CS225A Final Project Report

Mind Over WAM

Sunil Pai, Shirin Sadri, Michel Schoemaker, Jiheng Zhao

June 29, 2016

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 project, we present a method for controlling a 4-dof RRRR WAM robotic arm with alpha brain waves of a test subject obtained via electroencephalography (EEG). We use the OpenBCI system electrodes and board to detect alpha waves and transmit them to digital signal. We then implement a support vector machine to classify alpha waves for robot control. A robust serial communication interface was developed to convert OpenBCI data into robot commands. An accelerometer embedded in the OpenBCI board was used to implement left-right motion of the robot. To assess the performance of our system, we successfully demonstrate two primary tasks: (1) alpha wave robot control and (2) alpha wave and accelerometer robot control. The methods developed in our project 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.

1 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.

Our project 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.

2 Implementation

2.1 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 8-13Hz range [5]. 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.

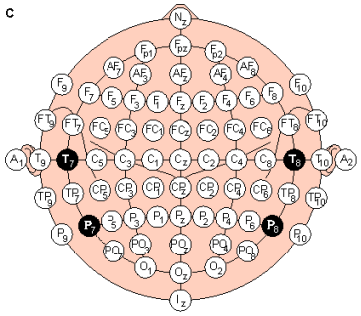

Figure 1: International 10-20 System of Electrode Placement for EEG [6]

2.2 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

Table 1: EEG Lead Placement for OpenBCI System

Central Lobe	Parietal Lobe	Occipital Lobe	Ground and Reference
C_z	P_z	O_1	A_1
C_3	P_3	O_2	A_2
C_4	P_4		

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:

1. an inner hole to fix the pen direction while moving
2. 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.

2.3 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 [3]. 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, as shown in Figure 5 in the Results section. 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 $s_f = b * s$, where s_f 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` [4] library. The classifier results are shown in the Results section.

2.4 Overall Program Structure

We run three threads simultaneously in order to implement the robot control: the serial thread (functions inside `OpenBCIBoard` 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.

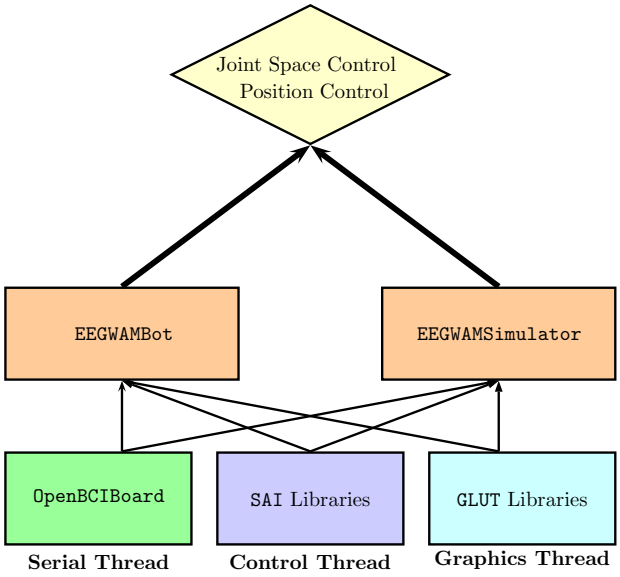

Figure 2: The serial thread, control thread and graphics thread are all necessary to control the robot in real time using the EEG device. Not included in this diagram is the plotting component, which is shown simultaneously using Cython.

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 our project.

2.5 Serial Communication

Because we use the OpenBCI as our central EEG system, we write a serial communication interface in `C++`. To do this, we convert the already-provided Python interface into a `C++` interface to parse data packets streaming from the EEG system into our data processing algorithm for analysis. Because 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. Our implementation of this is contained in the class `OpenBCIBoard`. In order to actually transfer the data analysis onto a robot action, we allow a callback function as an input into our streaming function.

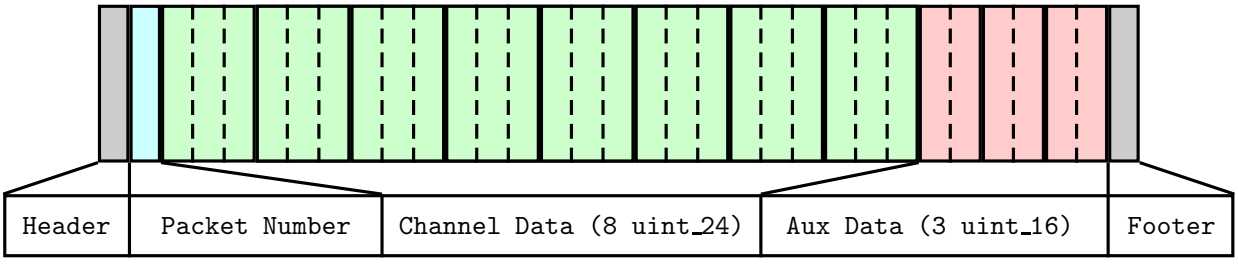

Figure 3: The structure of the data packet, where each block represents a byte of data coming in from the serial port. The auxiliary data represents the accelerometer data which we use for Task 2.

The callback function accepts as input a buffer sequence (which stores all the data to be analyzed in the previous few seconds) and we update the buffer sequence every second to issue the next command to the robot. Since the callback function is in a `while(true)` loop after each group of 250 samples is collected (the sample rate of the EEG device is 250 Hz), we run our analysis every second on the buffer, which we define to be 1250 samples, or the past 5 seconds. This analysis includes the filtering and classification tasks discussed in the previous section. 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 that is also accessed by the robot control function.

2.6 Robot Control

We use the WAM robot, which is set up as a 4-dof RRRR-bot without orientation control, as shown in Figure 4.

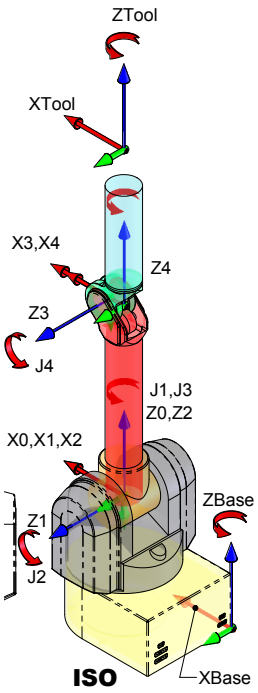

Figure 4: In the WAM, there is a base revolute joint (along $Z_{\text{Base}}, Z_0, Z_2$) which exerts little work and thus requires minimal torque to rotate, the revolute "heavy-lifter" (along Z_1) joint that is used to lift the arm for various tasks, another revolute joint (along $Z_{\text{Base}}, Z_0, Z_2$) 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.

2.6.1 Task 1: Alpha Wave Control

Moving onto our first robot control task, we started out with a few simple tasks in simulation for debugging using `EEGWAMSimulator`. 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 (where $\Lambda = (J_v A^{-1} J_v^\top)^{-1}$).

$$\Gamma' = J_v^\top \left(\Lambda \left(k_p(x_d - x) - k_v \dot{x}\right)\right) - g$$

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 (since we had three degrees of freedom already from our other three joints). 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:

$$\begin{aligned}\Gamma &= \Gamma' + \beta_{\text{ns}}(\Gamma_{\text{ns}} + \Gamma_0) \\ &= \Gamma' + \beta_{\text{ns}}(\Gamma_{\text{ns}} + [k_p q_0 + k_v \dot{q}_0, 0, 0, 0])\end{aligned}$$

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 : $\beta_{\text{ns}} = 3.0, k_p = 200, k_v = 30$. 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:

$$\begin{aligned}x_d[0] &= x_{\text{init}}[0] \\ x_d[1] &= x_{\text{init}}[1] + 0.15 \cdot \sin(2\pi f \text{sgn}[\alpha > \alpha_{\text{thresh}}](t - t_{\text{toggle}})) + x_{d,\text{toggle}}[1] \\ x_d[2] &= x_{\text{init}}[2] + 0.15 \cdot \cos(2\pi f \text{sgn}[\alpha > \alpha_{\text{thresh}}](t - t_{\text{toggle}})) + x_{d,\text{toggle}}[2]\end{aligned}$$

where f is the frequency, α is the magnitude of the 10 Hz frequency bin in the EEG signal, $x_{d,\text{toggle}}$ represents the last position the robot was at before toggling, t_{toggle} is either 0 or the last time the robot toggled, and $\text{sgn}[\alpha > \alpha_{\text{thresh}}]$ is +1 when $\alpha > \alpha_{\text{thresh}}$ and -1 otherwise.

For α_{thresh}, we use hysteresis control by defining a max-threshold of $7 \times 10^{-4}\mu$V and a min-threshold of $5 \times 10^{-4}\mu$V when alpha waves are not detected and detected respectively. This allows us to smooth the task over time. Bringing these limits closer together allows us to get finer control but may also allow noise of alpha wave detection to start dominating.

2.6.2 Task 2: Alpha Wave + Accelerometer Control

For our next 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:

$$
\begin{aligned}
x_d[0] &= x_{\text{init}}[0] \\
x_d[1] &= x_{\text{init}}[1] + v_\alpha \cdot \text{sgn}[\alpha > \alpha_{\text{thresh}}](t - t_{\text{toggle},1}) + x_{d,\text{toggle},1} \\
x_d[2] &= x_{\text{init}}[2] + a \cdot (t - t_{\text{toggle},2}) + x_{d,\text{toggle},2}
\end{aligned}
$$

where $t_{\text{toggle},i}$ 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/s^2). We specified limits for the robot based on positions on our demonstration board.

3 Challenges

3.1 EEG Protocol

Our initial system for detecting brain waves used an Emotiv headset, which is essentially a brain wearable for consumer EEG measurements. The Emotiv system is designed for research and advanced BCI applications and has high spatial and temporal resolution. While we were able to detect alpha waves with this system, we ran into several obstacles that eventually prevented us from using this system in our final implementation. We had trouble using the provided Emotiv libraries to work properly with our system and also ran into coding errors such as a floating point exception. We reached out to employees of Emotiv through GitHub to understand how to overcome these issues. However, it was difficult for them to understand the problems we were encountering with their software specifically for our project application. In addition, because their software is proprietary, Emotiv could not release their implementation to us so that we could further understand the issues. We also considered using a clinical-grade EEG cap to gather our data. While this would have provided significant improvements in signal quality, it would have been necessary to implement software to interpret the signals by hand. However, we collectively decided that we wanted to focus on other aspects of the project instead, and decided that this implementation would be outside of the scope of our project.

We decided to switch to the OpenBCI system, an open source biosensing microcontroller that can be used to gather EEG data. This enables us to have much more control over the system because the accompanying software was open-source. One drawback of using this system as compared with Emotiv is the protocol for lead attachment. While Emotiv consists of an easy-to-use headset that can be configured on a test subject in a matter of seconds, the OpenBCI system relies on manual attachment of electrodes onto the subject's head using attachment glue. Despite some sacrifices in signal quality and system convenience, the OpenBCI system proves to be fairly robust for our application.

3.2 End Effector Design

One of the big challenges for the end-effector design is to get the correct dimensions to connect the designed products to the robot arm. Moreover, for more advanced end-effector designs to fulfill more functions, how to build-up good electrical connections and implementation to robot control system is really critical and challenging. Due to time issues, we only consider the simple end-effector design without any electrical components. The fabrication of end-effector is also challenging. 3D printing is a good way for quick prototype making but is not suitable for the designs require a large amount of materials and complex structures.

3.3 Data Analysis

When working with alpha waves, one of the biggest challenges is to determine the 'frequency bins' over which to perform the Fast Fourier Transform. As explained in Section 2.3, alpha waves occur in the 9 to 14 cycle frequency range, but the range of frequency 'bins' cannot be too wide (e.g. 9 to 14) because it obfuscates the calculations for alpha wave detection. Based on the graph outlined in Section 2.3, we decided to only consider frequency ranges from 9 up to but not including 10. Another challenge with alpha waves is determining the time length over which the data should be collected before analyzing it. A small time frame (for example, half a second) allows for quicker robot motion control, but since alpha waves do not occur at a constant rate, it induces instability in the motion as well. A larger time frame allows for a more stable motion control using alpha waves, but only up to a point - too long of a time frame makes it harder to find a clear limit between alpha waves and no alpha waves.

For the right vs left classifiers, there are various challenges that did not permit us to perform right or left robot motion using only brain signals. The data, even after processing it with a Fast Fourier Transform quickly overfit (even at a low number of iterations). This was due to our main source of OpenBCI data coming from only one team member, and only being taken for two continuous minutes each. A larger sample of data from multiple sources and taken at multiple occasions would have helped mitigate the overfitting. Furthermore, many unintended facial and body motions cause spikes in the data, among other unexpected behaviors that obfuscate a clear left vs right divide. Electrode placements are also hard to ensure that they stay constant, making the raw data very noisy.

3.4 Serial Communication

The main challenge in the serial communication was ensuring that the endian order was correct while parsing the incoming bits in the data packet. For a while, since the endian order was wrong during data collection, our results were noisy and impossible to convert into a viable task, but once the byte order was reversed, data collection and analysis was relatively fluid.

Another challenge was the activation of the notch filter implementation in the board. For a while, we were obtaining a strong 60Hz signal because of outlet interference during serial communication, but we were able to stop this 60Hz signal by enabling filters internal to the board (namely, the 60Hz signal).

3.5 Robot Control

One major challenge in implementing our control law was in implementing a path trajectory. To define a path, we define a list of positions so that the robot can cycle through these positions. Next, we define a threshold for the robot's tracking error so that the robot knows it has gotten close enough to the next position and can move to the next position in the path. Unfortunately, even after manipulating the gains, the robot would not reach our desired positions within reasonable thresholds because the internal gravity compensation of the robot was different than what we had in simulation. This meant that we had to interpolate positions in our path in order to accomplish the task and remove the threshold requirement.

Another implementation challenge was to control multiple directions with the robotic arm for Task 2. Because alpha waves are spuriously detected when the head or muscles on the face move, we had to specify in our implementation that `ALPHA_WAVE_VALUE` be set to zero when the magnitude of `ACCEL_VALUE` was above some threshold (we specified about $0.2g$). We had a bit of latency in our control due to the WAM's suboptimal gravity compensation (or, in the case of the demo, the alpha waves took a while to show up), but the robot eventually succumbs to our commands, which is a great sign.

Finally, for all tasks, we implemented a `SIGINT` handler that allows us to gracefully exit a program using `Command-C` by setting all torques to zero before quitting.

4 Results/Conclusions

Below is the plot for the Fast Fourier Transform magnitudes for Alpha waves, which were the most reliable way of controlling the robot. The full demonstration can be found at the following link:

`https://m.youtube.com/watch?v=01yOD-EtjYw`

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.

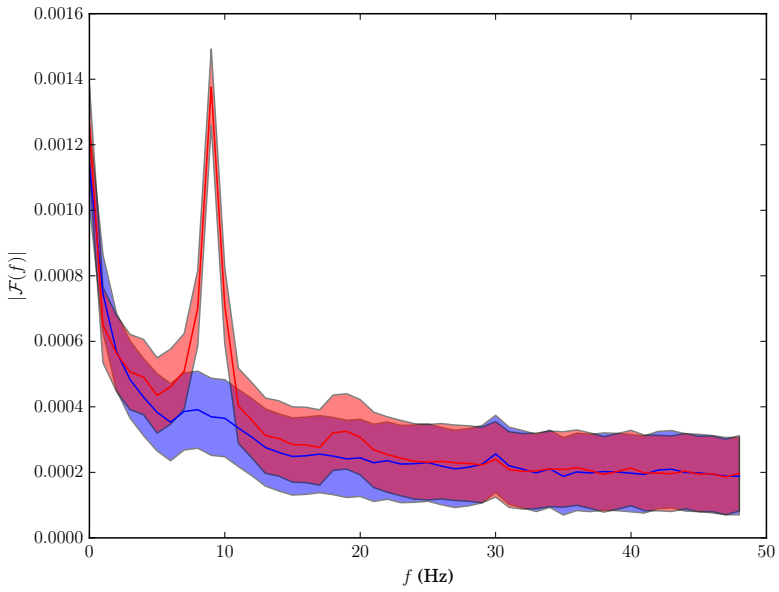

Figure 5: Plot of average FFT readings as a function of frequency for eyes closed (red) and eyes open (blue), with corresponding standard deviations. The peak around frequency bin 10 is clearly visible.

The confusion plots for the right vs. left classifiers are below. A linear SVM was used, with a penalty parameter of 0.1 and a hard limit of training iterations of 5000.

As mentioned in the challenges sections, this particular classifier was not particularly generalizable for general right, left classification purposes because more data and trials were necessary to effectively apply results to a real time task.

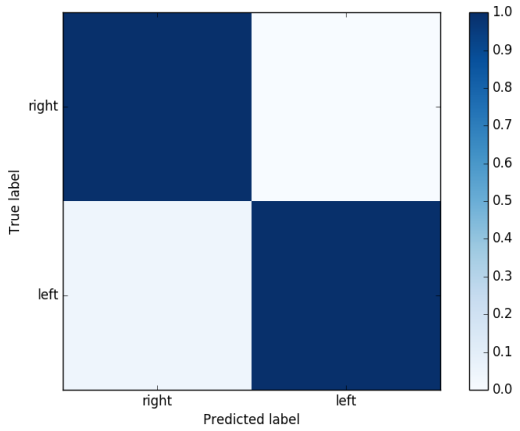

Figure 6: Confusion matrix for the right squeeze vs. left squeeze classifier. The training error was 0%, and the test error (which was 20% of the original training data) was 2.1%

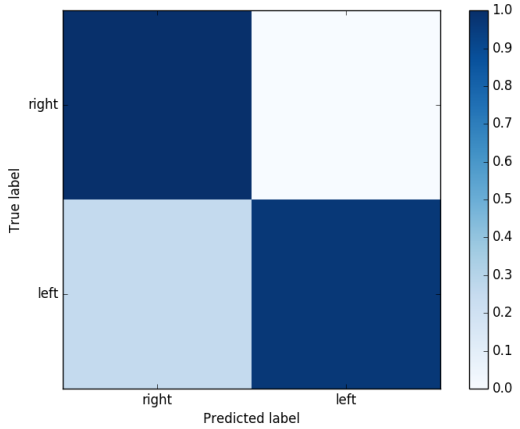

Figure 7: Confusion matrix for the right tap vs. left tap classifier. The training error was 0%, and the test error (which was 20% of the original training data) was 10.4%

5 Feedback

The Experimental Robotics class was very helpful because it helps us to put our theoretical robotics knowledge into real applications. Working with people from different backgrounds was also a good experience, enabling us to learn different thinking strategies and problem-solving techniques. The professor and TAs were helpful and knowledgeable with the course material and were a great resource when we ran into issues throughout the project. For future quarters, it would be helpful if there were a database of previous projects for students to look at before making decisions on their own projects. In addition, it is important to emphasize the skills required to make the class interesting for everyone (i.e. programming, machine learning, computer vision, etc). The first few homeworks were not very indicative of the level of knowledge necessary to create a meaningful project. Nonetheless, we would definitely recommend this class to our friends– it was a very fun and useful class.

6 References

[1] Abdulkader, S.N. et al. Brain computer interfacing: Applications and challenges. Egyptian Informatics Journal

[2] She, Q. et al. Multiclass Posterior Probability Twin SVM for Motor Imagery EEG Classification. Computational Intelligence and Neuroscience (2015)

[3] http://www.scientificamerican.com/article/what-is-the-function-of-t-1997-12-22/

[4] http://scikit-learn.org/stable/modules/svm.html

[5] Kirschfeld, K. "The physical bases of alpha waves in the electroencephalogram and the origin of the 'Berger effect'". *Biological Cybernetics* (2005)

[6] http://www.bem.fi/book/13/13.htm