RECOGNIZING SEATBELT-FASTENING ACTIVITY USING WEARABLE SENSOR TECHNOLOGY

An Undergraduate Research Scholars Thesis

by

JAKE LELAND AND ELLEN STANFILL

Submitted to the Undergraduate Research Scholars Thesis program
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by Research Advisor: Dr. Tracy Hammond

May 2017

Major: Computer Science

ABSTRACT

Recognizing Seatbelt-Fastening Activity Using Wearable Sensor Technology

Jake Leland and Ellen Stanfill
Department of Computer Science
Texas A&M University

Research Advisor: Dr. Tracy Hammond
Department of Computer Science
Texas A&M University

Many fatal car accidents involve victims who were not wearing a seatbelt, even though systems for detecting such behavior and intervening to correct it already exist. Activity recognition using wearable sensors has been previously applied to many health-related fields with high accuracy. In this paper, activity recognition is used to generate an algorithm for real-time recognition of putting on a seatbelt, using a smartwatch. Initial data was collected from twelve participants to determine the validity of the approach. Novel features were extracted from the data and used to classify the action, with a final accuracy of 1.000 and an F-measure of 1.000 using the MultilayerPerceptron classifier using laboratory collected data. Then, an iterative real-time recognition user study was conducted to investigate classification accuracy in a naturalistic setting. The F-measure of naturalistic classification was 0.825 with MultilayerPerceptron. This work forms the basis for further studies which will aim to provide user feedback to increase seatbelt use.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis (or) dissertation committee consisting of Professor Tracy Hammond of the Department of Computer Science & Engineering and Graduate Student Mentor Mr. Josh Cherian of the Department of Electrical and Computer Engineering.

The data analyzed was collected and all user studies performed by the undergraduate authors of this thesis, Jake Leland and Ellen Stanfill, under the advisement of Dr. Tracy Hammond and Mr. Josh Cherian. The undergraduate authors also wrote and designed their own IRB, also under the advisement of Dr. Tracy Hammond and Mr. Josh Cherian. The IRB was approved on October 15, 2017, and was given the number: IRB2016-0705D.

All other work conducted for the thesis (or) dissertation was completed by the student independently.

Funding Sources

Equipment, materials, and funds to support the graduate mentor, Mr. Josh Cherian, were supplied in part by the TEES AggiE_Challenge program.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Motor vehicle crashes are a leading cause of death of Americans under the age of 60. More than half of those under 45 who died in crashes in 2014 were not wearing a seatbelt at the time of the crash [1]. According to statistics published by the CDC, young adults are the least likely age group to wear a seatbelt [2, 3]. Furthermore, men are less likely to wear their seatbelt than women [3]. Detection and warning systems exist to determine whether a driver or passenger has fastened their seatbelt. Most of these mechanisms are integrated within the car [4] or buckle itself [5] and provide interventions such as a tone or a light on the dashboard [6]. However, these existing mechanisms do not appear to be sufficient to prompt safe user behavior.

Activity recognition has received much attention in recent years with the advent of portable personal computing devices like smartphones and smartwatches. Data is collected from sensors, often accelerometers [7], within these devices. After the data is preprocessed according to standard techniques [8], it is used to train machine learning classifiers to recognize the activities being performed. Activity recognition is central to achieving context-aware computing, which aims to change the behavior of systems based on what the user is doing at the time of use [7]. Activity recognition also serves as the foundation for many health-related applications which monitor eating habits or other self-care activities [9–11] and intervene to correct undesirable behaviors [7, 12, 13]. However, this technology has not yet been applied to the task of checking whether a driver has fastened their seatbelt. A more detailed review of prior research is included in the next section.

This work seeks to apply activity recognition to accelerometer data collected from a smartwatch in order to identify when the user has bucked their seatbelt. A system based on such recognition would have many applications. The system could detect that the user

is moving at the speed of a car, check whether the user has buckled their seatbelt, and then intervene to correct the user's behavior. Such a system could also be used to track user behavior, i.e. for insurance purposes. A behavior tracking and intervention system would be particularly useful for young drivers. Parents could monitor the safety habits of their teenaged driver, while insurance companies could make use of the data gathered by the system in order to more accurately determine insurance rates. Because this system would correspond to the individual driver and not the car, system data would not be contaminated by the bad habits of other drivers and would thence be useful to set individual insurance rates.

Additionally, the system would have an advantage over existing seatbelt-detection systems because these systems can be easily beaten. For example, people may buckle their seatbelt across the seat and then sit on top of it in order to quiet the warning tone. There are also not typically any seatbelt-detection systems installed in the middle or back rows of the seats in a car. A wearable system does not require the user to sit in a specific car seat in order to function. It would simply need to check the speed at which the user is travelling. Thus, the system would be useful for all passengers in a car, not just the driver.

In this work, the feasibility of such activity recognition is first investigated by attempting to classify the motion using features of the data commonly found in literature. Recognition accuracy is then improved by incorporating novel features unique to the action of putting on a seatbelt. Finally, recognition is tested in real time with moderately successful results, indicating that our approach is effective in recognizing the buckling motion.

2. RELATED WORK

When designing an activity recognition system, several factors must be considered, as laid out by Lara and Labrador [10]. The four most essential of these factors are:

1. *Selection of attributes and sensors*: It is desirable that only the features which most effectively discriminate between activities are observed. To be useful, a recognition system should also minimize costs as far as possible. Therefore, a system should be composed of carefully selected sensors which collect only the data which is necessary.

2. *Obtrusiveness*: In order for a recognition system to be practical for everyday use, it must not interfere with the user's activities.

3. *Data collection protocol*: Data is typically collected in a laboratory setting, but a more naturalistic protocol may allow the recognition system to better scale to real-life applications.

4. *Recognition performance*: The system must accurately recognize the desired activity with minimal false positives or false negatives.

Each sub-areas of activity recognition performs better on some of these factors than on others. Work in these sub-areas will be discussed next.

2.1 Vision-Based Recognition

Activity recognition can be partitioned into vision-based and sensor-based recognition [7]. Vision-based recognition uses cameras or other imaging platforms to monitor a user's behavior and interaction with the environment. An example of vision-based recognition is found in [14]. Although sensor-based recognition is the focus of this paper, some work has

been done using low-resolution infrared imaging to identify and track driver postures [15]. The costs associated with this method are lower than those of other imaging systems, but they are still substantial. In contrast, this work uses off-the-shelf smartwatches to perform recognition, which cost less, are not obtrusive, and can be used for additional purposes besides recognition.

2.2 Sensor-based Recognition

Sensor-based activity recognition may involve sensors attached to the user (*wearable sensing*) or sensors attached to objects in the environment (*dense sensing*) [7]. Examples of object interaction recognition using dense sensing can be found in [16–19]. Eye-tracking and associated applications fall into this category as well because the sensors are integrated with the environment, not the user [20–22]. Existing seatbelt-buckling detection systems [4, 5] are examples of dense sensing because they rely on sensors within the vehicle. Wearable-based activity recognition was chosen for this task instead of dense sensing because the recognition application may be implemented on a platform (in this case, a smartwatch) that the user already owns and uses for other tasks.

2.2.1 Recognition Through Wearable Sensing

Much work has been done with wearable-based recognition in literature. Early work used multiple accelerometers [23] or many other types of sensors [9, 24–27] attached to various locations on the body. These other types of sensors included barometers, gyroscopes, heart rate sensors, humidity sensors, light sensors, microphones, and thermometers. Lester et al. noted that not all of these sensors were necessary to successfully classify user activities and demonstrated similar precision and recall using a sensor subset containing accelerometers, microphones, and barometers [9]. In each of these experiments, sensors were arranged in custom-built arrays which are intrusive and impractical for widespread use. Two notable exceptions are Maurer et al. (2006), which used sensors

integrated with an eWatch, a platform similar in shape and size to a modern smartwatch, and Győrbíró et al. (2008) which used MotionBands that were also similar in shape and size to a smartwatch [26, 28]. However, neither of these platforms are widely available and they cannot be used for other tasks beyond sensor data collection.

Further research has demonstrated that collecting data from only a few biaxial or triaxial accelerometers is sufficient to recognize many activities [7, 23, 25, 28–31]. Bao and Intille (2004) provides a summary of previous sensor-based works which collected data from two or more accelerometers positioned at multiple locations on the body [32]. These data were used to recognize multiple activities, typically some combination of ambulation (walking, running, etc.), posture, typing, talking (gesticulating), shaking hands, writing, and other activities. These previous studies were largely conducted in a laboratory setting, so Bao and Intille implemented a semi-naturalistic collection protocol [32]. Again, these studies used multiple custom sensor arrangements, which are intrusive and not easily obtained by consumers.

Recent work has focused on off-the-shelf products such as smartphones [33, 34] and smartwatches [11, 35, 36], with the idea that commercial devices will not confer social stigma upon the user [9], are more practical for daily wear, and may be used for other tasks to compensate for the initial hardware cost. These studies use the devices' built-in accelerometers and/or gyroscopes to collect data. None of these wearable-based recognition studies have considered the motion of buckling a seatbelt, instead focusing on other health-related applications [7, 27] such as meal tracking [11, 24, 36], monitoring cleanliness (brushing teeth, showering) [9, 31, 35, 37], and exercise encouragement [12, 25, 38]. This work implements smartwatch-based recognition to detect the action of putting on a seatbelt, an activity which has been neglected in the literature.

3. DATA COLLECTION

This section focuses on the selection of activities to be observed, the system for data collection, and the two user studies that were conducted over the course of this research.

3.1 Activity Selection

To best recognize when a seatbelt is being fastened, the activities included in the user studies were carefully chosen to involve motions similar to those of the desired action. When buckling a seatbelt, the motion consists generally of an arm-raising motion (to reach up and grab the seatbelt) immediately followed by an arm-lowering motion (to pull the buckle down and secure it). With this in mind, we selected a number of control motions:

1. Reaching up to remove something from a shirt pocket.
2. Putting a phone in a pants pocket after sending a text.
3. Putting a phone in a pants pocket after ending a phone call.
4. Putting on a backpack.
5. Taking off glasses/sunglasses.
6. Putting on a jacket.
7. Reaching up and touching one's face or adjusting hair.

These activities were chosen to ensure that we could distinguish seatbelt motion from similarly-patterned activities. The full set of activities included in the user study may be broadly thought of as *buckling* and *non-buckling* activities. Note that the movement to put a phone in one's pocket after texting has a different starting position than the movement to

put a phone in one's pocket after calling. Therefore, both were included in the initial data collection.

3.2 System Implementation

Because the arm is the primary limb involved in fastening a seatbelt, during the study the sensor is attached to the user's wrist. Using accelerometer data from the wrist is well established in literature, especially for activities concentrated in the upper body and arms [11, 29, 32, 35]. To collect this user movement data, we used a Pebble smartwatch, which possesses a 4G 3-axis accelerometer [39]. These watches were chosen above other smartwatches containing accelerometers because they are simple to use, easy to program, relatively inexpensive, and already available to the Sketch Recognition Lab. The watches were paired via Bluetooth to Android phones running a data collection application. The Pebble and Android applications used for data collection were identical to those used by Cherian et al. [37]. When the Android application is activated, the watch transmits packets of accelerometer data to the phone via Bluetooth and the phone stores the data in a database for future use. Accelerometer data was sampled for all three axes at a rate of 25 Hz. Each entry in the database is marked with a timestamp, as well as a custom label noting the activity that the user was performing at the time of collection.

3.3 User Studies

Our data collection occurred in two phases. Phase I consisted of an initial user study, which provided the data used to develop the first iteration of our algorithm. Phase II consisted of iterative improvements to the recognition algorithm through real-time naturalistic data collection.

3.3.1 Phase I: Controlled Testing

As mentioned in Section 3.1, two forms of data were collected: buckling gestures and non-buckling gestures. During the first user study, data was collected in a discrete fashion. That is, an experimenter started the data collection, the user performed the specified action, and the experimenter halted data collection with minimal time between steps. This provided us with individual instances of gestures, separated by periods of time when data was not being collected.

First, users performed the action of fastening their seatbelts. Each user completed 10 trials of seatbelt buckling. Here, data collection was complicated by the fact that people may put on seatbelts in any of three different ways:

1. Reaching up with their left arm, then bringing the buckle all the way down to completion.

2. Reaching up with their right arm, then bringing the buckle all the way down to completion.

3. Reaching up with their left arm, transferring the buckle from their left hand to their right hand, then fastening the buckle with their right arm.

Because of this, each user was encouraged to perform the trial once without wearing the Pebble watch, thereby revealing which method they used to fasten their seatbelt. The experimenters then put the watch on whichever arm the subject used to perform the initial reach toward the buckle. The user continued to wear the watch on that same arm for the remainder of the data collection process. After the buckling action, each user completed 5 trials each for all seven of the supplemental actions.

Data was collected from twelve research participants of mixed ages, ethnicities, and genders. Two participants used buckling method 1 (left hand only), seven participants

used buckling method 2 (right hand only), and three participants used buckling method 3 (transferring hands). Therefore, five participants (left hand and transferring hands) wore the smartwatch on their left hand, while the remaining seven participants (right hand) wore the smartwatch on their right hand. In total, this resulted in 120 instances of buckling and 60 instances of each supplemental activity, for a total of 540 instances all together.

3.3.2 Phase II: Naturalistic Testing

Data collection via discrete trials allowed construction of the initial algorithm. To investigate the algorithm's effectiveness in real time, a second, more naturalistic user study was conducted where each participant buckled their seatbelt at some point during a specified time period. The algorithm was expected to recognize the buckling action when it occurred.

The Android and Pebble applications were modified to run the previously developed detection algorithm in real time. Five research participants participated in this study. Data was collected from each participant over a period of 25 minutes, wherein the participants buckled their seatbelt at least once. During the remainder of the period, participants walked down a hallway, ascended and descended stairs, and repeated at least one action from the previous study (Section 3.1). Each session was observed by a researcher.

4. INITIAL RECOGNITION AND FEATURES

After collecting accelerometer data from our user study, we began the process of preparing the data for analysis.

4.1 Recognition Methodology

The general process of activity recognition begins by collecting raw data from the chosen sensors. This data cannot be successfully classified in its raw form because it is typically full of environment noise. Instead, representative features (characteristics of the data) must be extracted from the raw signal. The chosen classification algorithm may then classify the data as representing a particular activity based on those extracted features [8]. More detail will provided for each of these steps in subsequent sections.

4.1.1 Early Features

To prepare the accelerometer data for analysis, a set of features was extracted from the data over a specified window of time. This window might, for example, be one second long. In this case the feature set would be generated by somehow aggregating all of the sensor packets from each second. This is standard procedure for activity recognition [8, 10]. The chosen features were selected due to their simplicity and their precedent for use in literature [8]. We wished to focus only on simple features that require very few computational resources to calculate in on-the-fly, since these would be most useful for our goal of real-time recognition. The results provided by analyzing these basic features acted as a baseline against which we could judge the effectiveness of our improvements.

Twelve initial features were extracted. The set of initial features consisted of:

- Average `X, Y, Z`
- Minimum `X, Y, Z`

- Maximum `X, Y, Z`
- Correlation between axes: `X×Y, X×Z, Y×Z`

Note: the "correlation" measures were approximated by pairwise multiplying the axes together (resulting in a product axis), and then calculating the average value of that axis. Equation 4.1 provides an example of how the XY correlation value is calculated.

$$XY = \frac{\sum_{n=1}^{N} x_n * y_n}{N} \quad \text{where } N \text{ is the size of the data set} \tag{4.1}$$

4.1.2 Data Preprocessing

To extract features from our raw data set, we divided the data points into windows of 0.5 seconds and then generated the features from the data within each window. This window length was chosen in an attempt to recognize individual components of the buckling activity: raise arm, grab buckle, lower arm, fasten buckle. Therefore, the accelerometer data was segmented into 0.5-second long time segments and the twelve features discussed in Section 4.1.1 were then extracted from the data within each window.

A simple Python script was written to traverse the raw accelerometer data (where each row was a Pebble watch packet) and generate an output CSV (Comma-Separated Values) file. Starting with the oldest packet in the data set, the script created a new window and added to that window every subsequent data point that fell within the 0.5 second range. Once the script reached a data point that was outside of the 0.5 second range, the features were calculated from the collected data points and appended to a second file. Next, a new window was created beginning with the most recently seen data point, and this process repeated until the end of the data set was reached. When the script was complete, the original data was unmodified and the second file contained features for all windows in the

Time Epoch (ms)	X (mG)	Y (mG)	Z (mG)	Activity
1479433861850	173	93	-1064	Buckling
1479433861900	33	163	-734	Buckling
1479433861940	-125	221	-393	Buckling
1479433861980	-221	204	-307	Buckling
1479433862020	-385	326	-91	Buckling
1479433862060	-809	245	419	Buckling
1479433862100	-554	-168	464	Buckling
1479433862140	-707	3	512	Buckling
1479433862180	-715	25	618	Buckling
1479433862220	-736	484	591	Buckling
1479433862260	-327	844	2522	Buckling
1479433862300	-673	-413	996	Buckling
1479433862340	-684	-1	838	Buckling

Table 4.1: A sample of Pebble data packets that all fall within the same 0.5-second window.

data. An example of this transformation may be observed in Tables 4.1 and 4.2. A series of raw Pebble data packets within the same 0.5-second window are shown in Table 4.1. The resulting feature entry is in Table 4.2.

Additionally, it is worth noting that the raw data points were each labeled with the activity being performed at the time ("Buckling", "Shirt", "Texting", "Calling", "Backpack", "Glasses", "Jacket", "Face", or "Nothing"). Since this research is focused only on recognizing seatbelt-fastening activities as opposed to discriminating between multiple activities, we were only concerned with knowing if the data point was a buckling action or not. Therefore, each completed feature set was re-labeled as either "Buckling" or "Not_Buckling" according to the activity that was being performed.

4.1.3 Classification Algorithms

To evaluate the effectiveness of the initial feature set, seven classification algorithms were selected. Each classifier is commonly used in prior literature [10]. These classifiers were:

Average X	**Average Y**	**Average Z**
-440.769230769	155.846153846	336.230769231
Minimum X	**Minimum Y**	**Minimum Z**
-809.00	-413.00	-1064.00
Maximum X	**Maximum Y**	**Maximum Z**
173.00	844.00	2522.00
X * Y	**X * Z**	**Y * Z**
-68692.1893491	-148200.177515	52400.2721893

Number of Points	**Start Time**	**End Time**	**Activity**
13	1479433861850	1479433862340	Buckling

Table 4.2: The features generated from the packets in Table 4.1.

1. *k-Nearest Neighbors*: kNN classifies data points based on the most similar points in the data set. Similarity is defined based on a Euclidean distance function which is used to compare each data point to every other data point within the set [10, 23].

2. *C4.5 Decision Tree*: C4.5 constructs a hierarchical tree where each edge represents a possible value (or range of values) for the given feature [10, 34]. This classifier is particularly good for real-time or mobile applications where computational resources are limited, because the generated decision tree is easy to implement and quick to execute.

3. *MultilayerPerceptron*: A Multilayer Perceptron is a type of Neural Network. This classifier relies on back-propagation to classify data points [34, 40]. Neural Networks are unsuited for real-time or limited-resource applications because they require large amounts of time and computational energy.

4. *Naive Bayes*: Naive Bayes calculates probabilities for each class using conditional probabilities calculated from features of the data point. This classifier assumes that every feature is independent, which may lead to errors when applied to data with correlations between features [10, 31].

5. *Random Forest*: Random Forest constructs a series of decision trees based a random sample of data points. When classifying a given point, each tree gets one vote. The class with the most votes is assigned to the data point [11, 41].

6. *Support Vector Machine*: SVMs attempt to find linear decision boundaries (aka support vectors) between classes [10, 11].

7. *ZeroR*: ZeroR is a type of straw man classification wherein the most common class is applied to every data point in the set. That is, if the majority of points belong to a given class, ZeroR will predict that every point belongs to that class. [34, 42]. ZeroR provides a baseline against which to judge the performance of other classifiers.

4.2 Results

The effectiveness of the initial feature set was evaluated using the classification algorithms from Section 4.1.3 as implemented in the WEKA data analysis tool [43]. In WEKA, the implementation of kNN is called IBk [44, 45], the implementation of C4.5 is called J48 [46, 47], and the implementation of SVM is called SMO [48–51]. The other algorithm implementations keep their original names. We shall refer to the classifiers by their WEKA implementation nomenclature. Every classifier was run in WEKA with its default settings [43]. The results of these classifiers are shown in Tables 4.3 and 4.4.

As seen in Table 4.3, all of the classifiers reported overall accuracy measures of approximately 80-90%. However, these results must be compared to the frequency that each activity actually occurred in the data. In this case, 79% of the data belonged to the Not_Buckling class. This means that even the ZeroR classifier was correct for 79% of the data points. The large number of Not_Buckling data instances falsely inflates the accuracy metric, since it is calculated across all classes (Equation 4.2). Instead, we look to F-measures to provide a more realistic picture of classification success.

Classifier	Accuracy
IBk	88.4%
J48	86.7%
MultilayerPerceptron	87.8%
Naive Bayes	84.7%
Random Forest	90.1%
SMO	85.0%
ZeroR	79.0%

Table 4.3: Initial Classifier Accuracy using the early feature set on controlled data.

Classifier	Activity	F-Measure
IBk	Buckling	0.727
	Not_Buckling	0.926
J48	Buckling	0.702
	Not_Buckling	0.914
MultilayerPerceptron	Buckling	0.716
	Not_Buckling	0.922
Naive Bayes	Buckling	0.685
	Not_Buckling	0.899
Random Forest	Buckling	0.760
	Not_Buckling	0.938
SMO	Buckling	0.604
	Not_Buckling	0.907
ZeroR	Buckling	0.000
	Not_Buckling	0.883

Table 4.4: Initial Classifier F-Measures using the early feature set on controlled data.

F-measure is based on the precision and recall values of the classifier. Precision is the ratio of correctly classified positives (true positives) to the total number of classified positives (true positives and false positives). The formula for Precision is shown in Equation 4.3. Recall is the ratio of true positives to total positives (true positives and false negatives); it is calculated according to Equation 4.4. F-measure, then, combines both Precision and Recall as in Equation 4.5 [10].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{4.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.4}$$

$$F\text{-}measure = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4.5}$$

The F-measures from each of the classifiers (Table 4.4) reveal mediocre recognition performance. Closer investigation of the confusion matrices generated by the classifiers reveals a significant number of both false negatives and false positives (Tables 4.5 - 4.10). It is apparent that applying only a simplistic set of traditional features is not effective to recognize seatbelt activity. Therefore, we must generate novel features specific to the motion of buckling a seatbelt in order to most effectively recognize that motion.

a	b	<– classified as
535	191	a = Buckling
211	2522	b = Not_Buckling

Table 4.5: IBk Confusion Matrix using early feature set on controlled data.

a	b	<– classified as
541	185	a = Buckling
275	2458	b = Not_Buckling

Table 4.6: J48 Confusion Matrix using early feature set on controlled data.

a	b	<– classified as
533	193	a = Buckling
299	2504	b = Not_Buckling

Table 4.7: MultilayerPerceptron Confusion Matrix using early feature set on controlled data.

a	b	<– classified as
575	151	a = Buckling
379	2354	b = Not_Buckling

Table 4.8: Naive Bayes Confusion Matrix using early feature set on controlled data.

a	b	<– classified as
541	185	a = Buckling
156	2577	b = Not_Buckling

Table 4.9: Random Forest Confusion Matrix using early feature set on controlled data.

a	b	<– classified as
396	330	a = Buckling
189	2544	b = Not_Buckling

Table 4.10: SMO Confusion Matrix using early feature set on controlled data.

5. IMPROVING RECOGNITION WITH ADDITIONAL FEATURES

Previously, none of the features we applied were unique to the seatbelt action itself. In order to improve the gesture recognition algorithm, we plotted each individual seatbelt motion trial and began to look for patterns.

5.1 Data Preprocessing

Upon studying the accelerometer data, it became apparent that our data needed to be additionally preprocessed in order for it to best be useful. Due to the sampling frequency of the sensors, the data was rather noisy even after feature extraction. The Pebble watch accelerometers transmitted data at a rate of 25 Hz, meaning that there was far more granularity than we were interested in. Because of this, we applied a smoothing function to the data before extracting features from it. This step is recommended in literature [8]. The purpose of smoothing was twofold:

1. The smoothing function acts as a filter, filtering out fine movements and other anomalies and leaving only the broad, sweeping motions that we were interested in, *i.e.* gross arm motion.

2. The smoothing function makes it easier to analyze the overall patterns in the data, without having to sift through noise.

There is a tradeoff associated with filtering values too crudely. Excessive smoothing runs the risk of masking important information about the data. Therefore, it is important to carefully choose the number of data points which are smoothed together.

To achieve the desired smoothing effect, we used a rolling average function. Given a window size w, the rolling average function is applied such that each data point becomes an average of its w surrounding points. Figure 5.1 illustrates how a rolling average

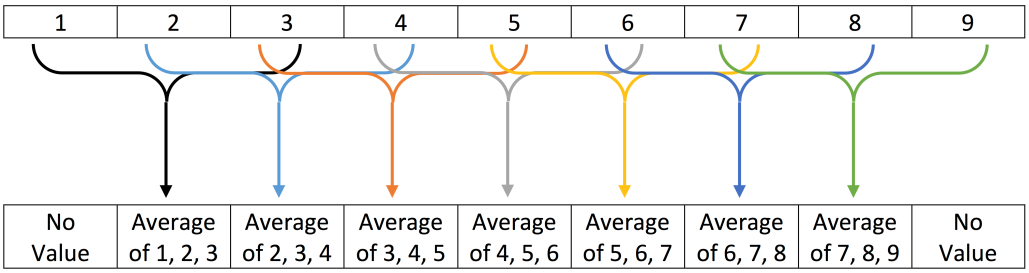

Figure 5.1: Illustration of a rolling average function with window size $w = 3$.

function works, with each "smoothed" point being calculated using Equation 5.1. For this study, the data was smoothed with $w = 10$. This value for w was chosen because it reduced data noise without hiding important information. Originally, the 25 Hz sampling frequency yielded one packet every 0.04 seconds. After application of the filter function, each point was an average of every packet within a 0.4 second range. This mimicked the initial procedure of splitting the data into 0.5-second windows before calculating features. As with that early effort, the goal was to capture information about only the individual components of the buckling motion, as opposed to the exact quiver of a hand. Figure 5.2 illustrates the difference between raw data and the corresponding smoothed data.

$$x'_n = \frac{1}{w} * \sum_{i=n-\lceil \frac{w-1}{2} \rceil}^{n+\lfloor \frac{w-1}{2} \rfloor} x_i \quad \text{where } n \text{ is the index of the data point being smoothed} \quad (5.1)$$

5.1.1 Data Analysis

Upon investigating the plotted data trials, it was apparent that the motion of buckling has two distinct sections, or "halves". The first section corresponds to when the user raises their arm to reach for their seatbelt. The second section corresponds to when the user

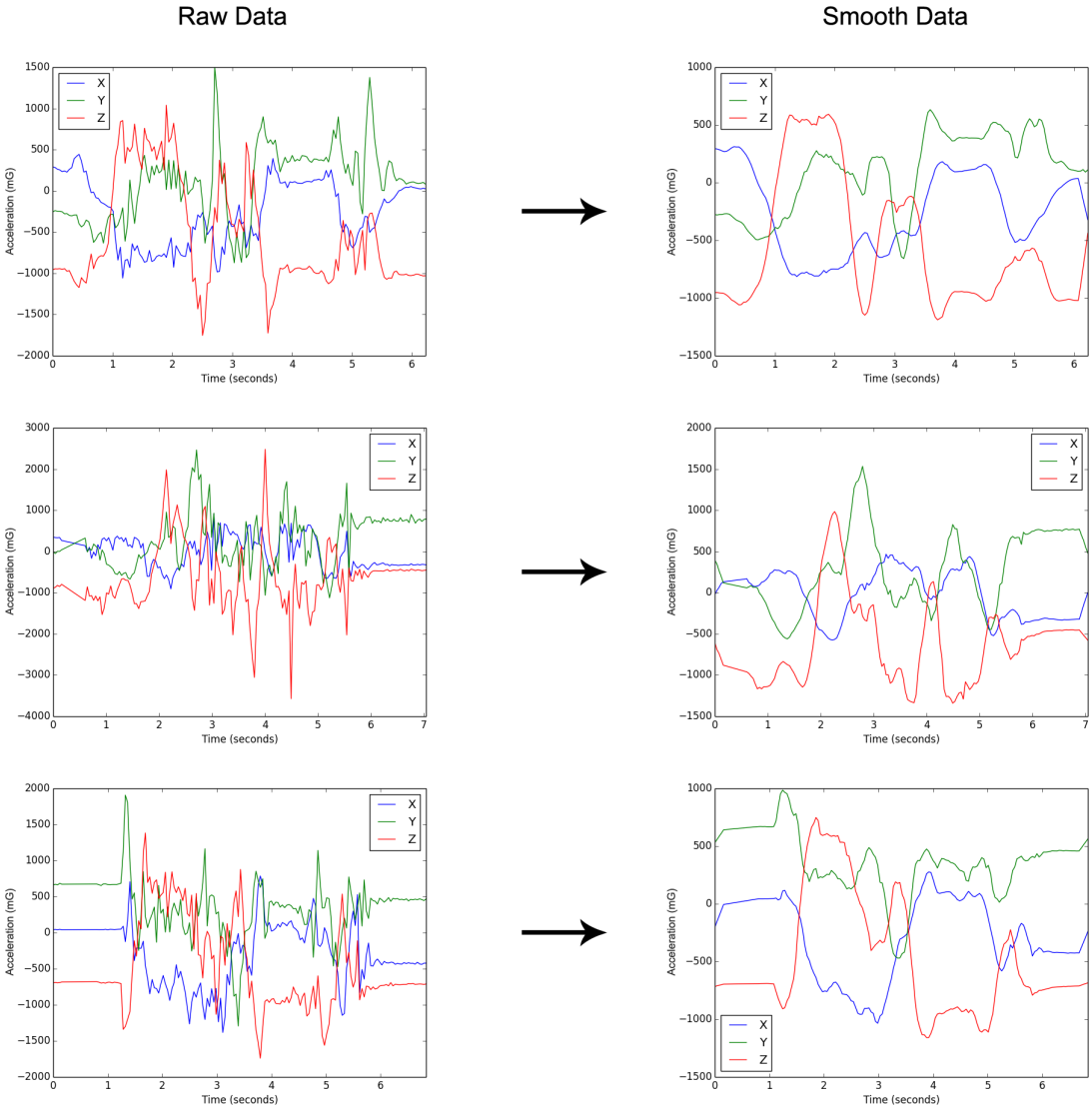

Figure 5.2: Data plots from three different seatbelt-buckling activities, each before and after the smoothing function was applied.

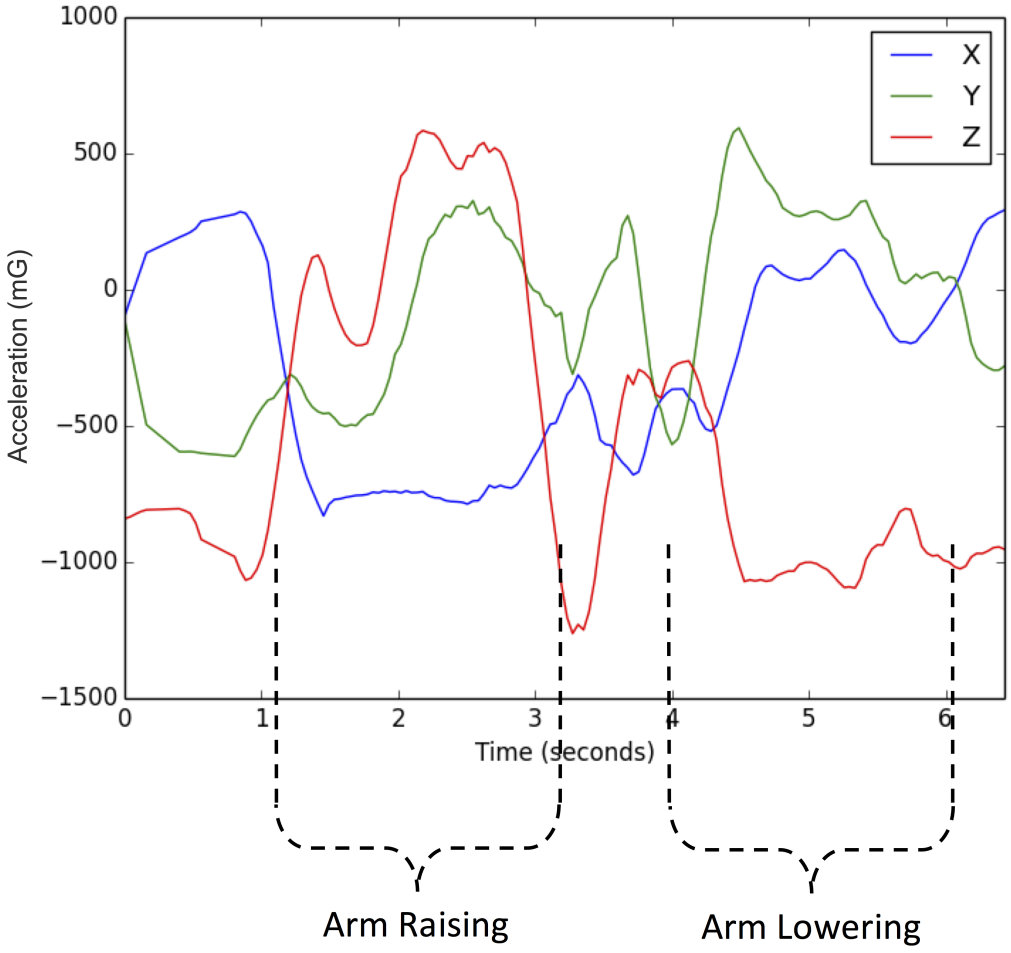

Figure 5.3: A plot of smoothed accelerometer data for one buckling instance with each component marked. The first "half" of the data corresponds to the period where the user lifts their arm to grab the buckle. The second "half" corresponds to the period where the user lowers their arm to fasten the buckle.

lowers their arm to fasten the buckle. This two-section structure holds no matter which arm is used to buckle, because every user wore the watch on the arm that performed the initial reach. With this structure in mind, we modified our feature generation procedure.

Because a seatbelt motion displays less periodicity than other activities such as running or brushing teeth, a 0.5-second window was recognized as too short to capture any useful component of a seatbelt motion. Therefore, the window size was expanded to 5 seconds, the observed average total duration of the activity. Additionally, to mitigate the chances of an activity being "caught" between two windows, we implemented a sliding window. Because of the precise nature of the gesture and its two distinct components, we used a 3-second window overlap, illustrated in Figure 5.4. Every two seconds, a new window is calculated with the result that every window overlaps by three seconds.

Even though our feature window size now encompassed the entire gesture, we still wanted to recognize the distinct characteristics of the first half of the motion as compared to the second half of the motion. To capture the nuances of these different components, we considered each feature independently over the first half of the window and again over the second half of the window. Thus, we have three columns for each feature, with each column representing a different time period: one entry that is independent for the entire window, one entry calculated from the first half of the window, and one entry calculated from the last half of the window.

5.1.2 Additional Features

Plotting the motions revealed correlations between the three data streams that were not well captured by the original feature set. As shown in Figure 5.3, during the first half of the motion, Y and Z appear to be directly related, while X is inversely related. Additionally, the Y and Z axes appear to be very close numerically. During the second half of the motion, Y and X appear to be directly related, while Z is inversely related.

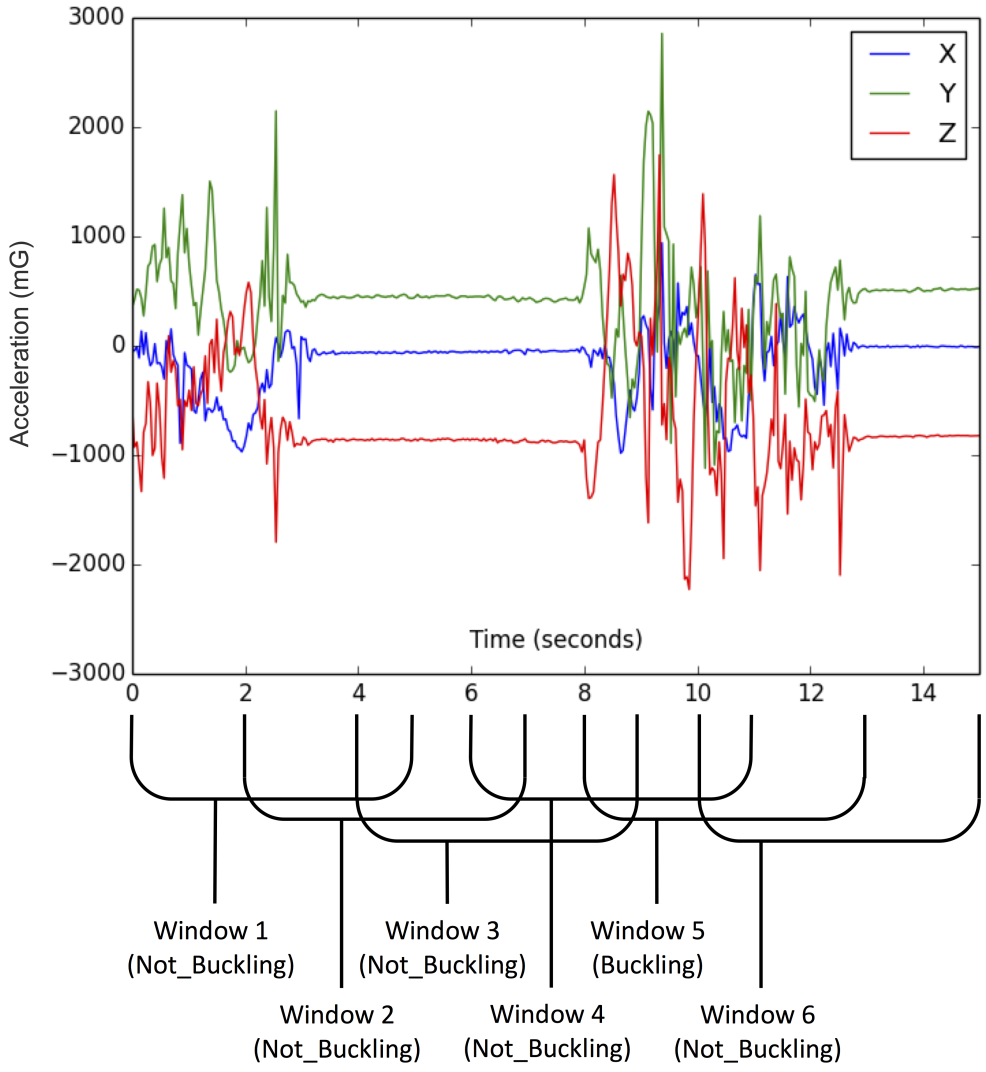

Figure 5.4: An illustration of the sliding window used to calculate features from the raw accelerometer data.

Appropriately, the Y and X values are very close. Based on this, we constructed features from the differences between axes: X - Y, X - Z, and Y - Z. We also added the absolute value of these differences, yielding the true differences between the axes. Finally, a feature was created to reflect the difference between the X-Y and the Y-Z difference. Because there is a high X-Y difference and a low Y-Z difference in the first half, followed by a low X-Y difference and a high Y-Z difference in the second half, the difference between the two difference measures should theoretically stay more constant throughout the whole motion.

The following eight features were used to augment our existing feature set. Sample formulas are included for each group of features. In each case, a new axis is created by performing the desired operation on each pair of points within the window (for instance, by pairwise subtracting Y values from X values). Then, these results are averaged together across all n points in the window to create the final value of the feature for that window.

- Difference between axes: `X-Y, X-Z, Y-Z`

$$\texttt{X-Y} = \frac{\sum_{i=1}^{n} X_i - Y_i}{n} \quad \text{Difference between X and Y axes.} \tag{5.2}$$

- Absolute difference between axes: `|X-Y|, |X-Z|, |Y-Z|`

$$\texttt{|X-Y|} = \frac{\sum_{i=1}^{n} |X_i - Y_i|}{n} \quad \text{Absolute difference between X and Y axes.} \tag{5.3}$$

- Difference between XY-difference and YZ-difference: `|X-Y|-|Y-Z|`

$$\texttt{|X-Y|-|Y-Z|} = \frac{\sum_{i=1}^{n} |X_i - Y_i| - |Y_i - Z_i|}{n} \tag{5.4}$$

Classifier	Accuracy
IBk	99.8%
J48	97.9%
MultilayerPerceptron	100%
Naive Bayes	99.4%
Random Forest	99.4%
SMO	99.8%
ZeroR	89.2%

Table 5.1: Improved Classifier Accuracy using the expanded feature set on controlled data.

- Absolute difference between XY-difference and YZ-difference: `||X-Y|-|Y-Z||`

$$\texttt{||X-Y|-|Y-Z||} = \frac{\sum_{i=1}^{n} ||X_i - Y_i| - |Y_i - Z_i||}{n} \tag{5.5}$$

Because each feature was calculated three times for different components of the window, there was a total of 60 features for each window. These features were generated on the original data set, so no new data was collected.

5.2 Results

The new feature set was evaluated against the same set of classifiers from WEKA used in the initial study: IBk, J48, MultilayerPerceptron, Naive Bayes, Random Forest, SMO, and ZeroR. The results are shown in Tables 5.1 and 5.2. The confusion matrices for each classifier are in Tables 5.3 - 5.8.

The new feature set greatly improved classification success on the data from the first user study. F-measures for both activities were above 0.900 for every classifier tested (excluding ZeroR). The IBk classifier had no false negatives. Several classifiers (Naive Bayes, Random Forest, and SMO) had no false positives. MultilayerPerceptron returned no misclassifications at all. These results indicate that the new feature set is much more representative of the unique characteristics of the buckling motion.

Classifier	Activity	F-Measure
IBk	Buckling	0.990
	Not_Buckling	0.999
J48	Buckling	0.902
	Not_Buckling	0.988
MultilayerPerceptron	Buckling	1.000
	Not_Buckling	1.000
Naive Bayes	Buckling	0.970
	Not_Buckling	0.996
Random Forest	Buckling	0.970
	Not_Buckling	0.996
SMO	Buckling	0.990
	Not_Buckling	0.999
ZeroR	Buckling	0.000
	Not_Buckling	0.943

Table 5.2: Improved Classifier F-Measures using the expanded feature set on controlled data.

a	b	<– classified as
51	0	a = Buckling
1	420	b = Not_Buckling

Table 5.3: Improved IBk Confusion Matrix using expanded feature set on controlled data.

a	b	<– classified as
46	5	a = Buckling
5	416	b = Not_Buckling

Table 5.4: Improved J48 Confusion Matrix using expanded feature set on controlled data.

a	b	<– classified as
51	0	a = Buckling
0	421	b = Not_Buckling

Table 5.5: Improved MultilayerPerceptron Confusion Matrix using expanded feature set on controlled data.

a	b	<– classified as
48	3	a = Buckling
0	421	b = Not_Buckling

Table 5.6: Improved Naive Bayes Confusion Matrix using expanded feature set on controlled data.

5.3 Real-Time Recognition

A second user study was conducted to test the validity of this approach and feature set for real-time recognition.

5.3.1 Algorithm Implementation

The Python script which extracted features from the data was translated into Java and integrated with the Android application so that features could be generated on the fly. The J48 classifier outputs the decision tree it creates, so we integrated this tree with the Android application as a method for real-time recognition. As a result of these two additions, the Android application not only collected data, but also generated features in real time and checked those features against the J48 decision tree in an attempt to recognize a buckling action during the current time window. A diagram of the updated application structure is included in Figure 5.5. The application alerts the user when the classification algorithm detects a buckling activity. It also contains fields where the user may report true positives, false positives, and false negatives. The resulting input may be used to continuously improve classification. During the first few rounds of naturalistic testing, we used the real-time results to manually update our classification and improve our results. However, this improvement process should be automated within the application. This improvement is left for future work. Screenshots of the application interface are shown in Figures 5.6 and 5.7.

5.4 Real-Time Results

Although only the J48 decision tree was implemented to run in real time during the second user study, other classifiers were run after the fact on the naturalistic data obtained during that study. The results of these classifiers are in Tables 5.9 and 5.10. The confusion matrices for each classifier are in Tables 5.11 - 5.16.

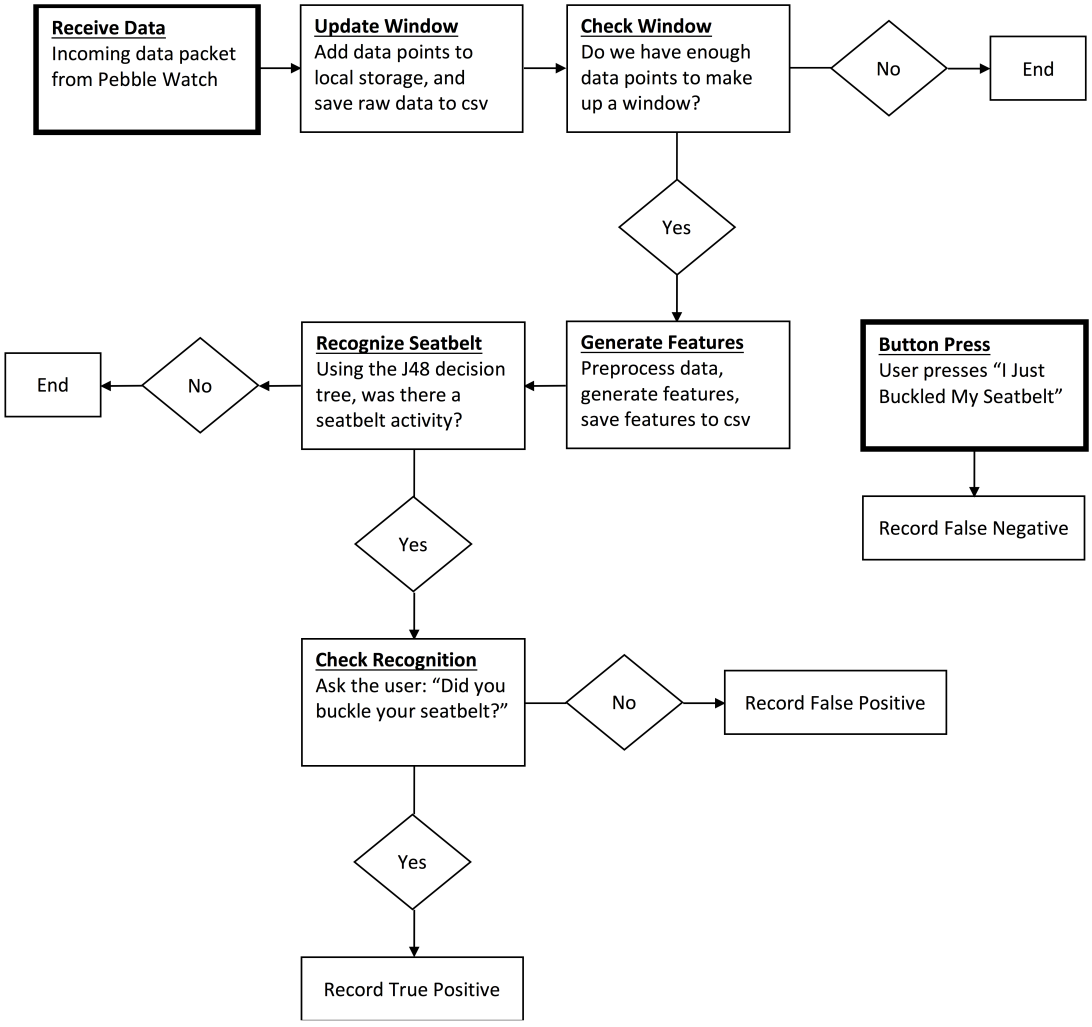

Figure 5.5: Structure of the Android application used for real-time recognition. Each incoming data packet triggers the chain of actions from the beginning, starting with the "Receive Data" box.

a	b	<– classified as
48	3	a = Buckling
0	421	b = Not_Buckling

Table 5.7: Improved Random Forest Confusion Matrix using expanded feature set on controlled data.

a	b	<– classified as
51	1	a = Buckling
0	421	b = Not_Buckling

Table 5.8: Improved SMO Confusion Matrix using expanded feature set on controlled data.

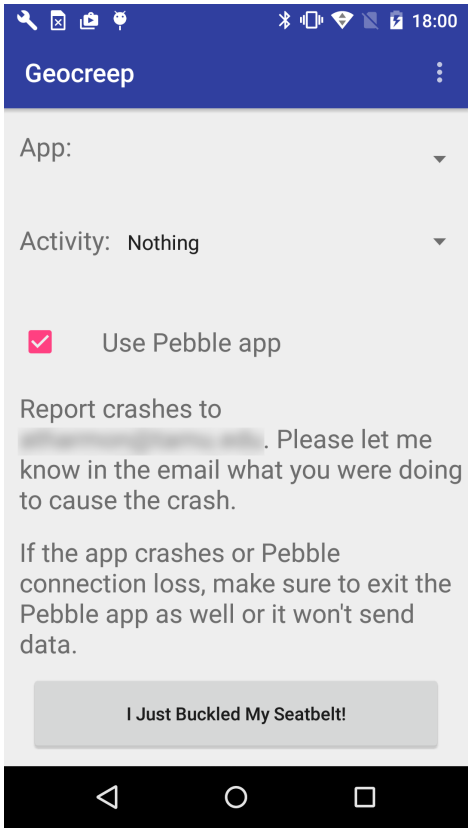

Figure 5.6: Interface of the Android application used for real-time recognition testing. The button at the bottom of the screen allows the user to indicate a false negative (that is, they buckled their seatbelt but the application did not recognize the motion as buckling).

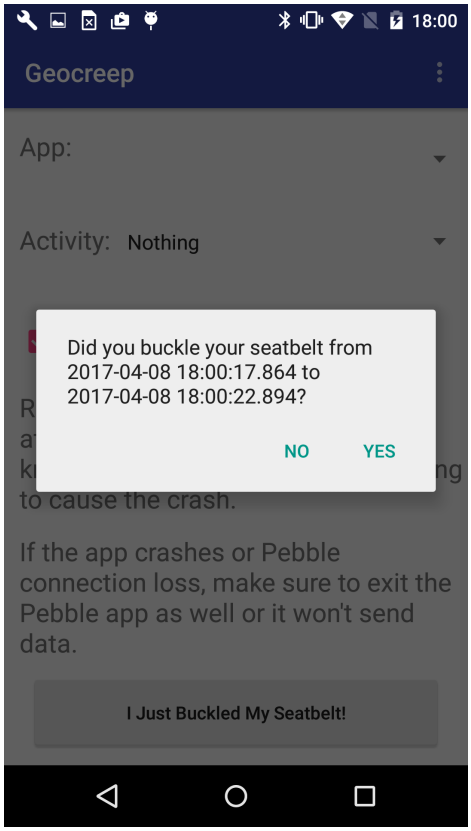

Figure 5.7: During real-time recognition testing, a popup is shown when the classification algorithm detects a buckling motion. The user has the option to mark the recognition as a true positive (they did buckle) or a false positive (they did not buckle).

The F-measures show a significant drop between the controlled data and the naturalistic data. However, an F-measure of 0.825 is still achieved with the MultilayerPerceptron classifier. J48, the classifier which was run in real time on the Android application, yielded an F-measure of 0.734. More investigation is needed to improve the effectiveness of the real-time system, but we know from the lab data results that such effectiveness is possible. Possible improvements will be discussed in Section 6.1.

Classifier	Accuracy
IBk	97.0%
J48	96.5%
MultilayerPerceptron	97.6%
Naive Bayes	90.4%
Random Forest	97.5%
SMO	97.3%
ZeroR	93.6%

Table 5.9: Real-Time Classifier Accuracy using expanded feature set on naturalistic data.

Classifier	Activity	F-Measure
IBk	Buckling	0.775
	Not_Buckling	0.984
J48	Buckling	0.734
	Not_Buckling	0.981
MultilayerPerceptron	Buckling	0.825
	Not_Buckling	0.987
Naive Bayes	Buckling	0.550
	Not_Buckling	0.946
Random Forest	Buckling	0.772
	Not_Buckling	0.987
SMO	Buckling	0.769
	Not_Buckling	0.986
ZeroR	Buckling	0.000
	Not_Buckling	0.967

Table 5.10: Real-Time Classifier F-Measures using expanded feature set on naturalistic data.

a	b	<– classified as
74	18	a = Buckling
25	1325	b = Not_Buckling

Table 5.11: Real-Time IBk Confusion Matrix using expanded feature set on naturalistic data.

a	b	<– classified as
69	23	a = Buckling
27	1323	b = Not_Buckling

Table 5.12: Real-Time J48 Confusion Matrix using expanded feature set on naturalistic data.

a	b	<– classified as
80	12	a = Buckling
22	1328	b = Not_Buckling

Table 5.13: Real-Time MultilayerPerceptron Confusion Matrix using expanded feature set on naturalistic data.

a	b	<– classified as
85	7	a = Buckling
132	1218	b = Not_Buckling

Table 5.14: Real-Time Naive Bayes Confusion Matrix using expanded feature set on naturalistic data.

a	b	<– classified as
61	31	a = Buckling
5	1345	b = Not_Buckling

Table 5.15: Real-Time Random Forest Confusion Matrix using expanded feature set on naturalistic data.

a	b	<– classified as
65	27	a = Buckling
12	1338	b = Not_Buckling

Table 5.16: Real-Time SMO Confusion Matrix using expanded feature set on naturalistic data.

6. FUTURE WORK

6.1 Further Recognition Improvements

Although the recognition approach presented in this paper was very successful at classifying historical data, it was less successful at real-time classification. One possible explanation for this is that the data that was gathered during the initial user study is not representative of naturalistic data. For example, users during the real-time study often experienced false positives (the Android application told them they were buckling their seatbelt) when they were sitting still or walking. No data for these normal activities was collected during the initial user study, so the classification algorithm had not been trained to discriminate between these actions and buckling actions. Once we began to conduct naturalistic trials, we supplemented our original test data with the new, real-time data. We manually sent this data back through the WEKA J48 classifier in order to generate a significantly more accurate decision tree. In the future, this process of improvement should be made automated. This idea is elaborated below.

One method to overcome shortcomings of incomplete data sets is to implement adaptive recognition. Adaptive recognition does not require prior knowledge about the user or the activities being performed [23]. The classifier is adjusted to assimilate new data as it is provided. This technique should be investigated in the context of real-time recognition through the Android application. The input which is accepted from the user (denotation of true positives, false positives, and false negatives) may be used to reconstruct the classifier automatically. Additionally, such an adaptive classification implementation could allow each Android application to build a unique model for each user, leading to more personalized buckling detection. Individually constructed models have shown success in other research [33]. Individual classification models may handle classification errors which stem

from underlying differences in how users perform the seatbelt-buckling action, such as the dominant hand used. Future studies should consider adaptively adjusting classifiers in real time, in addition to constructing unique classifiers for each user.

6.2 Additional Detection

The goal of this research is to provide a foundation to implement a warning system which detects whether a user who is in a car has fastened their seatbelt. If they have not done so, it should provide an appropriate intervention to the user. Components of this process are:

1. Detect whether the user has fastened their seatbelt.
2. Detect whether the user is driving or riding in a car.
3. If statement 1 is not true but statement 2 is true, alert the user.

This work offers a method for statement 1. Further studies should seek to integrate our method with a method for detecting that the user is driving [52].

6.3 Feedback & Intervention

To promote behavioral modification, the detection system must alert the user that they are engaging in undesirable behavior. It must do so in a way which is helpful but not annoying, as this might discourage user participation. Suitable modes of feedback might include haptic vibrations [53–56], auditory stimuli, or visual cues [13]. Additional studies should determine the optimal mode of feedback and test the effectiveness of the integrated system. Together with an intervention, the process of detection can be used as the basis of many safety applications which track and improve behavior.

7. CONCLUSION

This research investigated the use of activity recognition to detect the action of buckling a seatbelt, using accelerometer data collected from a Pebble smartwatch. This work is necessary because although most cars contain seatbelt-buckling detection systems [4, 5], these systems are frequently ignored, resulting in many deaths related to car crashes [2]. Activity recognition is a popular field with many applications to healthcare [12, 13], but no prior work has sought to apply wearable activity recognition techniques to detect the action of buckling a seatbelt.

We began by applying simple traditional features found in literature [8] to accelerometer data collected through an initial user study. These features were moderately successful at discriminating between data that indicated buckling and data that did not. We were able to greatly improve recognition performance by expanding this feature set to reflect characteristics of the data that are unique to the buckling motion. The final F-measures for controlled recognition of buckling reached 1.000 using MultilayerPerceptron. Finally, we tested the effectiveness of the new feature set in a naturalistic user study, resulting in an F-measure of 0.825 with MultilayerPerceptron.

These results indicate the high effectiveness of our feature set and of activity recognition in general for recognizing the action of buckling a seatbelt. This recognition may be incorporated into future systems that aim to promote safe behavior. Through these future systems, public health and safety can be improved.

REFERENCES

[1] N. H. T. S. Administration, "Traffic safety facts, 2014 data: occupant protection. Washington, DC: US department of transportation, national highway traffic safety administration; 2016," 2016.

[2] Centers for Disease Control and Prevention (CDC), "Seat belts: Get the facts." http://www.cdc.gov/motorvehiclesafety/seatbelts/facts.html, July 5, 2016. Accessed: 2017-04-09.

[3] T. W. Strine, L. Beck, J. Bolen, C. Okoro, and C. Li, "Potential moderating role of seat belt law on the relationship between seat belt use and adverse health behavior," *American journal of health behavior*, vol. 36, no. 1, pp. 44–55, 2012.

[4] H. Yoshihiro, "Patent us3689881 - device in an automobile for detecting a pull-out motion of a seat belt." https://www.google.com/patents/US3689881, 1971. Accessed: 2017-04-09.

[5] K. Aoki, T. Yasuda, S. Masanori, K. Osamu, and A. Kaisha, "Patent us4885566 - apparatus for detecting the wearing of a seat belt assembly." https://www.google.com/patents/US4885566, 1987. Accessed: 2017-04-09.

[6] R. Yano, "Patent us6498562 - seat belt buckle engagement detector and seat belt system." https://www.google.com/patents/US6498562, 2000. Accessed: 2017-04-09.

[7] L. Chen, J. Hoey, C. D. Nugent, D. J. Cook, and Z. Yu, "Sensor-based activity recognition," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, pp. 790–808, Nov 2012.

[8] D. Figo, P. C. Diniz, D. R. Ferreira, and J. M. Cardoso, "Preprocessing techniques for context recognition from accelerometer data," *Personal and Ubiquitous Computing*, vol. 14, no. 7, pp. 645–662, 2010.

[9] J. Lester, T. Choudhury, and G. Borriello, "A practical approach to recognizing physical activities," in *International Conference on Pervasive Computing*, pp. 1–16, Springer, 2006.

[10] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors.," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 3, pp. 1192–1209, 2013.

[11] E. Thomaz, I. Essa, and G. D. Abowd, "A practical approach for recognizing eating moments with wrist-mounted inertial sensing," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 1029–1040, ACM, 2015.

[12] J. Bartley, J. Forsyth, P. Pendse, D. Xin, G. Brown, P. Hagseth, A. Agrawal, D. Goldberg, and T. Hammond, *World of workout: a contextual mobile RPG to encourage long term fitness*. 2013.

[13] V. Rajanna, F. Alamudun, D. Goldberg, and T. Hammond, "Let me relax: Toward automated sedentary state recognition and ubiquitous mental wellness solutions," in *Proceedings of the 5th EAI International Conference on Wireless Mobile Communication and Healthcare*, pp. 28–33, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2015.

[14] J. M. Peschel, B. Paulson, and T. Hammond, "A surfaceless pen-based interface," in *Proceedings of the Seventh ACM Conference on Creativity and Cognition*, C&C '09, (New York, NY, USA), pp. 433–434, ACM, 2009.

[15] I. J. Amin, A. J. Taylor, and R. M. Parkin, "Driver tracking and posture detection using low-resolution infrared sensing," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 221, no. 9, pp. 1079–1088, 2007.

[16] B. Paulson and T. Hammond, "Office activity recognition using hand posture cues," in *Proceedings of the 22Nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction - Volume 2*, BCS-HCI '08, (Swinton, UK, UK), pp. 75–78, British Computer Society, 2008.

[17] B. Paulson, D. Cummings, and T. Hammond, "Object interaction detection using hand posture cues in an office setting," *Int. J. Hum.-Comput. Stud.*, vol. 69, pp. 19–29, Jan. 2011.

[18] P. Taele and T. Hammond, "Invisishapes: A recognition system for sketched 3d primitives in continuous interaction spaces," in *Proceedings of the 2015 International Symposium on Smart Graphics, Chengdu, China*, SG, p. 12, 2015.

[19] J. Miller and T. Hammond, "Wiiolin: A virtual instrument using the wii remote," in *Proceedings of the 2010 Conference on New Interfaces for Musical Expression (NIME)*, (Sydney, Australia), p. 497âĂŞ500, June 15-18 2010.

[20] P. Kaul, V. Rajanna, and T. Hammond, "Exploring users' perceived activities in a sketch-based intelligent tutoring system through eye movement data," in *ACM Symposium on Applied Perception (SAP '16)*, SAP, p. 1, 2016.

[21] F. Alamudun, H.-J. Yoon, T. Hammond, K. Hudson, G. Morin-Ducote, and G. Tourassi, "Shapelet analysis of pupil dilation for modeling visuo-cognitive behavior in screening mammography," in *Proc. SPIE*, vol. 9787 of *SPIE*, pp. 97870M–97870M–13, 2016.

[22] F. Alamudun, H.-J. Yoon, K. B. Hudson, G. Morin-Ducote, T. Hammond, and G. D. Tourassi, "Fractal analysis of visual search activity for mass detection during mammographic screening," *Medical Physics*, 2017.

[23] K. Van Laerhoven and O. Cakmakci, *What shall we teach our pants?*, pp. 77–83. IEEE Press, 2000.

[24] J. Parkka, M. Ermes, P. Korpipaa, J. Mantyjarvi, J. Peltola, and I. Korhonen, "Activity classification using realistic data from wearable sensors," *IEEE Transactions on information technology in biomedicine*, vol. 10, no. 1, pp. 119–128, 2006.

[25] E. M. Tapia, S. S. Intille, W. Haskell, K. Larson, J. Wright, A. King, and R. Friedman, "Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor," in *Wearable Computers, 2007 11th IEEE International Symposium on*, pp. 37–40, IEEE, 2007.

[26] U. Maurer, A. Smailagic, D. P. Siewiorek, and M. Deisher, "Activity recognition and monitoring using multiple sensors on different body positions," in *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on*, pp. 4–pp, IEEE, 2006.

[27] D. Brhlik, C. Young, and T. Otuyelu, "Enhancing blind navigation with the use of wearable sensor technology," undergraduate honors thesis, Texas A&M University, May 2016.

[28] N. Győrbíró, Á. Fábián, and G. Hományi, "An activity recognition system for mobile phones," *Mobile Networks and Applications*, vol. 14, no. 1, pp. 82–91, 2009.

[29] F. Foerster, M. Smeja, and J. Fahrenberg, "Detection of posture and motion by ac-

celerometry: a validation study in ambulatory monitoring," *Computers in Human Behavior*, vol. 15, no. 5, pp. 571–583, 1999.

[30] C. Randell and H. Muller, "Context awareness by analysing accelerometer data," in *Digest of Papers. Fourth International Symposium on Wearable Computers*, pp. 175–176, Oct 2000.

[31] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, "Activity recognition from accelerometer data," in *Aaai*, vol. 5, pp. 1541–1546, 2005.

[32] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *International Conference on Pervasive Computing*, pp. 1–17, Springer, 2004.

[33] T. Brezmes, J.-L. Gorricho, and J. Cotrina, "Activity recognition from accelerometer data on a mobile phone," in *International Work-Conference on Artificial Neural Networks*, pp. 796–799, Springer, 2009.

[34] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SigKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011.

[35] E. Garcia-Ceja, R. F. Brena, J. C. Carrasco-Jimenez, and L. Garrido, "Long-term activity recognition from wristwatch accelerometer data," *Sensors*, vol. 14, no. 12, pp. 22500–22524, 2014.

[36] G. M. Weiss, J. L. Timko, C. M. Gallagher, K. Yoneda, and A. J. Schreiber, "Smartwatch-based activity recognition: A machine learning approach," in *Biomedical and Health Informatics (BHI), 2016 IEEE-EMBS International Conference on*, pp. 426–429, IEEE, 2016.

[37] J. Cherian, V. Rajanna, D. Goldberg, and T. Hammond, "Did you remember to brush? : A noninvasive wearable approach to recognizing brushing teeth for elderly care.," in *11th EAI International Conference on Pervasive Computing Technologies for Healthcare.*, ACM, 2017.

[38] V. Rajanna, R. Lara-Garduno, D. J. Behera, K. Madanagopal, D. Goldberg, and T. Hammond, "Step up life: a context aware health assistant," in *Proceedings of the Third ACM SIGSPATIAL International Workshop on the Use of GIS in Public Health*, pp. 21–30, ACM, 2014.

[39] Pebble, "Pebble time steel." `https://www.pebble.com/pebble-time-steel-smartwatch-features`. Accessed: 2017-04-09.

[40] M. Ware, "weka.classifiers.functions class MultilayerPerceptron." `http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/MultilayerPerceptron.html`. Accessed: 2017-04-09.

[41] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[42] E. Frank, "weka.classifiers.rules class ZeroR." `http://weka.sourceforge.net/doc.dev/weka/classifiers/rules/ZeroR.html`.

[43] E. Frank, M. Hall, and I. Witten, *The WEKA Workbench. Online appendix for "Data Mining: Practical Machine Learning Tools and Techniques"*. Morgan Kaufmann, 4 ed., 2016.

[44] S. Inglis, L. Trigg, and E. Frank, "weka.classifiers.lazy class IBk." `http://weka.sourceforge.net/doc.dev/weka/classifiers/lazy/IBk.html`. Accessed: 2017-04-09.

[45] D. Aha and D. Kibler, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37–66, 1991.

[46] E. Frank, "weka.classifiers.trees class J48." `http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html`. Accessed: 2017-04-09.

[47] R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers, 1993.

[48] E. Frank, S. Legg, and S. Inglis, "weka.classifiers.functions class SMO." http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/SMO.html. Accessed: 2017-04-09.

[49] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods - Support Vector Learning* (B. Schoelkopf, C. Burges, and A. Smola, eds.), MIT Press, 1998.

[50] S. Keerthi, S. Shevade, C. Bhattacharyya, and K. Murthy, "Improvements to platt's smo algorithm for svm classifier design," *Neural Computation*, vol. 13, no. 3, pp. 637–649, 2001.

[51] T. Hastie and R. Tibshirani, "Classification by pairwise coupling," in *Advances in Neural Information Processing Systems* (M. I. Jordan, M. J. Kearns, and S. A. Solla, eds.), vol. 10, MIT Press, 1998.

[52] M. Kadous, "Detecting driving with a wearable computing device," May 19 2015. US Patent 9,037,125.

[53] M. Prasad, M. I. Russell, and T. A. Hammond, "A user centric model to design tactile

codes with shapes and waveforms," in *Haptics Symposium (HAPTICS), 2014 IEEE*, pp. 597–602, Feb 2014.

[54] M. Prasad, P. Taele, A. Olubeko, and T. Hammond, "Haptigo: A navigational tap on the shoulder," in *Haptics Symposium (HAPTICS), 2014 IEEE*, pp. 339–345, Feb 2014.

[55] M. Prasad, M. Russell, and T. A. Hammond, "Designing vibrotactile codes to communicate verb phrases," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 11, pp. 11:1–11:21, Oct. 2014.

[56] M. Prasad, P. Taele, D. Goldberg, and T. A. Hammond, "Haptimoto: Turn-by-turn haptic route guidance interface for motorcyclists," in *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, (New York, NY, USA), pp. 3597–3606, ACM, 2014.