ANALYSIS OF MACHINE LEARNING CLASSIFIER PERFORMANCE IN ADDING CUSTOM GESTURES TO THE LEAP MOTION

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Eric Yun

September 2016

COMMITTEE MEMBERSHIP

TITLE:	Analysis Of Machine Learning Classifier Performance In Adding Custom Gestures To The Leap Motion
AUTHOR:	Eric Yun
DATE SUBMITTED:	September 2016
COMMITTEE CHAIR:	Franz Kurfess, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	Christopher Lupo, Ph.D. Associate Professor of Computer Science
COMMITEE MEMBER:	Hisham Assal, Ph.D. Associate Professor of Computer Science

ABSTRACT

Analysis Of Machine Learning Classifier Performance In Adding Custom Gestures To The Leap Motion

Eric Yun

The use of supervised machine learning to extend the capabilities and overall viability of motion sensing input devices has been an increasingly popular avenue of research since the release of the Leap Motion in 2013. The device's optical sensors are capable of recognizing and tracking key features of a user's hands and fingers, which can be obtained and manipulated through a robust API. This makes statistical classification ideal for tackling the otherwise laborious and error prone nature of adding new programmer-defined gestures to the set of recognized gestures.

Although a handful of studies have explored the effectiveness of machine learning with the Leap Motion, none to our knowledge have run a comparative performance analysis of classification algorithms or made use of more than several of them in their experiments. The aim of this study is to improve the reliability of detecting newly added gestures by identifying the classifiers that produce the best results. To this end, a formal analysis of the most popular classifiers used in the field of machine learning was performed to determine those most appropriate to the requirements of the Leap Motion. A recording and

evaluation system was developed to collect recordings of gestures that could then be used to train a classification prediction model, as well as calculate the training run time and performance statistics for each of the classifiers tested. It is from these measurements made under the framework of this study that a recommendation of classifiers can be made.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

The use of visual-based gesture recognition as a form of interaction with computational devices is a continuously expanding field with significant potential. This interaction approach seeks to achieve a less invasive and more natural form of interaction, allowing a high degree of freedom and intuitive feel without the need to touch anything. The effective application of gesture recognition offers significant potential in a range of fields such as human-computer interaction, smart home systems, virtual reality, video gaming, sign language recognition, and robotics [7, 14]. With the introduction of low cost depth and time-of-flight camera devices over the past few years, such as the Microsoft Kinect and the Intel RealSense Camera, the acquisition of 3D data has become readily available to the mass market. This has made it possible for natural interfaces based on the obtained 3D data to be employable in commercial applications.

The Leap Motion, a small USB-powered controller initially released in July 2013, is one such motion sensing input device designed specifically to more accurately track hand features at the expense of the rest of the body to aid in hand gesture recognition. Equipped with two monochromatic IR cameras and three infrared LEDs, the Leap Motion is capable of tracking the hand orientation and the position of fingertips and finger joints [20]. The software provided with the

Leap Motion is capable of extracting the relevant data points without needing to use computer vision algorithms, and allows the device to recognize a small set of basic gestures out of the box. The gestures recognized by the Leap Motion API as of the version 2 release in May 2014 are a finger-drawn circle, a forward 'screen' tap, a downward 'keyboard' tap, and a sideways finger swipe.

Unfortunately, the Leap Motion cannot simply scan in and detect custom gestures with a quick button press or a single function call built into the standard API. For this device to be of use to software developers with unique development goals, it must be able to support different static and dynamic gestures of significant complexity [9]. This requires coding in the desired gestures by tracking hand features in each frame of a recorded gesture throughout the length of the recording. Developing an algorithm that can track changes in these features in each frame without losing significant accuracy as the set of gestures grows can be a difficult and error prone process. Some recent attempts to address this challenge include manipulating a Leap Motion-controlled adaptive robotic arm by comparing the current frame of an arm movement to its previous frame [1], and uniquely identifying hand gestures by treating key frames of motion trajectories as still images [11, 22].

Another increasingly popular means of tracking hand features in a more efficient and less painstaking manner is through the use of machine learning classifiers [6, 14]. By recording gestures as a list of frames, tagging each list for identification, and then serializing the data, a reasonably large training set can be

put together for use in a supervised training model using ten-fold cross validation. A classification algorithm designed for machine learning is used to build the model between human actions and computer responses to correctly identify a gesture. This model can be further improved upon by adding to the existing training data and retraining with the data.

Although a number of studies and papers have explored the effectiveness of machine learning with the Leap Motion over the past couple of years, none to our knowledge have performed a formal study of the effectiveness of the most popular classifiers for identifying hand gestures. The primary focus of this study is to improve the overall effectiveness of adding gestures through machine learning by examining which classifiers provided the best results in terms of accurate gesture identification. To this end, a sizable data set of recordings for 12 different gestures were collected and used to train prediction models with a popular machine learning suite called Waikato Environment for Knowledge Analysis (Weka). The workbench software currently provides 47 different classifiers that support cross validation, numeric attributes, and multi-value nominal classes, making them suitable for supervised learning with the gesture data set. Performance measures are tracked for each classifier to see which ones produced the most accurate prediction models in a reasonable amount of time.

The rest of this document is organized as follows: Chapter 2 provides background information on machine learning, classifications algorithms, the

Weka suite, and performance evaluation metrics. Chapter 3 discusses related works. Chapter 4 outlines the methodology used to run the study of classification performance. Chapter 5 reviews the results obtained from the study. Chapter 6 proposes possible future work for this study. Chapter 7 provides concluding remarks about this thesis.

Chapter 2

BACKGROUND

2.1 Machine Learning

Machine learning is a subfield of computer science derived from the study of pattern recognition and learning theory in artificial intelligence. It has been described as a methodology that can “detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty” [17]. The data that was collected for this study are hand gestures, which are scanned in by the Leap Motion device as a list of frames. Each gesture was tagged with the correct gesture type by the user. Each frame within each gesture contains a number of features regarding physical attributes about the motion and position of the hand. These features are directly provided by the Leap Motion, and can be obtained through method calls provided by the included API. Since a large amount of tagged user input was needed, the project made use of supervised learning, meaning that the data was already labeled before the training stage of machine learning to produce more accurate predictors.

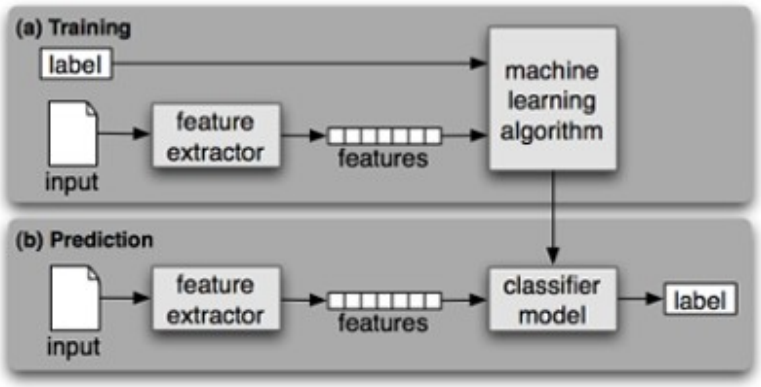


Figure 2.1 Supervised Learning Workflow [2]

Once a large enough set of user-provided hand gesture data is collected and properly tagged, the supervised learning process begins. A feature set relevant to the expected output must be extracted from the collected hand frames. Some examples of relevant features for this project include the average grip strength, the average distance between each adjacent fingertip, the ratio of the number of consecutive frames in which the hand moves forward, and the ratio of the number of consecutive frames where all five fingers of a hand are extended. A significant body of other relevant features were also used throughout the study, and will be noted in the methodology section of this document. Considering the large amount of data provided by these hand frames, selecting the most suitable features is a critical and difficult task, and remains an open problem in machine learning to date.

Once the feature set has been extracted, the data is divided into a training set and a testing set. Typically, at least half of the data set is used for training the classification model, with the remaining data used to test the classifier

afterwards. A classifier is an algorithm that attempts to predict the correct classification based on patterns in the feature set. In this study, a model validation technique called ten-fold cross validation was used to divide the feature set and train the classification model, and will be discussed further in the next section. During the training process, a classifier looks at both the features and the expected output to learn which features are the best predictors for certain outputs [19]. Once a classifier has been trained, it is applied to the testing set to make predictions without looking at the expected output. Once a prediction is made, the expected output is compared with the prediction.

2.2 Ten-Fold Cross Validation

Cross validation, also known as rotation validation or rotation estimation, is a technique for subpartitioning a data set into the necessary training and test sets, such that the entire data set will eventually be used in both subsets [10]. In k-fold cross validation, the data is first divided equally into k different subsets, also known as folds. One of these folds will be made into a test set, with the remaining folds being used as the training set. Once a classifier has been trained and made its predictions on the test set, the test set becomes part of the training set while one of the other training folds becomes the new training set. The classifier is trained once again, and the cycle repeats until all k data folds have had its turn as the test set.

The advantage of this validation method is that all observed data are used in both the training and test sets, and each observation in a data set is used for validation exactly once. This helps to address the issue of overfitting while developing an effective prediction model. Using a higher number of folds in cross validation will typically produce a more accurate predictor, although it will require significantly more data to ultimately achieve this accuracy. Ten-fold (k=10) cross validation is commonly used by researchers, since it is largely considered to have a good balance of producing good prediction results with a reasonable amount of data to train with [16].

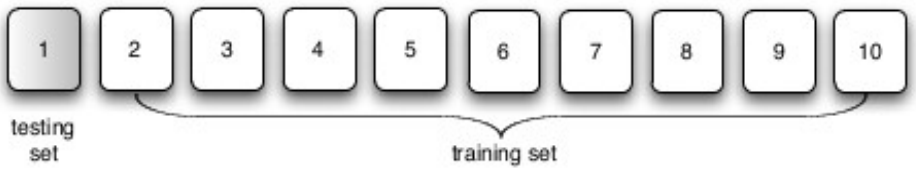


Figure 2.2 10-fold Cross Validation [10]

2.3 Classifiers

In machine learning, classification is the identification of which categories an observation belongs to. To achieve this, algorithms known as classifiers are used to concretely correlate identifying features to an observation's correct categories. Classification can be either binary, meaning that an observation can be one of only two possible classes, or multiclass, meaning that an observation can be one

of a larger range of classes. The features that are observed for classification may be categorical (e.g. plane, car, motorcycle, train), ordinal (e.g. high, medium, low) or numerical. Classifiers that were analyzed for the purposes of this thesis must support both multiclass classification and numerical features, since they must identify from a large number of possible hand gestures and handle numerical data derived from the Leap Motion.

47 different classifiers which were capable of managing the data provided by the Leap Motion and were packaged with the Weka suite were tested in this study. The analysis of the results performed in Chapter 5 indicated that Random Forest was the best overall performing classifier. As such, only Random Forest will be discussed in detail in the next section.

2.3.1 Random Forest

The random forest technique for classification is derived from an earlier meta-algorithm called bootstrap aggregating, also known as bagging. To address the weakness of overfitting inherent in decision trees, bagging generates a large 'forest' of decision trees and takes the mode of the output of the entire set of trees [23]. Given a training set *S* of size *n*, bagging generates *m* new training subsets by sampling from *S* with replacement, where *m* is equal to the desired number of trees generated to create the forest. Typically, between 64 and 128

trees are used within the machine learning community, and the Weka suite defaults to *m*=100 for its forest generation process [18].

By sampling with replacement, some observations may be repeated in multiple subsets, and the odds of selecting any one observation are not altered in subsequent selections. A decision tree is generated from each of the subsets, where each node corresponds to a feature and each leaf corresponds to either a class or the probability distribution over the list of possible classes. Each tree is formed by splitting its source set by its best predicting features via a top-down greedy algorithm, repeating until splitting no longer adds value to the prediction.

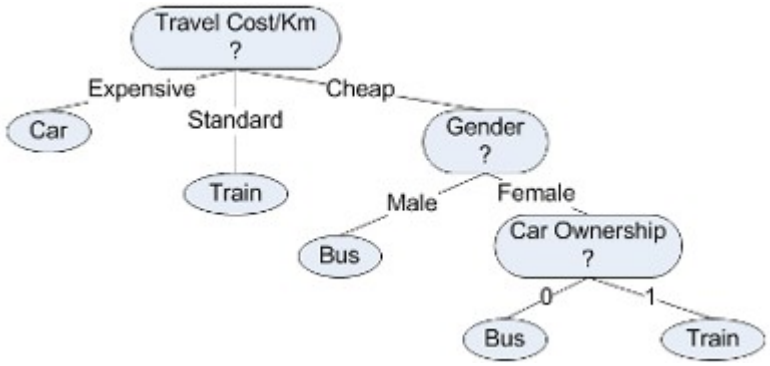


Figure 2.3 Decision Tree Example

Random forests operate on the same scheme as bagging, generating a forest in the same way with only one major difference. Instead of selecting from the entire set of features at each candidate split of the tree generation process, the random forest algorithm selects from a random subset of the features at each split [21]. Typically, if one or a few features in the entire feature set are very

strong predictors for the class output, these features will be selected in a disproportionate number of *m* trees in the forest. This alteration of the selection step is intended to address the issue of heavy correlation of trees in a forest using the standard bagging method. The typical size of a feature subset from the total feature set of size *f* is the square root of *f* rounded down.

2.4 Performance Measurements

Once a classification model has been trained, performance metrics are necessary to measure how accurate its predictions are. Four commonly used measurements in machine learning are accuracy, precision, recall, and F-score [12]. These values are calculated from the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) generated by the classifier during testing. Positives are defined as being selected by the classifier, while negatives are defined as not selected by the classifier [8].

Accuracy represents the overall correctness of a classifier's predictions, and is calculated by dividing the number of correct classifications by the total number of classifications.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Figure 2.4 Accuracy Formula

Accuracy alone is not a sufficient measure of a classifier's performance, as it is vulnerable to outliers in the data set. For example, if a data set has very few observations of a particular type, and a classifier never predicts that type under any circumstances, a high accuracy score can still be achieved since the number of TN would be very high.

Precision and recall are additional metrics that can be used to supplement the accuracy score. Precision is calculated by dividing the number of correct classifications of a type by the total number of predicted classifications of that type. This is particularly useful when the user desires reliable results from the classifier. Recall is calculated by dividing the number of correct classifications of a type by the total number of instances of that type. This metric is important when the user wishes to capture the largest number of correct results.

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

Figure 2.5 Recall And Precision Formulas

To better balance the results of precision and recall, a weighted harmonic mean of the two values is often used. This mean, called the F-score, F1 score, or F-measure, is calculated as follows:

$$F1 = \frac{2PR}{P+R}$$

Figure 2.6 F-Score Formula

If either the precision or recall is considered to be the more important of the two, than a weight can be applied to adjust the F-score accordingly. However, the precision and recall are usually weighted equally, resulting in what is called a Balanced F-score [12].

2.5 Waikato Environment For Knowledge Analysis (Weka)

Weka is a popular suite of machine learning tools and algorithms designed for data mining tasks and publicly available. It is developed by University of Waikato, New Zealand, and is currently licensed under the GNU General Public License. It provides data pre-processing, classification, and visualization tools for data analysis, as well as a large collection of classification algorithms that can be called from Java code or accessed via a user-friendly GUI. The popularity of Weka within the machine learning community, combined with its extensive functionality provided through its Java packages, make it an ideal machine learning engine for classifier analysis. This study uses version 3.613 of Weka, which at the time of implementation was the most recent stable release.

Chapter 3

RELATED WORKS

Although there are currently no known studies which extensively catalog classifier performance with the Leap Motion, there are a handful of academic papers covering experimental systems which explore the overall effectiveness of applying machine learning principles to improve gesture recognition performance. Each of these systems utilize different classification algorithms and collect a unique set of gestures to stress test the Leap Motion's functional capabilities. These works provide both the inspiration for this work and a solid foundation with which to build on to improve the Leap Motion's performance. This section briefly covers the implementation, strengths, and weaknesses of each of these systems.

3.1 American Sign Language Recognition System

Chuan et al. developed a Leap Motion-based recognition system which detects the 26 letters of the American Sign Language (ASL) alphabet in 2014 [4]. Over 7900 observations, each of which were 5 seconds in length, were collected from two faculty members at the University of North Florida, one of whom was a deaf person in deaf education. 5 different features were collected throughout the

recording process for machine learning, namely pinch strength, grab strength, average distance, average speed, and average tri-spread. 4-fold cross validation was used for supervised classification training.

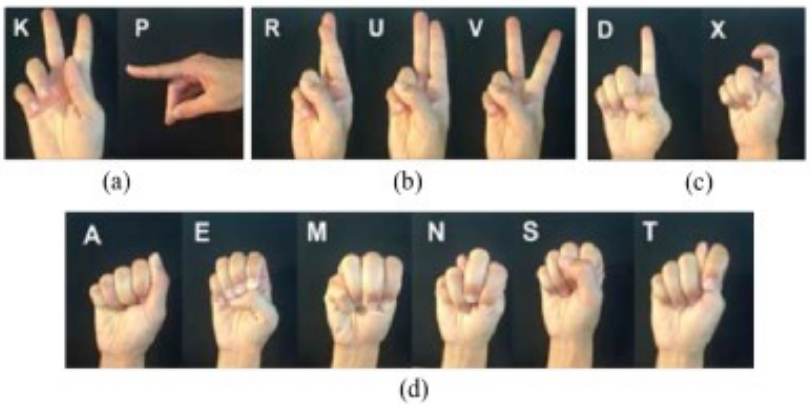


Figure 3.1 Example Signs For Letters In ASL

K-nearest neighbor (k = 7) and support vector machines (SVM) were used to classify the letters, with results showing highest average accuracy rates of 72.78% and 79.83% respectively. Although the collection source size was very small, a respectively large data set was collected (over 300 per gesture) for each gesture, and the gesture set was significantly large enough to stress the capabilities of the classifiers and the developed classification system. Unfortunately, the nature of ASL is such that only one of the gestures involves real time movement during a recording, with the rest being static gestures.

3.2 Arabic Sign Language Recognition System

A very similar sign language detection system was developed by Elons et al. and applied to a set of 50 different gestures representing words in the Arabic Sign Language (ArSL) [5]. The size of the data set that was collected, the feature set used, and the participants in the study were not stated in their academic paper. The main difference from the work performed by Chuan et al. is that a Multi-Layer Perceptron neural network (MLP) was selected as the classifier of choice for training. The reported results indicate an average accuracy rate of 88% using MLP, although direct comparisons between the works of Chuan and Elons are difficult due to the use of widely different gesture sets.

3.3 CNN Gesture Recognition System

McCartney et al. developed a Leap Motion gesture recognition system which utilized convolutional neural networks (CNN) to classify the data set [15]. 100 participants from the staff and student body of the Rochester Institute of Technology recorded 9600 observations (800 per gesture) over a set of 12 different gestures. The gesture set consists of a one finger tap, two finger tap, swipe, wipe, grab, release, and a pinch, as well as a finger drawn check mark, figure 8, lower case 'e', capital case 'E', and capital case 'F'. Recordings generally lasted around 100 to 200 frames at 100 frames per second, and 15

different features were stated to be recorded for each frame. The results from the use of CNN as the classifier show an accuracy rate of 92.4%. The paper written on this system discussed exploring other classifiers such as the hidden markov model, but was unable to get far in that direction at the time of writing.

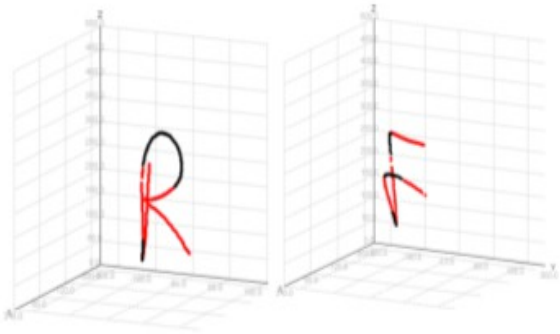


Figure 3.2 Gestures As Sequence Of Lines Detected By Leap Motion

3.4 Rapid Dynamic Gesture Recognition System

Chen et al. developed a separate Leap Motion gesture detection system which relied on SVM as its main classifier, and was designed to correctly classify a gesture before the full gesture had been performed [3]. Data was collected for 36 different gestures, which consist of finger drawings of Arabic numerals (0 to 9) and the English alphabet capitalized (A to Z). 3600 observations (100 per gesture) were recorded, although it is not stated how many participants contributed to the data set. 70% of the data set was used for training and the

remaining used for testing, but cross-validation was not used. Which features are tracked are not indicated in their academic paper.

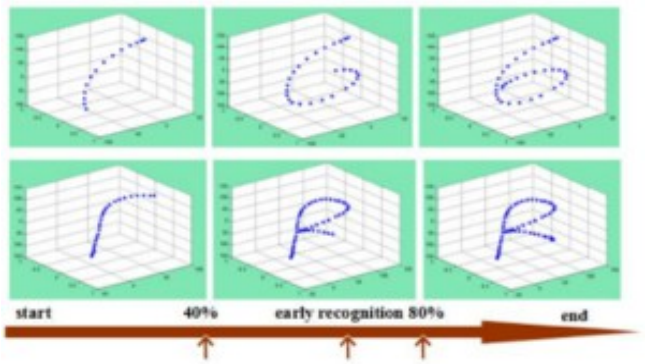


Figure 3.3 Early Recognition Of Dynamic Gestures

Separate classification models were developed using 50%, 70%, and then 80% of each recorded observation in the data set. This was done to see how well the Leap Motion can correctly and rapidly classify gestures without reading in the entire gesture. The average classification accuracy rates when trained on 50%, 70%, 80%, and 100% of the sample data using SVM were observed to be 91.63%, 94.06%, 96.03%, and 98.24% respectively. The results showed that using more of each observation in the data set resulted in diminishing returns. Unfortunately, the development team did not test out other existing classification algorithms or developed one of there own, although they did list these shortcomings as part of their future work.

Chapter 4

METHODOLOGY

4.1 Gesture Selection

A total of 12 different gestures representing a variety of dynamic hand movements were used in this study. The goal was to test classifier performance with a number of gestures of varying complexity, which fall into two main categories. The first category consists of 6 single-finger drawings of various shapes and signs, where certain features such as the number of fingers held up or whether the palm is closed would not help uniquely identify gestures. The second category consists of 6 hand gestures which are commonly used in everyday interaction and are clearly distinguishable from one another.

Only right-handed gestures were recorded and used during the training portion of this study. This was done to simplify and expedite the data collection process, as a left handed gesture must be classified as a separate gesture from its right handed counterpart while still needing to be mapped to the same action. Ultimately, making new gesture classifications for the left hand would not have added to this project.

4.1.1 Finger-Drawn Dynamic Gestures

This set of 6 gestures consists of various shapes and letters drawn by a single finger of an otherwise closed-fist hand. These gestures are also of varying complexity to test classifier performance, ranging from simple triangles that typically last not much more than 100 frames, to more complex shapes that approach or even exceed 300 frames. Participants were shown drawn images with direction arrows, and were asked to draw the same shapes with one finger only. They were encouraged to vary the performance of each recorded gesture in terms of speed, location, and size, to reflect the performance variations of the same gesture that might take place in practical usages of the Leap Motion. The most complex of these gestures are not likely to be utilized in any practical applications of the Leap Motion, but are not so complex as to be ridiculous or difficult to perform. Sample visual images of these performed gestures read in by the Leap Motion can be found in Appendix A.

4.1.2 Full Hand Dynamic Gestures

This set of 6 gestures consists of commonly used hand motions to express particular ideas with people or perform particular actions with certain items. They were in part selected for their features which help to uniquely identify gestures from one another. The gestures used in this set are a hand wave, a downward

hand chop, a sideways hand slap, an a-ok gesture, a closed-fist punch, and a thumb-index finger pinch. Sample visual images of these performed gestures read in by the Leap Motion can be found in Appendix B.

Participants providing data were asked to perform gestures as they best saw fit, so long as people could clearly and readily identify the gestures. Gestures could be performed anywhere within the detection range of the Leap Motion, and they could be performed at whichever speed the participant wished so long as it was within reason. That said, recorded gestures in this set were generally between 100 and 200 frames in length.

4.2 Feature Selection

This project collects a number of features that are provided by the Leap Motion API to improve the overall performance of the generated classification models. These features provide direct information regarding the positioning and movement of the recorded hand and fingers. A total of 23 different features regarding the fingers, palm, and entire hand were utilized. Some of these features, such as the number of frames the entire hand was closed or the number of frames the hand moved to the right, were used as ratios over the total number of frames in a gesture to better handle gestures of varying length. The table of features used can be found in Appendix C.

4.3 Data Collection

A total of 11 different participants consisting of students on the campus of the California Polytechnic State University volunteered to contribute to this project. They were asked to perform each of the 12 gestures a number of times within the recording range of the Leap Motion. A total of 3000 observations evenly split over the set of gestures (250 observations per gesture) were collected. To avoid potential overfitting issues with the data, a concerted effort was made so that data collection could be as evenly distributed among the participants as possible. Furthermore, participants were encouraged to perform gestures in different spots of the Leap Motion's camera field, and to vary the speed and space required to perform their gestures within reason.

To both aid participants in correctly performing the requested gestures and provide active feedback, a visualizer provided with the Leap Motion SDK was displayed on a screen. This allowed participants to see how the Leap Motion was detecting the hand, fingers, and joints in real time, and served as visual confirmation of the gestures they were performing. Furthermore, it served to show if the Leap Motion was not properly recording a gesture in cases where there were clear and significant differences between the actual gesture performed and the detected observation displayed on the screen.

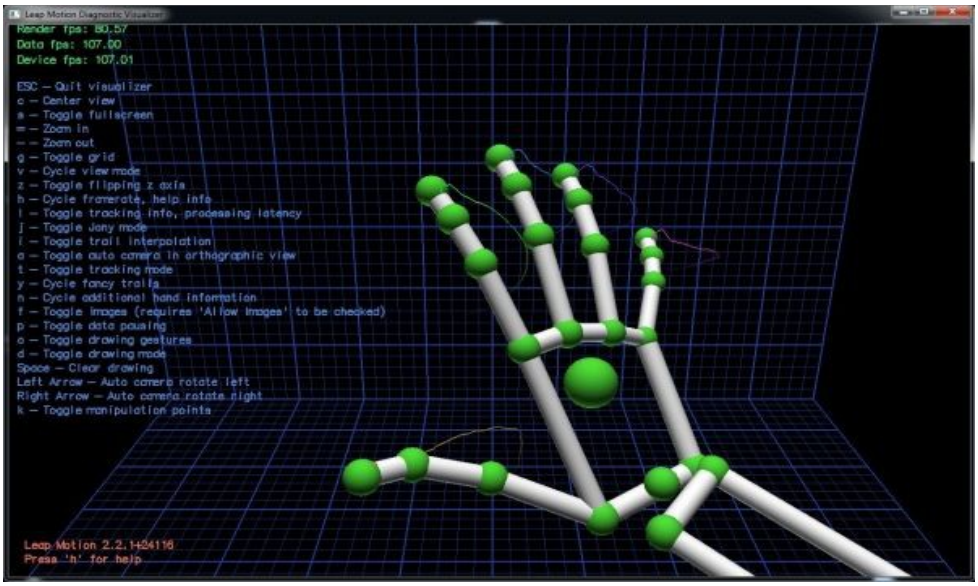


Figure 4.1 Visualizer Screenshot

Possibly due to hardware limitations in the Leap Motion cameras, a small but notable portion of gestures were not detected properly. The visualizer would show that the cameras would sometimes start recording halfway through a gesture, cut off before the end of a gesture, or detect a completely different (and sometimes humanly impossible) gesture then the one actually performed. Some gestures were more error prone than others, which added to the difficulty for the participants and ultimately limited the amount of data that could be collected. These erroneous recordings were not included in the data set of 3000 observations, as the study is not designed to address hardware issues. As a result, performance scores in this study are expected to be significantly higher than what would be achieved in practical applications of the Leap Motion device.

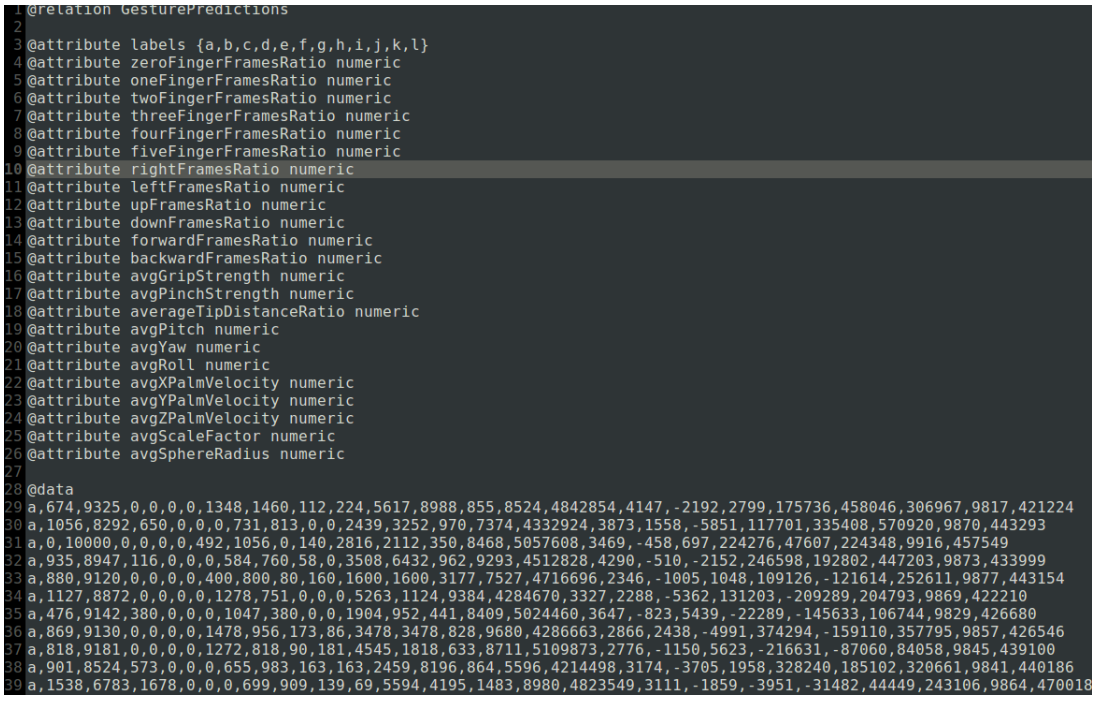
4.4 Labeling

Observations recorded by the Leap Motion can only be saved as a list of Frame objects, which are defined in the Leap Motion API. In order to perform feature extraction for use in classifier training, the frame lists must be serialized into byte arrays. These observations must then be labeled in order to correctly identify them. These labels are obtained during recording, where the end of each performed gesture is followed up by a user prompt to name the gesture. A simple wrapper class containing a field for the gesture name given by the user is applied to each serialized list. For this study, each of the 12 gestures was assigned a letter from a to l for identification. Simple letters were chosen to expedite the manual entry of gesture names. The letter assigned to each of these gestures can be found in Appendix A and B.

Since gesture names must be manually entered, the following systematic data collection process was implemented to handle potential entry errors. Each participants was requested to record 20 observations of a gesture at a time, until all 12 gestures had been cycled through. This collection process was repeated until the participant could no longer contribute further to the study. The data set was then immediately checked for any incorrect naming errors. Since each data entry had the name of the gesture at the beginning of each line in the data file, and observations of the same gesture were saved in blocks of 20, finding and correcting incorrect names proved to be a relatively painless process.

4.5 Extraction

Once every gesture in the data set has been properly labeled, each gesture is then deserialized and mapped to its feature set. These features are extracted from each frame in the gesture and are tracked in a map of feature names to its values. This information is saved to a generated Attribute-Relation File Format (ARFF) file, which Weka requires in order to perform classification training. The ARFF was designed by the developers of Weka to hold compartmentalized header and data sections in an ASCII text file. The header section contains the names of each feature, whereas the data section contains the rows of data value known as instances.



```
@relation GesturePredictions

@attribute labels {a,b,c,d,e,f,g,h,i,j,k,l}
@attribute zeroFingerFramesRatio numeric
@attribute oneFingerFramesRatio numeric
@attribute twoFingerFramesRatio numeric
@attribute threeFingerFramesRatio numeric
@attribute fourFingerFramesRatio numeric
@attribute fiveFingerFramesRatio numeric
@attribute rightFramesRatio numeric
@attribute leftFramesRatio numeric
@attribute upFramesRatio numeric
@attribute downFramesRatio numeric
@attribute forwardFramesRatio numeric
@attribute backwardFramesRatio numeric
@attribute avgGripStrength numeric
@attribute avgPinchStrength numeric
@attribute averageTipDistanceRatio numeric
@attribute avgPitch numeric
@attribute avgYaw numeric
@attribute avgRoll numeric
@attribute avgXPalmVelocity numeric
@attribute avgYPalmVelocity numeric
@attribute avgZPalmVelocity numeric
@attribute avgScaleFactor numeric
@attribute avgSphereRadius numeric

@data
a,674,9325,0,0,0,0,1348,1460,112,224,5617,8988,855,8524,4842854,4147,-2192,2799,175736,458046,306967,9817,421224
a,1056,8292,650,0,0,0,731,813,0,0,2439,3252,970,7374,4332924,3873,1558,-5851,117701,335408,570920,9870,443293
a,0,10000,0,0,0,0,492,1056,0,140,2816,2112,350,8468,5057608,3469,-458,697,224276,47607,224348,9916,457549
a,935,8947,116,0,0,0,584,760,58,0,3508,6432,962,9293,4512828,4290,-510,-2152,246598,192802,447203,9873,433999
a,880,9120,0,0,0,0,400,800,80,160,1600,1600,3177,7527,4716696,2346,-1005,1048,109126,-121614,252611,9877,443154
a,1127,8872,0,0,0,0,1278,751,0,0,0,5263,1124,9384,4284670,3327,2288,-5362,131203,-209289,204793,9869,422210
a,476,9142,380,0,0,0,1047,380,0,0,1904,952,441,8409,5024460,3647,-823,5439,-22289,-145633,106744,9829,426680
a,869,9130,0,0,0,0,1478,956,173,86,3478,3478,828,9680,4286663,2866,2438,-4991,374294,-159110,357795,9857,426546
a,818,9181,0,0,0,0,1272,818,90,181,4545,1818,633,8711,5109873,2776,-1150,5623,-216631,-87060,84058,9845,439100
a,901,8524,573,0,0,0,655,983,163,163,2459,8196,864,5596,4214498,3174,-3705,1958,328240,185102,320661,9841,440186
a,1538,6783,1678,0,0,0,699,909,139,69,5594,4195,1483,8980,4823549,3111,-1859,-3951,-31482,44449,243106,9864,470018
```

Figure 4.2 Example ARFF Text File

4.6 Training

Weka operates by taking in a labeled set of gesture data, which is split into a training set and a testing set. In this study, data sets of 1200, 1800, 2400, and 3000 observations were used to examine if there were any significant differences in classifier performance as the data set used for training and testing grew. Every observation used in the smaller data files were also used in the larger data files, with additional recordings added to the smaller data files to create the larger data files. These data file sizes were selected to keep the number of observations for each of the 12 gestures equal, meaning that the files contained 100, 150, 200, and 250 observations of each gesture respectively.

The training set produced in ten-fold cross validation is used to perform the actual training, although the portions of the data set thrown into each of the ten folds can be randomized. Weka allows this feature by taking in a seed to determine the randomization of the data allocation process. To determine whether this randomization produces wildly divergent performance results after training, 5 randomly generated seeds were passed into Weka in order to produce 5 different training and test runs for each of the 4 data sets. It is both desired and expected that there should be no significant differences among the different runs in the performance results, which are analyzed in detail in Section 5.

As previously mentioned, Weka provided 47 compatible classifiers at the time of implementation. A classification model with a system of rules generated

during training had to be created for each classifier. The performance of these models are evaluated using the test set by essentially hiding the gesture labels during testing. While the list of classifiers covered in this study is not comprehensive, it does cover some of the most commonly used classifiers as provided by one of the most popular machine learning suites in use. Furthermore, Weka is regularly maintained and continually adds more classifiers to its library, meaning that it is likely to remain relevant and suitable for further testing for many years to come. The listing of every classifier used in this study can be found in Appendix D.

4.7 Evaluation

Once training has been completed and produced a classification model for every classifier, the performance of these models with a set of unlabeled data must be assessed. The test fold of the data set provides the unlabeled data with which to evaluate performance. Features are extracted in the same way they were during training, and are passed to each classifier. The predictions for each gesture found in the test set are compared with the actual label for each gesture, and performance is evaluated based on these comparisons. All performance metrics for each run were printed to a CSV file for analysis.

In addition to calculating the performance metrics for each classification model, this study also measured the run time of the training and testing stages

for each model with a data set of 1200, 1800, 2400, and 3000 observations. The ability of a classifier to handle large amounts of data in a reasonable amount of time becomes crucial as data sets in the real world continue to grow. Development concerns in machine learning may be such that very slight increases in prediction performance at the cost of significant runtime increases may not be worthwhile.

Chapter 5

RESULTS

The study generated a significant body of results which were analyzed to better understand classifier performance with different kinds of gestures in tandem with the Leap Motion. Scores from each of the test runs were compared and were found to be very consistent across all metrics and classifiers. The largest variance in performance scores never exceeded 3%, and most were within 1%. This proved true even between runs of different data sizes, indicating that classifiers gained diminishing returns despite increased run times after a certain data size is reached. However, several classifiers had issues with the data set of 1200 observations, and the Logistic classifier would not run with such a small data set. The table of performance metrics for each classifier in each run can be viewed in the link provided in Appendix D.

5.1 RQ1: Which Gestures Worked Best For This Study?

One of the more notable findings from the results was that most classifiers did extremely well in accurately predicting the second set of gestures consisting of full hand movement. This set of gestures generally had a wide range of uniquely identifying features that distinguished gestures from one another. With few

exceptions, most performance scores were generally within the 90th percentile, with a handful of the best performing classifiers reaching 100% scores for some gestures. While the high scoring can be attributed in part to the exclusion of faulty gesture recordings from the data set, it is also likely that the gestures in this set were too unique from one another to be much of a challenge for any halfway competent classifier.

Table 5.1 Sample Of Classifier Results For Full-Hand Gestures

	Gesture G (Wave)			Gesture H (Punch)		
Classifier	**P**	**R**	**F**	**P**	**R**	**F**
Bagging	96.12%	99.20%	97.64%	99.19%	98.40%	98.90%
Best-First Tree	95.64%	96.40%	96.02%	98.80%	98.80%	98.80%
Dagging	98.40%	98.40%	98.40%	98.01%	98.40%	98.20%
Random Forest	99.60%	99.60%	99.60%	99.21%	100.00%	99.60%
Simple Logistic	100.00%	99.60%	99.80%	99.60%	100.00%	99.80%

Thankfully, the first set of gestures containing closed-hand finger drawings proved to be a more significant challenge, and hence a more useful barometer for classifier performance. These gestures share a significant number of features in common, such as the number of raised fingers, the extremely short distance between fingertips, and the small space within the closed palm. This meant that prediction models had less uniquely identifying information to work with, and prediction accuracy dropped as a result. Although the results obtained from the

second set of gestures were not counted out in this study, further discussion in this document regarding classifier performance pertains mostly to the first set of gestures.

On average, the most complex hand gestures produced better performance scores across all of the classifiers tested. The more time and motion a particular gesture took to perform during recording, the fewer false positives and false negatives a classifier generally suffered with that gesture as a result. It had been hypothesized before running this study that simpler and shorter gestures would be easier to accurately predict due to there being less noise to muddle through. However, it turned out that the additional frames and the larger set of uniquely identifying features improved predictions significantly.

Table 5.2 Sample Of Classifier Results For Finger-Drawn Gestures

	Gesture A (Triangle)			**Gesture B (Uppercase W)**		
Classifier	**P**	**R**	**F**	**P**	**R**	**F**
Bagging	71.26%	70.40%	70.83%	85.97%	76.00%	80.68%
Best-First Tree	61.18%	58.00%	59.55%	69.03%	74.00%	71.43%
Dagging	52.54%	24.80%	33.70%	79.79%	60.00%	68.49%
Random Forest	74.31%	75.20%	74.75%	87.30%	88.00%	87.65%
Simple Logistic	76.32%	69.60%	72.80%	90.00%	90.00%	90.00%

5.2 RQ2: What Were The Best Performing Classifiers?

Assuming that run times are not a concern, a solid set of classifiers performed very well across all gestures in this study. However, several in particular stood out among all of the classifiers used. In terms of pure predictive performance, Logistic Model Trees (LMT) came out on top. It consistently topped 90% accuracy and averaged over 80% on its overall f-score across all test runs. However, the run time for the predictive model trained with this classifier was consistently over 1200 seconds for a data set of just 3000 observations, making it too slow for most practical applications. The use of smaller data sets somewhat reduced the run time, but LMT still proved far slower than alternative classifiers.

Table 5.3 Average Results For Best Scoring Classifiers Over 5 Runs

Classifier	Time(sec)	Accuracy	Precision	Recall	F-Score
Random Forest	64.59	89.99%	89.94%	89.99%	89.95%
Decorate	1512.98	85.53%	85.60%	85.53%	85.55%
END	55.39	88.04%	88.06%	88.04%	88.04%
FT	54.36	88.83%	88.84%	88.83%	88.82%
LMT	1244.57	89.88%	89.88%	89.88%	89.87%
Logistic Regression	2167.03	88.47%	88.53%	88.47%	88.49%
Rotation Forest	78.72	89.37%	89.32%	89.37%	89.32%
Simple Logistic	225.89	88.65%	88.70%	88.65%	88.66%

A handful of classifiers with only slightly inferior performance but much more reasonable run times stood out in this study, in particular the Ensemble of Nested Dichotomies (END), Function Trees (FT), Random Forest, Rotation Forest, and Simple Logistic Regression. Of these classifiers, the best performing and one of the fastest was Random Forest. It averaged only 64.59 seconds of run time with a data set of 3000 observations, and its accuracy scored between 89.57% to 90.27% compared to LMT's range of 89.03% to 90.30% over 5 test runs. The f-score range for Random Forest was between 79.86% to 80.77% compared to LMT's range of 78.69% to 81.33%. In fact, every one of the average performance metrics for Random Forest were slightly higher than that for LMT due to a low scoring outlier run with LMT which slightly brought down it's average scores.

The overall best classifier based on these results is Random Forest due to its spectacular accuracy and small run times. However, it must be stressed that this study does not conclude that Random Forest, or any other classifier that performed well with the Leap Motion, should be considered the zenith of classifiers. Classifier performance depends heavily on the data used, and there is simply no one classifier that can be applied with consistent results across all given problems. Determining the best classifier for a particular problem is unfortunately still more of an art than science currently, and can often devolve into pure trial-and-error. A similar experiment with the Leap Motion which doesn't

track every single feature covered in this study or use a significantly different data set could potentially produce different results.

5.3 RQ3: Did Certain Classifiers Favor Particular Gesture Types?

Part of the aim of this study was to see if certain classifiers would perform better with certain types of gestures or even a particular gesture, while performing worse with other gestures compared to other classifiers. Since this study sought to improve machine learning accuracy as much as possible, it was anticipated that perhaps the best approach would be to mix and match with a set of classifiers to fit different situations and needs. However, the results showed that the best classifiers performed consistently better than other classifiers across all of the gestures tested in this study. This indicates that perhaps the best approach to machine learning is to simplify the process and stick to only one or two classifiers that are known to reliably produce good results across the board.

Chapter 6

FUTURE WORK

The primary focus of this study is to stress test classifiers in order to determine its suitability for human-computer interaction with gesture-based devices such as the Leap Motion. Considering the lack of challenge that the full hand dynamic gestures in this study provided, it would have been better to add additional gestures that were similar to each other, such as a fist pump versus a fist bump. Alternatively, static gestures up to and including sign language could be explored in future work to examine classifier performance.

This study stuck with preset features provided by the Leap Motion API with the idea that software developers wishing to integrate the Leap Motion to their products would avoid developing their own features in order to streamline development. That said, it would be worthwhile to explore additional features developers could implement to potentially improve the performance of the Leap Motion. Some example features that could be implemented include the fingertip angle, which calculates the angle of each finger in relation to the hand orientation, and the fingertip elevation, which calculates the distance of the fingertips from the plane corresponding to the palm region [13].

A major limitation of the Leap Motion that was discovered during data collection was that the device had a very limited camera field range. The limited

angle of view of the cameras meant that certain gestures could not be recorded properly depending on the placement of the Leap Motion. In a simple but critical example, the device could not detect the thumb of a thumbs up gesture when the device was laid flat on the table, meaning that the gesture was indistinguishable from a simple closed fist. An idea worth considering would be putting together multiple Leap Motions to capture gestures at different angles, or even pairing up a Leap Motion with another gesture detection device, such as a Kinect or a RealSense camera.

Weka has been continuously maintained and updated throughout the implementation of this study, and now features additional classifiers which were not available at the time of implementation. Furthermore, a number of user-developed plugins have recently been released for Weka, adding additional classifiers not normally included with the standard release such as Convoluted Neural Network and Hidden Markov Models. This study aims to cover as many relevant classifiers as possible, and the training code can easily be added to for future work.

Chapter 7

CONCLUSION

This project sought to examine the most suitable classifiers for adding developer-defined gestures to the Leap Motion as accurately and efficiently as possible. A simple system was developed to store recordings from different participants, then use the data to train, test, and evaluate the classification models. The results showed that certain classifiers were found wanting due to poor prediction accuracy and/or significantly slow run times. Furthermore, a small subset of classifiers performed particularly well and are suitable for development with the Leap Motion. Although Random Forest shone the most in this particular study, developers have a number of classifier options to fall back on, even if they extract a slightly different feature set or add a different set of custom gestures.

Previous known studies had experimented with pairing machine learning with the Leap Motion, but only explored a handful of classifiers at best. By contrast, this document covers a formal study of a large set of commonly used classifiers to statistically establish the best performing classifiers. This study will hopefully serve as a foundation for evaluating classifier performance in a number of development scenarios involving gesture based devices such as the Leap Motion.

BIBLIOGRAPHY

[1] Bassily, D., Georgoulas, C., Guettler, J., Linner, T., & Bock, T. (2014, June). Intuitive and adaptive robotic arm manipulation using the leap motion controller. In ISR/Robotik 2014; 41st International Symposium on Robotics; Proceedings of (pp. 1-7). VDE.

[2] Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python*. " O'Reilly Media, Inc.".

[3] Chen, Y., Ding, Z., Chen, Y. L., & Wu, X. (2015, August). Rapid recognition of dynamic hand gestures using leap motion. In Information and Automation, 2015 IEEE International Conference on (pp. 1419-1424). IEEE.

[4] Chuan, C. H., Regina, E., & Guardino, C. (2014, December). American Sign Language Recognition Using Leap Motion Sensor. In Machine Learning and Applications (ICMLA), 2014 13th International Conference on (pp. 541-544). IEEE.

[5] Elons, A. S., Ahmed, M., Shedid, H., & Tolba, M. F. (2014, December). Arabic sign language recognition using leap motion sensor. In Computer Engineering & Systems (ICCES), 2014 9th International Conference on (pp. 368-373). IEEE.

[6] Fujita, K., Shiga, K., & Takahashi, H. (1997). Analysis of FES-induced upper limb motion for machine learning control. In Engineering in Medicine and Biology Society, 1996. Bridging Disciplines for Biomedicine. Proceedings of the 18th Annual International Conference of the IEEE (Vol. 1, pp. 445-446). IEEE.

[7] Hsieh, C. C., Liou, D. H., & Lee, D. (2010, July). A real time hand gesture recognition system using motion history image. In Signal Processing Systems (ICSPS), 2010 2nd International Conference on (Vol. 2, pp. V2-394). IEEE.

[8] Karita, S., Nakamura, K., Kono, K., Ito, Y., & Babaguchi, N. (2015, June). Owner authentication for mobile devices using motion gestures based on multi-owner template update. In Multimedia & Expo Workshops (ICMEW), 2015 IEEE International Conference on (pp. 1-6). IEEE.

[9] Katan, S., Grierson, M., & Fiebrink, R. (2015, April). Using Interactive Machine Learning to Support Interface Development Through Workshops with Disabled People. In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (pp. 251-254). ACM.

[10] Kim, S., Zhang, H., Wu, R., & Gong, L. (2011, May). Dealing with noise in defect prediction. In Software Engineering (ICSE), 2011 33rd International Conference on (pp. 481-490). IEEE.

[11] Kumarage, D., Fernando, S., Fernando, P., Madushanka, D., & Samarasinghe, R. (2011, August). Real-time sign language gesture recognition using still-image comparison & motion recognition. In Industrial and Information Systems (ICIIS), 2011 6th IEEE International Conference on (pp. 169-174). IEEE.

[12] Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Vol. 1. No. 1. Cambridge: Cambridge university press, 2008.

[13] Marin, G., Dominio, F., & Zanuttigh, P. (2014, October). Hand gesture recognition with leap motion and kinect devices. In Image Processing (ICIP), 2014 IEEE International Conference on (pp. 1565-1569). IEEE.

[14] Marin, G., Dominio, F., & Zanuttigh, P. (2015). Hand gesture recognition with jointly calibrated Leap Motion and depth sensor. Multimedia Tools and Applications, 1-25.

[15] McCartney, R., Yuan, J., & Bischof, H. P. (2015, January). Gesture Recognition with the Leap Motion Controller. In Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV) (p. 3). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

[16] McLachlan, Geoffrey, Kim-Anh Do, and Christophe Ambroise. *Analyzing microarray gene expression data*. Vol. 422. John Wiley & Sons, 2005.

[17] Murphy, Kevin P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[18] Oshiro, Thais Mayumi, Pedro Santoro Perez, and José Augusto Baranauskas. "How many trees in a random forest?." *International Workshop on Machine Learning and Data Mining in Pattern Recognition*. Springer Berlin Heidelberg, 2012.

[19] Porter, R., Theiler, J., & Hush, D. (2013). Interactive machine learning in data exploitation. Computing in Science & Engineering, 15(5), 12-20.

[20] She, Y., Wang, Q., Jia, Y., Gu, T., He, Q., & Yang, B. (2014, December). A real-time hand gesture recognition approach based on motion features of feature points. In Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference on (pp. 1096-1102). IEEE.

[21] Wang, Z., Lai, C., Chen, X., Yang, B., Zhao, S., & Bai, X. (2015). Flood hazard risk assessment model based on random forest. *Journal of Hydrology*, *527*, 1130-1141.

[22] Wenjun, T., Chengdong, W., Shuying, Z., & Li, J. (2010, March). Dynamic hand gesture recognition using motion trajectories and key frames. In Advanced Computer Control (ICACC), 2010 2nd International Conference on (Vol. 3, pp. 163-167). IEEE.

[23] Zhao, Y., Chen, F., Zhai, R., Lin, X., Wang, Z., Su, L., & Christiani, D. C. (2012). Correction for population stratification in random forest analysis. *International journal of epidemiology*, dys183.

APPENDICES

Appendix A

FINGER DRAWN DYNAMIC GESTURES

This appendix contains images of each of the 6 gestures that make up the second gesture set used for this project. The images were obtained from the Visualizer program included with the Leap Motion API, and reflect what the Leap Motion cameras capture from the user.

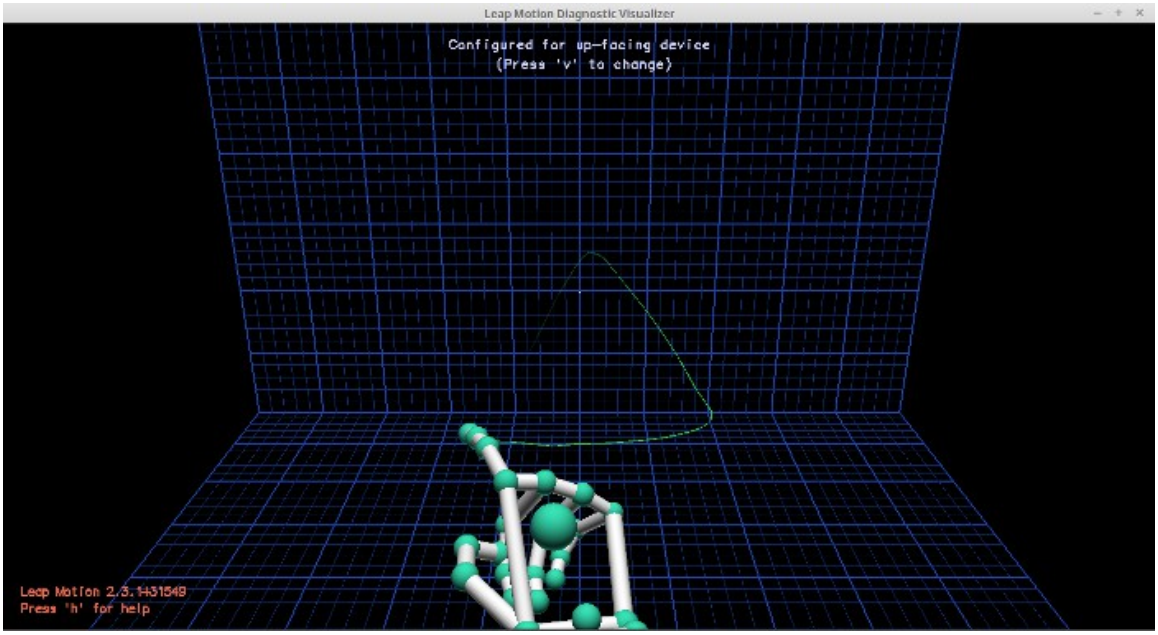


Figure A.1 Gesture A (Triangle)

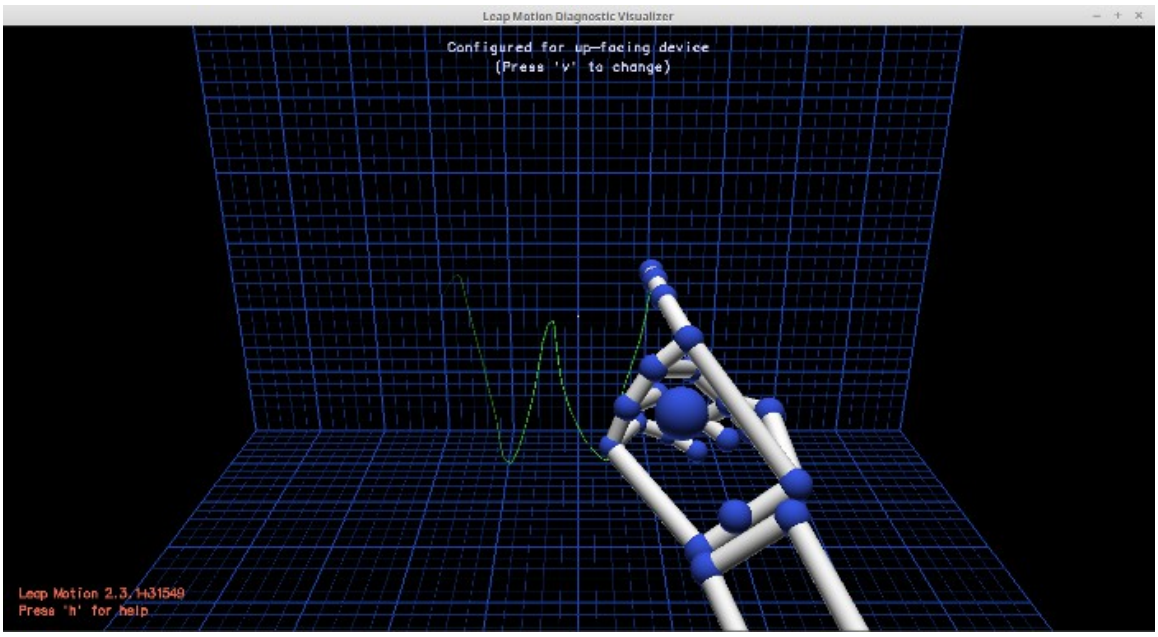


Figure A.2 Gesture B (Uppercase W)

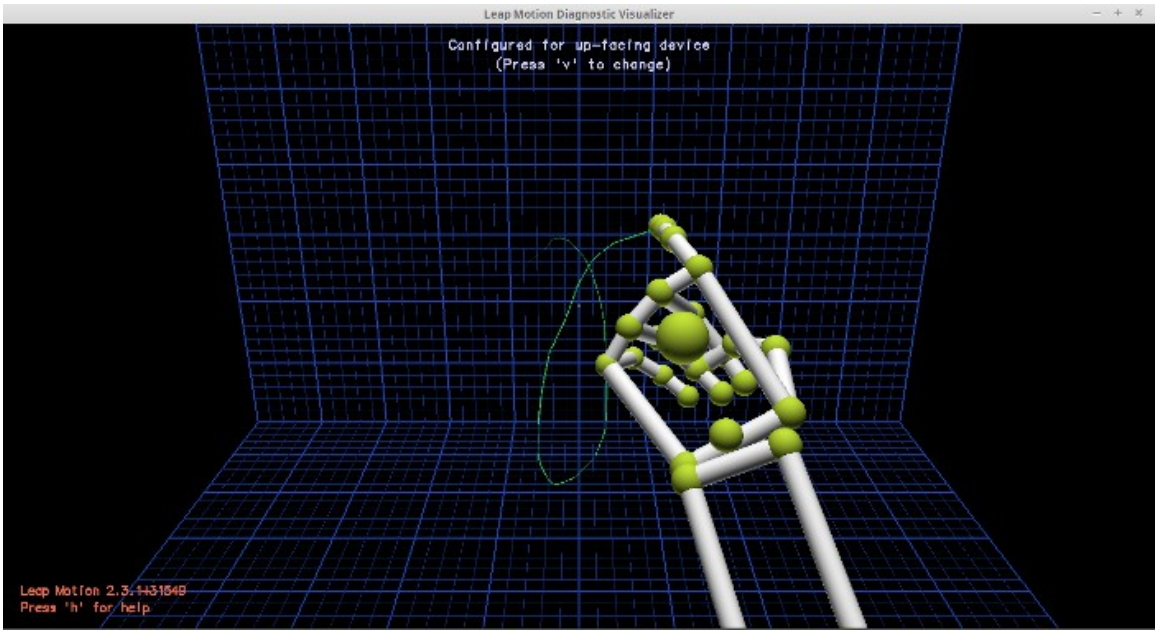


Figure A.3 Gesture C (Loop)

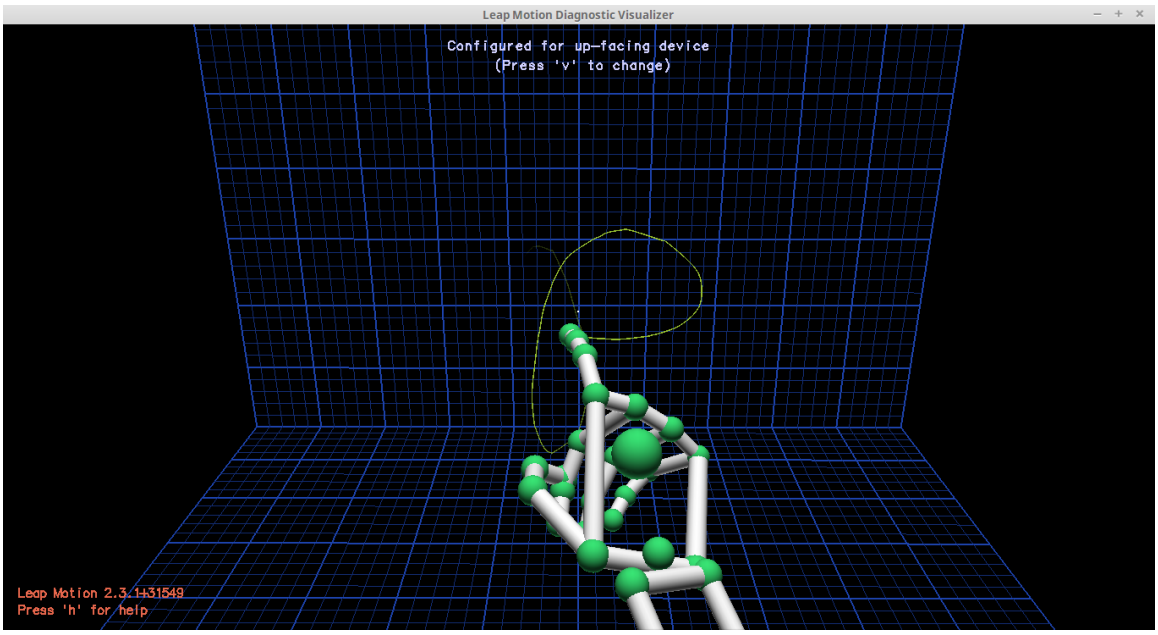


Figure A.4 Gesture D (Uppercase P)

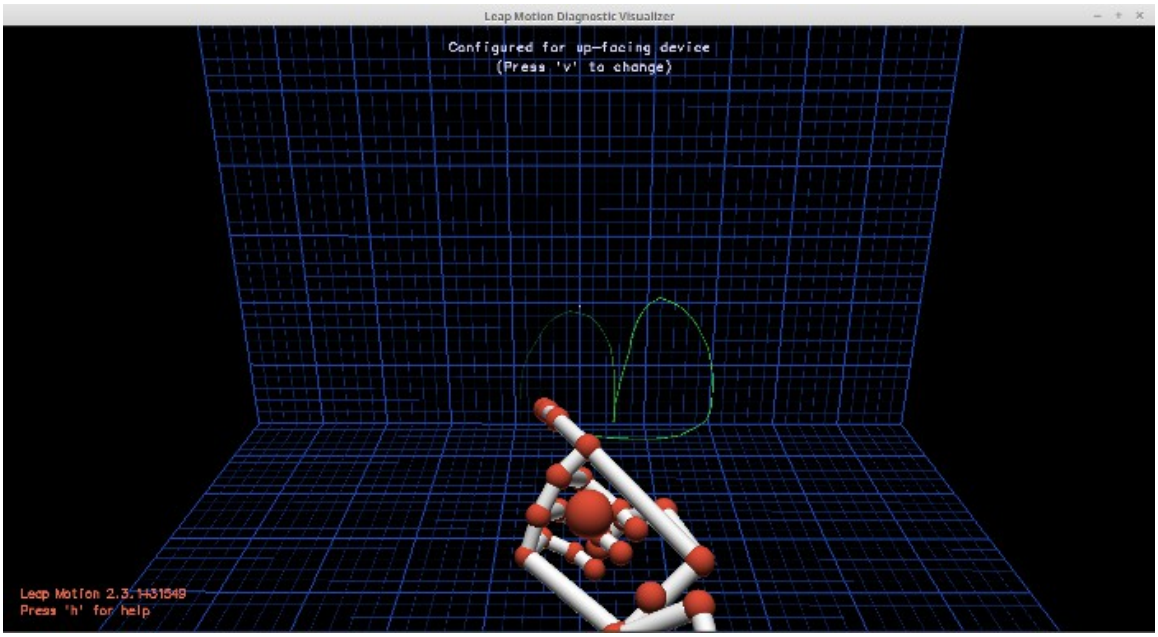


Figure A.5 Gesture E (Squiggle 1)

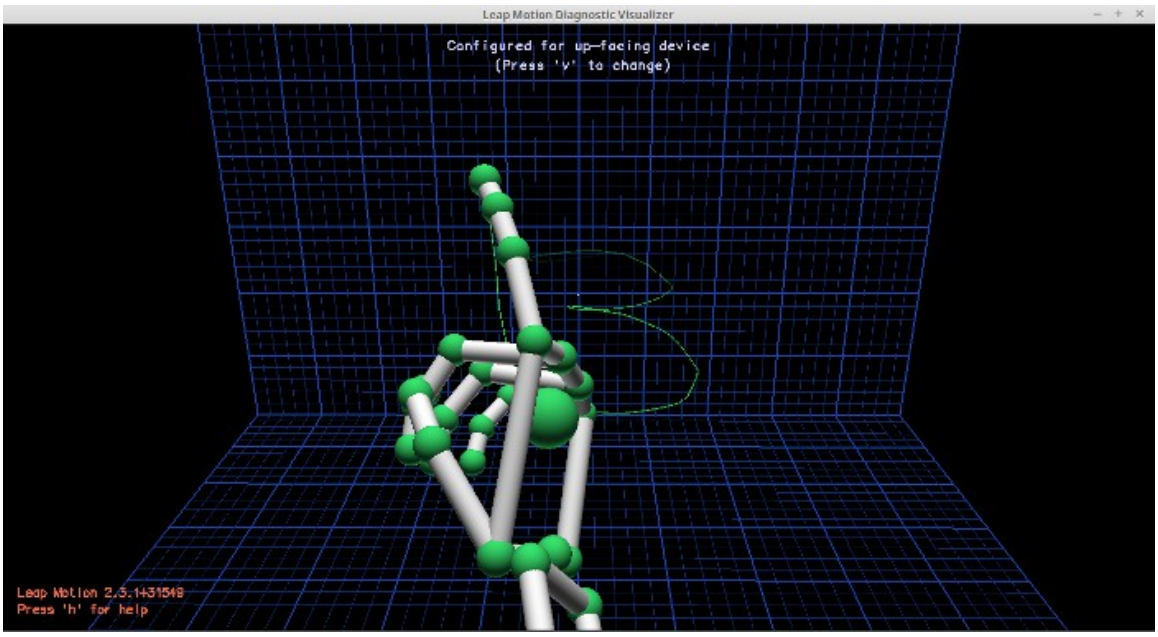


Figure A.6 Gesture F (Squiggle 2)

Appendix B

FULL HAND DYNAMIC GESTURES

This appendix contains images of each of the 6 gestures that make up the first set of gestures used for this project. The images were obtained from the Visualizer program included with the Leap Motion API, and reflect what the Leap Motion cameras capture from the user.

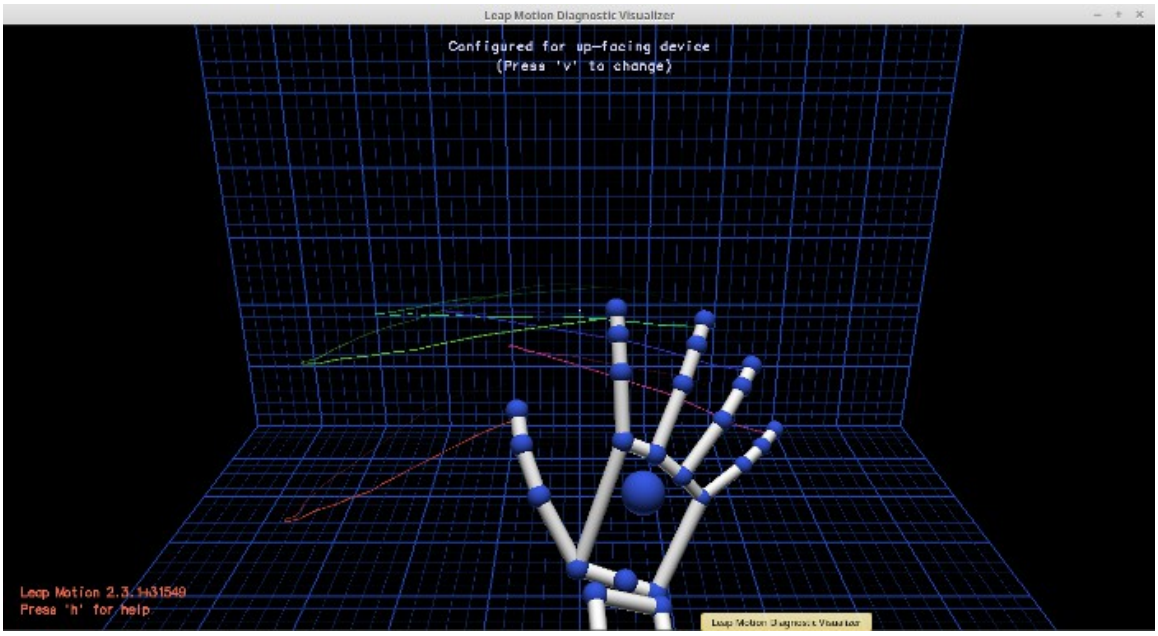


Figure B.1 Gesture G (Wave)

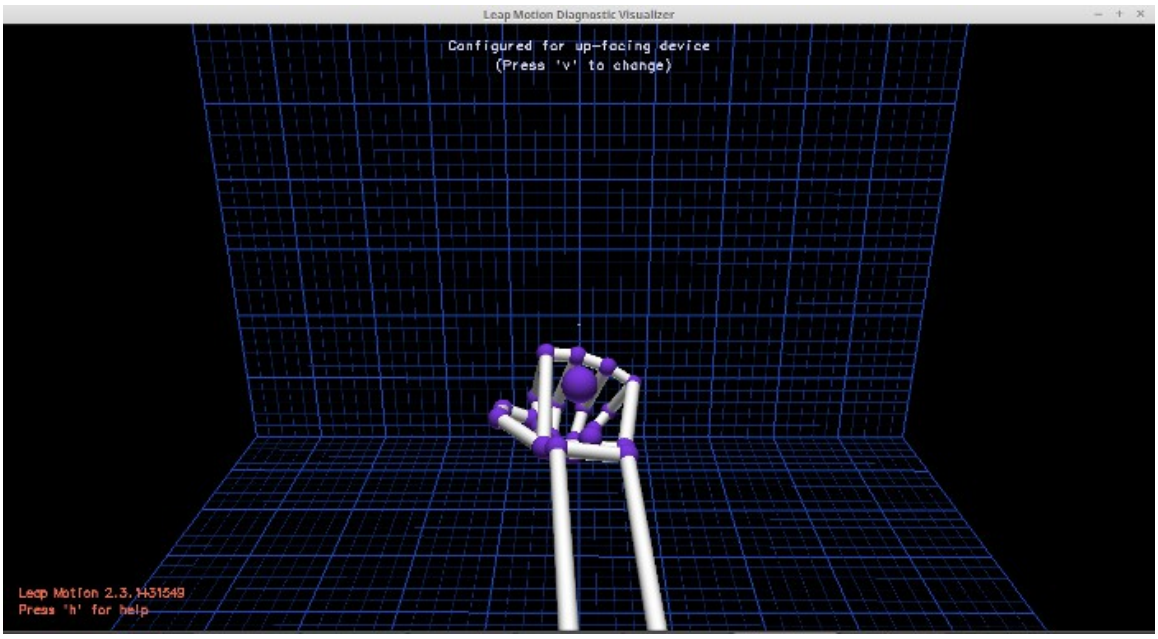


Figure B.2 Gesture H (Punch)

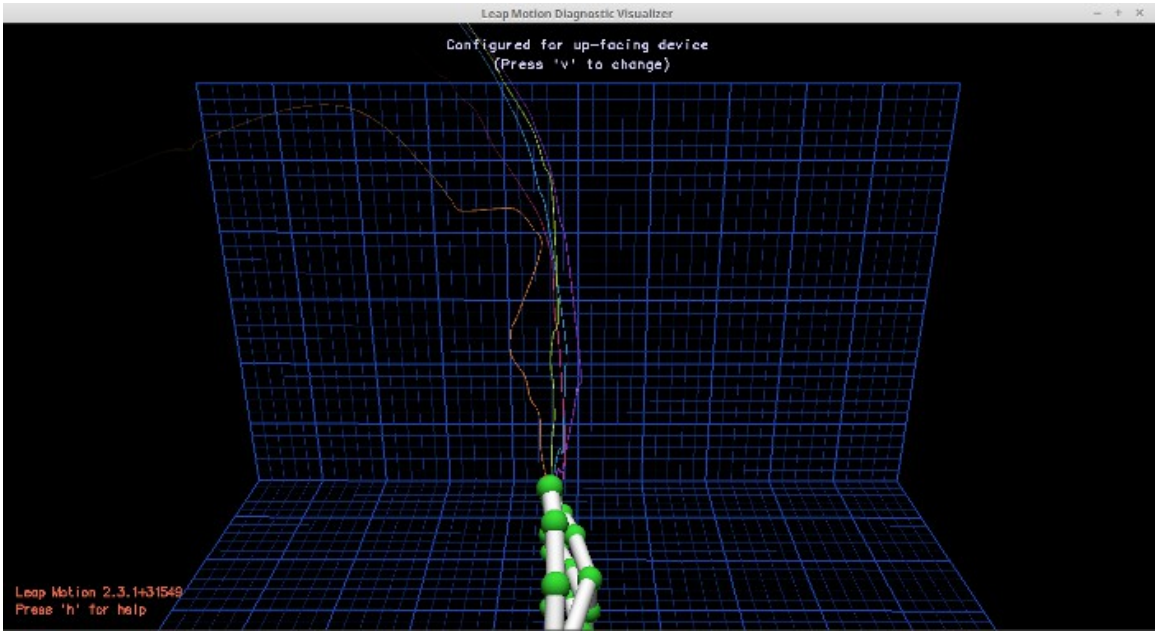


Figure B.3 Gesture I (Chop)

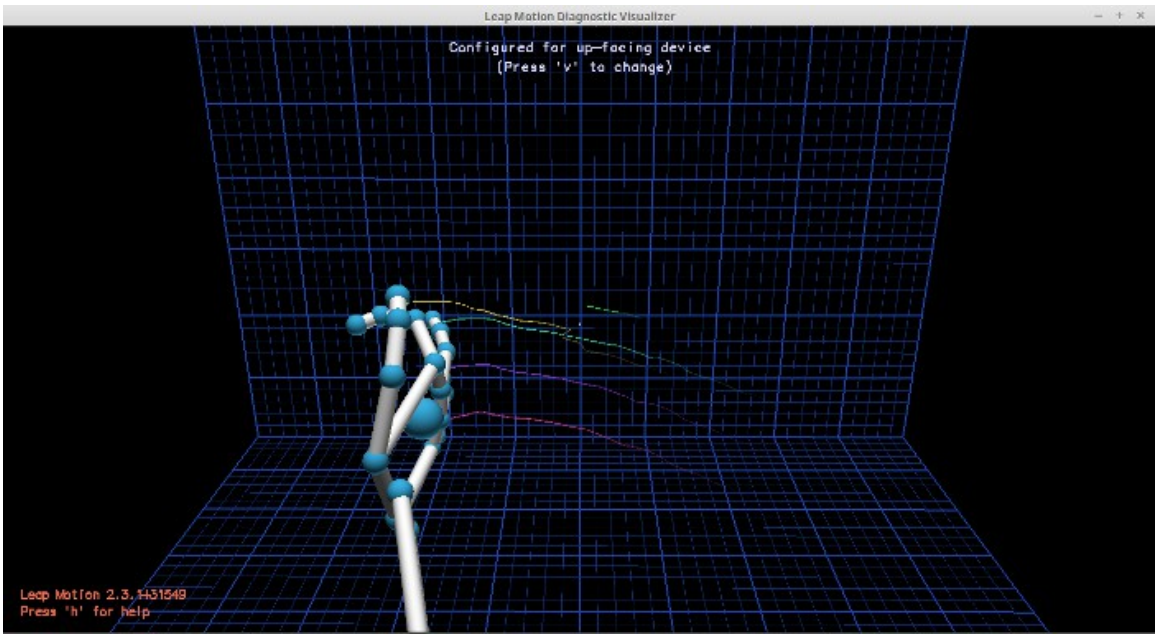


Figure B.4 Gesture J (Slap)

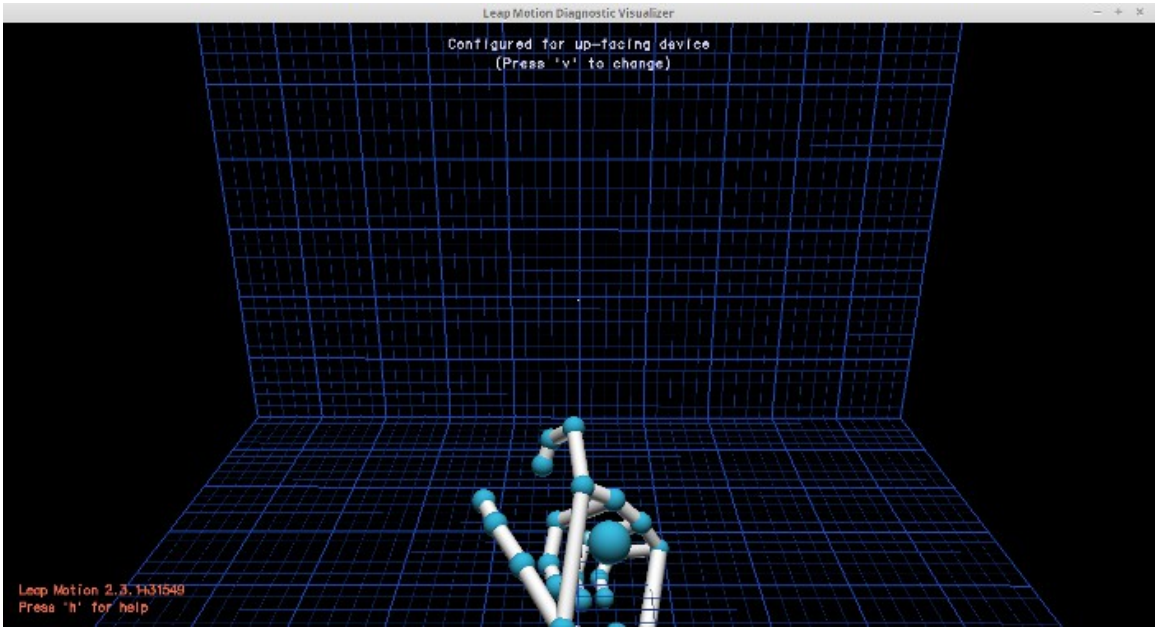


Figure B.5 Gesture K (Pinch)

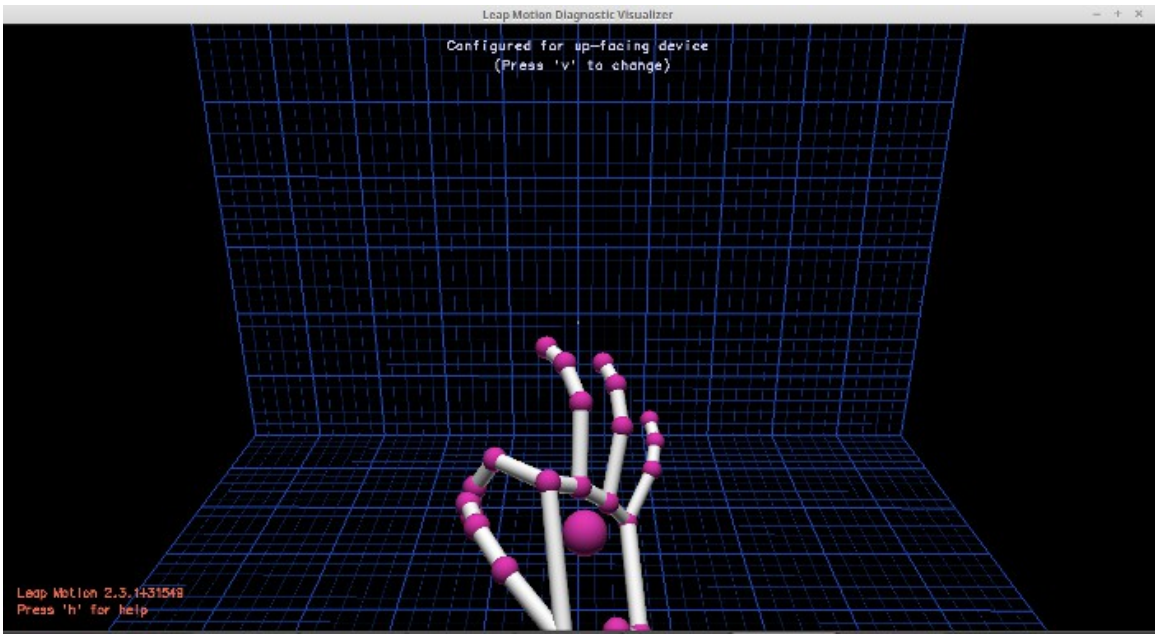


Figure B.6 Gesture L (A-Ok)

Appendix C

FEATURE SET

Features	Description
ZeroFingerFramesRatio	The ratio of frames over the entire gesture where the hand has no fingers extended
OneFingerFramesRatio	The ratio of frames over the entire gesture where the hand has one finger extended
TwoFingerFramesRatio	The ratio of frames over the entire gesture where the hand has two fingers extended
ThreeFingerFramesRatio	The ratio of frames over the entire gesture where the hand has three fingers extended
FourFingerFramesRatio	The ratio of frames over the entire gesture where the hand has four fingers extended
FiveFingerFramesRatio	The ratio of frames over the entire gesture where the hand has all five fingers extended
RightFramesRatio	The ratio of frames over the entire gesture where the hand is moving in the positive direction along the x-axis (right)
LeftFramesRatio	The ratio of frames over the entire gesture where the hand is moving in the negative direction along the x-axis (left)
UpFramesRatio	The ratio of frames over the entire gesture where the hand is moving in the positive direction along the y-axis (up)
DownFramesRatio	The ratio of frames over the entire gesture where the hand is moving in the negative direction along the xyaxis (down)
BackwardFramesRatio	The ratio of frames over the entire gesture where the hand is moving in the positive direction along the z-axis (back)
ForwardFramesRatio	The ratio of frames over the entire gesture where the hand is moving in the negative direction along the z-axis (forward)
AverageTipDistance	The average distance in mm between the fingertips throughout the gesture
AverageGrabStrength	The average grab strength measured as a float between 0 and 1 (0 for open hand, 1 for fully closed)
AveragePinchStrength	The average pinch strength measured as a float between 0 and 1 (0 for open thumb and index finger, 1 for fully

	closed)
AveragePitch	The average pitch angle (radians) of the palm with respect to the horizontal plane
AverageYaw	The average yaw angle (radians) of the palm with respect to the horizontal plane
AverageRoll	The average roll angle (radians) of the palm with respect to the horizontal plane
AverageXPalmVelocity	The average velocity of the palm along the x-axis (mm per frame)
AverageYPalmVelocity	The average velocity of the palm along the y-axis (mm per frame)
AverageZPalmVelocity	The average velocity of the palm along the z-axis (mm per frame)
AveragePalmSphereRadius	The radius in mm of an imaginary sphere fitted to the curvature of the palm of the hand
AverageScaleFactor	The average scale change between frames derived from the hand's movement over the entire gesture

Table C.1 Gesture Features

Appendix D

DATA AND RESULTS REPOSITORY

The performance metrics of each classifier for every test run, as well as the data set used to run the training and testing, has been made available at https://github.com/hobbes1021/thesis-results for public viewing. The metrics tables provide the precision, recall, and f-score for each of the 12 gestures, as well as the overall run time, accuracy, precision, recall, and f-score.