learning phase, the system will ask the user to perform the designed body movements while she is in different walking contexts (walking in office, walking at home, walking on the parking lot, etc.). The system will then run clustering algorithms on the collected accelerometer data, and cluster them into a small number of groups, representing the data within her typical walking contexts. For each such group, we maintain a certain number of reference data.

In the classification phase, we adopt a two-level classifier. At the top level, we determine whether the user is stationary or mobile. This is rather straightforward because walking in general has much more energy than music-stimulated body movements as shown in Figure 5. At the bottom level, if the user is walking, we classify the test data against the user's walking reference data group by group. If none of the groups return TRUE, we reject the user.

3.2.4 Stronger Authentication Through Body Movement Based Passwords

Like any behavioral biometric characteristic, the body movement pattern may fail to differentiate users in some cases. For those users who desire a more secure authentication method, we propose to use movement-based passwords, which offers more security, reduces the processing demands on the device, but requires the user to remember their passwords.

Like traditional text passwords, body movement passwords also contain a list of elements – how many elements are in the password and what they are determine the password. However, unlike tradition passwords in which each element is a character, each element of the body movement passwords is a type of body movement. For example, a password for a smart glass user can have the following four movements: a head nod, an eye blink, a head nod, and a head shake; a password for a smart watch user can have the following four movements: upward movement, right-award movement, left-award movement, and downward movement. For each element, the system also designs a unique prompt tone that the user needs to remember.

The proposed password-based authentication system works as follows. First, we ask the users how many elements are contained in her password. Then for each element, the system will play a tone to mark the beginning of the element. We make sure two consecutive tones will be reasonably apart so that the user has sufficient time to finish each movement. In this case, the system doesn't need to differentiate users solely based upon their movement signatures, but the system only needs to recognize the movement. We will also provide a list of supported movements such as a head nod, a head shake, or an eye blink. Finally, for each authentication session, the system plays the prompt tones in a *random* order to avoid shoulder surfing. Compared to using body movements as biometrics, this method is simpler, more secure, and easier to reconfigure than purely movement pattern based authentication.

4 Challenge II: Building an Efficient Body Movement Based Authentication System

The second challenge we will deal with in this project is to ensure the proposed authentication system can run on seriously resource-constrained wearable devices. Our preliminary results show that even a highly simplified classifier will incur high processing latencies, sometimes leading to the catastrophic device overheat exception. To tackle this challenge, we propose to carefully select efficient features and classifiers, pipeline the authentication operations, and dynamically adapt the sampling rate.

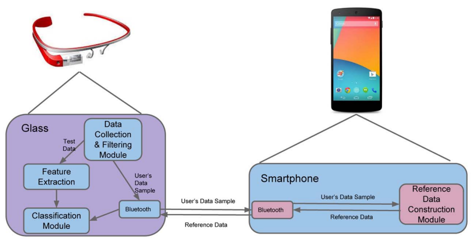


Figure 6: The software modules for the Headbanger authentication app we implemented in the preliminary study.

4.1 Preliminary Work

Wearable devices have severe resource limitations in many aspects; to name a few, energy, computing, networking and storage. Building an efficient authentication system that can smoothly run on such devices, therefore, becomes a significant challenge. In order to investigate whether such a goal is attainable, we have implemented a *Headbanger* app on the Google Glass. Figure 6 shows the software modules the app consists of. In the implementation, we established the reference data on the smartphone since this step is offline, consumes the most resources, and doesn't need to run on the device. The online authentication process is implemented on the device. In the implementation, we adopted a much simplified classification method. First, instead of having multiple reference samples, we only use one reference sample for each user. After computing the DTW distance between the test sample and the reference sample, we simply compare the distance to a pre-set threshold value to determine the classification result. In this implementation, we kept the algorithm to the bare minimum to test the processing capability of the Google Glass and did not worry about the classification accuracy.

Figure 7 shows the measured processing latency (the time that elapsed between when we finish collecting the test sample and when the authentication result is generated). The results show that even with a significantly reduced implementation, the processing delays are still rather substantial. For example, if the sample duration is 10 seconds, the processing delay is a little more than 20 seconds! More importantly, if there were other processes such as camera running at the same time, or if we continuously ran the *Headbanger* app for multiple times, then the processing became very slow, and the glass became overheated and displayed the following message "It is too hot. Glass needs to cool down." Then we needed to wait for 1 or 2 minutes for the glass to cool down. After the glass finally cools down, we had to start over the authentication process from the beginning. We refer to this catastrophic event as *glass overheat exception*.

sample duration (s)	processing delay (s)
2	1.29
3	2.74
6	9.04
10	20.48

Figure 7: Measured processing latencies on Google Glass with different sample durations. In this set of experiments, we tried to understand the processing capability constraints and didn't optimize the code.

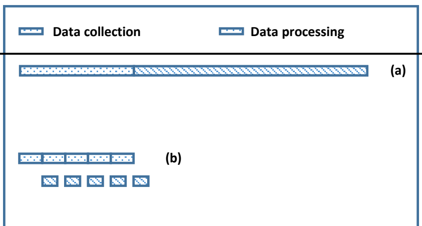
From the preliminary investigation, it becomes very natural that the second research challenge we have to address in this project is to optimize the system design of *Headbanger* to enable efficient execution on severely resource-constrained wearable devices.

4.2 Proposed Research

We propose to investigate several runtime optimization techniques to make *Headbanger* suitable for wearable devices without changing the off-the-shelf hardware by minimizing the authentication latency and energy consumption. Please note that we could choose to offload computation to the device-paired smartphone to conserve cycles/energy on the wearable device, but *we will not pursue this route in this proposal.* Instead, we focus on *direct authentication* where wearable devices are not dependent upon smartphones or other more powerful devices for authentication, which we believe is more convenient.

As discussed in Sec 3, a *Headbanger* authentication session consists of the following steps: (1) collecting test data, (2) extracting features from the test data, and (3) classifying the test data. Among these three steps, steps (2) and (3) usually consume more processing cycles/energy. In this project, we focus on optimizing these two steps. Before we present the proposed optimization techniques, we note that the first thing we need to do is to carefully select features and classifiers in *Headbanger* from the set of features discussed in Sec 3.2.1. We already know that using too many features may lead to noises and classification errors [26], but on wearable devices, this is even more catastrophic as it likely leads to device overheat. As a result, we need to pay extra attention to what features we use and which classifier we adopt on the device. In the project, we will carefully measure the power consumption, processing delay, and classification accuracy of different combinations of features and classifiers, and then choose accordingly.

Pipelined Authentication: An important parameter for *Headbanger* is the sample duration T. So far, we assume that the system first collects ACC sample for T sec-


Data collection
Data processing
(a)
(b)