DISTRACTION DETECTION AND INTENTION RECOGNITION FOR GESTURE-CONTROLLED UNMANNED AERIAL VEHICLE OPERATION

by

DEBAYAN DATTA

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTERS OF SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2022

To

Guruji

and

My Beloved Parents

ACKNOWLEDGEMENTS

The completion of this dissertation wouldn't be possible if I wouldn't find the invaluable assistance, support, and guidance of few people whom I would like to acknowledge.

Firstly, I would like to show my gratitude towards my supervising professor, Dr. Manfred Huber, who is an exemplary researcher himself, and helped me throughout the dissertation by providing his extraordinary feedback and guiding me to structure the work and formulate the methodology. Your constant encouragement and guidance have helped me every day to strive to find plausible answers to the important research questions that developed during the dissertation.

I would further like to thank Dr. David Levine and Dr. Vamsikrishna Gopikrishna for gracing the defense committee and providing your valuable feedback on my work. Your ideas and thoughts will surely help me to shape the future works.

Then, I would like to thank my manager, Mr. Bito Irie, who always guided me towards perfection. Without the support and financial aid, you provided me, it would be very difficult for me to carry on with my graduate studies.

Life has given me two best friends without whom I would never be able to keep the composure and belief to complete this work. I would like to offer my gratitude to Sambuddha Majumder and Sreekanth Bhavaraju who has always been there when I

needed them the most.

I would have never reached to this point, without the blessings of my Guruji and the constant support of my parents. Without them, I would have never made in to my graduate studies in the United States.

Lastly, I thank almighty god for his choicest blessing that he has always offered me with.

May 13, 2022

ABSTRACT

DISTRACTION DETECTION AND INTENTION RECOGNITION FOR GESTURE-CONTROLLED UNMANNED AERIAL VEHICLE OPERATION

DEBAYAN DATTA, M.S.

The University of Texas at Arlington, 2022

Supervising Professor: Manfred Huber

Gesture Control as a way to replace more conventional remote-control operations has been pursued for a significant period of time with different levels of success. The use of gestures to control different types of human interfaces is today predominantly seen in the multimedia sector. People perform easy and intuitive gestures to control their televisions, to interact with multimedia, and to play games. Also, much research has been carried out to experiment with human interfaces to numerous augmented and virtual reality devices and tasks. The results of these experiments were so exciting that researchers started to expand the use of gestures to control physical real life objects using gestures.

One such gesture control platform is controlling an Unmanned Aerial Vehicle (UAV) during flight using hand movements and other types of significant gestures. While the use of gestures as an intuitive way to control such a platform is very promising, it also brings with it a number of challenges and risks that need to be addressed. The main challenges here arise from the versatility and universal character of gestures which make them more susceptible to misinterpretations in terms of user

intentions in particular in the context of real world distractions which can lead to the unintentional generation of gestures outside the context of the control task. The scope of hand gestures is so varied that predicting the gesture using a designated framework could, at a point, be risky for the UAV, where a non-gesture movement possibly generated in the context of natural or artificial distraction could be classified as a control signature and cause a dramatic failure of the UAV. For instance, if a human subject entitled to control the UAV using their hand movements is subjected to an unanticipated situation that made him busy to perform any other similar-looking gesture, and the framework misinterprets it to be a known gesture, the result of executing the unintended command might end the life of that drone.

In the work presented here, we have mainly focused on an instance where the human subject was considered to be subjected to different types of distraction while controlling the UAV using hand gestures. The UAV controls are suspended to avoid any self or collateral damage whenever any distraction from the brain or hand movements is identified. Finally, once distractions abate, the UAV shifts to ready mode in order to receive new commands.

The experimentation is performed using wearable Electromyography (EMG) Sensor armbands and non-invasive Electroencephalography (EEG) Sensors worn on the head. The Armband is co-fitted with both EMG and Inertial Measurement Unit (IMU) Sensors responsible for extracting the muscle movement and the motion sensing data. The EEG Sensor is used to get the brain data to identify the distracted brain sequences during the operation.

An intuitive gesture set for drone operation is designed and a neural network pipeline was set up which was used to identify the gesture performed as well as the current state of the brain. The pipeline constituted a Long-Short Term Memory (LSTM) network classifying the gestures and an Anomaly Detector, which identified

periods of operation that correspond to distractions where the human was likely no longer focusing on UAV operations. The LSTM was able to classify 99.52% of the gesture set correctly and the anomaly detector had an accuracy of 90.0006%. The precision of the distraction classifier was 91.86% and the recall was 93.89% for a dataset having nearly 30% of distracted data samples. The signals were recorded from a single human subject over 3 minutes interval and 20 repetitions with a rest period of 5-10 minutes to avoid brain and muscle fatigue.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

Since the word 'Artificial Intelligence' was first adopted in 1956 at the Dartmouth Conference organized by American Computer Scientist John McCarthy, it has grown in all possible directions and is still expanding. The past couple of decades has seen even more research and innovation in Artificial Intelligence. A part of this research was to develop Unmanned Vehicles that can perform operations considered risky or impossible for human beings. Beyond that, robots or remote computers or machines operated from a distance have paved the way for distant control applications and multimedia. From children's toys to military equipment, remotely controllable devices have been adapted over the years to make the system more intuitive and engaging. Even in the field of mass manufacturing, activities that need to be performed under strict human supervision have been performed using highly efficient semi-autonomous robots which are partially controlled by a human from a distance.

In modern days, the flexibility of remote communication and the advancement of artificial intelligence jointly made the researchers more innovative in experimenting with different forms of control systems, especially the ones that increase human engagement to a greater extent, in order to extend the flexibility of a human being into the mechanical world. One such model of communication is communication through gestures performed by hand movements.

Human Beings are blessed with the most flexible forelimbs in the animal kingdom. Whatever actions a person performs in a day involve the forelimbs to a great

degree. In the area of the control systems, the most common ones are while we drive a car or use the remote to change the channel on the television. We are so used to this kind of communication that we don't even need to concentrate entirely on doing these activities. In other words, we are surrounded by remote communication in today's fast-changing world.

This motivated the present-day researchers to use gesture-based control mechanisms to drive or monitor a robot or unmanned vehicle system. There are different ways to recognize a gesture known to people. The most common is by image recognition using a digital or depth-sensing camera [1, 2, 3, 4, 5] . The other is by using wearable devices that provide helpful information from muscle movement or physical intentions. The use of a live camera is easy to set up and cleaner. But it restricts mobility greatly by requiring to stay within the area covered by the camera. On the other hand, while wearable sensors are generally harder to set up initially than camera-based systems, the ability to be physically flexible and to move around provided by theses systems increases the chances of long-time communication and a provides the potential for a more intuitive and stress-free environment.

The only way to obtain this flexibility with minimal setup is by using a smartly built wearable device that is easy to put on and is able to recognize the muscle movements while performing various actions. A different combination of muscle states is used to depict a certain gesture. This will help increase the mobility of the person as these wearable devices are mostly wireless. Typically, the sensors output a range of electronic signal values that is similar for a specific type of gesture. And any clustering or supervised learning algorithm can be used to classify them. a useful addition to these types of wearable devices is by adding an Inertial Measurement Unit (IMU) to its core so that any type of accelerometer or gyroscope value extracted from the

sensors and pointing to a specific hand movement can be used in addition to learn the sequence of operations.

The amount of flexibility in these types of gesture recognition is so fascinating that even tough control systems like operating an Unmanned Vehicle System (UVS) using intuitive gestures have been experimented with so many times in the past [6, 7, 8]. Most of these research projects used only muscle data to control the UVS and did not include any other types of sensors. This imposes limitations on what gestures are identifiable from a limited set of sensor locations, either dramatically reducing the extractable movements or requiring sensor probes to be placed all over the arm, making it more inconvenient and harder to use¿ Including other sensors such as the sequence of information from the IMU can overcome some of this, for example by giving us a good representation of the overall arm movements made by the person while controlling the vehicle and thus partially replacing the need for EMG sensors to read shoulder and elbow muscle activations and just concentrating on wrist and finger muscles which can be extracted from a single device mounted on the lower arm. Identifying the hand movements as time sequence data could also be useful in creating vivid and more intuitive gestures. Recognizing these type of control signatures makes user interaction easier to the control interface and could be used in a regular setting. However, to introduce a plug-and-play mechanism, the proposed system should be robust and not require any user to have significant amounts of training on the specific modules just to control the vehicle or use it for multimedia purposes and also should minimize sensing information. This could be incorporated by introducing the Neural Network Classifiers which can detect the gestures in a general sense and not require calibration to the system every time a new user tries it. But use of a neural network or advanced learning algorithms usually requires large-scale data, which is itself a challenge. Despite these kinds of challenges, if we are able to train the classifiers to

understand variations and not just the trend, we will be able to open the recognition scope to a great extent even if we have less data to train our models on [6].

If we are able to reliably classify the gestures, we will be able to send them to a common interface to generate commands for driving an unmanned vehicle system. To do that, we need to create a gesture vocabulary that the common interface is able to map to certain commands to the UVS and finally send them as control signatures to the UVS. Later, the UVS will be able to send a feedback signal before and after the execution of the operation to make sure the command was successfully registered and deployed. This process loop will make any control system using a hand movement and subsequent gesture control robust and mobile at least when concentrating on controlling the system.

Understanding the freedom of mobility of this mechanism, however, we recognize that the subject which is supposed to control the UVS will no longer be required to be in a closed and socially disconnected environment. The subject will be able to move freely in the open world, and parallely control the UVS. As the gesture recognition is also facilitated using the wearable device, the subject doesn't have to be in close communication with the UVS as the UVS wouldn't need to look at the subject for real time movement detection using the on-board camera. This, to some extent, can be considered as a great feature.

However, in this setting, the outer world is full of distractions which the subject controlling the UVS could fall prey to. The distractions could be any kind of visual, audible or physical information that is foreign and unpredictable. The nature of the distractions could be introducing a change of emotion or state of mind of the person controlling the UVS, moving the person's attention away from the control task. This might let his hand engage in activities that are not gestures for the command system but that utilize the same muscles in similar ways, such as waving to a friend or

gesturing to an oncoming car or person. This, in turn, increases the risk that the activity might be classified as a gesture and could be very harmful for the Unmanned Vehicle especially if it is an aerial vehicle.

Therefore, it becomes very important to identify and monitor the state of mind of the person concerned to control the aerial vehicle. This is particularly important since any person who is handling an aerial vehicle (the controller) and the vehicle might not have a line of sight. As a result, the system is already physically disconnected and thus has no ability to judge the state of the controller by itself. At this point, if the controller has a change of intentions which was unpredictable or unforeseeable once the vehicle is airborne, it might be very expensive for the vehicle to erroneously register non-gestures as gestures.

This provides the motivation to use an Electroencephalography (EEG) Sensor as a second sensor that would be worn on the head. This sensor would monitor the electrical activity of certain brain regions, which will be further processed to identify a distracted or a non-distracted brain state. The benefit of the use of this sensor is that it is non-invasive, which increases the wearability of the sensor as part of the overall system.

The combination of these two sensing models can be used to finally generate the control sequences and send them to the aerial vehicle to respond while avoiding the generation of incorrect commands due to reactions to distractions. The overall model is given below in Figure 1.1:

1.2 Contribution

This thesis proposes, develops, and tests an integrated system for remote commands to control an UVS using gestures that utilizes an EMG, IMU, and EEG data from easy to use mobile devices to address both automatic command extraction and

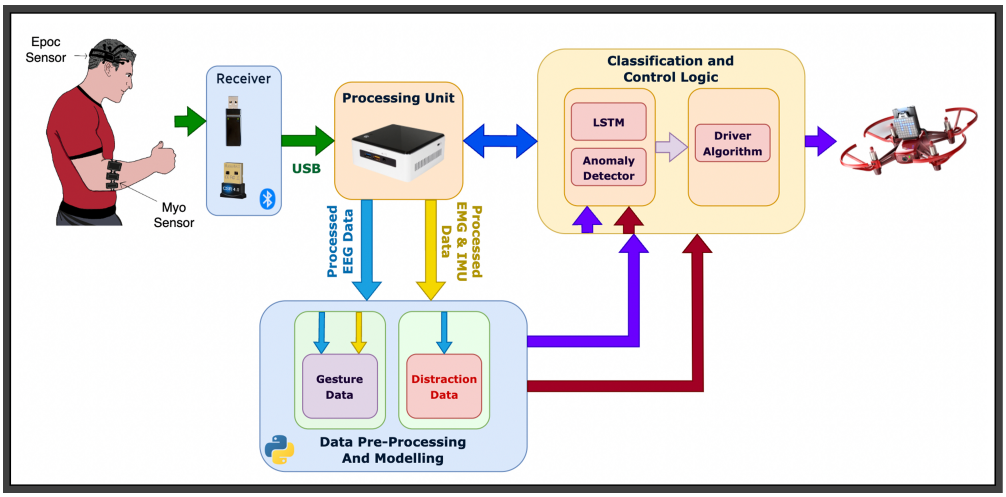


Figure 1.1: Overall Model

distraction identification to obtain a more robust and usable overall system that can be used in more general environments. Particular contributions in this thesis are:

- Design of a Long-Short Term Memory Recurrent Neural Network that was trained on both Electroencephalography and Electromyography Data to produce the intended gesture using the hand movements.
- Design of an anomaly detector using auto-encoders to identify the anomaly in the Electroencephalography Sensor Information. The non-distracted brain data was regarded as non-anomaly.
- Design of a common sequence generator from the classification sequences derived from both the neural network models which was capable of generating real-time command structure to be sent to the unmanned aerial vehicle.

To evaluate the system, it was implemented on a real system using a Tello drone, for which the following additional design and implementation tasks had to be performed:

- Design and implementation of a command sender and receiver using the Tello SDK Tool to talk to the Unmanned Aerial Vehicle.

- Design and implementation of an algorithm to process data in real-time to produce a single data frequency for the model to interpret the data.
- Integration of all components to allow real time sensor fusion, classification and command generation in a single interface.
- Calibration of the Electromyography and Electroencephalography Sensors to produce the desired data stream using pre-defined Python Libraries, open-myo [9] and python-emotiv [10].
- Collection of activity data from the human controller in a live setting with sensor data streaming at different streaming frequencies.
- Pre-processing of the data stream for producing better input data for the neural network models.
- Training and evaluation of the neural network models using the collected data.

CHAPTER 2

RELATED WORKS

2.1 Gesture Detection using Electromyography

Controlling any semi-autonomous robot system using gestures is a commonly researched topic. But inducing the flexibility in the entire control mechanism is very important. The need to make it more usable and ready for people who were never exposed to this kind of technology should always be a primary thought. Using different classification strategies to incorporate this immediate thought was depicted in this former study by Joseph DelPreto and Daniela Rus at MIT [6]. They presented a form of clustering unlabeled streaming data performed in real-time. This was subsequently used to adaptively threshold muscle and motion signals acquired via electromyography (EMG) and an inertial measurement unit (IMU). They devised a plug-and-play algorithm for detecting arm stiffening, fist-clenching, rotation gestures, and forearm activation. They also successfully augmented a neural network pipeline trained on carefully and logically chosen data from users who used this control architecture in the past. This was essential to demarcate the left, right, up, and down gestures. In their terms, they generated a gesture vocabulary suitable for controlling a robot, which was a drone. They proposed an algorithm evaluated from data received from six subjects by ensuring a good classifier performance and interface efficacy. They were able to identify 97.6 % of 1200 cued gestures, and the drone was able to respond to a set of 1535 unstructured gestures with an accuracy of 81.6 %. Their flight time for the entire experiment was 119 minutes. As the work in this thesis also revolves around remotely controlling a drone using gestures generated from hand movements, we de-

ployed this central idea in our project to finally use it as our primary activity while recognizing distraction patterns. However, we combine it with additional sensors to address distractions and allow it to be moved to a less structured environment.

The motivation for using gestures for real-time classification purposes was also generated from a former study by Beau Crawford, Kai Miller, Pradeep Shenoy, and Rajesh Rao [11], where they proposed a technique to use the EMG Signals that were recorded from arm surfaces and in a completely non-invasive setting for controlling a robotic arm. This was mainly focused on advancing bioengineering for the sophistication of prosthetic devices for amputees and paralyzed individuals. They started their study by a physiologically informed selection of the forearm muscles for recording the Electromyography Signals. They considered the usage of intact muscles that amputees could exercise with varying degrees of control. They recorded the activation patterns from the muscles in the forearm and classified them to formulate a control strategy. They experimented with their work on different 3-dimensional goal-directed movements like obstacle avoidance and picking up or accurately placing objects in their proper place. They used a simple and sparse feature representation that can be computed in real-time and passed onto a well-designed linear Support Vector Machine to classify the data. They achieved a classification accuracy of over 90% with an 8-class classification problem. The sampling rate of the EMG Signals was 2048 Hz. As they classified the steady-state signals from the hand gestures, they had to select windows of EMG data from all channels. They had a sliding window of 128 samples for each channel and computed the amplitude of each channel over that window. The final feature vector would then consist of 7 values per channel. This classification method proved to have 16 commands/sec, allowing them for excellent control of the robotic arm. They aimed to work on discriminating the finger movements as a part

of their future study. They also thought to incorporate an EEG segment into their research to understand motor patterns.

The above studies concentrated on the electric signals extracted from the muscles, known as Electromyography. But the movement of the hand in the 3-dimensional space doesn't uniquely define a gesture as the activation on the muscles could be similar in nature and hence not be able to be classified as a unique sample. As the electric potentials might look similar due to the similar activation of muscles. Therefore, using an Inertial Measurement Unit that keeps a record of the had movement by throwing values from its accelerometer and gyroscope. A study using IMU information for gesture recognition was performed by Ari Y. Benbasat and Joseph A. Paradiso at the MIT Media Laboratory [12]. They describe an inertial gesture recognition framework that consists of three segments. The first was a wireless six-axis inertial measurement unit to fully capture the three-dimensional motion. The other was a gesture recognition algorithm used to analyze the data and categorize it on an axis-by-axis basis as simple motions that depicted magnitude and direction. Finally, an application designer would combine recognized gestures on a concurrent and consecutive basis to define a range of composite gestures which would trigger output routines. Their proposed system required a lot of custom hardware and software development for each application framework. The necessity of the system to be compact and wireless was huge, so they designed their hardware architecture very carefully. They divided the gesture framework into different components identifying them as atomic and activity gestures. The atomic gestures referred to a small action performed by the hand. And on the other hand, an activity gesture is any type of movement cause due to performing an activity. They researched goals that would conceptualize precise hand movements into a classification environment. They designed an algorithm that took advantage of a priori knowledge of human arm muscle

motion structure. 'Since the velocity of the arm is zero at the ends of the gesture, the integral of the acceleration across it must be zero as well' [12]. They only recorded the peak activity moments that would be significant in the classification problem. Later they used this framework to develop a software application that uses this principle to identify hand movements.

Another study on the comparison of control strategies was published, which focused on EMG controlled exoskeleton for the hand, by Mathew DiCicco, Lenny Lucas, and Yoky Matsuoka [13]. The main focus of this paper was to ensure a control mechanism to restore dexterity to paralyzed hands. This was performed by extracting the user's residual non-invasive electromyography signals of a quadriplegic subject. They validated that the EMG signals coming in from the ipsilateral biceps could be used to develop a reliable natural reaching and pinching algorithm. They had another segment to the project: a voice-activated sensing and control strategy. The voice triggers were verbal commands like 'grasp' and 'grip' to trigger the opening and closing of the actuated clasping mechanism at the location where the user has their hand. They proposed three different strategies for achieving their goal. One is a binary algorithm, the other is a variable control algorithm, and the final one is a natural reaching algorithm. They did not implement any learning algorithm but were actually able to achieve some excellent outcomes. The pinching motion that they targeted mainly was very effective for those who lack mobility regardless of the control algorithms used. The hardware was successful in being comfortable and had minimal materials on the palmer side of the hand and also never interfered with the manipulated objects weighing only 6.67 oz. Among their approaches, their binary algorithm had the fastest outcomes. However, much of their success had to do with the ability of the human controller to adapt to the behavior of the system and their ability to target specific muscles. In our application where we are aiming for intuitive

gestures to control an external UAS and where we are aiming at easy to deploy sensors, a robust and adaptive gesture classification system will be necessary as users should not modify the signals (and thus gestures) in order to improve recognition rates.

EMG-based robot control is a fascinating topic, and works related to this kind of control mechanism never lack efficiency. But only limited research focused on the time-varying EMG signal features. Artemiadis et al. have researched a new concept of focusing on the time-varying components of EMG signals. They have used these signals from muscles of the human upper limb and used them for the control interface between an anthropomorphic robot arm and the user. The proposed interface focuses on the time-based changes caused by muscle fatigue or adjustments of contraction levels. The data recorded from the muscle activity was then transformed into kinematic variables that were finally used to control the robot arm [14]. The user could be unaware or have never seen the robotic arm (as it was anthropomorphic). Natural arm motions were enough to control the robot arm with the interface mapping. According to a former study, the relationship between the EMG signals and arm motion is highly non-linear; to overcome this, they used a discrete form of control. All the former studies in this research never realized the time-varying component. Thus, they claimed to have performed a good experiment even if that was not investigated the implications in a real-life scenario until then. The time relative data was needed here not necessarily to capture movement but rather to address dynamic changes even in static settings as the muscle could be exposed to fatigue, changes in the level of force production, heat at the recording site, or smaller electrode movements relative to its initial position. The duration of the training phase was decided to be 4 min [14] as the user would prefer to rest his arm before the next training phase. They deployed four decoding models: a switching model, a stationary model, a linear filter, and finally, a

Support Vector Machine. The effectiveness of VM for CC_x was 0.82 ± 0.03, Cc_y was 0.84 ± 0.02 an 0.82 ± 0.05 for CC_z. The proposed method to compensate for EMG Changes through time is quite important for the field [14].

One of the former studies by Rong Song Di Ao on EMG-driven Hill-type movement performance of Human-Robot Cooperation Control and Propositional models for an ankle power-assist was an enlightening piece of work. . He was successful in designing an EMG-driven hill-type neuromusculoskeletal model (HNM) and a linear proportional model (LPM) that were calibrated through maximum isometric voluntary dorsiflexion (MIVD). The action of the two models used for control efficiently measured the real-time ankle joint torque, and the collaborative HNM model accurately accounted for changes in the joint angles and muscle dynamics. For this study, the author had eight volunteers who were asked to wear the ankle exoskeleton robot, and they completed a series of sinusoidal tracking tasks in the vertical plane based on the two calibrated models. The subjects were then asked to track the target displayed on the screen as perfectly as possible by performing ankle dorsiflexion and plantarflexion. He used two measurement techniques, root means square error (RMSE) and root means square jerk (RMSJ), in the assistant torque and kinematic signals to categorize the movement performance. Also, the amplitudes of the recorded EMG signals collected from the tibialis anterior and the gastrocnemius were obtained to show the muscular efforts. His results depicted a scenario where the increase of the assistant ratio would cause the decrease of any firm action or smoothness of tracking movement. This study mainly focused on the EMG signals that could be derived from the lower limbs for performing communication with a robotic exoskeleton fitted on the lower limb itself. The author designed a multi-channel EMG amplifier for performing the power-assist exoskeleton robot experiment. Also, the recorded end signals were amplified 5000 times, and a fourth-order Butterworth filter was used for

band filtering, which had a band of 10-400 Hz. During the experimentation, they also took care of the safety factor of the subjects that were involved in the experiment by implementing two mechanical stops, which, when applied to limit the rotational range of the motor and the software program, would stop the operations if an output torque exceeded the present range of negative 12 to 12 newtonmeters or the ankle joint angle exceeded a preset range of -35 to 30 deg [6]. They concluded by saying that HNM exhibits significantly less error than LPM at ankle joint angles of 60, 75, 105, and 120 deg.

The use of low dimensional embeddings for generating EMG- based control of a robot arm was another research performed by Artemiadis et al. [15]. They have used a mathematical model to decode upper limb motions from EMG recordings. Their proposed method uses a dimensionality reduction technique represented by muscle synergies and motion primitives. This method shows a 2D embedding of muscle activations that can be decoded to perform a continuous arm motion profile and described in a 3D cartesian space. The curse of dimensionality is a significant factor when dealing with vector data shown in a high-dimensional area. This could be overcome by efficiently reducing the dimension of the data. The data that they have collected shows a high dimensionality for the muscle activations and joint angles. They implemented the most commonly used dimension reduction technique known as principal component analysis to perform their research. They propose implementing this algorithm twice, once for finding a new representation of muscle activation data and then for the joint angles. Using the PCA algorithm would return a new coordinate system of lower dimensions using the first eigenvectors of the original data as axes. Their research was a pioneering study for a continuous profile of 3D-arm motion using only EMG signals. The previous works extracted only the discrete information about the arm motion while they would estimate the constant arm motion. However, they

are constrained to isometric movements with a single degree of freedom or subjected to very smooth moves. The methodologies that were being proposed in the paper would take little time to build the decoding model and have a very low computational load during the real-time operation [15].

2.2 Distraction Detection using Electroencephalography Signals

In a study based on distraction detection in motor rehabilitation using a high variable EEG sensor performed by Apicella et al. [16], a precise identification of distraction was proposed. The authors have used a wireless cap that had dry electrodes and a deficient number of channels and collected data sequences for experimental validation from 17 volunteers. They had a relatively elementary study based on supervised classifiers. As a part of the feature extraction, they implemented a 12 band-pass filter bank (FB) and a typical spatial pattern (CSP) algorithm. They had a segment that removed the artifacts from the EEG signal during their data collection and then passed it on to the filter bank. Their study implemented a set of five supervised classifiers, where they tuned different hyperparameters and variation ranges to derive their results. The classifiers used in their study were k-nearest neighbor (k-NN), a support vector machine (SVM), an artificial neural network with hidden layers ranging from 25 to 200 units, linear discriminant analysis (LDA) that had a gamma range from zero to one and a delta parameter. Their study implemented two domains of analysis; frequency domain and time domain and finally concluded that the results from the state-of-the-art classifiers had a mean accuracy of 92.8 ± 1.6 % and a mean recall of 92.6 % using the k-NN classifier.

Another research that was carried out to detect the temporal impact on cognitive distraction detection for car drivers using EEG was performed by Schneiders et al. [17]. Their research presented a self-designed model named DeCiDED, which

incorporates electroencephalography and machine learning techniques to detect cognitive distraction in car drivers. The results that they concluded with indicated that if the training and evaluation data had originated from the same driving session, their proposed models would have performed with the highest accuracy. In this paper, the authors focused on the temporal impact, which refers to the effect of the time interval between the collection of training and evaluation data on distraction detection accuracy. The authors conventionalized the data collection process by using the data collected from the same subject for the training and validation. This step would make the system reliable for individual participants even after tuning parameters separately for that specific individual. The sensors used to collect the EEG signals have a sampling rate of up to 200 Hz. The experimentation was done in a virtual setting where a real-time driving scenario was augmented for the driver. The temporal impact was identified from the training data collected on day one, and subsequently, the evaluation data that was collected after seven days. Once the data was collected, it went through segmentation, cleaning and extraction, selection. To identify the channels of EEG, they used a binary mode of classification with classes depicting distraction or focus in their study. Their algorithm correctly identified 76.77 % of all cognitive distraction samples, corresponding to about 5594 out of 7287 two-second time windows across all test subjects [17]. They concluded by saying that if the data was collected on the same day, it yielded better results than between days.

The primary motivation of this research is to detect the distraction from an attentive mind while controlling significant semi-autonomous robots. There was considerable research that was carried out on this specific area of distraction detection, most of which concentrates on the direct control of the vehicle, i.e., by being a part of the vehicle system. Kumar et al. [18] worked on the vulnerabilities of distraction during driving using Electroencephalogram (EEG). They stated that textit “Data from

10 different subjects were obtained and categorized into different frequency bands. Distractive driving is related with the Theta band, so the Theta frequency band was decomposed using Discrete Wavelet Transform (DWT), and 17 different features were extracted." – Kumar et al. [18]. Consequently, they used the Theta Band of the EEG Signal, which was then pre-processed, and simple feature extraction was employed. Finally, the EEG was classified as Visual and Cognitive Distraction. They had used a 21-channel EEG Extraction Device that fitted the head like a cap and data was collected under the supervision of a visual infrared camera interface. This was used to timestamp the signal that was collected. The product of the feature extraction was classified using SVM and KNN. The highest accuracy was recorded as 71.1 %. Their research was quite astounding, but the experimental setup was too organized. Along with the bulky design, it is hard to be replicated in a real-life scenario.

Another research carried out by Bajwa et al. [19] had the same perspective: identifying the distraction pattern, but a slightly different goal: to pave the way for future intelligent transportation systems. In this type of setting, the person interested in controlling the intelligent vehicle doesn't have to be a part of the vehicle system by being inside it. Their approach was slightly more conclusive as they included two phases of detection. The first one was almost similar to Kumar et al.[18], but the distraction recording was more specific. The drivers are asked to perform the following operations: read, text, call and use the phone camera in their first phase. The second phase was *"using a survey, comparing driver's order/perception of the severity of distraction with the derived distraction index from EEG bands."*- Bajwa. et al. [19]. In this experiment, they used 14 electrodes and recorded from a set of 13 subjects. But the best part was that they used on-the-road driving, which is always preferred to a virtual-reality driving simulator. The results were impressive as they detected distracted driving with 91.54 ± 5.23 % mean accuracy for the detection of

distraction and 76.99 ± 8.63 % while distinguishing the five distraction cases they considered. .

The two research works mentioned above employ more of a manual way of detecting the distraction where the classification was annotated by supervisors a priory. In contrast, one of the former research works by Vuckovic et al. [20] was more of an intuitive approach to classifying alertness and drowsiness. Their paper *"Automatic recognition of alertness and drowsiness from EEG by an artificial neural network"* mentioned the novel method of using interhemispheric and intrahemispheric cross-spectral densities of the full spectrum EEG as the input to an artificial neural network with the two outputs of alertness and drowsiness. They trained the ANN from a full-spectrum EEG extracted from 17 subjects. They employed multiple algorithms alongside the linear network, including an LVQ neural network. Their conclusion included a statement that LVQ was the best classification in comparison to any other ones they used. They used t-distribution to evaluate the similarity between the human assessment and the network output, 94.37 ± 1.95 %, which is the highest among the others . Their motivation was only to derive the classification and not trigger any other control interfaces from this kind of environment.

A similar kind of research was carried out by Abdulhamit Subasi [21], who analyzed the data using wavelet transformation. The main band was converted into multiple sub-bands using wavelet transformation. Finally, different statistical measures were extracted, acting as ANN input. The signals were taken from 12 subjects and the system could approximately classify them as alert, drowsy, and sleeping with 93.3 ± 4 % accuracy.

For the control interfaces research, some former work was being carried out for EEG Signal classification. Nagabushanam et al. [22] classified the EEG Signals using LSTM and other neural networks. This study was mainly concentrated on finding a

neural network that could be used to classify the EEG signal. They primarily used SVM and Logistic Regression. Then they added a two-layer LSTM and four-layered improved Neural Network to improve the classification performance. While all the other researchers used the signals with their frequency components, This specific research concentrated on the 2D Grayscale Imagerepresentation developed from an EEG PSD which was first seen by Leracitano et al. [23]. Describing their work they state: *"To improve the performance of NN architecture, a four-layer improved NN and LSTM are designed. Both are implemented using Python, and comparative analysis is carried out with statistical features. LSTM is a long short-term memory approach which removes overfitting problems caused in basic RNN structure of deep learning classification algorithm."* - Nagabushanam et al. [22]. They concluded that the LSTM and improved NN boosted the performance by a considerable margin to 71.3 % and 78.9 %, respectively. In future work, they proposed a multi-stage LSTM architecture and pooling layers in the proposed NN to increase the performance of deep learning algorithms for EEG signal classification.

Fourier Transform was also a part of the classification of EEG, which was introduced by Cecotti et al. [24]. The research switched from a time-domain to a frequency domain analysis. They divided it into steps, stating: *"The first step allows the creation of different channels. The second step is dedicated to transforming the signal in the frequency domain. The last step is classification. It uses a hybrid rejection strategy that uses a junk class for the mental transition states and thresholds for the confidence values."* - Cecotti et al. [24]. They concluded with results with six electrodes on two subjects for 1s. The system was reliable to almost 95 % with a rejection criteria. As the system must be very reliable, they combined two decisions for the rejection. First, they dedicated one class for the transition states. Then there are two thresholds for each category. They are determined function for the validation

database. The learning, validation, and test database comprises 6392, 2130, and 2134 patterns for each subject. The classes are equally distributed in the three databases. It was an outstanding work with a minimum constraint on statistical measures.

Another EEG-based motor imagery classification using a support vector machine and multi-layered perceptron was performed by Chatterjee et al. [25]. They focused on the category of motor imagery of the left-right hand movements of a healthy subject. They incorporated the elliptic bandpass filter to discard the unwanted segments of the signal. For their part of the research, C3 and C4 electrodes in the 10:20 EEG system were used mainly for left-right limb movements. They performed the feature extraction based on the wavelet, band power, and average power. However, their research was based on an experimental data set from BCI competition II 2003 provided by the Department of Medical Informatics Institute for Biomedical Engineering, University of Technology, Graz [26]. The data set for this research was unique to the other researchers in this area as they used a predefined data set collected from a female 25 years old. They have variable feature vectors generated from the different feature extraction processes, resulting in a vector size of 140 x 26 to do the classification. In the end, the classifiers were able to produce an output with the accuracy of 85% with the SVM and 85.71% with an MLP. They concluded by saying the energy entropy feature set gives the most promising results for both the classifiers[25].

Most of the researchers on EEG classification mentioned above were to identify the distraction from the EEG signals. But the following research performed by Peng et al. [27] Depicted the attentiveness recognition system using Hilbert-Huang Transform (HHT) and Support Vector Machine. This research did not focus on the subject's specific type of task. Instead, it proposed an algorithm to detect the attentiveness or the mental state of concentration of humans performing different tasks. The study

mentioned the use of HHT in the analysis of nonlinear or nonstationary bio-signals, including brain waves. So they used a single-channel EEG signal from the frontal area acquired from participants at different levels of attentiveness and then decomposed it into a set of intrinsic mode functions by using the empirical mode decomposition. Later, the Hilbert transform was applied for each intrinsic mode function to obtain the marginal frequency spectrum. Finally, the band powers and the spectral entropies were chosen for the two-category classification task performed using the Support Vector Machine. Their proposed algorithm used segments of alpha and beta waves and their spectral entropies, which exceptionally classified between attentiveness and relaxed states with the classification accuracy of 84.80 %.

2.3 Anomaly Detection from Electroencephalography Data

Anomaly detection is an unsupervised learning method that could be used on time varied signals to identify any anomaly from the normal state of behavior. Electroencephalography being a time-varying signal, could be sent to an anomaly detector to identify specific changes in the signal's amplitude. Chen et al. [28] performed anomaly detection on EEG data to investigate the impacts of different similarity metrics on anomaly EEG detection. The author considered a few potentially available metrics for the EEG analysis. In this research, two indicators were used to evaluate the performance after the detection algorithm. The first one reflects the measured similarity between the two compared EEG signals, and the other quantifies the detection accuracy [28]. The experiment was performed on two datasets depicting normal and abnormal movements. The positive impacts of different similarity metrics on an abnormal EEG detection were demonstrated in the results. Specifically, the Hellinger distance and the Bhattacharya distance metrics showed excellent performance over the accuracy of 91.67 % on the live collected data set. As a part of the future work,

they mentioned combining both the distance metrics for the similarity measure of EEG signals.

2.4 Some more highlights on gesture control

The gesture detection and identification research can also be performed using motion controllers. In the study, Sarkar et al. [29] used a Leap Motion Controller to recognize the gestures using advanced algorithms for such operations. This device uses two monochromatic infrared cameras and three infrared LEDs to track any hand moved up to about 1 meter or about three feet directly above it [29]. As a result, the gestures performed on top of the Leap Motion can be recorded using the infrared cameras built inside it and later be used to send to the ground station. They used the leap SDK on the ground station, which would translate the gestures and send them to a ROS topic. This would send the commands to the subscriber method and later relays them onto the twist method. Finally, the twist method interprets the forwarded message and then commands the Drone to perform the appropriate action based on the hand gesture [29].

There are other methods to identify gestures and use them as a control for a designed environment. Accelerometer-based gesture control is one such supplementary interaction modality. Kela et al. [30] perform two user studies in their research. The first study was to find gestures for controlling a design environment like a TV, VCR, or lighting. This is where they identified that different people usually prefer other gestures for the same task, and hence the algorithm needs to personalize them. The second part of the study evaluated the usefulness of any gesture modality compared to any other interactional modality for controlling a design environment. In this scenario, they considered RFID-based physical, tangible objects, laser track pens, or a PDA stylus. Finally, they concluded that gesture commands are natural,

especially for commands with a spatial association to any design environment control. According to their studies, any user being subjected to gesture control would prefer intuitive gestures that are more readily understandable and interactable. They used a discrete Hidden Markov Model to recognize gestures. The model successfully returned typically high accuracy of recognition with most of the recognition rate versus the number of training vectors above 90 %. Their research not only focused on recognizing gestures but also experimented with the usability of gestures over any other control strategies for designing a controlled environment.

CHAPTER 3

UNDERSTANDING THE SIGNALS

3.1 Electromyography

Electromyography (EMG) is a way to recognize the electrical activity produced by the skeletal muscles due to the electrical or neurological activation of muscle cells generating electric potential. This methodology is often used to evaluate and record muscle activity. The first documented experiments dealing with EMG started with Francesco Redi in 1666 [31]. In medical sciences, electromyography is used to diagnose muscles' health and the nerve cells responsible for controlling them. This technology can identify any nervous dysfunction, muscle dysfunction, or problem triggered by nerve-muscle signal transmission. There are two types of Electromyography: Intramuscular EMG and Surface EMG. The Invasive EMG uses needles to prick inside the muscles with an electrode attached to the end of the needle. These receive the electrical signals as electrode potentials, which can later be realized in a 2-Dimensional Time Distributed Sequential Graph. The Non-Invasive EMG, often known as Surface Electromyography or sEMG, is a non-medical technique that recognizes the surface activation of that muscle. This type of technology is often used for non-medical purposes by several bioengineers. In the field of Computing, researchers have often used this kind of technology to set up a middleware in indemnifying hand movements and map it to a structure of gestures for any Human-Robot or Human-Computer Collaboration Interfacing [31].

The electrical source is the muscle membrane potential of about -70mV. Measured EMG potentials range between less than 50 μV and up to 20 to 30 mV, depending on

the observed muscle. The typical repetition rate of muscle unit firing is about 7–20 Hz, depending on the size of the muscle (eye muscles versus seat (gluteal) muscles), previous axonal damage, and other factors [32]. EMG signals are composed of superimposed motor unit action potentials (MUAPs) from several motor units [31].

EMG signals have been targeted as a control for flight systems. The Human Senses Group at the NASA Ames Research Center at Moffett Field, CA, seeks to advance man-machine interfaces by directly connecting a person to a computer [31]. In this research work, we have implemented the non-invasive technique of recording the electrical signals from muscles to recognize the gestures. EMG can be obtained from any part of the body. Still, for this part of our research, as we are more concentrated on the hand movement, we will be receiving the EMG Signals from the forearm as this area extends to most of the muscles connected to the fingers and is ideal for gesture recognition.

3.1.1 Muscle Anatomy

There are over 30 muscles in the hand [33] where most of the movements of the hands use the muscles in the forearm, which are connected to thin tendons. The placement of the forces that relate to the finger movements is generally distant from the forearm surface skin, relatively more than that of the hand movement [34]. And that is usually the reason for the sEMG signals containing lower amplitude or a large amount of noise for finger movements than those responsible for arm movement, caused due to the attenuation and filtering of the forearm signals [35]. This is the primary reason for using more electrodes which would help to determine the finger movements [36].

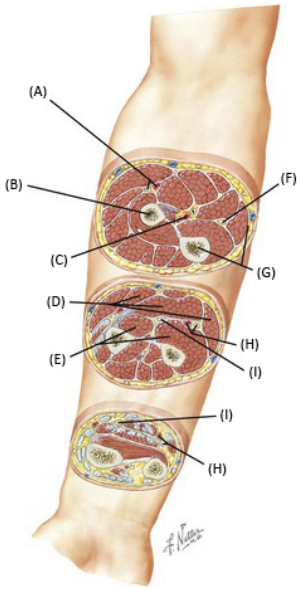


Figure 3.1: Forearm: Serial Cross Sections [37].
(A) Radial artery and superficial branch of radial neuron,(B) Radius, (C) Ulnar artery and median neuron, (D) Superficial flexor muscles (E) Deep flexor muscle, (F) Ulnar neuron, (G) Ulna and antebrachial facia, (H) Ulnar artery and neuron, (I) Median neuron.

The above figure shows the different parts of the forearm and the muscles that an sEMG sensor would be able to read at that position. As we go down the forearm, the number of muscle data we can read using sEMG goes down. Let us divide the three exposed parts in the above figure into Part A, Part B, and Part C.

The muscles that an sEMG sensor would be able to read in Part A are:

- Radial Artery and Superficial Branch of Radial Nerve.
- Radius.
- Ulnar Artery and Median Nerve.
- Ulnar Nerve
- Ulna and Antebrachial Fascia

In Part B:

- Superficial Flexor Muscles.

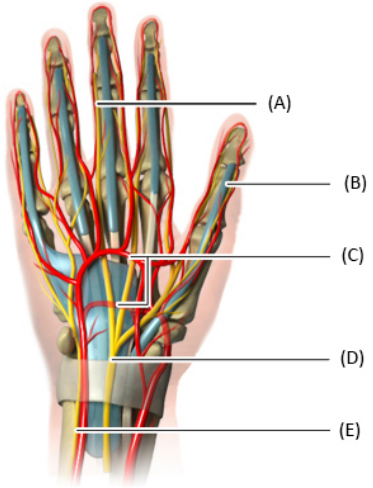


Figure 3.2: Location of the main nerves and blood vessels in hand [38].
(A) Blood vessel and nerve cord of the fingers,(B) Thumb tendon sheath, (C) Double loop of the hand's two arteries, (D) Median nerve (nervus medianus), (E) Ulnar nerve (nervus ulnaris).

- Ulnar Artery and Ulnar Nerve.
- Deep Flexor Muscles.
- Median Nerve.

In Part C:

- Median Nerve.
- Ulnar Artery and Ulnar Nerve.

Also, there are short muscles that are essential for any gesture recognition. They usually lie between the individual metacarpal bones of the hand. Not only do they allow us to spread our fingers, but also pull them back together. In addition, they help us to bend and stretch our fingers too.

Muscle tendons, nerves, and the other blood vessels running from the forearm to the hand pass through a tunnel-like passageway on the palm side of the wrist. This is known as the carpal tunnel.

We can see from the figure above that the activity recorded in the upper forearm encompasses most of the triggers that are received from the hand. This conceptualizes the placement of the Electromyography Sensor that we will discuss later.

3.2 Electroencephalography

Electroencephalography (EEG) is a technique to recognize the electrical activity of the human brain. This is performed by placing electrodes or electro-conductors on the human scalp. This usually records the miniscule electrical activity on the surface layer of the brain underneath the scalp [39]. In 1835, Richard Caton presented his findings on the electrical phenomena of the exposed cerebral hemispheres of rabbits and monkeys for the first time. Since then, electroencephalography has been used in different fields for various purposes. In Medical Sciences, EEG is mainly used to identify and diagnose epilepsy, sleep disorders, depth of anesthesia, the state of coma, brain death, etc. Apart from that, EEG is the first-line method of diagnosis for tumors, stroke, and other focal brain disorders [39].

EEG measures voltage fluctuations resulting from ionic current within the brain's neurons [39]. This data can be recorded in different ways. They are routine EEG (collected while asking you to rest quietly), sleep EEG (collected while you're asleep), ambulatory EEG (accumulated over a period of one or more days), video telemetry (the user is filmed while recording the EEG signals) and invasive electrocorticography (the electrodes are inserted in the brain). We have tried to record the data following the video telemetry for our research study.

The usual scalp EEG is performed using scalp electrodes that have conducting material used to detect the electrical activity in the brain. The placement of the electrodes is critical as they would be considered to aid the nearby electrode to produce the potential difference between them, finally giving the EEG montages. The locations and names of the electrodes are specifically conventionalized in this research following the International 10-20 system (an example of this system is given below in Figure 3.3).

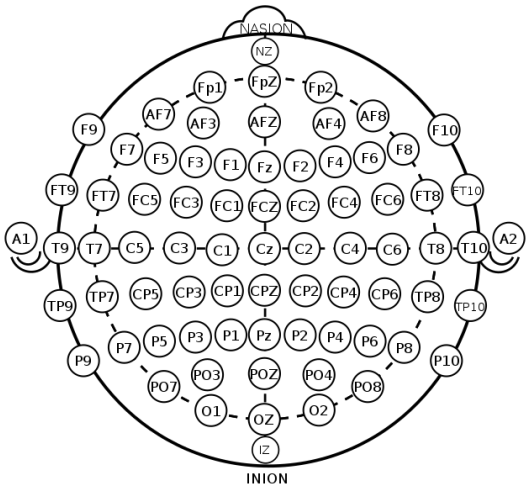


Figure 3.3: International 10-20 System to identify EEG electrode location and names [40].

The 10 and 20 in the system refer to the actual distance between adjacent electrodes which are either 10 % or 20 % of the total front to back or the left to right measurement of the skull. The electrodes are named according to their placement on the scalp and its closest lobe on the brain's surface. The following are the conventions for the naming of the electrodes:

1. Fp: Pre-Frontal Cortex
2. F: Frontal Lobe
3. T: Temporal Lobe
4. P: Parietal Lobe
5. O: Occipital Lobe
6. C: Central Lobe
7. Z: Refers to Zero or the Midline Sagittal Plane of the Scalp

The numbering system of the electrodes is unique to the side of the brain they are placed. The even number of electrodes is placed on the brain's right side, starting

from 2 in the front and running until the back. Likewise, the left side electrodes are given odd numbers starting from 1.

Since an EEG voltage signal represents a difference between the voltages at two electrodes, the display of the EEG for the reading recorder may be set up in several ways. The representation of the EEG channels is referred to as a montage [39]. The potential difference is created by the difference in the potential of the gyrus and a sulcus. This is better described in the image given below in Figure 3.4. We can see that the electrode is placed on the area that corresponds to b-c-d on the brain surface, b and d are the points on the gyrus, and c is a point on the sulcus. This is how we can find the potential difference between the two points.

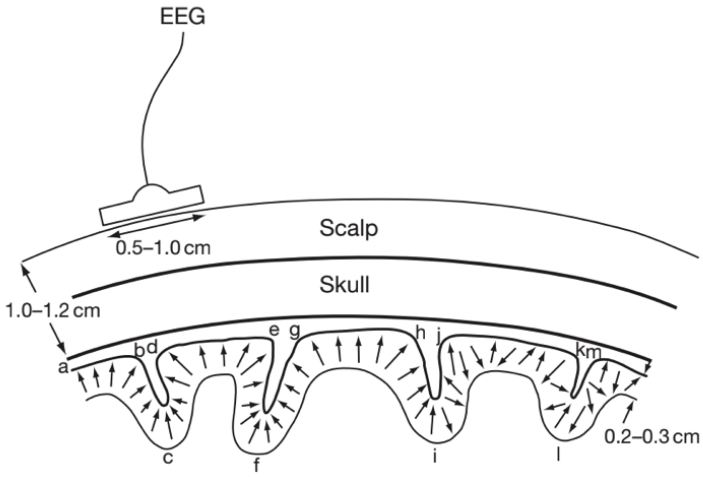


Figure 3.4: Source of EEG [41]

There are multiple bands of EEG that represent different types of activity in the brain. The various bands are demonstrated in Figure 3.5 and Tale 4.2 given below.

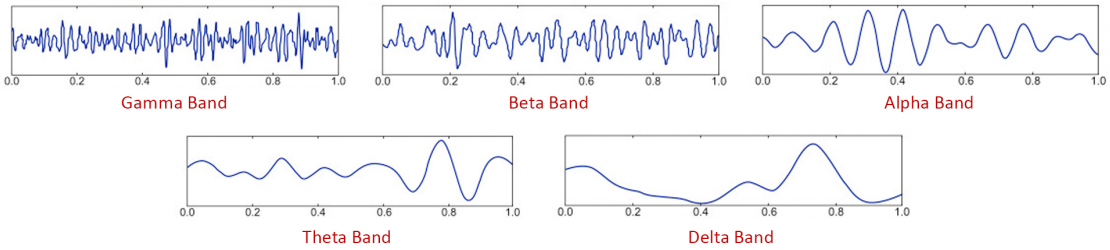


Figure 3.5: Different EEG Band Waves [42]

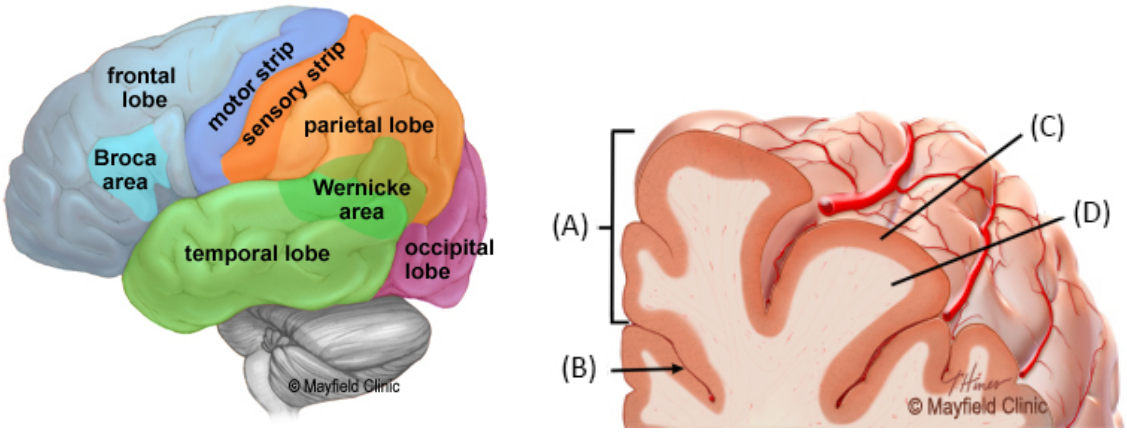
EEG Band	Range of Frequency	Corresponding Activity
Delta Band	0.1 Hz – 4 Hz	Deep, Dreamless, non-REM sleep, unconscious
Theta Band	4 Hz – 7.5 Hz	Light Sleep, Deep Meditation, Creative, Recall, Fantasy
Alpha Band	7.5 Hz – 12 Hz	Relaxed, Light Meditation, Creative, Super Learning, Conscious
Beta Band	12 Hz – 30 Hz	Normal waking state, Concentration, Focus, Five Physical Sense, Integrated
Gamma Band	30 Hz – 100+ Hz	Motor Functions, Higher Mental

Table 3.1: Description of different EEG Bands

From this table we can identify that for the scope of this research work, we have to concentrate on the regions of the Gamma Band as it mainly depicts a motor movement or a Higher Mental Activity.

3.2.1 Brain Anatomy

The brain's cerebrum is the most significant part of the brain that initiates and coordinates movements and regulates temperature. This region is responsible for enabling speech, judgment, thinking and reasoning, problem-solving, emotions, and learning [44]. This is the area that controls the activity of a human, and as of our research, we are primarily interested in a person's brain activity during any specific task.



(i) (ii)

Figure 3.6: (i)Lobes of the Brain, (ii) Cortex Structure [43]
(A) Gyrus, (B) Sulcus, (C) Grey matter, (D) White matter

In Figure 3.6, given above, we see how the cerebrum is divided into four lobes that perform different functions. The functions of the different lobes are listed below [44]:

1. Frontal Lobe: The lobe that is of our interest. This is the largest division of the cerebrum. The main functions of this lobe are personality characteristics, decision-making, and movement. Also, the front part of the frontal lobe is specifically for identifying the concentration, attention, and distraction in the brain.
2. Parietal Lobe: The middle of the brain helps a person recognize objects and identify spatial relationships. This also interprets the pain and touch of the body. It also helps the brain understand spoken language.
3. Occipital Lobe: The occipital lobe is responsible for our vision.
4. Temporal Lobe: This lobe takes care of our short-term memory, speech, music understanding, and smell.

From the understanding of the brain anatomy, we can identify that the prefrontal cortex is the area that we are interested in for the research-oriented goal of this project.

3.3 Inertial Measurement Unit

The Inertial Measurement Unit (IMU) is an electronic device that measures the angular rate and orientation of the body it is attached to. It uses the combined response from a gyroscope, accelerometer, and sometimes magnetometers. The working of the IMU is organized by detecting the linear acceleration using accelerometers and the rotational rate from the gyroscope. The use of IMU is typically in a Navigation System of a Vehicle that has a current position, initial heading, and destination. Below is a schematic interpretation of an Apollo IMU which identifies different parts inside it [45].

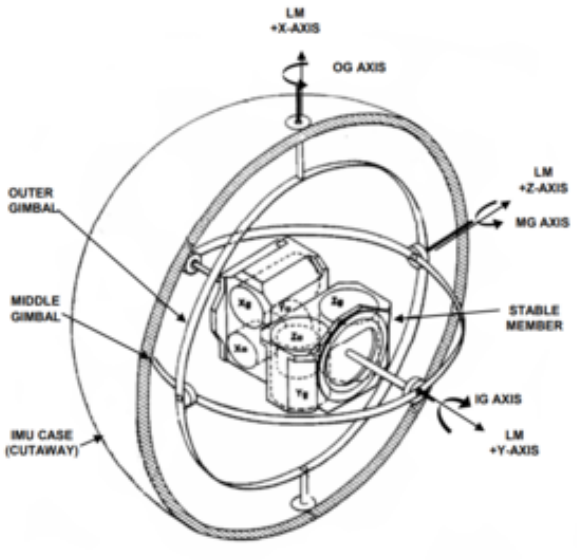


Figure 3.7: Apollo Inertial Measurement Unit [46]

3.3.1 Accelerometer

As an accelerometer usually measures inertial acceleration. It also helps to recognize the change in velocity over time. There are different accelerometers, including mechanical accelerometers, quartz accelerometers, and MEMS accelerometers. Usually, when an accelerometer is projected for any linear change in velocity along the axis of sensitivity, the proof mass shifts to a side with the amount of deflection proportional to the acceleration [47].

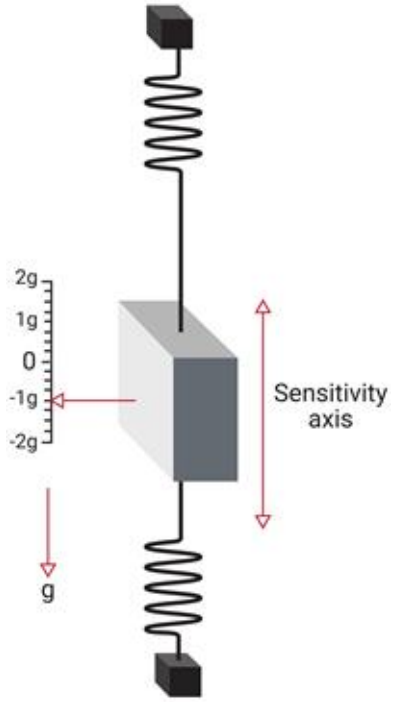


Figure 3.8: Simple Accelerometer Model [48]

3.3.2 Gyroscope

The gyroscope is responsible for measuring the angular rate of an object concerning the inertial reference frame. There are different gyroscopes: mechanical gyroscopes, fiber-optic gyroscopes, ring laser gyroscopes, and quartz/MEMS gyroscopes

[47]. The stability of the gyroscopes usually ranges from 0.0001°/hour to 1°/hour.

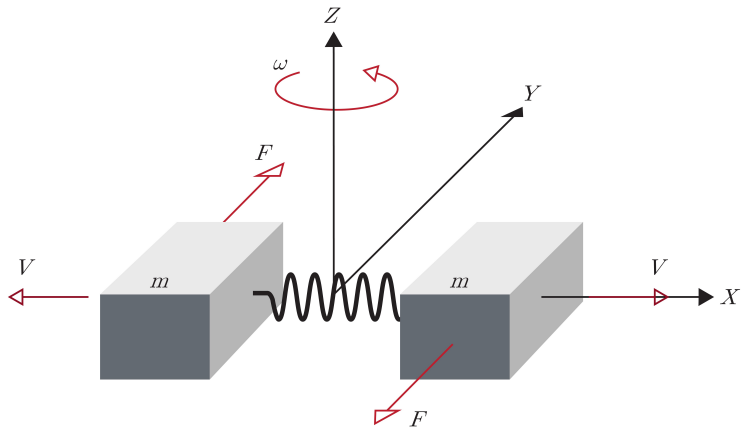


Figure 3.9: Simple Gyroscope – Tuning Fork Configuration [49]

As an individual inertial sensor will only be able to sense a component along a single axis, there have to be three different inertial sensors mounted in an orthogonal cluster to recognize the measurement along with the three-dimensional environment. Thus, 3-axis-accelerometer and a 3-axis gyroscope will finally produce a variance of raw or filtered body-frame acceleration and angular rate.

3.3.3 Quaternions for Orientation

Quaternions for Orientation is a unique mathematical concept to visualize orientation and altitude. This comprises a four-element vector that defines the orientation of a body. This vector is an amalgamation of a scalar element and a 3-element unit vector. The scalar value, here (w), has a direct correspondence with the angle of rotation. And the 3-element vector, here (x, y, z), actually defines the axis of rotation

about which the angle of rotation is performed [50].

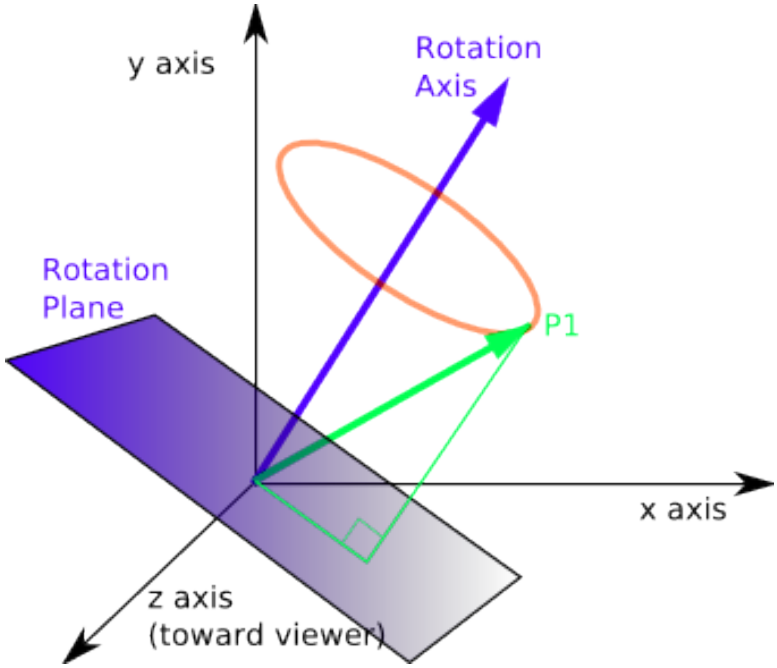


Figure 3.10: Quaternion [51]

A quaternion can be described by [52]:

$$q = [q_0 + q_1 + q_2 + q_3]^T = \begin{bmatrix} q_w & q_x & q_y & q_z \end{bmatrix}^T \tag{3.1}$$

And we can construct a quaternion by [50]:

$$\begin{gathered} n = |V \times V| = |V| \left|V'\right| \sin\theta \\ \theta = \arcsin\left(\frac{n}{|V|\,|V'|}\right) \end{gathered} \tag{3.2}$$

$$\begin{gathered} q_w = \cos\cos\frac{\theta}{2} \quad q_x = \sin\sin\frac{\theta}{2} \times nx \quad q_y = \sin\sin\frac{\theta}{2} \times ny \\ q_z = \sin\sin\frac{\theta}{2} \times nz \end{gathered} \tag{3.3}$$

CHAPTER 4

SYSTEM ARCHITECTURE AND CONTROL MECHANISM

4.1 System Architecture and Approach

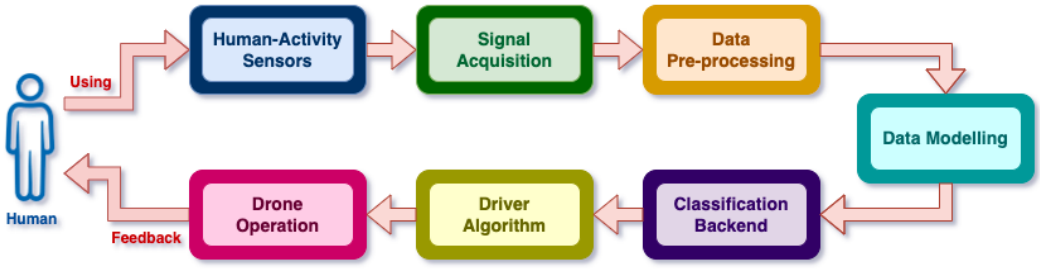


Figure 4.1: System Overview

In Figure 4.1 given above, we show the system overview of our research. The approach of this project was to identify the distraction through brain-sensing information and develop a control mechanism that could drive a drone using the help of hand gestures during a non-distracted brain state. To design such an approach, we had to consider several factors in each stage for a robust classification and good control mechanism.

Firstly, we extracted electrical signals from the brain and the forearm muscles using high-performance human activity sensors. To ease the data extraction and pre-processing, we used pre-defined python libraries [9, 10] that processed the signals removing noise and filtering it to a certain constant frequency. But, due to the bursting of the data signals, the signals received were stochastic in nature. Also, the two different sensors streamed data at different frequencies. Therefore, simple real-time signal merging algorithms couldn't be used to merge the incoming data.

To facilitate this issue, a server-client architecture was designed. The function of the server was to mimic the data extraction from the sensors by receiving the data from the sensors and reorganizing the timestamps of the received signals so that they could be re-streamed to the client. Before streaming the data to the client, each data sample was normalized from 0 to 1 and a new data packet was put together to be finally sent to the client.

Inside the client interface, the data is ordered by placing one muscle data for every brain data. Another, stream of data was prepared for the distraction classifier which had only the processed brain signals. Later, the sampled data was taken to process for the classifier. We used two different classifiers for our problem. The first one we used was a Long-Short Term Memory Recurrent Neural Network that classified the sequences of gestures. The second one we used was an Anomaly Detector using Autoencoders to classify the distraction in the brain.

For the LSTM Network, we prepared sequences of data packets that contained 50 combined signals. To do that a sliding window algorithm was used to glide over the original sample size and prepare a chunk of 50 samples after every 10 samples ensuring an overlap of 80%.

For the Anomaly Detector, we received brain signals through the client and recorded them. We labeled the data based on the state of the mind, namely distracted or non-distracted. Finally, an autoencoder made of dense layer networks was used to learn the non-distracted signals and later used to test on the distracted segments.

Finally, a driver algorithm was used to receive the outputs from the classifiers and generate control statements to be sent to the drone using a UDP connection. The logic was implemented in order to check the state of the mind and the subsequent gesture. A reverse algorithm was designed to check if there were any unknown gestures, even with a non-distracted mind. For a given scenario, at any point, if the algorithm

returned an unknown gesture or distracted mind, an altitude hold command would be sent to the drone to avoid misoperation.

To confirm the execution of the command, the drone has a feedback operation. It sends an 'ok' after the execution of the sent command. Along with that, the video output of the drone was received for the user to follow and control the drone.

Individual stages of operation are described below:

- Human: The person who is responsible to control the drone using gestures. The sensors in the next stage will detect the data from this individual.
- Human-Activity Sensors: Incorporates two human activity sensing technologies, Electroencephalography Sensor and Electromyography Sensor having an Inertial Measurement Unit.

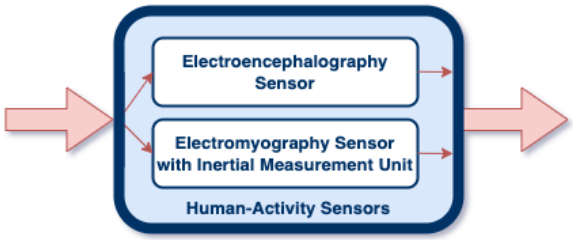


Figure 4.2: Human Activity Sensor Phase

- Signal Acquisition: Includes two individual Bluetooth Data Exchange environments to receive the signals from the sensors.

Figure 4.3: Signal Acquisition Phase

- Data Pre-processing: In this phase, the data was synchronized for modeling as well as normalized.

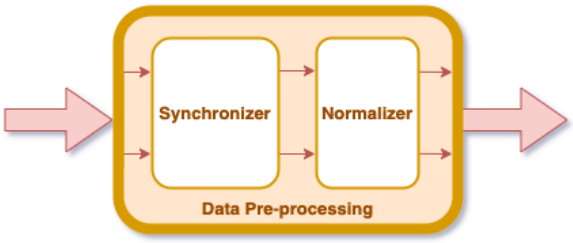


Figure 4.4: Data Pre-processing Phase

- Data Modeling: This phase contains the algorithm that orders the processed data samples and prepares them for the classifier.

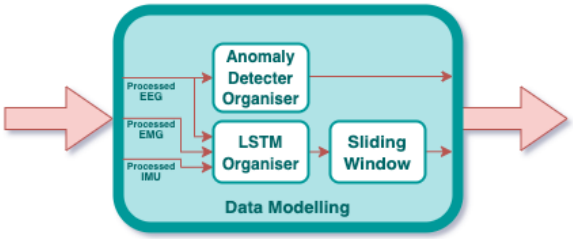


Figure 4.5: Data Modelling Phase

- Classification Backend: In this stage, the two data streams were sent to the two classifiers. The output was cleaned and prepared for the driver algorithm.

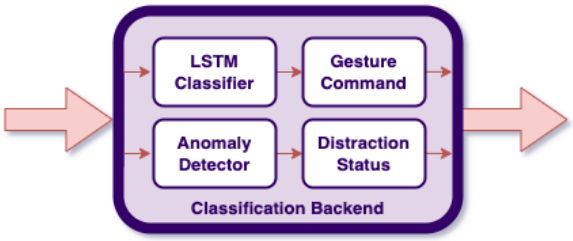


Figure 4.6: Classification Backend Phase

- Driver Algorithm: A driver algorithm was implemented in this stage that received output from the classification backend and processed to generate commands for the drone.

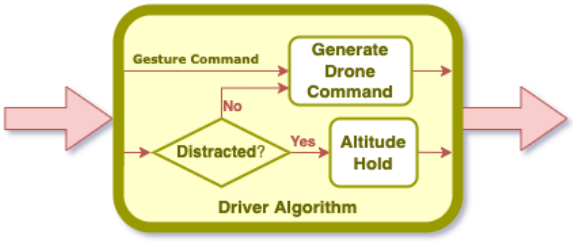


Figure 4.7: Driver Algorithm Phase

- Drone Operation: In this stage, we incorporate a Drone Control SDK that helps us to communicate with our Drone using UDP Communication. After the command is executed, the feedback is available for the human controlling the drone through real visuals available in closed environments and video outputs available in outdoor settings.

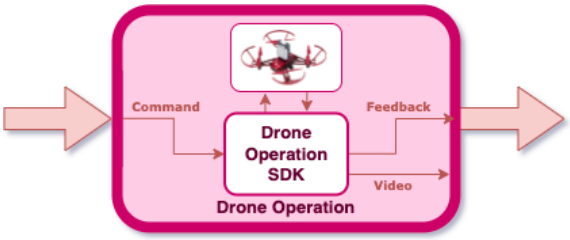


Figure 4.8: Drone Operation Phase

4.2 Understanding the Gesture Vocabulary

The primary control structure of our research work is based mainly on the gesture control module, as the drone receives the processed output of the classifier and

then translates them into the commands it needs to operate remotely. Therefore, to make the system more robust and user-friendly, we designed a set of intuitive gestures for the users to control the UAV. The gestures are detected using the wearable muscle interface and the in-built motion sensors. We have defined the basic structure of the gesture vocabulary and expect to extend it to the user's comfort after user research.

We have tried to discretize the gestures evenly so that they do not interfere with each other based on their motion and orientation, as the wearable device would receive some deceiving inputs that might later cause the classifiers to behave abnormally. All of our gestures are contained using actions involving the hand and the forearm movement. This makes it easier to detect as the wearable device we are using for the recognition is a forearm band and will be able to mainly detect the movements and pose differences in that area and below. The critical factor in this gesture vocabulary design is the reference frame. When we control a drone using a remote-control transmitter having a joystick interface, the control is sent to the drone with reference to itself. That is, we are aware that the command that the drone will reflect is with respect to the direction the front of the drone is facing. But during gesture control control commands usually have to be interpreted relative to the controller's frame. The reason for this, besides it being more intuitive to the operator, is that if a vehicle-centric frame were chosen, if the drone faces toward the controller, it will be difficult for the brain to identify the moving instruction, which might cause a distraction and thus make control ver difficult. Due to this, we have considered that the drone will, by convention, face towards the direction the controller is facing. If the drone, due to a rotation command, turns towards the controller, the video output from the drone will help the controller not get distracted by the directional ambiguity.

To discretize and formulate the gesture definitions, we first recognized the drone controls and later mapped all the gestures individually to the controls to mimic con-

ventional remote-control functions but more flexible and user-oriented. The gestures are designed so that it would be helpful to interface with any other navigation-related application or multimedia, but we are restricting the application to co-ordinate with our integrated system primarily. We have defined the gesture vocabulary for the common understanding below:

Take off / Land: Marked by pointing the hand orthogonally to the body posture and stretching the fingers. This is a toggle command used to start and stop the drone. This is a non-frequent gesture usually performed at the beginning and the end of the trial. It is detected by stretching arm muscles due to the flexing of fingers. It is one of the three commands in our gesture vocabulary that doesn't depend on the sequential operation of the gesture. That means we do not perform it in a sequence of tasks. The state of the muscle is constant throughout the gesture execution. A visual representation of the gesture is given below.

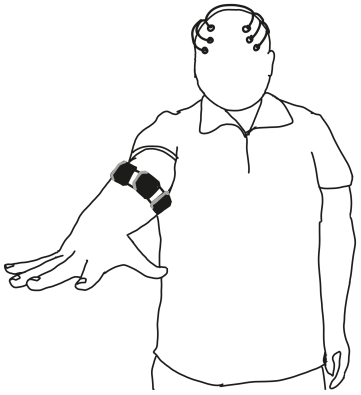


Figure 4.9: Take off / Land

Hover: Marked by pointing the hand to the body posture orthogonally without stretching the fingers. This gesture is relatively similar to the one above. It just differs by not extending the hand. It is recognized by both the orientation, altitude, muscle

relaxation, and no acceleration. It is another command that doesn't follow a sequence of operations. The state of the muscle and the armband remains constant throughout the gesture performance.

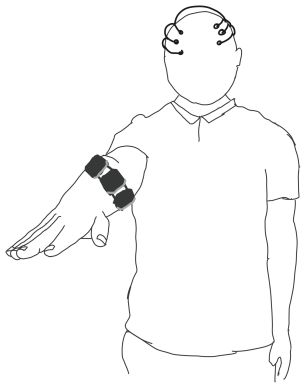


Figure 4.10: Hover

Relax: Marked by putting down the hand, straight and relaxed. This gesture is the last non-sequential gesture. The value registered by the wearable device doesn't change over time. Also, this is linked to the hover command for the drone. This was initially designed to identify any condition when the operator has muscle fatigue and requires resting his hand before the next instruction to be sent to the drone.

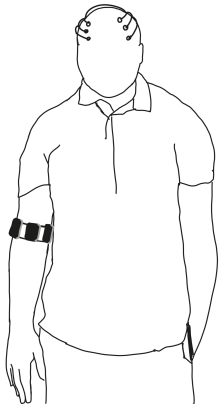


Figure 4.11: Relax

Go Left: Marked by moving the hand and forearm from right to left, as shown in the figure below. The plane of the palm should be facing towards the left side of the person. This is one of the sequential gestures that require the Inertial Measurement Unit to record the acceleration, orientation, and other positional data to recognize the gesture. The data is recorded, keeping into consideration the range of hand movement that is possible under different conditions. Therefore, the user will be able to move the hand to any plausible point toward the left, and the algorithm will still be able to recognize the gesture. The speed of the hand movement is not mapped to the drone movement. Therefore, to keep the drone moving left, we do not perform any other gesture after performing Go Left. To stop the drone from moving left, we make the hover gesture.

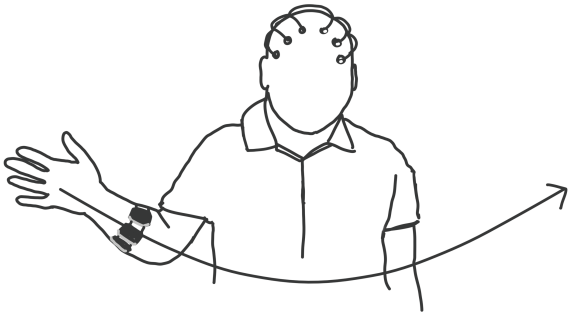


Figure 4.12: Go Left

Go Right: Marked by moving the hand and the forearm from left to right with the palm facing the right. This is another sequential gesture that takes the IMU to record its data. This gesture works precisely opposite to the Go Left Command, and hence it is designed to act as the opposite of the Go Left command. Also, the transitions are essential, and the description is given below.

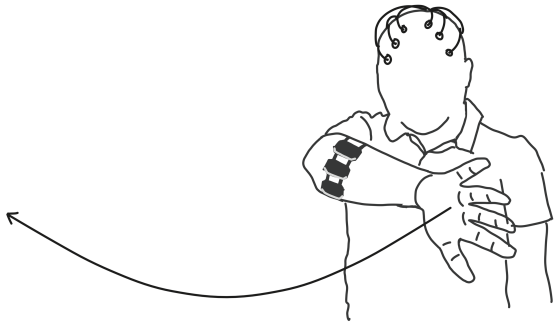


Figure 4.13: Go Right

Go Up: Marked by moving the hand and the forearm from bottom to up with the palm facing up. This is mapped to attain a higher altitude for the drone. It goes up whenever this gesture is performed. This is a sequential gesture.

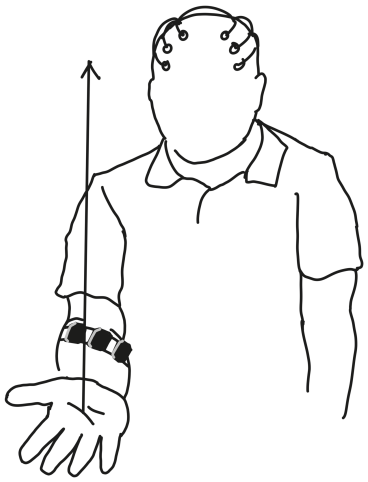


Figure 4.14: Go Up

Go Down: Marked by moving the hand and the forearm from up toward the ground with the palm facing down. This is mapped to lower the flying altitude of the drone. It goes down whenever this gesture is performed. This is also a sequential gesture.

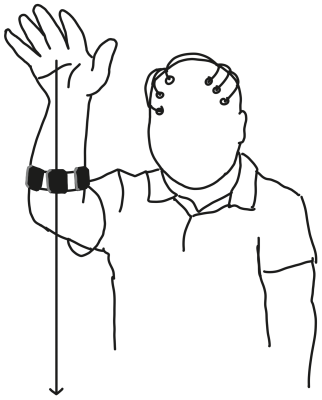


Figure 4.15: Go Down

Forward: Marked by moving the hand and the forearm with the palm of the hand facing outward and pushing it forward away from the body. This is mapped to going forward from the current position straight for the drone. This is also a sequential gesture.

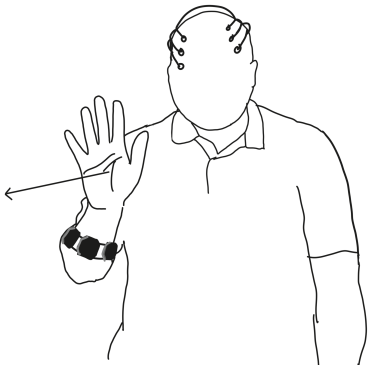


Figure 4.16: Forward

Backward: Marked by moving the hand and the forearm facing inward towards the controller and pulling it back towards the body. This is mapped to coming back from the current position straight. This is also a sequential gesture.

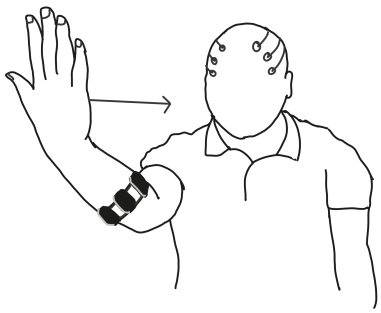


Figure 4.17: Backward

Rotate Clockwise: Marked by holding the hand orthogonal to the body with a thumbs up and turning the hand clockwise 90°. This is mapped to the rotation of the drone towards the right. It rotates clockwise whenever this gesture is performed. This is also a sequential gesture.

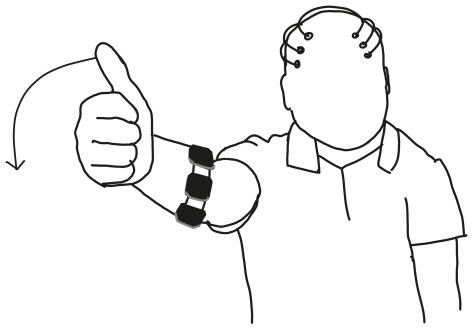


Figure 4.18: Rotate Clockwise

Rotate Counter-Clockwise: Marked by holding the hand orthogonal to the body with a thumbs up and rotating the hand counter-clockwise 90°. This is mapped to the rotation of the drone towards the left. It turns counter-clockwise whenever this gesture is performed. This is also a sequential gesture.

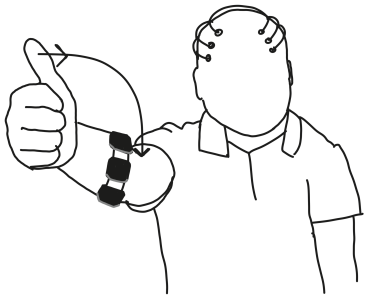


Figure 4.19: Rotate Counter-Clockwise

Transition: This is not actually a gesture. This is the transition from one gesture to the other. This is not mapped to any of the drone controls. Instead, the drone control suspends for a split second when it is detected. This is also a sequential move.

Distracted: These are sequences of operations that do not correspond to any gesture for the drone control. These are usually hand movements other than those regarded for the gesture vocabulary. For example, waving a hand to a friend, etc.

4.3 Distraction: A simple overview

In general, distraction is the process of diverting the attention of an individual or group from a desired area of focus and thereby blocking or diminishing the reception of desired information [53]. In our context, it is that point in time, during the drone control operation, when the controller is subjected to unanticipated disruption from his current activity of controlling the drone. This state gives a state of unconscious activity which might be harmful to the drone. A scenario where the person being inattentive to the drone control performs a gesture within the understanding of the classifier might end the life of the remotely flying drone.

The human brain is prone to distraction. It is just the opposite state to being attentive or concentrating. An article in Time magazine by Kevin McSpadden [54]mentioned, "You have a shorter attention span than a goldfish". Similar studies by Microsoft claimed that the human attention span dropped from 12 seconds to 8 seconds [21]. And we also know there are several accidents that are caused due to distracted driving, which is a cognitive activity and requires higher cognitive ability related to reaction time, focus, sense of judgment, rapid decision making, and much more. On average, distracted driving causes 3000 deaths per year, with the number growing over time [55], which roughly make up 8.7 % of the total fatalities in a year. In our case, the distractions might be causing damage to the drone while flying, or it might not be a direct hazard. Instead, it might disrupt its environment, causing chaos and resulting in an induced accident or compromising the safety of components of the environment it is flying in.

There can be different types of distractions. They are majorly classified into visual, audible, manual, and cognitive. They are described below:

- Visual: A form of distraction that occurs due to the introduction of an unanticipated static or non-static element in the environment that could change sight. This could be a sudden event or multimedia, especially video starting or flashing nearby.
- Audible: This is a form of distraction that occurs due to the introduction of sudden noise or hasty sound that could drive the controller's attention towards it.
- Manual: This is not an environment-induced distraction. However, it might cause one to be utterly inattentive to the current activity to a great extent. This could be when the person voluntarily starts engaging in any other form of activity that refrains him from pursuing the recent activity.

- Cognitive: This could or couldn't be an environment-induced distraction. But mainly it is classified as a distraction caused by the current state of mind of the person trying to be attentive. This distraction causes a break in a person's cognitive thinking and alleviates the higher-order thinking skills required for intelligent decision-making.

For the first two types of distraction, the environment is mainly responsible for the distraction; therefore, as soon as the environment withdraws from the source of distraction, there is a higher chance of the person returning to the attentive state. However, the later types of distraction could be typically long enough and difficult to withdraw from depending on the person concerned.

These distractions can cause the mind and brain to stop being attentive for a more significant amount of time, resulting in a break in the activity cycle. As a result, it could be difficult for the person to return to the controlled environment mentally and resume the process. The physical importance of this comeback is precious in the context of our work as the distraction oversees the gesture control. And if the mind is not in an attentive state, the control process might not even start.

The distraction sequences used in our research are mainly environment triggered or manually triggered. The reason behind this is that it is easier to introduce an environment-triggered distraction because of its randomness and unordered activities that continuously happen around a person. The distraction we are discussing could be caused by a person walking by who possibly could be known to the person controlling the drone or by a car honking or loud music being played in close proximity. Apart from that, an unanticipated event like a sudden surprise or shock could cause him to be distracted from his current state of mind. The manual distractions are simply any kind of physical activity that requires his hands to perform certain gestures (that has

a high possibility to get classified as a gesture) to which the controller of the drone gets attached during the operation.

4.4 Unmanned Aerial Vehicle Control and Operation

We have used a programable drone manufactured by Dji for the purpose of research and development. The drone is branded by the name Robomaster TT. The SDK of the drone lets the user perform control operations using a Wireless UDP Protocol using text commands [56].

Figure 4.20: Drone Kit

4.4.1 Sending and Receiving Commands

To set up the drone to send and receive commands, we followed the following procedure [56]:

1. Connected to Robomaster TT Wi-fi Interface.

2. We set up a UDP Client on the PC, which is able to communicate through Tello IP: 192.168.10.1 and UDP Port: 8889.
3. To be able to set the drone into SDK mode, we sent “command” as the first text command over the UDP Connection.

4.4.2 Receive Tello Video Streams

To set up the drone to send a video stream to the PC, we followed the following procedure [56]:

1. Connected to Robomaster TT Wi-fi Interface
2. We set up a UDP server on the PC to receive messages from IP 0.0.0.0 through UDP Port: 11111
3. After connection, we sent the “streamon” command to Tello UDP port 8889 to start receiving the Tello Video Streams.

4.4.3 Tello Command Types and Results

1. Control Command
   (a) The drone sends ‘ok’ if the command was executed successfully.
   (b) The drone sends an ‘error’ if the command fails to execute.
2. Sending any command to Tello (for a command list see Table 4.1)
   (a) Send any command with the format <command ><parameter >
   (b) The drone sends ‘ok’ if the command was executed successfully.
   (c) The drone sends an ‘error’ if the command fails to execute.

Command	Description
takeoff	Auto Take off
land	Auto Land
emergency	Stop the motor from running
up X	Fly Upward by x cm
down X	Fly downward by x cm
left X	Fly leftward by x cm
right X	Fly rightward by x cm
forward x	Fly forward by x cm
back x	Fly backward by x cm
cw x	Rotate clockwise by x°
scw x	Rotate counter-clockwise by x°
stop	Stop moving and hover immediately
speed x	Set current speed to x cm/s

Table 4.1: Control Commands for Drone

4.5 Integrated Control System

This chapter discussed the system architecture and approach of our project. We have discussed how the gesture and distraction classifier produces output and that stream used by the driver algorithm to decide commands for the drone. This module will elaborately discuss how the driver function determines the command using a flow diagram.

As mentioned earlier, we used two neural network pipelines and used their classification output later in the command generation. The first network comprises two stacked Long-Short Term Memory (LSTM) hidden layers, which take a batch of 32 samples as input during training. The samples are 2-dimensional arrays of length 50 and breadth 34. The output is a probability distribution of 14 classes. On the other side, we implemented an Anomaly Detector using autoencoders to classify segments of distraction. Both the encoder and the decoder had three hidden dense layers each. The input to the classifier was a one-dimensional array with 16 elements.

So the output of the decoder was a 16-element array as well to be able to reconstruct the signal elements.

The LSTM classifier produces streams of a binary array of length equal to the total gesture vocabulary, later decoded to produce the type of gesture classified from the data received. A classifier produces 50 output arrays every second in a given time. These output arrays are considered for the algorithm to decide the command finally. A flow diagram is demonstrated below to elaborate the understanding.

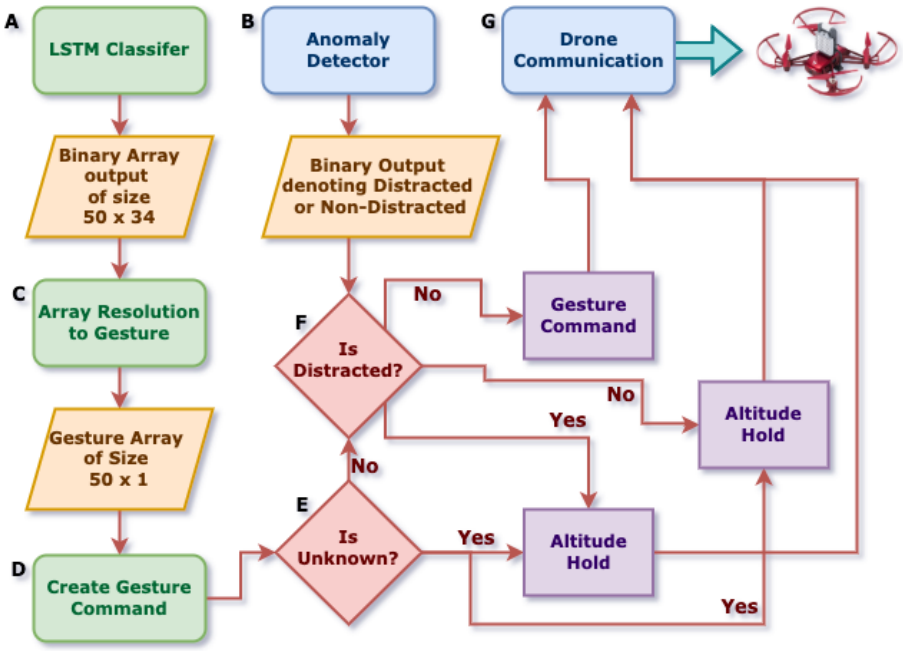


Figure 4.21: Integrated Control System

The stages in the flow diagram are explained below:

A It is the LSTM Classifier we use to classify the gesture sequences.

B It is the anomaly detector we use to classify the distraction.

C The binary array that the classifier sends as output is converted into a text array when a gesture represents each element.

D The text array is now processed to find the maximum repeated element, the first, the middle, and the last element of the array, to create the gesture command finally.

E The generated gesture is further checked and identified if it is Unknown or any other Gesture Command.

F The binary output from the anomaly detector is checked if it is distracted or non-distracted.

G This is the communication window used to communicate with the drone.

Is Distracted?	**Is Gesture 'Unknown'?**	**Command**
Yes	Yes	Hover
Yes	No	Hover
No	Yes	Hover
No	No	<Gesture Command>

Table 4.2: Control Logic

The above logic table is considered to control the drone.

CHAPTER 5

DATA EXTRACTION AND PRE-PROCESSING

5.1 Sensor Overview

For the system proposed here, two main sensor systems have been used, a ThalmicLabs Myo band EMG sensor with an integrated IMU and a Emotiv Epoc EEG headset.

5.1.1 ThalmicLabs Myo Armband for Electromyography and Inertial Measurement Unit

ThalmicLabs produced an intelligent gadget that fits on the forearm and records the electrical activity of the muscle as well as the orientation and acceleration of the hand known as the Myo Armband. The features of the armband are handy as well as it is excellent in terms of wearability and mobility.

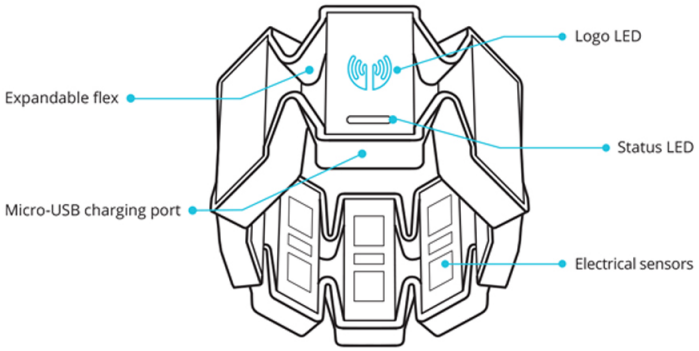


Figure 5.1: Myo Armband [57]

The Myo has eight channels of medical-grade stainless steel EMG sensors. The device is designed to record the Surface EMG from the forearm of the person wearing the sensors. The electric signals that are returned by the sensors are purely electric potentials of the muscles, resulting from muscle activation. The electric potential of the muscle that the myo records are minimal, possibly in millivolts, and therefore signals are sensitive to electric noise induced by wall electricity. Myo records range of potentials is -128 to 128 which is described in activation units. The activation units are generally integer values that amplify the potentials measured by the sensors. The streaming frequency of the Myo Armband is usually 200 Hz.

We used a pre-defined library coded in Python 3. x to talk to the Myo Armband. We used the pre-processing provided in the library to convert the 200 Hz sample into a 50 Hz sample. So, the derived signal was filtered to avoid contamination.

The Myo Armband is also fitted with a 9-axis Inertial Measurement Unit (IMU), which contains a three-axis accelerometer, a three-axis gyroscope, and a three-axis magnetometer. The orientation and acceleration of the forearm and the hand can be recorded by determining and analyzing the spatial data that is provided by the band. The orientation data usually indicates the positioning by the conventional terms of roll, pitch, and yaw. The figure below represents the orientation/acceleration axis of the Myo armband.

The angular velocity of the Myo armband represents the acceleration of the Myo armband at a given time and is represented in a vector format. However, we have understood from our experiments that the Myo armband provides the relative position of the hand more than the absolute position. The armband streams the IMU data at a frequency of 50 Hz. The Myo armband fits on the upper forearm as described in the previous chapters to provide the most information to record the

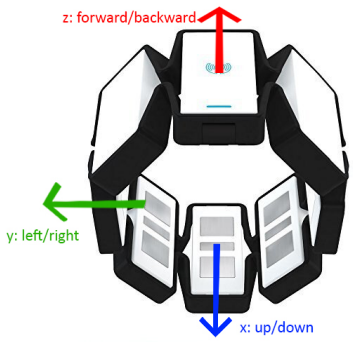


Figure 5.2: Myo Axes

activity in the arm. The diagram in Figure 5.3 below shows the placement of the Myo concerning the hand in a cross-sectional view.

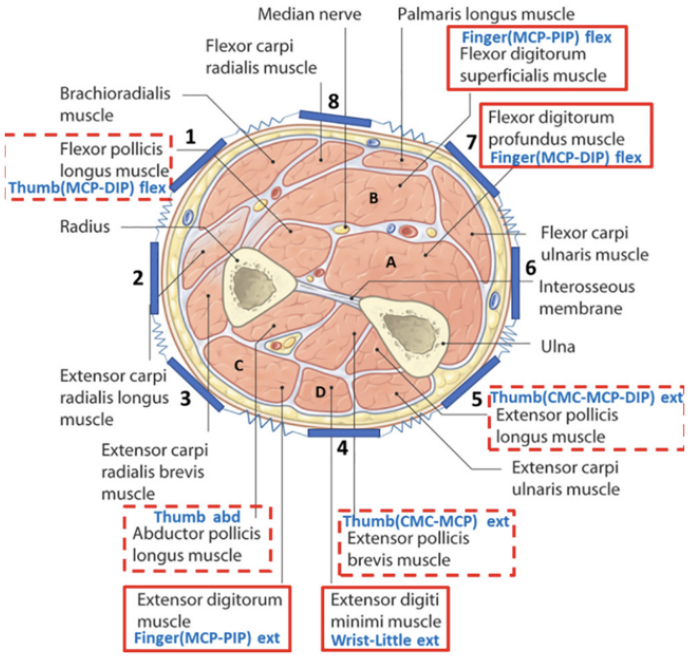


Figure 5.3: Cross-Section of the right middle forearm and Myo placement [58]

We can see how the different muscles get covered by the comprehensive eight-sensor sensing technology to record the muscle activity. This helped us process the segmented data sequences to perform our classification.

5.1.2 Emotiv EPOC Electroencephalography Headset

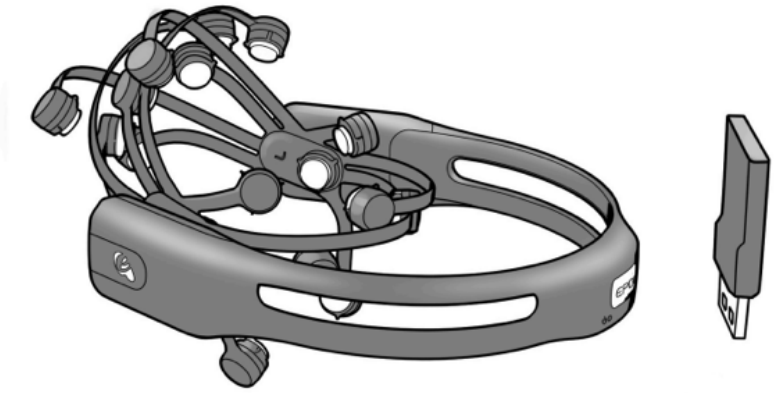


Figure 5.4: Emotiv EPOC Headset with Universal Epoc Bluetooth Receiver [59]

The Emotiv EPOC EEG Headset is a device purposed for the recording of Electroencephalography signals from the surface of the scalp. This is a non-invasive EEG Data recorder with connective pods that conduct through a saline solution. EPOC is a 14-channel EEG Headset capable of recording data at a sampling rate of 128 Hz. The EPOC also has an in-built gyroscope that records the horizontal and vertical orientation of the head. The gyroscope data is also streamed at a frequency of 128 Hz. One more feature of the EPOC headset is that the connectivity of the channel ends to the scalp is also recognized by the device for quality check. The Emotiv EPOC is designed to identify as much as facial expressions, blinks, and the latent brain activity of the person wearing the device.

The EPOC follows the 10:20 alignment system of the EEG channel and has aligned the 14 channels in the following way.

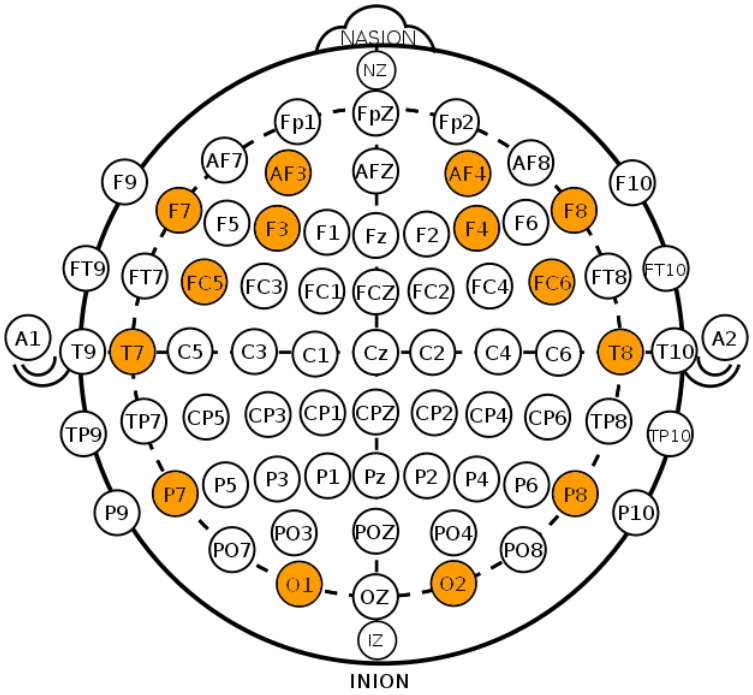


Figure 5.5: EPOC Headset Channel Placement in 10:20 System

The Channels that EPOC can read are AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, and AF4. These cover most of the head and mainly the frontal cortex, which interests us more in this research work.

Both the sensors were used in our project to record gesture and distraction classification data.

5.2 Preparing the Gesture Data

While different phases of experimentation with the data, we recognized that Electroencephalography also plays a vital role in identifying the intuition behind gesture performance. And we were optimistic about finding better results after having

the classifier learn from the phases of EEG data received to enhance the gesture recognition. Therefore, we decided to incorporate the EEG data along with the EMG and IMU Data for a more informative gesture classification. However, there was a problem to be dealt with. The Myo armband streamed the processed signal at a rate of 50 Hz for both the IMU and the EMG. And the Electroencephalography Data that was received from the EPOC Headset was streamed at a frequency of 128 Hz. So, technically we did not have an ordered frequency for the standard merging algorithm to work. That is, we have three to four EEG data for every stream of EMG and IMU. This way, we would either lose valuable information from the EEG Data or have repeated EMG and IMU data streams in a disoriented manner.

Therefore, we designed a data preprocessing pipeline that is capable of receiving both the signals in real-time and merging them to form a single data element at a given time. The algorithm incorporates a server-client architecture. The reason behind creating a server-client architecture was that we had different data extraction methods followed by two of the sensors. EPOC followed an event-triggered loop, and Myo used an interrupt handler to feed the data. Also, both the pe-defined libraries were written in two python versions. So, to integrate the data, we created two different servers for both the sensors and designed a data processing algorithm on the server-side to process the data before sending it to the client.

The client connected to both the servers simultaneously and received data using two multiprocessing queues. The client ran two processes that received the data from the servers and pushed it to a queue. The architecture is given below (in Figure 5.6).

The reason behind using a server is also a different issue. This was mainly because the Myo data received was not organized in a regular interval. This means the data, even if streamed at 50 Hz, each data sample was not sent at a 20 ms interval. The original data recording had a random selection of intervals, making

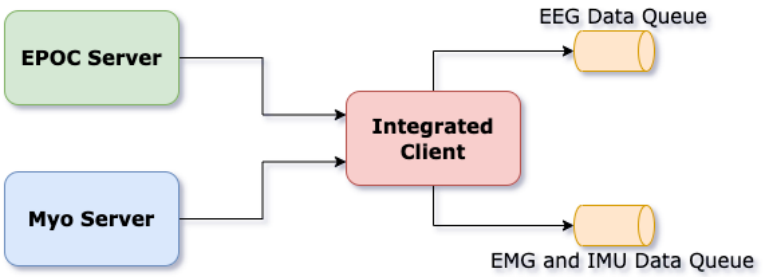


Figure 5.6: Server-Client Architecture for Gesture Data

the data integration even more difficult. As the EEG samples were streamed at a regular rate of approximately 7.8 ms interval, it was tough to map the signals that came simultaneously. We had the intuition that even if the data was streamed at an unordered pace, however, the generation of the signals was not unordered.

This intuition helped us create a server code that reorganizes the timestamps to make them regular and remove burst inputs whenever we receive a signal. The algorithm considers the current timestamp of the received signal and the previous two timestamps to organize the new timestamp that has an order close to 20 ms.

There was another issue noticed but now with the EPOC headset. The data streamed at a frequency of 128 Hz and approximately 7.8 ms apart; there was a 7 – 8 ms delay after every 128 data sample. This delayed the streaming mechanism too. This is because, after 128 seconds of data collection, we might result in a 1-sec delay, which will continue to increase every 128 seconds. Though the real-time delay was inevitable, we used another preprocessing algorithm of like nature to the one we used during the gesture data collection to reorganize the timestamps and aid in merging the data samples.

We created the mock server-client architecture to address these issues, where the server acts as a middleware between the data accumulator and the streamer. We first organize the data samples in the server using the algorithms mentioned above

to prepare the data for streaming to the client. We have described the algorithms in detail [Algorithm 1, 2 in Appendix A]. These algorithms take the unprocessed timestamps for each data sample as input and output an organized timestamp mapped with the same data sample to reserve the information and ensure data integrity. Here we manage the timestamps by considering the difference between the last and current samples. And then, we send the data to the client.

We have a separate algorithm waiting for the data to be received on the client-side. On the client-side, we sort them into a queue and finally attach them with respect to their arrival time [Algorithm 3 in Appendix A]. The algorithm to perform the operation is elaborated on later in the chapter.

After all the above pre-processing of the data is completed, we finally normalize it to each data sample with respect to the homogeneity of the signals in a given sample. The normalization is performed to perform better weight training in the neural network pipeline.

5.3 Preparing the Distraction Data

The distraction data was not very difficult to organize. It simply uses Algorithm 2 to manage the timestamps. A significant part of the acquisition was to contain the timestamps. This majorly impacts the processing of the distraction as the delay in time might cause a considerable control suspension. In most of the EEG-related studies, they have performed a time-domain analysis or a frequency-domain analysis. Also, in some of the works, they have performed Independent Component Analysis. But for, any of the given operations requires a different algorithm that would process the sample in batches or create the Fast Fourier Transform (FFT) for the analysis. But, for our project, we were classifying the distraction using an Anomaly Detector, which needed non-distracted samples to identify the distracted pieces using the clas-

sifier. Therefore, we experimented with the dataset to find out the non-distracted triggers. To perform the experimentation, we recorded data samples of a regular person controlling the drone, unaware that he would be subjected to repeated distractions during the operation. We subjected him to multiple distraction events and recorded the timestamps of the sequence of distractions. After annotating the dataset with binary results, we performed specific visualizations to identify the distraction patterns in the dataset. The time-driven data visualization is given below:

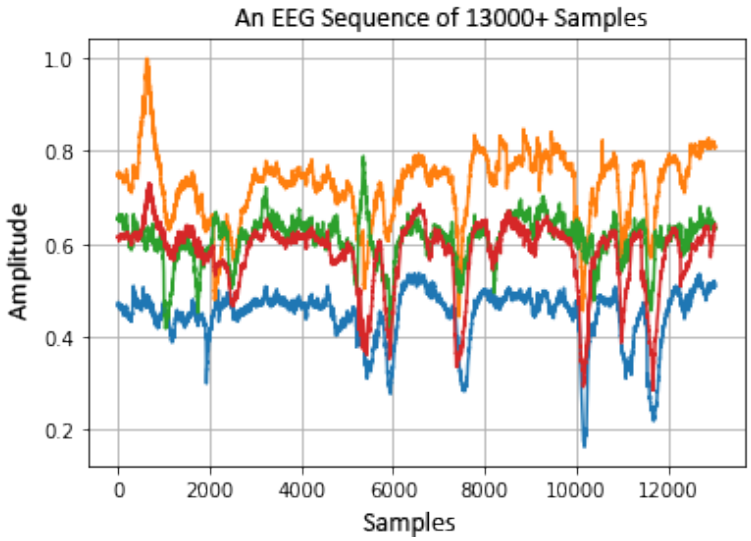


Figure 5.7: Time-Driven EEG Visualization of 4 Channels in the Frontal Cortex

We have concluded that even if there are distraction patterns in the data, it is hard to distinguish the length of distraction in mind. We are more interested in the length of distraction that is cased in the humans and the intensity, if possible, for our scenario, which is difficult to be interpreted using this kind of data visualization.

So, we developed a new technique to identify any possible distractions in the sequence and the length of the distraction. So, we visualized each sample after receiving the normalized data. We found that any normalized data could be plotted in

an XY coordinates system with the x-axis representing the signals received and the y-axis representing 0 to 1. The following is a representation of a Non-Distracted EEG Sample after normalization.

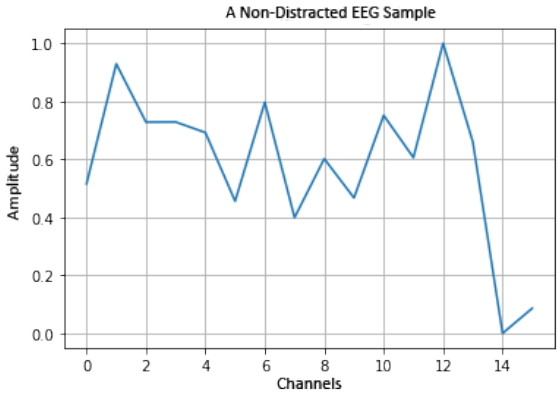


Figure 5.8: Non-Distracted EEG Sample

Also, a distracted sample looks utterly different from a non-distracted as shown below.

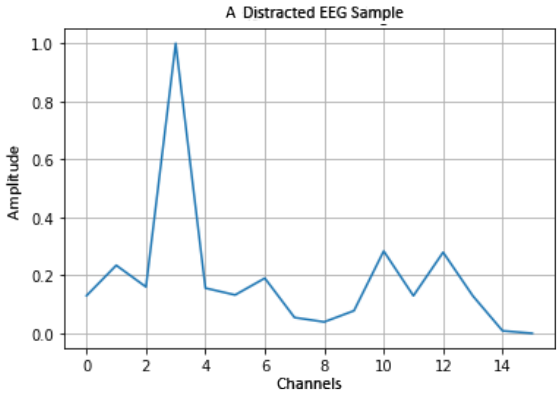


Figure 5.9: Distracted EEG Sample

This can be easily identified from the graph. It starts to change shape as the person enters the distraction phase. This would help us to identify the length of

distraction. Also, the benefit of using an Anomaly Detector is that we can find how different the anomaly is from the standard graph. This way, we can grade the distraction of a human during a particular sequence. This method is independent of any sequence information, making it more reliable in identifying the distraction as early as possible to suspend the control operation. As we look into each sample, we do not need a sliding window algorithm which will be discussed in the chapters later, to facilitate the classification.

It is fascinating to understand that there is a beautiful transition that takes place in the EEG when a person approaches towards distraction sequence. This phenomenon aided our purpose in identifying the intention of the person controlling the drone. We have also seen significant randomness in the data when determining the length of the distraction. In no instance were we able to find a non-distracted EEG sequence during the distraction, which says, even if the person tries to put attention in any kind of distracted sequence to the source of distraction, the drone will not receive any ambiguous command.

Finally, after performing all the preprocessing techniques, the data was now ready to be classified. Not only the recorded data was processed, but also, using the same processing pipeline, we created a segment where the live data can be subjected to the same algorithm to behave like the recorded data. The server-client architecture was incorporated in the live setting to process the incoming data in real-time and produce no time mismatch for the client to recognize the data with proper time entries.

We created a pointer in the algorithm that collected the data that always pointed to the new data received and treated it as a new line for the algorithm to process. The data was sent to the classifier after a specific interval to maintain the input shape of the data that was previously used to train the classifiers.

CHAPTER 6

ALGORITHMS, IMPLEMENTATIONS AND RESULTS

6.1 Gesture Classification

The gesture classifier is the driver in our research work. The drone would receive controls to fly, and the control generator would receive classified gestures to decide the final command to be sent to the drone. In this section, we will discuss the different algorithms that led to the training of the classifier and the experimental setup required to process the data for this classifier.

6.1.1 Experimental Setup

There were two phases of data recording for this classifier. The first phase was a simple iterative process to record only gestures. A human subject was chosen and given the responsibility to perform gestures pre-discussed in the earlier chapters. Before the training, the subject was trained on the gesture vocabulary to perform the gestures as required. The data was recorded for the duration of approximately three minutes. The time of recording each iteration for the gestures was kept low because as the gestures were performed using hand movement, they caused stress in the arm after a specific time. This time is not constant and varies between different people. These three-minute iterations were performed 20 times, giving us a full data recording of 60 minutes. Each sample had the first 14 elements corresponding to the 14 channels of the EEG Data, two elements from the EEG Device Gyroscope, eight elements from the eight channels of EMG, four elements from the Quaternion from the IMU, three elements from the Gyroscope of the IMU and three elements from

the Accelerometer of the IMU. Each sample of this recording has 34 features for the classifier to learn from. This set of 34 features was sequential data and each sample corresponded to a part of gesture execution.

This means each sample was necessary for the classifier to recognize a gesture signature but was insufficient. Therefore, we conducted experiments to find the minimum number of samples required to fulfill the sufficient condition for recognizing the gesture. We understood that a regular gesture requires, on average, 2 seconds from the start to the end of the gesture. The execution time of each gesture could be 2 seconds on average. But, if we have to wait for the gesture until its completed, there will be a significant delay in recognizing the gestures and sending the command to the drone. A two-second delay for every control could be tormenting. So, we divided two-second data into four pieces to find the difference each phase creates for at least understanding the result manually.

So, we understood that every gesture had four phases of execution. The first and the last phases are more about entering and leaving the phase. The two phases in the middle were the actual activation that was caused by the muscles due to the unique hand and forearm movement. So, now we had 500 ms of the gesture phases labeled as the gesture class for the classifier to start predicting the output right at the beginning of execution. The figure below describes the total data samples received in each iteration. The best part about choosing a four-phase strategy was, that the transitions had an average execution time of 500 ms, equivalent to every phase. So, they did not have different phase structures which might have increased the computational complexity of the algorithm.

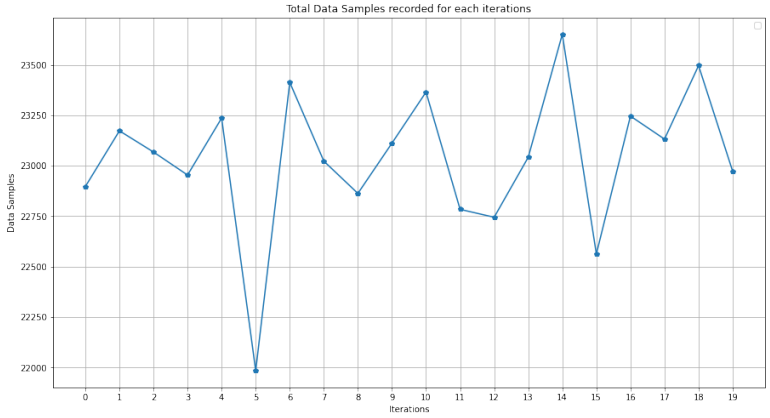


Figure 6.1: Total Data Samples recorded for each iteration

The recorded data was timed and labeled from a cross-referencing video shot during the gesture recording. The video was layered by a UNIX timestamp likewise the data. From the video, the start time, and the end time of a gesture along with the gesture it referred to, were entered into a pool file. And from the pool file, the dataset was labeled using a simple algorithm [Algorithm 4 in Appendix A]. This algorithm takes the start and the end time from the pool file along with the corresponding class of gestures. Then it iterates over that sequence of the dataset that has a timestamp greater than the start and less than or equal to the end time from the pool file. For all the satisfying rows, the label is entered corresponding to the activated label from the pool file. Once the dataset was labeled and ready, we had to prepare segments of the data for the Long-Short Term Memory (LSTM) Recurrent Neural Network Algorithm to recognize, train, and predict.

6.1.2 Sliding Window Algorithm

To prepare segments of data, we designed a sliding window algorithm [Algorithm 5 in Appendix A]. This algorithm performs like a window that slides over the data and prepares segments of data samples treated as one sample by the LSTM Classifier.

The algorithm takes the whole dataset as input. Along with that, it defines the size of the window (which in our case was the length of each phase of execution) and the offset for the overlap. Here, the offset is initially decided, and it depends on the amount of overlap we decide to segment the dataset samples. This offset is very important as a higher amount of overlap will help to identify the sequential changes that happen when we move along the gesture to perform operations.

In the sliding algorithm, the pointer loops for a certain period of time depending on the size of the window and the length of the dataset. Every loop is subjected to another iterating loop of samples that is equal to the window length. It selects that chunk of samples and ushes it to the final array. The main loop progresses at a step corresponding to the offset to perform the segmentation. The sliding window algorithm [Algorithm 5] is also used to segment the class labels corresponding to the samples.

After the application of the sliding window algorithm, the dataset looks like a three-dimensional NumPy array of sample size depending on the length of the dataset, 50 rows, and 34 feature columns. Similarly, the mapped class labels take the dimension of 50 x 1.

We will discuss later in this chapter the loss function used in the classification, but for the data pre-processing we had to create binary arrays of the classes using one-hot encoding for the loss function to recognize the labels well. As this is a multi-class multi-modal classification, we used Categorical Cross Entropy as the loss function for the classifier which will be discussed later in detail. Finally, we converted the class labels into a shape of 50 x 14 corresponding to each feature set in the input data.

We had 14 gesture and non-gesture class labels. They were organized as: 0: Start/Stop (This is a toggle gesture), 1: hover, 2: left, 3: right, 4: up, 5: down,

6 forward, 7: backward, 8: rotate_left, 9: rotate_right, 10: relax, 11: to_hover, 12: to_relax, 13: unknown (This is when a person performs a non-gesture).

After performing the one-hot encoding of the class labels, they took the shape of binary arrays as shown in the figure below.

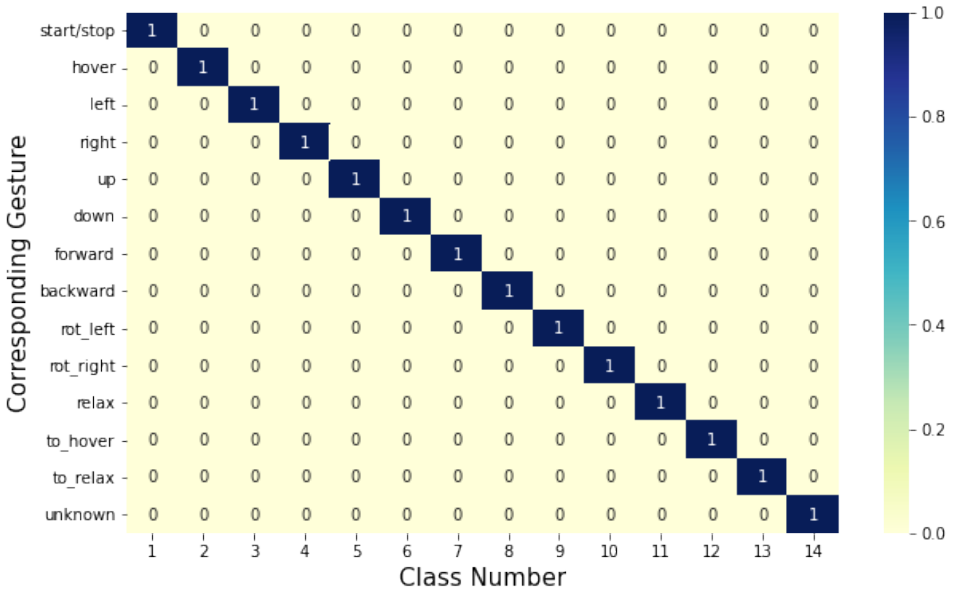


Figure 6.2: Classes after One-Hot Encoding

6.1.3 Stacked Long-Short Term Memory Classifier

A stacked LSTM is not very different from the original LSTM model. Usually, an original LSTM model has a single hidden LSTM layer followed by a standard feedforward output layer. However, the stacked LSTM differs by an extension of the model that has multiple hidden LSTM layers containing numerous memory cells. The reason for increasing the network's depth is majorly credited to the efficiency of the approach of a complexified prediction. This arrangement is faster in terms of a single layer LSTM which is layered by multiple nodes in a single hidden layer. And also,

the level of attention the stacked LSTM can put on a sequence of data to learn the occurrence of patterns is much higher because of the depth of the network. It adds to the level of abstraction of input observation over a time frame. This can be enhanced by our method of using the sliding window to prepare the data sample and create a pattern.

We have used a Stacked Long-Short Term Memory Recurrent Neural Network to classify gesture sequences. Our model contains two hidden LSTM Layers with increased memory cells on the second layer. The layers are separated by dropout layers. The visual representation of the stacked-lstm network is described in the figure given below.

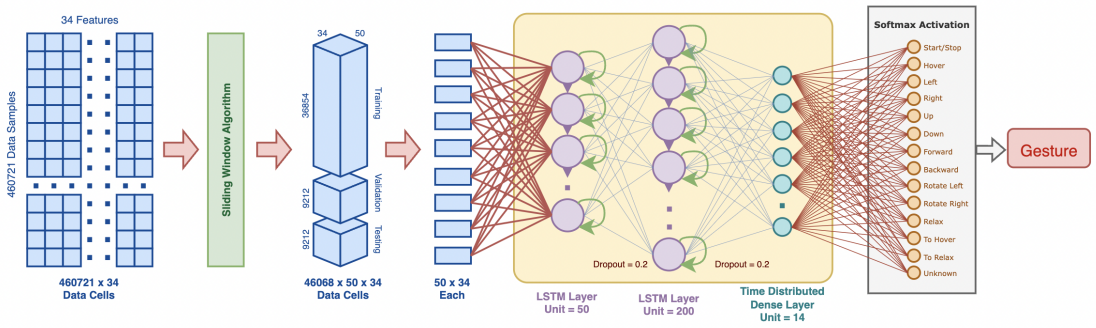
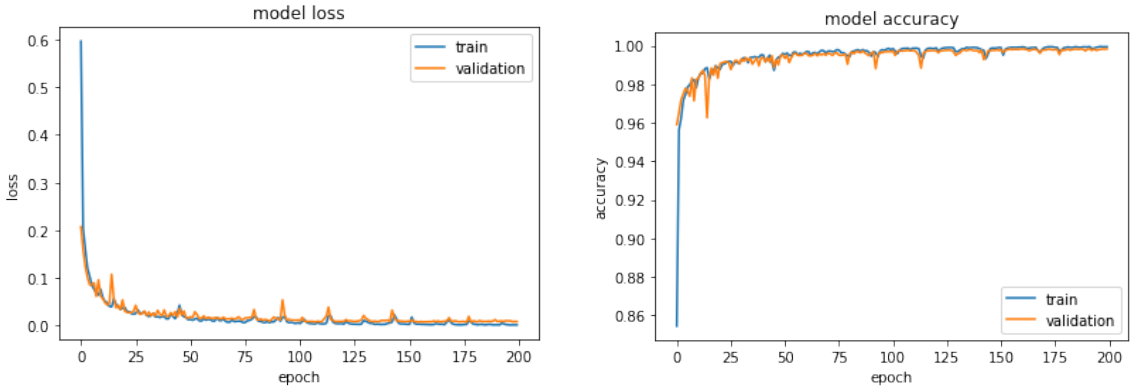


Figure 6.3: LSTM Model Architecture

The first LSTM Layer has 50 memory cells and passes the output to an LSTM layer with 200 memory cells. The dropout between the transitions is 0.2. Finally, the output of the LSTM is sent to a Time Distributed Dense layer with 14 nodes corresponding to the 14 classes of gesture. And then, a SoftMax activation is used to predict the probability of the classes. The architecture proved to be highly efficient in terms of the accuracy and precision of the model.

6.1.4 Results of Gesture Classification

We have performed multiple training sessions with different seed values of the random data shuffling. The results were fascinating as they performed well in predicting the gestures correctly. The confusion matrix for the LSTM classifier is given below. The training of the model was performed over an Epoch length of 200. The training performance is shown below with the loss and accuracy trends.



(i) (ii)

Figure 6.4: (i)Gesture Loss Trend, (ii) Gesture Accuracy Trend

From the confusion matrix, we can really say that the model performed really well even with the test data. The stacked nature of the model aided the classification process not only by looking for the patterns in the data but also the ambiguous sequences that might take place during unknown gesture recognition.

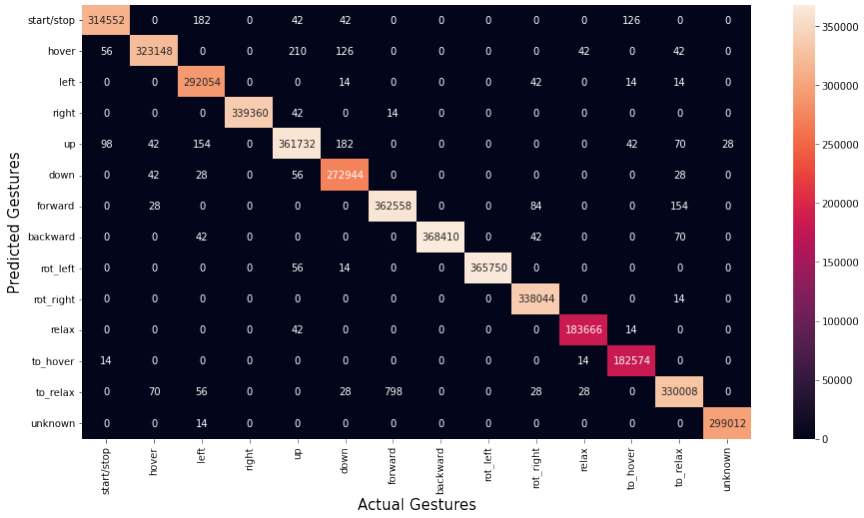


Figure 6.5: Confusion Matrix for LSTM Classifier

The model returned an excellent accuracy for the range of experiments performed (Number of Trials = 10) as shown in the figure given below.

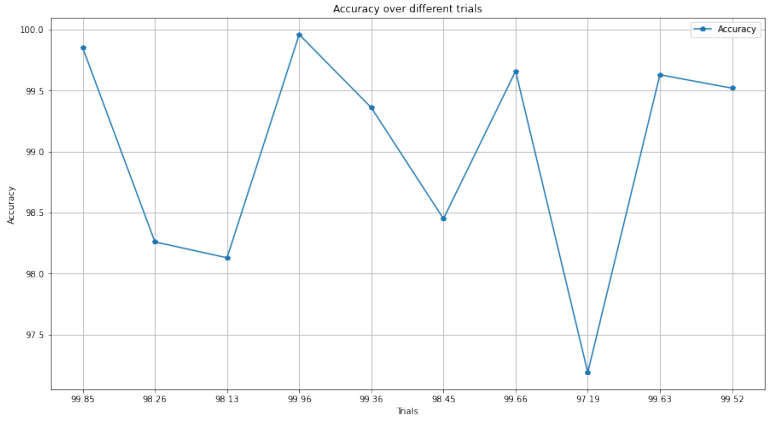


Figure 6.6: Accuracy over Trials for LSTM Classifier

From the confusion matrix, we derived the recall, precision, and f1-score of the model, and even here, we observed a promising trend in the scores of different classes. The figure given below is the average metrics over the trials.

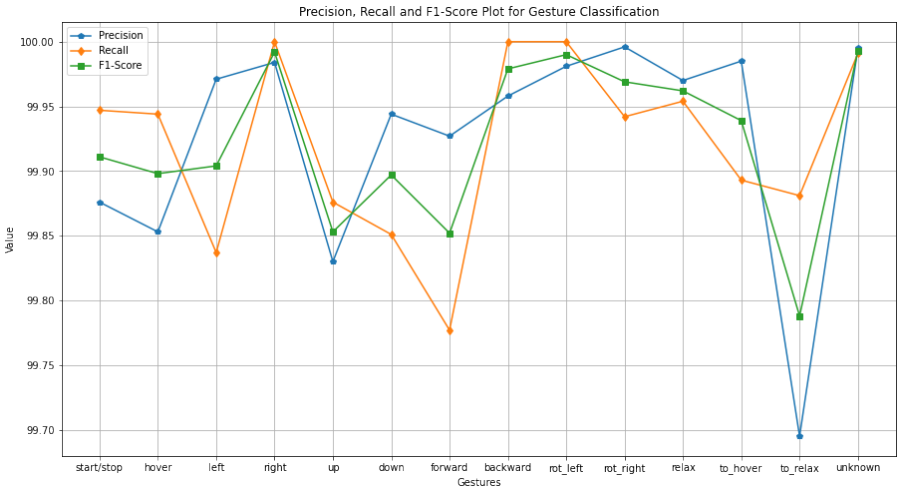


Figure 6.7: Recall, Precision, and F1-Score for LSTM Classifier

6.2 Distraction Classification

Experimental Setup

The experiment was set up after we had classified the LSTM Classifier and then used it to fly the drone while recording the data for the distraction classifier. The data was again recorded from a single subject over a time period of 4 min and iterated 10 times. In this scenario, the subject was not aware of the process of distraction that was going to be sourced to record sequences of distraction. The subject performed different gestures to fly the drone. The basic motions took place while the mind was concentrated. And hence the data received was later labeled as one as we were more interested in training the model over the non-distracted data.

Once the recording started, the subject was asked to start flying the drone. Once the drone was in the air and was controlled by the subject, a third person was appointed the duty to create distraction triggers during the flight. The types of distraction that were recorded are listed below:

Visual Distraction: The subject was shown a video sample on the screen while controlling the drone. This was a completely unanticipated step performed without prior notice to the subject. The video was played for a duration of about 20 sec.

Audio Distraction: The experimentation with the audio distraction was performed with different types of audio. The first was plain noise which did not have any vocal or music. The second was high-volume music. The third was the sound involved in an everyday operation like the horns of buses and cars, things falling with noise, phone ringing, etc.

During the distractions, the person concerned was allowed to perform anything he felt to while responding to the distraction source. The distraction sequence was marked by timestamps similar to the gestures. Later, these timestamps would be made available to the labeling algorithm [Algorithm 4] to set the label to the file.

The full recording of the distraction experiment was 40 minutes. This had a 70:30 ratio with 70 % data available from non-distracted sequences.

6.2.1 Annotation and Modelling

Annotating the data was more straightforward as the label was binary in this classification task. Although we labeled all the data, we used only the non-distracted data to train the classifier. The labeling was performed using the pool file, where a video corresponding to the recording was used with a UNIX timestamp to map the timing of the video to the actual recording. This was the first level of intuition which was derived to facilitate the second phase to mark the label. In the second phase, the

normalized graph of individual data samples was generated for every sample over a period of time. The animation was then created to find the point where the distraction started and ended. Firstly, we divided the distraction phase into three. The first was the part that showed the entry to the distraction phase, and the third was the exit from the distraction phase. It was the second phase which was stable enough to be classified as a distraction. But for the perspective of our research, we labeled the entire sequence as distracted data.

Once the data was labeled, we modeled it for the classification. Here the sequence of input was not essential; as a result, we only sent individual samples to the classifier for the training. But for the classifier to recognize it more precisely, we created a different feature set—one for the EEG Data and the other for the Gyroscope. The gyroscope data was two-axis data. And our intuition said, if a person is distracted, most likely he will be moving his head to a different position which might not be the position during a proper control operation. So, we prepared the data sample so that we could process the EEG and Gyroscope data during our train.

Since we had very few sequences of non-distracted data and the stability of the data was more with the non-distracted phase, we segmented only the non-distracted data for batch training of the anomaly detector. Now, the data was ready to be sent to the training.

6.2.2 Anomaly Detector using Autoencoders

In this unpredictable world, it is tough for a person to say the mode and level of distraction that could be caused during any operation, which makes it utterly impossible to find enough data to train a model of classification that could identify the difference between a distracted or a non-distracted phase of operation. The more challenging task would be identifying the distraction when the classifier had never

seen something of that kind before or during the training. Anomaly Detectors were built on this principle of training data based on only one class that can be trained and referenced. This refers to a more stable type in terms of its availability and randomness. The other way to look into the use of Anomaly detectors is when we have very little data to train on. This means that even if we have enough samples, the nature of the randomness is minimal. As a result, the identification could be dramatically tricky. There are three different types of anomaly detectors. They have Supervised Anomaly Detection, where the classes are marked as 'normal' or 'abnormal.' But this kind of training is not practiced in a real-life scenario just by its existence. If we have enough samples to train a classifier for both classes, we will probably use some other models.

The second type of Anomaly Detector is Semi-Supervised Detection. This assumes that some of the data are labeled. The label can be assigned to any fragment of the dataset, both normal and anomalous. The third type is unsupervised detection which assumes the data to be unlabeled. For our project, we have used the Unsupervised Anomaly Detection.

In our project, we implemented an Anomaly Detector using a dense autoencoder network. The primary use of autoencoders is dimensionality reduction. But we used it to find a reduced graph from the normal EEG samples. This reduced graph was used to find the error percentage with a normal EEG sample and an abnormal EEG sample. Later, the error of reproduction was identified by testing the model by regenerating a normal EEG and abnormal EEG using the trained dense layers. Usually, the error from generating a normal EEG would be lower than that of an abnormal EEG. This was the intuition behind designing the model. The architecture of the anomaly detector is described in the figure given below.

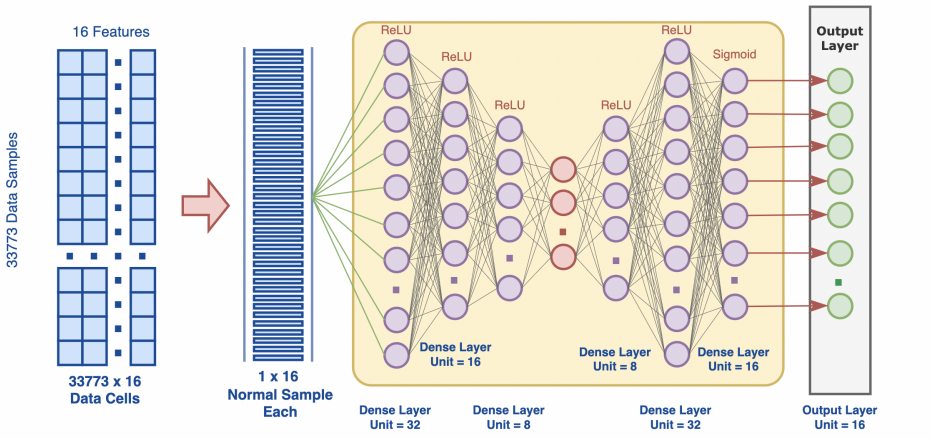


Figure 6.8: Anomaly Detector using Autoencoder Architecture

In the architecture described above, we can find an encoder and a decoder model to depict the autoencoder. The output of this model is a 16-element array that is the reconstruction of the sample that is passed to the classifier for performing the classification task.

6.2.3 Results of the Distraction Classification

For the anomaly detector, we have trained the classifier for hyper-parameter tuning. But the training was performed once to derive the optimal solution possible. The training of the model on the data sample was accuracy centric. So, the model worked to improve the accuracy over epochs. The loss curve during the training on a normal dataset is given below.

The loss indicates the training success of the model with the regular dataset. After the training was completed, the model was tested with the normal and abnormal data samples. The error after reconstruction has been plotted to identify the difference between the two types of EEG Signals received. The plots below show a significant

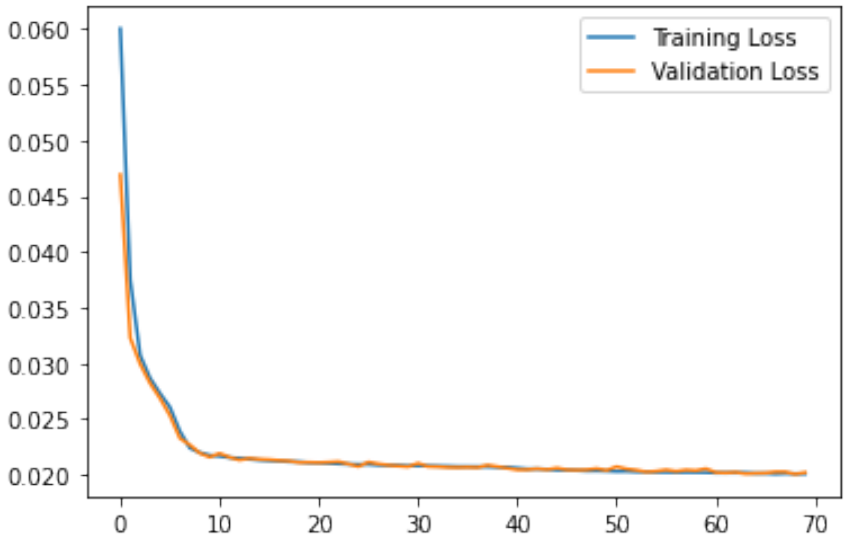
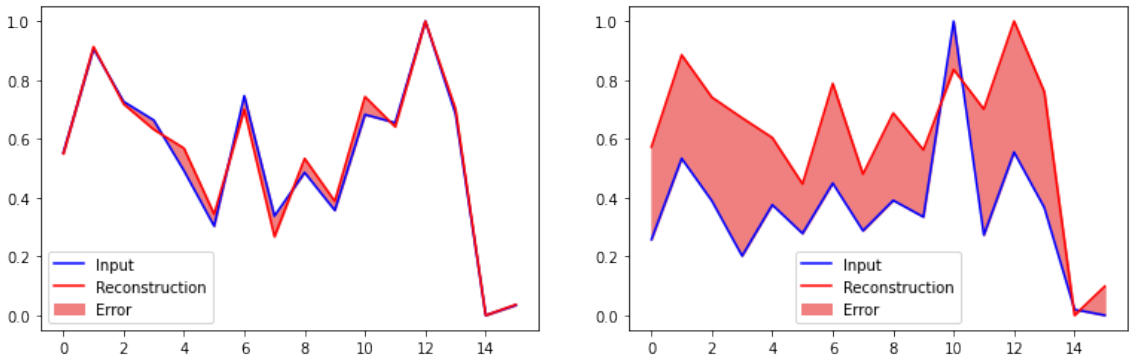


Figure 6.9: Loss over 70 epochs and Batch Size 20

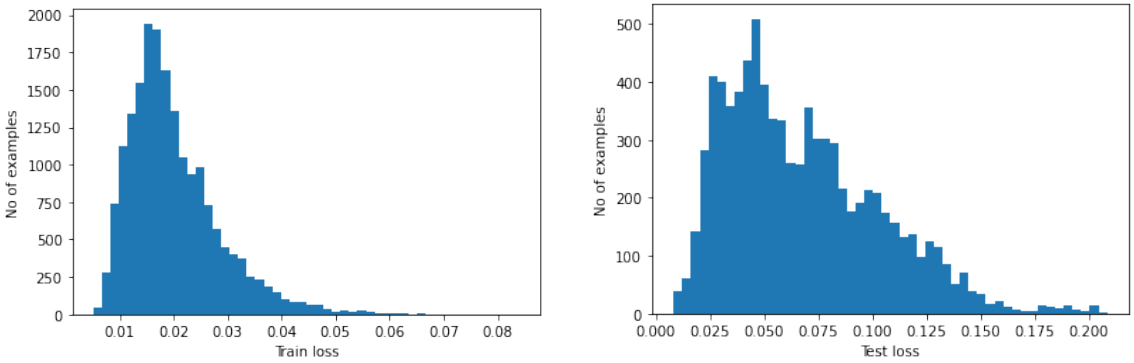
difference from the original and reconstructed graph for the abnormal, whereas the normal EEG doesn't have a high loss.



(i) (ii)

Figure 6.10: (i)Difference between input and reconstructed plot for normal EEG, (ii) Difference between input and reconstructed plot for anomalous EEG

After this experiment, we plotted the number of samples with their losses to find out the threshold which will be used to differentiate the normal EEG from the anomalous EEG.



(i) (ii)

Figure 6.11: (i)Training Loss for Anomaly Detector, (ii) Testing Loss for Anomaly Detector

After performing the mean and standard deviation operations, we derived a threshold by adding both. The result of this operation was approximately 0.028. But, it did not produce the optimal solution. So, we experimented by moderating the threshold manually to derive an optimal condition. After the moderation, we determined the new threshold as 0.0355. With this threshold, the model performed better. The metrics shown by the model with this condition were:

Accuracy: 90.0006% **Precision**: 91.8653% **Recall**: 93.8976%

The confusion matrix was then plotted (Figure 6.13) to identify the prediction quality on the test set. The confusion matrix is given below. The predicted class is on the y-axis, and the actual class is on the x-axis. Here, 1 represents non-distracted class, and 0 illustrates distracted samples.

From the confusion matrix, it wasn't easy to identify if the misclassified examples were in a streak or randomly spread in the entire dataset. So, we segment the dataset to show the True Positive, False Positive, False Negative, and True Negatives

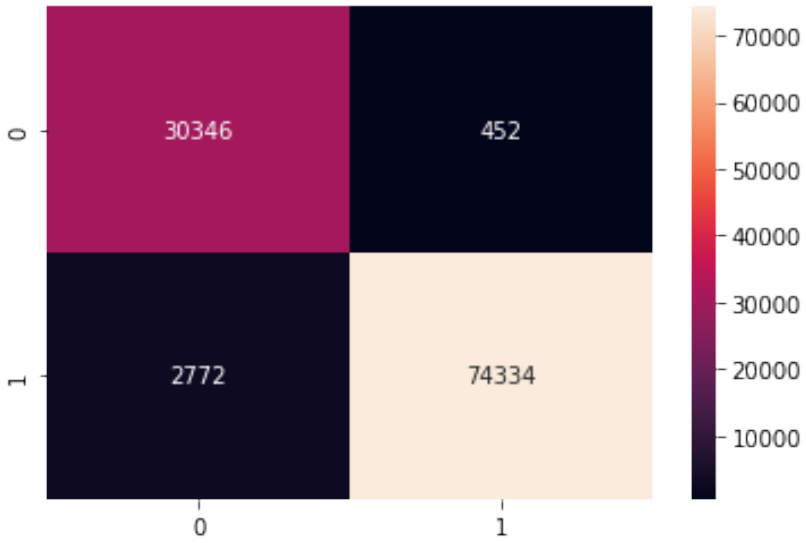


Figure 6.12: Confusion Matrix for Anomaly Detector

of our classification. The levels in the plot are arranged in the order given, with the lowest level corresponding to True Positive and the highest level denoting True Negative. The colors of the plot are defined by the gesture classified. Black represents the unknown gesture.

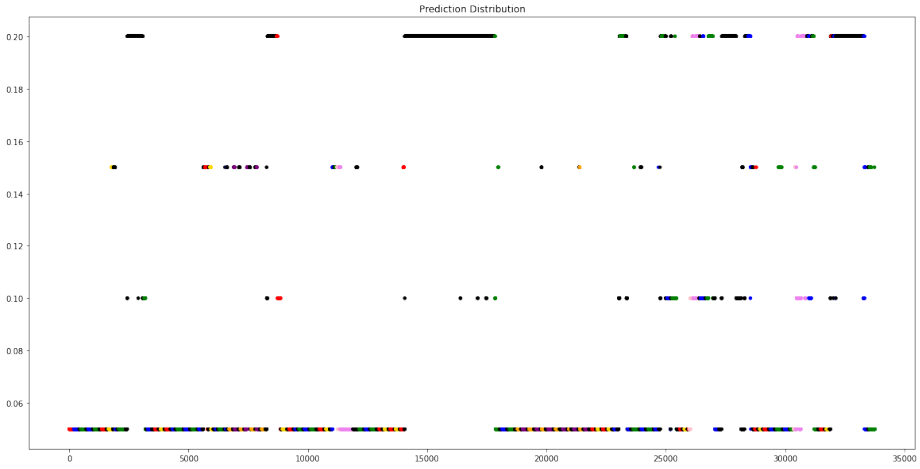


Figure 6.13: Prediction Quality

From the above plot, we notice that the False Positives and False Negatives are actually spread and not always in a streak. The only time it was in the bar was due to a mislabeled classification which can be fixed during live interfacing.

From the results of both the classifiers, we can conclude saying the classifiers perform significantly well with the given condition. The joint performance of the classifiers can be evaluated during the live operation of the drone.

CHAPTER 7

CONCLUSION AND FUTURE WORKS

7.1 Conclusion

The logical understanding of the work represented in this thesis was identifying the distraction phases in a person and recognizing the intention to control a drone using hand movements classified as gestures. This is an activity that requires processing human-activity data in real-time and sequencing information for drone operation. Therefore, the scope of the project reflected not only on distraction identification, but it was on processing human interface data and making it available for lightweight classifiers to recognize the difference in particular cued sequences that could be related to a specific type of gesture. To pursue such a level of recognition and identification, we deployed several real-time data processing algorithms that are able to organize and attach data fragments streamed at different unorderable frequencies. After the base model of data, extraction was available, we designed data recording sequences of varying lengths and parsed the data on the basis of timestamps from a video recording of the data recording interfaces. Once the data recording was completed, human effort was employed to annotate the time-varying data based on the timestamps and human understanding. Later, the same data was used to classify gestures and distractions. Two types of the classifier were used to facilitate the recognition process. The first was a stacked LSTM Classifier which was trained to recognize fixed sequences of gestures performed over time. Another classifier was designed to detect anomalies in the EEG Data. The function of this classifier was to oversee the gesture classifier

and suspend command generation at distracted segments. The algorithms performed exceptionally to perform the specified task.

The metrics of the classifier were then discussed to compare and contrast between different hyper-parameters that led to a lightweight model capable of performing with higher accuracy and precision. Finally, we can conclude that the experimentations adopted in this work were successful in creating a robust plug-and-play gesture recognition system with a model classifier to recognize the intention and distraction of a person controlling the drone. The models efficiently balanced the accuracy and precision intelligently and produced commendable outcomes.

7.2 Future Work

As part of this research's future scope and goals, we have concluded with the following points that require attention and can be pursued to aid the performance of the currently proposed system.

- We plan to optimise the current model training to answer more control driven questions like, how often the classifier wrongly classifies a gesture or as a whole. Also, the performance testing in a real world scenario will give more clarity on the current state of training.
- We plan to design a one versus all anomaly detector to detect the gestures individually and generate a gesture activation potential to aid the current classification algorithm.
- We plan to perform segmented experimentations to predict the state of mind from the EEG Samples to have a better understanding of the human intention and predict plausible loss of mindfulness or attention before the absolute occurrence.

- Currently, the drone was provided with an Altitude Hold Command in a distracted or unknown gesture situation. But a prospective autonomous capability of the drone could help the person controlling the drone perform the operations better later in the distraction sequence. Also, in a looping distraction condition, the drone can navigate back to safety. One more possible source of mindlessness is the physical health of the controller. We plan to incorporate more human activity data to determine the physical health of the person to aid the classifiers and also by the mobility and the wearability of the system.
- We look forward to also try to recognize the level of attention, concentration of a person without the need of individual calibration from brain data which could aid in any control operation, and not only flying a drone.

APPENDIX A

ALGORITHMS

Algorithm 1 This algorithm organizes the EMG and IMU timestamps

1: **procedure** ORGANIZER($emg \parallel imu$) ▷ This function is called by the interrupt handler

Require: $(emg \parallel imu) \neq 0$

2: $offset = 20ms$

3: $old\_time \leftarrow 0$

4: $last\_time \leftarrow 0$

5: **while** true **do**

6: **if** old_time **then**

7: $\Delta time \leftarrow current\_time - old\_time$

8: **if** $\Delta time \leq offset$ **then**

9: $new\_time \leftarrow min(last\_time + offset, current\_time)$

10: $last\_time \leftarrow new\_time$

11: **else**

12: $new\_time \leftarrow old\_time + offset$

13: $last\_time \leftarrow new\_time$

14: **end if**

15: $send(new\_time, (emg \parallel imu))$

16: $old\_time \leftarrow current\_time$

17: **else**

18: $old\_time \leftarrow current\_time$

19: $last\_time \leftarrow current\_time$

20: $send(current\_time, (emg \parallel imu))$

21: **end if**

22: **end while**

23: **end procedure**

Algorithm 2 This algorithm organizes the EEG timestamps

1: **procedure** ORGANIZER(eeg) ▷ This function is called by the event handler

Require: $eeg \neq 0$

2: $offset = 7.87ms$

3: $old\_time \leftarrow 0$

4: $last\_time \leftarrow 0$

5: **while** true **do**

6: **if** old_time **then**

7: $\Delta time \leftarrow current\_time - old\_time$

8: **if** $\Delta time \leq offset$ **then**

9: $new\_time \leftarrow min(last\_time + offset, current\_time)$

10: $last\_time \leftarrow new\_time$

11: **else**

12: $new\_time \leftarrow old\_time + offset$

13: $last\_time \leftarrow new\_time$

14: **end if**

15: $send(new\_time, eeg)$

16: $old\_time \leftarrow current\_time$

17: **else**

18: $old\_time \leftarrow current\_time$

19: $last\_time \leftarrow current\_time$

20: $send(current\_time, eeg)$

21: **end if**

22: **end while**

23: **end procedure**

Algorithm 3 This algorithm attaches data from queues

```
1: procedure ATTACHER(emg_queue ∧ imu_queue ∧ eeg_queue)
2:     Δtime ← 100ms
3:     front_emg ← emg_queue.pop
4:     front_eeg ← eeg_queue.pop
5:     while true do
6:         if ¬front_eeg||front_eeg > current_time − Δtime then
7:             front_eeg ← eeg_queue.pop
8:             waitabit(8ms)
9:         else
10:            attach(eeg[front_eeg)
11:            if ¬front_emg then
12:                front_emg ← emg_queue.pop
13:            end if
14:            while front_emg ∧ (front_emg < front_eeg) do
15:                front_emg ← emg_queue.pop
16:            end while
17:            if front_emg then
18:                attach(emg)
19:            else
20:                attach(0)
21:            end if
22:            front_eeg ← eeg_queue.pop
23:        end if
24:    end while
25: end procedure
```

Algorithm 4 This algorithm labels dataset from pool file

1: **procedure** ORGANIZER($dataset \wedge pool$)

Require: $(lenth(pool)) \neq 0$

2: $start_{time} \leftarrow 0$

3: $end_{time} \leftarrow 0$

4: $label_{list} \leftarrow 0$

5: **for** $index, row$ *in* $pool$ **do**

6: $start_{time} \leftarrow row.pop[index, start]$

7: $end_{time} \leftarrow row.pop[index, end]$

8: $label_{list} \leftarrow row.pop[index, label]$

9: **for** row *in* $dataset$ **do**

10: **if** $start_{time} < row[time] \wedge end_{time} \geq row[time]$ **then**

11: $row.push[label] \leftarrow label_{time}$

12: **end if**

13: **end for**

14: **end for**

15: **end procedure**

Algorithm 5 This algorithm implements the Sliding Window Algorithm

1: **procedure** ORGANIZER($dataset$)

Require: $(lenth(dataset)) \neq 0$

2: $length \leftarrow length(dataset)$

3: $window \leftarrow 50$

4: $offset \leftarrow 10$

5: **for** $num\ in\ range(0, (length - window),\ offset$ **do**

6: $x_{data} \leftarrow list()$

7: $y_{data} \leftarrow list()$

8: **for** $index\ in\ range(num, num + window)$ **do**

9: $x_{data}.push(dataset.pop[idx, 0])$

10: $y_{data}.push(dataset.pop[idx, 1])$

11: **end for**

12: $data.push(x, y)$

13: **end for**

14: **end procedure**

REFERENCES

[1] Zhou Ren et al. “Robust part-based hand gesture recognition using kinect sensor”. In: *IEEE transactions on multimedia* 15.5 (2013), pp. 1110–1120.

[2] Yi Li. “Hand gesture recognition using Kinect”. In: *2012 IEEE International Conference on Computer Science and Automation Engineering*. IEEE. 2012, pp. 196–199.

[3] Orasa Patsadu, Chakarida Nukoolkit, and Bunthit Watanapa. “Human gesture recognition using Kinect camera”. In: *2012 ninth international conference on computer science and software engineering (JCSSE)*. IEEE. 2012, pp. 28–32.

[4] Wei Zeng, Cong Wang, and Qinghui Wang. “Hand gesture recognition using leap motion via deterministic learning”. In: *Multimedia Tools and Applications* 77.21 (2018), pp. 28185–28206.

[5] Anshul Sharma et al. “Analysis of movement and gesture recognition using Leap Motion Controller”. In: *Procedia computer science* 132 (2018), pp. 551–556.

[6] Joseph DelPreto and Daniela Rus. “Plug-and-play gesture control using muscle and motion sensors”. In: *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*. 2020, pp. 439–448.

[7] Bin Hu and Jiacun Wang. “Deep learning based hand gesture recognition and UAV flight controls”. In: *International Journal of Automation and Computing* 17.1 (2020), pp. 17–29.

[8] Yuntao Ma et al. “Hand gesture recognition with convolutional neural networks for the multimodal UAV control”. In: *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*. IEEE. 2017, pp. 198–203.

[9] [Online; accessed 7-May-2022]. URL: `https://github.com/Alvipe/Open-Myo`.

[10] [Online; accessed 7-May-2022]. URL: `https://github.com/ozancaglayan/python-emotiv`.

[11] Beau Crawford et al. “Real-time classification of electromyographic signals for robotic control”. In: *AAAI*. Vol. 5. 2005, pp. 523–528.

[12] Ari Y Benbasat and Joseph A Paradiso. “An inertial measurement framework for gesture recognition and applications”. In: *International Gesture Workshop*. Springer. 2001, pp. 9–20.

[13] Matthew DiCicco, Lenny Lucas, and Yoky Matsuoka. “Comparison of control strategies for an EMG controlled orthotic exoskeleton for the hand”. In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*. Vol. 2. ieee. 2004, pp. 1622–1627.

[14] Panagiotis K Artemiadis and Kostas J Kyriakopoulos. “An EMG-based robot control scheme robust to time-varying EMG signal features”. In: *IEEE Transactions on Information Technology in Biomedicine* 14.3 (2010), pp. 582–588.

[15] Panagiotis K Artemiadis and Kostas J Kyriakopoulos. “EMG-based control of a robot arm using low-dimensional embeddings”. In: *IEEE transactions on robotics* 26.2 (2010), pp. 393–398.

[16] Andrea Apicella et al. “High-wearable EEG-based distraction detection in motor rehabilitation”. In: *Scientific Reports* 11.1 (2021), pp. 1–9.

[17] Eike Schneiders et al. “Temporal Impact on Cognitive Distraction Detection for Car Drivers using EEG”. In: *32nd Australian Conference on Human-Computer Interaction*. 2020, pp. 594–601.

[18] S Pradeep Kumar et al. “Detecting distraction in drivers using electroencephalogram (EEG) signals”. In: *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*. IEEE. 2020, pp. 635–639.

[19] Garima Bajwa, Mohamed Fazeen, and Ram Dantu. “Detecting driver distraction using stimuli-response EEG analysis”. In: *arXiv preprint arXiv:1904.09100* (2019).

[20] Aleksandra Vuckovic et al. “Automatic recognition of alertness and drowsiness from EEG by an artificial neural network”. In: *Medical engineering & physics* 24.5 (2002), pp. 349–360.

[21] Abdulhamit Subasi. “Automatic recognition of alertness level from EEG by using neural network and wavelet coefficients”. In: *Expert systems with applications* 28.4 (2005), pp. 701–711.

[22] Perattur Nagabushanam, S Thomas George, and Subramanyam Radha. “EEG signal classification using LSTM and improved neural network algorithms”. In: *Soft Computing* 24.13 (2020), pp. 9981–10003.

[23] Cosimo Ieracitano et al. “A convolutional neural network approach for classification of dementia stages based on 2D-spectral representation of EEG recordings”. In: *Neurocomputing* 323 (2019), pp. 96–107.

[24] Hubert Cecotti and Axel Graeser. “Convolutional neural network with embedded Fourier transform for EEG classification”. In: *2008 19th International Conference on Pattern Recognition*. IEEE. 2008, pp. 1–4.

[25] Rajdeep Chatterjee and Tathagata Bandyopadhyay. “EEG based motor imagery classification using SVM and MLP”. In: *2016 2nd International Conference on Computational Intelligence and Networks (CINE)*. IEEE. 2016, pp. 84–89.

[26] [Online; accessed 7-May-2022]. URL: `https://www.bbci.de/competition/ii/`.

[27] Chia-Ju Peng et al. “An EEG-based attentiveness recognition system using Hilbert–Huang transform and support vector machine”. In: *Journal of Medical and Biological Engineering* 40.2 (2020), pp. 230–238.

[28] Guangyuan Chen et al. “Anomaly detection in EEG signals: a case study on similarity measure”. In: *Computational intelligence and neuroscience* 2020 (2020).

[29] Guangyuan Chen et al. "Anomaly detection in EEG signals: a case study on similarity measure". In: *Computational intelligence and neuroscience* 2020 (2020).

[30] Juha Kela et al. "Accelerometer-based gesture control for a design environment". In: *Personal and Ubiquitous Computing* 10.5 (2006), pp. 285–299.

[31] [Online; accessed 7-May-2022]. URL: https://en.wikipedia.org/wiki/Electromyography.

[32] [Online; accessed 7-May-2022]. URL: https://psychology.fandom.com/wiki/Electromyography.

[33] [Online; accessed 7-May-2022]. URL: https://www.ncbi.nlm.nih.gov/books/NBK279362/.

[34] J DILLING WALTE. *GRAYS ANATOMY*.

[35] [Online; accessed 7-May-2022]. URL: https://github.com/PerlinWarp/pyomyo/wiki/Myo-Placement.

[36] Ali H Al-Timemy et al. "Classification of finger movements for the dexterous hand prosthesis control with surface electromyography". In: *IEEE journal of biomedical and health informatics* 17.3 (2013), pp. 608–618.

[37] [Online; accessed 7-May-2022]. URL: https://clinicalgate.com/forearm-fractures-2.

[38] [Online; accessed 7-May-2022]. URL: https://www.ncbi.nlm.nih.gov/books/NBK279362/.

[39] [Online; accessed 7-May-2022]. URL: https://en.wikipedia.org/wiki/Electroencephalography.

[40] [Online; accessed 7-May-2022]. URL: https://en.wikipedia.org/wiki/File:International-10-20-system-for-EEG-MCN.svg.

[41] R Srinivasan. *Electroencephalography*.

[42] *Technological Basics of EEG Recording and Operation of Apparatus*.

[43] [Online; accessed 7-May-2022]. URL: https://mayfieldclinic.com/pe-anatbrain.html.

[44] [Online; accessed 7-May-2022]. URL: https://www.hopkinsmedicine.org/health/conditions-and-diseases/anatomy-of-the-brain.

[45] [Online; accessed 7-May-2022]. URL: https://en.wikipedia.org/wiki/Inertial-measurement-unit.

[46] [Online; accessed 7-May-2022]. URL: https://en.wikipedia.org/wiki/Inertial-measurement-unit.

[47] [Online; accessed 7-May-2022]. URL: https://www.vectornav.com/resources/inertial-navigation-articles/what-is-an-inertial-measurement-unit-imu.

[48] [Online; accessed 7-May-2022]. URL: https://www.vectornav.com/resources/inertial-navigation-articles/what-is-an-inertial-measurement-unit-imu.

[49] [Online; accessed 7-May-2022]. URL: https://www.vectornav.com/resources/inertial-navigation-articles/what-is-an-inertial-measurement-unit-imu.

[50] [Online; accessed 7-May-2022]. URL: https://ece.montana.edu/seniordesign/archive/SP14/UnderwaterNavigation/Quaternions.html.

[51] [Online; accessed 7-May-2022]. URL: https://ece.montana.edu/seniordesign/archive/SP14/UnderwaterNavigation/Quaternions.html.

[52] [Online; accessed 7-May-2022]. URL: https://en.wikipedia.org/wiki/Conversion-between-quaternions-and-Euler-angles.

[53] [Online; accessed 7-May-2022]. URL: https://en.wikipedia.org/wiki/Distraction.

[54] [Online; accessed 7-May-2022]. URL: https://time.com/3858309/attention-spans-goldfish/.

[55] [Online; accessed 7-May-2022]. URL: https://www.bankrate.com/insurance/car/distracted-driving-statistics/.

[56] *ROBOMASTER TT - SDK 3.0 User Guide*.

[57] [Online; accessed 7-May-2022]. URL: http://devinmancuso.com/blog/2015/unboxing-and-hands-on-with-thalmic-labs-myo.html.

[58] Antonio Pallotti, Giancarlo Orengo, and Giovanni Saggio. "Measurements comparison of finger joint angles in hand postures between an sEMG armband and a sensory glove". In: *Biocybernetics and Biomedical Engineering* 41.2 (2021), pp. 605–616.

[59] *Emotive EPOC User Manual*.

BIOGRAPHICAL STATEMENT

Debayan Datta, was born in Tripura, India. He received his B.Tech (Bachelor of Technology) degree from National Institute of Technology (NIT), Agartala, India where he worked on Natural Language Processing identifying internet humor using memes. He also worked on several Robotics and Artificial Intelligence algorithms. His primary areas of research interests are Computational Neuroscience, Neural Networks, Autonomous and Semi-autonomous Robotics.

His current field placement is with the University of Texas at Arlington where he is pursuing his Master of Science degree and his is current area of research is Intention and Distraction Classification using Brain Data for Control Applications.

He is also a student member of IEEE North-America Chapter.

Debayan Datta

Department of Computer Science and Engineering

College of Engineering

University of Texas at Arlington, United States

Email:*datta.debayan19@gmail.com*