Software Requirements Specification

FROG Recognizer of Gestures

Team Better Recognize
Version 1.1
November 19, 2009

Revision Sign-off

By signing the following the team member asserts that he/she has read the entire document and has, to the best of his or her knowledge, found the information contained herein to be accurate, relevant, and free of typographical error.

Name	Signature	Date
Josh Alvord		
Alex Grosso		
Jose Marquez		
Sneha Popley		
Phillip Stromberg		
Ford Wesner		

Revision History

The following is a history of revisions of this document.

Document Version	Date Edited	Changes
Version 1.0	11/03/09	Initial Draft
Version 1.1	11/19/09	Specified corrections in content

Table of Contents

1. Introduction

1.1 Purpose

This document gives an overall description of the FROG Project version 1.0. It also describes the requirements, both functional and non-functional, of the FROG Project version 1.0 developed by Team Better Recognize. Functional requirements described include those for training, recognition, evaluation, and demonstration. Performance, safety, and software quality are the main non-functional requirements in this document.

1.2 Intended Audience and Reading Suggestions

This specifications document is intended for the customer and developers of this project. It ensures that the customer and Team Better Recognize are better synchronized and share the same vision.

1.3 Product Scope

The FROG Project has as its goal the development of a device and platform-independent gesture training and recognition system. In its initial release, FROG will be packaged with a plug-in for Sun SPOTs. Additional plug-ins may be written later based on a plug-in framework. In addition, FROG will contain an evaluation mode as well as a demo program for testing and demonstrating the capabilities of the project.

1.4 References

Hobermann, Rose. Durand, Dannie. HMM Lecture Notes. 2006. Carnegie Melon School of Computer Science. 10 September 2009. http://www.cs.cmu.edu/~durand/03-711/2006/Lectures/hmm-bw.pdf.

Schlömer, Thomas. Poppinga, Benjamin. Henze, Niels. Boll, Susanne. Gesture Recognition with a Wii Controller. 2008. http://wiigee.org/. 10 September 2009.

2. Definition of Terms

Accelerometer	An instrument for measuring acceleration. In particular, a 3D accelerometer measures acceleration in three dimensions.
Gesture	A continuous combination of motions made by an individual (usually with the hands) that are related and meaningful as a whole. Gestures are the entities which shall be modeled as well as recognized by the FROG project.
Hidden Markov Model	A doubly stochastic (as opposed to deterministic) math model being used to represent a gesture. Constructed HMMs are then used in recognition. "A statistical model in which the system being modeled is assumed to be a Markov process with unobserved state." This document uses the convention that an HMM λ is defined as: $\lambda = (S, O, a, b, \pi)$ where S = The set of hidden states. O = The set of possible observations (here, dealing with vectors). a = The transition matrix, where a[i][j] represents the probability of transitioning from state i to state j. b = The emission probability matrix, where b[i][k] represents the probability of emitting observation k while in state i. π = The initial condition (probability distribution for initial state).
K-means Clustering	Method of cluster analysis which aims to partition n observations into k clusters in which each observation is clustered with the nearest mean.
Sun SPOT	Sun SPOTS (Sun Small Programmable Object Technology) are small programmable wireless sensor devices developed as an experimental technology by Sun Microsystems. They contain a 180MHz 32-bit processor, 512K RAM, and a variety of sensors including a three-axis accelerometer used in the FROG project. Sun SPOTs communicate using a low-power IEEE 802.15.4 radio.
Training Instance	A training instance is a single motion of the mobile device by a user representing a gesture to be trained.
Training Session	A training session is a collection of training sets created by a user. A session is saved in a file format and reloaded to perform recognition. A training session can be thought of as a project file containing representations of multiple gestures.
Training Set	A training set is a sequence of training instances created by the user in order to train a gesture. A training set can be thought of as the complete set of data used to create an HMM representation of a gesture.

3. Overall Description

3.1 Product Perspective

FROG is a self-contained gesture recognition system. It is to be used in conjunction with plug-ins that contain device-specific code so that it has potential compatibility with any acceleration-based device.

3.2 Product Functions

The FROG Project will be a 3D acceleration-based gesture training and recognition system consisting of four main modes. These will be: Training, Recognition, Evaluation, and Demo. The Training mode will allow the user to train gestures for later use. The Recognition mode will allow the user to load previously trained gestures and perform gesture recognition using that library. The Evaluation mode allows the user to determine recognition accuracy as well as view other performance statistics to diagnose issues with their hardware or the FROG system itself. The Demo mode is a simple game to demonstrate the capabilities of the system.

3.3 User Classes and Characteristics

This product is being developed for use in an experimental, academic environment. This product is designed for use by anyone who understands how to operate a PC or Mac and reads the FROG User's Guide. The demo, however, is meant to be a fun and light-hearted way for anyone to use the system.

3.4 Operating Environment

The FROG Project was designed to operate with the following software installed

- Windows XP or later, Mac OSX, or Linux
- Java Runtime Environment 6.0 or later
- Sun SPOT SDK (for the driver; if using a Sun SPOT as a controller)

3.5 Design and Implementation Constraints

Time Constraint

    Limited by academic school year (ending on May 11, 2010)

Mobile Device Limitations

    J2ME library on Sun SPOTs
    Data communication between mobile device and host may be limited

Hardware Limitations

    System developed on older machines; this may cause performance issues

3.6 User Documentation

The complete FROG Project will come with both a User’s Guide and Developer’s Guide. These documents and more will be delivered to the project sponsor on a DVD on completion of the project. The User’s Guide will be a step-by-step guide that can walk a user through the installation and operation of the system. The Developer’s Guide will help those who might be developing a plug-in or making modifications to the existing system.

3.7 Assumptions and Dependencies

The FROG Project assumes the following:

- The end-user has a Java Virtual Machine compatible with their platform.
- The end-user has a background in gesture recognition or is at least somewhat familiar with gesture recognition technology so as to facilitate proper use of the product.

4. External Interface Requirements

4.1 User Interfaces

The user interface shall be clean and intuitively labeled to promote a high quality look that users will find easy to use. Each window shall contain an output window/pane that will show all status messages and data being processed. The user shall have the ability to select desired console and display output.

There shall also be a connection panel in each mode that allows the user to see the status of their device's connection with the system as well as help them connect/reconnect/disconnect the device.

4.2 Software Interfaces

The project shall have a plug-in based system for adding support for additional mobile devices. The plug-ins shall contain all device-specific code and any translation or handling needed for accelerometer data collected from the device. The plug-ins shall be derived from a common interface within the FROG software.

Since it is written in Java, FROG will have an obvious need to interact with a Java Virtual Machine Standard Edition 6.0 or later. Additional JVMs (such as ME or EE), if required, should be handled by plug-ins.

5. Functional Requirements

5.1 Requirements for All Modes

5.1.1 GEN-01

The system shall take 3D accelerometer readings from mobile devices as its input. From this data the system shall perform its gesture training and recognition.

5.1.2 GEN-02

The system shall *not* include internal device-specific support. Support for each mobile device shall be incorporated into a corresponding plug-in for the device and then may be added or removed from the system via a GUI option menu.

5.1.3 GEN-03

The system shall provide a console window or pane at all times to display relevant, user-selectable information.

5.2 Training Mode

In this mode the user records gestures, names them, and saves them to be used later in other modes. The user creates a series of training instances for each training set. The user’s training session will then be composed of the collection of these training sets.

5.2.1 TRA-01

The system shall limit the number of connected devices to *one* during training mode. This is to ensure that there is no ambiguity as to what device is currently feeding gestures to the training system.

5.2.2 TRA-02

The system shall provide the user with the ability to save and load training sessions for reuse. The file the system creates will be platform-independent and contain not only the gesture data but the user's name, comments (for describing what significance that particular session has), and what device plug-in was being utilized. The device plug-in will contain the device ID as well as the device type.

5.2.3 TRA-03

The system shall provide a user with an intuitive method of training gestures into the system. These gestures may be represented by a word and/or a picture or illustration.

5.2.4 TRA-04

The system shall allow the user to load an existing training session either to add additional gestures (training sets) to the file or to delete gestures (training sets) permanently from the file.

5.2.5 TRA-05

The system shall support a filtering framework with idle state and directorial equivalence filters as defaults (based on the Wiigee project). The framework will support the addition and modification of filters for each training instance.

5.2.6 TRA-06

The system shall support modification of the number of centers (k) in the k-means algorithm with a default of k = 14 (based on the Wiigee project) for each training instance.

5.2.7 TRA-07

The system shall support modification of the number of HMM states with a default of 8 states (based on the Wiigee project) for each training set.

5.2.8 TRA-08

The system shall support real-time graphing of 3D accelerometer data from user-selected connected mobile devices.

5.2.9 TRA-09

The system shall support logging of system status, incoming 3D accelerometer data, and execution time of algorithms. The user shall be given the capability to save the logged data in a user-selected file.

5.3 Recognition Mode

In this mode the user loads a previously saved training session so as to perform recognition.

5.3.1 REC-01

The system shall be able to connect and recognize the gestures of up to four devices simultaneously.

5.3.2 REC-02

The system shall allow each user/device connected to load a training session library for recognition.

5.3.3 REC-03

The system shall provide feedback *to each user* when a gesture is not recognized or, if it is recognized, report to the user what gesture it was.

5.3.4 REC-04

The system shall support logging of system status, incoming 3D accelerometer data, and execution time of algorithms. The user shall be given the capability to save the logged data in a user-selected file.

5.3.5 REC-05

The system shall support a display of available trained gestures in a particular library for a specific device.

5.4 Evaluation Mode

In this mode the user may again perform recognition, but the system shall also provide useful information for evaluating performance.

5.4.1 EVA-01

The system shall allow only one device to connect at a time.

5.4.2 EVA-02

The system shall provide real-time feedback of the performance of the recognition system. This shall include a tally of number correct, number not recognized, and number matched incorrectly, as well as average certainty.

5.4.3 EVA-03

The system shall provide the ability for the user to input the sample size (number to be requested for recognition) for each gesture from the library. Each gesture may be enabled or disabled individually, and they will be prompted either sequentially or at random based on the user's choice.

5.4.4 EVA-04

The system shall support logging of system status, incoming 3D accelerometer data, and execution time of algorithms. The user shall be given the capability to save the logged data in a user-selected file.

5.5 Demo Mode

5.5.1 DEM-01

The system shall allow up to four devices to connect and play.

5.5.2 DEM-02

The system shall contain at least one client-approved demo program to better showcase the abilities of the underlying recognition system.

5.5.3 DEM-03

The system shall keep track of each user's accuracy for evaluation of the system's (or perhaps the users') performance.

5.5.4 DEM-04

The system shall require each user to have trained the gestures needed for the demo. The users currently loaded training session file must contain gestures that share the same names as defined by the demo. A message to this effect must be displayed to the user so that he/she may go back to training mode and correct the situation.

6. Non-functional Requirements

6.1 Performance Requirements

6.1.1 PR-01

Communication between host and device shall be fast enough to support up to four connected devices. Recognition performance must not slow by more than 20% with four users connected.

6.1.2 PR-02

The speed at which a gesture is recognized shall not exceed 10ms per HMM.

6.1.3 PR-03

Code written for the Sun SPOT (i.e. for filtering) shall not hinder its ability to process accelerometer and radio transmission/reception data. Hinder is defined here as causing the threads associated with the above activities to be skipped for more than 1 period.

6.2 Safety Requirements

6.2.1 SR-01

Accelerometer-enabled mobile devices must be used with attention to surroundings. The user must not allow the accelerometer to become airborne, potentially causing injury or damage.

6.3 Software Quality Requirements

6.3.1 SQR-01

The FROG Project shall be able to recognize gestures with an 80% or better accuracy.

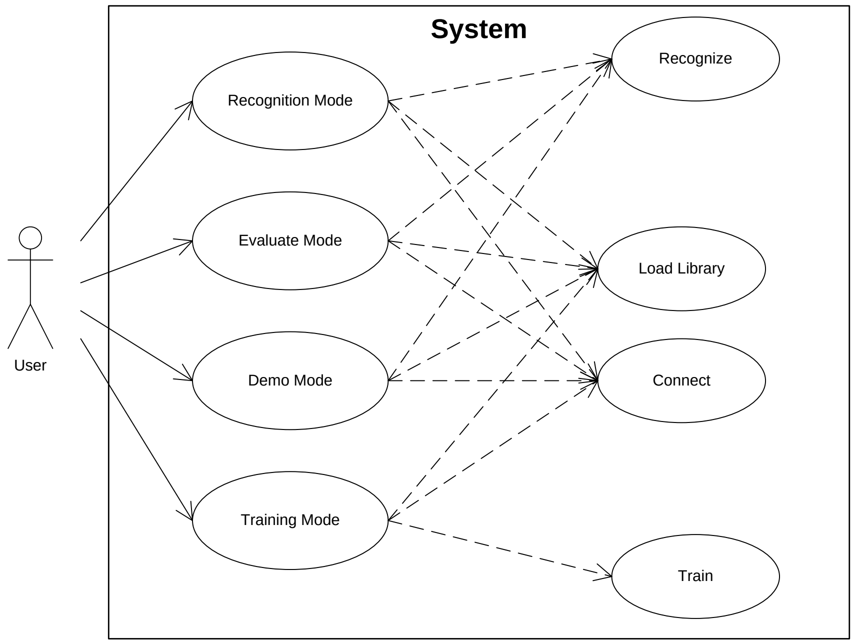
6.3.2 SQR-02

Plug-ins shall be usable to extend the versatility of the software.

6.3.3 SQR-03

FROG shall be a multi-platform framework.

Appendix A: Use Case Model



Connect

Main Success Scenario:

1. User selects Configuration option
2. System detects and displays available devices with a unique, meaningful ID
3. User can select one of the available devices to connect to
4. System formally connects to the device and adds it to a list of connected devices
5. User can disconnect and repeat from step 3
6. User selects to leave Configuration option

Demo Mode

Precondition:

Gestures to be recognized have been previously trained.

Main Success Scenario:

1. User selects Demo mode
2. System displays demo configuration window
3. User selects number of players
4. User(s) perform connection and select which libraries to use
5. User selects to begin game
6. System displays demo window and starts the game
7. User(s) follow the on-screen instructions and must make use of gestures to “win” the game
8. System displays game over and shows post-game statistics
9. User selects to leave Demo mode or play again

Extensions:

4a. User selected a gesture session that did not contain the demo’s required gestures
  - .1: System halts and informs user exactly what gestures are needed
  - .2: User may choose to be taken immediately to Training mode with the gesture session template needed for the game already selected for him/her or leave Demo mode

Evaluation Mode

Precondition:

Gestures to be recognized have been previously trained.

Main Success Scenario:

1. User selects Evaluation mode
2. User performs connection
3. User can select to load a new library
4. User selects evaluation parameters (sample size and random or sequential gesture prompting)
5. System displays a gesture
6. User makes that gesture
7. System displays information about the instance performed and keeps a running total of attempts to make a gesture
8. User can export data collected on evaluated gestures such as the number of correct and incorrect gestures as well as the average certainty
9. User can also view graphical information live about the above mentioned statistics
10. User can make the system display system messages, acceleration data, and execution time in the terminal or log it to a text file
11. User can repeat from steps 2, 3, or 5
12. User selects to leave Evaluation mode

Load Library

Main Success Scenario:

1. User chooses to load a library
2. System displays a file system window that gives the user the option of opening any previously saved library
3. System loads the gesture images, gestures names, and training data from the library

Recognize

Main Success Scenario:

1. System accepts raw acceleration data from the user
2. System applies user-chosen filters on the data
3. System compares the filtered acceleration data to computed centers for the k-means algorithm and quantizes the vectors
4. System uses the Bayes Classifier to compute the probability of the incoming gesture matching previously trained gestures in the library
5. System returns that the gesture was unrecognized if it is below a threshold or the most probable gesture if it is above the threshold

Recognition Mode

Precondition:

Gestures to be recognized have been previously trained.

Main Success Scenario:

1. User selects Recognize mode
2. User(s) performs connection
3. User(s) select(s) to load a new library
4. User(s) can choose to use the devices attached to their ID as well as choose the threshold attached to that device
5. User(s) make the gesture previously trained while holding the send button on their device
6. System responds with the name and, optionally, the image of the gesture it recognized
7. User can display system messages, acceleration data, and execution time in the terminal or log it into a text file
8. User(s) can repeat steps from 2, 3, or 4
9. User selects to leave Recognition mode

Extensions:

6a. System did not recognize the gesture

.1: System responds to user with an appropriate "unrecognized" response

Train

Main Success Scenario:

1. System accepts raw acceleration data from the user
2. System applies user-chosen filters on the data
3. System applies k-means algorithm on the current and previously filtered acceleration data to obtain quantized vectors
4. System applies the Baum Welch algorithm to create an optimized HMM

Training Mode

Main Success Scenario:

1. User selects Training mode
2. User performs connection
3. User chooses to train a new gesture set, which will require the naming of the gesture as well as the association of an image with the gesture
4. User can choose to either keep the default training settings or change training parameters such as sampling frequency and filters as well as number of centers for k means and states for the HMM
5. User can choose a file to dump raw of filtered acceleration data for further use
6. User makes a gesture while holding the send button on their device
7. User can display system messages, acceleration data, and execution time in the terminal or log it into a text file
8. User can turn on or off the different graphical displays that plot/graph vectors
9. System prompts to keep the instance or forget it
10. While want more instances, repeat from step 6
11. If there are no more instances, the user chooses to save the gesture set
12. While more gestures, repeat from step 3
13. User selects to leave Training mode or create a new session
14. System prompts the user to save session
15. System prompts for name/location of save file for session

Extensions:

3a. User wants to create gestures from a template (i.e. for compatibility with demo)

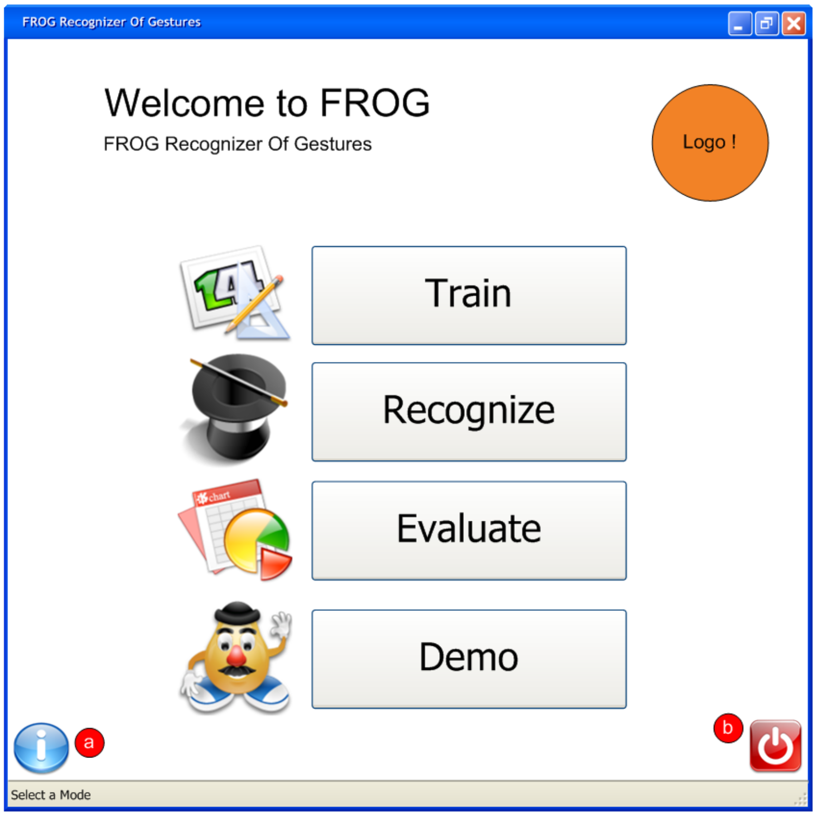
  .1: User selects to load a template gesture session

  .2: Continue like normal from step 3 and repeat for each empty gesture

Appendix B: User Interface Prototype

The following is a prototype of the FROG user interface. This prototype gives a screenshot-based walkthrough of the modes and use of the FROG Recognizer of Gestures product.

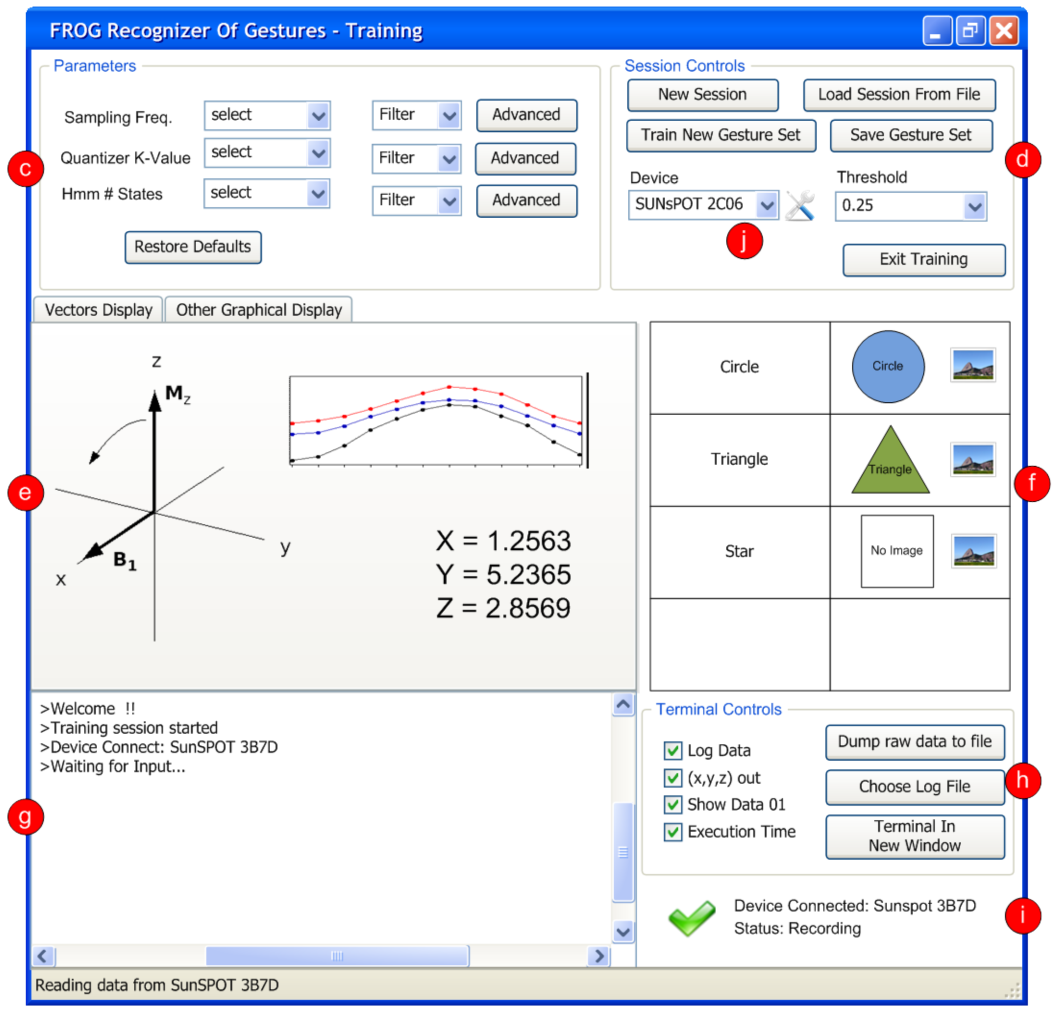
Screen 1.0: Main Menu



This is the main window of the program. From this window the user can access the different modes. In order for this window to display, it is necessary for the system to find at least one device plug-in to work with. If no plug-in is encountered, then the program will ask the user to install a plug-in.

- **a)** This button will display information about the program when clicked.
- **b)** Exit button.

Screen 2.0: Training Mode



Sampling Freq. select | Filter
Quantizer K-Value select | Filter
Hmm # States select | Filter

Device SUNsPOT 2C06
Threshold 0.25

X = 1.2563
Y = 5.2365
Z = 2.8569

```
>Welcome  !!
>Training session started
>Device Connect: SunSPOT 3B7D
>Waiting for Input...
```

☑ Log Data
☑ (x,y,z) out
☑ Show Data 01
☑ Execution Time

Device Connected: Sunspot 3B7D
Status: Recording

Reading data from SunSPOT 3B7D

This is the window for Training mode.

c) Parameters: This panel will allow the user to modify parameters for the training process.

d) Session Controls: This panel will allow the user to control the current gesture session.

- **i. New Session**: Will create a new gesture session. Each gesture session can be saved into a file.
- **ii. Load Session from File**: Will load a previously saved session from a file (using a standard open file dialog).
- **iii. Train New Gesture Set**: Will add a gesture set to the currently opened session. Upon clicking on this button, Screen 2.2 (Train New Gesture) will be displayed. The ability to train a new gesture set will not be available if the last one created has not yet been saved.

iv. **Save Gesture Set**: Will terminate the collection of instances for a gesture set and then perform the training of the gesture.

v. **Exit Training**: Will terminate the training session, close screen 2.0, and open screen 1.0.

e) **Graphical Displays**: This panel will show different graphical displays such as a graph of 3D acceleration values as well as the clusters from the k-means algorithm.

f) **Session Panel**: This panel will show the gestures that are part of the current gesture session and allow the user to load an image (using a standard open file dialog) to represent each gesture by clicking on the icon to the right of the current image.

g) **Terminal**: The terminal will serve as the primary method of providing feedback to the user in real-time about the actions being performed, performance etc. To the side of the terminal there will be controls for the user to select the feedback needed (see h.) This description applies to the terminals found in other modes.

h) **Terminal Controls**: The terminal controls will allow the user to choose the feedback he or she wants to appear in the terminal. In Training mode, there will be the option of displaying the acceleration values of each axis and performance of the algorithms involved. More controls may be added during the design process. Some other actions that those controls allow the user are:

i. **Dump Raw Data to File**: Will allow the user to choose a text file where the acceleration data received from a connected mobile device will be dumped. Note that this capability is exclusive to the Training mode screen.

ii. **Choose Log File**: Will allow the user to save the contents being posted in the terminal to a text file. Upon clicking this button, the user will be prompted with a standard save file dialog for the user to select the location and name of the text file.

iii. **Terminal in New Window**: Will display the terminal in a separate window for easier manipulation (see Screen 2.1).

i) **Device Status Display**: This panel will let the user know the current status of the mobile device used to input data to the system.

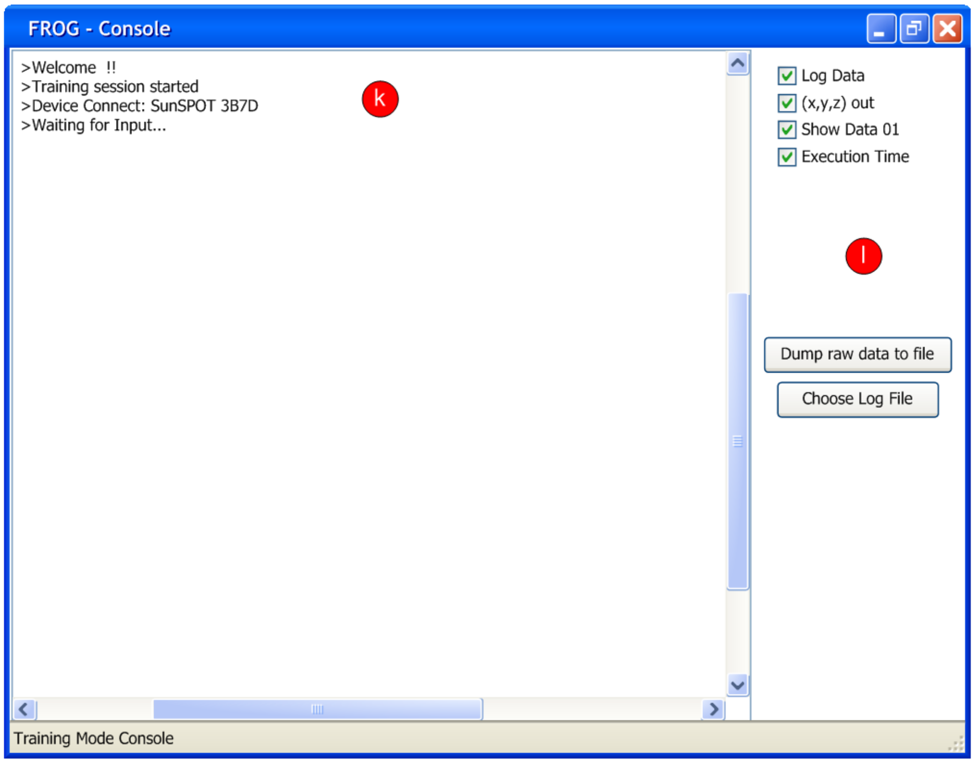
j) **Connection Panel**: This panel will give the user the option of setting up a device to work with the system. This panel will appear (with a different configuration) in other windows and will allow the user the following capabilities:

i. **Device**: Will allow the user to choose the kind of device to be set up. The choices of this combo box will correspond to the plug-ins installed.

ii. **Threshold**: Will allow the user to prescribe a threshold to be used with recognition performed with that mobile device.

iii. **Connect**: ( ) Will utilize the device's corresponding plug-in to carry out connection of the mobile device. If any further user action is necessary, the plug-in is responsible for requesting such actions from the user.

Screen 2.1: External Terminal

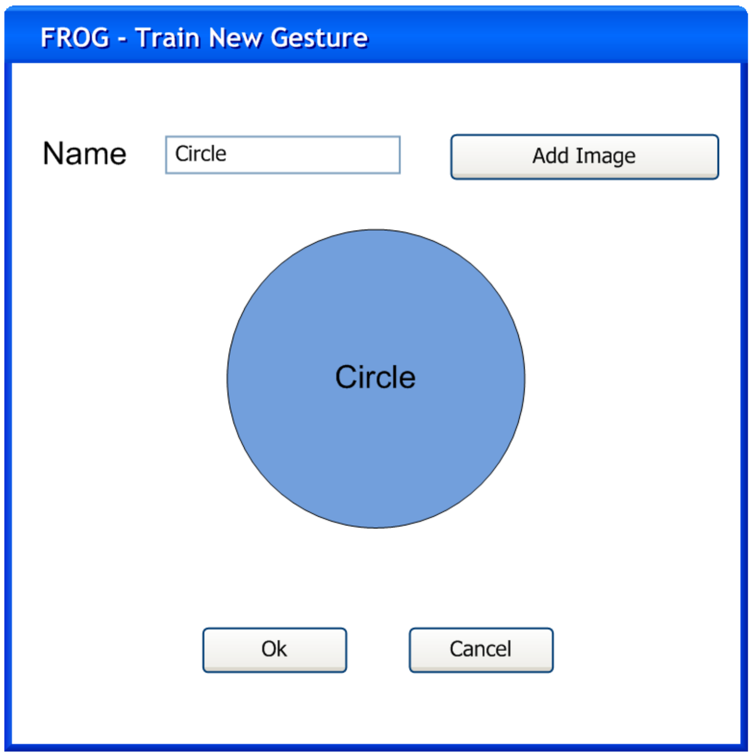


```
>Welcome  !!
>Training session started
>Device Connect: SunSPOT 3B7D
>Waiting for Input...
```

This window is an external terminal. It will display the output given to the user by the mode currently running.

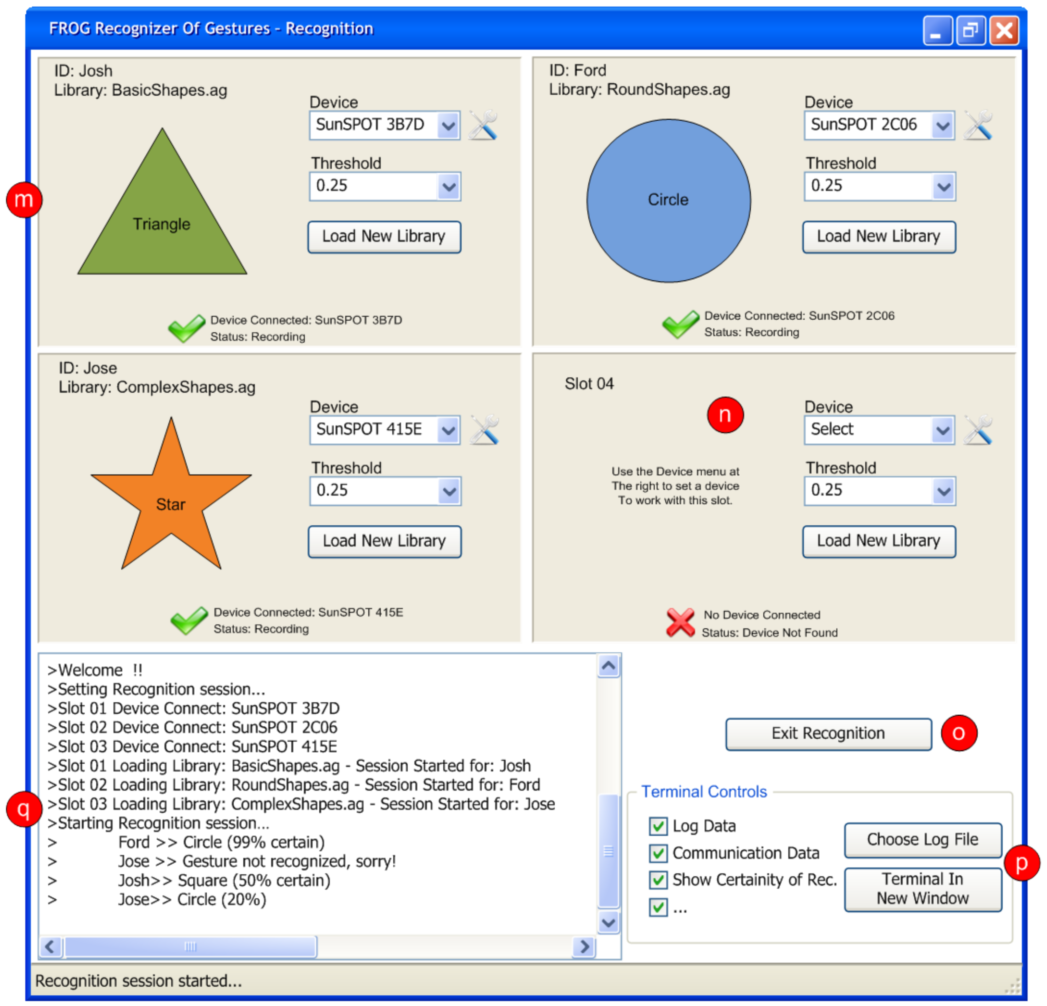
- **k) Terminal**: Refer to Screen 2.0, section g.
- **l) Terminal Controls**: Refer to Screen 2.0, section h.

Screen 2.2: Train New Gesture



This window will allow the user to name the new gesture to be created and will also give the user the ability to attach a symbolic image to the gesture for easier identification of the gesture's semantic meaning in other modes.

Screen 3.0: Recognition Mode



ID: Josh
Library: BasicShapes.ag
Device: SunSPOT 3B7D
Threshold: 0.25
Device Connected: SunSPOT 3B7D
Status: Recording

ID: Ford
Library: RoundShapes.ag
Device: SunSPOT 2C06
Threshold: 0.25
Device Connected: SunSPOT 2C06
Status: Recording

ID: Jose
Library: ComplexShapes.ag
Device: SunSPOT 415E
Threshold: 0.25
Device Connected: SunSPOT 415E
Status: Recording

Slot 04
Use the Device menu at The right to set a device To work with this slot.
Device: Select
Threshold: 0.25
No Device Connected
Status: Device Not Found

```
>Welcome  !!
>Setting Recognition session...
>Slot 01 Device Connect: SunSPOT 3B7D
>Slot 02 Device Connect: SunSPOT 2C06
>Slot 03 Device Connect: SunSPOT 415E
>Slot 01 Loading Library: BasicShapes.ag - Session Started for: Josh
>Slot 02 Loading Library: RoundShapes.ag - Session Started for: Ford
>Slot 03 Loading Library: ComplexShapes.ag - Session Started for: Jose
>Starting Recognition session...
>          Ford >> Circle (99% certain)
>          Jose >> Gesture not recognized, sorry!
>          Josh>> Square (50% certain)
>          Jose>> Circle (20%)
```

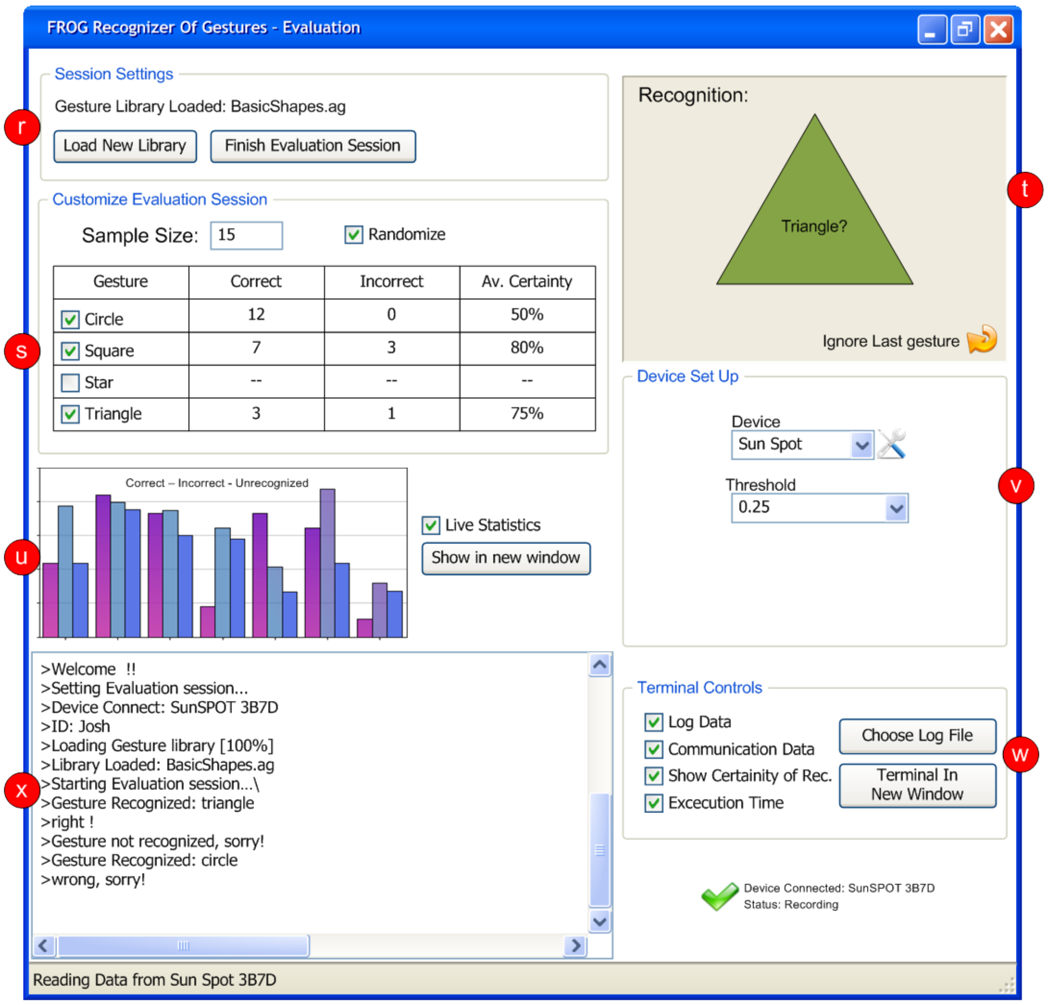
☑ Log Data
☑ Communication Data
☑ Show Certainity of Rec.
☑ ...

Recognition session started...

This is the window for Recognition mode.

- **m) User Panel**: This panel will handle one user. There will be four of these slots in Screen 3.0. Each one will allow the following actions:
  - **i. Connection Panel**: Refer to Screen 2.0, section j.
  - **ii. Load New Library**: Will allow each user to select his or her own library (gesture session file) to be used for recognition. Upon clicking this button, the user will be prompted with a standard open file dialog.
- **n) User Panel (Unconnected)**: This User Panel shows the appearance of such a panel when no user is connected.
- **o) Exit Recognition**: Will terminate the current recognition session, close Screen 3.0, and return to Screen 1.0.
- **p) Terminal Controls**: Refer to Screen 2.0, section h.
- **q) Terminal**: Refer to Screen 2.0, section g.

Screen 4.0: Evaluation Mode



Sample Size: 15 ✓ Randomize

Gesture	Correct	Incorrect	Av. Certainty
✓ Circle	12	0	50%
✓ Square	7	3	80%
Star	--	--	--
✓ Triangle	3	1	75%

Device: Sun Spot
Threshold: 0.25

✓ Log Data
✓ Communication Data
✓ Show Certainity of Rec.
✓ Excecution Time

```
>Welcome  !!
>Setting Evaluation session...
>Device Connect: SunSPOT 3B7D
>ID: Josh
>Loading Gesture library [100%]
>Library Loaded: BasicShapes.ag
>Starting Evaluation session...\
>Gesture Recognized: triangle
>right !
>Gesture not recognized, sorry!
>Gesture Recognized: circle
>wrong, sorry!
```

Device Connected: SunSPOT 3B7D
Status: Recording

Reading Data from Sun Spot 3B7D

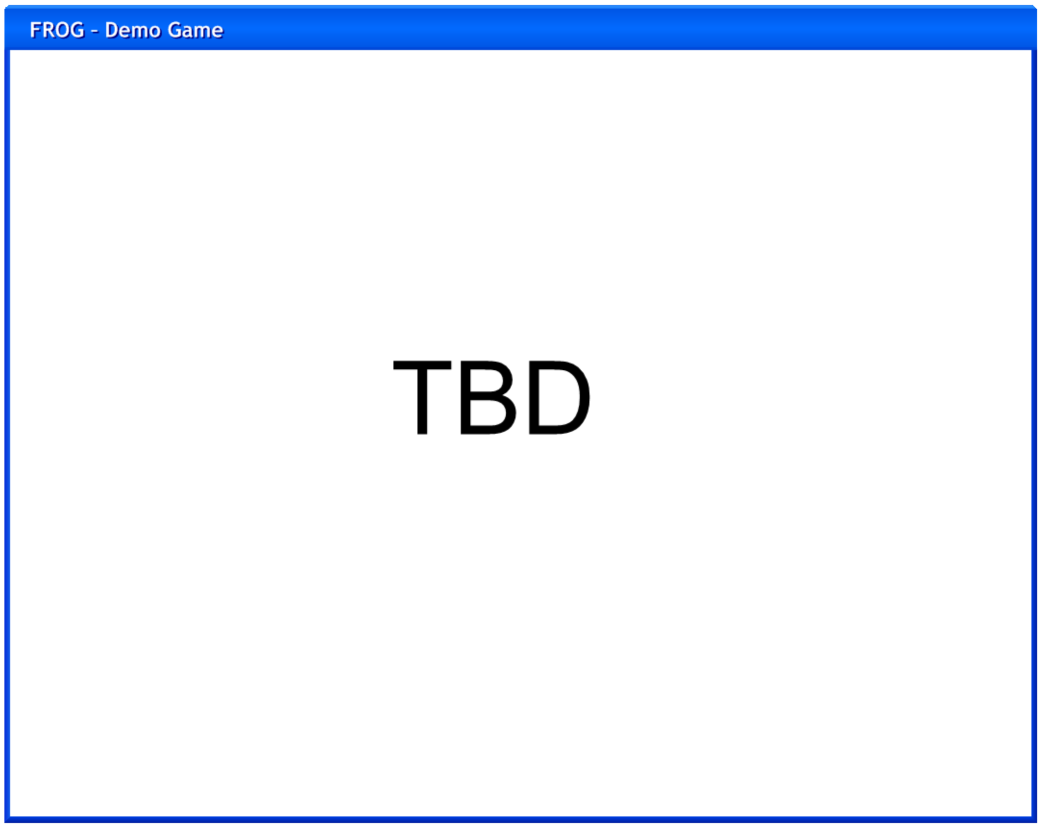
This is the window for Evaluation mode.

r) **Session Settings**: This panel will allow the user to load a library or finish the session.
   i. **Load New Library**: Refer to Screen 3.0, section m.
   ii. **Finish Evaluation Session**: Will terminate the current evaluation session, close Screen 3.0, and return to Screen 1.0.

s) **Customize Evaluation Session**: This panel will allow the user to modify the parameters of the evaluation session.
   i. **Session Table:** Will allow the user to select which gestures are to be part of the evaluation as well as display real-time statistics.
   ii. **Sample Size**: Will allow the user to input the sample size for evaluation.
   iii. **Randomize**: If selected, will allow the user to be prompted for gestures in a random fashion. Otherwise, gestures will appear sequentially.

t) **Gesture Prompt**: This panel will display the name and associated image of a gesture as the gesture is requested by the system.
u) **Statistics Displays**: This panel will display recognition evaluation statistics in graphical and tabular format.
v) **Connection Panel**: Refer to Screen 2.0, section j.
w) **Terminal Controls**: Refer to Screen 2.0, section h.
x) **Terminal**: Refer to Screen 2.0, section g.

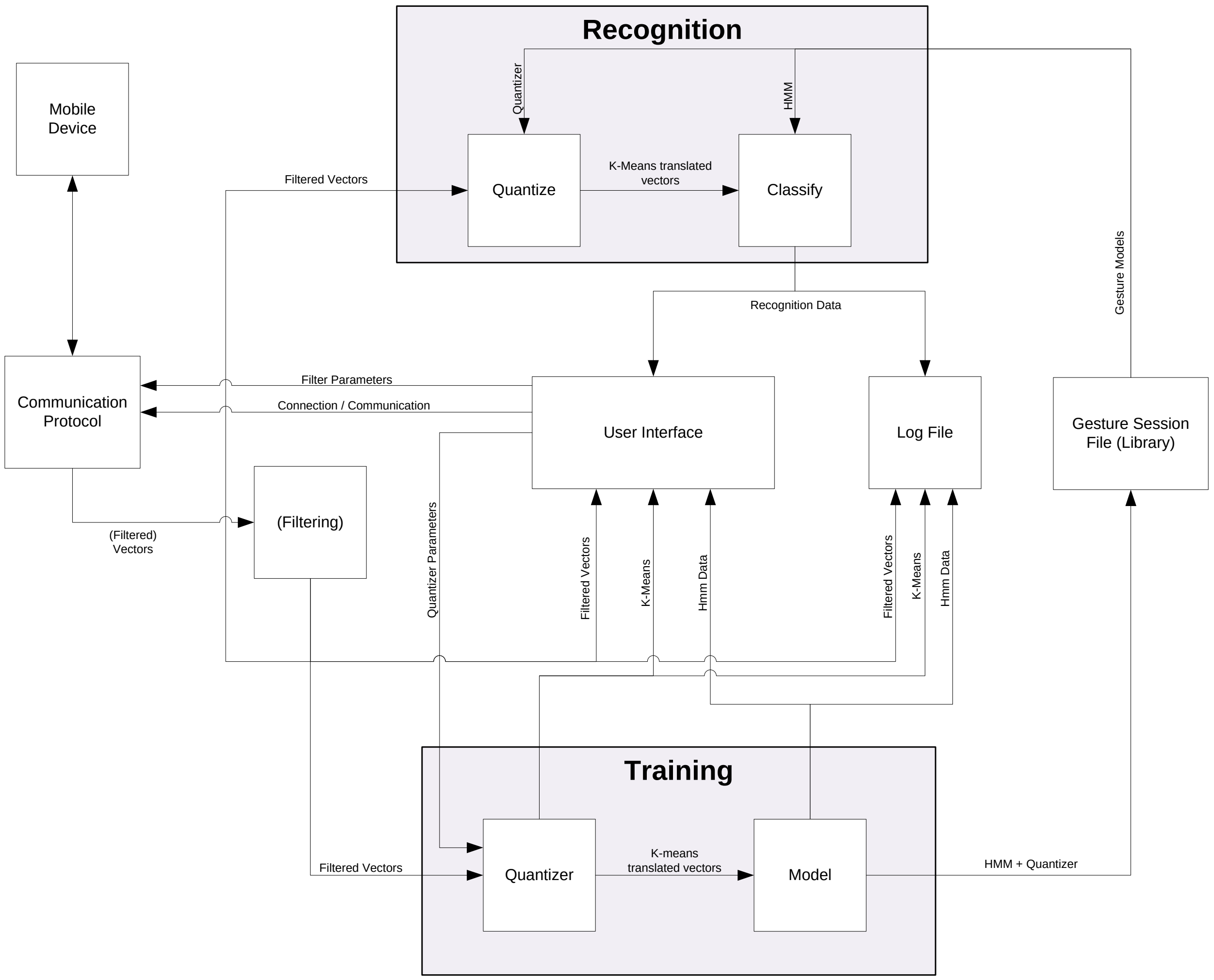
Screen 5.0: Demo



This is the window for Demo mode. To utilize Demo mode, an appropriate library must be loaded. That is, Demo mode will require a certain set of predefined gestures to be trained within the library chosen.

Appendix C: Architecture

The following represents the FROG architecture. It distinguishes system structure as well as data flow.



Appendix D: Algorithms

This section describes algorithms necessary for gesture recognition in the FROG project.

Forward Algorithm:
The forward algorithm is designed for computing the probability of observing a certain sequence of observations being emitted from an HMM. In other words, it deals with the problem of recognition. Its viability comes from its marked increase in efficiency as compared to that of a brute force approach. A brute force approach would involve traversing every possible state path and combining each path's probability of producing the sequence. The forward algorithm, on the other hand, utilizes the Markov property of an HMM to drastically improve this process. The algorithm relies on calculating partial probabilities at each time step in the observation sequence (that is, consider each observation in the sequence as arriving at a discrete time t corresponding to the place it occurs in the sequence). This document defines the partial probabilities recursively (using the notation defined in Section 2):

$$\lambda = (S,\ O,\ a,\ b,\ \pi)$$
$$\alpha_{t+1}(j) = b[j][o_{t+1}] * \Sigma_{i=1}^{n} \alpha_t(i) * a[i][j]$$
$$\alpha_1(i) = \pi[i] * b[i][o_1]$$

The variable $\mathbf{o_t}$ represents the observation occurring at time t. And so the algorithm is based on initializing the $\boldsymbol{\alpha_1}$**(i)** according to the initial probability multiplied by the emission probability of the first symbol in the observation sequence. Then $\boldsymbol{\alpha_{t+1}}$**(j)** is the emission probability for the next observation multiplied by the sum of all the previous partial probabilities multiplied by the probability of transitioning from each state to the new state j. The recursive computation of this value can yield the probability of observing any sequence by simply summing up over the $\boldsymbol{\alpha_T}$**(i)** where T is the total number of observations in the sequence (that is, sum over the last set of partial probabilities computed).

Backward Algorithm
The backward algorithm is based on exactly the same premises as the forward algorithm. It is also utilized for calculating the probability of an observed sequence from an HMM. This algorithm, however, calculates probabilities, as suggested by its name, starting from the last observation and working backward. The algorithm is defined recursively as:

$$\beta_T(i) = 1$$
$$\beta_{t-1}(j) = \Sigma_{i=1}^{n} \beta_t(i) * a[j][i] * b[i][o_t]$$

As it is exactly derivative of the forward algorithm above, only now using the β function, further explanation will not be given here.

Baum-Welch Algorithm
The Baum-Welch algorithm deals instead with the training problem. The Baum-Welch is an expectation maximization algorithm that utilizes the forward and backward algorithms. The algorithm is designed to optimize the parameters of an HMM so as to best model given training sequences. Thus it deals with maximizing the conditional probability of an observation sequence occurring given an HMM (to be optimized). The algorithm is only capable of carrying out local optimization, however, so there is no guarantee of the truly optimal HMM, which would require knowledge of a global optimum. Here let **a** again be the transition matrix, but use notation **$e_i(o_k)$** to represent the emission probability for observation k from state i. Then formally stated the Baum-Welch algorithm is:

```
Algorithm: Baum Welch

Input:
   A set of observed sequences, $O^1, O^2, \ldots$
Initialization:
   Select arbitrary model parameters, $\lambda' = a_{ij}, e_i()$.
         score = $\sum_d P(O^d|\lambda')$.
   Repeat
   {
      $\lambda = \lambda'$, $S = S'$
      For each sequence, $O^d$,
      {
         /* Calculate ‘‘probable paths’’ $Q^d = q_1^d, q_2^d, \ldots$ */
         Calculate $\alpha(t,i)$ for $O^d$ using the Forward algorithm.
         Calculate $\beta(t,i)$ for $O^d$ using the Backward algorithm.
         Calculate the contribution of $O^d$ to $A$ using (1).
         Calculate the contribution of $O^d$ to $E$ using (2).
      }
      $a_{ij} = \frac{A_{ij}}{\sum_l A_{il}}$
      $e_i(\sigma) = \frac{E_i(\sigma)}{\sum_\tau E_i(\tau)}$
      score = $\sum_d P(O^d|a_{ij}, e_i())$.
   }
   Until (the change in score is less than some predefined threshold.)
```

Where:

$$A_{ij} = \sum_d \frac{1}{P(O^d)} \sum_t \alpha(t,i)\, a_{ij}\, e_i(O^d_{t+1})\, \beta(t+1,i)$$

$$E_i(\sigma) = \sum_d \frac{1}{P(O^d)} \sum_{\{t|O^d_t=\sigma\}} \alpha(t,i)\, \beta(t,i).$$

$\mathbf{O}^d$ is the sequence of d observations of the complete observation sequence. α and β refer just as they did above to the respective forward/backward algorithm partial probabilities. And so with each iteration the HMM model parameters are modified towards a local optimum, training the model to the sequences provided.