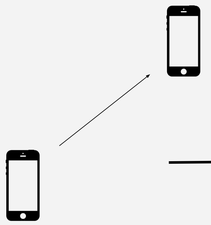
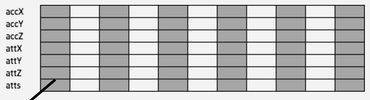
Gesture-Encrypted Location-Based Messaging

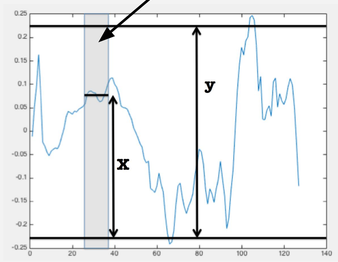
Gesture Recording and Feature Extraction



A user moves his or her phone, thus generating acceleration and attitude data.

The data generated by the user have 7 distinct attributes each with many numeric values. We refer to each of these attributes as a vector. We break each of these vectors into 11 separate segments. We only evaluate a subset of the segments (the odds) in order to avoid overfitting the data and reduce the amount of computation necessary. By selecting the odd segments we sample a fair spread of values of the entirety of the gesture and get the first and last segments.





For each of the chosen segments we compute a feature value:

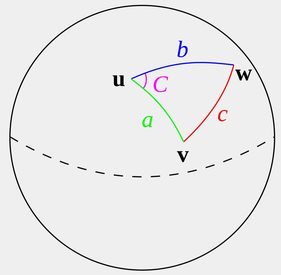
$$Feature = \frac{Segment\ Average - Vector\ Min}{Vector\ Max - Vector\ Min} = \frac{x}{y}$$

This feature represents where within the range of values the segment lies. We arrived at this feature extraction formula after experimenting with others, but this one seemed to work the best for us.

Location Tracking and App Architecture

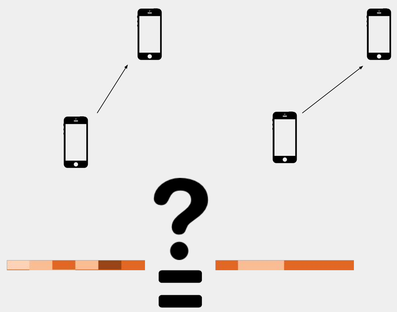
By making calls to the iPhone location manager we are able to track the location of the user. We schedule a timer for the phone to periodically ping the server with its location. The phone makes a URL request to the server every handful of seconds in order to determine whether or not the phone has come in the proximity of any locations that contain a message.

A location is determined by its latitude and longitude. If a user's reported latitude and longitude are within the proximity of a location's latitude and longitude, he or she will be notified by the app. In order to determine a proximity, a Euclidean distance between the two coordinate pairs is calculated. The Euclidean distance is determined using the haversine formula discussed previously in the semester.

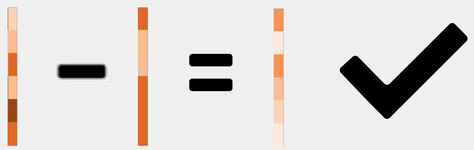


When a user is in a given location, he or she can create a message and associate it with a certain gesture. Other users of the app can attempt to read the message when they are in the proximity of the original location if they can match the gesture associated with the message.

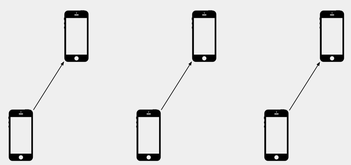
Fuzzy Matching



Now that we have a set of features to define a gesture, we can compare various gestures to determine whether or not they match. In order to decipher whether or not two gestures match, we compute a distance between the two sets of feature vectors.

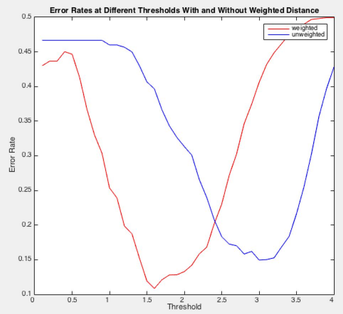


If the distance is below a threshold, we say the vectors match. We ultimately picked a threshold value by training on a subset of gesture data. For each of our predetermined gestures, we recorded the features for the gesture in multiple trials and computed the maximum distance between the pairings of trial runs. From this data we selected the maximum value of the maximum distances between our trial gestures as the threshold.



Results and Future Work

When actually determining the appropriate threshold value to choose in examining whether or not the gestures match, we calculate the "distances" between features. Success is determined using the error rate, which we defined as the average of the false positive and false negative rates. To improve our results, we ran the perceptron algorithm on our feature data to assign weights to features. This lowered the error weight from 15% for unweighted features to 11%.



On our set of 10 specific gestures, we were able to match gestures with very high accuracy. Our false negative rate, the percentage of time two matching gestures were said to not match, was 14%. Our false positive rate, the percentage of time two non-matching gestures were said to match, was 7%. We call false negatives lockouts, as the user was locked out from viewing a message he or she should have been able to see. On the other hand, we call false positives burglaries because a user is able to see a message without properly unlocking the message. We chose the threshold to favor lockouts as opposed to burglaries due to the fact that burglaries are more unfavorable than lockouts. This favors security over convenience.

We believe there are a handful of areas we could improve upon moving forward. To begin with, we could expand the gesture functionality by allowing for unique gestures, and not just the subset we currently have. Aside from messaging, we believe location-based gesturing could have other interesting applications. For example, one could easily envision future applications for unlocking cars or houses using location-based gesturing.