Design Document

FROG Recognizer of Gestures

Team Better Recognize
Version 2.0
February 2, 2010

Copyright ©2009-2010 Computer Science Department, Texas Christian University

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	11/24/09	Initial Draft
Version 2.0	2/2/10	Iteration 2 Update

Table of Contents

Revision Sign-off i
Revision History ii
1. Introduction 1
1.1 Purpose 1
1.2 Identification 1
1.3 Scope 1
1.4 Overview 1
2. System Architecture 2
2.1 Hardware and Communication Architecture 2
2.2 Software Architecture 4
3. Data Design 6
3.1 Logging 6
3.2 Gesture Libraries 6
4. Detailed Design 7
4.1 Hardware Detailed Design 7
4.2 Software Detailed Design 8
4.2.1 Use-case model 10
4.2.2 Sequence diagrams 11
5. External Interface Design 19
6. Human Machine Interface 20

1. Introduction

1.1 Purpose

This document gives the overall design description of the FROG Project. It details how the user will interact with the system. It also describes how the system passes information from one section to another, while outlining the specifics of the coding sections of FROG. This includes GUI prototypes, walkthroughs, system diagrams, and class diagrams along with their descriptions.

1.2 Identification

The goal of the FROG Project is the development of a device and platform-independent gesture training and recognition system. The FROG recognition framework is created with the intent to be used in motion-based recognition research through utilization of 3D accelerometer-enabled mobile devices. The FROG project shall consist of a Hidden Markov Model-based training and recognition framework supported by an intuitive and research-oriented GUI.

1.3 Scope

In its initial release, FROG will be packaged with a plug-in for Sun SPOTs. Additional plug-ins for compatible mobile devices may be written later based on a plug-in framework. In addition to the necessary training and recognition modes, FROG will contain an evaluation mode as well as a demo program for testing and demonstrating the capabilities of the project.

1.4 Overview

Section 2 - System Architecture: Description of hardware and communication architecture as well as software architecture. Hardware and communication description is concerned with Sun SPOT device and communication architecture. In software architecture a system architecture diagram and a package diagram are presented.

Section 3 - Data Design: Description of the details concerned with structure and storage of log and gesture session (library) files.

Section 4 - Detailed Design: Description of the detailed hardware and software design of FROG. In software detailed design an in-depth class diagram (including hierarchy and association) and a layout of user interaction with the system through the use of sequence diagrams are presented.

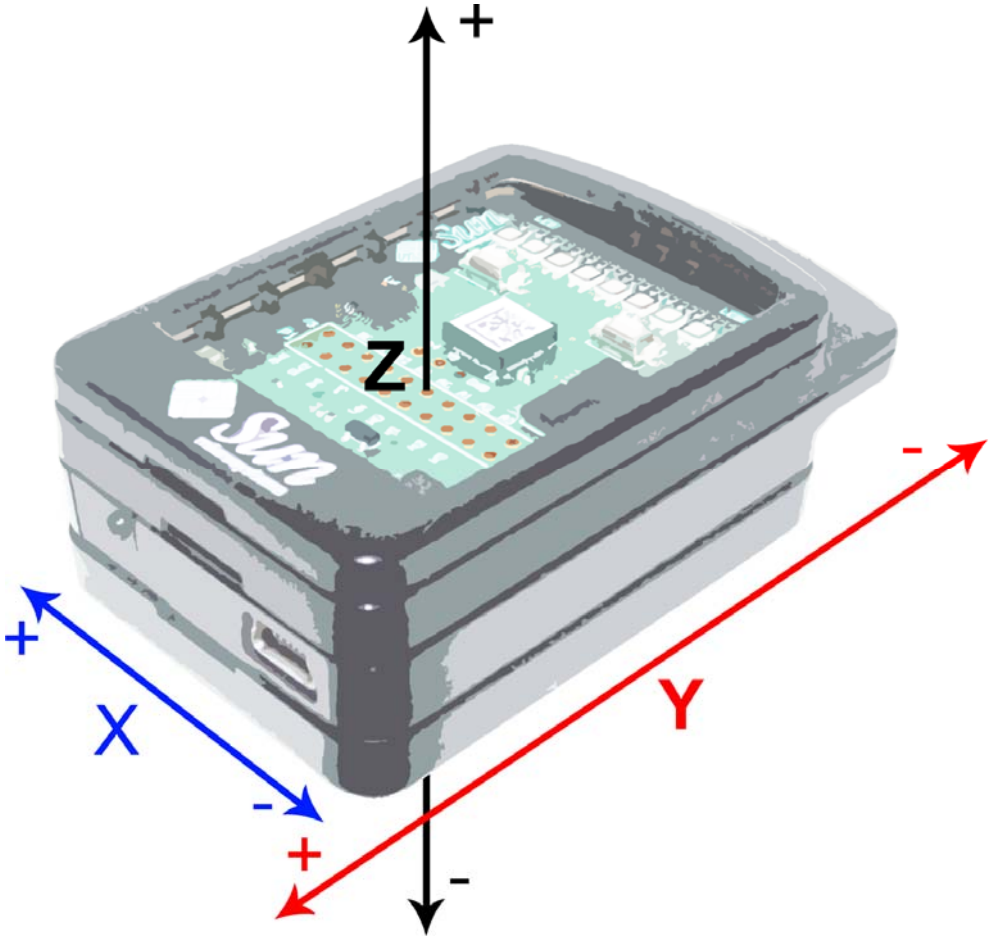
Section 5 - External Interface Design: Description of the structure of the plug-in framework constructed within FROG including details of what is needed in developing a plug-in.

Section 6 - Human Machine Interface (HMI): Layout of the intended GUI.

2. System Architecture

2.1 Hardware and Communication Architecture

The Sun Small Programmable Object Technology or Sun SPOT device will be the only device initially supported by FROG. Sun SPOTs feature accelerometers and a low-power, IEEE 802.15.4 radio. Acceleration vectors can be gathered by these accelerometers and transmitted via the radio to FROG for gesture training/recognition. The following diagram details the axes of each accelerometer.



Sun SPOTs are programmable in Java, running a limited Java Virtual Machine called "Squawk." All onboard Sun SPOT facilities, including radio and accelerometers, are accessible through the Squawk and Sun SPOT APIs. The Sun SPOT plug-in for FROG will require additional code designed to be executed on the Sun SPOT itself as well as a protocol to allow communication between device and plug-in.

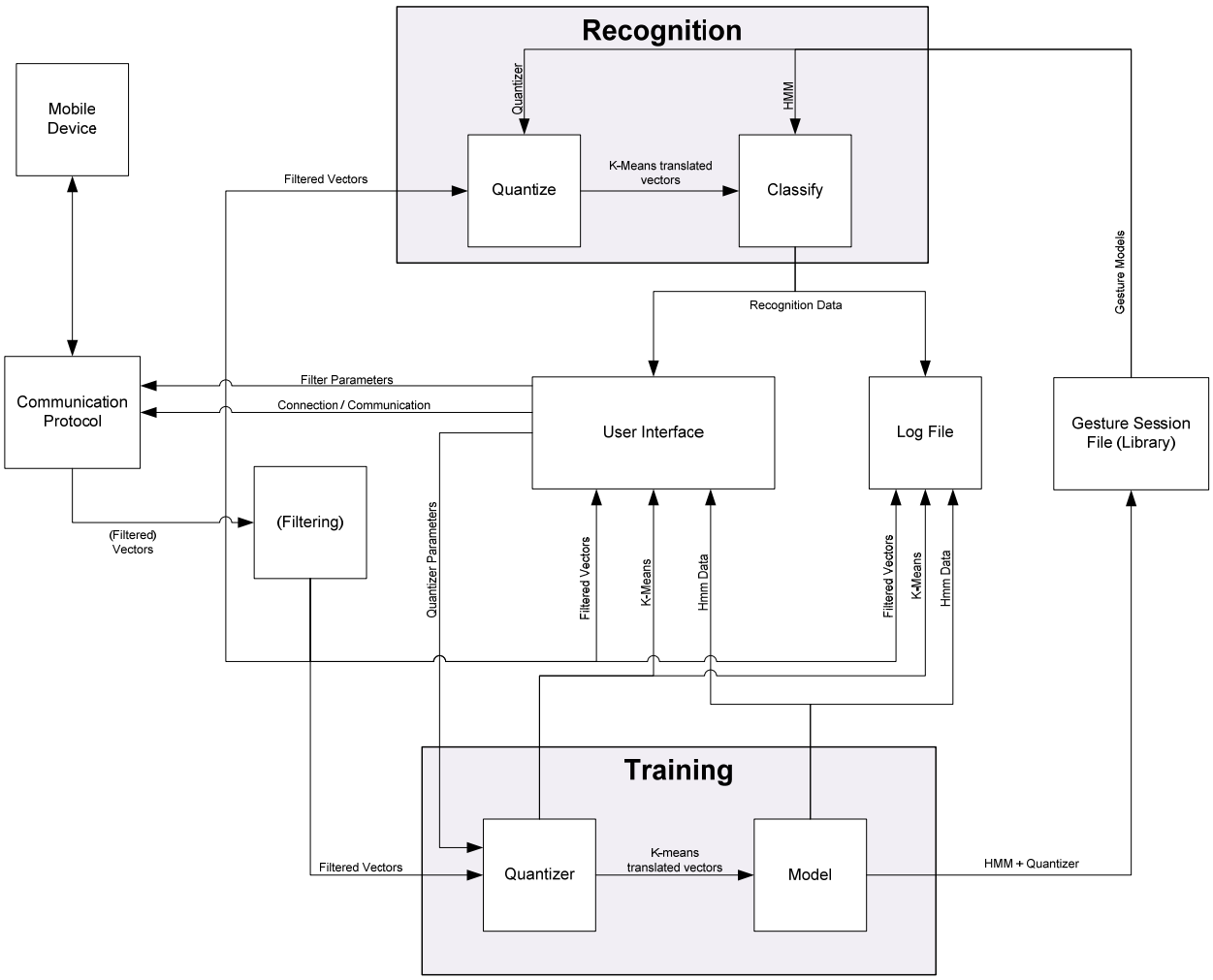
Five types of messages will be required in this protocol:

- ACK – from host to SPOT, when the END_OF_DATA along with all gestures before it have been received
- RESEND – from host to SPOT, if the host receives a datagram out of order, it can send this message to instruct the SPOT to start sending from the last one faithfully received
- ADD_FILTER – from host to SPOT, instructs the SPOT to add an additional active filter of the host's choosing
- REMOVE_FILTERS – from host to SPOT, instructs the SPOT to remove all its active filters
- ALIVE – from host to SPOT, a packet that makes sure the host and SPOT are still connected
- ACCELERATION – from SPOT to host, an acceleration message containing the order it should be received in as well as three 32-bit integers that represent the acceleration multiplied by 100,000,000
- END_OF_DATA – from SPOT to host, a final message that lets the host delineate between gestures

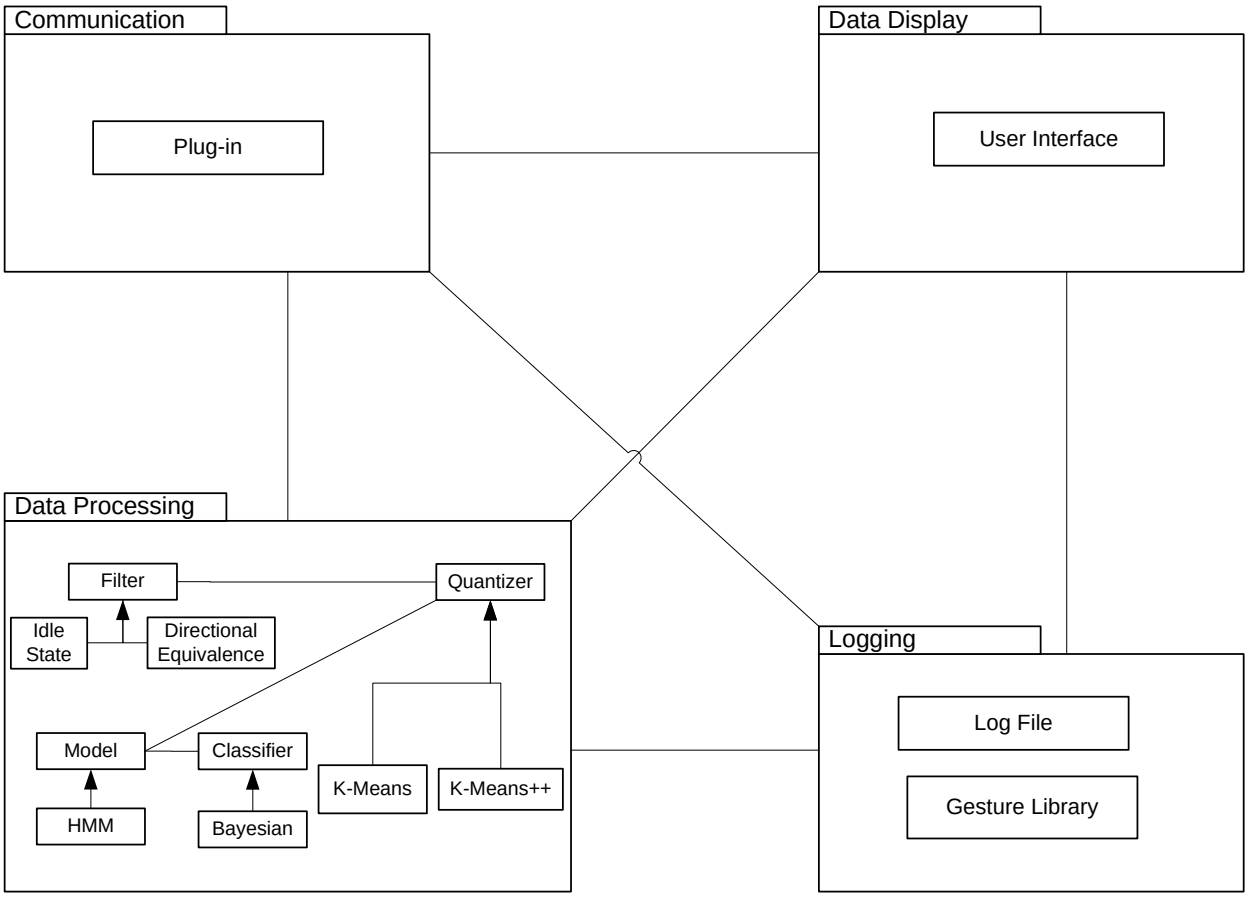
Keeping the protocol simple and communicating only what is necessary helps to alleviate the potential performances issues related to the limited facilities of the Sun SPOT. This shall aid in maintaining the ability to sample acceleration quickly enough to accurately map out a user's performed gesture.

2.2 Software Architecture

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



This is a diagram of the packages included in the FROG architecture.



3. Data Design

3.1 Logging

Logging is crucial as it informs the user about the status and performance of the system. It is also an extremely useful debugging tool for a developer.

The FROG Project logging system will be encapsulated in a simple class, Log. This class is never instantiated but rather performs all the necessary logging through "out", a method designed with standardization and ease of use in mind. This method will write messages from any object that calls it to a specified log file, the chosen JTextArea console window, and the system's standard out.

The format of the log file is:

```
Sat Nov 21 17:08:29 CST 2009 class frog.Log:  Start of Log
Sat Nov 21 17:08:30 CST 2009 frog.HMM:  Instantiated
Sat Nov 21 17:08:31 CST 2009 class frog.Log:  End of Log
```

Each line begins with a date and time the message was created, followed by the contents of the message – that is, a description of the relevant event.

Much of the Log's behavior can be customized. A user may choose the location and name of the file that the system is using for logging. Since FROG has a modular GUI, a newly written GUI can easily hook into Log to make use of and view system updates by specifying a JTextArea for the FROG system. The Log class also contains booleans for toggling output of FROG's subsystems such as the Classifier and Filters.

3.2 Gesture Libraries

Gesture libraries are saved using Java's built-in serialization feature. HMM and Quantizer are both packaged into a Java serialized object file to be opened later. This cuts down on development time for a save/load scheme for gesture libraries.

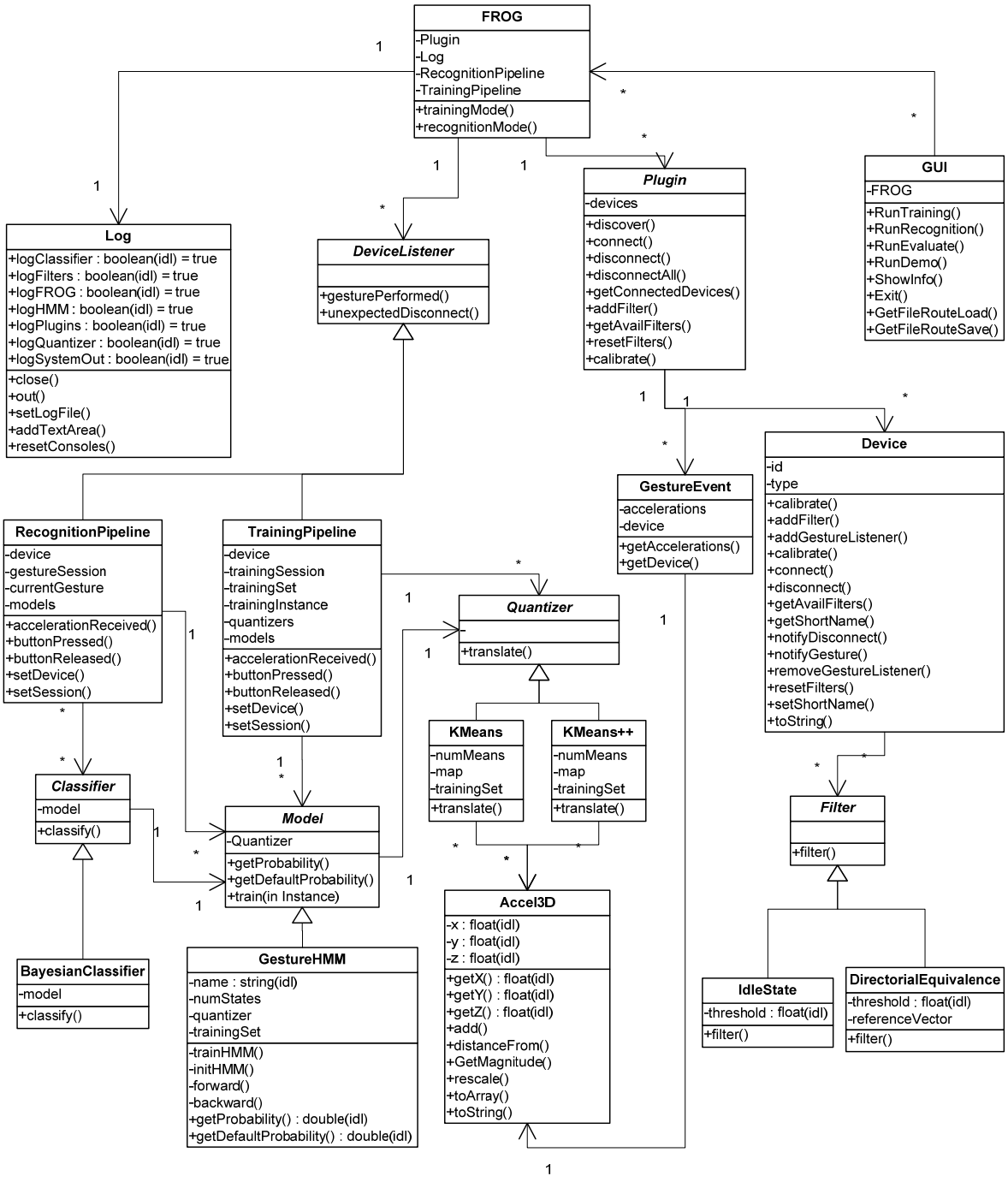
Gesture libraries contain only the HMM and Quantizer, both necessary for recognition of the gestures in the library. However, because training instances are not stored, it is impossible to add additional training instances to a gesture after saving the gesture. The user will have to delete the gesture and retrain it if the performance of a particular gesture is unsatisfactory.

4. Detailed Design

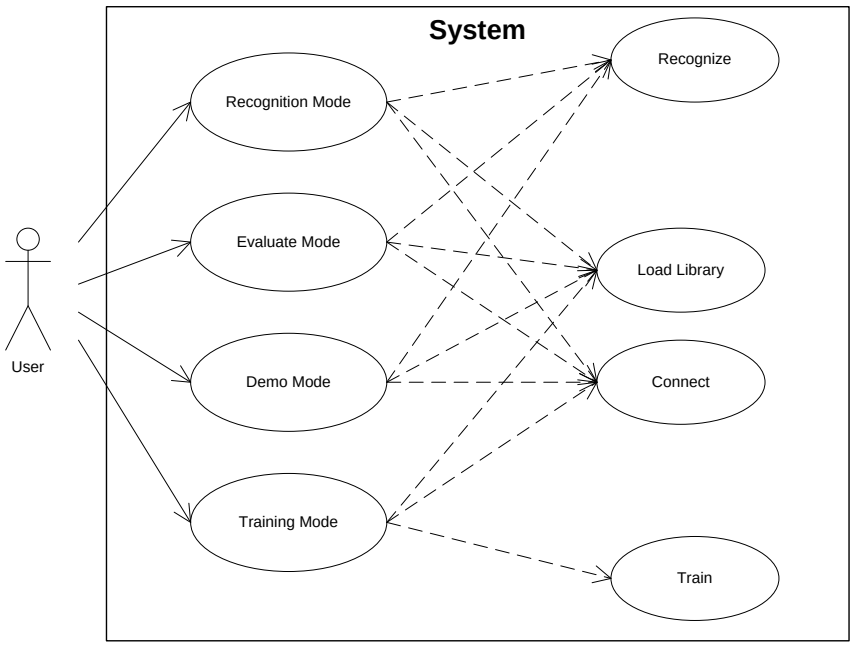
4.1 Hardware Detailed Design

We may be putting something in this section later. For now, it is still one of the great mysteries of the universe.

4.2 Software Detailed Design

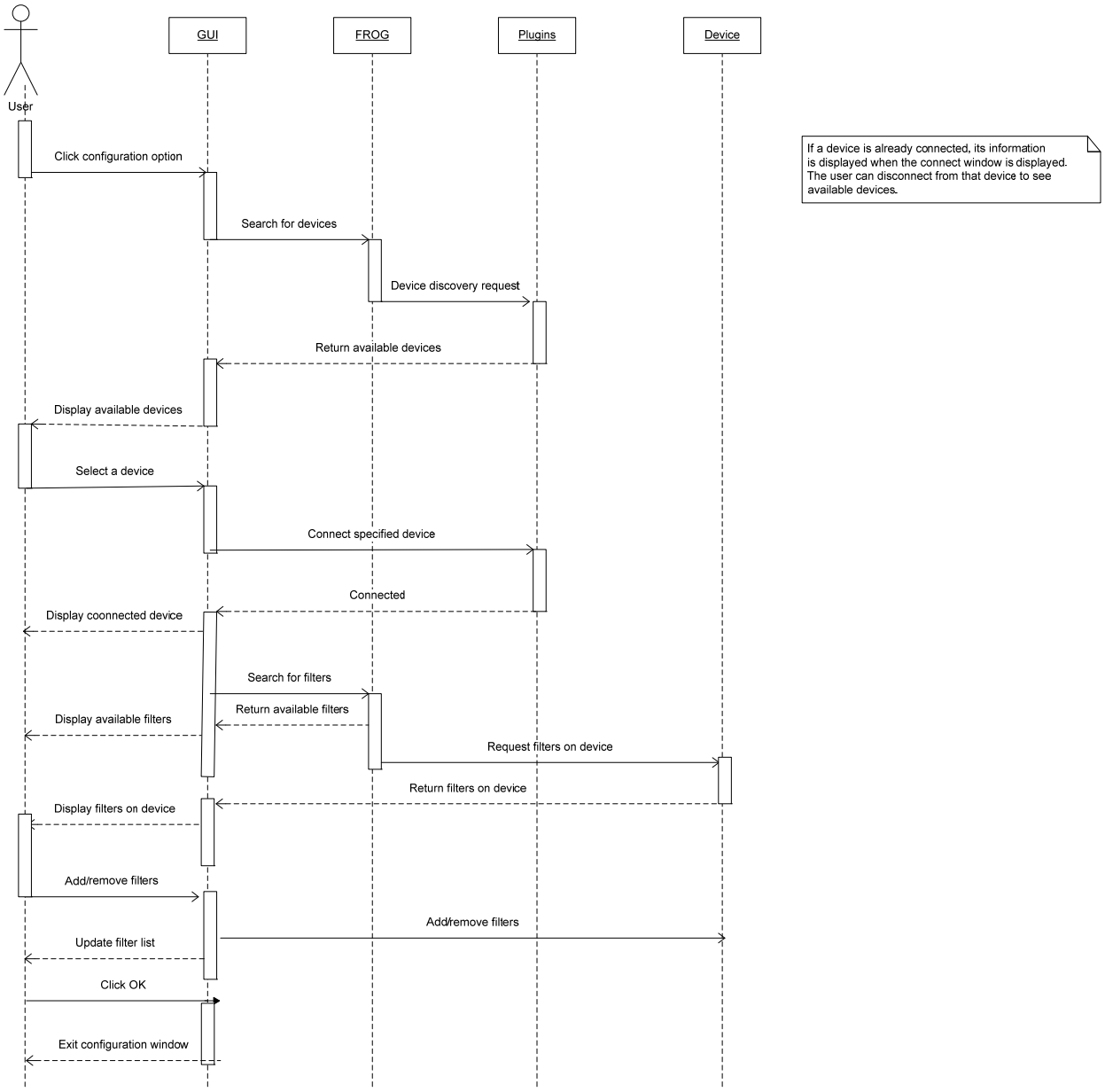

FROG
-Plugin
-Log
-RecognitionPipeline
-TrainingPipeline
+trainingMode()
+recognitionMode()
Log
+logClassifier : boolean(idl) = true
+logFilters : boolean(idl) = true
+logFROG : boolean(idl) = true
+logHMM : boolean(idl) = true
+logPlugins : boolean(idl) = true
+logQuantizer : boolean(idl) = true
+logSystemOut : boolean(idl) = true
+close()
+out()
+setLogFile()
+addTextArea()
+resetConsoles()
DeviceListener
+gesturePerformed()
+unexpectedDisconnect()
Plugin
-devices
+discover()
+connect()
+disconnect()
+disconnectAll()
+getConnectedDevices()
+addFilter()
+getAvailFilters()
+resetFilters()
+calibrate()
GUI
-FROG
+RunTraining()
+RunRecognition()
+RunEvaluate()
+RunDemo()
+ShowInfo()
+Exit()
+GetFileRouteLoad()
+GetFileRouteSave()
RecognitionPipeline
-device
-gestureSession
-currentGesture
-models
+accelerationReceived()
+buttonPressed()
+buttonReleased()
+setDevice()
+setSession()
TrainingPipeline
-device
-trainingSession
-trainingSet
-trainingInstance
-quantizers
-models
+accelerationReceived()
+buttonPressed()
+buttonReleased()
+setDevice()
+setSession()
GestureEvent
-accelerations
-device
+getAccelerations()
+getDevice()
Device
-id
-type
+calibrate()
+addFilter()
+addGestureListener()
+calibrate()
+connect()
+disconnect()
+getAvailFilters()
+getShortName()
+notifyDisconnect()
+notifyGesture()
+removeGestureListener()
+resetFilters()
+setShortName()
+toString()
Quantizer
+translate()
KMeans
-numMeans
-map
-trainingSet
+translate()
KMeans++
-numMeans
-map
-trainingSet
+translate()
Classifier
-model
+classify()
Model
-Quantizer
+getProbability()
+getDefaultProbability()
+train(in Instance)
Filter
+filter()
Accel3D
-x : float(idl)
-y : float(idl)
-z : float(idl)
+getX() : float(idl)
+getY() : float(idl)
+getZ() : float(idl)
+add()
+distanceFrom()
+GetMagnitude()
+rescale()
+toArray()
+toString()
BayesianClassifier
-model
+classify()
GestureHMM
-name : string(idl)
-numStates
-quantizer
-trainingSet
-trainHMM()
-initHMM()
-forward()
-backward()
+getProbability() : double(idl)
+getDefaultProbability() : double(idl)
IdleState
-threshold : float(idl)
+filter()
DirectorialEquivalence
-threshold : float(idl)
-referenceVector
+filter()

4.2.1 Use-case model

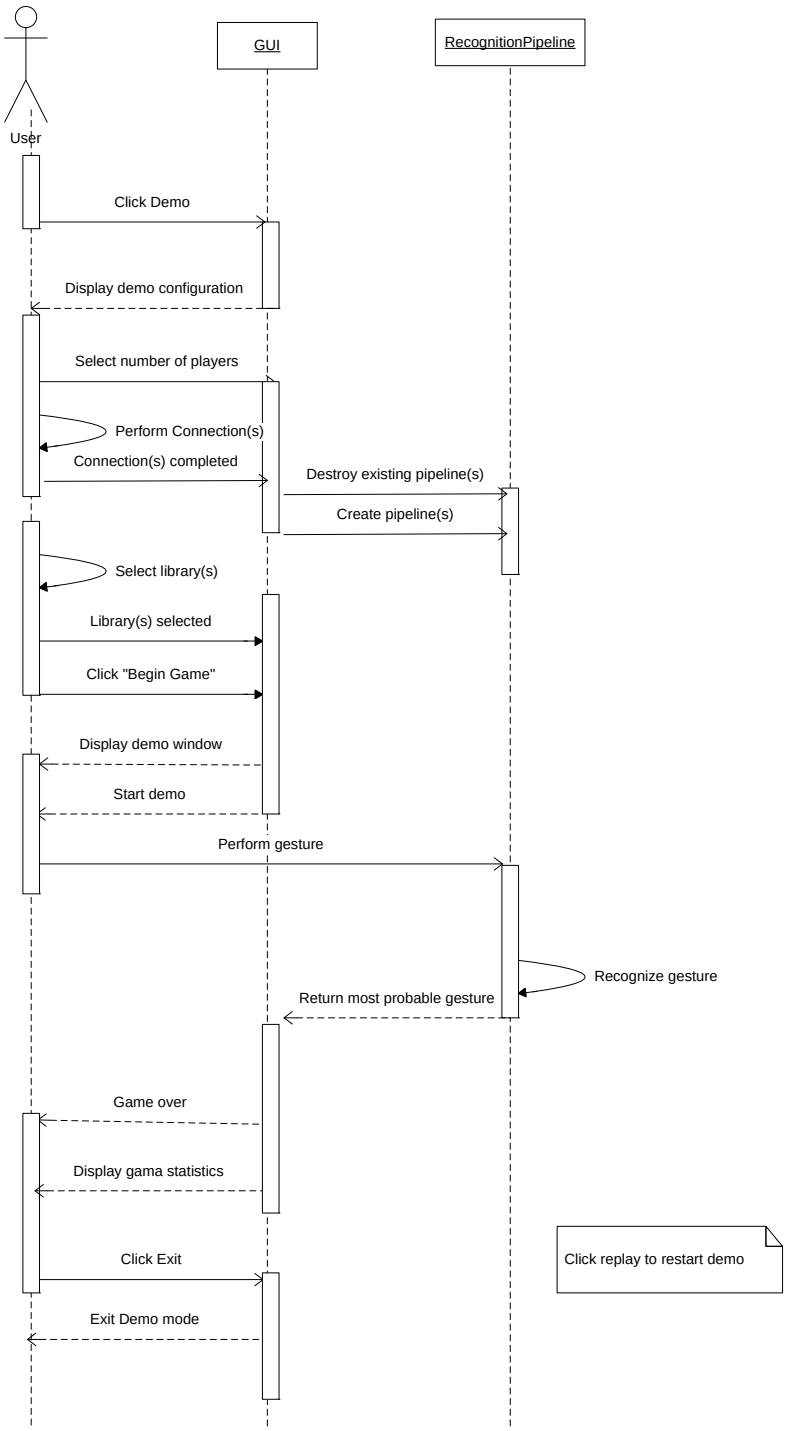


4.2.2 Sequence diagrams

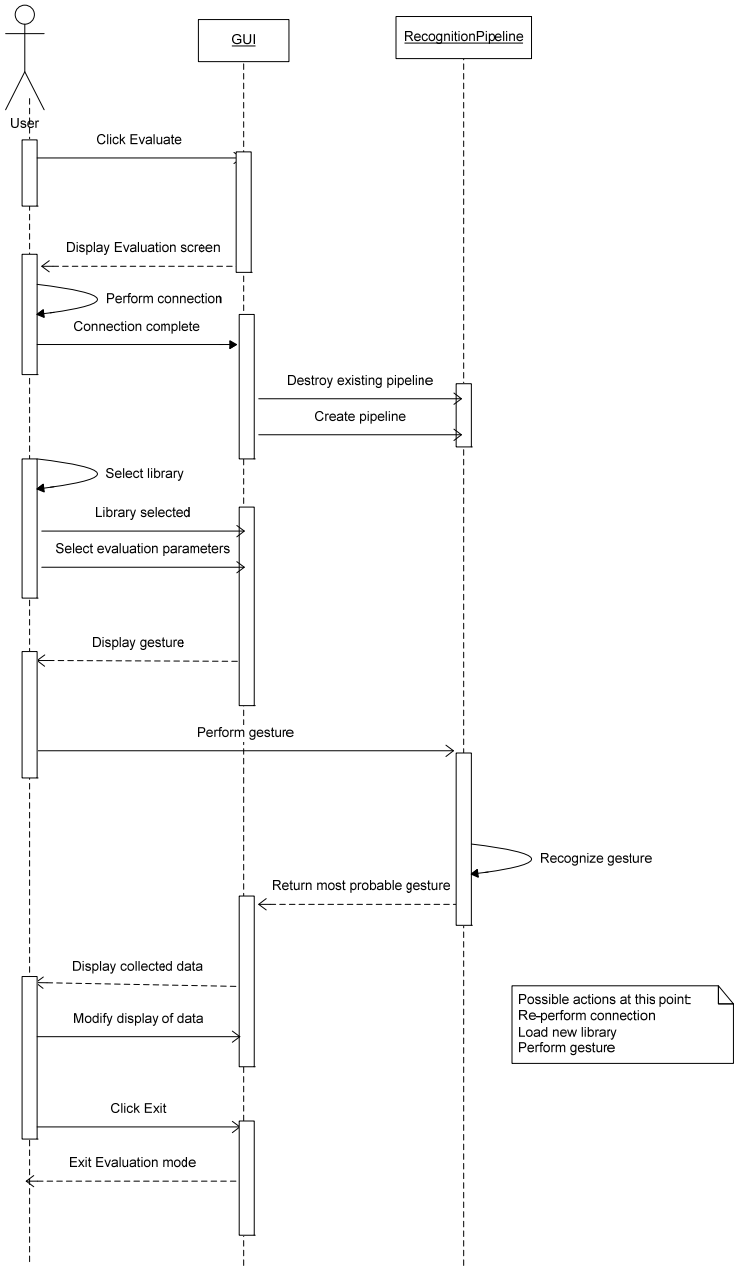
4.2.2.1 Connect



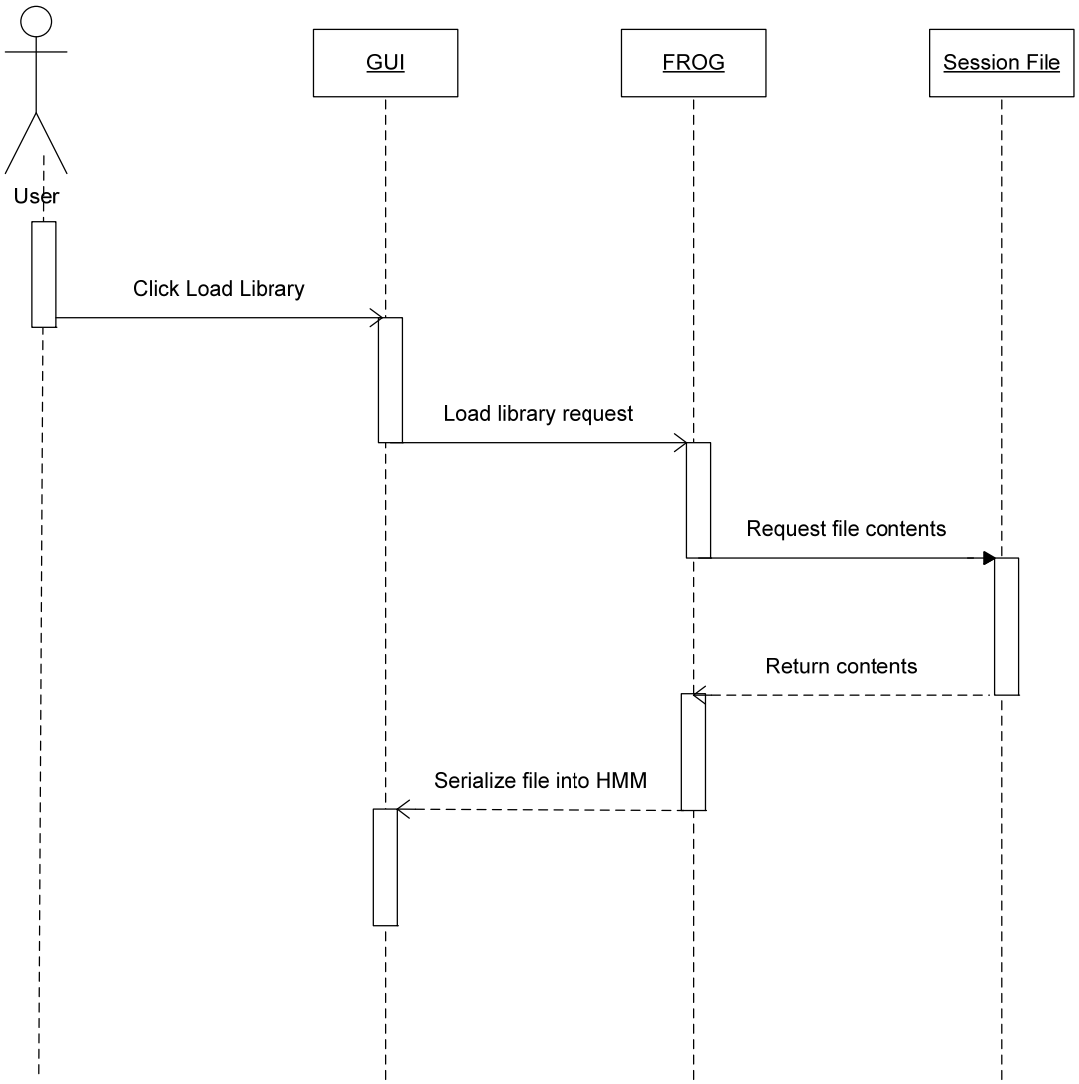
4.2.2.2 Demo Mode



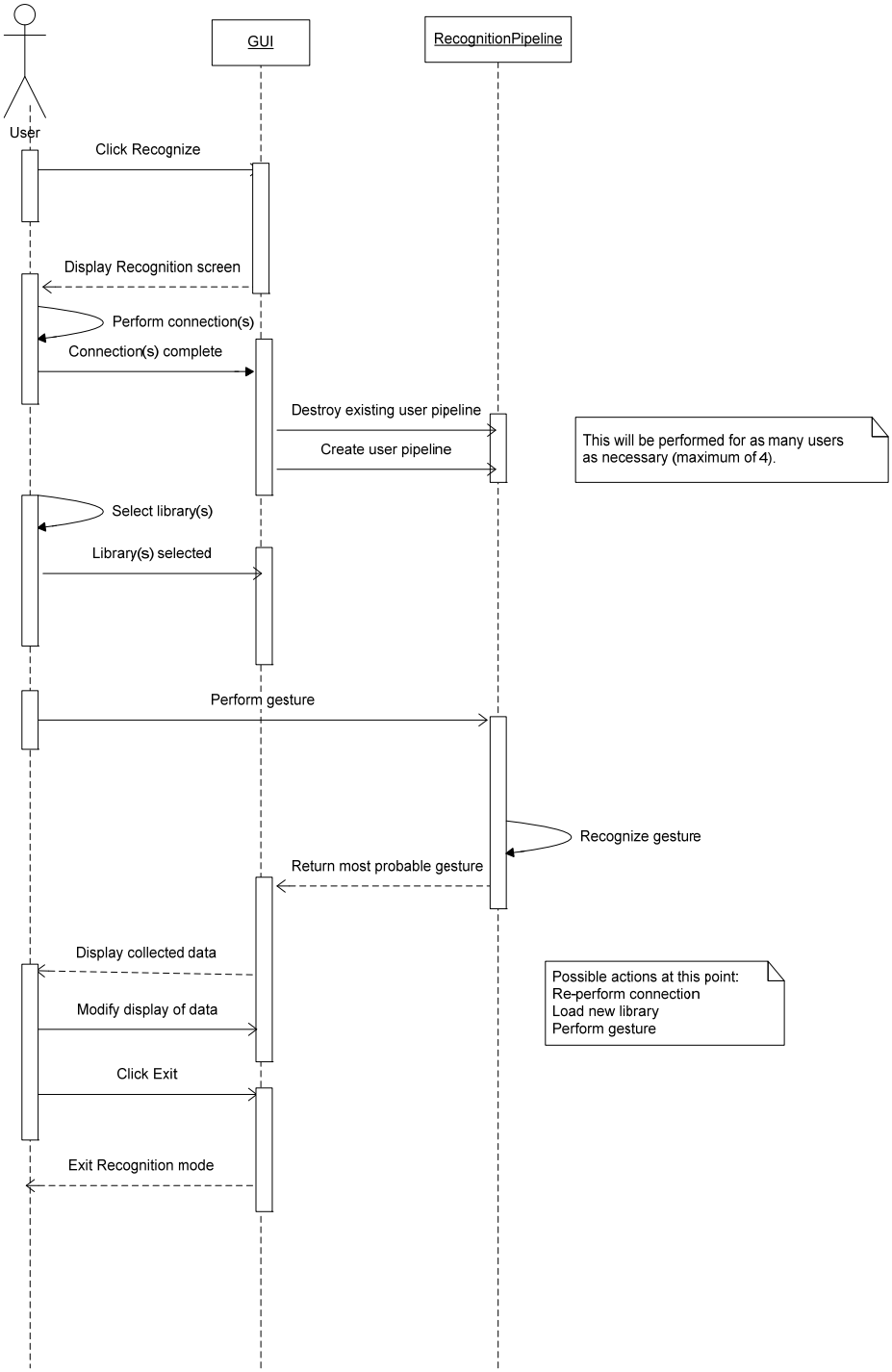
4.2.2.3 Evaluation Mode



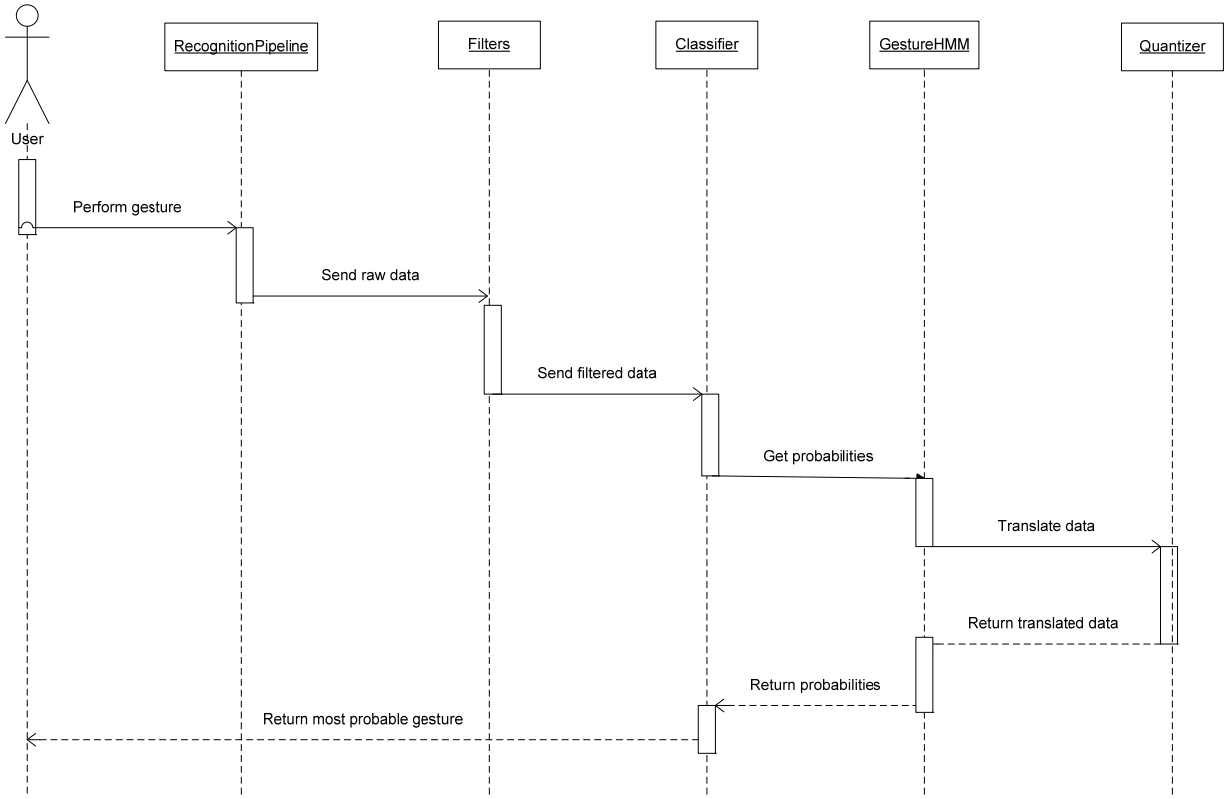
4.2.2.4 Load Library



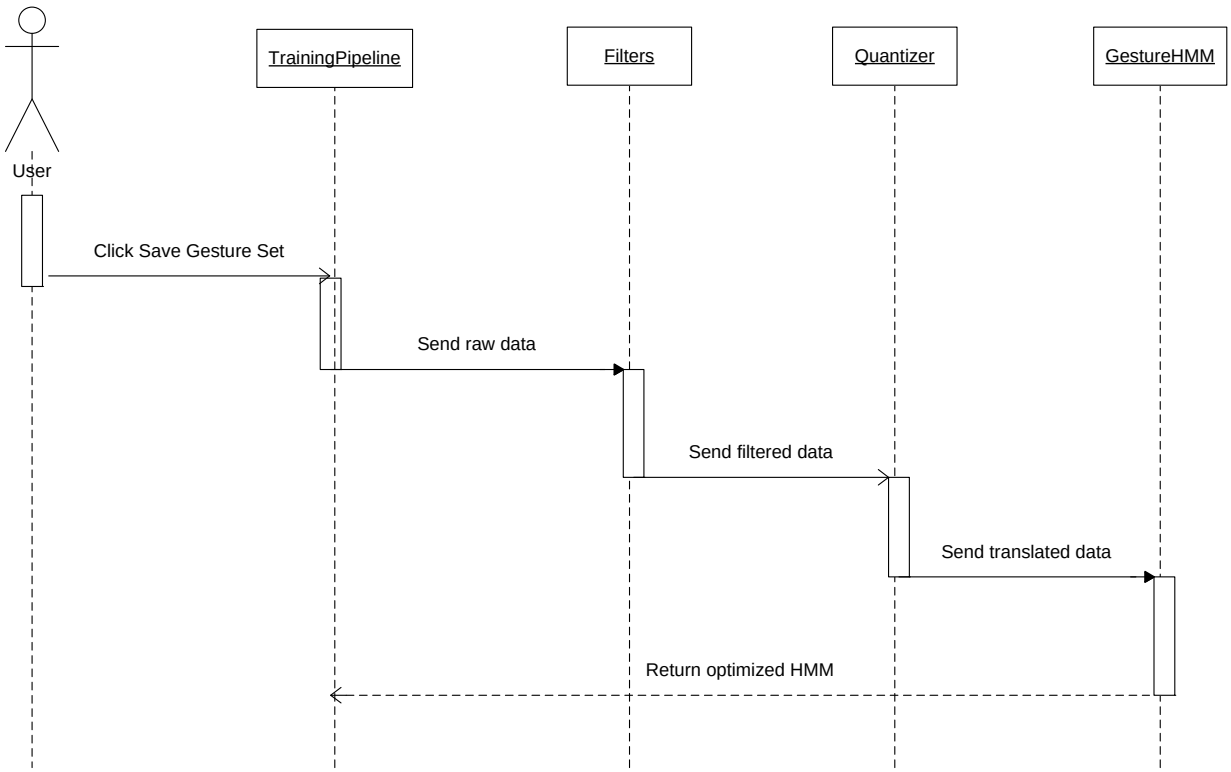
4.2.2.5 Recognition Mode



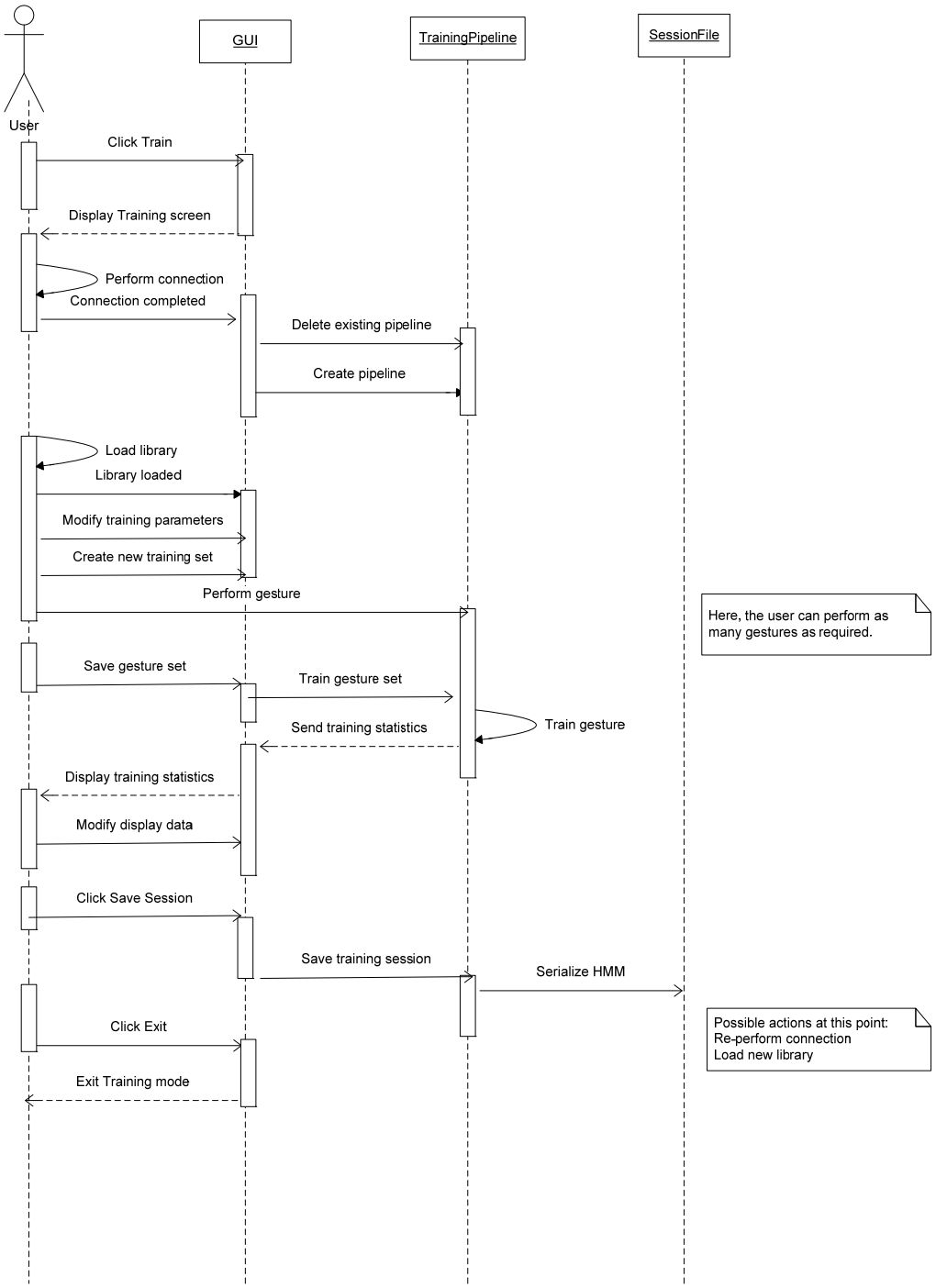
4.2.2.6 Recognize



4.2.2.7 Train


User
TrainingPipeline
Filters
Quantizer
GestureHMM
Click Save Gesture Set
Send raw data
Send filtered data
Send translated data
Return optimized HMM

4.2.2.8 Training Mode



5. External Interface Design

To ensure the versatility of the FROG Recognizer of Gestures system a plug-in framework for mobile devices shall be developed. The *Plugin* interface is to be implemented in the code of any developers wishing to add support for their device with FROG. Plug-ins must handle all communication with a device. Additionally, vectors must be normalized such that acceleration measured by accelerometers is measured in g's (and not m/s^2 or some other unit). Plug-ins will be loaded automatically at start up by searching the FROG "plugin\" directory for valid .class files (classes that implement *Plugin*). Some devices require calibration to perform correctly, and, as such, the *Plugin* interface features a "calibrate()" method which can open a custom-made GUI interface specifically for calibration.

The "Device" class is to be instantiated for each device the plug-in discovers. A device is not necessarily connected until the user has selected the device for use. The Device class is not abstract or meant to be extended, but rather it contains all the functionality needed for any device. Methods called on Device such as "getAccel()" or "connect()" are passed behind the scenes as requests to the Plugin interface. Devices will keep track of the *Plugin* object that created them in order to place the burden involved in plug-in monitoring solely within the Device class and not in the exterior FROG controller class.

In the event that a device unexpectedly disconnects in the middle of operation, "DeviceDisconnectedException" can be thrown by the methods in Device. This will be useful in the GUI design as this error allows the user to become aware of such problems and to save their current session before attempting to reconnect with the device.

6. Human Machine Interface

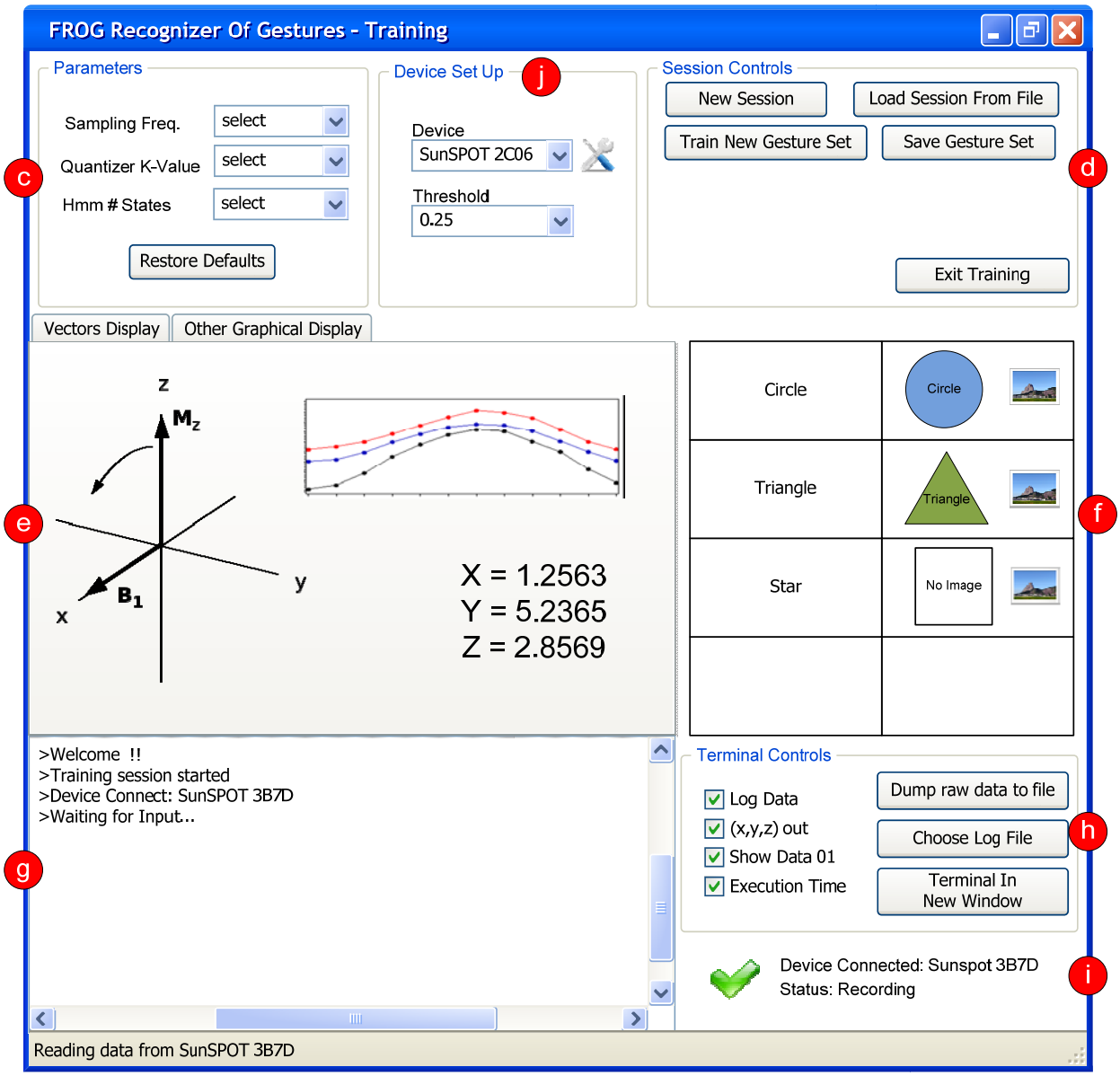
The following is a prototype of the FROG user interface. This prototype gives a screenshot-based walkthrough of the modes and use of the FROG Recognizer of Gestures product.

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



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.

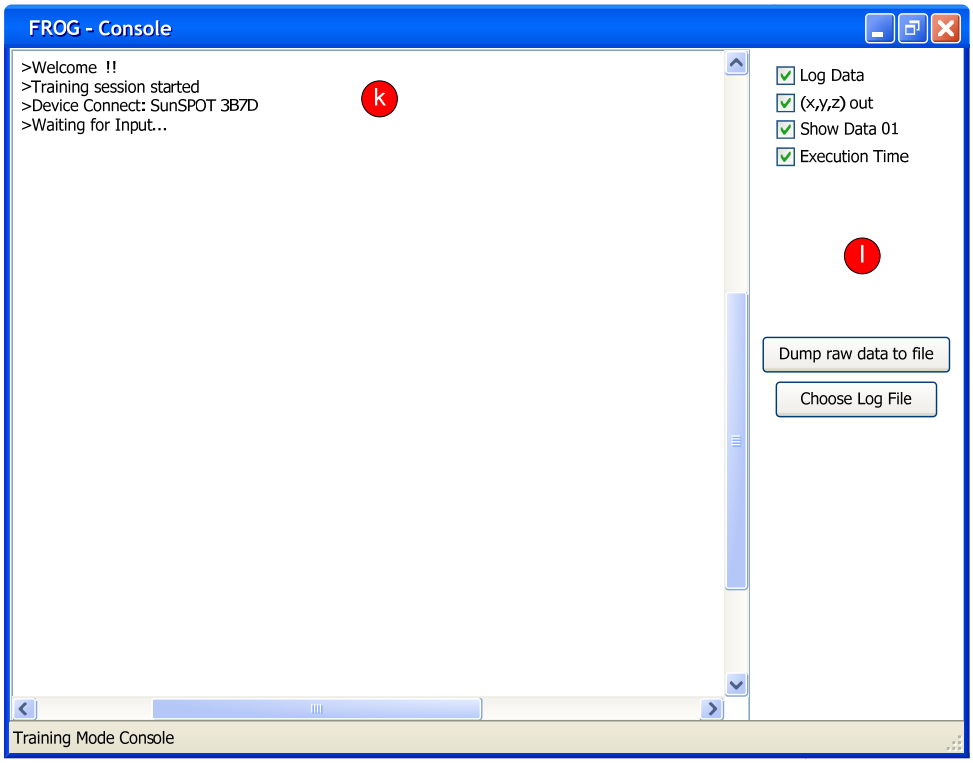
j) **Connection Panel**: This panel will give the user the option of setting up a device to work with the system. This panel will appear (with a different configuration) in other windows and will allow the user the following capabilities:

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

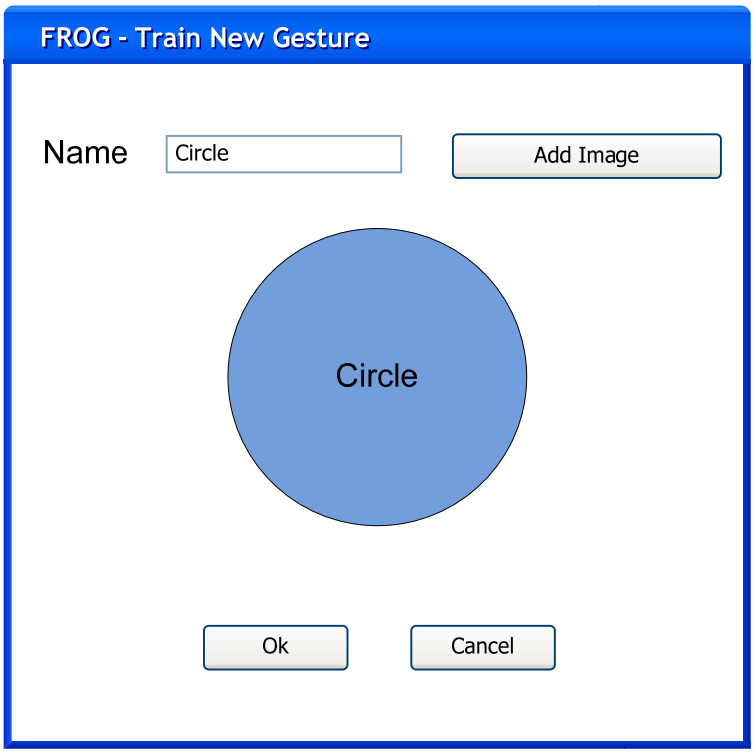


This window is an external terminal. It will display the output given to the user by the mode currently running.

k) Terminal: Refer to Screen 2.0, section g.

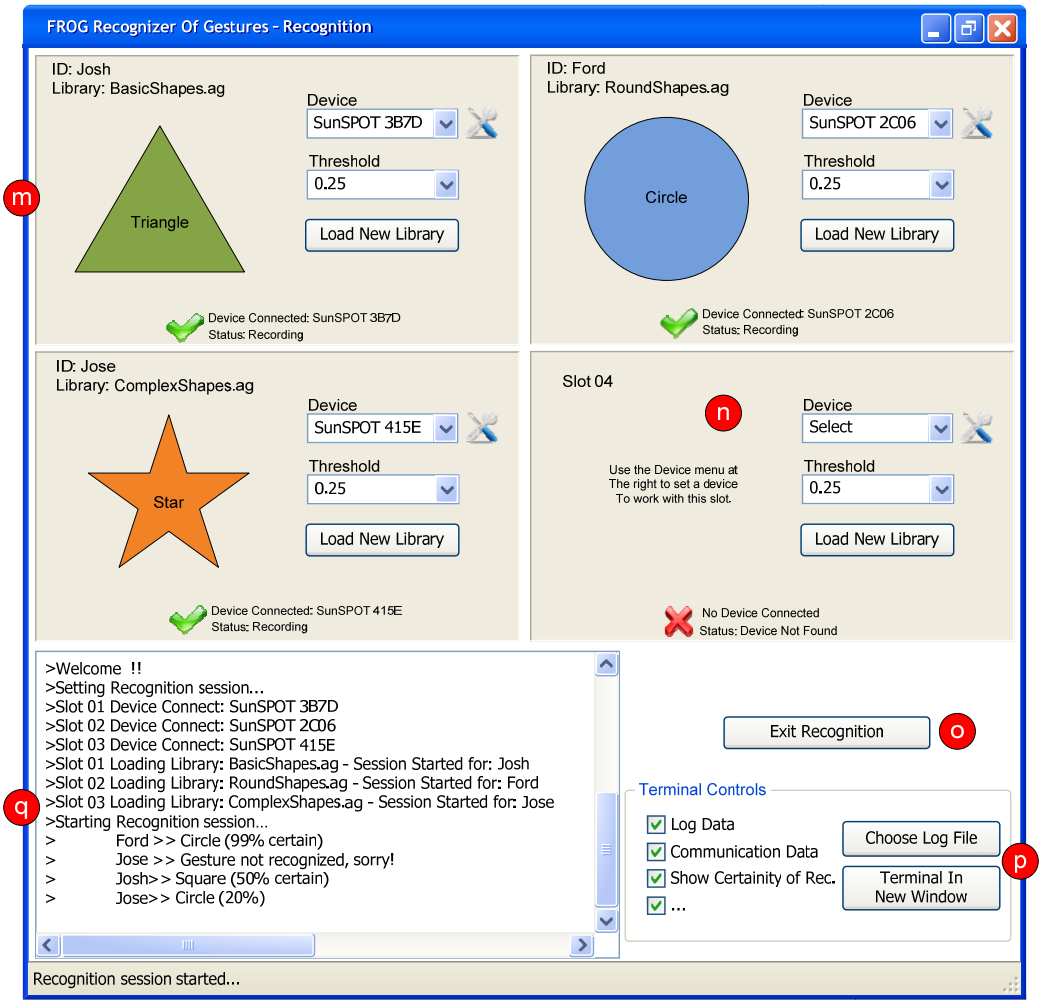
l) Terminal Controls: Refer to Screen 2.0, section h.

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.

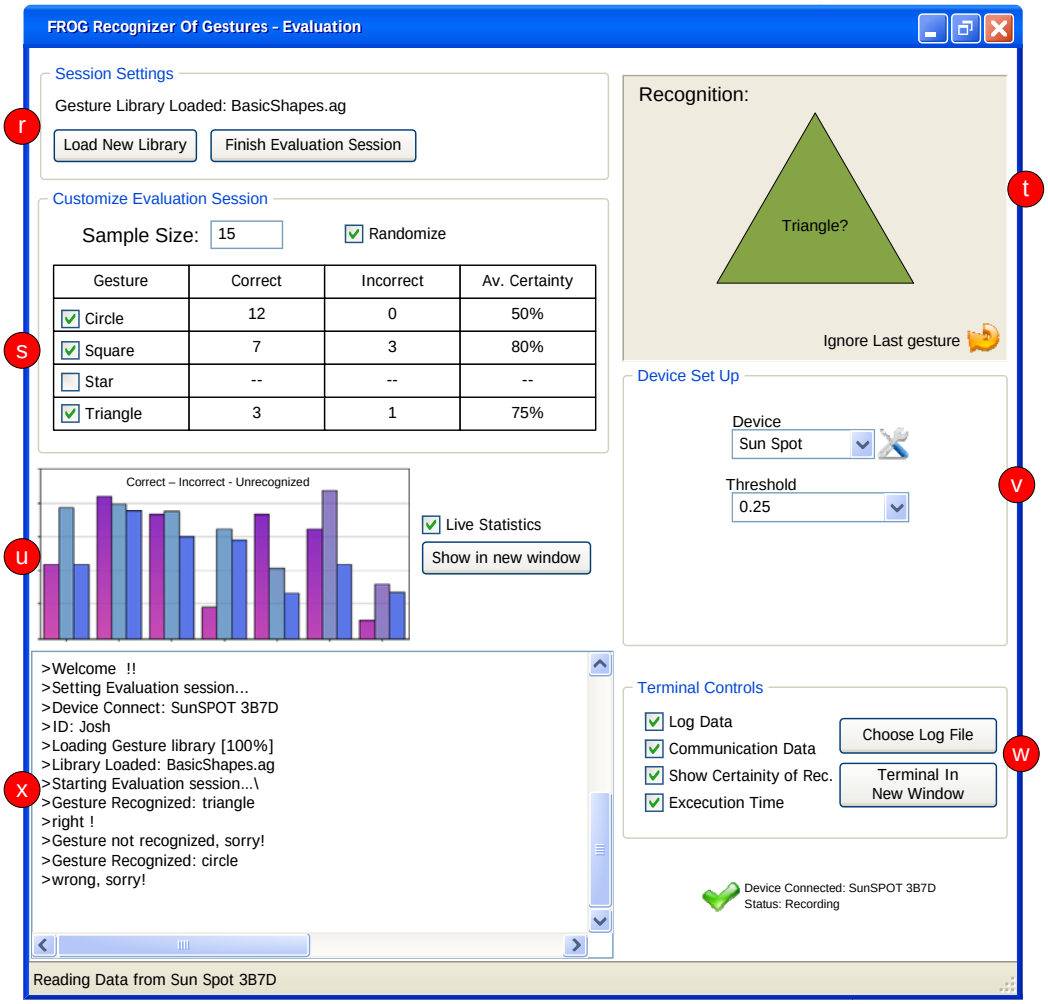
Screen 3.0: Recognition Mode



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



This is the window for Evaluation mode.

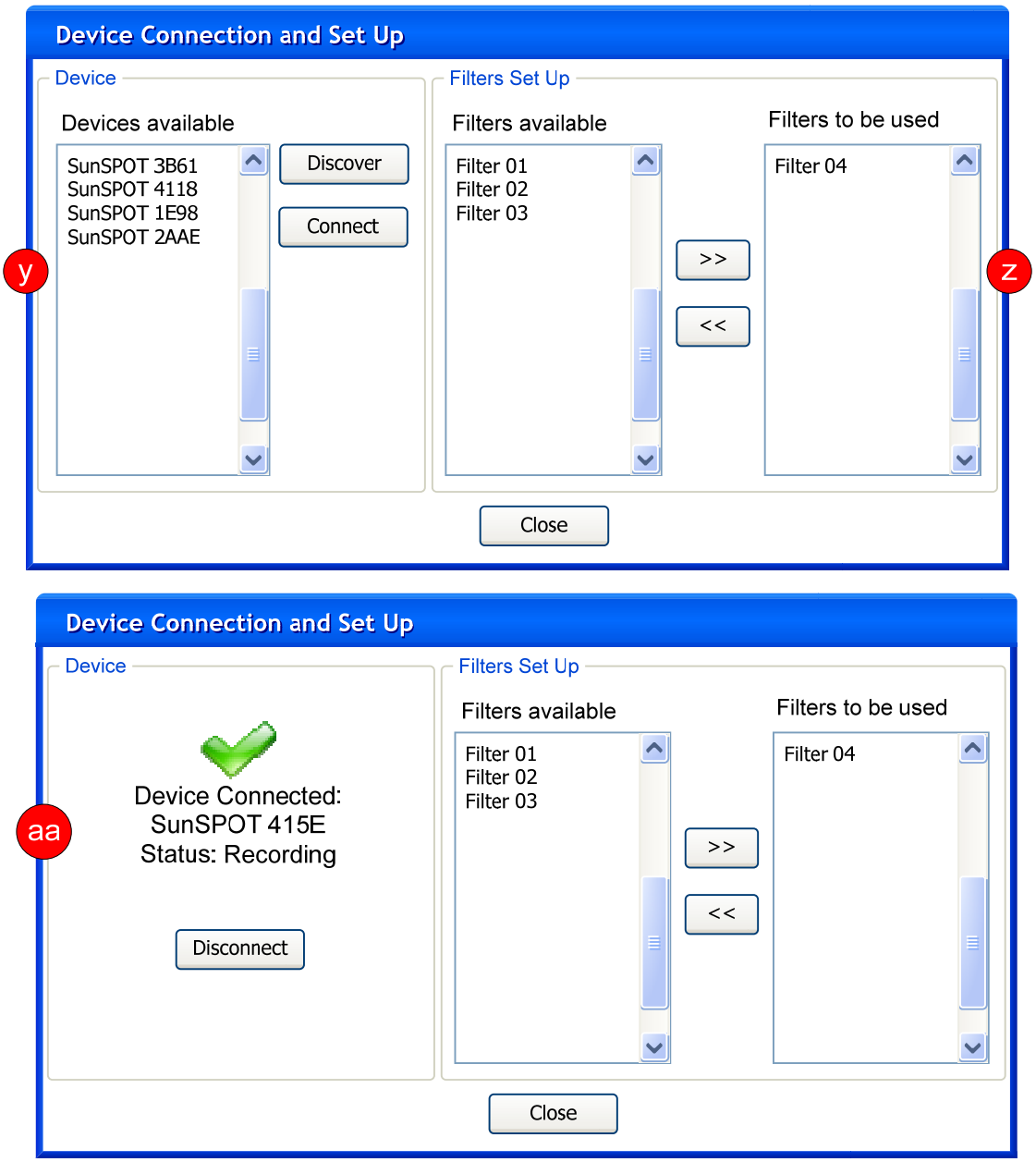
- **r) Session Settings**: This panel will allow the user to load a library or finish the session.
  - **i. Load New Library**: Refer to Screen 3.0, section m.
  - **ii. Finish Evaluation Session**: Will terminate the current evaluation session, close Screen 3.0, and return to Screen 1.0.
- **s) Customize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

Screen 6.0: Device Connection and Set Up



y) Devices: This panel will display devices that are successfully communicating with FROG. The list can be refreshed with the discover button, and a device can be selected with the connect button.

z) Filters Setup: This panel will display available filters in the left column and selected filters in the right column.

aa) Connection Dialogue: This panel will display the state of the selected device, including the device name and connection status.