MAE527 Final Report

By R. Lott & Y. Ji

Goal

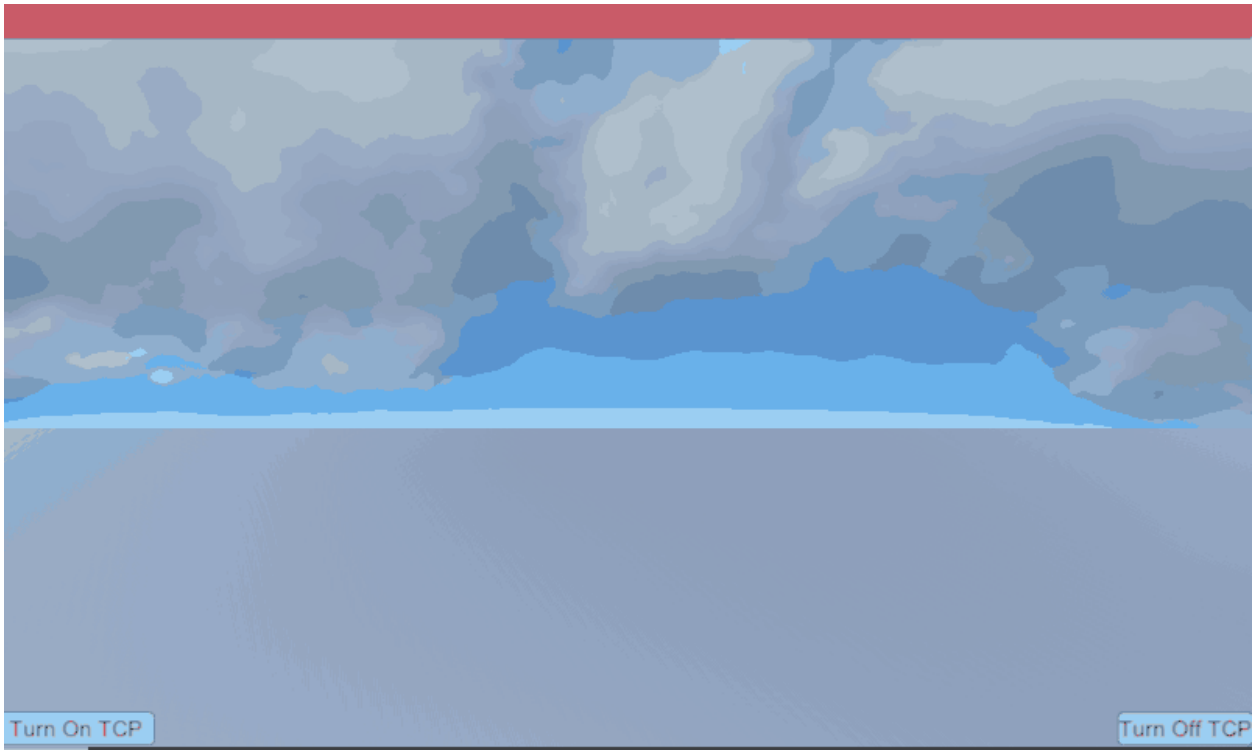
Create a gesture recognition app for 3D model creation.

Method

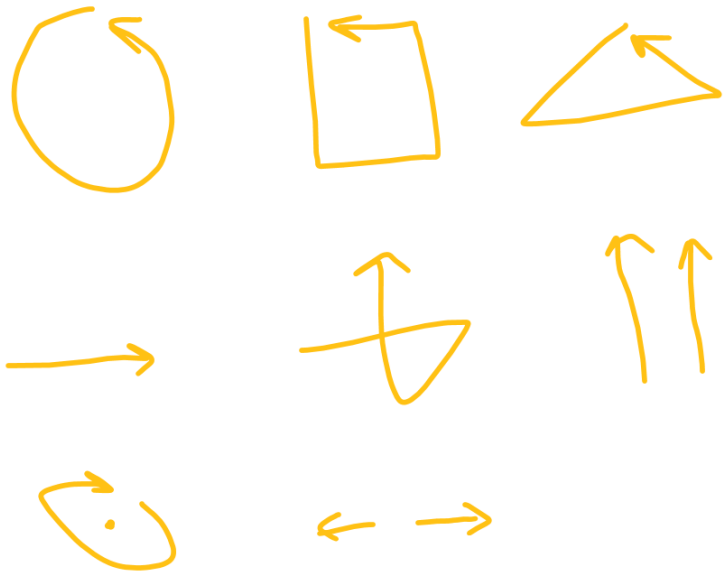
The classification of the data is done through pairwise AdaBoost. The weak classifiers used are Rubine features. 13 Rubine features were extracted from each of the 6 components of the hand (palm and the 5 fingers). Data from both the left and right hands are recorded but stored separately. When features are extracted if one hand is unavailable 0 values are used. The weighted means are trained from each hand and when classifying, which hand or hands are being used is used as a filter to limit the number of possible classes to eliminate classes.

The Leap Motion device requires a data driver meant to handle and translate the data set. Due to changes in the design philosophy the software was no longer Matlab friendly. So we began by placing a simple TCP client into Unity Game Engine and having it transmit the data freely. The application would then transmit the data to Matlab at which point we would segment the data by splitting it into Left and Right hands with sub-categories for Open and Close. Open is defined as when more than 3 fingers were parallel to one another. We opted to go with only the closed-hand data during segmentation and training as we found the most reliable gesture for cessation would be a flat open palm.

When the data was brought in, the left and right hand would be placed into a hand class, and have features extracted after flattening to 2D using either pca or mdscale depending on success or failure as not all data is good for dimensionality reduction due to data collection issues related to noise, loss of a hand, random lag spike during collection, or sudden loss of orientation. Additionally, during segmentation data was separated into open and closed groups, if a hand was briefly incorrectly detected as being closed it would count as a complete gesture.



The gestures used are the following:



From left to right, top to bottom, these gestures are:
circle: with index finger and thumb extended draw a circle.
square: with index finger and thumb extended draw a square.
triangles: with index finger and thumb extended draw a triangle.
translate: with all 5 fingers pinched, move horizontally.

poly create: with index, thumb, and middle finger extended, draw a plus.
extrude: with both hands' index finger and thumb extended, move both hands up.
rotate: with left hand's index finger pointing and right hand's all 5 fingers pinched, revolve right hand around left hand.
scale: with both hands' index finger and thumb extended, move both hands away from each other horizontally.

List of Features:

1:13 is rubine of palm
14:78 is rubine of finger tips
79 to 188 = comparison of finger positions relative to one another
189 = mean of “is left”
190 = mean of “is open”
191 = mean of “extended state”
192 = max of pinch strength
193 = max of pinch distance
194 = max of grabs strength
195 = max of grab angle

List of Rubine Features:

```
% 1st feature, cos angle of starting point
F(1) = acos(x(3)-x(1))/sqrt((x(3)-x(1))^2 + (y(3)-y(1))^2);

% 2nd feature, sin angle of starting point
F(2) = asin(y(3)-y(1))/sqrt((x(3)-x(1))^2 + (y(3)-y(1))^2);

% 3rd feature, diagonal length of Bondong Box(BB) normalized by x
F(3) = sqrt((xmax - xmin)^2 +(ymax - ymin)^2);

% 4th feature, angle of diagonal of BB
F(4) = atan2((ymax-ymin),(xmax-xmin));

% 5th feature, distance from start to end point
feature5 = sqrt((x(n-1)-x(1))^2 + ((y(n-1)-y(1))^2));
F(5) = feature5;
```

```
% 6th feature, cos angle of ending point
F(6) = (x(n-1)-x(1))/feature5;

% 7th feature, sin angle of ending point
F(7) = (y(n-1)-y(1))/feature5;

% 8th feature, total arc length
F(8) = sum(sqrt(dx.^2 + dy.^2));

% 9th feature, total arc angle
F(9) = sum(dtheta);

% 10th feature, total absolute arc angel
F(10) = sum(abs(dtheta));

% 11th feature, total square arc angel
F(11) = sum(dtheta.^2);

% 12th feature, maximum dynamic energy
F(12) = max((dx.^2 + dy.^2)./dt.^2);

% 13th feature, duration of gesture
F(13) = t(n-1) - t(1);
```

Results

A server is created to pass data from Matlab to a unity environment to visualize the tracking of the hands and record data. The data is then processed in Matlab. Initially, only the position of the palm was used for the extraction of Rubine features but when more fingers were used to classify the accuracy of recognition seemed to improve. Therefore the Rubine features of all 5 fingers, as well as the palm, were used for the classification of the gesture. A total of 78 features are used. We also considered using the number of hands used as a cascade classifier to filter out potential misclassifications. We decided to filter out classes by identifying how many hands are used and if the used hand was left or right. In order to do this, we processed the data of the left and hand data individually, and based on how many hands the input data uses, the votes generated from the left, right, or both are used. Classes that do not use the hand that is used in the input data are eliminated from the votes. The classification result has a high accuracy rate but some gestures are easily confused. More features that describe the state of the hands are added in hope of further improving the accuracy of the classifier. Such features include if the hand is a left hand, are the fingers extended or not, if the hand is open, and the maximum of the pinch strength, pinch distance, grab strength, and grab angle.

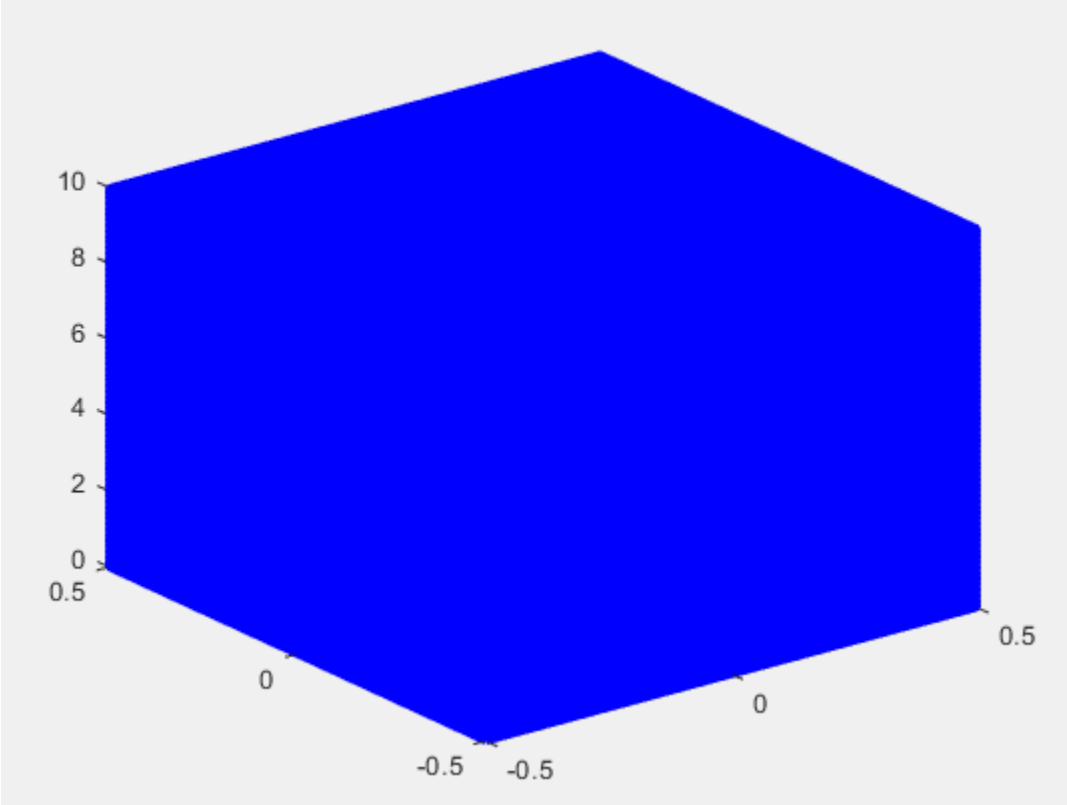
```
performance =

     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     0     1     1     1     1     0     1     1     1     1
     1     0     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     0     1     1     1     0     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     0     1     1     1     1     1     1


ans =

     6
```

When testing our AdaBoost on the sample data out of 10 samples and 9 classes, the classifier was able to successfully all but 6 samples. We also collected the relative distances of each finger relative to each other but these features did not improve accuracy and appear to decrease accuracy in some cases. In later implementations some of the classes used were dropped, the end result only used 8 classes which are included in the instructions.

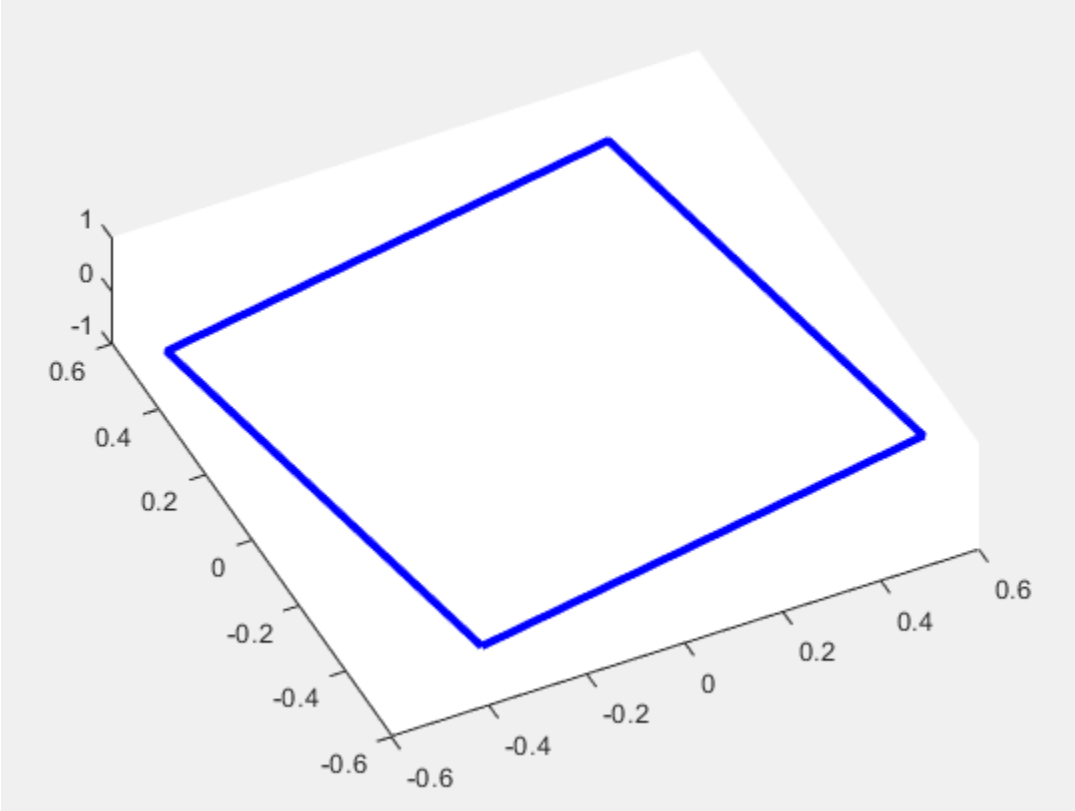
When attempting to implement a live interactive app to receive, segment, and classify data difficulties were encountered with the exchange of information among servers. Initially, the design was to have Matlab host multiple servers and both send and receive data from unity and blender. However, blender was not able to receive the data output from Matlab during live communication situations, making result visualization difficult and causing swift abandonment of the concept. To address this issue the design of the app was moved to Matlab completely. This also proves to be challenging.

In implementation, receiving a live stream of data and actively processing and handling requires an ample amount of logic to constantly modulate the rate of evaluation such that the data does not easily overwhelm Matlab and prove useless to the user. This challenge was an incredible one in its own right as neither of us was prepared for the massive amount of data from the leap motion device. Checking for a “stop” signal, looking for different problems in the data itself, segmenting, processing, and packaging the data rapidly second to second was a tremendous undertaking.

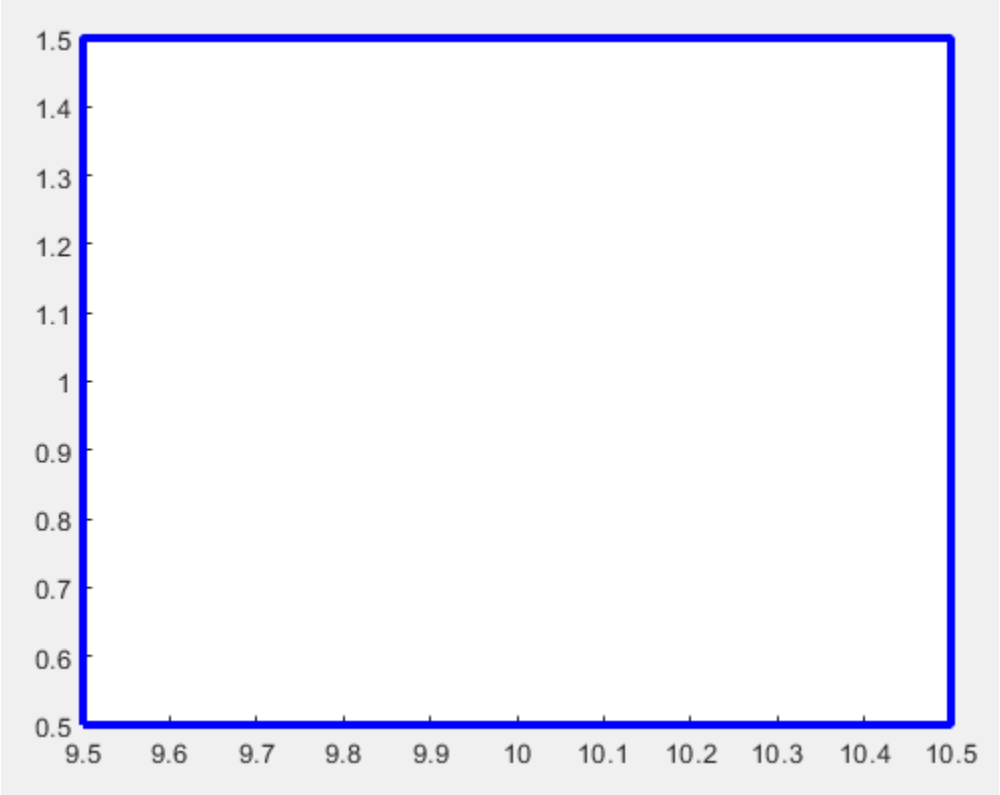
Results of extrusion



Results of rotation



Results of translation



Pseudo code for the matlab data retrieval

```
On Data Receive
If Index % 2000 == 0
	Index  = 1
dataBuffer(index) = newData
	Enter State Machine
NEUTRAL
	If Detect Open Hand > 30 times
		GetVoteFromClassifier
		If Vote == Circle → State.CREATE
CREATE
	If Detect Open Hand > 30 times
		GetVoteFromClassifier
			switch(vote):
			case Circle → DrawCircle
			case Square → DrawSquare
			case Triangle → DrawTriangle
			case Scale→ State.SCALE
			case Rotate → State.ROTATE
			case Translate → State.TRANSLATE
			case Extrude → State.EXTRUDE
ROTATE, TRANSLATE, SCALE, EXTRUDE
	Modifier*(Previous Fingertip Position - Current Fingertip Position)
		Enact Action
	If Detect Open Hand > 30 times
	State.NEUTRAL
```