual demonstration is built to run through every command, the virtual version of the robot assembly is designed with coded sensors and cameras to closely represent the real hardware. In addition, naturally occurring phenomena, like coefficients of friction between the wheels and floor, and moments of inertia for the motors driving the arm, hand, and wheels, are included. The virtual environment in Figure 9 is a single-floor space with typical objects (i.e., chairs, tables, and trash cans) and walls. This is used as a sandbox for testing the entire assistive robot system.

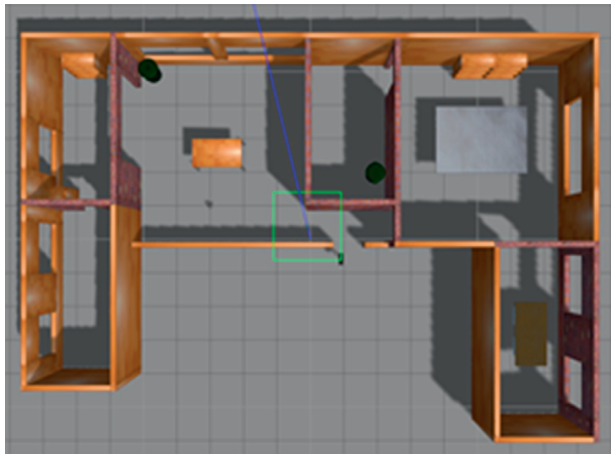


Figure 9: Environment in Gazebo.

7.2. Experimental Procedure. To generate the command input stream used for the tests, the BCILAB MATLAB plug-in is launched. The incoming EEG signal stream is configured to read the recorded brainwave data with the following parameters: input stream name "laststream," an update frequency of 20 Hz, an internal buffering length of 10, a conversion to double precision data type, and a looped playback. With the incoming stream being processed by the 6-class RVM BCI controller, the controller's output stream of integers is then configured to be written to LSL with the following parameters: command stream name "bci," output form "mode," and an update frequency of 10 Hz. A verbose output is checked to ensure that the command stream generated in MATLAB is being sent correctly to LSL. The robotic base and arm have a shared node which reads the stream as it comes in over LSL and collects data over the span of five seconds. The BCILAB "modes" form outputs what the BCI controller identifies as the most likely user input command, which can fluctuate slightly with artifacts and other oddities in the EEG signals. By recording all output during a five-second interval, the intended command output as determined by the controller is guaranteed to fall into that window; that is, how the EEG data is epoch. The ROS node reading the LSL stream then takes the mode of input signals to determine the appropriate action that the robotic base or arm should perform as referenced in Table 1.

Because both robotic subsystems operate independently, the virtual demonstration of controlling the mobile robotic arm is done in a modular approach. First, the recorded brainwaves and controller are used to move the robot base in the simulated environment based on the actions in Table 1. Next, the 6-class BCI controller goes through the string of commands in Table 1 to control a cursor on the user input camera feed and creates a goal point to initiate a pick-up or drop-off action. The entire system is tested using the goal point control. The processed BCI input signals move a cursor over a video feed and selects an object for the arm to grab with its manipulator. A grabbing pose is selected, and the corresponding inverse kinematics solves for the goal location of the mobile base to place the goal object on the arms grabbing point. The user selects an arm pose that dictates the set of inverse kinematic equations which needs to be solved. An application may require the manipulator to keep an object in its original orientation or to fully reach out to the end of the workspace. These conditions allow for a unique solution of joint angles with minimal user input. A sample of the inverse equations to solve an arm pose to keep the end manipulator horizontal is shown in Figure 10. With a selected goal point TP, the shoulder and elbow joints are solved to satisfy the final arm pose ends at TG after rotating and moving the mobile base a distance R_Y.

The shoulder angle θ_s is calculated using the first following equation, a horizontal arm pose desired θ_e is calculated using the second following equation, and the last equation is used to find the mobile base goal location R_Y. The vector space approach is used to derive the equations of different arm poses, including a fully extended pose that reaches the end of the workspace and is used in testing to verify max load conditions:

$$L_s C\theta_s = \left({}^T P_z + d_T\right), \tag{4}$$

$$\theta_s + \theta_e = 90^\circ, \tag{5}$$

$${}^T P_Y = L_s S\theta_s + \left(L_e + h_{\text{offset}}\right) + R_Y. \tag{6}$$

With the assistance of the implemented SLAM algorithm, the mobile base maneuvers until it reaches the goal location. Once the goal location is reached, the arm moves to the grabbing point and the grabbing/releasing operation begins. For these tests, the system did not use a live feed of BCI commands but rather a loop of prerecorded signals.

8. Results

The developed 6-class RVM BCI controller had an error rate percentage of 34.5%. When applying the recorded brainwave file to the controller, BCILAB recalculated the controller accuracy on the simulated data to 18.5%. The increase in accuracy is most likely due to the combination of reused brain recording segments used to train the controller. While this is not entirely representative of in-person testing, the overall objective is to demonstrate that the 6-class BCI controller can use brainwaves as a command input to control the robot. The controller designed for the three wheels of the omnidirectional mobile system only utilized the proportional and integral gains to achieve the desired results. Issues within the physics simulation prevented the controllers from removing all errors from the wheel velocity goal, but, as Table 2 shows, the steady state error was reduced as much as possible to produce acceptable results.

Figures 11–14 show the results of the experimental procedure for the entire system. Figure 11 shows the robot in the Gazebo environment waiting for a command. A point is selected through the user interface and the goal position visualization markers are created, as shown in Figure 12. The

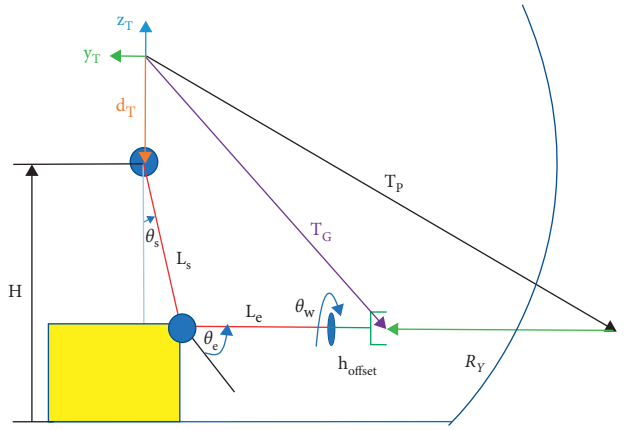


Figure 10: Workspace plane view of horizontal grabbing pose.

Table 2: PI gains and error for each omnidirectional wheel.

Controller	*P*	*I*	Error %
Wheel 1	0.060375	0.04	4.5
Wheel 2	0.06035	0.04	1.26
Wheel 3	0.052	0.04	3.93

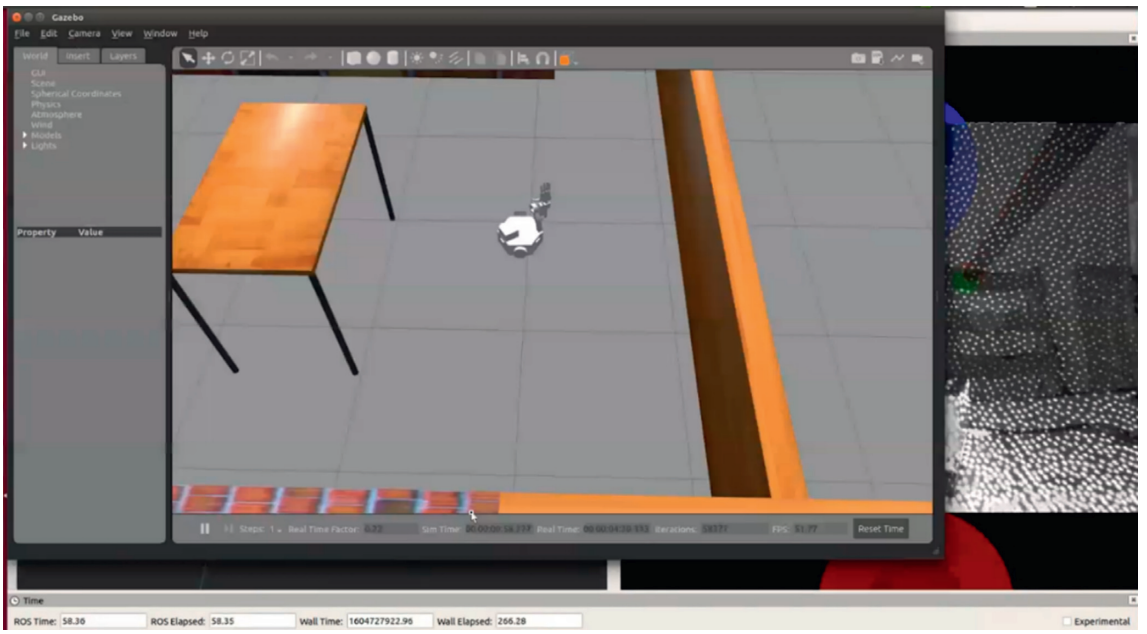


Figure 11: Starting robot position in Gazebo.

green vector is the user input and points to the goal point from the camera. The red vector points to the goal point from the manipulator frame and is used to solve for the movement of the mobile base. The blue vector is the location of the manipulator after it moves to its grabbing pose, in this case extending to the edge of the arm workspace.

The white vector is the difference between the goal point and the grabbing point, which defines the goal destination

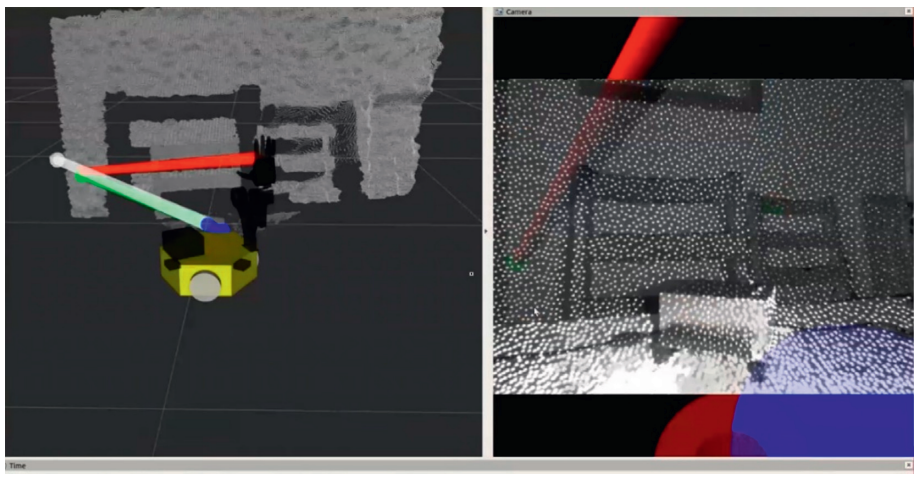


Figure 12: BCI cursor control to select point from camera feed in RViz.

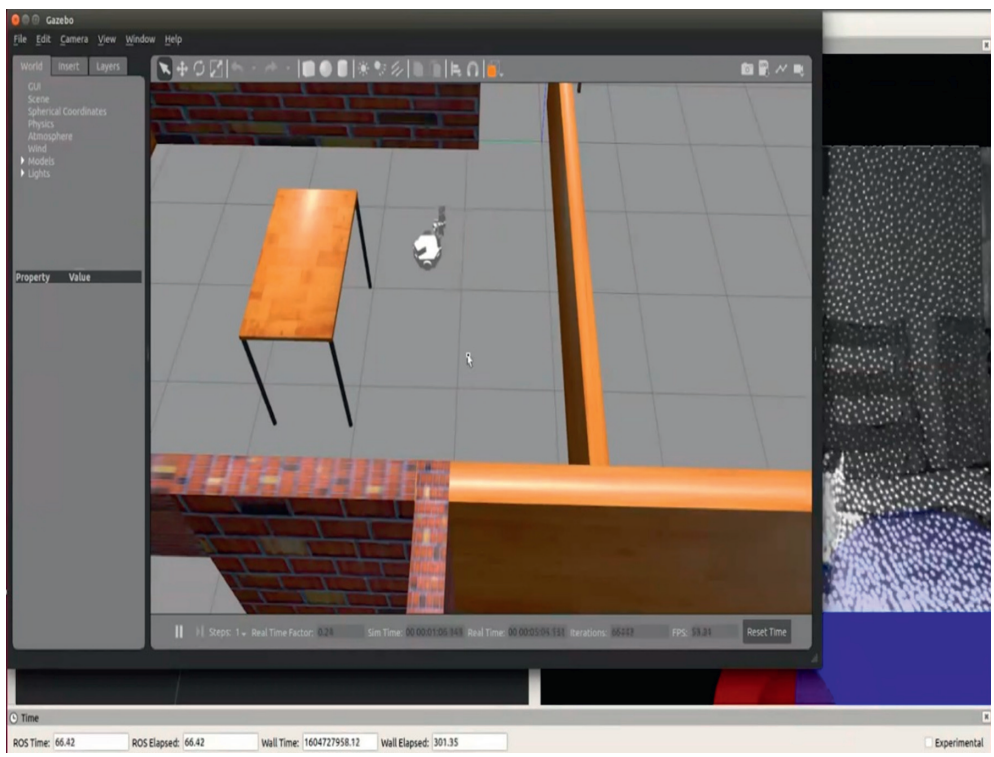


Figure 13: Robot moving in Gazebo after BCI input is processed.

for the mobile base. With the solved mobile base goal location, a path is planned for the robot to move in the Gazebo environment, as shown in Figure 13, using the SLAM algorithm. Figure 14 shows a top view of the robot arm pose to grab an object after the mobile base reaches its goal destination.

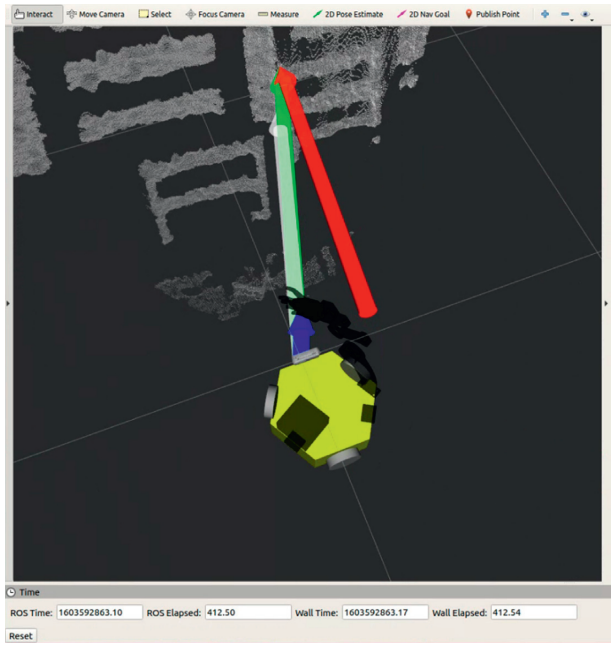


Figure 14: Robot arm movement to grabbing point.

9. Conclusion

This study implemented three independently developed subsystems: a hybrid BCI controller, an omnidirectional mobile base, and a robotic arm into one semiautonomous assistive robot. Each subsystem of the robot was tested independently and allows for changes to be easily made based on the desired application. In these tests, goal location steady state errors were minimized to ensure successful grabbing operations. The errors associated with the velocity controller performance of the wheels are addressed to have more realistic simulations, but the goal location of the mobile base using SLAM places the robot on the desired location. The process to tune the controllers of a physical robot follows the same procedure and can be done in the ROS environment, allowing for quick deployment based on simulation results and errors. Future work is to use the live data to test robot usability with added delays in the system. This test will accurately highlight the aspects of the system which need improvement to design a responsive, intuitive BCI controlled robot. The camera feed comes from an Intel RealSense Depth Camera D435 to verify its use in the physical build of the robot. To develop more robust tests using live data in this environment, virtual point cloud data should be used to interact with the virtual environment directly before physical testing. The manipulator was tested independently in a lab environment, but, due to pandemic access restrictions, different end-effectors were imported to this virtual environment to test the grabbing and placing command loops and confirmation using BCI. The robot control environment developed during this research allows a user to interact with a simple interface to achieve complex robot movements.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] World Health Organization (WHO), *World Report on Disability*, WHO Press, Geneva, Switzerland, 2011.

[2] P. F. Edemekong, D. L. Bomgaars, S. Sukumaran, and S. B. Levy, *Activities of Daily Living*, StatPearls Publishing, Treasure Island, FL, USA, 2022.

[3] M. Bergmann and M. Wagner, "The impact of COVID-19 on informal caregiving and care receiving across Europe during the first Phase of the pandemic," *Frontiers in Public Health*, vol. 9, pp. 1–17, 2021.

[4] R. Spataro, A. Chella, B. Allison et al., "Reaching and grasping a glass of water by locked-in ALS patients through a BCI-controlled humanoid robot," *Frontiers in Human Neuroscience*, vol. 11, no. 68, pp. 68–10, 2017.

[5] J. Tang, Y. Liu, D. Hu, and Z. Zhou, "Towards BCI-actuated smart wheelchair system," *BioMedical Engineering Online*, vol. 17, no. 1, p. 111, 2018.

[6] D. Huang, K. Qian, D.-Y. Fei, W. Jia, X. Chen, and O. Bai, "Electroencephalography (EEG)-based brain-computer interface (BCI): a 2-D virtual wheelchair control based on event-related desynchronization/synchronization and state control," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 20, no. 3, pp. 379–388, 2012.

[7] R. Leeb, D. Friedman, R. G. Muller-Putz, R. Scherer, M. Slater, and G. Pfurtscheller, "Self-paced (asynchronous) BCI control of a wheelchair in virtual environments: a case study with a tetraplegic," *Computational Intelligence and Neuroscience*, vol. 2007, Article ID 79642, 8 pages, 2007.

[8] R. Bousseta, I. El Ouakouak, M. Gharbi, and F. Regragui, "EEG based brain computer interface for controlling a robot arm movement through thought," *IRBM*, vol. 39, no. 2, pp. 129–135, 2018.

[9] A. Campeau-Lecours, V. Maheu, S. Lepage et al., "JACO assistive robotic device: empowering people with disabilities through innovative algorithms," in *Proceedings of the Rehabilitation Engineering and Assistive Technology Society of North America (RESNA)*, Alrington, TX, USA, 2016.

[10] S. H. Sunny, I. I. Zarif, I. Rulik et al., "Eye-gaze control of a wheelchair mounted 6DOF assistive robot for activities of daily living," *Journal of NeuroEngineering and Rehabilitation*, vol. 18, no. 173, pp. 1–12, 2021.

[11] M. Seeck, L. Koessler, T. Bast et al., "The standardized EEG electrode array of the IFCN," *Clinical Neurophysiology*, vol. 128, no. 10, pp. 1–8, 2017.

[12] K. Matsuno and V. K. Nandikolla, "Machine learning using brain computer interface system," in *Proceedings of the ASME 2020 International Mechanical Engineering Congress and Exposition*, Portland, OR, USA, 2020.

[13] C. A. Kothe, D. Medine, C. Boulay, M. Grivich, and T. Stenner, "LabStreamingLayer's documentation," 2019, https://labstreaminglayer.readthedocs.io/index.html.

[14] A. Florentina and D. Ioan, "Practical applications for mobile robots based on mecanum wheels—a systematic survey," *Romanian Review Precision Mechanics, Optics and Mechatronics*, vol. 40, pp. 21–29, 2011.
[15] V. Kálmán, *On Modeling and Control of Omnidirectional Wheels*, Budapest University of Technology and Economics, Budapest, Hungary, 2013.
[16] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*, Cambridge University Press, Cambridge, UK, 2017.
[17] J. Tang, Z. Zhou, and Y. Yu, "A hybrid brain computer interface for robot arm control," in *Proceedings of the 2016 8th International Conference on Information Technology in Medicine and Education*, pp. 365–369, Fuzhou, China, 2016.
[18] J. Kofrnan, X. Wu, T. J. Luu, and S. Verma, "Teleoperation of a robot manipulator using a vision-based human-robot interface," *IEEE Transactions on Industrial Electronics*, vol. 52, no. 5, pp. 1206–1219, 2005.
[19] B. Landavazo and V. K. Nandikolla, "Brain-computer interface application in robotic gripper control," in *Proceedings of the ASME 2018 International Mechanical Engineering Congress and Exposition*, Pittsburgh, PA, USA, 2018.
[20] P. Kim, J. Chen, and Y. K. Cho, "SLAM-driven robotic mapping and registration of 3D point clouds," *Automation in Construction*, vol. 89, pp. 38–48, 2018.
[21] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, "An overview to visual odometry and visual SLAM: applications to mobile robotics," *Intelligent Industrial Systems*, vol. 1, no. 4, pp. 289–311, 2015.
[22] R. Martinez-Cantin, N. D. Freitas, and J. A. Castellanos, "Analysis of particle methods for simultaneous robot localization and mapping and a new algorithm: marginal-SLAM," in *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, Rome, Italy, 2007.
[23] ROS Basics for AMCL Documentation, http://wiki.ros.org/amcl.
[24] ROS Basics for Move Base Documentation, http://wiki.ros.org/move_base.
[25] ROS Basics for DWA Local Planner Description, http://wiki.ros.org/dwa_local_planner?distro=melodic.