1-1-2013

Real-time Control of a Robot Arm Using an Inexpensive System for Electroencephalography Aided by Artificial Intelligence

James O'Connor
Connecticut College, joconno2@conncoll.edu

Real-time Control of a Robot Arm Using an Inexpensive System for Electroencephalography Aided by Artificial Intelligence

Jim O'Connor
Computer Science

Table of Contents

Introduction

The purpose of this work is to explore the methodology and uses of an inexpensive and non-invasive brain-computer interface for the control of robotic systems, and specifically a three degree of freedom robotic arm. Traditionally, brain computer interfaces have been explored in the control of various electrical and mechanical devices, but the brain-computer interfaces that have been studied in the past are very rarely non-invasive. The only non-invasive option for interfacing a brain with a computer is the electroencephalogram, or EEG, which is usually bulky, and expensive. Additionally EEGs are difficult to find outside of research institutions and require a trained technician to operate. For these reasons, this work explores the use of the Emotiv EPOC Electroencephalogram, which is a non-invasive brain-computer interface that is also inexpensive and commercially available. Shortcomings in the resolution of this system are made up for through a novel system of Artificial Intelligence, employing Linear Support Vector Machines to classify the data from the Emotiv EPOC into a set of outputs for a three degree of freedom robotic arm.

Brain-computer interfaces have been explored for years with the intent of using human thoughts to control mechanical systems. When one is not considering the accessibility of the electroencephalographic system, many technological feats in this area are already possible.

Electroencephalographic classification for the control of computers has been studied for control of software on computers as far back as 1991, by Wolpaw, McFarland, Neat, and Forneris [1]. Wolpaw et al. successfully created a system that allowed a user to control a cursor using electroencephalographic signals. Wolpaw would go on to explore the area in a series of papers describing the state of the art throughout the early 2000s [2],[3]. This work was built upon by Iáñez et al. [4], who used a large, high resolution electroencephalogram to control a robotic arm. These studies all worked towards showing the power of electroencephalography, but unfortunately did not

address the issues of prohibitive cost and difficulty of use for the end-user. The Emotiv EPOC is a device that has been explored for exactly this reason.

The Emotiv EPOC has been used in a variety of applications, due to its accessibility for consumers and researchers. The device was explored by Lievesley, Wozencroft, and Ewins, out of the UK, who found it to be useful for the control of computers by patients with no voluntary muscle control [5]. These patients were able to access computer based assistive technology and navigate a virtual space using two discrete thought patterns, all with a greater than random chance of success. Poor et al. from Bowling Green State University, in Ohio, also found the Emotiv EPOC to be an effective method for computer control, having the subjects of their experiments perform a 3D rotation task in virtual space [6].

The use of the Emotiv EPOC has also been explored in the area of controlling robotic systems through non-invasive BCI. In 2010, Wang and Fuhlbrigge published a review of the state of the art in brain-computer interface technology as applied to industrial robotics. They found the field to be growing quickly with the introduction of

Figure 1. The Nao humanoid robot, by Aldebaran Robotics. Kadam and Sheng [8] used this robot for electroencephalographic teleoperation.

systems such as the Emotiv EPOC, and exhibited high hopes for the future [7]. Also in 2010, Kadam and Sheng used the Emotiv EPOC to teleoperate a remote humanoid robot, showing yet another facet of robotics where non-invasive BCI can be successfully applied. Kadam and Sheng's system allowed for the remote control of an Aldebaran

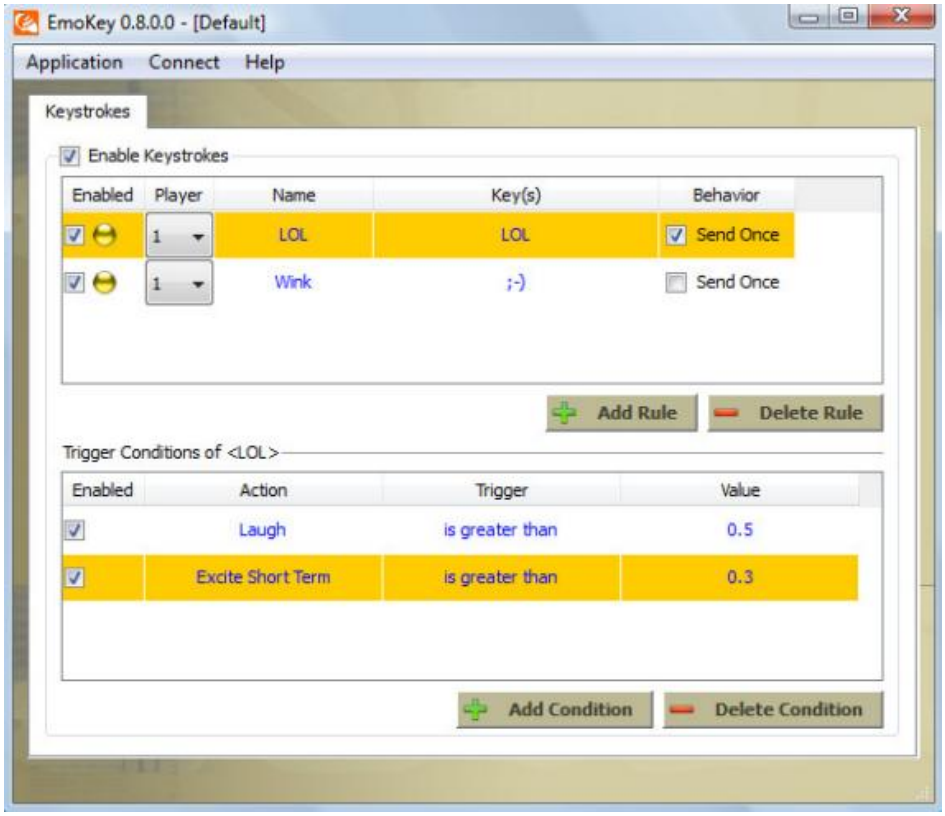

Figure 2. A screenshot of the Emotiv EmoKey software, which Vourvopolous and Liarokapis used to control a Lego NXT robot.

Robotics Nao humanoid robot (Figure 1), which the user was able to direct with electroencephalographic commands [8] to move forward, turn left, and turn right. Kadam and Sheng found that their system was effective to navigate a small course. In 2011 Vourvopoulos and Liarokapis published yet another example of teleoperation in mobile robotics through the use of the Emotiv EPOC to control a Lego NXT robot [9]. In order to control the Lego NXT robot, Vourvopolous and Liarokapis used the Emotiv Emokey software (Figure 2) to send key commands to a java program. The Emokey software is provided by Emotiv, and allows a user to select different keystrokes to virtually actuate based on electroencephalographic data, with up to four outputs being trained [10]. In

addition to the wide breadth of study that has been undergone in the area of non-invasive brain-computer interfaces for the remote con trol of mobile robots, Ranky and Adamovich in 2010 explored the use of the Emotiv EPOC for the control of robotic arms, and determined that the EPOC could be an effective system for control through electroencephalography [11].

In the work reported in this paper, we are looking to improve the effectiveness of each of these previous methods of control by increasing the number of possible outputs gathered from the electroencephalographic data. While the software provided with the Emotiv EPOC allows a user to train a maximum of four outputs, the consistency and effectiveness of those outputs are questionable. While prior work with this device has shown that effective control with as many as three and sometimes four outputs can be consistent [5,6,8,9], none of these systems use more than four outputs. In this work, we look to classify the electroencephalographic data of the user to six outputs, greatly increasing the complexity of any mechanisms being controlled by the user.

When one is working to classify a set of data as complex as that recorded by an electroencephalogram, machine learning holds many of the most effective solutions. In this work, we use support vector machines (SVMs), which are the standard method of classifying electroencephalographic data. Support vector machines have been used for a variety of complex classification tasks, ranging from the classification of EEG data (Leuthardt et al. 2004) [12], to the classification of vision-based obstacle detection (Ubbens 2009) [13]. We show that the use of SVMs can increase the effective degrees of freedom that can be controlled using an Emotiv EEG system.

Background

EEG Technology

The Electroencephalogram (EEG) is a device used to measure a patient's neuronal activity through the measurement of electrical currents along the surface of the scalp (Figure 3). The EEG was developed in the early 20th century by the Neurologist Hans Berger, when Berger was trying to find connections between extra sensory perception in humans and electrical activity in the brain [14]. Berger published the first use of his electroencephalogram on the human brain in 1929, and his discoveries were confirmed and furthered by British scientists Edgar Douglas Adrian and B. H. C. Matthews in 1934 [14]. The Modern EEG is not far different from that developed by these foundational scientists.

Figure 3. A man at Brandeis University wearing a modern EEG. He is wearing a 128-channel high spatial density, high-impedance system, which gives more accurate electroencephalographic data, but is even more bulky then other similar systems.

Modern EEGs function by measuring voltage fluctuations along the surface of the scalp caused by ionic current flows within the neurons of the brain. A neuron is a cell found in the brain that transmits nerve impulses from the body. Neurons can receive different kinds of signals from the body, either through a chemical change, such as a response from olfactory receptors, or from a physical change such as a touch receptor in the skin or a light receptor in the retina. The signals that the neuron receives cause Sodium and Potassium ions to move across the neuron's plasma membrane, which in turn causes an electrical current flow. This current flow can be used to evaluate the activity of the brain, and is therefore read directly by the EEG. The current flow is measured by electrodes on the scalp, which operate under a standard sensitivity of 7 μvolts/mm. The readings gathered by these electrodes are then reported by the EEG as a measure of frequency, along a number of different consistent electrode placements. Decisions for Electrode placement are based on a number of standards, but the most common system is the International 10-20 system, developed in 1958 by a committee of the International Federation of Societies for Electroencephalography and Clinical Neurophysiology, and published by Jasper [15]. The International 10-20 system uses anatomic landmarks on the scalp to give a consistency of electrode placement among varying individuals.

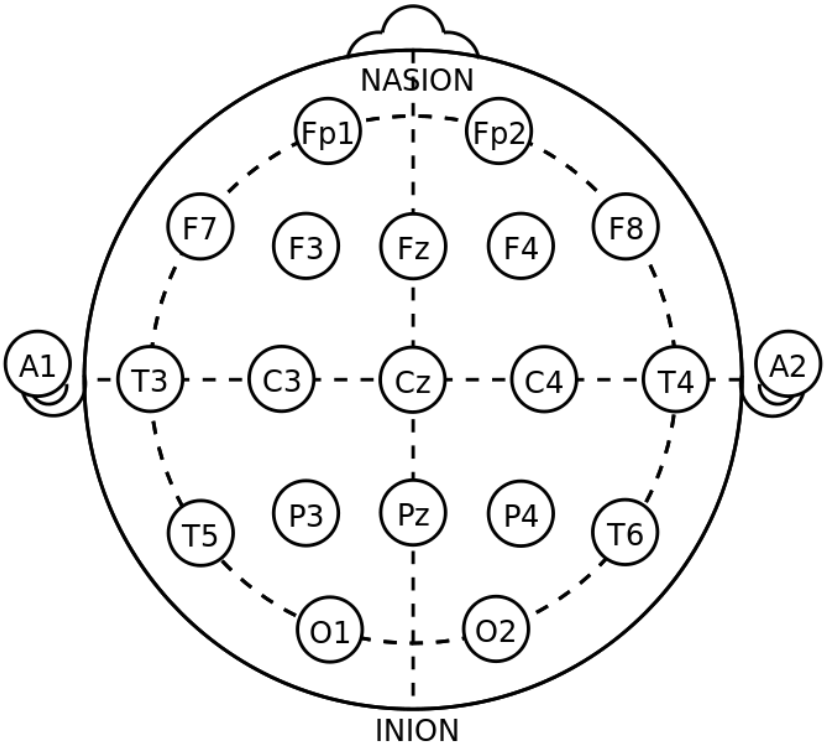

Figure 4. The standard electrode placement of an EEG, recommended by the American EEG Society for use in the International 10-20 system.

The anatomical landmarks used by this system (shown in Figure 4) are the nasion and the ision. The nasion is the depressed area of the skull just behind the eyes, and the ision is the bump on the back of a patients head, which is the lowest part of the skull. This system has shown to be effective over time, but requires a prohibitive number of electrodes.

Brain Computer Interfaces

A Brain-Computer Interface, or BCI, is a device that allows a user to interact with a computer using electrical signals from their brain. Most BCIs can be divided into three categories; invasive, partially-invasive, and non-invasive. Invasive BCIs are installed directly in the grey matter of the brain. These devices can receive the most clear and effective signals from the brain, and have been used effectively in both animal and

human tests for various applications, including the treatment of congenital blindness. Unfortunately, invasive BCIs are prone to a buildup of scar tissue around the implant, due to the body recognizing the implant as a foreign body. Invasive BCIs can also be dangerous to implant, due to the surgery involved, as well as extremely costly. For these reasons, invasive BCIs are becoming less common as alternative approaches are being found.

Partially invasive BCIs are another method of interacting with a computer through the activity of the human brain. A partially invasive BCI involves installing a device between the brain and the inside of the skull, which can then read brain waves without the interference of the human skull. These devices are nearly as effective as an invasive BCI, but involve significantly less complicated, dangerous, and expensive brain surgery to install. These devices are also not affected by scar tissue buildup, like the invasive BCIs. However, a partially invasive BCI still must be installed in the skull through a surgical procedure, which makes the device expensive and inaccessible to the majority of patients.

The third and most accessible method is the non-invasive BCI. A non-invasive BCI sits on the surface of the scalp, and can be removed or worn at the patient's discretion. There is no surgery involved, and most non-invasive BCIs can still interact with computers in a completely effective manner. Research has been done on using non-invasive BCIs, such as EEGs, to control many different computers, machines, and user interfaces, as well as to gather data about a patients mental health. Unfortunately the majority of BCIs are very expensive, awkward machines that are inaccessible to the average person. However, there are some smaller, lower-resolution consumer grade EEGs that can provide the safety and effectiveness of a non-invasive BCI with a low cost.

The Emotiv EPOC EEG

The Emotiv EPOC EEG is a revolutionary new electroencephalogram that is already being explored for BCI research. The EPOC uses fourteen sensors and two reference points to track fourteen different channels of electroencephalographic activity. The Emotiv Epoc gathers data from the standard channels named AF3, AF4, F3, F4, F7, F8, FC5, FC6, P3 (CMS), P4 (DRL), P7, P8, T7, T8, O1, and O2 (Figure 5). It uses sequential sampling, has a 0.2 - 45Hz bandwidth and 16 bit resolution. This leads to a lower resolution dataset than a standard EEG using the International 10-20 system, but the EPOC has many benefits to make up for this shortcoming. The Emotiv EPOC is available for personal or commercial use, and is extremely inexpensive compared to a standard EEG.

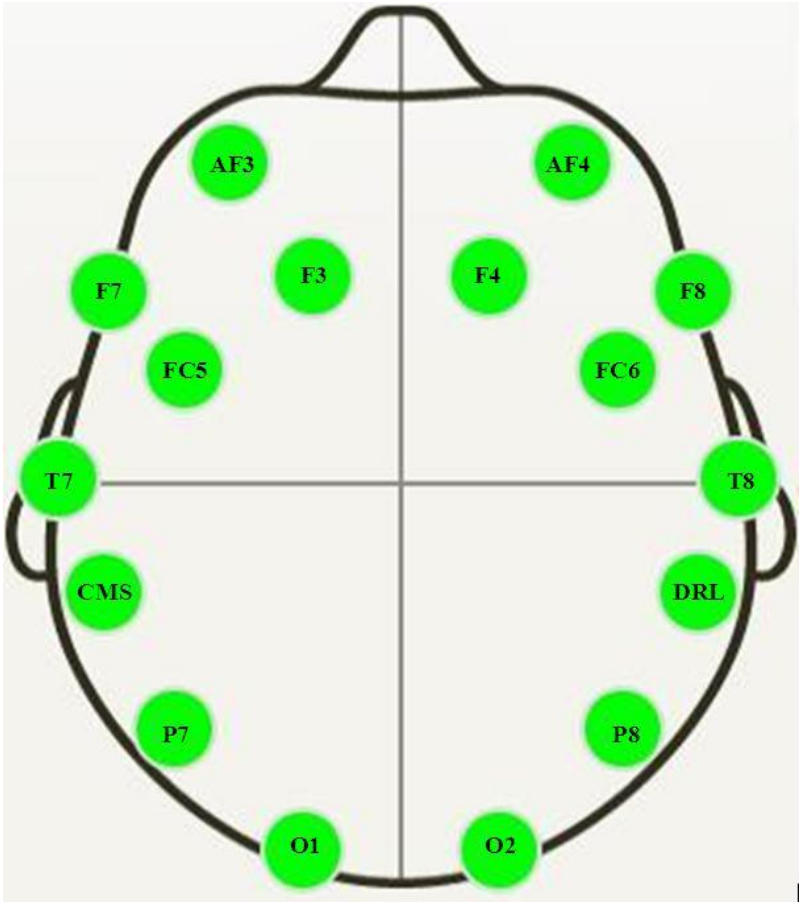

Figure 5. The location of the fourteen channels read by the Emotiv EPOC.

It can be worn and used without a trained technician or operator, and can feed data into a standard computer. The headset is also wireless, allowing a much greater range of experiments to be performed compared to that of a standard EEG. Effectively, if the fairly significant problem of a lower resolution device was to be overcome in the package and price of the Emotiv EPOC, great strides could be taken in the accessibility and availability of BCI technology to the average consumer. This technology could contribute not only to the improvement of handicap devices and scientific inquiry, but also to entertainment on a broader scale.

Support Vector Machines

In order to use the data from the Emotiv EPOC EEG, which is lower resolution and therefore provides data with less characteristics from which to draw conclusions, the data has to be classified into discrete outputs. Classification is a common task in Machine Learning, and has been studied in depth. Some of the most common tools used to classify data are Support Vector Machines. A Support Vector Machine, or SVM, is a supervised learning model that takes in a set of data, and predicts , for any given input, which of two classifications the input belongs to Cortes and Vapnik, 1995) [16]. In order to classify these inputs, the SVM must first find an optimal hyperplane that separates a small amount of training data (the support vectors). If the training vectors are separated without errors by an optimal hyperplane, we can assign a value to the expectation of the probability of committing an error in a test example. This value is bounded by the ratio of the expectation value of the number of support vectors and the number of training vectors, and gives an idea of the effectiveness of the SVM.

$$E[\Pr(\text{error})] \leq \frac{E[\text{number of support vectors}]}{\text{number of training vectors}}$$

Due to this ratio, if the optimal hyperplane can be constructed from a small number of support vectors relative to the training set, then the generalization ability of the SVM will be very good, even in a high-dimensional space. This ability is part of what makes SVMs such an effective method of classification, even in complex real-world problems.

In order to construct an optimal hyperplane to separate a set of training data, we use an algorithm devised by Vladimir Vapnik in his 1982 work on the subject. According to the introductory paper on Support Vector Machines, by Cortes and Vapnik in 1995 [16], the set of labeled training patterns

$$(y_1, x_1), \dots, (y_l, x_l), \qquad y_i \in \{-1, 1\}$$

is said to be linearly separable if there exists a vector w and a scalar b such that the inequalities

$$w \cdot x_i + b \geq 1 \text{ if } y_i = 1,$$
$$w \cdot x_i + b \leq -1 \text{ if } y_i = -1,$$

are valid for all elements of the training set described by the set of labeled training patterns above. Cortes and Vapnik rewrite the inequalities in the form

$$y_i(w \cdot x + b) \geq 1, \qquad i = 1, \dots, l.$$

The optimal hyperplane described as

$$w_0 \cdot x + b_0 = 0$$

is the unique hyperplane that determines the direction $\frac{w}{|w|}$ where the distance between the projections of the training vectors of two different classes is maximized, and w and

b are the arguments that maximize that distance. This maximized distance value refers to the optimality of the constructed hyperplane. The distance $p(w, b)$ is given by the equation

$$p(w, b) = \min_{\{x:y=1\}} \frac{x \cdot w}{|w|} - \max_{\{x:y=-1\}} \frac{x \cdot w}{|w|}.$$

The term Support Vector in Support Vector Machine refers to the training vectors used to classify input data to the machine. Support vectors are described as vectors x_i for which

$$y_i(w \cdot x_i + b) = 1.$$

The vector w_0 that determines the optimal hyperplane can also be written as a linear combination of training vectors:

$$w_0 = \sum_{i=1}^{l} y_i \alpha_i^0 x_i,$$

where $\alpha_i^0 \geq 0$, and α is a vector of Lagrangian multipliers with a length of i [16]. This equation holds because α is only greater than 0 for support vectors. If the training data cannot be separated without error one must work to separate the training set with the least number of errors possible. To express this idea formally, Cortes and Vapnik introduce a new set of non-negative variables $\varepsilon_i \geq 0, i = 1, \dots, l$. These variables allow us to describe the training errors as

$$\varphi(\varepsilon) = \sum_{i=1}^{l} \varepsilon_i^{\sigma}$$

for small $\sigma > 0$, subject to the constraints

$$y_i(w \cdot x_i + b) \geq 1 - \varepsilon_i, \quad i = 1, \dots, l,$$

$$\varepsilon_i \geq 0, \qquad\qquad i = 1, \ldots, l.$$

Minimizing the summation above allows one to find the minimal subset of training errors. This subset can then be omitted from the training data to allow linear separation of the remaining training data. This idea can be expressed formally as minimizing the function

$$\frac{1}{2}w^2 + CF(\sum_{i=1}^{l} \varepsilon_i^{\sigma})$$

subject to the aforementioned constraints, where $F(u)$ is a monotonic convex function and C is a constant. Using the described Support Vector Machines, one can separate a large set of data, whether or not it is linear classifiable, into two distinct groups. This classifying ability makes them useful for a variety of tasks, including classifying EEG data into discrete outputs.

Methodology

Using the Emotiv EPOC

Many of the benefits of the Emotiv EPOC EEG lie in its usability. In order to extract electroencephalographic data, the EEG must be prepared briefly before each use. On a normal EEG, a saline paste or gel must be applied to each of the many electrodes on the device, to maintain proper signal between the electrode and the scalp. The EPOC is simpler in that each felt electrode needs only to be wetted with a commercially available saline solution. After the EEG is prepared, the test subject sets it on their head (Figure 6), and begins checking the calibration through the feedback from the included graphical user interface. All communication between the device and the computer occurs through a standard USB dongle, transmitting on a proprietary wireless band at 2.4Ghz.

Figure 6. The Emotiv EPOC EEG in use

Once in use, the EPOC sends the electroencephalographic signal data from the scalp to the computer. This data travels in the form of encrypted packets, which are decrypted in the Emotiv TestBench software provided by the Emotiv Research edition SDK.

Using Emotiv TestBench

When the Emotiv TestBench software picks up the signals from the EPOC, each point of data is automatically displayed on the graphical user interface. The EPOC samples the data sequentially over fourteen electroencephalographic locations: AF3, AF4, F3, F4, F7, F8, FC5, FC6, P3 (CMS), P4 (DRL), P7, P8, T7, T8, O1, and O2. Each of the locations corresponds to a portion of the international 10-20 system, and were selected for an even distribution over specific areas of the scalp. The EPOC samples the data at a rate of 128HZ, with a 14 bit effective resolution. This resolution causes the data from the headset to be much more noisy than a traditional EEG but is still viable for use after classification.

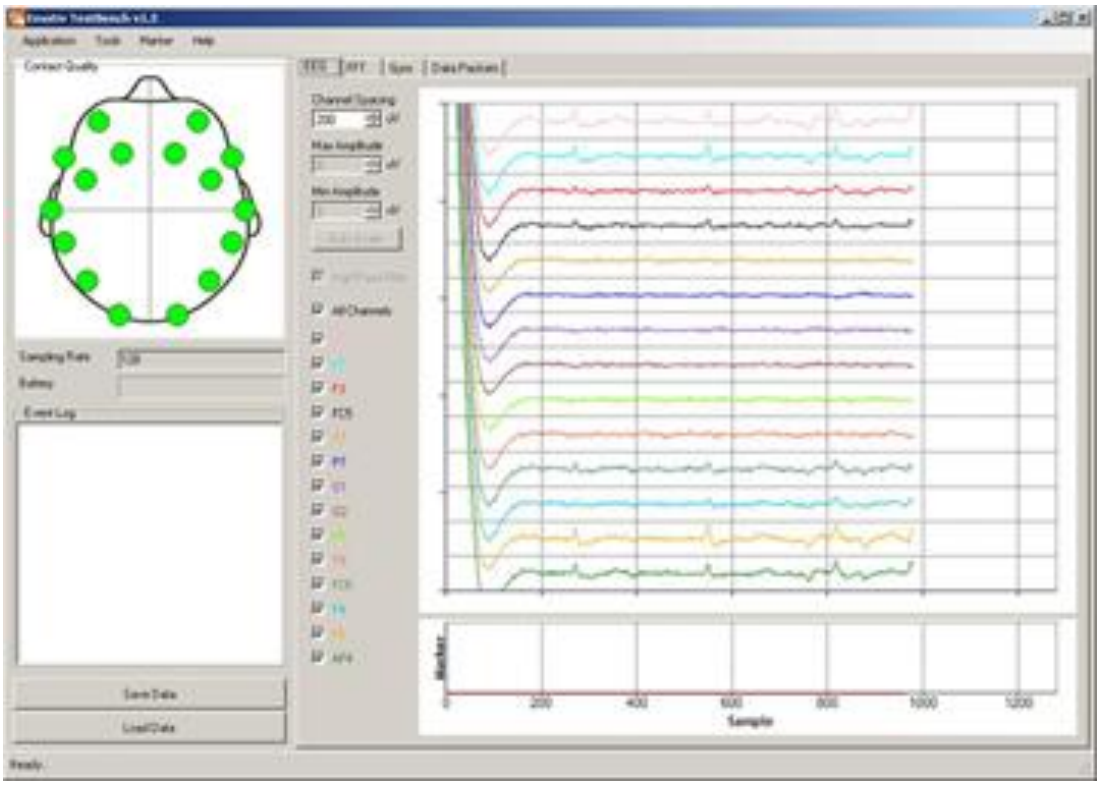

Figure 7. The Emotiv TestBench software in use. The data shown on the screen corresponds to the 14 channels being picked up by the EEG.

As shown in Figure 7, the Emotiv TestBench software displays the electroencephalographic data in real time along the fourteen monitored channels. Each of these channels will vary when the user thinks different thoughts, and will also pick up and electromyographical data, such as blinking and facial expressions. For that reason it is most effective to maintain a neutral expression both when training and testing the device. In addition to the data visualization aspect of the software, the Emotiv TestBench software also allows for the collection of data over a given period of time into the EDF physiological signal recording format. For our purposes this data is then converted to comma separated value (CSV) format (Figure 8), where it can then be read effectively into MATLAB for further processing. In order to effectively train the SVM in this experiment, a dataset was created for each of six different classes; passive, up, down, left, right, forward. Each dataset was recorded through a minute of concentration

on each movement, followed by a minute resting period. The corresponding data was then saved and converted to CSV format, where it existed as six files, each containing a header, seven thousand rows of data points, and thirty six columns of data. The first fourteen columns of data were gathered from electroencephalographic data, with the other twenty two being gathered from electromyographical signals and preprocessed data. Seven thousand rows of data represent approximately one minute of continuous electroencephalographic signals, with fourteen columns representing fourteen features.

```
title:Forward, recorded:14.04.13 20.03.06, sampling:128, subject:Jim, labels:COUNTER INTERPOLATED AF
62.00000000000000000000, 0.00000000000000000000, 4272.82040833643350000000, 4116.41015575090660000000
63.00000000000000000000, 0.00000000000000000000, 4278.46143383951860000000, 4127.17938625679560000000
64.00000000000000000000, 0.00000000000000000000, 4277.94861333923840000000, 4132.8204117598798000000
65.00000000000000000000, 0.00000000000000000000, 4275.89733133811660000000, 4138.4614372629649000000
66.00000000000000000000, 0.00000000000000000000, 4286.66656184400470000000, 4137.9486167626837000000
67.00000000000000000000, 0.00000000000000000000, 4301.53835635213730000000, 4142.5640012652075000000
68.00000000000000000000, 0.00000000000000000000, 4304.61527935381950000000, 4155.8973342724985000000
69.00000000000000000000, 0.00000000000000000000, 4297.94861285017400000000, 4151.2819497699747000000
70.00000000000000000000, 0.00000000000000000000, 4291.79476684680960000000, 4140.5127192640866000000
71.00000000000000000000, 0.00000000000000000000, 4295.38451034877200000000, 4147.6922062680123000000
72.00000000000000000000, 0.00000000000000000000, 4302.56399735269770000000, 4146.1537447671708000000
73.00000000000000000000, 0.00000000000000000000, 4298.97425385073530000000, 4133.3332322601600000000
74.00000000000000000000, 0.00000000000000000000, 4292.82040784737000000000, 4143.5896422657688000000
75.00000000000000000000, 0.00000000000000000000, 4294.87168984849180000000, 4161.5383597755836000000
76.00000000000000000000, 0.00000000000000000000, 4295.38451034877200000000, 4156.4101547727796000000
77.00000000000000000000, 0.00000000000000000000, 4294.35886934821160000000, 4151.7947702702550000000
78.00000000000000000000, 0.00000000000000000000, 4296.92297184961350000000, 4158.4614367739005000000
79.00000000000000000000, 0.00000000000000000000, 4290.76912584624820000000, 4156.4101547727796000000
80.00000000000000000000, 0.00000000000000000000, 4286.66656184400470000000, 4153.3332317710965000000
81.00000000000000000000, 0.00000000000000000000, 4295.89733084905220000000, 4157.4357957733400000000
82.00000000000000000000, 0.00000000000000000000, 4296.41015134933330000000, 4154.3588727716578000000
83.00000000000000000000, 0.00000000000000000000, 4290.76912584624820000000, 4143.5896422657688000000
84.00000000000000000000, 0.00000000000000000000, 4289.74348484568780000000, 4135.8973347615629000000
85.00000000000000000000, 0.00000000000000000000, 4293.84604884793040000000, 4133.3332322601600000000
86.00000000000000000000, 0.00000000000000000000, 4282.56399784176210000000, 4141.0255397643668000000
87.00000000000000000000, 0.00000000000000000000, 4247.69220382269300000000, 4152.3075907705361000000
88.00000000000000000000, 0.00000000000000000000, 4223.58964030951390000000, 4153.3332317710965000000
89.00000000000000000000, 0.00000000000000000000, 4229.23066581259810000000, 4156.4101547727796000000
90.00000000000000000000, 0.00000000000000000000, 4235.38451181596340000000, 4169.2306672797895000000
91.00000000000000000000, 0.00000000000000000000, 4236.41015281652380000000, 4171.2819492809112000000
92.00000000000000000000, 0.00000000000000000000, 4243.58963982045040000000, 4167.1793852786677000000
93.00000000000000000000, 0.00000000000000000000, 4242.05117831960890000000, 4170.2563082803508000000
```

Figure 8. An example of the data recorded by the EEG, in comma separated value format. Only the first four columns are shown here, because the full dataset is thirty-six columns wide. The first column of data is a sequential numbering of the samples recorded by the EEG. The second column is for spacing out the data set, while the rest of the columns consist of data collected by the EEG. Each row is another sample, recorded sequentially.

For our research only the first fourteen columns were used. The header of each data file was also removed, to leave a seven-thousand row, fourteen column matrix. This process was repeated five times for each of the six classes for a total of thirty data sets.

Linear SVM for Dataset Classification

After the thirty datasets are compiled, they are read into MATLAB and stored as matrices. Each of these matrices is made up of 7000 rows, and 14 columns. Each of the 14 columns in the matrix corresponds to a feature of the data; in this case, one of the 14 outputs from the EEG. The 7000 rows of the matrix come from the data read by the EEG, corresponding to one sample of data per row, for a total of about one minute of data per matrix. The 5 matrices for each classes are then concatenated vertically, to give 35,000 rows of data, or 35,000 samples for each of the 14 features, as shown in Figure 9. Each 35,000 value column is split into 50 subsets vertically (about 6 seconds of data per subset), giving each of the 14 columns 50 separate data sets. The data is

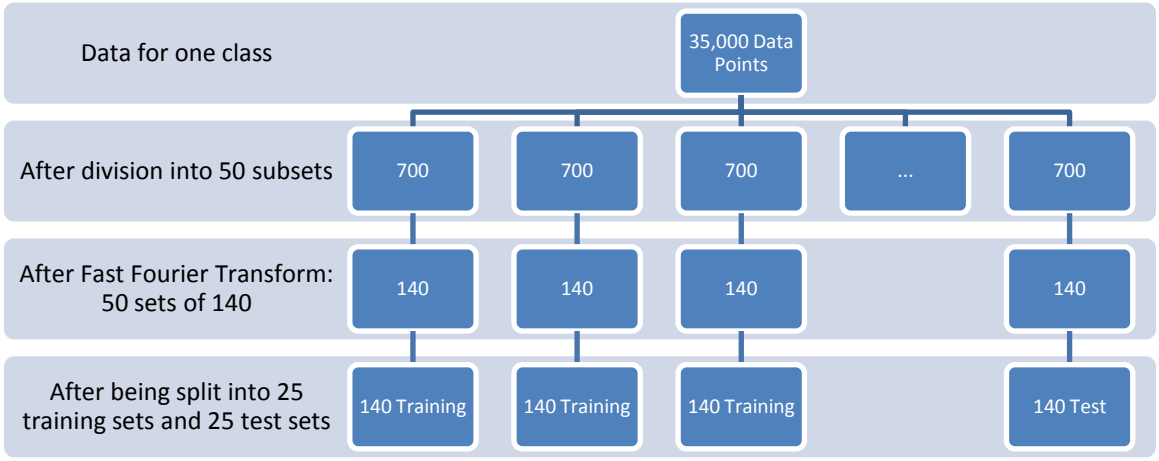

Figure 9. The data formatting prior to training of the SVM.

then formatted appropriately and fed into a Fast Fourier Transform with ten bins. The Fast Fourier Transform function computes the discrete Fourier Transform and its inverse, for each of the subsets of data in each of the fourteen classes. The resulting subsets of data are then 140 data points long each, giving the data a feature space of 140.

After the data is formatted properly and partitioned into the correct subsets, it is ready to be fed into the support vector machine. For the purpose of this work we used a linear support vector machine capable of multi-label classification. We classified the data that we gathered using the Emotiv EPOC across six labels; Passive, Up, Down, Left, Right, and Forwards. Each label corresponded to a class, which in turn contained 25 sets of data for training and 25 for testing. The training data was concatenated horizontally into an array, and then fed into the SVM, alongside a parallel vector of label training data. The linear SVM returns a struct describing the hyperplane that was constructed for classification in the format

$$w \cdot x + b$$

where w is the weight vector of the hyperplane, b is the bias, and x is the input data to be classified. The effectiveness of the given hyperplane to this classification problem can be evaluated by using the trained SVM to classify a set of test labels, which can then be compared to the training labels, and measured for effectiveness.

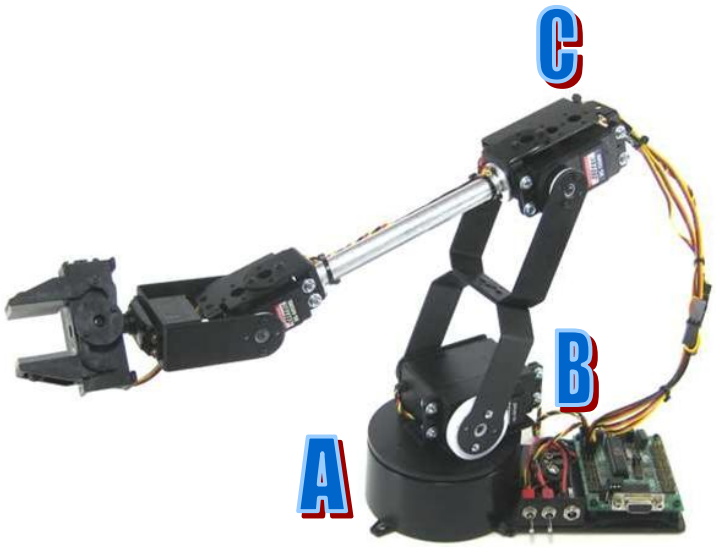

Figure 10. The Lynxmotion AL5D Robot Arm. After disabling the end effector and revolute wrist joint, it is a 3DOF elbow, or anthropomorphic, manipulator.

After the SVM is trained, it can classify any new data into one of the six trained labels. Each of the seven labels that were trained corresponded to one of the seven outputs mechanically for a robotic arm. For this experiment, we used the AL5D 4 degree of freedom robotic arm by Lynxmotion Robotics (Figure 10), with the revolute wrist joint disabled. The AL5D is an anthropomorphic arm in an R⊥R⊦ R configuration. This configuration involves three resolute joints, connected by one perpendicular axis and one orthogonal axis (Figure 11). This configuration is relevant both in its obvious anthropomorphic considerations as well as its use in industry, as R⊥R⊦ R arms make up almost 25% of industrial robots. The end effector of the arm was disabled for the purposes of this work, because movement through the robotic arm's reachable workspace was the method of evaluation. The arm is controlled through the use of an SSC-32 servo controller, which is communicated with through an open serial port. For evaluation purposes, Lynxmotion's HyperTerminal software was used to send commands to the arm. The SSC-32 servo controller received the commands along the serial port as ASCII strings formatted for the device. Figure 12 shows an example ASCII

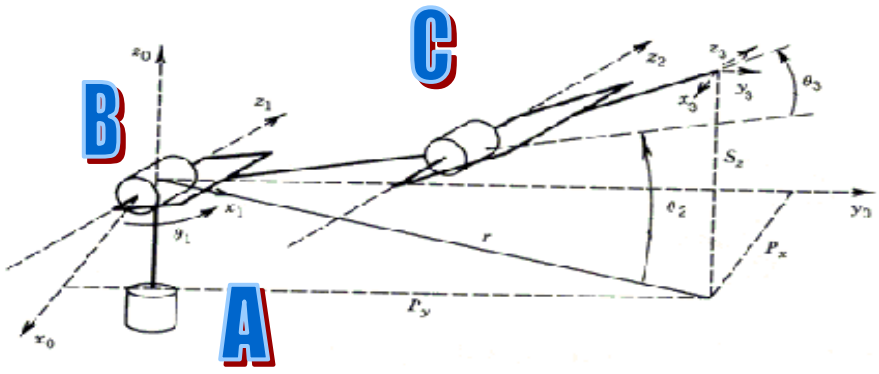

Figure 11. A bare-bones diagram of a PUMA industrial robot arm, which is another example of a 3DOF elbow manipulator.

string that controls the arm. This string consists of six commands. The first command shown, #2, refers to the number of the servo motor being addressed. Any number of servo motors can be addressed by the system, with the number corresponding to the output pin on the SSC-32 servo controller that the servo motor is connected to. After the output string addresses a specific servo, it gives a pulse width to the servo. In this example, the pulse width is given by P1500. This value indicates that a pulse width of 1500, in microseconds, is sent to the servo motor to control it. A pulse width of 1500 should move a servo motor as far as it can go in one direction, depending on the servo and its orientation. This pattern of commands is repeated for a second servo, and followed by the command T2000. This command indicates that all of the servos should reach their final destination at the conclusion of 2000 milliseconds of time elapsed from the moment at which the SSC-32 receives this command. Finally, the line is terminated by <cr>, which is the carriage return, and indicates that the SSC-32 should run the given line.

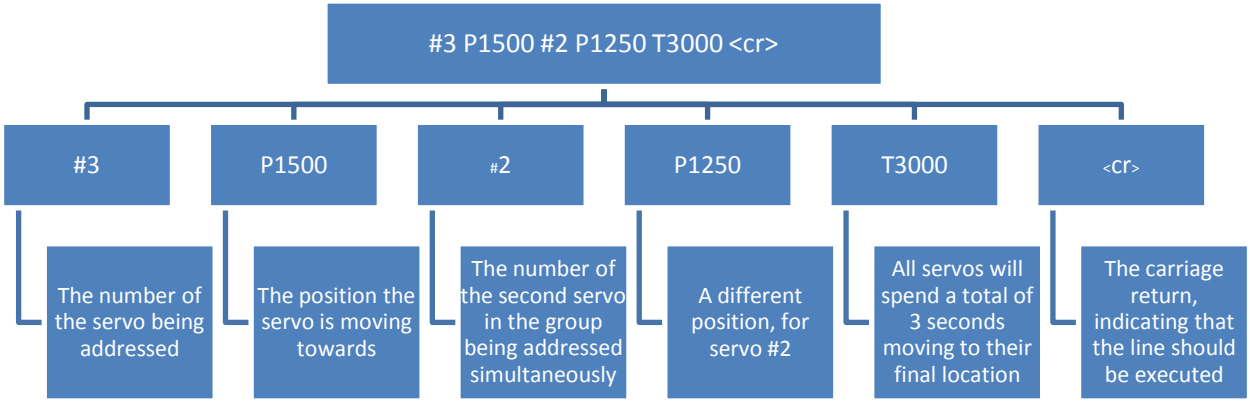

Figure 12. An example command sent via serial communication to the SSC-32 servo controller. The top box is the command in full. The second row of boxes breaks the command down into each token that the SSC-32 servo controller is accepting. The third row of boxes is an explanation of each specific token.

The movement task that we hoped to solve using the Emotiv EPOC EEG was to navigate the arm through its workspace and touch each of a set of five targets (Figure 13). Each of these targets is spaced out in a way such that the user must take advantage of 6 separate movements: forward, up, left, down, and passive. In this task, the passive class will direct the arm to move back to a neutral position, allowing the user to navigate towards a new target. This task is similar to a general setting in which a user may be trying to control an arm through teleoperation.

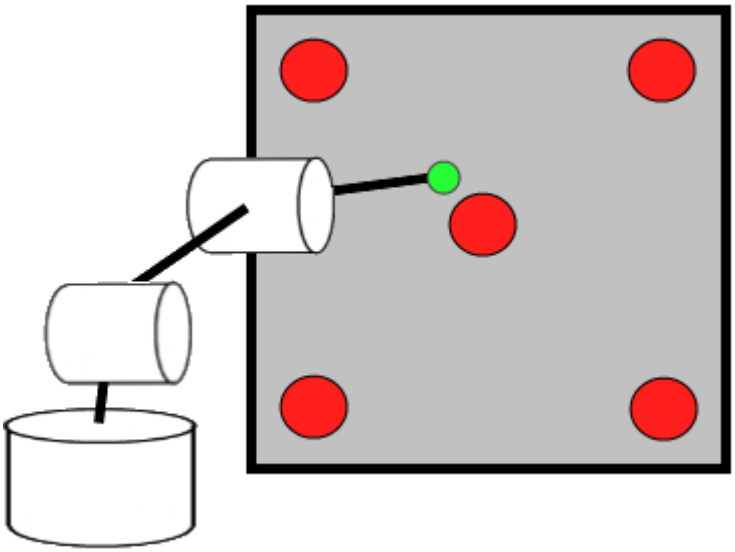

Figure 13. The movement task to be solved. The user must control the robotic arm to touch the green end effector to each of the 5 red targets. The five targets represent extremes in the robots frontal workspace.

Results

As a baseline for the evaluation of our system, we analyzed the success of the Emotiv Control Panel classification software, which was provided with the EEG. The Emotiv Control Panel allows a user to control a virtual object with their minds, allowing for the training of several distinct outputs, which are translated to the screen. The Emotiv system requires an 8 second training time for each action, and can only be trained for a maximum of 4 inputs. We set up the system to train for the maximum of 4 inputs, and tested a user to observe the successful classification rate of the Emotiv System. The user being tested was unable to see the screen, and did not know whether they were testing the Emotiv system or our system. We found that after 80 trials, the Emotiv system correctly classified the user input 16 times, or 20.00% of the time. This classification rate is actually slightly below random chance, which would classify the user input correctly approximately $\frac{1}{4}$ of the time, or 25.00% for 4 inputs (Figure 14). This

result leads us to believe that the Emotiv system is ineffective for classifying a user's electroencephalographic data to four classes.

Figure 14. The Emotiv Control Panel was ineffective in classifying data to 4 outputs, with a success rate of 5% below random chance.

To compare our system of classification with the default system provided by Emotiv, we first tested our system on four classes, which is what is available with the Emotiv software. Using the trained linear support vector machine, we classify the test data that we prepared prior to the training phase. The test data consists of half of the data set, which was organized into a set of 150 outputs. Each of these outputs was classified to one of the four output labels by the hyperplane constructed by the linear support vector machine. After the data is classified, it is put into a vector that corresponds to the label training vector, which contains the proper classification of the test data. The label training vector was created by ordering the labels in the same way that they correspond to the feature training data, to create a sort of answer key to both train and test the SVM. Once the data was classified, we compared the two vectors and calculated the percentage of correct classifications. Given that the SVM is classifying the user's electroencephalographic data into four possible outputs, a system that used random chance would achieve success 25% of the time. The baseline Emotiv system achieved success 20% of the time, making it effectively random and ineffective for classification. Our system, when classifying four outputs, was successful 37% of the time, which is a 12% increase over random chance (Figure 15). This success showed promise towards the classification of electroencephalographic data into larger numbers of classes.

Figure 15. Our system showed a significant increase over random chance in classifying data to four outputs, with a success rate 12% above random. This success rate shows that our system is much more effective than the Emotiv system at classifying data to four outputs.

After classifying the EEG data into four classes more successfully than the Emotiv system, we looked to increase the number of classes to six. Having six separate classes of a data would provide the range of movements needed for our robot arm to successfully complete the given movement task. Using the same system described above for the four output system, we classified the electroencephalographic data into six classes. If the SVM were to classify the data according to random chance, it would classify one of every six outputs correctly since there are six labels (six possible outputs). Therefore the SVM would have a $\frac{1}{6}$, or 16.66% chance of correctly classifying the input of the user. In reality this classification chance would mean that the robot arm that the user is controlling would move in the intended direction 16.66% of the time. When comparing our data to the label test data, we found that our system was able to classify the input of the user correctly 26.00% of the time, based on 150 test inputs. This indicates a nearly 10% increase in effectiveness over random chance (Figure 16).

Robot arm movement success rate (SVM with six classes)	
Random chance	Observed rate with SVM
•16.66%	•26.00%

Figure 16. Our system showed a 9.33% increase in effectiveness over random chance when classifying data to six outputs.

Conclusions and Future Work

In conclusion, the system that we developed for controlling the AL5D Lynxmotion robotic arm through electroencephalographic data shows promise. We were successfully able to classify electroencephalographic data more accurately than the system included with the Emotiv EPOC EEG, and more importantly, we were able to classify user data to 6 outputs instead of the 4 outputs given by the Emotiv system. Unfortunately, we were unable to control the arm with the accuracy necessary to complete our movement task. In order to complete the given task, we will need to either reduce the complexity of the task and therefore decrease the number of outputs needed to a more manageable number, or improve the effectiveness of our classification system. Our system could be further improved through gathering more data and using different optimization techniques to increase the classification rate of our support vector machine. A longer training time for the user would allow the support vector machine to more successfully construct a separating hyperplane. Additionally, different machine learning techniques and methods of optimization for support vector machines would allow one to increase the classification rate of the system. If the classification rate of the system could be increased to anything greater than 50%, then we believe the robot arm could be successfully controlled in a real world situation. For future work, we would like to explore these techniques to increase the classification rate over 50% so that we could start running trials on the effectiveness of this control system. We could also then evaluate the use of the system on different people, and in different experimental environments.

Works Cited

[1] Wolpaw, J. R., Birbaumer, N., McFarland, D. J., Pfurtscheller, G., & Vaughan, T. M. (2002). Brain-computer interfaces for communication and control. Clinical neurophysiology, 113(6), 767-791.

[2] Wolpaw, J. R., McFarland, D. J., Neat, G. W., & Forneris, C. A. (1991). An EEG-based brain-computer interface for cursor control. Electroencephalography and clinical neurophysiology, 78(3), 252-259.

[3] Wolpaw, J. R., McFarland, D. J., & Vaughan, T. M. (2000). Brain-computer interface research at the Wadsworth Center. Rehabilitation Engineering, IEEE Transactions on, 8(2), 222-226.

[4] Iáñez, E., Furió, M. C., Azorín, J. M., Huizzi, J. A., & Fernández, E. (2009). Brain-robot interface for controlling a remote robot arm. In *Bioinspired Applications in Artificial and Natural Computation* (pp. 353-361). Springer Berlin Heidelberg.

[5] Lievesley, R., Wozencroft, M., & Ewins, D. (2011). The Emotiv EPOC neuroheadset: an inexpensive method of controlling assistive technologies using facial expressions and thoughts?. Journal of Assistive Technologies, 5(2), 67-82.

[6] Poor, G. M., Leventhal, L. M., Kelley, S., Ringenberg, J., and Jaffee, S. D. Thought cubes: exploring the use of an inexpensive brain-computer interface on a mental rotation task. The proceedings of the 13th international ACM

SIGACCESS conference on Computers and accessibility (ASSETS ‘11). pp. 291-292.

[7] Zhang, B., Wang, J., & Fuhlbrigge, T. (2010, August). A review of the commercial brain-computer interface technology from perspective of industrial robotics. Automation and Logistics (ICAL), 2010 IEEE International Conference on (pp. 379-384). IEEE.

[8] Thobbi, A., Kadam, R., & Sheng, W. (2010). Achieving Remote Presence using a Humanoid Robot Controlled by a Non-Invasive BCI Device. International Journal on Artificial Intelligence and Machine Learning, 10, 41-45.

[9] Vourvopoulos, A., & Liarokapis, F. (2011, May). Brain-controlled NXT Robot: Tele-operating a robot through brain electrical activity. Games and Virtual Worlds for Serious Applications (VS-GAMES), 2011 Third International Conference on (pp. 140-143). IEEE.

[10] Emotiv, Emotiv Software Development Kit User Manual Release 1.0.0.3

[11] Ranky, G. N., & Adamovich, S. (2010, March). Analysis of a commercial EEG device for the control of a robot arm. Bioengineering Conference, Proceedings of the 2010 IEEE 36th Annual Northeast (pp. 1-2). IEEE.

[12] Leuthardt, E. C., Schalk, G., Wolpaw, J. R., Ojemann, J. G., & Moran, D. W. (2004). A brain–computer interface using electrocorticographic signals in humans. Journal of neural engineering, 1(2), 63.

[13] Ubbens, T.W.; Schuurman, D.C., "Vision-based obstacle detection using a support vector machine," Electrical and Computer Engineering, 2009. CCECE '09. Canadian Conference on , vol., no., pp.459,462, 3-6 May 2009

[14] Collura, TF. (1993) History and Evolution of Electroencephalographic Instruments and Techniques. Journal of Clinical Neurophysiology (476-504). Raven Press

[15] Jasper, H. H. (1958). The ten-twenty electrode placement system of the International Federation. Electroencephalography and Clinical Neurophysiology, 10, 371-375

[16] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273-297.

[17] Vapnik, V. N. (1999). An overview of statistical learning theory. *Neural Networks, IEEE Transactions on*, *10*(5), 988-999.

[18] Vapnik, V. (1999). *The Nature of Statistical Learning Theory*. Springer.

[19] van Vliet, M., Robben, A., Chumerin, N., Manyakov, N. V., Combaz, A., & Van Hulle, M. M. (2012, January). Designing a Brain-Computer Interface controlled video-game using consumer grade EEG hardware. Biosignals and Biorobotics Conference (BRC), 2012 ISSNIP (pp. 1-6). IEEE.

[20] Cauwenberghs, G., & Poggio, T. (2001). Incremental and decremental support vector machine learning. *Advances in neural information processing systems*, 409-415.

[21] Domingos, P., & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine learning*, *29*(2-3), 103-130.

[22] John, G. H., & Langley, P. (1995, August). Estimating continuous distributions in Bayesian classifiers. *Proceedings of the Eleventh conference on Uncertainty in*

artificial intelligence (pp. 338-345). Morgan Kaufmann Publishers Inc.

[23] Nouretdinov, I., Costafreda, S. G., Gammerman, A., Chervonenkis, A., Vovk, V., Vapnik, V., & Fu, C. H. (2011). Machine learning classification with confidence: application of transductive conformal predictors to MRI-based diagnostic and prognostic markers in depression. *Neuroimage*, *56*(2), 809-813.

[24] Schölkopf, B., Simard, P., Vapnik, V., & Smola, A. J. (1997). Improving the accuracy and speed of support vector machines. In *Advances in Neural Information Processing Systems 9: Proceedings of the 1996 Conference [on Neural Information... Held in Denver... 1996]* (Vol. 9, p. 375). The MIT Press.

[25] Suykens, J. A., & Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural processing letters*, *9*(3), 293-300.

[26] Wang, L. (Ed.). (2005). *Support Vector Machines: theory and applications* (Vol. 177). Springer Verlag.

[27] Blankertz, B., Curio, G., & Muller, K. R. (2002). Classifying single trial EEG: Towards brain computer interfacing. Advances in neural information processing systems, 1, 157-164.

[28] Gomez-Gil, Jaime; San-Jose-Gonzalez, Israel; Nicolas-Alonso, Luis Fernando; Alonso-Garcia, Sergio. 2011. "Steering a Tractor by Means of an EMG-Based Human-Machine Interface." Sensors 11, no. 7: 7110-7126.

[29] Iturrate, I., Antelis, J. M., Kubler, A., & Minguez, J. (2009). A noninvasive brain-actuated wheelchair based on a P300 neurophysiological protocol and automated navigation. Robotics, IEEE Transactions on, 25(3), 614-627.

[30] Millan, J. R., Renkens, F., Mouriño, J., & Gerstner, W. (2004). Noninvasive brain-actuated control of a mobile robot by human EEG. Biomedical Engineering, IEEE Transactions on, 51(6), 1026-1033.

Acknowledgements

Thanks to Professor Parker and Bo Xiong '13,
without whom this work would not have been possible.

We additionally thank Shaun Chung ’15,
for his contributions towards the success of this system

Appendix A

SixClassDataCollection.m

```
%Jim O'Connor
%EEG data analysis and formatting (6 classes)
%
%This script takes data returned fromo the Emotiv EPOC EEG
%TestBench software, which is stored as a series of comma
%separated value files, and performs a series of operations on
%the data. First, a Fast Fourier Transform is applied to the
%data, to get rid of the time domain and prepare the data for
%use by the Support Vector Machine. The absolute value function
%is then applied to the data, and the data is reshaped into an
%appropriate feature space. Next, the feature is split into two
%sections, one section composed of five samples for training,
%and the other five sample section for testing. These features
%are then ready to be used by the Support Vector Machine.


addpath('lsvm');
addpath('helpfun');
addpath('svm');

global bins
bins = 22; %0.26 at 22

%Initialize the variables used in the looping and formatting
processes
initial_index=0;
training_index=0;
test_index=0;
```

```
%Read in the comma separated value data to matlab. The 2nd
parameter of the
%csvread function allows the function to skip the header of the
CSV file,
%which is not part of the data being analyzed.
csv_data = vertcat(csvread('Shaun-Passive One-
28.04.13.20.00.16.CSV',1,0),...
    csvread('Shaun-Passive Two-28.04.13.20.12.01.CSV',1,0),...
    csvread('Shaun-Passive Three-28.04.13.20.20.49.CSV',1,0),...
    csvread('Shaun-Passive Four-28.04.13.20.32.12.CSV',1,0),...
    csvread('Shaun-Passive Five-28.04.13.20.42.21.CSV',1,0));

%The initial loop runs 50 times, allowing it to split the 7000
line data
%into 10 equal samples of 700 data points each, for 14 features.
A Fast
%Fourier Transform is then applied to the CSV data, followed by
an absolute
%value function. The resulting data is reshaped into 50 vectors
of length
%140, which is the feature space.
for n = 1:50
    initial_index = initial_index+1;
    fft_signal{initial_index} =
abs(fft(csv_data(((initial_index*700)-
699):initial_index*700,1:14),bins));
    new_fea{initial_index} =
reshape(fft_signal{initial_index},numel(fft_signal{initial_index
}),1);
end
```

```
%The resulting dataset is then split in half through the use of
two loops,
%each of which runs for half of the 50 vector long array, saving
the two
%halves in new matrices. Each of these matrices is named
according to its
%label and its use in either the training set or the test set
for the
%Support Vector Machine.
for i = 1:25
    training_index = training_index+1;
    Passive_Training_Set{training_index} =
new_fea{training_index};
end
for i = 1:25
    test_index = test_index+1;
    Passive_Test_Set{test_index} = new_fea{test_index+25};
end


%The index variables are re-initialized each time a new data set
is looped
%through and formatted.
initial_index=0;
training_index=0;
test_index=0;

csv_data = vertcat(csvread('Shaun-Up One-
28.04.13.20.03.19.CSV',1,0),...
    csvread('Shaun-Up Two-28.04.13.20.13.21.CSV',1,0),...
    csvread('Shaun-Up Three-28.04.13.20.22.18.CSV',1,0),...
```

```
    csvread('Shaun-Up Four-28.04.13.20.34.24.CSV',1,0),...
    csvread('Shaun-Up Five-28.04.13.20.44.01.CSV',1,0));

for n = 1:50
    initial_index = initial_index+1;
    fft_signal{initial_index} =
abs(fft(csv_data(((initial_index*700)-
699):initial_index*700,1:14),bins));
    new_fea{initial_index} =
reshape(fft_signal{initial_index},prod(size(fft_signal{initial_i
ndex})),1);
end
for i = 1:25
    training_index = training_index+1;
    Up_Training_Set{training_index} = new_fea{training_index};
end
for i = 1:25
    test_index = test_index+1;
    Up_Test_Set{test_index} = new_fea{test_index+25};
end

%Each new set of loops handles a new set of data, each
corresponding to a
%new class. There are six classes total.

initial_index=0;
training_index=0;
test_index=0;

csv_data = vertcat(csvread('Shaun-Down One-
28.04.13.20.05.55.CSV',1,0),...
    csvread('Shaun-Down Two-28.04.13.20.14.48.CSV',1,0),...
```

```
    csvread('Shaun-Down Three-28.04.13.20.24.12.CSV',1,0),...
    csvread('Shaun-Down Four-28.04.13.20.36.13.CSV',1,0),...
    csvread('Shaun-Down Five-28.04.13.20.45.49.CSV',1,0));

for n = 1:50
    initial_index = initial_index+1;
    fft_signal{initial_index} =
abs(fft(csv_data(((initial_index*700)-
699):initial_index*700,1:14),bins));
    new_fea{initial_index} =
reshape(fft_signal{initial_index},prod(size(fft_signal{initial_i
ndex})),1);
end
for i = 1:25
    training_index = training_index+1;
    Down_Training_Set{training_index} = new_fea{training_index};
end
for i = 1:25
    test_index = test_index+1;
    Down_Test_Set{test_index} = new_fea{test_index+25};
end


initial_index=0;
training_index=0;
test_index=0;

csv_data = vertcat(csvread('Shaun-Left One-
28.04.13.20.07.45.CSV',1,0),...
    csvread('Shaun-Left Two-28.04.13.20.16.18.CSV',1,0),...
    csvread('Shaun-Left Three-28.04.13.20.25.33.CSV',1,0),...
    csvread('Shaun-Left Four-28.04.13.20.37.42.CSV',1,0),...
```

```
    csvread('Shaun-left Five-28.04.13.20.47.22.CSV',1,0));

for n = 1:50
    initial_index = initial_index+1;
    fft_signal{initial_index} =
abs(fft(csv_data(((initial_index*700)-
699):initial_index*700,1:14),bins));
    new_fea{initial_index} =
reshape(fft_signal{initial_index},prod(size(fft_signal{initial_i
ndex})),1);
end
for i = 1:25
    training_index = training_index+1;
    Left_Training_Set{training_index} = new_fea{training_index};
end
for i = 1:25
    test_index = test_index+1;
    Left_Test_Set{test_index} = new_fea{test_index+25};
end


initial_index=0;
training_index=0;
test_index=0;

csv_data = vertcat(csvread('Shaun-Right One-
28.04.13.20.09.14.CSV',1,0),...
    csvread('Shaun-Right Two-28.04.13.20.17.37.CSV',1,0),...
    csvread('Shaun-Right Three-28.04.13.20.28.11.CSV',1,0),...
    csvread('Shaun-Right Four-28.04.13.20.39.11.CSV' ,1,0),...
    csvread('Shaun-Right Five-28.04.13.20.48.56.CSV',1,0));
```

```
for n = 1:50
    initial_index = initial_index+1;
    fft_signal{initial_index} =
abs(fft(csv_data(((initial_index*700)-
699):initial_index*700,1:14),bins));
    new_fea{initial_index} =
reshape(fft_signal{initial_index},prod(size(fft_signal{initial_i
ndex})),1);
end
for i = 1:25
    training_index = training_index+1;
    Right_Training_Set{training_index} =
new_fea{training_index};
end
for i = 1:25
    test_index = test_index+1;
    Right_Test_Set{test_index} = new_fea{test_index+25};
end


initial_index=0;
training_index=0;
test_index=0;

csv_data = vertcat(csvread('Shaun-Forward One-
28.04.13.20.10.41.CSV',1,0),...
    csvread('Shaun-Forward Two-28.04.13.20.19.26.CSV',1,0),...
    csvread('Shaun-Forward Three-28.04.13.20.29.56.CSV',1,0),...
    csvread('Shaun-Forward Four-28.04.13.20.40.45.CSV',1,0),...
    csvread('Shaun-Forward Five-28.04.13.20.50.32.CSV',1,0));
for n = 1:50
    initial_index = initial_index+1;
```

```
    fft_signal{initial_index} =
abs(fft(csv_data(((initial_index*700)-
699):initial_index*700,1:14),bins));
    new_fea{initial_index} =
reshape(fft_signal{initial_index},prod(size(fft_signal{initial_i
ndex})),1);
end
for i = 1:25
    training_index = training_index+1;
    Forward_Training_Set{training_index} =
new_fea{training_index};
end
for i = 1:25
    test_index = test_index+1;
    Forward_Test_Set{test_index} = new_fea{test_index+25};
end

%The Label Training Data consists of 150 values, each
corresponding
%to one of six different labels. These values are ordered in the
same
%way as the Feauture Training Data, so that each set of 25 test
samples
%in each feature corresponds to the correct label.
Label_Training_Data = [];
for n=0:5
    for i=1:25
        Label_Training_Data=(horzcat(Label_Training_Data,n));
    end
end
```

```
%A cell matrix is filled up with the different sets of training
data so
%that it can be distributed into the Training Array.
Feature_Training_Data{1} = Passive_Training_Set';
Feature_Training_Data{2} = Up_Training_Set';
Feature_Training_Data{3} = Down_Training_Set';
Feature_Training_Data{4} = Left_Training_Set';
Feature_Training_Data{5} = Right_Training_Set';
Feature_Training_Data{6} = Forward_Training_Set';

%The Training Array is initialized
Feature_Training_Array = [];

%The feature training data is formatted and moved from the cell
matrix to
%the training array. It is horizontally concatenated onto the
array so that
%each column of data corresponds to a feature, while each row is
a data
%point. There are 150 features that belong to 6 classes.
for i=1:6
    for n=1:25
        Feature_Training_Array =
horzcat(Feature_Training_Array,Feature_Training_Data{i}{n});
    end
end


%The test data is handled and formatted in the same way as the
training
%data.
Feature_Test_Data{1} = Passive_Test_Set';
```

```
Feature_Test_Data{2} = Up_Test_Set';
Feature_Test_Data{3} = Down_Test_Set';
Feature_Test_Data{4} = Left_Test_Set';
Feature_Test_Data{5} = Right_Test_Set';
Feature_Test_Data{6} = Forward_Test_Set';

Feature_Test_Array = [];

for i=1:6
    for n=1:25
        Feature_Test_Array =
horzcat(Feature_Test_Array,Feature_Test_Data{i}{n});
    end
end


%The SVM is trained using the feature training array and the
label training
%data with a lambda value of 2. The resulting SVM struct, which
contains
%the paramaters of the optimal hyperplane for classification, is
saved as
%model.
model = lsvm_train(Feature_Training_Array', Label_Training_Data',
2); %0.26 at 2 / 0.1,0.2

%The effectiveness of the SVM is tested through the lsvm_predict
function,
%which builds a label prediction array based on the
classification of a
%Feature set without labels.
predictlabel= lsvm_predict(Feature_Test_Array', model);
```

```
%The newly classified array, predictlabel, is compared to the
correct
%classification, or Label_Training_Data, which then returns a
percentage of
%accuracy.
sum(predictlabel == Label_Training_Data')/(length(predictlabel))
```

FourClassDataCollection.m

```
%Jim O'Connor
%EEG data analysis and formatting (4 classes)
%
%This particular version of the code uses four classes, in order
%to compare our system to the benchmark of the Emotiv system.

addpath('lsvm');
addpath('helpfun');
addpath('svm');

global bins
bins = 23; %0.37 at 23

%Initialize the variables used in the looping and formatting
processes
initial_index=0;
training_index=0;
test_index=0;



%Read in the comma separated value data to matlab. The 2nd
parameter of the
%csvread function allows the function to skip the header of the
CSV file,
%which is not part of the data being analyzed.
csv_data = vertcat(csvread('Shaun-Passive One-
28.04.13.20.00.16.CSV',1,0),...
    csvread('Shaun-Passive Two-28.04.13.20.12.01.CSV',1,0),...
    csvread('Shaun-Passive Three-28.04.13.20.20.49.CSV',1,0),...
```

```
    csvread('Shaun-Passive Four-28.04.13.20.32.12.CSV',1,0),...
    csvread('Shaun-Passive Five-28.04.13.20.42.21.CSV',1,0));

%The initial loop runs 50 times, allowing it to split the 7000
line data
%into 50 equal samples of 700 data points each, for 14 features.
A Fast
%Fourier Transform is then applied to the CSV data, followed by
an absolute
%value function. The resulting data is reshaped into 50 vectors
of length
%140, which is the feature space.
for n = 1:50
    initial_index = initial_index+1;
    fft_signal{initial_index} =
abs(fft(csv_data(((initial_index*700)-
699):initial_index*700,1:14),bins));
    new_fea{initial_index} =
reshape(fft_signal{initial_index},prod(size(fft_signal{initial_i
ndex})),1);
end

%The resulting dataset is then split in half through the use of
two loops,
%each of which runs for half of the 50 vector long array, saving
the two
%halves in new matrices. Each of these matrices is named
according to its
%label and its use in either the training set or the test set
for the
%Support Vector Machine.
for i = 1:25
```

```
    training_index = training_index+1;
    Passive_Training_Set{training_index} =
new_fea{training_index};
end
for i = 1:25
    test_index = test_index+1;
    Passive_Test_Set{test_index} = new_fea{test_index+25};
end


%The index variables are re-initialized each time a new data set
is looped
%through and formatted.
initial_index=0;
training_index=0;
test_index=0;

csv_data = vertcat(csvread('Shaun-Up One-
28.04.13.20.03.19.CSV',1,0),...
    csvread('Shaun-Up Two-28.04.13.20.13.21.CSV',1,0),...
    csvread('Shaun-Up Three-28.04.13.20.22.18.CSV',1,0),...
    csvread('Shaun-Up Four-28.04.13.20.34.24.CSV',1,0),...
    csvread('Shaun-Up Five-28.04.13.20.44.01.CSV',1,0));

for n = 1:50
    initial_index = initial_index+1;
    fft_signal{initial_index} =
abs(fft(csv_data(((initial_index*700)-
699):initial_index*700,1:14),bins));
    new_fea{initial_index} =
reshape(fft_signal{initial_index},prod(size(fft_signal{initial_i
ndex})),1);
```

```
end
for i = 1:25
    training_index = training_index+1;
    Up_Training_Set{training_index} = new_fea{training_index};
end
for i = 1:25
    test_index = test_index+1;
    Up_Test_Set{test_index} = new_fea{test_index+25};
end

%Each new set of loops handles a new set of data, each
corresponding to a
%new class. There are seven classes total.

initial_index=0;
training_index=0;
test_index=0;

csv_data = vertcat(csvread('Shaun-Down One-
28.04.13.20.05.55.CSV',1,0),...
    csvread('Shaun-Down Two-28.04.13.20.14.48.CSV',1,0),...
    csvread('Shaun-Down Three-28.04.13.20.24.12.CSV',1,0),...
    csvread('Shaun-Down Four-28.04.13.20.36.13.CSV',1,0),...
    csvread('Shaun-Down Five-28.04.13.20.45.49.CSV',1,0));

for n = 1:50
    initial_index = initial_index+1;
    fft_signal{initial_index} =
abs(fft(csv_data(((initial_index*700)-
699):initial_index*700,1:14),bins));
```

```
    new_fea{initial_index} =
reshape(fft_signal{initial_index},prod(size(fft_signal{initial_i
ndex})),1);
end
for i = 1:25
    training_index = training_index+1;
    Down_Training_Set{training_index} = new_fea{training_index};
end
for i = 1:25
    test_index = test_index+1;
    Down_Test_Set{test_index} = new_fea{test_index+25};
end


initial_index=0;
training_index=0;
test_index=0;

csv_data = vertcat(csvread('Shaun-Left One-
28.04.13.20.07.45.CSV',1,0),...
    csvread('Shaun-Left Two-28.04.13.20.16.18.CSV',1,0),...
    csvread('Shaun-Left Three-28.04.13.20.25.33.CSV',1,0),...
    csvread('Shaun-Left Four-28.04.13.20.37.42.CSV',1,0),...
    csvread('Shaun-left Five-28.04.13.20.47.22.CSV',1,0));

for n = 1:50
    initial_index = initial_index+1;
    fft_signal{initial_index} =
abs(fft(csv_data(((initial_index*700)-
699):initial_index*700,1:14),bins));
```

```
    new_fea{initial_index} =
reshape(fft_signal{initial_index},numel(fft_signal{initial_index
}),1);
end
for i = 1:25
    training_index = training_index+1;
    Left_Training_Set{training_index} = new_fea{training_index};
end
for i = 1:25
    test_index = test_index+1;
    Left_Test_Set{test_index} = new_fea{test_index+25};
end


%The Label Training Data consists of 100 values, each
corresponding
%to one of 4 different labels. These values are ordered in the
same
%way as the Feauture Training Data, so that each set of 25 test
samples
%in each feature corresponds to the correct label.
Label_Training_Data = [];
for n=0:3
    for i=1:25
        Label_Training_Data=(horzcat(Label_Training_Data,n));
    end
end

%A cell matrix is filled up with the different sets of training
data so
%that it can be distributed into the Training Array.
Feature_Training_Data{1} = Passive_Training_Set';
```

```
Feature_Training_Data{2} = Up_Training_Set';
Feature_Training_Data{3} = Down_Training_Set';
Feature_Training_Data{4} = Left_Training_Set';
%The Training Array is initialized
Feature_Training_Array = [];

%The feature training data is formatted and moved from the cell
matrix to
%the training array. It is horizontally concatenated onto the
array so that
%each column of data corresponds to a feature, while each row is
a data
%point. There are 100 features that belong to 4 classes.
for i=1:4
    for n=1:25
        Feature_Training_Array =
horzcat(Feature_Training_Array,Feature_Training_Data{i}{n});
    end
end


%The test data is handled and formatted in the same way as the
training
%data.
Feature_Test_Data{1} = Passive_Test_Set';
Feature_Test_Data{2} = Up_Test_Set';
Feature_Test_Data{3} = Down_Test_Set';
Feature_Test_Data{4} = Left_Test_Set';

Feature_Test_Array = [];

for i=1:4
```

```
    for n=1:25
        Feature_Test_Array =
horzcat(Feature_Test_Array,Feature_Test_Data{i}{n});
    end
end


%The SVM is trained using the feature training array and the
label training
%data with a lambda value of 2. The resulting SVM struct, which
contains
%the paramaters of the optimal hyperplane for classification, is
saved as
%model.
model = lsvm_train(Feature_Training_Array', Label_Training_Data',
2); %0.37 at 2/0.1/etc

%The effectiveness of the SVM is tested through the lsvm_predict
function,
%which builds a label prediction array based on the
classification of a
%Feature set without labels.
predictlabel= lsvm_predict(Feature_Test_Array', model);

%The newly classified array, predictlabel, is compared to the
correct
%classification, or Label_Training_Data, which then returns a
percentage of
%accuracy.
sum(predictlabel == Label_Training_Data')/(length(predictlabel))
```

Appendix B

Table of Figures