Real-Time Motion Classification for Wearable Computing Applications

Richard W. DeVaul, Steve Dunn

December 7, 2001

Abstract

In this paper we describe the development of a real-time motion classification system for use in the MIThril wearable computing platform. The purpose of this motion classification system is to make important information about the user's state (whether the user is walking, running, standing still, *etc.*) available to other applications in real-time. We developed a two-layer model that combines a multi-component Gaussian mixture model with Markov models to accurately classify a range of user activity states, including sitting, walking, biking, and riding the T.

1 Introduction

Context awareness, or knowledge about the state of the user, task, or environment derived through non-explicit user input, is widely seen as a key to reducing the complexity of human-computer interaction tasks for portable and wearable computing applications. Context awareness requires statistical inference techniques to turn noisy sensor data into useful contextual information. Precisely what contexts are useful depends on the application, but a list of generally useful contexts include the time of day, the user's location, and user's activity state.

Time of day is easily obtained with a real-time clock, and outdoor location is comparatively easy to determine through GPS (indoor location is another matter, but may be partly addressed through the use of active tags[1]or the use of a

[1]... such as the Crystal tag system based on the Squirt IR active tags (`http://www.media.mit.edu/~rich/`) or the Locust system developed by Thad Starner, *et. all*.

triangulation-based room positioning system[2]The user's physical activity state, such as walking, sitting, riding the T, etc., is more complicated to determine and also quite important, since someone who is walking is likely to have less available attention than someone who is stationary, and someone who is riding a bicycle is likely to have even less than someone who is walking. The kinds of information that the user needs under these three conditions is also likely to be different. We became interested in the problem of determining the user's physical activity state in real-time using a high-precision three-axis accelerometer worn on the torso and the computing power available in the MIThril wearable computing platform.

2 Methodology

The methodology used for this project is as follows:

1. Gather annotated data from real-world activities.
2. Analyze this data and feature selection.
3. Develop a suitable generative or discriminative model.
4. Implement the model on the wearable and analyze the results.

2.1 Data Collection

The first step of the project was to gather labeled data while engaging in a range of typical daily activities, such as standing, walking, riding the T, riding a bicycle, *etc.* This involved putting on the wearable, running the accelerometer logging program and a text logging program, and going about daily activities.

2.1.1 Sensor Hardware

We designed a microcontroller-based accelerometer sensor using the ADXL202JE part from Analog Devices. This two-axis accelerometer is accurate to better than 2% over a ± 2 G range. Two of these accelerometers are mounted at right angles to each other, resulting in a four-axis sensor with two redundant axes.

[2]such as the LCS Cricket project, LCS Cricket Project `http://nms.lcs.mit.edu/projects/cricket`.

2.1.2 Sensor Software

Sensor logging code polled the accelerometer sensor approximately 47 times a second, reading all four axes. Each sample was time stamped with Unix time plus milliseconds and logged in a text file.

2.1.3 Annotation System

The data was annotated in real time by typing lines of text on the Twiddler chording keyboard (a commercially available one-handed key entry device). When a line of text was completed (when the "Enter" key was pressed) the text was time stamped with Unix time plus milliseconds and logged in a text file. This annotation scheme allows for precise matching of labels with data. Labels recorded in this way included single character abbreviation for common states, such as "w" for starting to walk and "s" for stopping, as well as longer free-form annotation, such as " damn, still breathing hard - i'm out of shape."

2.1.4 Data Acquisition

Over the course of a couple of weeks, One experimenter (Rich DeVaul) wore the wearable on a daily basis and generated a number of data sets of varying size and quality. Once the hardware and software glitches were ironed out, this left two large (order 1/2 hour) data sets of uniform sampling rate, good annotation, and interesting subject matter. One of these data sets was gathered late in the evening when Rich was literally running for the last T home. The other was gathered one morning when Rich biked in to work. Another extremely interesting data set that included a bike crash had to be discarded because the accelerometer data wasn't properly time stamped and the sampling rate couldn't be determined.[3].

3 Analysis

We started with a four-component signal vector composed of unsigned bytes (0-255) for each axis, sampled at rate of about 47.8 Hz, and irregularly spaced text labels. Both the labels and accelerometer samples were time-stamped with millisecond resolution and stored in flat text files. We had two useful data sets with

[3] Ironically the crash was caused by Rich answering his cell-phone - precisely the type of interaction we are trying to obviate through better context sensing in wearable communications apps. Both Rich and the wearable were fine.

a combined total of 198,855 samples for 4,156 seconds of data and about 110 meaningful labels.

A pre-processing step averaged the two redundant components of the four-axis accelerometer vector to produce a three-vector. To simplify the representation and reduce dimensionality we converted the acceleration vector to a scalar magnitude. Experimenting with spectrograms in Matlab suggested that a power spectrum of the magnitude might be a good feature for classifying motion state. To reduce the computational burden on the wearable we chose a comparatively small 64 element FFT window (spanning 1.34 seconds) with 32 samples of overlap, producing one 32 element power-spectrum every 0.67 seconds. Because our data included a large DC offset that caused numerical stability problems we threw away the DC component, leaving a 31 dimensional feature vector.

4 Modeling

4.1 BIC, EM and Gaussian Mixture Model

Since our goal was to develop the simplest possible multi-class activity model (and we had known-working Gaussian mixture-model code in Matlab) we decided to use Basian Information Criterion and EM to cluster the data. Much to our surprise, this worked fairly well; Stating with our first data set, BIC chose a five-component model with one component clearly associated with running and another with walking.

Since our goal is to implement real-time classification on the wearable, the next step was to implement feature-extraction in C code that we could compile on the wearable. We chose FFTW[4] for the basis of our power-spectrum feature extraction code because it is both fast and free. Unfortunately, the transforms produced by FFTW differ from those produced using Matlab's FFT function. That, and a bug in some "known working" Matlab problem-set code[5] caused us no small amount of confusion until we realized what was going on.

With our FFTW-generated features, we re-ran BIC on the full labeled data set and produced an eight-component mode that also had strong clusters for walking and running, but other labeled activity sates such as sitting still or riding the T

[4]"The Fastest Fourier Transform in the West," – see FFTW home page `http://fftw.org/` for more information.

[5]There is a parenthesisation error in multigaussian.m that causes incorrect density evaluation.

appeared to have high-order dynamics that the model could not capture.

After implementing the multi-Gaussian model evaluation on the wearable and verifying that the class conditional probabilities were the same as in our Matlab implementation, we turned to the problem of better capturing the dynamics of our data.

4.2 Combining Models: Gaussian Mixtures and Markov Chains

Once we realized there was dynamic structure in our data that our model wasn't capturing we had several choices.

1. We could choose a larger window for our FFT in the hopes that the dynamic features of interest would fall within the larger window and rerun our BIC/EM model selection process.

2. We could keep our shorter feature vector and attempt to build an HMM that captured the dynamic features of interest.

3. We could use the output of our existing model and construct a Markov model on top of it that would capture the dynamics.

The first choice seemed infeasible due to the large computational requirements it would impose. The second approach seemed to be the most principled, but would have required additional theory (HMMs with continuous observable states) and code to implement. The third approach, while perhaps not optimal in performance, required little additional code beyond what we had already written or used for problem sets, could be implemented easily in C, and was thus the obvious choice.

The result are six first-order Markov models running on the output of the eight-component Gaussian mixture model. The Markov models represent the following states:

1. Garbage - a model trained on a section of the combined data set we don't care about. See Figure 1 for source accelerometer data and spectrogram and Figure 2. for the Gaussian component posteriors and Markov model classification results for this data.

2. Sitting - a model trained on data of Rich waiting for the T. See Figure 3 for the source accelerometer data and spectrogram and Figure 4 for the Gaussian component posteriors and Markov model classification results for this data.

Figure 1: "Garbage" class training data

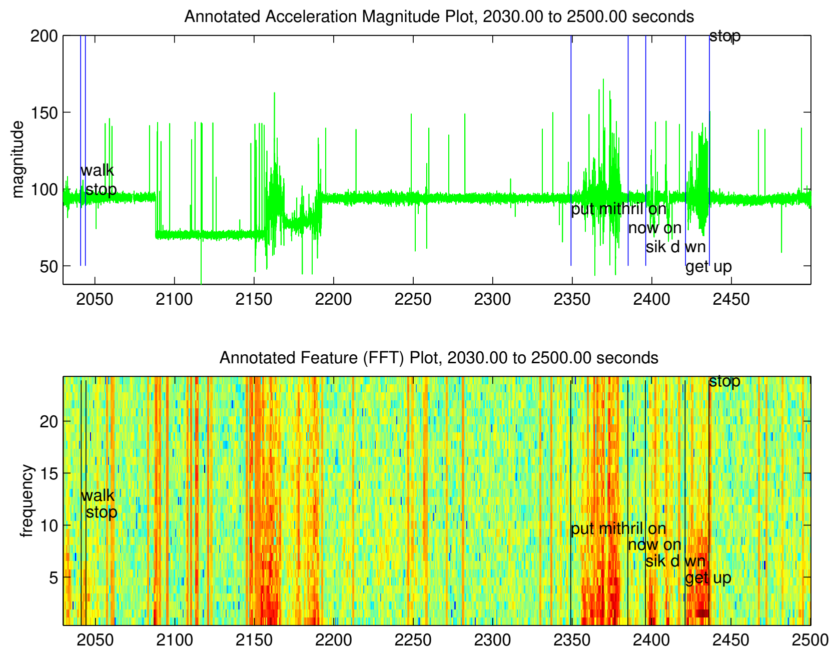


Figure 2: "Garbage" class model output

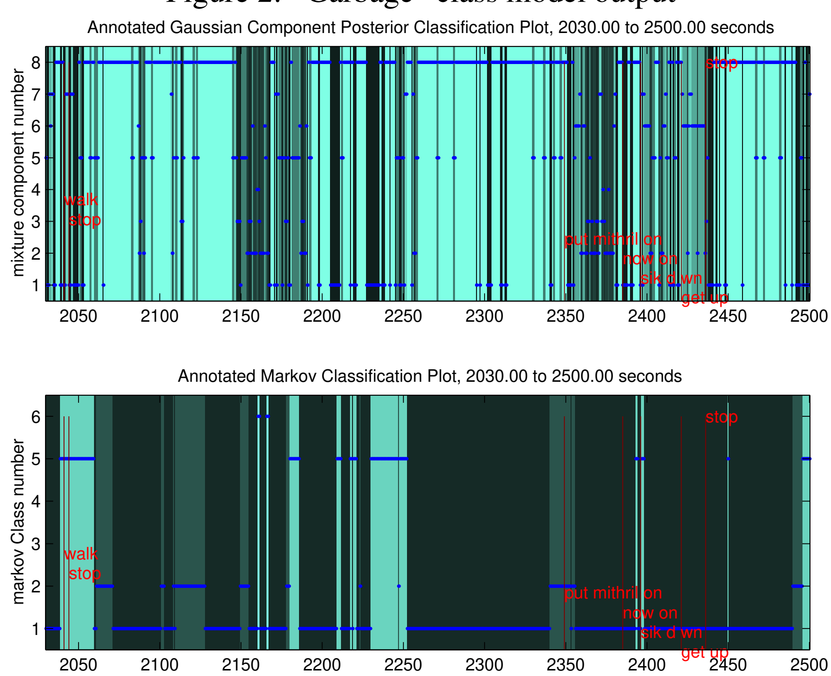


Figure 3: Data for “Sitting” class, with margins.

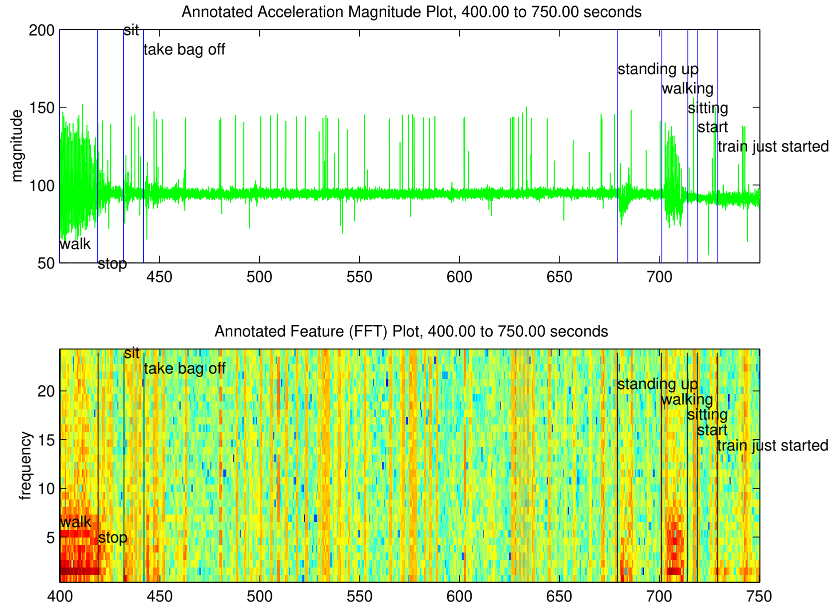


Figure 4: ”Sitting” model classification results.

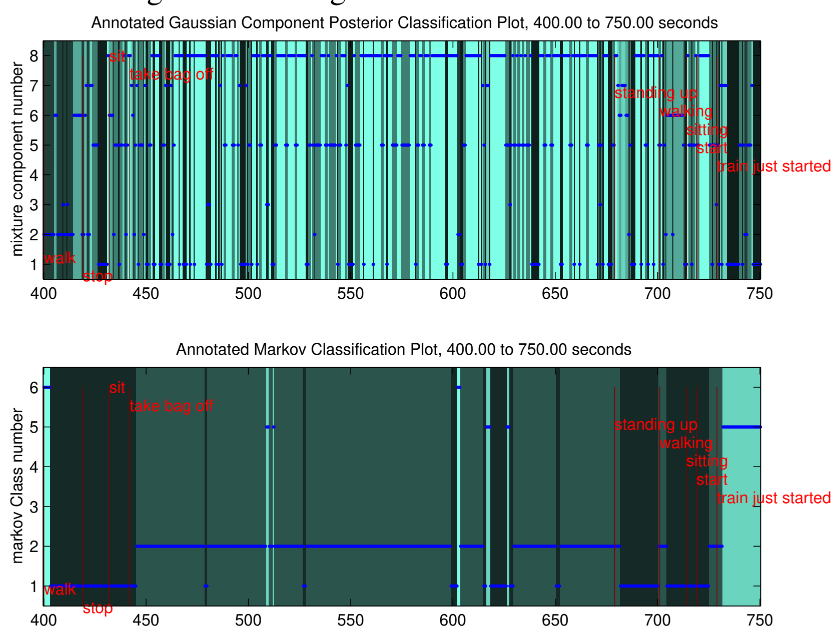
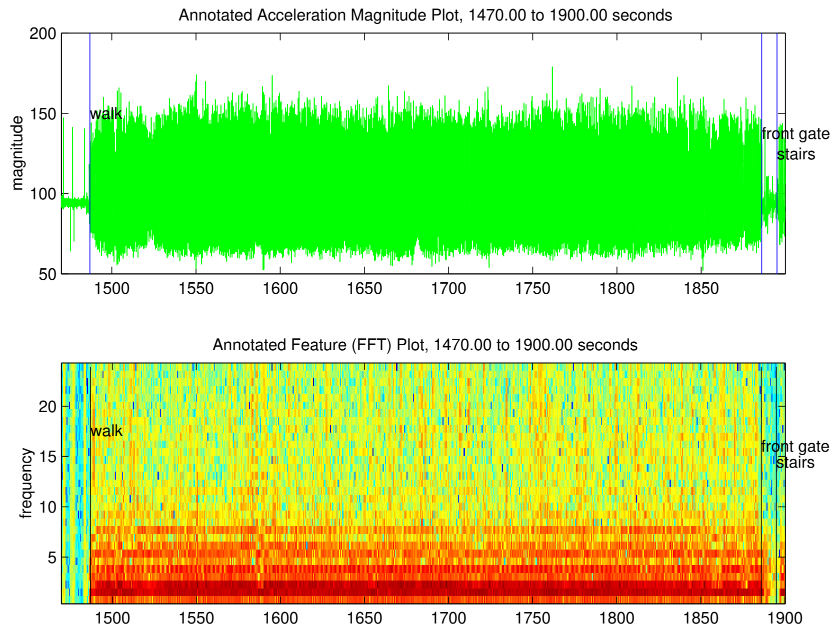


Figure 5: Data for walking class, with margins.



3. Walking - a model trained on a long stretch of Rich walking from the Davis square T stop to his house on Kidder Ave. See Figures 5 and 6

4. Running - a model trained on two short sections of Rich running from the Media Lab to the Kendall T stop. See Figures 7 and 8..

5. Riding the T - a model trained on a long section of Rich riding the T from Kendall to Davis. See Figures 9 and 10.

6. Riding a bicycle - the most complicated model. A model trained on a long section of Rich biking from his home in Davis Square to the Media Lab, including various pauses, stops, and changes in intensity characteristic of bike commuting in Boston. Figures 11 and f12.

The figures compare the results of the Gaussian class conditional assignments and the Markov Model classification running on those assignments as inputs. The Markov Model classification results are the result of 20-symbol inputs, which represents a delay of 13.4 seconds for the end of the window to catch up to the beginning of a new activity state.

Figure 6: Classification results for “Walking” data.

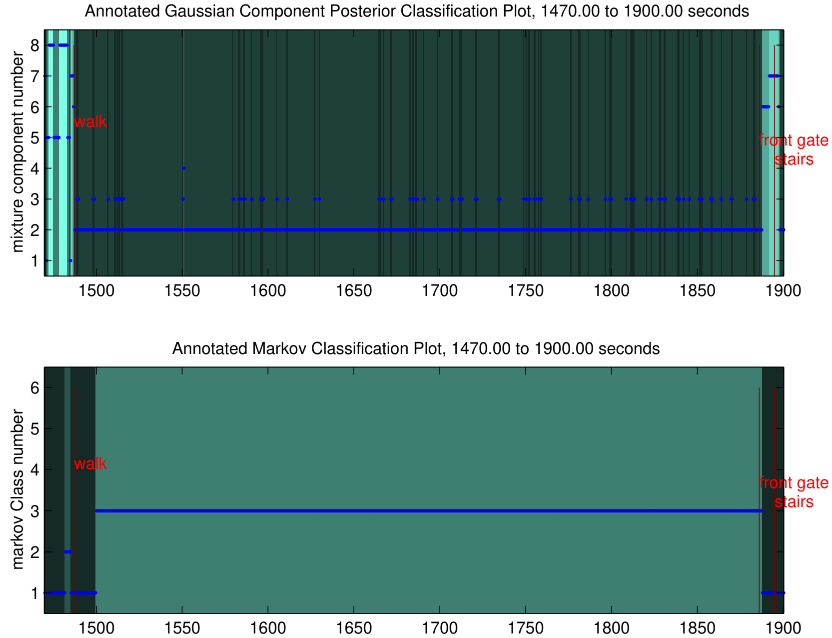


Figure 7: ”Running” data with margins

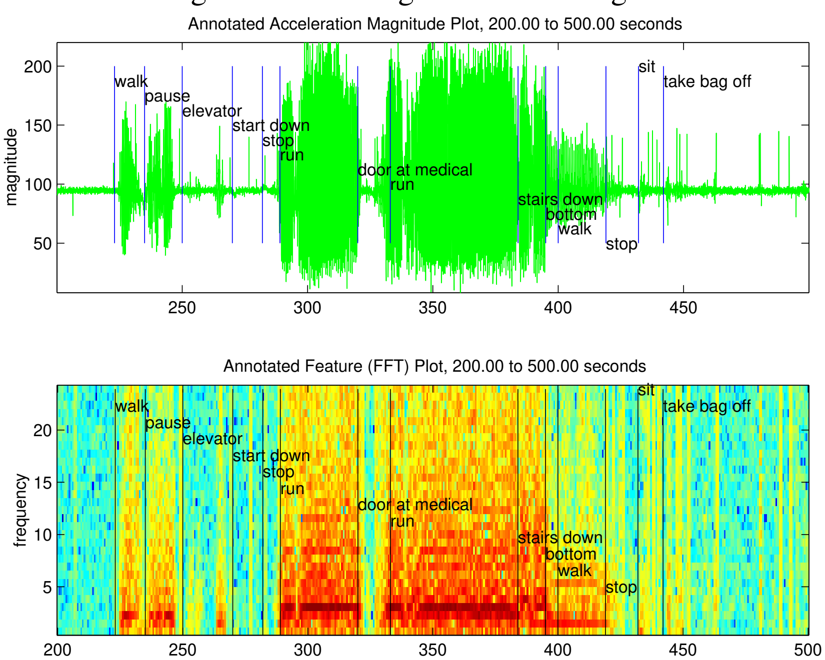


Figure 8: Classification results for "Running" data

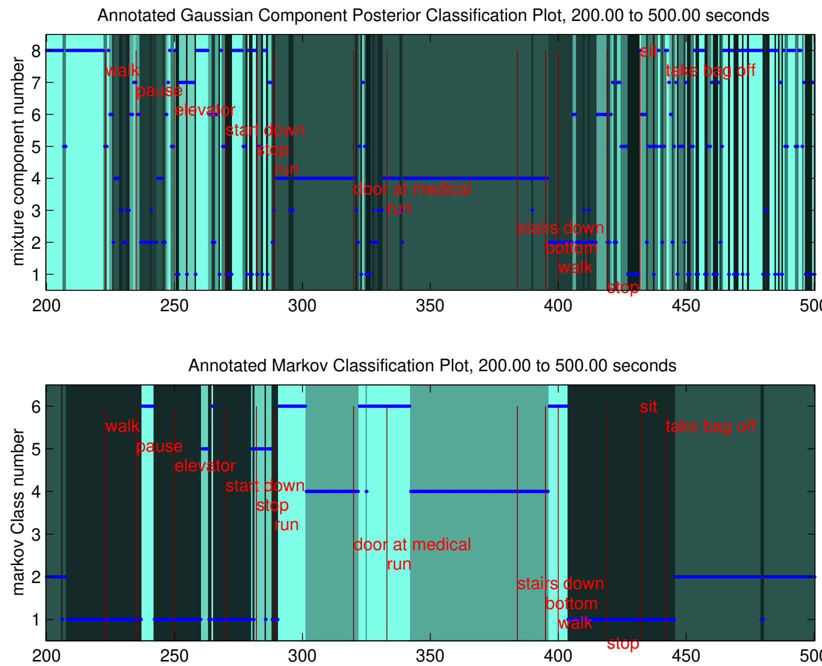


Figure 9: Data for "Riding the T" with margins.

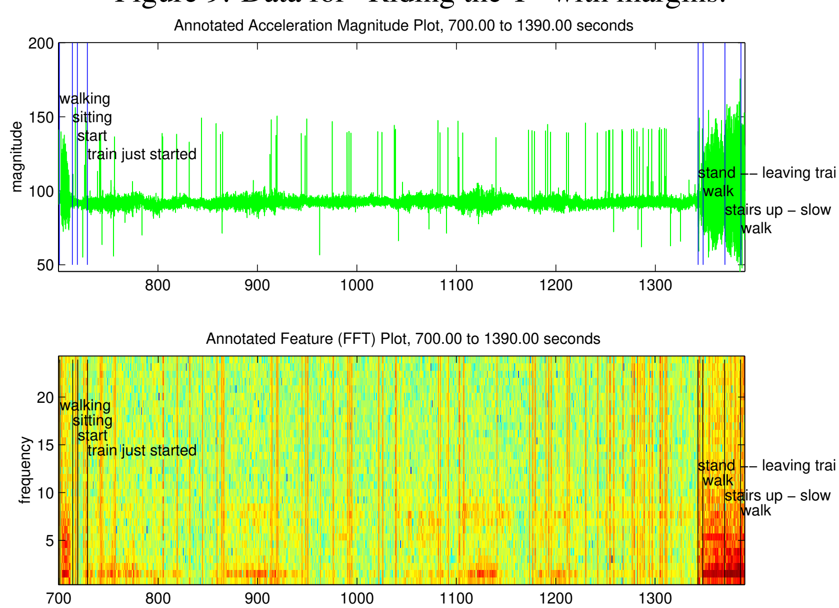


Figure 10: Results of classification for “Riding T” model.

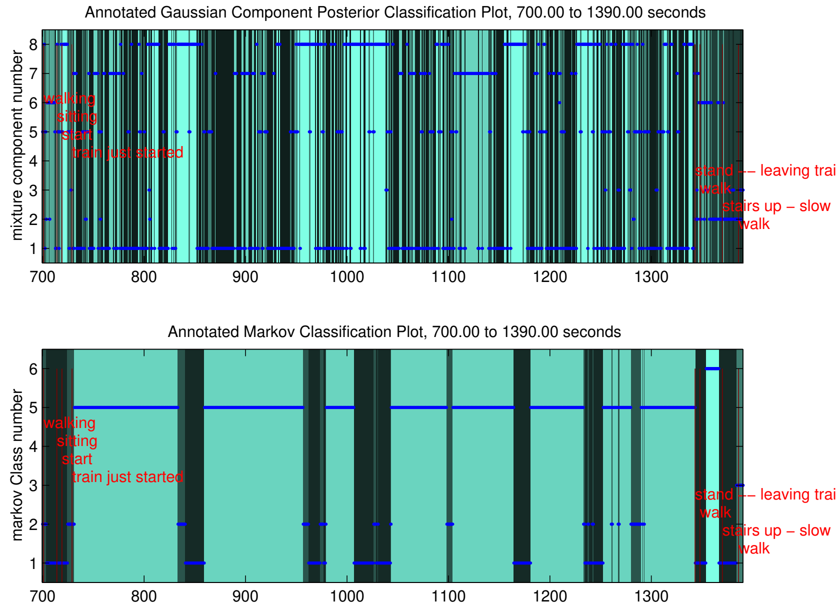


Figure 11: Data for “Riding a bicycle” plus margins.

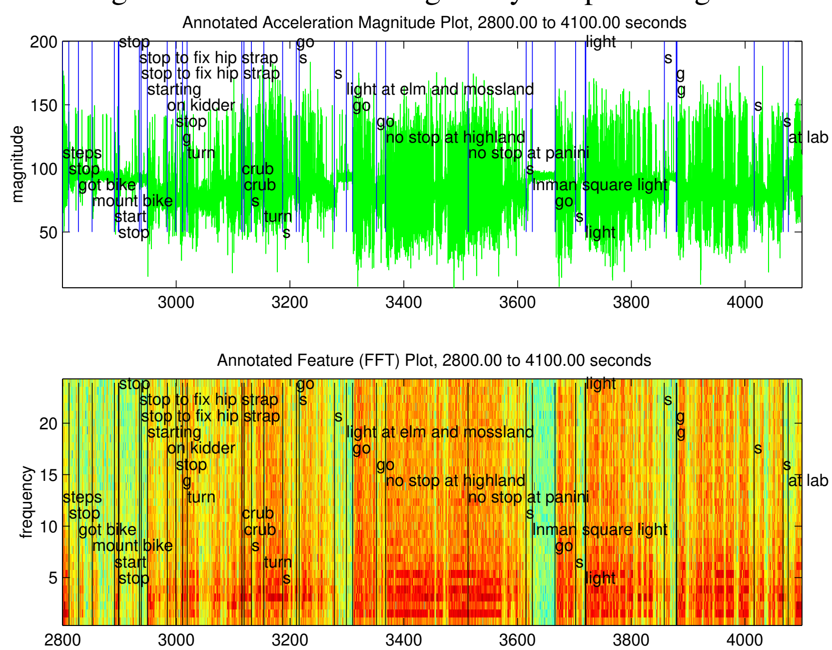
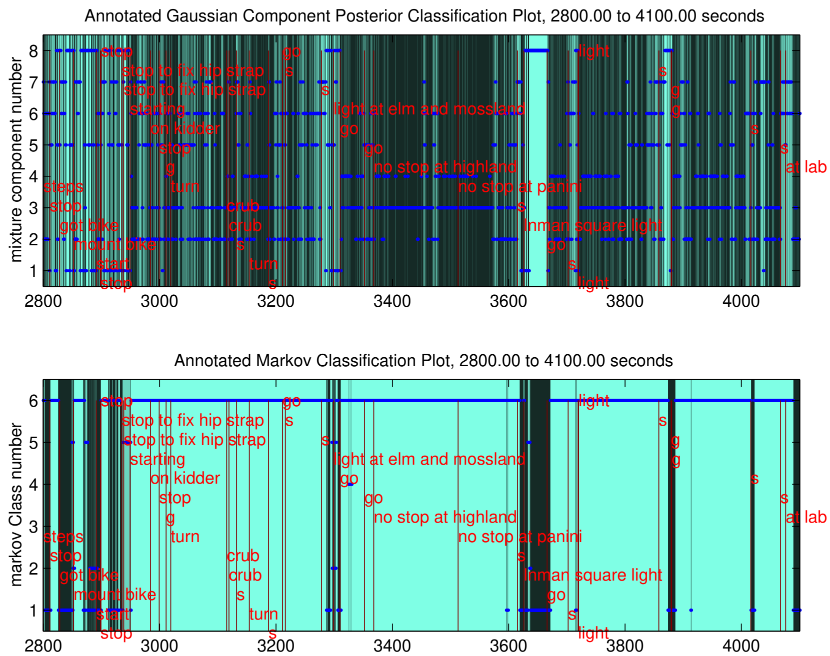


Figure 12: Results of classification for “Biking” data.



The Gaussian class conditional assignments do a good, though noisy job, of classifying the comparatively unambiguous actively states of walking *vs*. running *vs*. sitting, and do so quite quickly; a delay of a single FFT window (1.34 seconds) is all that is necessary. However, the graphs indicate that all of these states are easily confused with the more complex activity of biking, and sitting still *vs*. riding the T are essentially indistinguishable even at the one-count dynamics level.

The first-order Markov model classifier does a much better job of classifying the five activities of interest with a 20-symbol input. Although the inherent imprecision in the labeling process makes quantifying the “true” activity state of each sample difficult, the results of classification using the first-order Markov models appears surprisingly good, especially considering the complexity of the data and the arbitrariness (from the point of view of the Markov model) of the Gaussian mixture model parameters.

5 Implementation

The goal of this project is a real-time implementation of a motion classifier system running on the MIThril wearable computing platform. This requires that the steps of sensing, feature extraction. model evaluation, and classification take place in real-time in an environment of limited computing resources. Needless to say, running Matlab on MIThril is right out.

A the time of this writing, we have implemented and tested every part of the classification process except the Markov model evaluation. The feature extraction and Gaussian class-conditional posterior code produces identical results on the wearable as on the desktop under Matlab, running in real-time on a 200MIPS StrongArm with 32MB of ram. We have every reason to believe that the addition of first-order Markov model evaluation will be possible as well.

6 Conclusions

The goal of developing and implementing a real-time motion classification system for the MIThril wearable is largely accomplished. We speculate that the hybrid Gaussian mixture model/Markov model we developed may not be as accurate as a monolithic HMM, but may present certain advantages as well. For instance, the class conditional posteriors could be used to provide a rapid "is the user moving" classification for applications that are interested in only coarse motion data, while applications requiring more precision could use the results of the Markov model classification. Further, it should be possible to train new Markov models, perhaps of higher order or for different activity states, without retraining the underlying mixture model.

One concern is over fitting; We used all of our labeled data as training data so we have no way of empirically demonstrating the model's generalization. Our use of BIC in model selection provides some assurance that our Gaussian mixture model has a reasonable expectation generalization, but we have no such assurance with our choice of Markov model parameters. Only more data and testing will tell.

Future directions include completing the implementation of the Markov model evaluation on the wearable and developing the infrastructure to support other types of models or classifiers such as HMMs or SVMs.

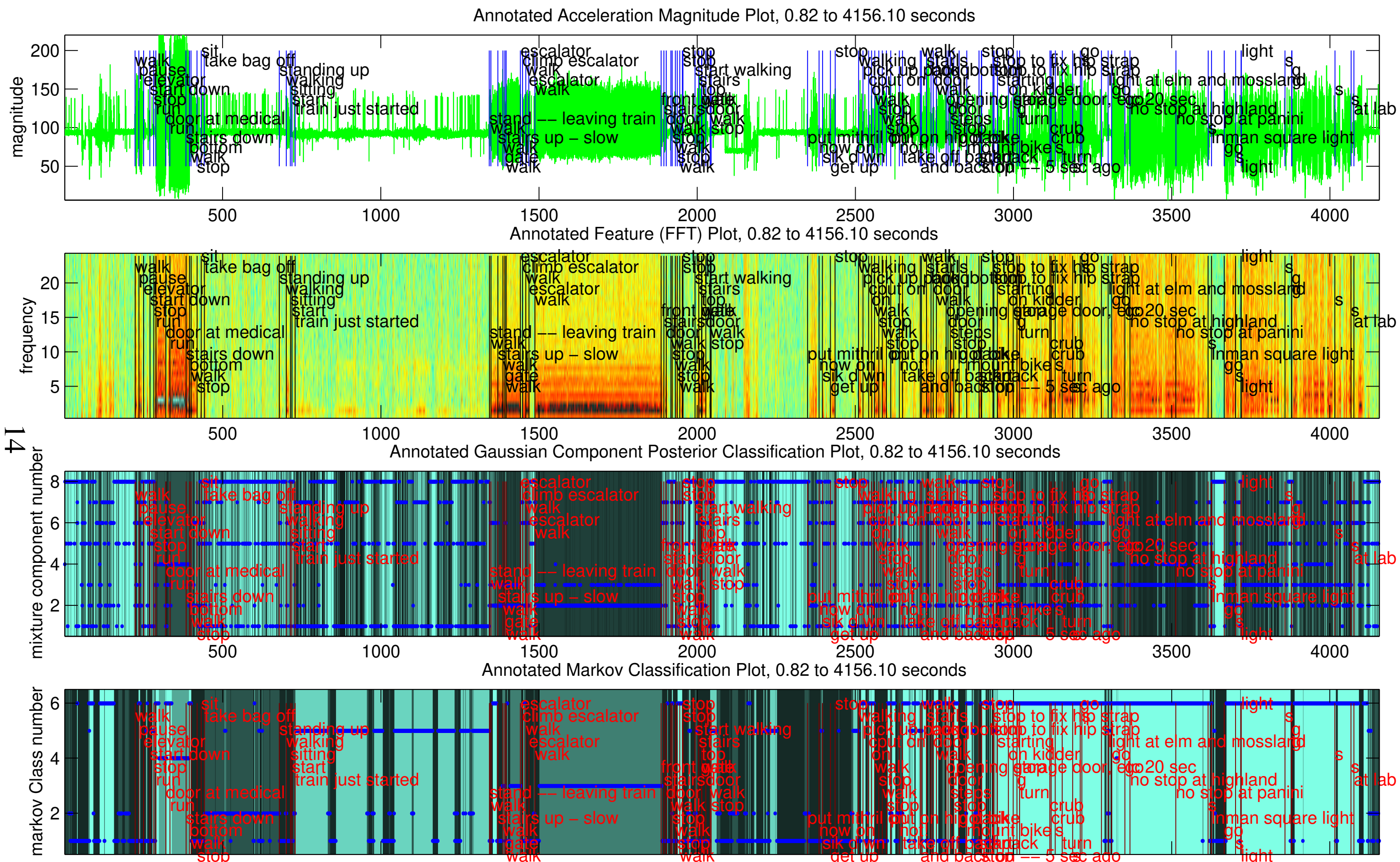


Figure 13: Full Data Set with Classification