Advances in Intelligent Systems Research, volume 143
2nd International Conference on Applied Mathematics, Modelling and Statistics Application (AMMSA 2018)

Study on Mind Controlled Robotic Arms by Collecting and Analyzing Brain Alpha Waves

Yue Han[1,*], Yihe Ma[1], Lingkai Zhu[1], Yanpeng Zhang[1], Li Li[2], Wei Zheng[1], Junshan Guo[1] and Yongqiang Che[1]
[1]State Grid Shandong Electric Power Research Institute, 2000 Wang Yue Road, 250002 Jinan, China
[2]State Grid of China Technology College, 500 Er Huan South Road, 250002 Jinan, China
[*]Corresponding author

***Abstract*—Assistive robotic technologies that use neural interface systems are designed to allow people with limited mobility to assert control with signals directly from their brains. These robotic systems require detection and analysis of raw brain signals, machine learning methods to extract these signals into useful commands, and the development of an interface between neural signals and robot control. In this paper, a method for controlling a 4-degree of freedom RRRR WAM robotic arm with alpha brain waves of a test subject obtained via electroencephalography (EEG) is presented. The OpenBCI system electrodes and board are ussubed to detect alpha waves and transmit them to digital signal. A robust serial communication interface is developed to convert OpenBCI data into robot commands. An accelerometer embedded in the OpenBCI board is used to implement left-right motion of the robot. To assess the performance of the system, we successfully demonstrate two primary tasks: alpha wave robot control and alpha wave and accelerometer robot control. The methods can be readily extended to include control from other brain regions and additional robotic tasks, paving the way for more complex interactions between robots and human brains.**

Keywords—mind controlled; robitic arms; brain alpha waves

I. INTRODUCTION

Brain-computer interfaces (BCIs), which describe the communication between a device and the human brain, are becoming a widely researched topic with applications ranging from gaming to neuromarketing and advertisement [1]. In addition to these non-medical applications, BCIs are proving to be useful tools in patient- assistive technologies as well. For patients with limited mobility, the majority of current assistive technologies rely on motor inputs for robotic control through manual interfaces such as joysticks and keyboards. However, for patients with extreme levels of motor impairment due to illnesses such as stroke, amyotrophic lateral sclerosis (ALS), and multiple sclerosis (MS), these technologies are ineffective at providing increased mobility. BCIs are beginning to fill this large gap in assistive technologies because they do not rely on motor input but rather use human brain waves alone to communicate with robots. In particular, electroencephalography (EEG)-based brain-controlled robots provide a robust, non-invasive method for assistive human technologies.

EEGs are particularly useful in brain-controlled assistive technologies due to their low cost, ease of use, and good temporal resolution [2]. However, there are several weaknesses in the use of EEG in robotics, such as the high level of noise in obtained measurements, which causes difficulty in task classification [2]. Current research efforts are aimed at addressing these challenges in order to provide robust and accurate pattern classification.

This study uses EEG signals obtained from a test subject to control a WAM robot arm. In particular, we use signal processing and support vector machines to classify brain waves from a test subject and feed the result into the control for the robot arm. In this way, we demonstrate a method to perform basic tasks using brain-controlled robots and proof-of-concept for future applications of complex human-robot interaction.

II. IMPLEMENTATION

A. EEG Protocol

The first task depends on detection of alpha brain waves. These can be detected on EEG by asking the test subject to close their eyes and observing a peak in EEG signal in the 8-13 Hz range [3]. In contrast, when the subject opens their eyes, there is a significant reduction in signal in this range. EEG placement on the subject's head is optimized in order to precisely record the alpha wave signal and use it to control robot motion. A custom EEG cap is created from elastic material to ensure secure lead placement on the subject's head.

Classically, alpha wave signal can be most strongly detected in the occipital lobe of the brain. While the bulk of the signal comes from this region, finer alpha signal can also be obtained from other areas in the brain– namely, the parietal lobe and central lobe. Three elastic bands are placed over the central lobe, parietal lobe, and occipital lobe.

Lead placement is determined according to the international 10-20 system of electrode placement, as seen in Figure 1, in order to accurately detect signal from the three areas of interest mentioned above. A summary of lead positions can be found in Table 1. In summary, the electrodes are placed at 10 and 20 percent intervals of various perimeter measurements of the skull. The electrodes are secured using Ten20 conductive neurodiagnostic electrode paste. This paste serves a dual function of securing the electrodes and ensuring high levels of conductivity.

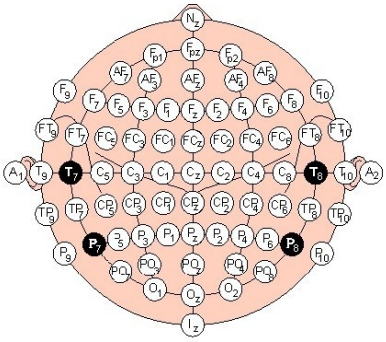


FIGURE I. INTERNATIONAL 10-20 SYSTEM OF ELECTRODE PLACEMENT FOR EEG [4]

TABLE I. FORMATTING SECTIONS, SUBSECTIONS AND SUBSUBSECTIONS.

Central Lobe	Parietal Lobe	Occipital Lobe	Ground and Reference
Cz	Pz	O1	A1
C3	P3	O2	A2
C4	P4		

B. *End effector Design*

Since the WAM robot does not possess an inherent end-effector, we present several designs for potential end- effectors to attach to the last joint and demonstrate versatility of the system. The first design is a thumbs-up model, which simply moves up and down when the alpha-wave is detected by the OpenBCI sensor. This is a simple way to verify that the implementation of the alpha-wave collection and robot control is successful. The second design is a Pen-holder that has two components: an inner hole to fix the pen direction while moving, and a spring support to increase the flexibility of the writing.

This design can potentially demonstrate more complex controls by brain signals. For instance, the robot can draw desired trajectories based on alpha wave detection. The third design is a Laser-pointer holder. This interesting design can help to show the desired path on target poster-board in response to varying detection of alpha wave.

C. *Data Analysis*

The first task consists of detecting alpha waves, which are waves in the 9 to 14 cycle frequency range that arise when a person is in a non-aroused or relaxed state [5]. The presence of alpha waves causes the robot to move in a particular direction, while the lack of alpha waves causes it to reverse its direction. To detect the relatively high-amplitude alpha waves, the following method is used: A Fast Fourier Transform computes the Discrete Forward Fourier of a filtered, five second sequence of data (updated each second). Specifically, we perform (1) Butterworth filtering and (2) Alpha wave extraction by detecting a 10 Hz signal amplitude in the computed FFT. We used the Eigen/FFT interface to perform FFT and IFFT operations on the signals for filter convolution within C++, and we gathered the Butterworth filter magnitudes using Python's scipy module which we saved into a file (butterworth.txt). Simply put, defining the butterworth frequency response as b and our raw data buffer signal as s, we perform the convolution sf=b∗s, where sf represents the filtered signal. The next task consists of moving the robot along a single dimension using mental signals that result from specific motions (such as moving the right hand, then the left hand) or by using the accelerometer provided on the OpenBCI unit.

We started by collecting data to enable right-left robot motion. Specifically, we collect data resulting from repeatedly clenching and un-clenching the right hand for two minutes, as well as tapping a hard surface for two other minutes. We then repeat the data collection for the same motions using the left hand. To classify the data as well as to determine which motion would perform better in moving the robot, we separate the data into a training and test set, and classify them using an SVM implemented in Python using the scikit library [6].

D. *Overall Program Structure*

We run three threads simultaneously in order to implement the robot control: the serial thread (functions inside OpenBCI Board and EEGWAMBot classes), the robot control thread (functions inside EEGWAMBot class), and the graphics thread (GLUT library). The overall structure of the classes is shown in Figure 2. The serial thread, control thread and graphics thread are all necessary to control the robot in real time using the EEG device.

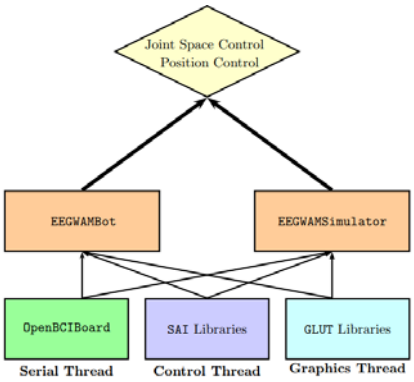


FIGURE II. OVERALL STRUCTURE OF THE CLASSES

Using a matplotlib plugin for C++, we integrated Cython into our program to do real-time plotting of the last five seconds of EEG data. This allowed us to improve our debugging capabilities as we reached the homestretch of the project.

E. *Serial Communication*

The already-provided Python interface was converted into a C++ interface to parse data packets streaming from the EEG system into our data processing algorithm for analysis. As the computer has to handle multiple threads, we used a thread-safe library called libserial to read and write to the port. The structure of the data packet is shown in Figure 3, where each block represents a byte of data coming in from the serial port. The auxiliary data represents the accelerometer data which we use for Task 2. This implementation is contained in the class OpenBCI Board. In order to actually transfer the data analysis onto a robot action, we allow a callback function as an input into our streaming function.

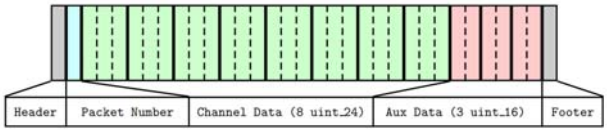


FIGURE III. THE STRUCTURE OF THE DATA PACKET

The callback function accepts as input a buffer sequence and the buffer sequence is updated every second to issue the next command to the robot. Since the callback function is in a while loop after each group of 250 samples is collected, we run our analysis every second, which we define to be 1250 samples, or the past 5 seconds. In order to communicate between the robot and the stream, we have a global variable ALPHA WAVE VALUE that we update in the callback function, which is also accessed by the robot control function.

F. Robot Control

The WAM robot is used, which is set up as a 4-dof RRRR-bot without orientation control, as shown in Figure 4. In the WAM, there is a base revolute joint (along Z_{Base}; Z_0; Z_2) which exerts little work and thus requires minimal torque to rotate, the revolute "heavy-lifter" (along Z_1) joint that is used to lift the arm for various tasks, another revolute joint (along Z_{Base}; Z_0; Z_2) at the same position as the heavy lifter to twist the arm, and finally, a revolute joint (along Z_3) to move the end effector around.

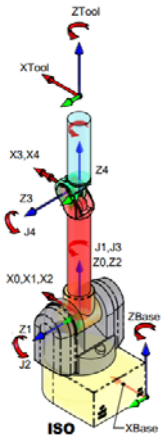


FIGURE IV. WAM ROBOT

1) Task 1: Alpha wave control

For the first robot control task, we started out with a few simple tasks in simulation for debugging using EEGWAM Simulator. The simple tasks include moving in an ellipse or cycling through a list of positions, implemented using a position control law that calculates the forces to move joints along our defined trajectory.

The control law looks like:

$$\Gamma^0 = J_v\,[\Lambda k_p(x_d - x) - k_v\,x] - g \qquad (1)$$

where x_d represents the desired path trajectory and x represents the current positions of the robot. The gravity compensation g is actually handled internally by the robot. Because we only have a need for three degrees of freedom, we made a slight modification to the force in order to reduce the null space movement of the first joint. When adding our null space damping term, we also exert a joint space force to keep the joint value of the first joint at zero. This means calculating the following for our final control law:

$$\Gamma = \Gamma^0 + \beta_{ns}\,(\Gamma_{ns} + \Gamma_0\,)$$

$$= \Gamma^0 + \beta_{ns}\,[\Gamma_{ns} + (k_p\,q_0 + k_v\,q_0)]\alpha\; + \beta\; = \chi \qquad (2)$$

Notice that here, we are only implementing joint space control on joint zero and the total sum of the forces makes the robot use the remaining three joints to execute our intended task. Our final parameters for the above task were: β_{ns}=3.0, k_p=200, k_v=30. In both the path generation and the ellipse following tasks, the robot moves along a given path and switches direction based on the value of ALPHA WAVE VALUE. In order to accomplish this, we had to define our x_d equation as follows

$$x_d\,[0] = x_{init}\,[0] \qquad (3)$$

$$x_d\,[1] = x_{init}\,[1] + 0.15 \cdot \sin(2\pi f\; sgn[\alpha > \alpha_{thresh}\,](t - t_{toggle}\,)) + x_{d,toggle}\,[1] \qquad (4)$$

$$x_d\,[2] = x_{init}\,[2] + 0.15 \cdot \cos(2\pi f\; sgn[\alpha > \alpha_{thresh}\,](t - t_{toggle}\,)) + x_{d,toggle}\,[2] \qquad (5)$$

where f is the frequency, α is the magnitude of the 10 Hz frequency bin in the EEG signal, $x_{d,toggle}$ represents the last position the robot was at before toggling, t_{toggle} is either 0 or the last time the robot toggled, and $sgn[\alpha > \alpha_{thresh}\,]$ is +1 when $\alpha > \alpha_{\text{thresh}}$ and -1 otherwise. For α_{thresh}, we use hysteresis control by defining a max-threshold of 7×10^{-4} μV and a min-threshold of 5×10^{-4} μV when alpha waves are not detected respectively. This allows us to smooth the task over time. Bringing these limits closer together enables to get finer control but may also allow noise of alpha wave detection to start dominating.

2) Task 2: Alpha wave + accelerometer control

For the second task, we implement a combination of alpha wave and accelerometer control. We have the WAM move in speeds proportional to the detected alpha wave signal and accelerometer magnitude along the roll axis of the board. To move the robot to either left or right, we tilt our heads to the respective directions. To lift the robot up, we close our eyes and enter a relaxed conscious state. To bring the robot back down, we open our eyes and enter an excited state. We created a very simple demonstration board to show people the robot moving along the different quadrants, which can be seen in the video link below.

Our desired trajectories were very similar to that of Task 1. We define four positions and moved at a speed proportional to

the accelerometer magnitude in left/right direction and we used hysteresis control (as we did with Task 1) and used position control to dynamically change x_d [2], all using position control to minimize drift due to the slightly mis-calibrated gravity compensation that would otherwise appear in velocity control (see Challenges). The trajectories were:

$$x_d[0] = x_{init}[0] \tag{6}$$

$$x_d[1] = x_{init}[1] + v_\alpha \cdot sgn[\alpha > \alpha_{thresh}](t - t_{toggle,1}) + x_{d,toggle,1} \tag{7}$$

$$x_d[2] = x_{init}[2] + a \cdot (t - t_{toggle,2}) + x_{d,toggle,2} \tag{8}$$

where ttoggle,i represents the last toggling time for control on dimension i and a represents a factor proportional to the magnitude of the gyroscope acceleration value (reported by the board in units of mm/s2). We specified limits for the robot based on positions on our demonstration board.

III. Results and Discussion

Figure 5 is the plot for the Fast Fourier Transform magnitudes for Alpha waves, which were the most reliable way of controlling the robot. It reads as a function of frequency for eyes closed (red) and eyes open (blue), with corresponding standard deviations. The peak around frequency bin 10 is clearly visible. For the demonstration, alpha waves are used to change the robot's direction going from right to left, and clockwise to counter clockwise. As described in Task 2, Alpha waves and the OpenBCI's accelerometer are used to move the robot across four different quadrants.

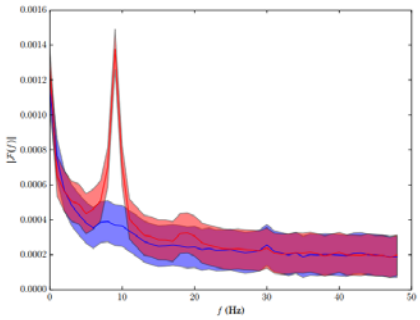
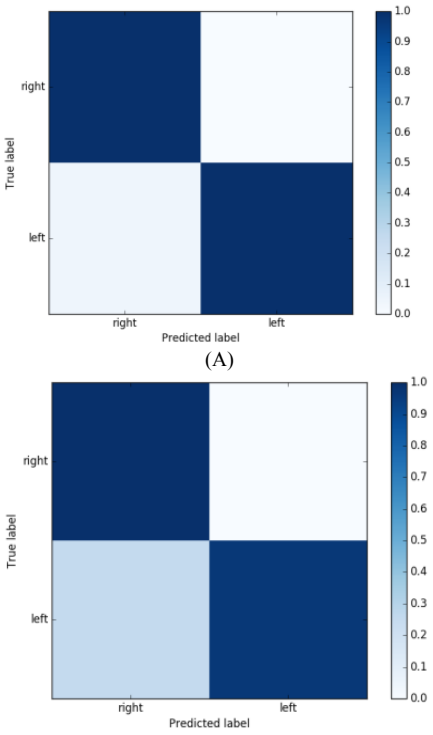


FIGURE V. PLOT OF AVERAGE FFT MAGNITUDES

The confusion plots for the right vs. left classifiers are shown in Figure 6. A linear SVM was used, with a penalty parameter of 0.1 and a hard limit of training iterations of 5000. For the squeeze and tap classifiers, the training error was both 0, and the test error (which was 20% of the original training data) was 2.1% and 10.4%, respectively. As mentioned above, this particular classifier was not generalizable for general right and left classification purposes because more data and trials were necessary to effectively apply results to a real time task.



(A)

(B)

FIGURE VI. CONFUSION MATRIX FOR THE RIGHT VS. LEFT CLASSIFIER: (A) SQUEEZE CLASSIFIER; (B) TAB CLASSIFIER

IV. Conclusion

In this study we have successfully collected and analyzed the EEG signals obtained from a test subject to control a WAM robot arm. In particular, we use signal processing and support vector machines to classify brain waves from a test subject and feed the result into the control for the robot arm. In this way, we demonstrate a method to perform basic tasks using brain-controlled robots and proof-of-concept for future applications of complex human-robot interaction.

References

[1] S.N. Abdulkader, A. Atia, M.S.M. Mostafa, "Brain computer interfacing: Applications and challenges," Egyptian Informatics J., vol. 16, pp. 213–230, 2015.

[2] Q. She, Y. Ma, M. Meng, Z. Luo, "Multiclass Posterior Probability Twin SVM for Motor Imagery EEG Classification," Comput. Intell. Neurosci., vol. 2015, pp. 95, 2015.

[3] K. Kirschfeld, "The physical basis of alpha waves in the electroencephalogram and the origin of the "Berger effect",". Biological Cybernetics, vol. 92, pp. 177, 2005.

[4] J. Malmivuo, R. Plonsey, Bioelectromagnetism, UK: Oxford University Press, 1995.

[5] E. Başar. Structures, Brain Waves, and Their Functions. Brain Function and Oscillations.GE: Springer Berlin Heidelberg, 1999.

[6] F. Pedregosa, et al., "Scikit-learn: Machine Learning in Python," J. Mach. Learn. Res, vol. 12, pp. 2825–2830, 2011.