HUMAN ACTIVITY MONITORING BY SMART DEVICES

By

Chongguang Bi

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science – Doctor of Philosophy

2020

ABSTRACT

HUMAN ACTIVITY MONITORING BY SMART DEVICES

By

Chongguang Bi

The topic of the Internet-of-Things (IoT) has been discussed and studied extensively since 2010. It provides various solutions for enhancing the user's experience, monitoring the user's behaviors, and improving the lifestyle. With careful design, these systems can be built with off-the-shelf smartphones and wearables. The detected result can be used as feedback for the user to understand his/her behavior, improve the lifestyle, or avoid the danger. Furthermore, the result also provides a valuable data source for the studies in psychology and sociology.

However, designing an IoT system to monitor human activities is challenging due to multiple factors. Some systems require high computing capability or a long time of data collection; some systems must detect some specific behaviors as quickly as possible in real-time; some systems suffer constant and irregular noise. In order to address these challenges, the designer must carefully consider the use case of the IoT system and select proper machine learning algorithms. This dissertation shows three designs of the IoT systems for the improvement of family mealtime experience and driving safety. The procedure for each design is introduced in detail, including the architecture of the system, the selection of features, and the evaluation of algorithms.

From the case studies in this dissertation, several special aspects of monitoring human activities are discovered. Since human activity is strongly related to the time-series and may change along time, the algorithm should be sensitive to context, be adaptive to dynamic conditions, process readable features, and benefit directly from prior knowledge. This discovery will serve as a guide about how to analyze and solve a problem with the IoT systems in the future.

Thanks to my family, who always supports me.
Thanks to my advisor, who always leads me.
Thanks to my friend, who shares the laughter with me.
Wish everyone all over the world a healthy and happy life,
and tomorrow will be better.

ACKNOWLEDGEMENTS

Here, I would like to express my deepest gratitude to all – my family, my advisor, my friends, and everyone who helps me in this academic career.

None of my achievement could be possible without the support from my family. Since I was 18, I was away from my hometown. Although they were far away, they always showed their hands and cheered me up when I was down. My wife, Linghan Kong, has spent more than six years with me and walked me through the most difficult time. Also I want to say thanks to my son, William, and my daughter, Joanna. They brought surprises and beauties to my life just like the twinkling stars above the night sky.

Thanks, Dr. Guoliang Xing, my dear advisor. He offered me a chance to explore the academic world. He guided me from the development of a small App to the processing with millions of data points. His insight helps me solving numerous challenging problems in my research.

Thanks, my Phd Committee members, Dr. Joyce Chai, Dr. Wei Peng, and Dr. Richard Enbody. From the courses offered by Dr. Joyce Chai, I have learned all the basic and advanced knowledge in this dissertation. Dr. Wei Peng provided valuable ideas that inspire me to refine my design and give more reliable and useful information for the study of human and sociology. The recent study of Dr. Richard Enbody about IoT helped me understand the relationship between users and devices, which is the major motivation of this dissertation.

Thanks, my friends, old ones and new ones. Life of a researcher is sometimes lonely, but I am so lucky to have all of you on my way. We have worked together, played together, and laughed together. One day in the future, we will get back together and share more stories of our colorful life.

Thanks, the world, the earth, and the universe. I will not waste the chance of the life as a human. If I am just a tiny stone, I will contribute to a tiny step to the tower of the science.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The topic of the Internet-of-Things (IoT) has been discussed and studied extensively since 2010. A large amount of hardware with embedded sensors and applications is developed to enable network connectivity and data exchange between vehicles, buildings, and other items. The IoT provides cheap and convenient solutions for some problems, such as augmented reality, activity and health monitoring, voice control, motion control, etc. The off-the-shelf smartphones and wearables are common choices for a host in the IoT, because of the variety of built-in sensors. The sensors include motion sensors or *inertial measurement unit* (IMU), microphone, camera, light sensors, thermometers, etc. The data from these sensors contain detailed and accurate information about our daily life, such as the user's movement, voice, and ambient noise. If that information is carefully processed, the user's action or the ongoing activity can potentially be inferred. This inspires an idea to develop a system that can monitor the user's activity based on the sensed data.

According to the clinical study, monitoring the activity is important in our daily life. It is not only a critical process of a large-scale study in human and sociology but also stimulation for the user's to improve their skills or daily routine. For example, the mealtime activity is important evidence for a family's status of employment and income, and it is strongly related to the children's obesity or relationship between the family members. However, except for keeping diary or video-recording, the mealtime activity is rarely monitored by an unobtrusive approach. Another example is the driving scenario. By monitoring the driver's action, the dangerous behavior can be detected and reported, to invoke the driver's alertness and reduce the risk for a road accident. Moreover, gesture recognition is a hand-free controlling method for the in-vehicle infotainment (IVI), which reduces the time of distraction while driving.

Due to the large variety of human activities, it is still challenging to conduct a study that captures all the detailed daily behaviors of a person. However, research on some common

activities can be a good start towards the ultimate goal. In this dissertation, the case studies focus on two major parts of daily life – eating and transporting. Meal plays a critical role in the physical wellness [37]; the conversation around the family's dining table is the most important sharing time between family members [21]. Driving is a popular choice of transportation, but the road accident is a major cause of injury nowadays [66]. If those activities can be monitored, the log of the daily behavior can stimulate the user to improve their lifestyle and fix improper habits. This also benefits other studies in sociology, like understanding the behavior of a group of people from the daily routine [45] or design training courses [11].

From the case study of those activities, several interesting aspects of designing an IoT system to monitor human activities are discovered. This is referred to as "**Three Rules**" hereafter. Specifically, a machine-learning algorithm that recognizes human activity should,

- **be sensitive to context**. Human activity is based on time-series. One activity is strongly related to the activity before and after it as well as other ambient conditions. For example, TV viewing and conversation may occur around the family mealtime. The *Conditional Random Fields* (CRFs) is an example that is sensitive to context. A simple implementation of the *Support Vector Machine* (SVM) is an opposite example of this, which only recognizes the activity within a short time frame.

- **be adaptive to dynamic condition**. Although in the short term, the activity has a relation to the context, one's behavior and habit may gradually change in the longer term. For example, the posture of the driver may constantly change during a long trip. Reconstructing the training data set and model is an example of the adaption to the dynamic condition. Using the fixed algorithm, training data set, and model is an opposite example of this.

- **process readable features**. Human activity can be generally described and recognized by a human. The information in human activity is mostly readable features,

like the clattering sound and conversation during a family meal. Different from the feature-based pattern recognition, the neural network uses a massive amount of the unreadable parameters, which is an opposite example here.

This dissertation introduces the systems that monitor those activities based on mobile devices, then evaluates the performance and summarizes the discovery from those studies. The three studies in this dissertation are from three levels of human activities respectively. High-level monitoring means recognizing an activity that lasts for minutes, like eating, TV viewing, driving. Mid-level monitoring means recognizing some details of the user's action but presenting the result with a classifier, like detecting whether the hand is holding the steering wheel. Low-level monitoring means describing the human's action or voice with real numbers, like localizing the hand position in a coordinate system in real-time. This dissertation argues that the algorithms for the monitoring of all levels should be selected according to the **Three Rules** introduced above. In the last chapter of this dissertation, the reason for selecting those rules are discussed and concluded.

CHAPTER 2

MONITORING FAMILY MEALTIME ACTIVITIES

This chapter introduces FamilyLog – a system to monitor the family mealtime activities with mobile devices. This chapter is adapted from a publication [16]. The author of the dissertation is the first author of the original work. "We" in this chapter refers to the author of the original publication. This work contains the software design on Android devices and the algorithm design in Matlab. The author recruited all the subjects, then collected and processed the data and the ground truth.

Monitoring the family mealtime activities is a high-level monitoring. It cares about the occurrence and duration of the family meal as well as the TV viewing and conversation. It does not recognize the detailed action or voice of the user.

2.1 Background

Research has shown that the family mealtime plays a critical role in establishing good relationships among family members and maintaining their physical and mental health [21][37][33]. In addition to the implications for family health, fine-grained analysis of family mealtime enables important studies in sociology and home economy. For instance, research has showed that the amount of shared time (including conversation and eating) between spouses and between parents and children have strong links with family income, mother's employment status, ages of children, and geographic location (urban or rural) [22][45][44]. However, according to a national survey in 2014, American families with children on an average work day spend almost 3 hours watching TV accounting for more than half of the leisure and sport time, while only 1 hour for family meal [91]. Moreover, the TV viewing during the family meal decreases the communication among the family members and increases the food consumption [10].

It is shown that activity logging is a very effective approach to improve the self-awareness

and motivate people to modify their behaviors toward a healthy lifestyle [39]. According to a study [116], a detailed analysis of the existing family routine helps parents choosing how to make positive changes to avoid the developmental delay of their children. Moreover, one's internalization of knowledge and behavior change can come from sharing and observing each others' family routines and the consequences of them [7]. Unfortunately, to date, there has been no unobtrusive and convenient methods to log family meals and related activities. Some of the available methods for family activity monitoring rely on video-taping [35], which not only incurs considerable installation/analysis costs, but also raises privacy concerns. There has been a number of studies on activity recognition using personal wearables and smartphones [62] [110]. However, as we argue in this work, detecting the activity of individual family members separately is insufficient for studying family communications, e.g., due to the fact that young children are usually not allowed to carry personal devices.

2.2 Related Work

In order to monitor family mealtime activities, several systems are designed to detect the usage of electrical appliances based on the electromagnetic interference and ambient sensors [94][47][57]. However, these systems can only detect the activities that involve substantial appliance usage. Recently, activity monitoring using mobile devices has received significant attention. Several systems are designed to detect food and drink intakes. For example, [62] presents the design of a fork with sensing abilities to help track and improve user's eating behaviors. In [110], the authors propose an approach of profiling user's gesture while eating using motion sensors on smartwatches. However, these systems are focused on tracking eating behavior of individuals, and are not suitable for detecting family mealtime activities, which may involve children without wearing any devices, and conversations among family members. Moreover, some mobile health systems are designed based on off-the-shelf smartphones to monitor human activities, such as sleep quality [48] or physical activities [97]. Several recent studies are focused on user experiences with mobile health systems such as privacy concerns

[6] and sharing behaviors [96]. However, these efforts are not concerned with studying family meals or group activities.

Acoustic event recognition algorithms have been widely adopted in smartphone-based activity monitoring systems. *Auditeur* [89] is designed as a mobile-cloud service platform to allow client's smartphone to recognize various sound events such as car honks or dog barking. *SoundNet* associates environmental sounds with words or concepts in natural languages to infer activities [84]. Recent work shows that the eating activity can also be detected by the acoustic features [111]. However, this work does not pinpoint main features for detecting family meals. It requires a large amount of data, and employs complex signal processing and machine learning methods, which raise burden of the implementation on mobile devices.

In order to detect the participants in the conversation, *Crowd++* [118] counts the number of speakers using MFCC (Mel-frequency cepstral coefficient) [104] features. Row mean vector of spectrogram [65] is a simple but effective method for speaker recognition by comparing the Euclidean distance of the energy distributional features.

2.3 Requirement and Challenges

In order to improve the self-awareness and motivate people to modify their behaviors toward a healthy lifestyle, a detailed analysis of the existing family routine is proved to be necessary [39] [116]. It is shown that, by reviewing detailed activity logs, not only people are motivated to modify their behaviors [39] [22] [45] [44], but also researchers in sociology are inspired to propose better solutions toward a healthy lifestyle [7].

There has been a number of studies on personal activity recognition using wearables and smartphones [62] [110]. However, we argue that detecting the activity of individual family members separately is insufficient. First, the existing solutions typically require the mobile device (smartphone or wearable) to be carried by the user. As a result, they cannot be applied to detect many activities of young children who are usually not allowed to carry personal devices. Second, many people do not carry smartphone or wear watch constantly

at home, making it difficult to monitor one's activity continuously. Moreover, detecting each individual's behavior is often unnecessary or significantly more challenging when she/he is participating in a group activity. For instance, detecting whether a particular family member is eating based on sound is more difficult when the family is having dinner together due to the higher level of ambient noise.

FamilyLog is designed to be an unobtrusive system that helps users keep track of their family mealtime activities. It employs the built-in accelerometer and microphone of smartphones and smartwatches to detect various information and activities related to a family meal. Specifically, FamilyLog is designed to meet the following requirements: 1) FamilyLog should be able to detect the occurrence of each family meal, which is defined as a shared dining activity among the family members [39]. Specifically, in our design, we focus on the participants and conversations during the family meal to avoid video/audio-recording the detail of the activities, which raise privacy concerns and are beyond the design based on mobile devices. 2) Since FamilyLog needs to operate in parallel with family mealtime activities. It must to be unobtrusive to use. It should minimize the burden on the user, e.g., without requiring the users to carry extra devices, and should not interfere with the users' daily activities by any means. 3) FamilyLog needs to monitor the details of family meals, including their start/end time, participants, and possible TV viewing, in a robust fashion, i.e., across different users, smartphones, smartwatches and households. 4) Since family meals involve privacy sensitive activities such as family conversation, the privacy of the family needs to be strictly protected. For example, the system should process the collected sensor samples on the fly and only keep the results, instead of storing or transmitting any raw data, which may contain sensitive information such as contents of the conversations. The sensing algorithms we develop can accurately classify a number of important contextual features of activities such as arm gestures from wearables, eating sounds, environmental noise, conversations, etc. As a result, in the future, these algorithms can be adapted and used as building blocks to detect a wide range of family activities such as parties, family meetings, gaming etc.

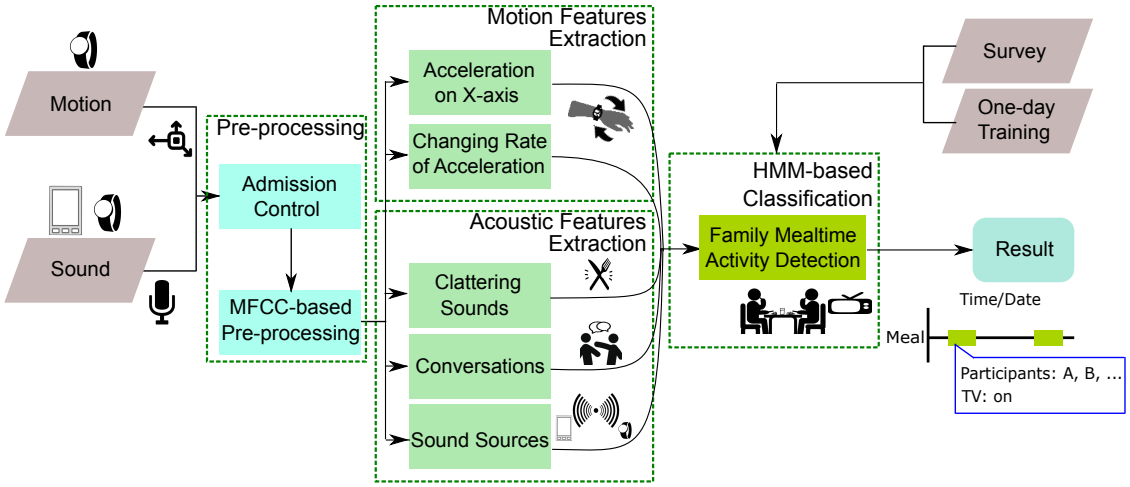


Figure 2.1: Architecture of FamilyLog

2.4 System Design

FamilyLog detects family meals by using the built-in sensors of mobile devices, namely microphone on smartphone/tablet and both microphone and accelerometer on smartwatch[1]. However, FamilyLog is designed to leverage these sensing modalities in an opportunistic manner depending on the availability of mobile devices in a home. In particular, FamilyLog may achieve satisfactory sensing performance even with a single smartphone when it is placed in the proximity of family activities (see Section 2.5). When multiple devices are available, FamilyLog runs separately on each individual device and fuses the detection results to achieve better performance and extended coverage.

As shown in Fig. 2.1, the proposed design consists of four components: *pre-processing*, *acoustic feature extraction*, *motion feature extraction*, and *CRFs-based activity classification*.

In pre-processing, sensors are sampled at certain rate and the samples are framed. Further processing will not be performed if the device is taken out of home (detected by the wireless network status) or the acoustic data only contains noise. Otherwise, the acoustic data is processed to extract energy features using filters based on *Mel-frequency cepstrum coefficients* (MFCC). In the acoustic and motion feature extraction components, FamilyLog groups data

[1]Most off-the-shelf smartwatches ship with microphone for voice control and making calls.

frames ($50ms$ by default, which covers the sound with frequency larger than 20Hz, i.e. the range of the human's auditory perception) into a detection window ($3min$ by default), and extracts a set of distinct features for each window. Specifically, FamilyLog extracts gesture-related motion features such as the average X-axis acceleration and changing rate, and acoustic features to detect the clattering sounds and the human voice.

To detect activities from extracted features, FamilyLog adopts a CRFs-based classifier. Compared with several commonly used classifiers like *Support Vector Machine* (SVM) that are only applicable to discrete event detection, CRFs can naturally capture the temporal pattern of family activities by incorporating continuous sensor input. The CRFs-based classifier is trained by a combination of short period of sensor data, e.g., a one-day family activities labeled by users, and some general knowledge of family meals which can be obtained from a one-time user input or a brief survey with simple questions such as "how much time does your weekday dinner usually take?".

2.4.1 Pre-processing

The primary objective of admission control and pre-processing is to reduce unnecessary computation as well as to prepare data for feature extraction. Specifically, it consists of the following three components.

First, FamilyLog reduces the unnecessary computation by discarding detection windows that likely contain only environmental noises (e.g., noise of appliances). Specifically, the noise detection is achieved by first calculating the *root mean square* (RMS) (i.e., the sound volume of signal) for each frame, and then computing the variance of RMS of all the frames within each window. A key observation is that a window with low RMS variance only contains ambient noise. Similarly, FamilyLog discards the motion data with low RMS variance, which typically indicates a stationary smartwatch not worn by the user.

Second, to increase computational efficiency, FamilyLog represents acoustic data with MFCC-based features, which will be used in later feature extraction. For each frame, Fami-

lyLog first calculates its energy spectrum from $80Hz$ to $8kHz$ with the *Fast Fourier Transform* (FFT) [109]. Then the resulting spectrum is transformed into 21 energy channels on the frequency domain by applying the Mel Filters [69][107], as shown in Fig. 2.2. In order to describe the pitch and the volume of the sound, we measure the energy of channel i, which will be represented as e_i hereafter. Different from MFCC, which is applied widely in speech recognition, especially in cases that requires user speaking directly towards microphone [93], our design does not apply logarithm and discrete cosine transform. Although MFCC can eliminate the impact from sound's pitch, it is not very robust in the presence of additive noise. In our case, to identify a sound as human voice or clattering sound, pitch is the most effective feature. The ambient noise are necessary to detect mealtime activities, such as sound from TV. Fig.2.3 shows the reason that we prefer the Mel Filter to the whole process of MFCC. Furthermore, if the result from Mel Filter satisfies our requirement, we can reduce the computational overhead by skipping other steps in MFCC.

Third, to preserve power, FamilyLog turns on sensor sampling only when the device is home, which can be determined by the system's location. Moreover, as an optional feature, FamilyLog can start the sensor sampling of a new detection window probabilistically based on the percentage of historical noise frames in a predefined time window. For example, for a particular family, they generally do not have a meal around 3:00pm on any day, FamilyLog can skip the detection at that time based on historical observations.

2.4.2 Feature Extraction

FamilyLog identifies the occurrence of the family meals by several key characteristics, based on sounds and gestures associated with dining and whether the family members are currently in close proximity to one another. Specifically, we use the following features to characterize the family meals. The first feature is the clattering sound caused by clashes between tableware. This is because the clattering sound is the most distinctive acoustic characteristic of family dining activity, regardless of other dynamics, such as the type of food and variation of

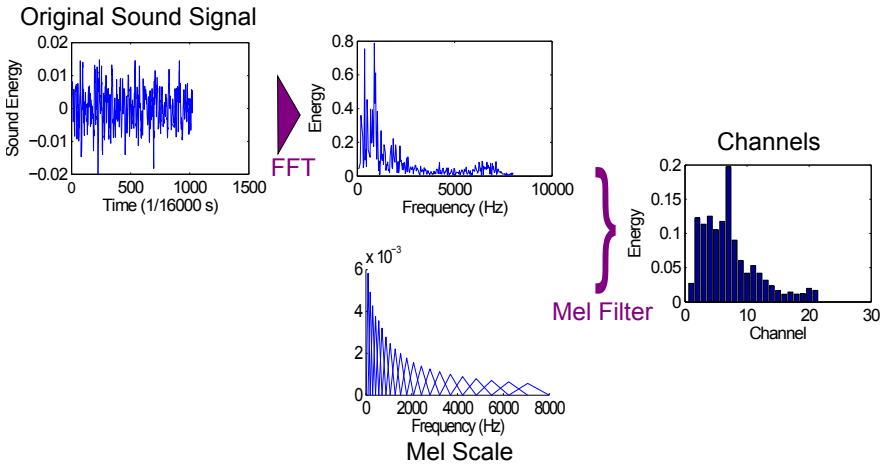


Figure 2.2: Transforming raw sound signal to 21 channels

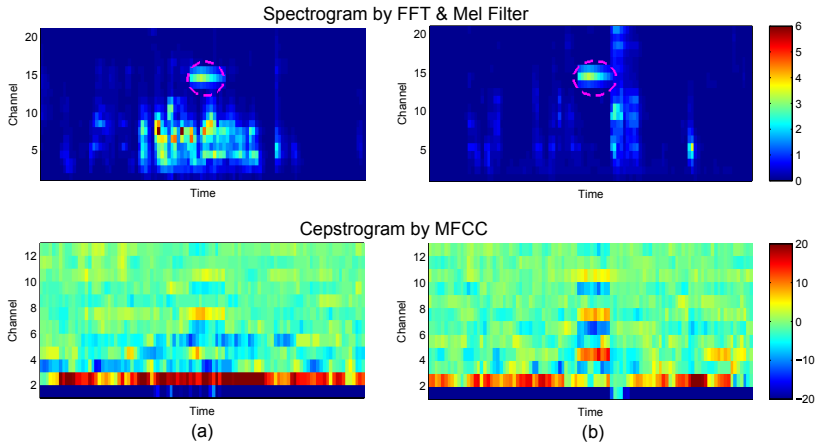


Figure 2.3: FFT & Mel Filter vs. MFCC

tableware. The second feature is the gesture of the users captured by smartwatches. When the user is holding food or using tableware, the arm of the user often exhibits a certain pattern of movements. The third feature is the human voice, i.e. the conversation between family members, which implies that the family members are near each other.

2.4.2.1 Clattering Sound

To infer family meal events, FamilyLog calculates the occurrences and frequency of clattering sound within a detection window. It looks for an energy peak from channel 12 to 16

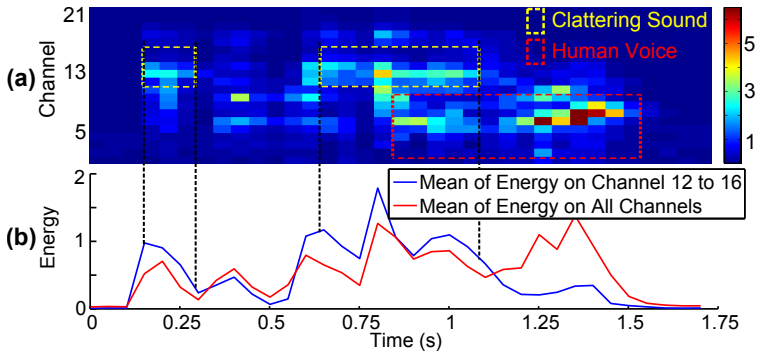


Figure 2.4: Example of clattering sound detection

(associated with frequency ranging from $1 - 4kHz$) for each $50ms$ frame. Specifically, for each frame, it computes $\overline{e}_{all}$, the average energy over all channels, and $\overline{e}_{12-16}$, the average energy across channel 12 to 16. The feature associated with clattering sound is calculated as $r = \overline{e}_{12-16}/\overline{e}_{all}$. For example, Fig.2.4 shows an example of clattering sound detection in a typical family meal scenario. Fig.2.4(a) shows the energy on 21 channels over time, and Fig.2.4(b) shows the corresponding $\overline{e}_{12-16}$ and $\overline{e}_{all}$. We can see that one occurrence of clattering sound may result in several continuous clattering frames with higher $\overline{e}_{12-16}$, even when the clattering sound and human voice overlapped around 1 second. Therefore, comparing $\overline{e}_{12-16}$ and $\overline{e}_{all}$ is a simple and effective way of detecting clattering sound in typical family meal scenarios. After obtaining r for each acoustic frame, FamilyLog calculates $E[N_{clattering}]$ which represents the expectation of amount of clattering sound contained in a detection window. Specifically, $E[N_{clattering}]$ is calculated as the sum of $P(clattering|r)$ which is preset in the system and generated using the data collected from 5 families.

Fig.2.5 shows an example of clattering sound detection based on the real data set collected in a home. We can see that all family meal windows contain large numbers of clattering frames. The clash of other objects such as keys and coins can also produce a similar sound. Different from clattering frames of dining activity, such false alarms are usually isolated and not likely to occur in a burst.

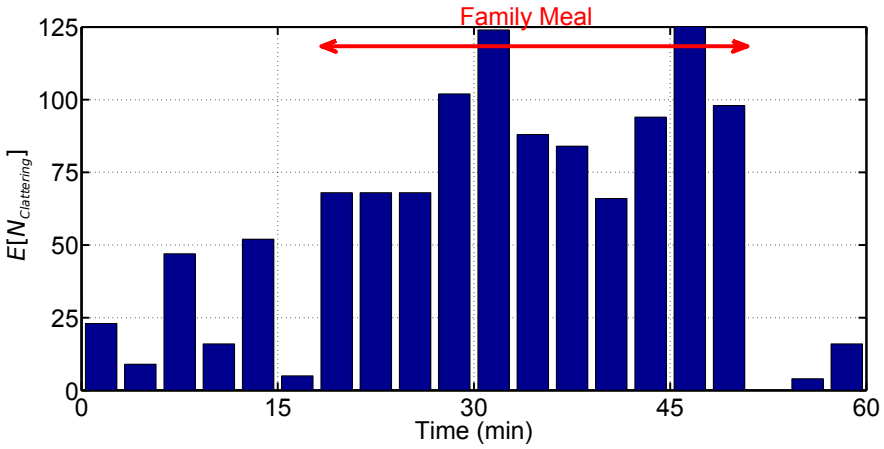


Figure 2.5: Example of family meal detection

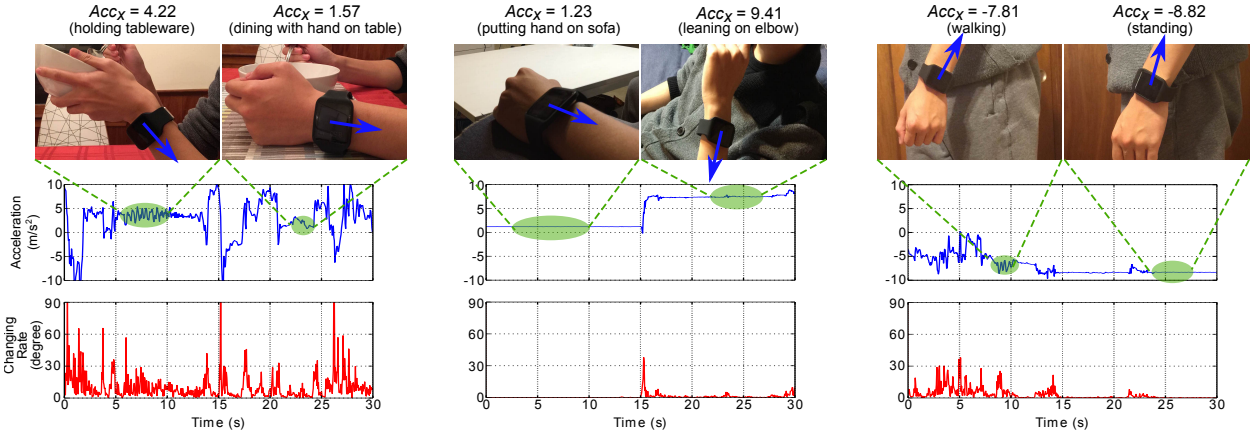


Figure 2.6: Examples of activities and motion features

2.4.2.2 Arm Movement

When smartwatch is available, FamilyLog also extracts motion-based features that characterizes dining behavior, which include the acceleration on the X-axis ($\overline{Acc}_x$) and the changing rate of the acceleration ($\overline{R}_c$). According to the fact shown in Fig. 2.6, the direction of the X-axis is always parallel to the arm. Furthermore, the acceleration on X-axis on a smartwatch is mainly determined by the gravity and the overall gesture of the arm. Therefore, it can be used as a simple and effective feature for inferring arm gesture while avoiding the overhead of data processing on the other two dimensions. Specifically, FamilyLog samples the built-in motion sensor on smartwatch and calculates two features for each frame. The

X-axis acceleration is directly read from the accelerometer. The changing rate between two frames can be computed as the angle between two acceleration vectors from them. Since the acceleration is mostly corresponded to the gravity, the angle describes how much the orientation of the watch face is turned along with the user's action. For a detection window, $\overline{Acc_x}$ is calculated as the average acceleration on X-axis for all frames, and $\overline{R}_c$ is calculated by the average changing rate of all neighboring frames. Fig.2.6 shows three typical activities and the motion features. We can see that the arm gesture and the movements of wrist during meal show distinct distributions.

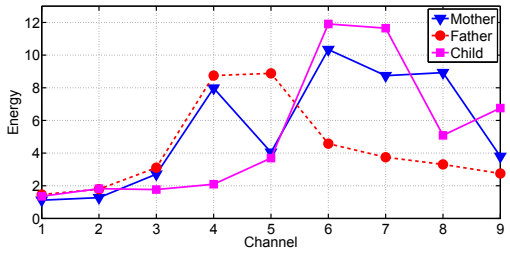


Figure 2.7: Average energy distribution of voice on the first 9 channels

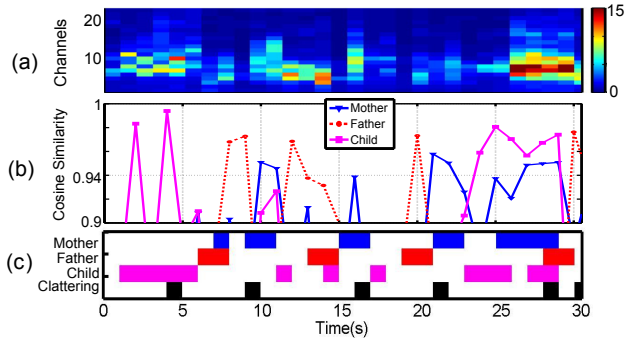


Figure 2.8: Example of conversation detection

2.4.2.3 Human Voice

An important acoustic feature for the detection is the conversation, which identifies human speech, as well as the family members who participate in it. Among all the family communications, the family meal is typically accompanied by a considerable amount of conversations.

The speaker recognition technique presented in [38] shows that pronunciation of vowels is a identical characteristic of human. However, maintaining a database for voice of each family member is costly for mobile devices. Here, row mean vector of spectrogram [65] provides an effective and efficient approach to recognize speakers by measuring Euclidean distance of energy distribution on frequency domain. Specifically, the family members are required to register their voice to FamilyLog by reading a short sentence. Fig.2.7 shows an example of the extracted signature of each family member's voice. We can see that each one's voice exhibits distinctive energy distribution. For example, in this family, the energy of the child's voice is mainly concentrated on channel 6 and 7, whereas the energy of the father's voice is concentrated on channel 4 and 5. For each frame, FamilyLog compares the vector from MFCC-based processing with the ones obtained during training, and calculates the probability that the frame contains voice of at least one family member by cosine similarity, represented as $P(voice|\boldsymbol{E})$, where $\boldsymbol{E}$ is the energy distribution in the frame, as shown in Fig.2.8. In a detection window, FamilyLog sums $P(voice|\boldsymbol{E})$ for each frame to extract $E[N_{voice}]$, representing the expection of number of frames that contains family members' voice.

2.4.2.4 Localization-based TV Sound

TV viewing detection is challenging because its acoustic signal often consists of a vast variety of different sounds. Even for a particular TV program, it is often challenging to find the underlying acoustic features to uniquely identify the TV viewing activity. Therefore, instead of relying on frame-based acoustic features, FamilyLog exploits the characteristics within a detection window to detect TV viewing. These characteristics, reflecting energy distribution and variance of pitch, are not only efficient to calculate, but also more robust across different TV programs, and much less susceptible to dynamics such as distance between the smart devices and TV. Specifically, to detect TV viewing, the system applies three features for each detection window, which are the volume distribution, pitch variance, and the sound

source. The pitch is defined as,

$$pitch = \arg \max_{i \in [1,21]} e_i \tag{2.1}$$

The P_{low} reflects the energy distribution within the window, and works well in separating TV sound from other "foreground" sounds that often involve human activities, such as family meal and conversation. This is primarily due to the fact that TV sound is usually more continuous (i.e., containing less pauses or quiet frames), as opposed to foreground sounds. Therefore, the energy distribution of TV sound is more right-skewed, resulting in less low-energy frames, and therefore has smaller P_{low}. Var_{pitch} also focuses on the low-energy frames within a detection window and describes the stability of pitch, making it a good supplement to identify TV sound. Due to its continuous nature, TV sound has a more stable pitch for low-energy frames, compared with other foreground sounds. Fig.2.9 shows an example of identifying TV viewing activity based on the feature space formed by P_{low} and Var_{pitch}. We can see that, the dining and TV viewing activities can be separated in the feature space.

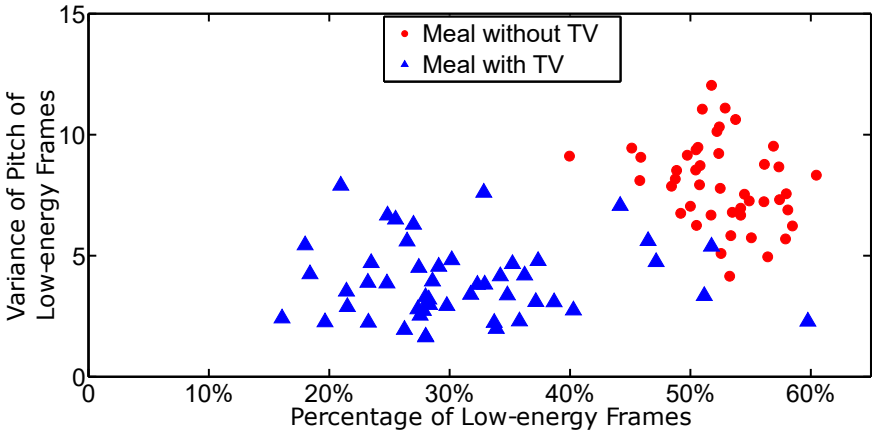


Figure 2.9: Data set of TV viewing

In order to detect TV viewing, two features is effective for each detection window. 1) *Percentage of low-energy frames* (P_{low}): The percentage of frames with RMS less than 50% of the mean RMS within a detection window. 2) *Variance of pitch of low-energy frames* (Var_{pitch}): The variance of pitch for frames with RMS less than 50% of the mean RMS

within a detection window. In our design, the pitch of the sound signal in a frame corresponds the channel with the highest energy. Thus, the number of dominating channel can be used to describe the pitch feature of acoustic signal.

Moreover, the performance of TV deteciton can be significantly improved by inferring sound sources in a detection window. Our idea is based on the following observation: TV is a sound source with fixed location, whose volume stays within a limited range for a relatively short period of time. Therefore, TV viewing detection can be translated into detecting single sound source with fixed location and limited volume range. To infer the location change of the sound source, FamilyLog employs an innovative approach based on *Interaural Level Difference* (ILD) [17] that fuses acoustic features captured by different devices to determine the origin of sounds. In this section we only focus on the fusion algorithm for two devices although it can be extended to more generic scenarios. Specifically, the process of feature fusion consists of three steps: similarity check, sound source detection in high-energy frames, and sound source detection in low-energy frames. In the first step, the goal of similarity check is to figure out whether two devices are at home and near each other by examining the similarity between sound captured by two devices. We define the detection windows that cover the same period of time on two different devices as the binaural detection windows. The similarity between binaural detection windows A and B can be calculated as folllows,

$$C(A,B) = \frac{\sum_{i=1}^{l} cos(\boldsymbol{E}(A,i), \boldsymbol{E}(B,i))}{l} \tag{2.2}$$

where vector $\boldsymbol{E}(X,i)$ is the energy distribution for frame i in detection window X. FamilyLog only proceeds to conduct sound source detection if $C(A,B)$ is above a threshold, indicating the two devices are in proximity to one another. The sound source detection aims to detect the number of sound sources in binaural detection windows. A key observation is that if all the acoustic signal originates from a single sound source, it is more likely caused by TV. In contrast, if the acoustic signal originates from multiple sound sources, it is more

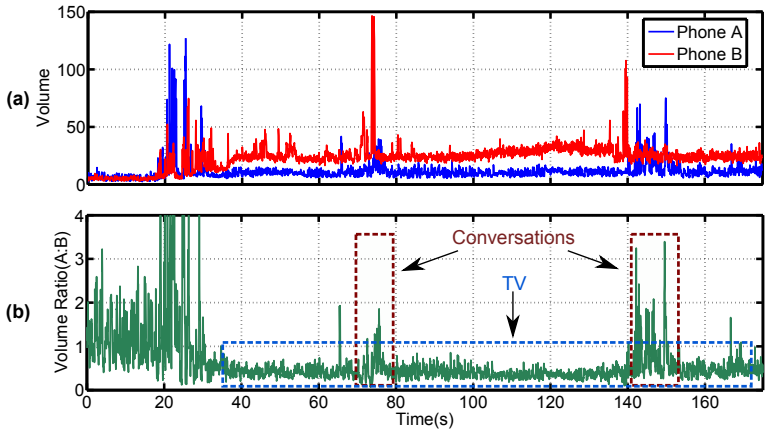


Figure 2.10: Example of TV viewing with conversation

likely to be caused by human activities other than TV. The method we use to detect sound sources is based on acoustic localization by ILD . Specifically, if the acoustic signal is from a single source and captured by two receivers, it satisfies $V_1/V_2 = d_1^2/d_2^2 = \Delta_V$, where V_1 and V_2 are volumes received by receivers and d_1 and d_2 are distances between receivers and sound source, calculated by the RMS. This equation can be applied to compute the relative distances between the sound source and the devices. In indoor scenarios, Δ_V may be impacted by various factors (e.g., echoes and obstacles), but its coefficient of variation is limited when d_1 and d_2 are fixed. To detect whether the acoustic signals come from the same source, we define *Coefficient of Variation of Volume Ratio per Frame* ($CV(A,B)$) in binaural detection windows A and B as:

$$\begin{aligned} CV(A,B) &= \frac{\sigma(\Delta_V(A,B))}{\mu(\Delta_V(A,B))} \\ \Delta_V(A,B) &= \left\{ \frac{V_{A,i}}{V_{B,i}}, i \in [1,l] \right\} \end{aligned} \tag{2.3}$$

Here, the volume of frame i in detection window X is represented by $V_{X,i}$, $\mu(\Delta_V(A,B))$ is the mean of volume ratios between A and B, and $\sigma(\Delta_V(A,B))$ is the standard deviation of volume ratios. $CV(A,B)$ thus is the ratio of the standard deviation to the mean. The lower $CV(A,B)$ is, the more likely the acoustic signals come from a single source. Fig.2.10 shows an example of how to detect sound sources by volume ratio. In the first 20 seconds, phone

B is carried by user from the dining table to the sofa. TV is turned on at the 30th second. During the 70th-75th second and the 140th-150th second, the subjects talk to each other. We can see that when the frames only contain TV sound, volume ratio is relatively stable. In contrast, as conversation involves multiple sound sources, the variance of the volume ratio is significantly increased.

By detecting the sound source with multiple devices, the accuracy of the detection of family meals can be improved in several challenging scenarios. Although TV programs that contain similar sound as family meal or conversation may be misclassified, the frames contain clattering sound and conversation still come from a single source, and they will be more likely from TV than family activities.

TV sound during the family meal can be separated from "foreground" sounds (clattering, conversation, etc.) by extracting low-energy frames, i.e. the frames that have a RMS less than the average RMS in a detection window. To detect whether TV is on during the family meal, we can check the volume of sound from all low-energy frames, and whether the acoustic signal is probably from a single sound source. If the TV is on, the continous sound from TV will rise the volume of low-energy frames, and $CV(A, B)$ of all low-energy frames will have a relatively low value, indicating the sound comes from a single sound source with fixed location.

2.4.3 CRFs-based Classification

Similar to speech and gesture recognition, the family meal detection involves identifying a temporal pattern rather than detecting discrete events. We design the classifier of FamilyLog based on CRFs, where we treat extracted features as observations, and the family event contained in each detection window as hidden state. Therefore, the primary goal of the our CRFs-based classification is to recover the family events overtime using the features extracted from a sequence of detection windows.

Fig.2.11 shows the graph of the CRFs. We can see that in this case, the state is either

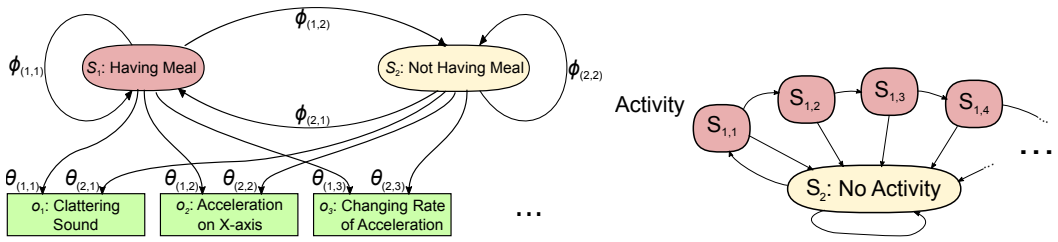


Figure 2.11: CRFs model of one family mealtime activity

Y_1 as "activity" or Y_2 as "no activity", and the emissions includes all features related to this activity from each detection window. Because the transition probability from Y_2 to Y_1 is correlated to the duration of the activity, we additionally define state $Y_{1,i}$, which means the activity lasts for i detection windows. With the features in a sequence of the detection windows as the input, the classifier outputs a sequence of the states for the detection windows with the maximum likelihood.

2.4.4 Transition Probabilities

The set of transition probabilities contains four entries $\{p(Y_1|Y_1), p(Y_1|Y_2), p(Y_2|Y_1), p(Y_2|Y_2)\}$. According to the definition of our CRFs, we have $p(Y_1|Y_1) + p(Y_1|Y_2) = 1$, and $p(Y_2|Y_1) + p(Y_2|Y_2) = 1$. This means, in order to calculate all the transition probabilities, we only need two known values, which are the probability to start an activity and the probability to stop an activity. The first value corresponds to the probability distribution of one activity's occurrence related to time/date. The second value corresponds to the probability distribution of the duration of one activity. These can be estimated based on one-time user answers to questions like "What's the typical frequency and duration of your weekday family meals?". Alternatively, they can be derived from historical detection results.

To improve the accuracy of such an approach, FamilyLog presents intuitive system UIs that allow users to rate previous detection results. The characteristics of family meal, including time, duration, and frequency are often highly dependent on the day of the week. Therefore, FamilyLog generates different models for the weekdays and weekends.

2.4.5 Emission Parameters

The set of emission probabilities contains entries as $p(X|Y)$, which describes the probability to observe the features X in state Y. The observation p within a detection window is represented as a vector of features, i.e. $X =< x_1, x_2, x_3, ... >$, where x_i corresponds to a feature related to the activity. For the detection of the family meals and TV viewing, the features are shown in Table.2.1 and Table.2.2. Here, each feature is represented as a real number.

Table 2.1: Features for the family meal detection

Term	Description
$E[N_{clattering}]$	The expectation of number of frames containing clattering sound
$E[N_{voice}]$	The expectation of number of frames containing the family members' voice
$\overline{Acc_x}$	The average acceleration on X-axis
$\overline{R_c}$	The changing rate of acceleration

Table 2.2: Features for the TV viewing detection

Term	Description
P_{low}	The percentage of low-energy frames
Var_{pitch}	The variance of pitch of low-energy frames
$CV(A, B)$ (optional)	The coefficient of variation of volume ratio per frame in binaural detection windows A and B

The CRFs classifier is trained by a period of sensor data. Typically, at least a whole day is required to fully cover the communication of family. After training, we apply Gaussian *Kernel Density Estimation* (KDE) to calculate the *Probability Density Function*, corresponding to the observations associated to each state [53].

2.5 Performance Evaluation

In order to evaluate the performance of FamilyLog, we have collected 77 days of data from 37 subjects in 8 families (details shown in Table 2.3). The procedure of the data collection has been approved by the *Institutional Review Boards* (IRB) at the Michigan

State University. The period of data collection was one or two weeks for each family. We intentionally chose families with young children for this study because family routine analysis has important implications for children's health. Our results also showed that small children often sometimes presented challenges to event detection due to the excessive noise they make at home.

Table 2.3: Families that participated in the experiment

	Children (Ages in Years)	Phone	Smartwatch	Data (Weeks)	Meals
1	1 daughter(5)	Nexus 4	N/A	1	4
2	1 daughter(4)	Nexus 4	N/A	1	6
3	2 daughters(5, 8),2 sons(1, 3)	Nexus 4	Sony Smartwatch 3	2	9
4	3 sons (1, 3, 5)	Nexus 3	Sony Smartwatch 3	2	16
5	2 sons (3, 5)	Moto G	N/A	1	5
6	2 daughters(1,3),1 son(7)	Moto G2 × 2	Sony Smartwatch 3 × 2	2	22
7	2 daughters(3,11),2 sons(7,13)	Moto G2 × 2	N/A	1	10
8	3 daughters(7,10,18)	Moto G2 × 2	Sony Smartwatch 3	1	6

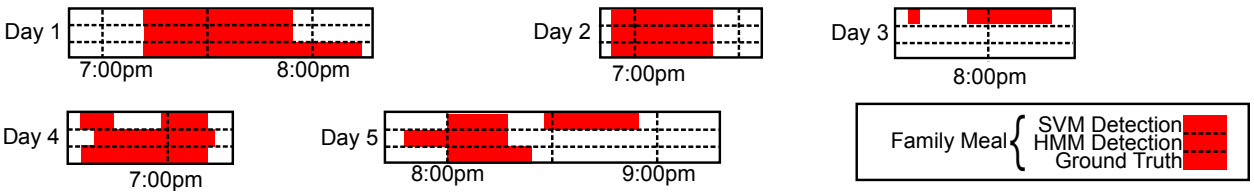


Figure 2.12: Detected family meals based on data collected from family 4 during 5 days

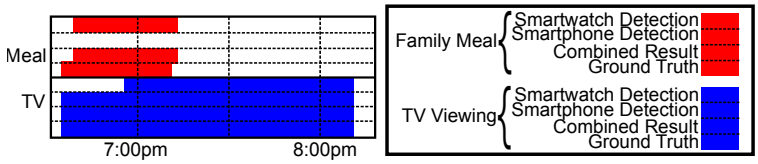


Figure 2.13: Detected family routine by one smartwatch, one smartphone, and their combination

We provided each family one or multiple devices. An app pre-installed on the devices continuously records audio and motion unless the device is taken out of home. Users may manually start/end the app on any device. The parents of the family are required to carry

a smartphone as his or her personal smartphone, and another smartphone is kept at a relatively fixed location at home (e.g., kept charging in the living room). At least one of the parents of the family are required to wear a smartwatch if smartwatches are provided. These requirements take into account the habits of different users (i.e., carrying the phone or leaving the phone at a relatively fixed position at home, etc.). We also offer family members the opportunity to review the recording and delete the part of recording that raises privacy concerns. We adopted two methods to obtain the ground truth, which include an interview with family members immediately after data collection is finished, and listening to the recordings to manually label family activities.

2.5.1 Micro-scale Routine Analysis

FamilyLog is designed to provide fine-grained family routine logging, which allows family members to review their activities and improve family lifestyle. To evaluate the performance of our CRFs-based classifier, we compare our classification result with the result classified by the *Support Vector Machine* (SVM), which recognizes the activities only based on features in individual detection windows rather than considering their temporal nature. The overall performance of FamilyLog and its comparison with SVM will be discussed later in Section2.5.2.

Fig.2.12 shows the detection results along with the ground truth of the data from 5 days in Family 4. We can see that the family usually has dinner around 7-8 pm for about an hour, except for day 5, which is Friday, when they started dinner at around 8 pm for about 20 minutes. Compared with the ground truth, we can see that FamilyLog is accurate in detecting most of the meals. In day 3 and 5, the SVM classifier yields a few misclassifications due to the interferences caused by TV viewing. However, FamilyLog's CRFs-based classifier is able to avoid such false negative errors. Furthermore, by taking into account the temporal nature of family routine activities, CRFs is able to minimize the short false negative and false positive classification results.

When multiple devices are available in the family, the output of FamilyLog is a combination of detection results from all devices. Fig.2.13 shows the detail of the detection result of the 4th day in Fig.2.12. In this case, the smartphone was put on a coffee table near TV, and the smartwatch was worn by a subject. The TV sound was correctly detected by the smartphone. However, some of the family members including the wearer of the smartwatch were having a meal around 7:00pm, and they were far away from the TV. Therefore, the smartwatch was the only device that was able to detect the family meal, but it failed to detect the TV viewing at that moment. The combination of both results shows a completed detection result by FamilyLog, which included all activities that are able to be detected.

2.5.2 Evaluation of Activity Detection

In this section, we investigate the overall performance of HomeLog in detecting family routine activities. For each individual family, the CRFs-based classifier is trained using the information from the survey and data labeled by the subjects collected in the first day. We use the precision and recall as the metrics for this evaluation. Specifically, the precision of detecting activity A is defined as the ratio of the number of true-positive windows to the total number of windows detected as A. The recall of detecting activity A is defined as the number of true-positive windows divided by the total number of windows labeled as A. We do not take into account the true negatives, because most of the windows containing no activities are able to be detected and discarded. In addition, we also present the evaluation result after making certain relaxation (e.g., $\pm 3min$) on the start/end time of family activities. Note that our design objective will not be affected by minor errors in start/end time, as long as the the system is able to accurately identify the occurrences of the family routine activities.

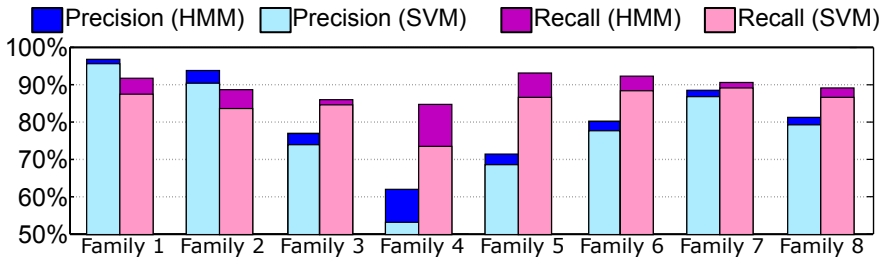


Figure 2.14: Overall accuracy of family meal detection in detection windows

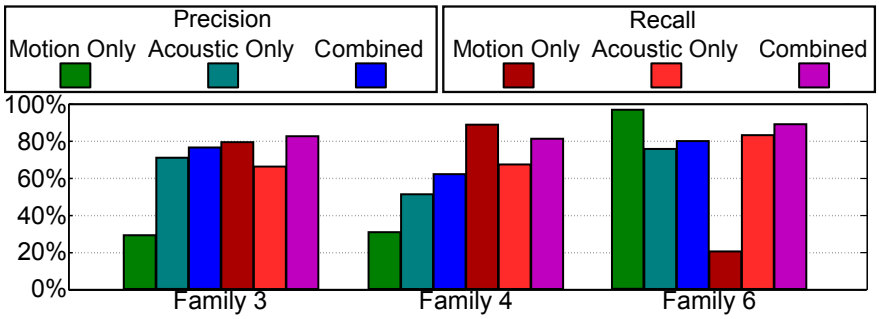


Figure 2.15: Accuracy of family meal detection using only motion or acoustic data

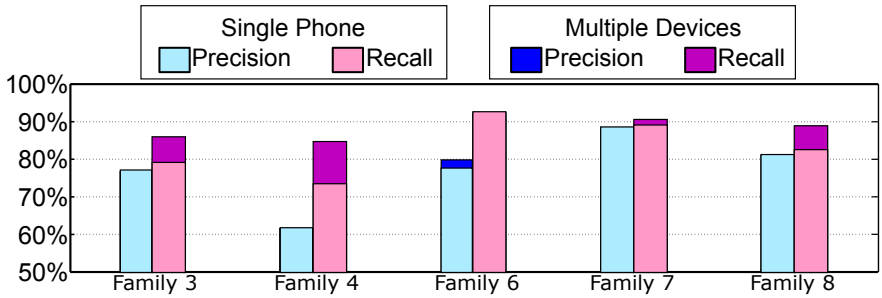


Figure 2.16: Accuracy of family meal detection by a single phone or all the available devices

2.5.2.1 Family Meal Detection

The evaluation result of family meal detection is shown in Fig.2.14. Our CRFs-based classifier outperforms SVM by 6.82% on average in recall. This is primarily because CRFs is more effective in correcting isolated false negatives. We can also observe that FamilyLog achieves an overall precision of 81.1%, with the highest being 91.1% for family 1 and lowest being 62% for family 4. We found that the two major causes of the relatively low precision in

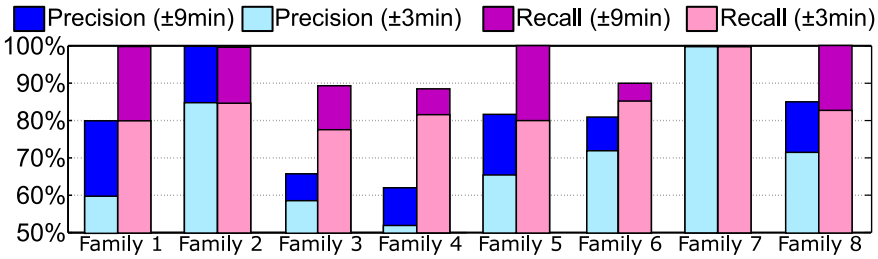


Figure 2.17: Accuracy of detection for each occurrence of family meal by relaxing the start/end time by 3min and 9min

family 4 and 5 are the high pitch voice from children and music, which have similar acoustic features as clattering sound during meal. However, since these sounds usually have a short duration, FamilyLog is able to correct a considerable amount of the resulting false positives.

For the detection that is only based on the motion data from the smartwatch or the acoustic data, the accuracy is shown in Fig.2.15 for Family 3, 4, and 6. For the Family 3 and 4, the precision is very low when only motion data is used. The reason is that the motion data for the eating action can be very similar to some activities like reading or writing, especially when the smartwatch is wear on the non-dominant hand. On the other side, the recall is relatively high, because most of the family meals are able to be correctly detected by the motion data. Moreover, the smartwatch in Family 6 is rarely worn when they are at home, and the detection based on the motion data is not always reliable. Generally, the features from the acoustic data contribute the major part of the detection, and the motion data can assist the detection in some special cases. For example, depending on the food, the clattering sound may be weak for a family meal, but the detection result can still be correct due to the conversation between the family members and the eating action.

Fig.2.16 shows a comparison of the accuracy of the detection by a single smartphone or all the available devices in a family. If FamilyLog only runs on a single device, the sound source detection will be unavailable. This happens in Family 6, where the sound from a TV program about cooking is wrongly detected as a meal without knowing the sound sources. Furthermore, during a family meal, if one device is left far away from the dining table but

another device is near, it is possible that the meal can only be detected by one device. By combining the results from all the available devices, FamilyLog is less likely to miss a family meal than only relying on one of them.

Fig.2.17 shows the detection accuracy of the occurrence of each family meal. We can see that FamilyLog rarely fails to detect an occurrence of family meals with the $9min$-relaxation on the starting/ending time, achieving 88.7% precision and 93.3% recall on average. The detection error in each family meal's duration is about 4 minutes on average.

2.5.2.2 TV Viewing Detection

Next, we investigate the performance of TV viewing detection with data collected from 7 families. From the interview with the family, we learned that family 1 usually does not view TV. Fig 2.18 shows the accuracy for each family. We can see that HomeLog is able to achieve more than 95% precision for family 4 and 5, and over 90% recall for family 2, 3 and 4. Moreover, compared to other families, family 3 has a relatively lower precision (76.65%). The main reason is that the data from family 3 contains a considerable amount of sound from the parents and children singing along with guitar, which was only captured by one device. Therefore, the system is not be able to infer the number of sound sources which is considered as the key feature that helps differentiate between such activity and TV viewing. Family 5 has watched several TV programs containing discontinuous sound, which is also a typical scene where the TV detection loses accuracy. When more devices are deployed in family 6, 7, and 8, the sound source can be accurately detected, which helps distinguish the dining sounds from TV. The precision is raised by 5% comparing with other families on average. In our experiment, a number of party-like events are observed in which children were playing and talking to each other. The noise (e.g., those from pets or party-like events) can also possibly be detected as continuous sound from TV. By taking into account the detected sound sources, we observe that the recalls in family 6, 7 and 8 also outperform other families, because the discontinuous low-volume TV sounds can be correctly classified.

It is worth mentioning that this kind of TV programs is not often observed in our recordings (less than 20min/week in each family).

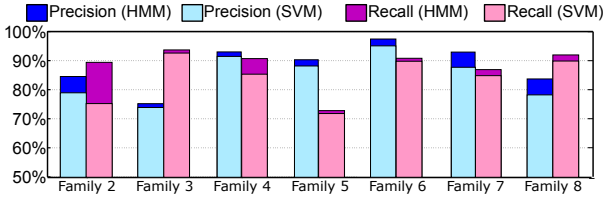


Figure 2.18: The evaluation of TV viewing detection

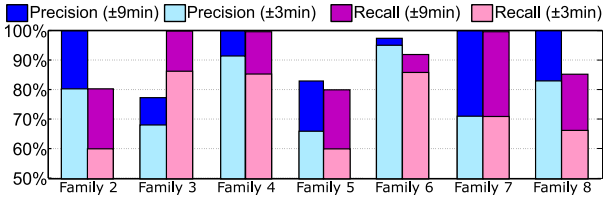


Figure 2.19: Accuracy of detection for each occurrence of TV viewing by relaxing the start/end time by $\pm 3min$ and $\pm 9min$

Fig.2.19 shows the detection errors of the occurrence of the TV viewing. The detection error in each TV viewing's duration is about 6 minutes on average. Under the $9min$ relaxation on the starting/ending time, HomeLog achieves 92.4% precision and 90.8% recall on average.

2.5.2.3 Conversation Detection

The human voice serves as a clue for family meals, and is also unique feature of a participant. With the permissions from the Family 1-5, we listened to the raw acoustic data provided by them, and manually labeled the family members who have talked during each family meal. For each family, we only focus on the mother, the father, and one selected child aged between 5-12. We count the number of detection windows that FamilyLog can correctly detect all the participants, and calculate the precision and recall. Fig.2.20(c) shows that, FamilyLog achieves high accuracy in participant detection across different families, with the average

precision and recall being 97.8% and 92.8%, respectively. This means most of the detection windows yields correct results for participant detection. This also ensures a high accuracy for detecting all participants for each occurrence of the family meal.

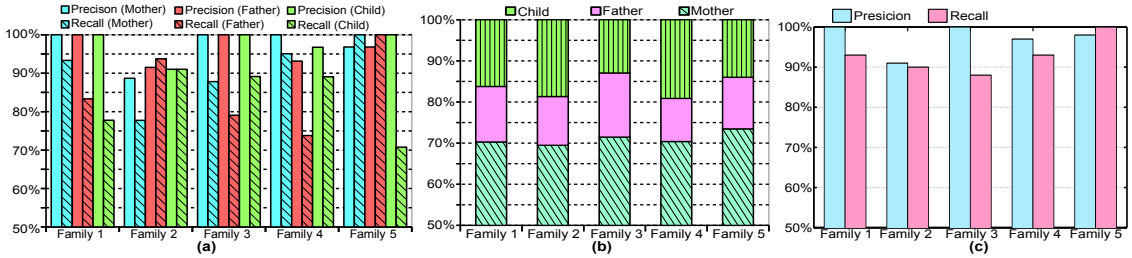


Figure 2.20: Evaluation of participant detection

Fig.2.20(a) shows the participant identification result for each family member. One key observation is that the overall recognition accuracy of other family members are better than that of fathers. This is mainly due to the fact that father usually speaks with short sentences or only phases, which are more difficult to detect. Fig.2.20(b) shows the proportion for each family member in overall conversation. We can see that father speaks less frequently that other family members, which is also consistent with the findings from social behavior studies [108]. Another observation is that the child from family 5 has a relatively low recall. This is mainly because he often speaks with different tones, thus the voice is difficult to identify using the signature extracted from his training data.

2.6 Interview and Discussion

In addition to quantitative performance evaluation, we also have investigated user experience by conducting 1-hour interviews with each of the 6 families (family 3 to 8) that participated in our experiments. The feedbacks from interviews shed lights on the usability of FamilyLog as well as the challenges of long-term deployment in real families. During the interview, the subjects discussed following topics about the performance of FamilyLog, the user experience, and the ideas of improvement.

- Was FamilyLog convenient to use?
- Were there any privacy concerns?
- Was FamilyLog accurate?
- What was your own opinion about your family routine?
- Would FamilyLog be useful, and why?
- Which new function should be added to FamilyLog in the future?

According to the feedback, all six families expressed great interests in tracking their routines by using mobile devices. They believed a precise and obtrusive monitoring can help the improvement of their behaviors. They were also very interested in comparing results with other families since they could be more aware of the problems of themselves. This discovery may inspire a new application on the social media in the future.

Only one subject felt uncomfortable of recording and uploading acoustic features, and the most of the subjects mentioned that they only felt slightly uncomfortable on the first one or two days. To mitigate the privacy concern, FamilyLog processes sensor data locally and only uploads extracted features, which cannot be used to recover the raw data, to the cloud. It does not store or upload raw data without the user's permission. We believe FamilyLog can be developed along with the emerging IoT devices like Amazon Echo [29] to meet the requirement of the privacy and security.

2.7 Conclusion of Study

FamilyLog monitors high-level activities, like conversation, TV viewing, and meals. It satisfies the **Three Rules**. Specifically, FamilyLog

- **is sensitive to context**. The CRFs-based classifier considers the transition probabilities among states. This is a strong context-sensitive classifier. The assumption is

a complete activity should not be discrete or only lasts for a short time. It is proved that this classifier has better performance than simple SVM.

- **is adaptive to dynamic condition**. FamilyLog is able to build the training data set for each family individually. It could use the user's feedback and historical features as the training data set for the future processing.

- **processes readable features**. All the features that are used in FamilyLog are readable by human, like the clattering sound, action of eating, and human voices. They are listed in Table 2.2 and Table 2.1.

CHAPTER 3

MONITORING DRIVER'S HAND POSITION

This chapter introduces SafeWatch. This is a system that aligns the coordinate systems of the mobile devices to the vehicle, thus it enables the devices to know their attitude and position in the real time. This chapter is adapted from a publication [14]. The author of the dissertation is the first author of the original work. "We" in this chapter refers to the author of the original publication. This work contains the App design on Android devices. The author recruited all the subjects, then collected and processed the data and the ground truth.

Monitoring the hand position is a mid-level monitoring. It outputs the current position of the hand like "at the higher position of the steering wheel" or "at the lower position, away from the steering wheel". The system works when the user is driving and it does not describe the hand position in a coordinate system.

3.1 Background

Recent studies revealed that the distractions and the secondary tasks occurring inside the vehicle increase the risk of a road accident by 2 to 10 times [67]. Examples of secondary tasks include using a mobile phone, adjusting air conditioners, operating the on-vehicle entertainments, eating and drinking, etc [23] [59]. Relaxing due to the boredom may also lead to unsafe driving behavior, e.g., one hand unconsciously moves away from the steering wheel. When the drivers are distracted or losing alertness, their braking response time is significantly longer than usual, and they could fail to maintain the control of the vehicle [72]. It is shown that similar to aggressive driving and risky driving, lost concentration and minor loss of control are among the top conditions related to traffic accidents [31].

It is recommended by the *American Automobile Association* (AAA), the driver should hold the steering wheel firmly with both hands at the 9 o'clock and 3 o'clock positions [5].

However, it is difficult for a driver to maintain self-awareness of their hand positions due to boredom or drowsiness of a relatively long trip [103] [101]. Stimulation like music may improve the alertness of the driver while sometimes causing distraction and increasing their mental efforts [113]. Research shows that a system that accurately detects the incoming danger and warns the driver can not only invoke the driver's alertness but also enable them to self-improve their driving skills [11]. Moreover, the detection can provide valuable information for constructing a *Vehicle-to-Vehicle* (V2V) system for safety[50]. Recently, the emerging of Internet of Things (IoT) systems for smart driving provides a promising solution. Several methods have been developed to detect the driver's dangerous actions [70] [68] [73] [74] [75]. However, these designs require additional devices such as cameras, PPG sensors or pressure sensors, presenting the barrier to wide adoption. More recently, several systems are developed to track the user's hand movement with smartwatches [121][88]. However, they often yield unreliable performance while driving due to the impact of the vehicle's movement. Some study shows that the smartwatches can be used to detect several important driving behaviors like the angle of the steering wheel [79] [64]. However, these detections are not effective in some cases, especially when the hands are staying somewhere other than the steering wheel.

Several major challenges must be addressed in the design of high-performance driving monitoring systems based on wearable devices. The motion sensor samples from a smartwatch in a moving vehicle not only contains the hand's movement, but also includes impacts from the vehicle's acceleration, turning, and *Noise, Vibration, and Harshness* (NVH) from the engine and the road condition. Due to the significant variation across different devices, drivers, and vehicles, it is difficult to design a robust classification algorithm to detect the hand's position based on motion features. Moreover, the driver may switch postures during one driving trip, resulting in several different sets of the motion data.

3.2 Related Work

The dangerous action by the driver is mainly caused by drowsiness and distraction [63]. To keep the driver concentrating on the driving, the effective methods are alerting the driver for the incoming danger [11] or giving feedback for the driver's level of drowsiness [4]. These methods require a way to monitor the driver's mental or physical status. The driver's behavior can be recorded by a camera. However, video-recording raises privacy concerns. Another approach is to leverage various sensors including motion sensor, barometer, and GPS on the smartphone to monitor the driving style (e.g. risky or aggressive), track the vehicle and learn the road information [61] [26] [51]. However, this approach only analyzes the vehicle's movement, and it cannot acquire knowledge about the driver's behavior inside the vehicle. Several efforts attempt to monitor the drivers' drowsiness with proprietary biomedical sensors (e.g. heart rate sensors) [76][20] . However, these sensors are not readily available on off-the-shelf mobile devices.

In recent years, several approaches are proposed to detect the driver's hand position. For example, using the sensors around the steering wheel, the driver will be alerted when his or her hand is not holding the steering wheel due to drowsiness or distraction [70] [68]. Another viable method is to detect the grip strength of the hands with pressure sensors on gloves [73]. However, these methods require additional equipment and raise the burden of usage.

Hand posture recognition using motion sensors on wearables or smartphones has been studied extensively. For instance, the user's finger-writing [119] and the hand gesture [121][88] can be classified using motion features. A common limitation of these methods is that they assume the movement of the user's arm or hand is the only cause of the smartwatch's movement. However, in our case, the vehicle's movement continuously impacts the motion data captured by any device inside the vehicle, and thus those methods cannot be applied here. If the hand is always on the steering wheel, the turning operation can be traced by the smartwatch, including the starting and ending position [79]. A recent study shows that, by detecting the direction of the hand's movement, it can be inferred that when

the hand leaves from or returns to the steering wheel [75]. However, these methods require precise alignment between the coordinate systems among multiple devices based on magnetometer and compass. Those sensors are often highly inaccurate or unavailable on some devices. Moreover, the direction of the hand's movement only provides partial information of the hand's position. For example, if the driver is handling a complex task like eating, drinking or grabbing an item, the hand's movement will contain a series of different directions, resulting in difficulty to infer when the hand returns to the steering wheel.

Another study of ours shows that the secondary task while driving can be classified by the angle of the driver's forearm rotation [100]. Although this method is effective to detect and identify the driver's behavior, it still needs to know the moment that the hand is on the steering wheel, because the basic assumption is the hand's movement always starts at the steering wheel.

3.3 Requirements and Challenges

A system called SafeWatch is designed to help drivers keep concentrated on driving. Specifically, it detects whether the driver holds the steering wheel with both hands, and reports the dangerous actions, e.g. one hand is away from the steering wheel or keeping moving.

SafeWatch needs to meet the following requirements: 1) As it operates in parallel with driving, it must be unobtrusive to use. It cannot interfere with the driver's activity or require any manual input by the drivers at runtime. (2) To ensure wide adoption practice, the training process of the system should be intuitive and require the minimum amount of efforts/time. (3) It needs to detect the positions of hands relative to the steering wheel robustly, i.e., across different drivers, vehicles, and smartwatches.

To meet these requirements, three challenges need to be addressed. First, SafeWatch must be able to detect hand motions using accelerometer and gyroscope readings in the presence of significant interference from the movement of the vehicle. For example, the

gyroscope on the smartwatch produces highly similar motion features when the vehicle is turning left or the driver is rotating the arm towards left. As a result, the driver hand motion may be falsely classified due to the impact of vehicle movement.

Second, it is challenging to design the training process due to several reasons. First, to detect fine-grained hand motions, a training process is necessary for each combination of driver and vehicle. However, the ground truth is difficult to collect without a user's manual input or video-recording equipment. Moreover, traditional machine learning algorithms require the training set to contain data from both positive and negative classes. That is, SafeWatch should record the motion data not only when the driver's hands are holding the steering wheel, but also when the driver's hands are away from the steering wheel. Such a process is not feasible as it poses potential dangers for drivers. Furthermore, the driver may handle various secondary tasks during driving. Thus the distribution of motion data is highly unpredictable when a hand is away from the steering wheel, presenting challenges for training an accurate classifier.

Third, the motion data captured by the smartwatch is highly dependent on how it is worn. A typical issue is that the position of the smartwatch on the user's wrist might change due to the hand/arm movement. Moreover, the driver's posture also can change during one driving trip. SafeWatch must adapt itself to such dynamics to maintain the accuracy of detection.

3.4 System Design

3.4.1 System Overview

SafeWatch is a wearable sensing system that can accurately track distracted driver hand gestures. Specifically, it detects whether the driver's hands are on/off the steering wheel or on the upper/lower positions of it, which is an enabling primitive for various applications of smart driving and has important implications for improving driving safety. To this end, SafeWatch senses the motion of the vehicle and the driver's hands using a smartphone

placed in the vehicle, along with the smartwatches worn on the driver's left or right wrist, respectively. A detector running on the smartphone collects data from different devices and accurately classifies driver hand gestures despite the strong interference introduced by the vehicle's acceleration and turning, as well as noise, vibration, and harshness from engine and road conditions.

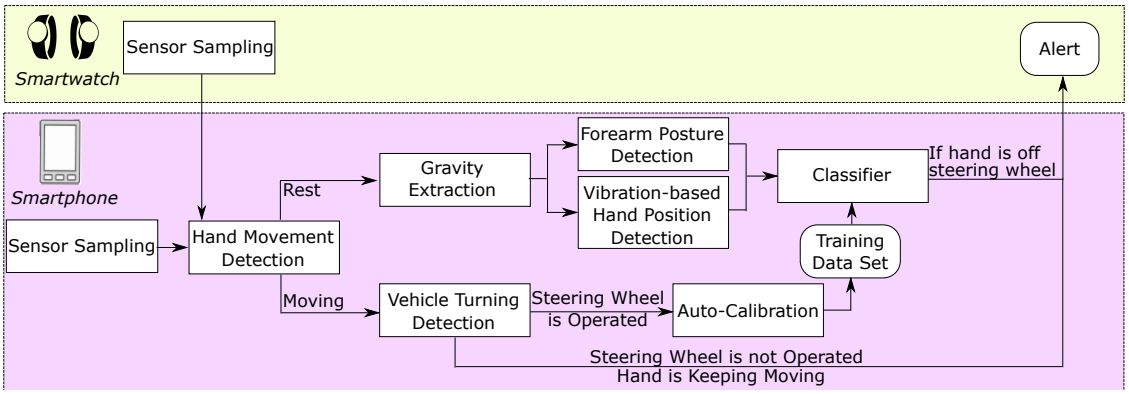


Figure 3.1: Architecture of SafeWatch

Fig. 3.1 illustrates the architecture of SafeWatch's sensing pipeline. SafeWatch continuously samples and processes the built-in accelerometers and gyroscopes of smartwatches and smartphone. Samples collected from every device are transferred to the smartphone and fused with a hand movement detector, which first mitigates interference introduced by the move of vehicle, and then detects hand movement based on cleaned motion signals. When the hand is moving, SafeWatch detects driver distraction by inferring whether the gesture is a steering wheel manipulation or behavior related to secondary tasks such as drinking/eating, tuning radio, etc. When the hand is still, SafeWatch classifies hand postures based on two features, including the posture of the driver's forearm learned from the gravity direction of smartwatch, and the vibration sensed on the driver's wrist, which manifests distinct magnitudes when the hand is on/off the steering wheel. In practice, the above features may exhibit different characteristics depending on various factors including the user's driving habits, the posture of the smartwatch, as well as the model of engine that may affect the vibration magnitude of car body. To maintain detection accuracy across different environments, SafeWatch

employs an auto-calibrator that leverages sensing data collected while driving to train the hand posture classifier at run-time. Once the distracted driver hand gestures are detected, the alert will be triggered by the smartwatch. In the following sections, we will describe the design of SafeWatch components in details.

3.4.2 Sensor Sampling

In SafeWatch, the smartphone placed in the vehicle is employed to monitor the vehicle's movement, while the smartwatches on the driver's wrists track the motion and posture of the driver's hands. To collect motion data, SafeWatch continuously samples the built-in accelerometers and gyroscopes on the smartphones and the smartwatches. The sampling rate is set to 50Hz. Each sample contains an acceleration vector $\overrightarrow{\mathbf{a}}$ and a rotation vector $\overrightarrow{\mathbf{w}}$. For one smartwatch, the motion samples are continuously transferred to the smartphone through the Bluetooth with 5Kbps rate. The classification is done in a sliding window, which contains 1 second of data and is built for every 0.5 seconds. The window size is determined based on two observations. First, it is not necessary to trigger an alert when the driver's hand is only away from the steering wheel less than $1.0s$, because the unsafe actions last more than $2.5s$ as defined by the *American Society of Safety Engineers* (ASSE) [12]. Second, if we wish to trigger the alert when the hand is away from the steering wheel for a relatively long time, e.g. $5.0s$, we wish to analyze the motion data from a considerable amount of previous windows. Thus, we select the length of the window as $1.0s$ for each $0.5s$, in order to capture the detailed motion of the driver's hand while minimizing the computing overhead.

3.4.3 Hand Movement Detection

The goal of hand movement detection is to determine whether the driver's hand is moving. Although the variance of $\overrightarrow{\mathbf{a}}$ is effective to detect whether a device is moving at a constant speed or at rest, it cannot be used for detecting whether a device is moving in a driving environment, where $\overrightarrow{\mathbf{a}}$ is interfered by the movement of the vehicle. SafeWatch addresses

this challenge by comparing $\overrightarrow{\mathbf{a}}$ from the smartwatch and $\overrightarrow{\mathbf{a}}$ from the smartphone. Since the coordinates of the smartphone and the smartwatches are not aligned, we cannot directly compare the direction of $\overrightarrow{\mathbf{a}}$ from those devices. However, an important observation is that $|\overrightarrow{\mathbf{a}}|$ from those devices should be similar if no relative movement exists between them. Based on this observation, SafeWatch determines if the driver's hand is moving by comparing $|\overrightarrow{\mathbf{a}}|$ from each device, and examining the following inequality,

$$\frac{\sum_{i=1}^{l} ||\overrightarrow{\mathbf{a}_{i,watch}}| - |\overrightarrow{\mathbf{a}_{i,phone}}||}{l} \leq \epsilon \tag{3.1}$$

where l is the length of the window. When the inequality is satisfied, SafeWatch claims that there is a relative movement between the driver's hand and the vehicle. The performance of the detector given in Eq. 3.1 depends on the choice of ϵ. Specifically, a small ϵ may degrade the classifier's robustness in the presence of noise motion signals, resulting in an increased false alarm rate. If ϵ is too large, SafeWatch may fail to recognize the movement of the hand, reducing the detection rate. We optimize the performance of the detector shown in Eq. 3.1 by choosing ϵ based on empirical measurements. Fig. 3.2 illustrates detection accuracy for different values of ϵ. According to our experiment, $\epsilon = 1.0m/s^2$ can be common settings for most cases, considering the native offset of the sensors across various devices.

According to a study by ASSE, such distracted movements usually last for longer than $2.5s$ [12]. SafeWatch checks the hand movement in L consecutive sample windows and whether these windows contains the operation of the steering wheel. If the steering wheel is detected as not operated by the method introduced in Section 3.4.8, but the hand keeps moving, an unsafe action will be detected. The performance of the detection can be adjusted by the different choice of L. The hand movement with a short time can be detected as unsafe when L is small. If L is large, SafeWatch will only detect the unsafe action when the hand movement lasts for a long time. In our design, SafeWatch provides an open interface for choosing L. We recommend this value to be chosen by the professionals of safety engineering.

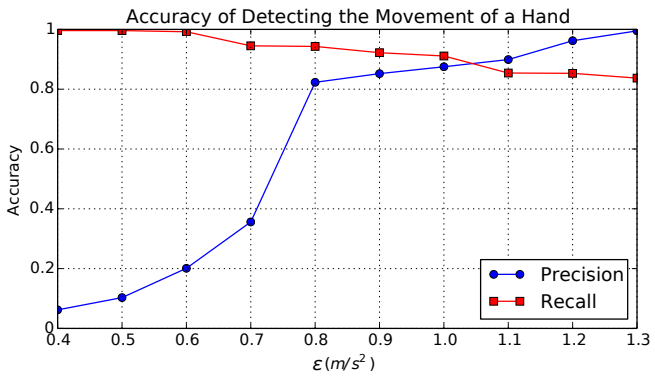


Figure 3.2: Accuracy of detecting the movement of a hand

3.4.4 Gravity Extraction

The goal of the gravity extraction module is to learn the attitudes of a device. Moreover, it separates the real acceleration of the device from the effect of the gravity. Since the measurement data $\overrightarrow{\mathbf{a}}$ from the accelerometer is always interfered by the gravity, two acceleration vectors can be obtained by decomposing $\overrightarrow{\mathbf{a}}$. The first one is a virtual acceleration that neutralizes the impact of gravity, which is denoted as $\overrightarrow{\mathbf{g}}$. The other one is the real acceleration, which is denoted as $\overrightarrow{\mathbf{a}_L}$. According to the characteristics of the gravity, we have:

$$|\overrightarrow{\mathbf{g}}| \approx 9.8, \quad \overrightarrow{\mathbf{a}} = \overrightarrow{\mathbf{g}} + \overrightarrow{\mathbf{a}_L} \tag{3.2}$$

Here, $\overrightarrow{\mathbf{g}}$ indicates the attitude of the device, and $\overrightarrow{\mathbf{a}_L}$ can be used to track the movement. Fig. 3.3(a) and Fig. 3.3(b) illustrate the components of $\overrightarrow{\mathbf{a}}$. Traditionally, $\overrightarrow{\mathbf{g}}$ can be derived by applying a low-pass filter on $\overrightarrow{\mathbf{a}}$ [41]. The idea is, while the device is moving at a constant speed or at rest, $\overrightarrow{\mathbf{a}_L}$ is close to 0, and $\overrightarrow{\mathbf{a}}$ is close to $\overrightarrow{\mathbf{g}}$. If the variance of $\overrightarrow{\mathbf{a}}$ is low and $\overrightarrow{\mathbf{a}}$ is close to $9.8m/s^2$ in a period of time, the low-pass filter is the most effective method to calculate $\overrightarrow{\mathbf{g}}$.

Due to the movement of the vehicle or the driver's action, we cannot expect the variance of $\overrightarrow{\mathbf{a}}$ is always low. When the device is moving at non-uniform speed, the variance of $\overrightarrow{\mathbf{a}}$ will be high. For example, the variance of $\overrightarrow{\mathbf{a}}$ of the smartwatch is usually high when the driver is moving his or her hand or the vehicle is accelerating. As shown in Fig. 3.4(a), the

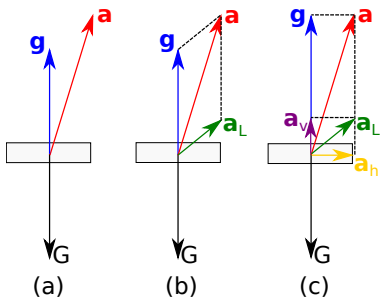


Figure 3.3: The components of the measurement from the accelerometer

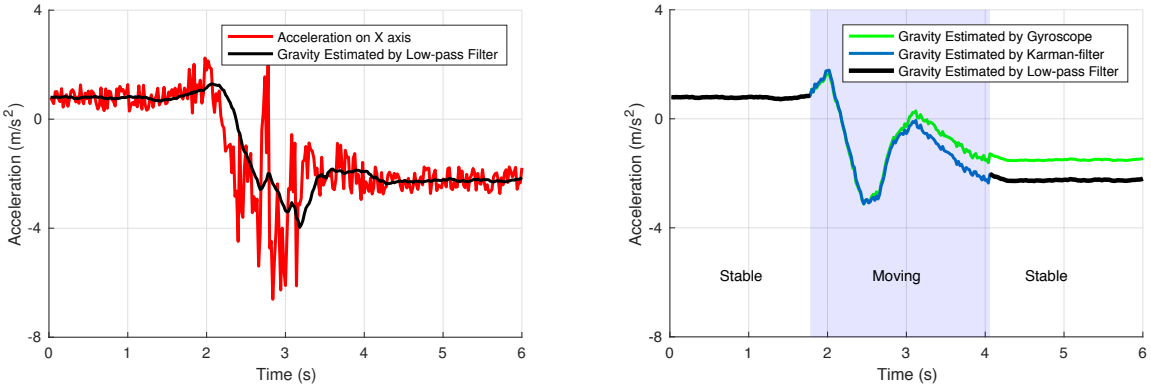


Figure 3.4: The acceleration on X-axis with hand movement

impact of gravity on X-axis estimated by the low-pass filter is interfered by the significant motion. However, by considering that the direction of gravity is only changed due to the device's rotation, we can use the rotation value $\overrightarrow{\mathbf{w}}$ measured by the gyroscope to calculate the attitude of the watch. Unfortunately, the value $\overrightarrow{\mathbf{w}}$ contains errors and bias, and the calculated gravity drifts away from the truth as shown in Fig. 3.4(b). To address the challenge, we apply *Kalman Filter* [85][13][82]. This algorithm estimates the watch's attitudes (i.e. the direction of gravity) at each moment based on all the factors, including the directions of gravity before and after the movement, $\overrightarrow{\mathbf{w}}$ by the gyroscope.

Here, we introduce how to apply Kalman Filter in our case. A typical movement can be described as three phrases, which are starting, moving, and ending. Before the movement starts, the device is at rest, which means $\overrightarrow{\mathbf{a}}$ has a low variance and $\overrightarrow{\mathbf{g}_{before}}$ can be calculated

by the low-pass filter as a known state. It is the same case after the movement ends, and $\overrightarrow{\mathbf{g}_{after}}$ can also be calculated as another known state. The states when the device is moving are hidden, but the only factor to change the direction, i.e. change the state, is rotation, which can be represented as $\overrightarrow{\mathbf{w}}$ including some errors and bias. Our goal is to estimate $\overrightarrow{\mathbf{g}}$ at each moment during this movement with maximum-likelihood. Suppose the movement contains a series of $n+1$ samples, and we label the index of the initial state (before the movement) as 0 and the final state (after the movement) as n. Here $\overrightarrow{\mathbf{g}_0}$ and $\overrightarrow{\mathbf{g}_n}$ are given by the low-pass filter. According to the concept of the rotation matrix [30], we have

$$\overrightarrow{\mathbf{g}_i} = \prod_{j=1}^{i} \mathcal{M}(\overrightarrow{\mathbf{w}_j} - \overrightarrow{\mathbf{e}_j}) \cdot \overrightarrow{\mathbf{g}_0}, \tag{3.3}$$

where $\overrightarrow{\mathbf{w}_j}$ is the rotation vector for sample j, $\mathcal{M}(\overrightarrow{\mathbf{w}})$ is the rotation matrix based on rotation vector $\overrightarrow{\mathbf{w}}$, and $\overrightarrow{\mathbf{e}}$ is the error from the gyroscope. In order to calculate the exact value $\overrightarrow{\mathbf{g}_i}$, we need prior knowledge of $\overrightarrow{\mathbf{e}_j}$ at each moment. Instead, we estimate the maximum-likelihood value of $\overrightarrow{\mathbf{g}_i}$ by calculating the expectation of $E[\overrightarrow{\mathbf{e}}]$. The calculation of $E[\overrightarrow{\mathbf{e}}]$ can be done by setting $i = n$ in the previous equation,

$$\overrightarrow{\mathbf{g}_n} = \prod_{j=1}^{n} \mathcal{M}(\overrightarrow{\mathbf{w}_j} - E[\overrightarrow{\mathbf{e}}]) \cdot \overrightarrow{\mathbf{g}_0}, \tag{3.4}$$

where $E[\overrightarrow{\mathbf{e}}]$ is the only unknown variable. The *Kalman Filter* provides a method to solve such a equation [82]. Once $E[\overrightarrow{\mathbf{e}}]$ is known, we have

$$\overrightarrow{\hat{\mathbf{g}}_i} = \prod_{j=1}^{i} \mathcal{M}(\overrightarrow{\mathbf{w}_j} - E[\overrightarrow{\mathbf{e}}]) \cdot \overrightarrow{\mathbf{g}_0}. \tag{3.5}$$

As shown in Fig. 3.4(b), the impact of gravity is calculated by combing the result from the low-pass filter and the measurement by the gyroscope. We can see that it keeps the result from the low-pass filter before and after the movement, and it estimates how the sensed gravity changes along with the device's rotation.

3.4.5 Forearm Posture Detection

In a typical driving scenario, the driver's forearm postures will remain unchanged when the driver's hands are holding the steering wheel. As shown in Fig. 3.5, for a typical driving posture, both the driver's arms will be stretching forward, and the elbows will be naturally lying down, yielding a unique pattern when measuring the posture of smartwatch wearing on the driver's wrist. When a hand is taken off from the steering wheel, the posture of the forearms is difficult to predict. For example, the hand can hold something (mobile phone, food, etc.) or stay on the leg. However, those postures are rarely the same as the posture as when the hands are holding the steering wheel. Thus, in order to detect whether the hands are on the steering wheel, the posture of the driver's forearms can be used as a feature.

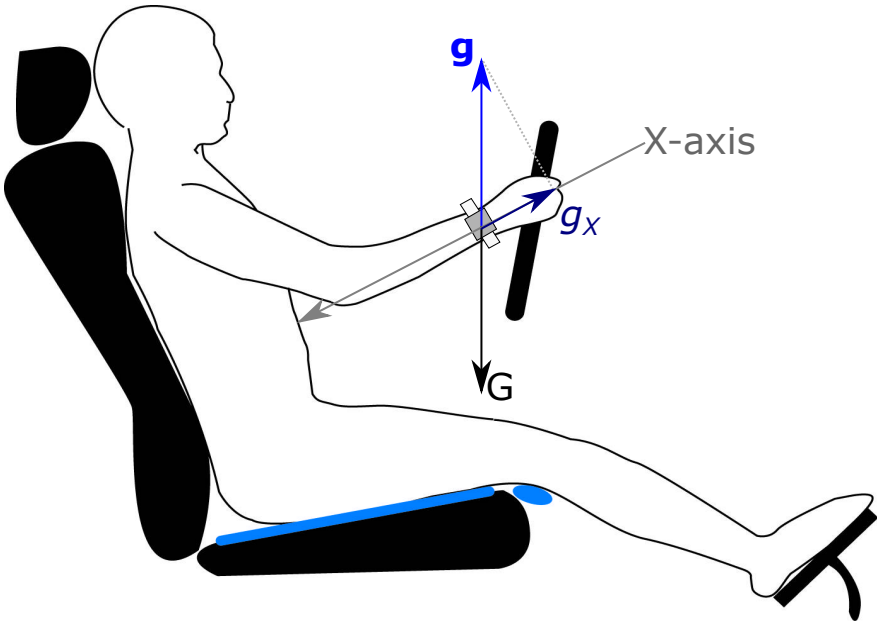


Figure 3.5: A typical driving posture

According to the coordinate system of the motion sensors, as shown in Fig. 3.6, the attitude of X-axis on the smartwatch is always parallel to the forearm. Thus, the average value on X-axis of $\overrightarrow{\mathbf{g}_{watch}}$ in a window, which is denoted as $g_{X,watch}$, can be used to characterize the vertical posture of the forearm. Specifically, different forearm postures will yield different patterns of $g_{X,watch}$ measurements. For the right hand, the value decreases when the hand

Figure 3.6: The 3-axis coordinate system for the accelerometers and gyroscopes on the smartwatches

moves up. For example, as shown in Fig. 3.7, when the driver is holding the steering wheel, $g_{X,watch}$ of the smartwatch on the right hand will be around $-3.5m/s^2$. When the driver is holding something in hand, the forearm rises up, and the $g_{X,watch}$ will be around $-8.2m/s^2$. If the driver puts the hand on the leg, the forearm falls downward, and the $g_{X,watch}$ will be around $2.5m/s^2$. To further validate this assumption, we record the driving behaviors of 2 subjects using video cameras and log the trace of $g_{X,watch}$ measured by the smartwatches worn on the driver's wrist. Fig. 3.8 shows the *Probability Density Function* (PDF) of $g_{X,watch}$. It can be seen that $g_{X,watch}$ distributes in a narrow space when the hand is holding the steering wheel. Furthermore, if we can confirm that the hand is on the steering wheel, the value of $g_{X,watch}$, i.e. the forearm posture can be used to identify whether the hand is on the upper or lower part of the steering wheel. Specifically, if $g_{X,watch}$ is greater than the average of the training data set that the hand is on the steering wheel, it means the right hand is on the upper part of the steering wheel or the left hand is on the lower part of the steering wheel.

3.4.6 Vibration-based Hand Position Detection

When the engine of a vehicle is on, a continuous vibration along vertical direction will be generated and radiated into the cabin, and then be sensed at the steering wheel and the seat [8]. If the hand is holding the steering wheel, the vibration will be conducted to the driver's wrist, resulting in an increased magnitude of vibration sensed by the smartwatch.

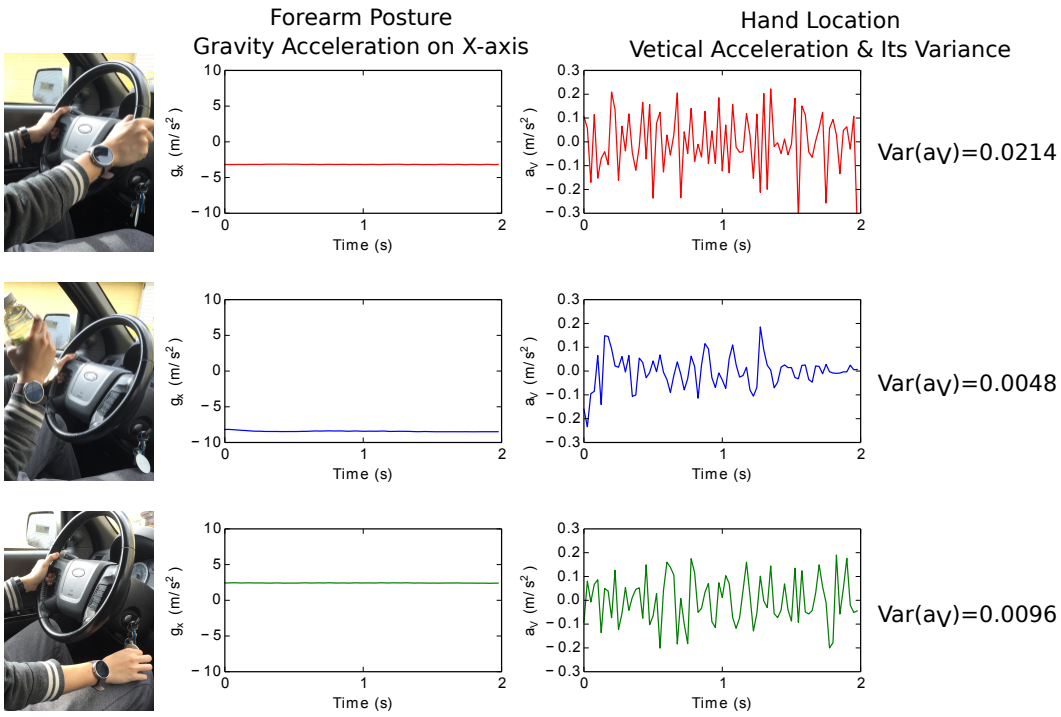


Figure 3.7: Three typical actions during driving

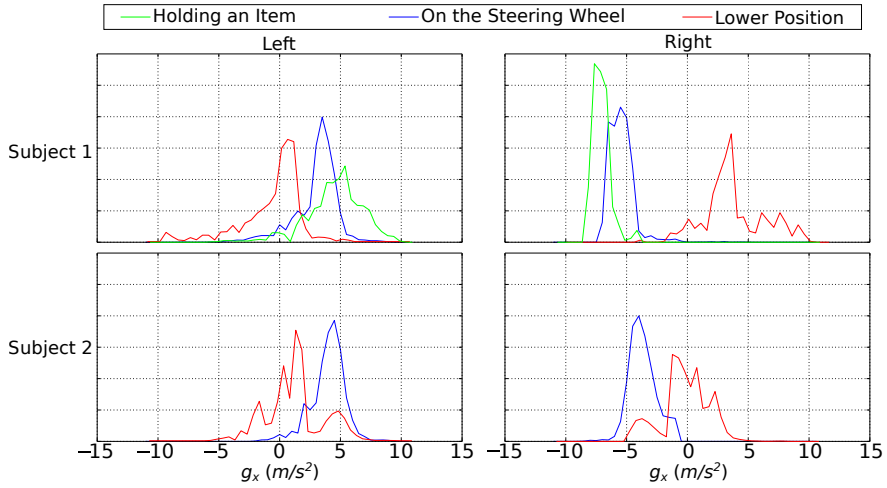


Figure 3.8: Normalized PDF of $g_{X,watch}$ for three different positions of the hand

Otherwise, vibration conducted to the driver's wrist will be much weaker as shown in Fig. 3.8. The reason is the vibration is significant attenuated when it is transmitted from the seat via the driver's body. Based on the above observation, SafeWatch leverages the vibration sensed by the smartwatch's accelerometer as another feature for inferring if the driver's hand is on/off the steering wheel when the hand is still.

Specifically, SafeWatch extracts the vibration-based feature as follows. It derives the vertical signal component by computing,

$$a_V = \frac{\overrightarrow{\mathbf{g}} \cdot \overrightarrow{\mathbf{a}_L}}{|\overrightarrow{\mathbf{g}}|}. \tag{3.6}$$

Then, SafeWatch measures the magnitude of vibration by computing the variance of a_V. Fig. 3.9 illustrate the vibration-based feature extraction algorithm based on two real cases. Generally, when the hand is on the steering wheel, $Var(a_{V,watch})$ is slightly larger, as shown in Fig. 3.9.

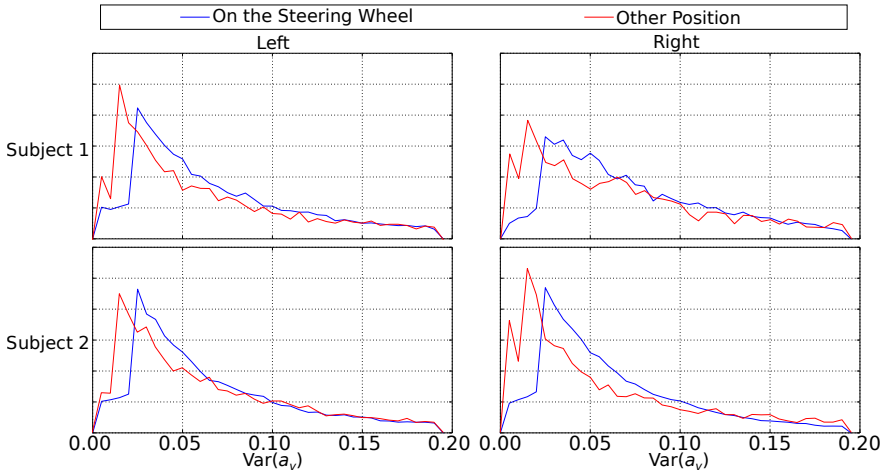


Figure 3.9: Normalized PDF of $Var(a_{V,watch})$ for subject 1 and 2.

In some cases, the vibration is mainly caused by the movement of the vehicle and the road condition, $Var(a_{V,watch})$ will be much larger than usual, and its distribution cannot provide evidence for detecting the hand position. A key challenge in realizing this idea is to address the interfering motion signals introduced by the movement of vehicle, which is usually orders of magnitude stronger than the vibration signal of interest. If the hand holds the steering wheel firmly, the vertical movement on the wrist will be similar to the vehicle's. SafeWatch estimates the interference caused by the vehicle's movement along the vertical direction, by measuring a_V observed on the smartphone. Because the smartphone is placed in the vehicle, its measurement of a_V characterizes the vertical movement of the vehicle.

SafeWatch then mitigates interference by computing,

$$R_{a_V} = \frac{Var(a_{V,watch})}{Var(a_{V,phone})}. \tag{3.7}$$

In this case, we have $R_{a_V} \approx 1$ as shown in Fig. 3.10, which means the magnitudes of the vibrations captured by the smartphone and the smartwatch are close. For the detection of the hand position, $Var(a_{V,watch})$ and R_{a_V} are both important features.

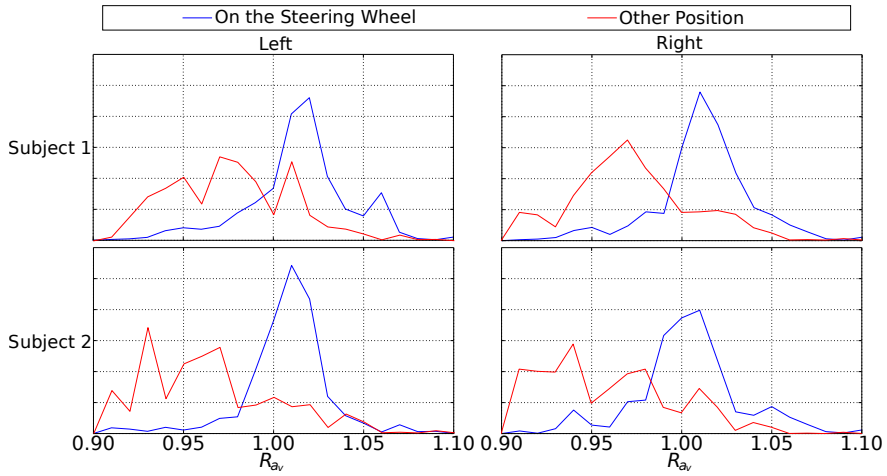


Figure 3.10: Normalized PDF of R_{a_V} for subject 1 and 2.

3.4.7 Hand Position Classifier

The classifier determines if the driver's hand is on/off the steering wheel based on forearm posture and the vibration-based features. Specifically, it builds an vector $\overrightarrow{\mathbf{c}} =< g_{X,watch}, Var(a_{V,watch}), R_{a_V} >$ containing three features from previous modules. A training data set is necessary for this classifier. In our design, the training data set is initialized once per driver and per vehicle, by asking the driver to start the engine and move his/her hands from the top to the bottom of the steering wheel. As shown in Fig. 3.11, the feature vector $\overrightarrow{\mathbf{c}}$ has a narrow distribution when the hand is on the steering wheel, but it has a board distribution when the hand is away from the steering wheel. Because it is unrealistic to collect the data samples when the hand is away from the steering wheel, we build a training data set that only includes the motion samples when the hand is on the steering wheel.

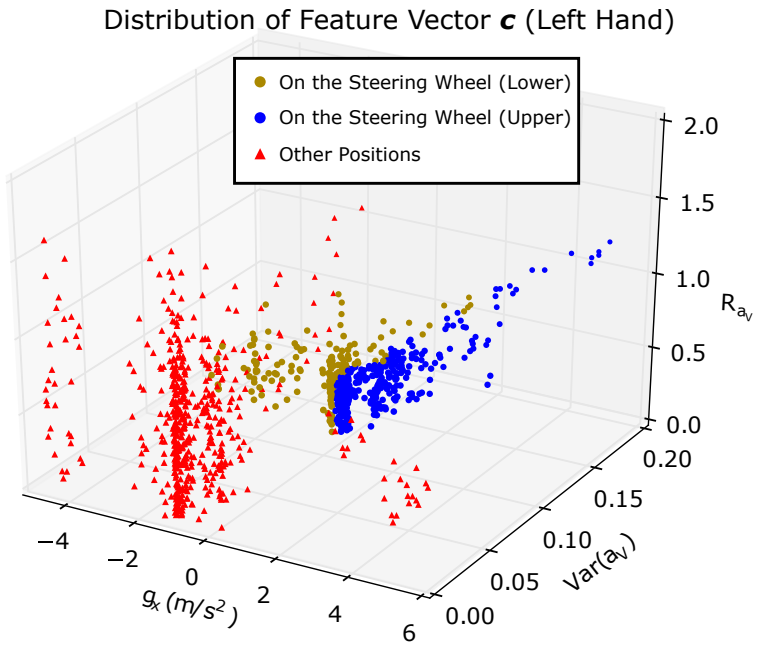


Figure 3.11: Distribution of feature vector **c** while driving

For the test data, the classifier computes the probability of the test data fitting into the distribution of the training data set by the statistical hypothesis testing (e.g. *Welch's t-test* [117]). Specifically, for the test data samples collected within time T, SafeWatch compares the distribution of the test data and the training data set. A high similarity means a high probability that the test data belongs to the training data, i.e. hand is on the steering wheel. Then, by comparing the average of $g_{X,watch}$ of the test data and the training data, the hand position on the upper or lower part of the steering wheel can be deduced according to the introduction in Section 3.4.5.

3.4.8 Vehicle Turning Detection and Auto Calibration

A critical challenge for SafeWatch is it requires frequent training, even during one driving trip. The driver may switch the posture, and the smartwatch can be moved to various positions on the wrist. In order to maintain the high accuracy of the classifier, SafeWatch must continually adapt itself into the most recent status. An important observation is the hand must hold the steering wheel in order to turn the vehicle. Thus, the ground truth

feedback can be obtained around the moment of turning. In order to detect the vehicle's turning, SafeWatch analyzes the data collected by the gyroscope on the smartphone [25]. Specifically, the rotation around the direction of the gravity indicates the angle of the vehicle's turning. The samples right before and after the turning is automatically labeled as "hand is on the steering wheel" and imported into the training data set.

3.5 Evaluation

We evaluate SafeWatch with 75 real driving trips collected from six subjects. Each subject is provided with two smartwatches and one smartphone, all installed with apps that run in the background to record the raw data of motion sensors. In order to obtain the ground truth of the driver's hand gestures, we collect location data using the GPS of the smartphone and then record the driver's hand gestures through video-recording. The details of collected driving trips are summarized in Table 3.1.

Table 3.1: Information of the experiment

	Watch (Left)	Watch (Right)	Phone	Trips	Data
1	Moto 360 2	Moto 360	Moto G	8	132min
2	Moto 360 2	Moto 360 2	Moto G 2	10	125min
3	Sony Smartwatch 3	Sony Smartwatch 3	Moto G 2	4	86min
4	Moto 360 2	Sony Smartwatch 3	Moto G 2	12	124min
5	Moto 360 2	Moto 360 2	Moto G 2	12	118min
6	Moto 360 2	Moto 360 2	Moto G 2	19	179min

3.5.1 Hand Movement Detection

As we discussed in Section 3.4.3, the performance of detecting unsafe hand movement depends on the choice of L. Fig. 3.12 shows the accuracy of distraction detection under different values of L. It can be seen that if the hand movement lasts for more than $2.5s$, the recall is over 97.1% and the precision is over 91.0%. The detection accuracy further improves when the hand moves for a longer period of time.

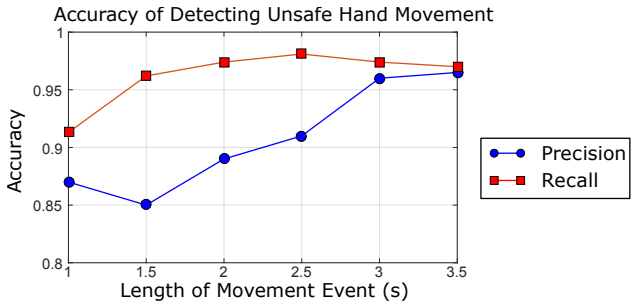


Figure 3.12: Accuracy of detecting unsafe hand movement

3.5.2 Hand Position Classification

3.5.2.1 Classification Accuracy

When the driver is still, SafeWatch employs the hand position classifier introduced in Section 3.4.7 to infer if the driver's hand is on/off the steering wheel. Specifically, SafeWatch classifies hand position by first training a Gaussian model using the approach introduced in Section 3.4.8, and then applying *Welch's t-test* to check whether the test data collected while driving fits into the trained model. Similar to hand movement detection, the hand posture classification is performed based on sample windows of $0.5s$. SafeWatch reports a distraction event if the driver's hand is detected as off the steering wheel for T consecutive sample windows.

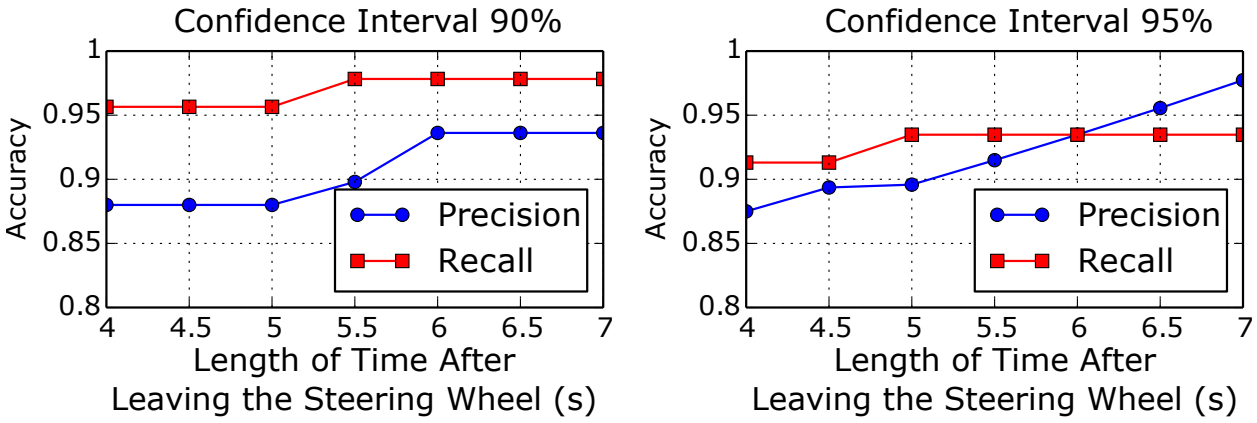


Figure 3.13: Accuracy of the classifier and confusion matrix (Confidence interval = 90%, T = 6s).

Fig. 3.13(a)(b) shows the accuracy of detection for each value of T. Here, the precision is calculated by "number of successfully detected dangerous actions/number of all detected dangerous actions", and the recall is calculated by "number of successfully detected dangerous actions/number of dangerous actions in ground truth". As shown in the figure, for T larger than $6s$, both of the precision and recall are higher than 90%. When the hand is on the steering wheel, the upper or lower position can be detected with high accuracy as shown in Fig. 3.13(c). Based on our study, the error appears when the training data set drifts away, where the average value of $g_{X,watch}$ cannot represents the center of the steering wheel accurately. This can be fixed by our auto-calibration as introduced in the next section.

3.5.2.2 Auto-calibration

If the posture of the driver changes during a driving trip, the training data set should be updated, in order to adapt to the new environment. As we introduced in Section 3.4.8, this happens when the driver is manipulating the steering wheel. To keep the size of the training data set, we replace some of the oldest data to the new data. The amount of the replacement is defined as the update rate. Here, we evaluate how the errors can be reduced with the auto-calibration. The parameters of the classifier are set to 90% confidence interval and $6.0s$ sensitivity.

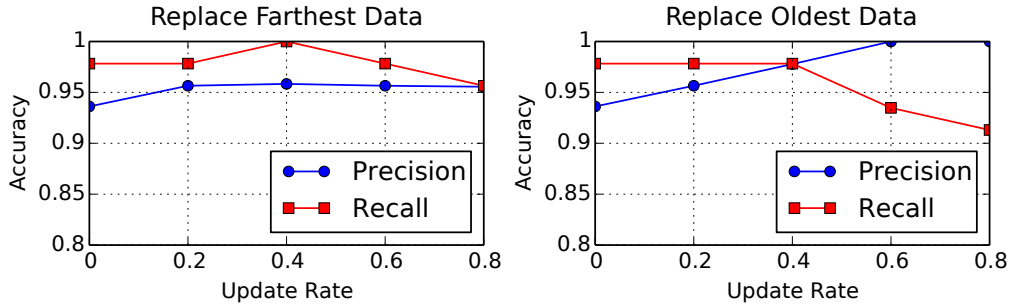


Figure 3.14: Auto-calibrated training data set and corresponded accuracy

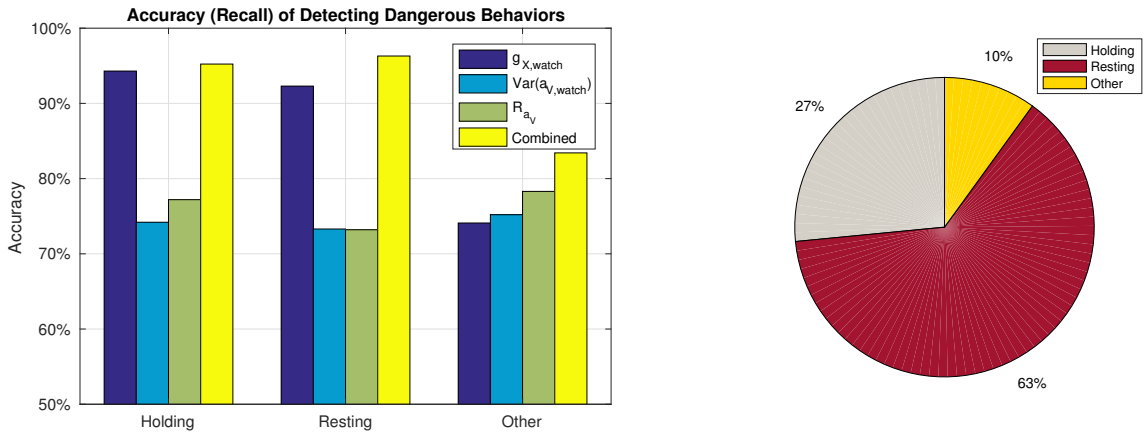
The performance of SafeWatch with auto-calibration is shown in Fig. 3.14. In Fig. 3.14(a), we use the $g_{X,watch}$ to demonstrate how the training data set evolves when 40%

of the training data set are updated after each turn (update rate is 0.4). Specifically, the coverage expands, and the average value of $g_{X,watch}$ is closer to the ground truth of the center of the steering wheel. This means the training data set is more accurate after each update. With proper setup, the overall recall and precision rate can be both higher than 97%. In this case, the errors can be reduced to less than 1.5 per hour. However, less unsafe actions are detected if the selected update rate is too large, resulting in lower accuracy in the recall. The reason is a large amount of the new data brings bias to the training data set, which is initialized when the driver's hand moves from the top to the bottom of the steering wheel completely and uniformly.

3.5.2.3 Contribution of Features

In our design, the positions of the hands are detected based on the forearm's posture and the vibration. In this section, we run the classifier using every single feature individually, in order to understand the contribution of each feature to the detected result. We carefully checked each moment in the collected data. When the subject's hand is away from the steering wheel, we labeled the subject's behavior as three events. The "holding" event means the hand is holding something such like a bottle or a phone. The "resting" event means the hand is resting on the leg or the seat. The "other" event means the hand is handling other secondary tasks that are not included in "holding" or "resting", such as operating the on-vehicle music player or touching the GPS navigation system on the windshield.

Fig. 3.15(a) shows the accuracy of detection when those dangerous events happen. We can see that the forearm posture is the most effective feature for the classifier. Moreover, it works best in identifying "resting" and "holding" events. The reason is that the forearm is less likely to be pointing to the steering wheel when the driver is holding something or resting the hand on the leg. The vibrations on the smartwatch can provide important information about whether the hand is touching the inner parts of the vehicle, thus it is supposed to be effective in every case. However, the classifier has relatively low accuracy when it only



(a) (b)

Figure 3.15: Accuracy of detecting different type of dangerous behaviors

uses the features from the vibrations, because the vibration is related to many factors and it is not reliable all the time. If the classifier combines all the available features, it can correctly detect over 90% of dangerous behaviors in "holding" and "resting" events. For "other " events, SafeWatch only fails in rare cases. For example, SafeWatch believes the driver's hand is on the steering wheel when the driver's hand stays on the central console for a long time. In this case, the forearm's posture sensed by the impact of gravity on the smartwatch could be similar to a driving posture, and the vibration on the central console may be similar to the vibration on the steering wheel. However, as shown in Fig. 3.15(b), "other" events rarely happen during a driving trip, thus SafeWatch is still able to maintain high overall accuracy.

3.5.3 Micro-scale Driving Behavior Analysis

We next conduct a micro-scale driving behavior analysis using the result from SafeWatch. It discovers several driving habits of the subjects. For example, in which case the driver tends to move hands away from the steering wheel, which hand is more likely to move away from the steering wheel, what is the method the driver uses for steering, how the road condition affects the driving habits, etc.

Fig. 3.17 shows the detection results along with the ground truth during a 20-minute driving trip, which covers the route on a city road and a highway as shown in Fig. 3.18. We can observe that, in the highway trace, the frequency of the subject's hand movements significantly reduces, mainly because there are fewer curves and turns on the road. In this case, the driver's right hand moves less frequently than his left hand. Meanwhile, the right hand tends to stay away from the steering wheel, which implies that the right hand is relaxed. Another observation is that the positions of the hands keep changing throughout the driving trip. For example, both hands are on the steering wheel at the beginning of driving, but they tend to move away from the steering wheel more frequently after the first 3-minute.

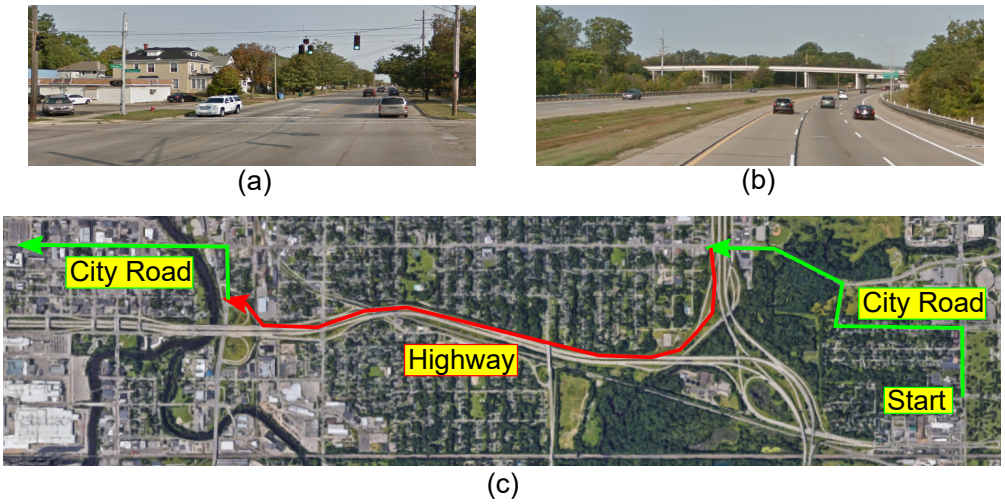


Figure 3.16: Log of the 20-minute driving trip

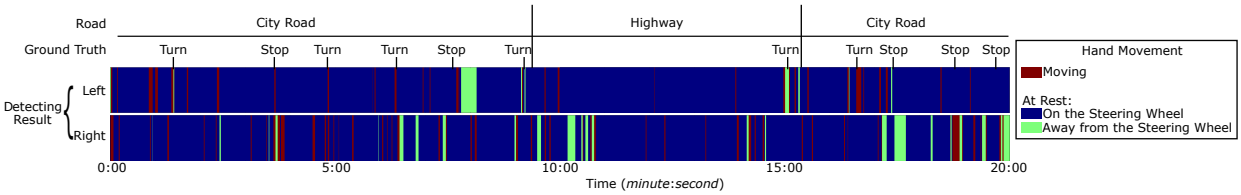


Figure 3.17: Hand movement and position of the 20-minute driving trip

In order to study this in detail, we analyze the outputs of our classifier for different driving trips of other subjects. Specifically, we check how the roads and the driving time impact driving behavior. Fig. 3.16 shows the driving behaviors of four subjects on different

roads. On the city road, the drivers are more likely to hold the steering wheel firmly. The duration of each hand's movement is short, but the frequency is high. The reason is that drivers have to adjust the steering wheel more frequently on city road that has more turnings. We can also observe that the dominant hand moves more frequently than the other hand because drivers tend to use their dominant hands to operate the steering wheel. SafeWatch also observes that the subjects we studied have different driving habits on the highway. For example, Subject 1 and 4 tend to frequently put their non-dominant hands on the leg. Specifically, their dominant hands always hold the top part of the steering wheel, in order to turn it conveniently towards any direction only with one hand. For most of the time, the dominant hand is the only hand on the steering wheel, and it rarely moves away from it. Subject 2 and 3 usually use both hands to hold the steering wheel at 9 o'clock and 3 o'clock positions. On the highway, they tend to relax the dominant hand and move it away from the steering wheel, then hold the steering wheel only with the non-dominant hand.

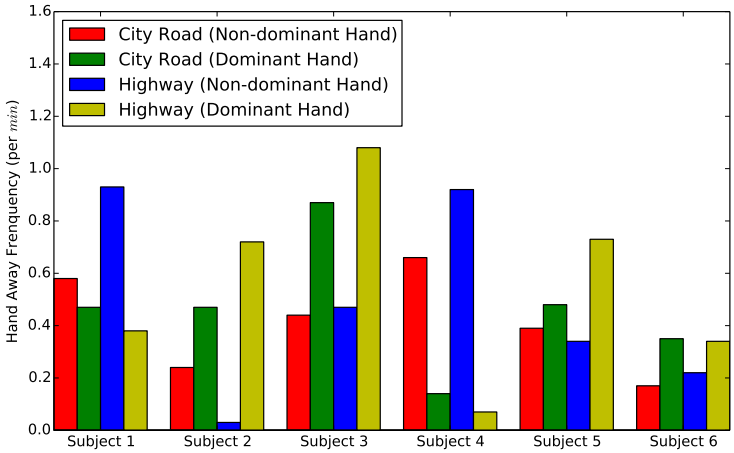


Figure 3.18: Driving behaviors and type of roads

Fig. 3.19 shows the driving behaviors of the subjects along with the driving time. At the first minute of driving, the vehicle usually is being moved out of the parking lot. A series of jobs are completed during this time, such as shifting the gear, adjusting the A/C, playing the music, etc. The hands stay on the steering wheel at most of the time during the 2nd

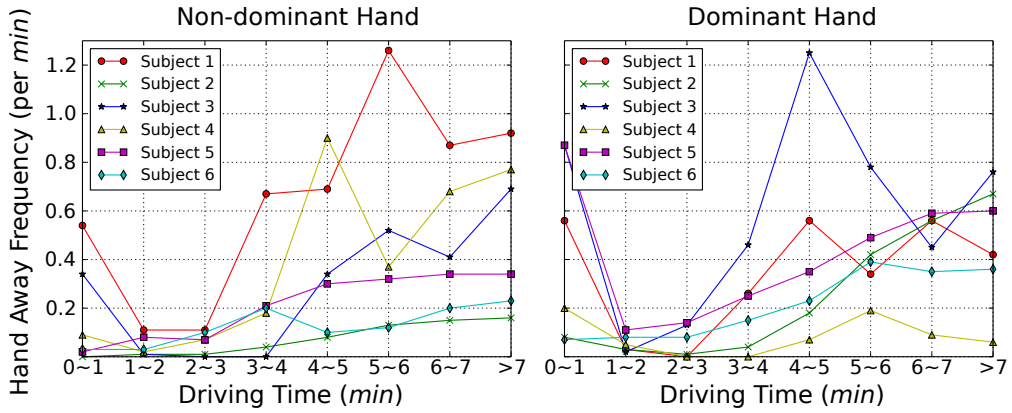


Figure 3.19: Driving behaviors and driving time

to 3rd minute. We assume that the driver just enters in a good status, and the alertness is very high at this moment. However, after 3 minutes, the hands begin to move away from the steering wheel, which corresponds to the driver's lower alertness. Another study shows the same observation with the *electroencephalogram* (EEG) monitoring [92].

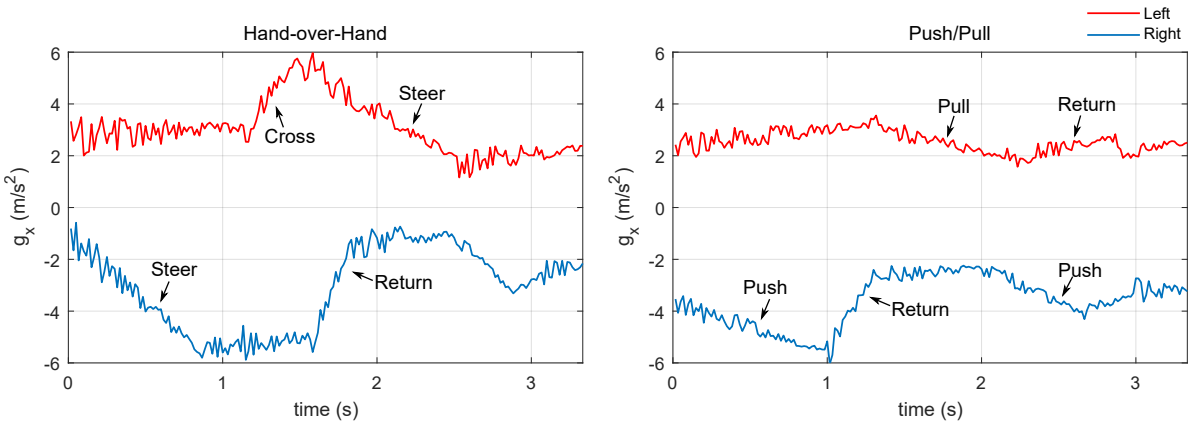


Figure 3.20: Example of left turn

The result of SafeWatch also shows the preferred steering methods, like hand-over-hand or push/pull techniques [114]. Fig. 3.20 shows the example of the output of SafeWatch with these two steering methods. As we mentioned in Section 3.4.5, the increasing value of $g_{X,watch}$ means raising the left hand or lowering the right hand, and vice versa. Based on this observation, we can see that the hand-over-hand steering method includes a "cross" step, where the left hand moves up after the right hand reaches the highest position. This

is different from the push/pull steering method, which does not involve in a cross of two hands at the upper part of the steering wheel. This fact shows the upper/lower position of the hand is a clue to classify the steering method. Another observation is, in push/pull methods, two hands move up or down more simultaneously.

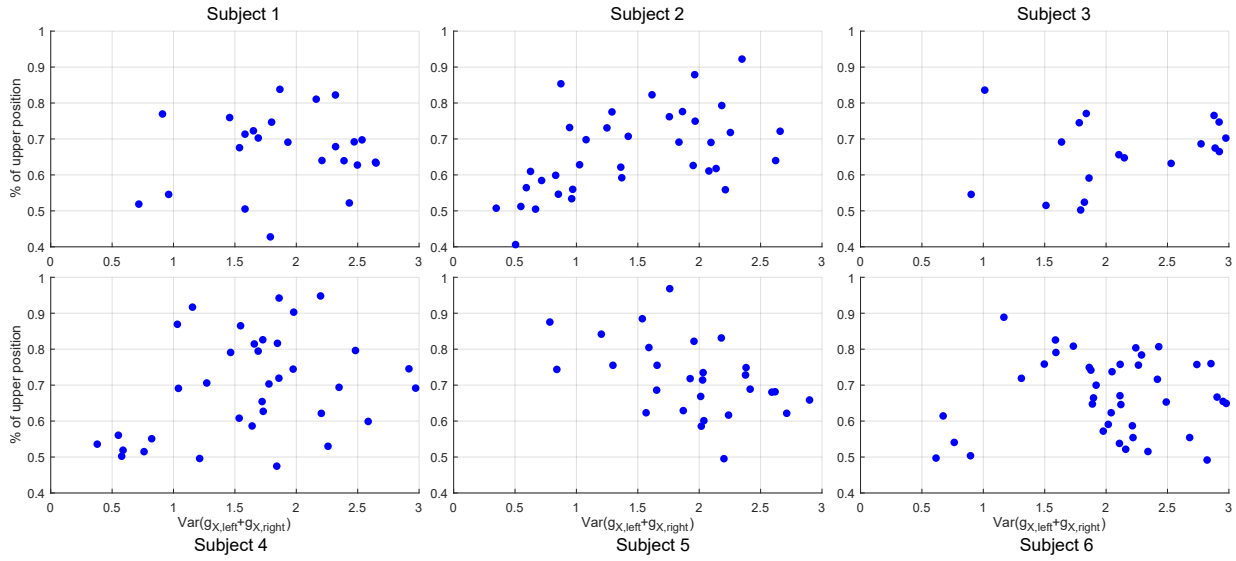


Figure 3.21: Distributions of samples of turning

Because the driver's behavior can be affected by many factors, it is difficult to clearly recognize each steering action as a technique of steering method by reviewing the video-recorded ground truth. To avoid bias and further investigate this, we define two values to describe the hand movement and use the distribution of these values to show the driver's preference. For all the samples from a complete turn, first, we calculate the percentage of the samples when the hand is at upper position; second, we calculate the variance of $(g_{X,left}+g_{X,right})$ from two smartwatches, which corresponds to the similarity of the forearm posture on both side at each moment. Fig. 3.21 shows the distribution of these two values for each subject. Though the distributions are discrete, which means the driver's micro behavior is highly unpredictable, we can still see that subject 2 has an overall lower hand position and higher similarity between the movements of two hands. We have confirmed that subject 2 performs relatively more push/pull steering method by reviewing the video recordings. We

also have found that all the subjects prefer the hand-over-hand method, especially subject 5. This proves that the result of SafeWatch can reveal some details of the driving habits and it can potentially serve for monitoring and training of drivers.

3.5.4 Feedback & Discussion

As we mention in the previous subsection, the collected ground truth and the detected result show most of the subjects' driving habits, such like the favorite position of the hand on the steering wheel and preferred steering methods. We summarized the driving behaviors and sent these to the subjects as the feedback of our experiment. It turns out that 5 out of 6 of the subjects were not aware of some dangerous actions or habits before we provided the recorded video clips to them. This fact emphasizes the importance of designing such a system because of the low self-awareness of the drivers. In addition, according to [114], the hand-over-hand steering method is not recommended in recent years, because of the chance to be hurt by airbags. However, the ground truth of our experiment and the result of SafeWatch both show that the hand-over-hand steering method is still popular among the subjects.

3.6 Conclusion of Study

SafeWatch monitors mid-level activities, like where the hand stays in the vehicle. It satisfies the **Three Rules**. Specifically, SafeWatch

- **is sensitive to context**. The classifier takes a set of the most recent time frames as the input rather than a single time frame. The state of the classifier is constantly switched between two states – hand is moving or still. The alert will be triggered by different rules according to the current state.
- **is adaptive to dynamic condition**. The training data set keeps updated. Based on the fact that the hand must be on the steering wheel while operating the steering

wheel, SafeWatch can rebuild the training data set for each turn of the vehicle.

- **processes readable features**. The major features used in SafeWatch are forearm posture and vehicle's vibration. Those numbers are readable. In this case, it is convenient to track the reason of error of SafeWatch or improve it.

CHAPTER 4

IN-VEHICLE REAL-TIME ATTITUDE AND MOTION TRACKING

This chapter introduces *Real-time Attitude and Motion Tracking* (RAMT) of the mobile devices in a moving vehicle. This is a system that aligns the coordinate systems of the mobile devices to the vehicle, thus it enables the devices to know their attitude and position in the real time. This chapter is adapted from a publication [15]. The author of the dissertation is the first author of the original work. "We" in this chapter refers to the author of the original publication. This work contains the App design on Android devices. The author recruited all the subjects, then collected and processed the data and the ground truth.

Monitoring the attitude and motion of a mobile device is a low-level monitoring. It describes the current state of a device with numbers. Specifically, it can represent the directions of the X, Y, Z axis of a device's IMU in the vehicle's coordinate system. Furthermore, the user's action can be tracked in detail. This work provides valuable data for further processing like gesture recognition.

4.1 Background

Recent years have witnessed the emergence of a class of new in-vehicle technologies for improving driving safety and experience. For instance, the smart steering wheel proposed in [68] can detect dangerous driving behaviors and warn the driver. The systems proposed in [106] [83] employ gesture control and auditory-only displays to avoid visual distraction. By installing additional equipment, these systems allow the driver to focus on driving without being distracted or handling secondary tasks. Such technologies can effectively improve driving safety since the major causes of road accidents are the distraction and secondary tasks, such as texting or operating the in-vehicle infotainment [23] [66] [59]. However, the aforementioned systems are based on proprietary technologies, presenting barriers to wide adaptation. Recently several in-vehicle systems based on off-the-shelf mobile and wearable

devices have been developed to improve the driver's awareness. Specifically, smartwatches can be used to detect the driver's action based on *inertial measurement units* (IMUs) and monitor the driving performance by recognizing possible second tasks [60] [14].

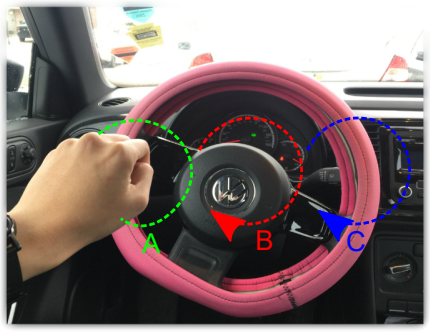


Figure 4.1: The problem of horizontal heading

However, these systems often have poor performance due to the significant dynamics of moving vehicles. In particular, the motion of the smartwatch worn by the driver is induced by the movement of both the driver's hand and the vehicle. As a result, it is challenging to use the motion sensors (accelerometer and gyroscope) of the smartwatch to accurately track the hand motion or recognize the gesture in real time. A typical difficulty of motion-based tracking is the estimation of horizontal heading [40]. As shown in Fig. 4.1, those gestures cannot be distinguished based on motion signals without prior knowledge of the horizontal heading of the smartwatch, i.e. an alignment between the smartwatch and the driver or the vehicle. A possible solution to address this issue is leveraging GPS and compass [46] to compute the vehicle's moving direction and the device's attitude relative to the earth, which can then be combined to derive the device's attitude in the vehicle. However, the GPS signal is not always available or reliable. Moreover, the low sampling rate of GPS (around 1Hz) makes it impractical to track hand gestures in real time. The compass often fails to achieve accurate measurement because of the magnetic interference from the vehicle or other electronic devices [56]. Instead, most existing solutions still employ additional equipment like cameras and infrared distance sensors to capture the image and calculate the gesture in

a 3-D space [34] [9] [52] [71] [95] to capture the real-time position of the wearable.

4.2 Related Work

Recently, a number of proprietary in-vehicle technologies have been deployed to monitor driver's performance, record the hand motion, and improve driving safety. For instance, drivers' level of attention can be monitored by measuring the vehicle's acceleration and speed [61] [26] [51] [54], driver's brainwave [20], heart rate [76] [19], and the holding force on the steering wheel [73]. The hand gestures can be classified based on the video captured by cameras [9] [52] [71] [95]. The systems proposed by [112] [106] use capacitive sensors to track the hand motion. By introducing image-based and depth-based sensors like Kinect, the gestures can be classified with a high accuracy based on the trajectory [115] [36]. However, these systems are based on proprietary technologies and incur significant costs of adoption.

Several studies have been focused on gesture recognition using mobile devices in static environments. For instance, the gesture of arm can be identified by a smartwatch by applying a particle filter on the motion signals [105]. Deep learning algorithms are developed to utilize motion sensors to control games [43], recognize hand gestures [18] [99], and analyze performance of physical activities [18] [58]. However, in these works, the devices still fails to estimate the horizontal heading. They rely on an initial coordinate system alignment between the device and the user or the earth. They also assume that the motion of mobile devices is solely induced by users.

With the high penetration of mobile devices, several systems utilize smartwatches and smartphones to detect whether the driver's hands are on the steering wheel and classify the secondary tasks [60] [14]. However, these systems only detect the shape of the gesture but cannot track hand motion in the 3D space. Several systems can sense the movement of the vehicle, including speed, braking, acceleration, and turning of the vehicle [77] [120] and infer the driving habits [79]. However, to our best knowledge, there does not exist an approach to align the coordinate systems of mobile devices and the vehicle solely based on motion

sensors, which is a key enabling technology to precisely track hand motion and recognize hand gestures.

For the gesture recognition based on motion data, some existing methods define the segments of gestures, then recognize a single whole gesture as a combination of the segments using the Hidden Markov Model [34] [32]. Furthermore, because the gesture is related to the samples in the time-series, Dynamic Time Wrapping helps reducing the errors of recognizing the gestures with various sizes and duration [24] [78]. RAMT can improve the precision of the gesture recognition by providing fine-grained tracking of the attitude of the wearable. In this work, we demonstrate an application with a deep learning model on mobile phone, which shows a hand gesture can be recognized with both shape and position based on the output of RAMT.

4.3 Requirements and Challenges

The primary goal of RAMT is to let the mobile device recognize and keep track of the vehicle's coordinate system. As shown in Fig. 4.2, the devices must learn the vehicle's forward, right, and up directions, and represent these directions under the devices' own coordinate systems as unit vectors $\boldsymbol{r}$, $\boldsymbol{f}$, and $\boldsymbol{u}$. Once these unit vectors are determined, we can transform any vector $\boldsymbol{a}$ under the original coordinate system of a mobile device into a vector $\boldsymbol{a'}$ under the vehicle's coordinate system, by

$$\boldsymbol{a'} =< \textsc{Project}(\boldsymbol{a}, \boldsymbol{r}), \textsc{Project}(\boldsymbol{a}, \boldsymbol{f}), \textsc{Project}(\boldsymbol{a}, \boldsymbol{u}) > . \tag{4.1}$$

Specifically, the requirements of our design are, 1) the mobile device in the moving vehicle should learn its attitude relative to the vehicle; 2) the mobile device should be able to distinguish the vehicle's motion and the user's action from the collected motion signals; 3) the user's hand gesture in the vehicle should be effectively and accurately recognized based on tracking.

However, several challenges must be addressed. First, without prior knowledge, the

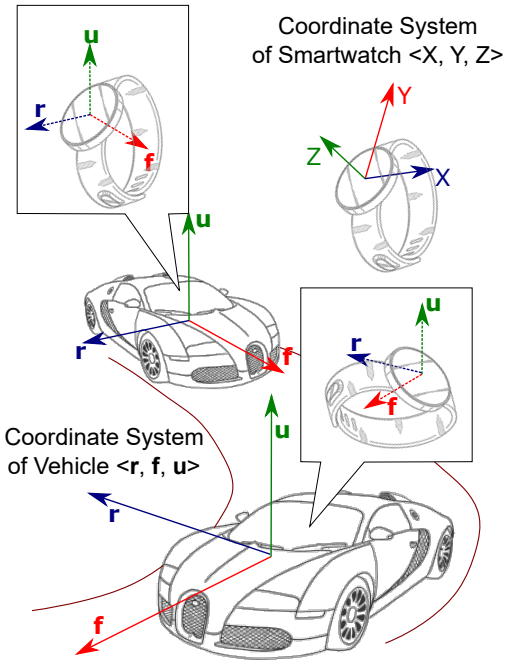


Figure 4.2: The goal of RAMT

vehicle's motion and the user's action are very difficult to be distinguished by the wearables. The motion sensors can sense either motion signal but they only output a combination of them. Second, it is difficult to learn the vehicle's coordinate system solely based on the motion signal, because it does not provide a concrete reference like location and orientation. Third, the hand motion tracking and the coordinate system alignment must be performed with a high level of accuracy and in real time during a driving trip. This requires processing high-dimension motion signals on resource-constrained mobile devices.

4.4 System Design

4.4.1 System Overview

RAMT targets to align the coordinate system of each mobile device to that of the vehicle. To learn the attitude and track the motion of the mobile devices, e.g. a smartphone and a smartwatch, we process and analyze the fused motion signals from both devices. The smartphone, which is not being used while driving, is able to pick up the vehicle's accelerating,

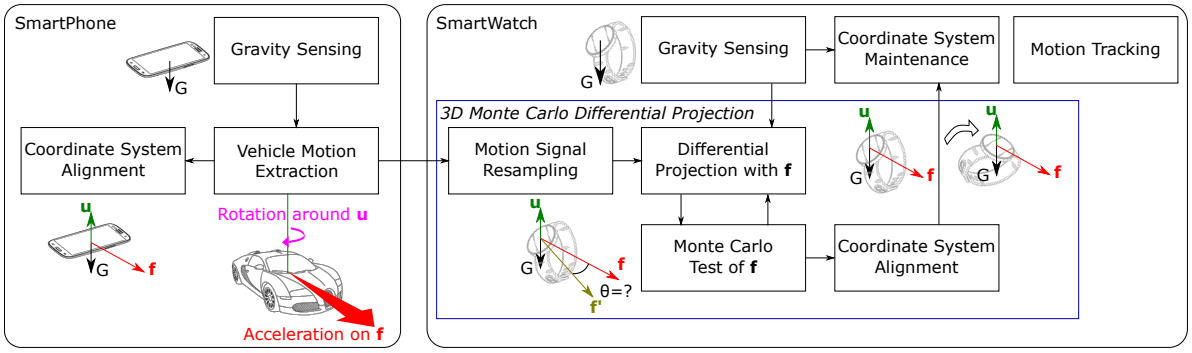


Figure 4.3: The pipeline of RAMT.

braking and turning. On the other side, the motion data collected by the smartwatch contains not only the vehicle's motion but also the driver's hand motion. Our key idea is that the smartwatch receives the information about the vehicle's motion from the smartphone via the Bluetooth, and uses this information to learn the attitude of itself, then extracts the hand motion from the mixed motion signal.

Fig. 4.3 illustrates RAMT's pipeline. Each device calculates the up direction of the vehicle individually; the smartphone senses the vehicle's movement and calculates the forward direction, then sends this information to the smartwatch via the Bluetooth; the smartwatch calculates the vehicle's forward direction based on the received information and recognizes its attitude in the vehicle. According to the concept of the orthogonal coordinates, the right direction can then be calculated by the vector's cross product by the up and forward directions. Specifically, the motion sensors, i.e. accelerometers and gyroscopes on the smartphone and the smartwatch, continuously sample the motion signal. The direction of the gravity is calculated on each device by processing the collected data. This information is used to derive the up direction of the vehicle. The smartphone calculates the forward direction of the vehicle by processing the captured vehicle's motion. After the alignment of the coordinate systems between the phone and the vehicle, the vehicle's acceleration and turning at each moment are sent to the smartwatch. By resampling the received vehicle's motion, the smartwatch compares the motion data collected by itself with the vehicle's motion. Then,

we use the gradient descent method to minimize the differential of the speed between the smartwatch and the vehicle, which searches for the best alignment between the two coordinate systems. Finally, the forward and the right directions of the vehicle are determined and maintained continuously by the smartwatch. The motion tracking of the smartwatch is performed simultaneously based on its real-time attitude.

4.4.2 Gravity Sensing

A common solution to determine the attitude of a device is to sense the direction of the gravity under the device's coordinate system. Since the measurement $\boldsymbol{a}$ from the accelerometer is always interfered by the gravity, the real acceleration caused by the motion of the device, which is denoted as $\boldsymbol{a}_L$, must be calculated by subtracting the impact of gravity from $\boldsymbol{a}$. The existing studies show that the reading of accelerometer can be used to represent the gravity by applying a low-pass filter, if the device is at rest [81]. However, for the mobile devices in a dynamic environment, e.g. in a moving vehicle, the traditional approach does not work. In this work, we introduce a new approach that can achieve highly accurate gravity sensing even when the device is constantly in motion. .

Hereafter, we denote the impact of the gravity on a device as $\boldsymbol{g}$. According to the characteristics of gravity, we have:

$$|\boldsymbol{g}| \approx 9.8, \quad \boldsymbol{a} = \boldsymbol{g} + \boldsymbol{a}_L \ . \tag{4.2}$$

The value of $\boldsymbol{g}$ can be derived by applying a low-pass filter on $\boldsymbol{a}$ while the device is moving at a constant speed or at rest, i.e. when $\boldsymbol{a}$ is close to $9.8m/s^2$ in a period of time and the reading from the gyroscope is close to 0 [42]. This state is referred to as "stable". The other state "moving" represents the condition in which the low-pass filter cannot yield an accurate value of $\boldsymbol{g}$, and this happens in the following two cases,

1) The motion data from the gyroscope shows that the device is rotating. For the smartphone, it may be picked up but not move along with the vehicle. This is a common

case for smartwatch since it usually rotates with the wrist of the user. This may often occur to a smartphone in a vehicle when it is picked up by the user.

2) The device is not rotating but $\boldsymbol{a}$ is not close to $9.8m/s^2$. In this case, both the smartphone and the smartwatch may pick up the vehicle's horizontal acceleration, i.e. when the vehicle is accelerating, braking, or turning.

In these cases, we must calculate $\boldsymbol{g}$ by leveraging the rotation, which is sampled by the gyroscope and denoted as $\boldsymbol{\omega}$ at each moment. Assume $\boldsymbol{g}$ at the previous sampling moment is known, which is denoted as $\boldsymbol{g}_{old}$, and the rotation sensed by the gyroscope is also known as $\boldsymbol{\omega}$ at the current moment, then we can calculate the current impact of gravity $\boldsymbol{g}_{new}$ with the following steps:

1) Transform the rotation data $\boldsymbol{\omega}$ from the gyroscope to the axis-angle representation $\boldsymbol{R}$, following

$$\boldsymbol{R} = \{\frac{\boldsymbol{\omega}}{|\boldsymbol{\omega}|}, |\boldsymbol{\omega}|\} \ . \tag{4.3}$$

2) Transform the axis-angle representation $\boldsymbol{R}$ to the rotation matrix $\mathcal{M}$ [49].

3) Calculate $\boldsymbol{g}_{new}$ by the matrix mulplication, as

$$\boldsymbol{g}_{new} = \mathcal{M}^{-1}\boldsymbol{g}_{old} \ . \tag{4.4}$$

In short, gravity sensing follows two distinct procedures in two cases. If the device is at rest or moving at a constant speed, the measurement by the accelerometer is exactly the impact of gravity. Otherwise, the direction of gravity can be calculated based on previous status and the current rotation. This process can be done in real time on each device. However, due to the bias and errors of the gyroscope, the accuracy of the sensed gravity gradually decreases if the device keeps moving, and it will be recovered once the device enters in the "stable" state.

The output of the gravity sensing is used to determine the upward direction of the vehicle, which is the same as the upward direction of any device under the earth's coordinate system.

Since $\boldsymbol{g}$ has the same direction as $\boldsymbol{u}$, we have

$$\boldsymbol{u} = \frac{\boldsymbol{g}}{|\boldsymbol{g}|} , \tag{4.5}$$

where $\boldsymbol{u}$ is the unit vector pointing to the up direction of the vehicle.

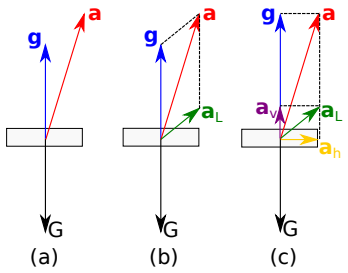


Figure 4.4: The components of the accelerometer's measurement

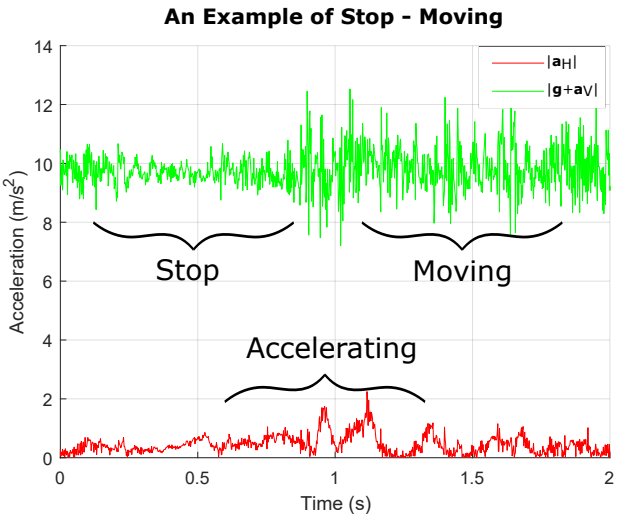


Figure 4.5: An example of the measured vehicle's acceleration

As shown in Fig. 4.4, after the gravity sensing, we can calculate the real acceleration $\boldsymbol{a}_L$, then decompose it with a horizontal acceleration $\boldsymbol{a}_H$ and a vertical acceleration $\boldsymbol{a}_V$. For the smartphone, $\boldsymbol{a}_H$ mainly contains the vehicle's acceleration, braking, and turning, because the vehicle is supposed to move horizontally; $\boldsymbol{a}_V$ is mostly induced by from the noise, vibration, and harassment caused by the engine and the road condition. For the smartwatch, the real acceleration $\boldsymbol{a}_L$ not only contains the vehicle's motion but also the motion of the user's hand.

4.4.3 Vehicle Motion Detection

After deriving the gravity and calculating $\boldsymbol{a}_H$, the smartphone can detect the forward direction of the vehicle based on the vehicle's horizontal movement. Our design is based on the following observations. First, since the smartphone is placed in the vehicle and not supposed

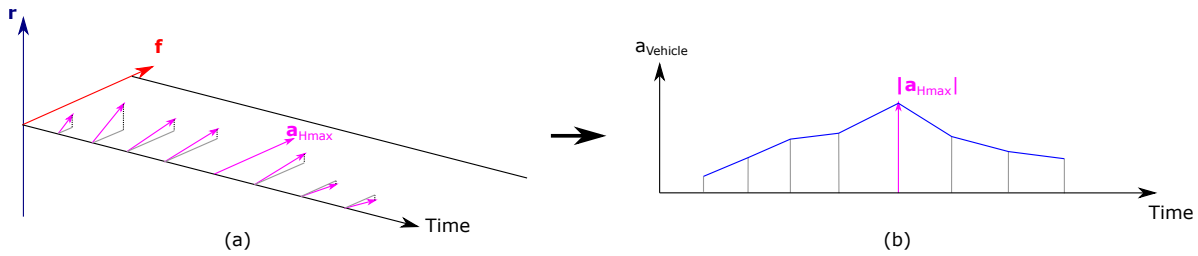


Figure 4.6: The detection of the motion of the vehicle

to be used while driving, its motion sensors only pick up the movement of the vehicle. Second, except turning, the vehicle's movement does not involve the rotation that changes the direction of gravity under the vehicle's coordinate system, and the vehicle only moves along the axis pointing to the forward direction. Considering these features, we conclude that the smartphone can detect the forward direction of the vehicle by analyzing the collected motion data. Specifically, once the smartphone phone senses a significant $\boldsymbol{a}_H$, the detection of the vehicle's movement can be done in three steps:

1) In a short period of time T, smartphone samples the rotations around the gravity. These samples correspond to the vehicle's turning. We use β_i to denote the value of the rotation, which is a counter-clockwise angle between the beginning of the period and the moment i. This value can be calculated by,

$$\beta_i = \begin{cases} \Delta_t|\boldsymbol{\omega}_i|, & \text{if } \boldsymbol{\omega}_i \cdot \boldsymbol{u}_i \geq 0; \\ -\Delta_t|\boldsymbol{\omega}_i|, & \text{if } \boldsymbol{\omega}_i \cdot \boldsymbol{u}_i < 0; \end{cases}, \tag{4.6}$$

where $\boldsymbol{\omega}_i$ and $\boldsymbol{u}_i$ are the readings from the gyroscope and the unit vector pointing to the vehicle's up direction at the moment i in a frame with length Δ_t, respectively.

2) To detect the vehicle's acceleration, the smartphone finds the largest $\boldsymbol{a}_H$ in T. This vector of acceleration is denoted as $\boldsymbol{a}_{Hmax}$. The direction of $\boldsymbol{a}_{Hmax}$ is assumed to be either the forward or the backward direction of the vehicle, and we can calculate the vehicle's forward direction under the smartphone's coordinate system by

$$\boldsymbol{f} = \begin{cases} \dfrac{\boldsymbol{a}_{Hmax}}{|\boldsymbol{a}_{Hmax}|}, & \text{if accelerating;} \\ -\dfrac{\boldsymbol{a}_{Hmax}}{|\boldsymbol{a}_{Hmax}|}, & \text{if braking;} \end{cases}, \tag{4.7}$$

where $\boldsymbol{f}$ is the unit vector pointing to the vehicle's forward direction.

3) The $\boldsymbol{a}_H$ at each moment in the 1 second is projected to $\boldsymbol{a}_{Hmax}$ as the vehicle's acceleration. We define the vehicle's acceleration at the moment i as b_i, and we have

$$b_i = \begin{cases} |\textsc{Project}(\boldsymbol{a}_{H,i}, \boldsymbol{f})|, & \text{if } \boldsymbol{a}_{H,i} \cdot \boldsymbol{f} \geq 0; \\ -|\textsc{Project}(\boldsymbol{a}_{H,i}, \boldsymbol{f})|, & \text{if } \boldsymbol{a}_{H,i} \cdot \boldsymbol{f} < 0; \end{cases}, \tag{4.8}$$

By calculating b_i, the smartphone learns the axis that points to the forward or the backward direction of the vehicle. In order to identify the direction of $\boldsymbol{a}_{Hmax}$, we need to know whether the vehicle is actually accelerating or braking at this moment. A practical solution is to examine the amplitude of the vertical vibration. Based on our experiments, the vibration is much more significant when the vehicle is moving as shown in Fig. 4.5. According to this observation, whenever the vehicle starts to move or comes to a stop, the smartphone can determine the forward and the backward direction. In some rare cases, the measurements of the accelerometer on the smartphone cannot be directly used to derive the forward direction of the vehicle. The first case is when the vehicle is moving backward. The acceleration's direction is the opposite of the forward direction. This case is rare when the car is in normal driving condition and does not significantly affect the performance. The second case is driving on a long curved road like the ramp on the exit of the entrance of the freeway. The constant turning leads to a centripetal acceleration that can be sensed by the smartphone. This error can be fixed soon when the vehicle is moving straightly. The third case is related to the horizontal vibration caused by the road condition. However, compared with the accelerating or braking, the vibration has a special pattern with a large variance within a short period of time, which can be easily identified and filtered out.

Along with the detection of the vehicle's motion, the smartphone logs a series of the ro-

tations $\{\beta\}$ as well as the vehicle's accelerations $\{b\}$. These data are sent to the smartwatch. In the next section, we introduce how the smartwatch learns the coordinate system of the vehicle based on the vehicle's motion.

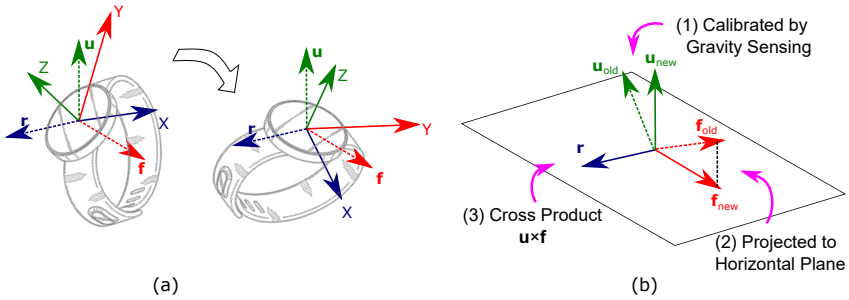


Figure 4.7: The maintenance of the vehicle's coordinate system on the smartwatch

4.4.4 Adaptive Coordinate System Maintenance

Unlike the smartphone, since the smartwatch's motion may contain the user's hand motion, we must ensure that the recognized vehicle's coordinate system keeps absolutely still but not varies with the motion. For example, as shown in Fig. 4.7 (a), the smartwatch is rotated due to the motion of the hand. As a result, the vehicle's forward and right axes should also be rotated and point to the absolute directions under the smartwatch's coordinate system. To accomplish this, we need to process the rotations from the gyroscope and continuously update the vehicle's forward direction $\boldsymbol{f}$ on the smartwatch. Just like how we keep track of the gravity, we calculate the $\boldsymbol{f}_{new}$ based on the $\boldsymbol{f}_{old}$ and $\boldsymbol{\omega}$. Specifically, we transform $\boldsymbol{\omega}$ into axis-angle representation $\boldsymbol{R}$, then to the rotation matrix $\mathcal{M}$ [49]:

$$\boldsymbol{f}_{new} = \mathcal{M}^{-1}\boldsymbol{f}_{old} \ . \tag{4.9}$$

However, due to the error and bias of the gyroscope, we need to calibrate $\boldsymbol{f}$ when the sensed gravity $\boldsymbol{g}$ is adjusted after the smartwatch enters the "stable" state. In other words, the recognized vehicle's coordinate system is calibrated adaptively whenever the accumulated errors of the gyroscope are eliminated through the gravity sensing. At this moment, $\boldsymbol{f}_{old}$ is

projected to the horizontal plane to calculate the direction of $\boldsymbol{f}_{new}$, as shown in Fig. 4.7 (b). Moreover, the right direction of the vehicle can be determined by calculating the cross product of $\boldsymbol{u}$ and $\boldsymbol{f}$.

By adaptively adjusting the recognized vehicle's coordinate system of the smartwatch, $\boldsymbol{f}$ points to an absolute direction no matter how the smartwatch moves. The smartwatch keeps logging $\boldsymbol{f}$ and combines them with the received vehicle's motion to align itself to the actual vehicle's coordinate system, which will be discussed in detail in the next section.

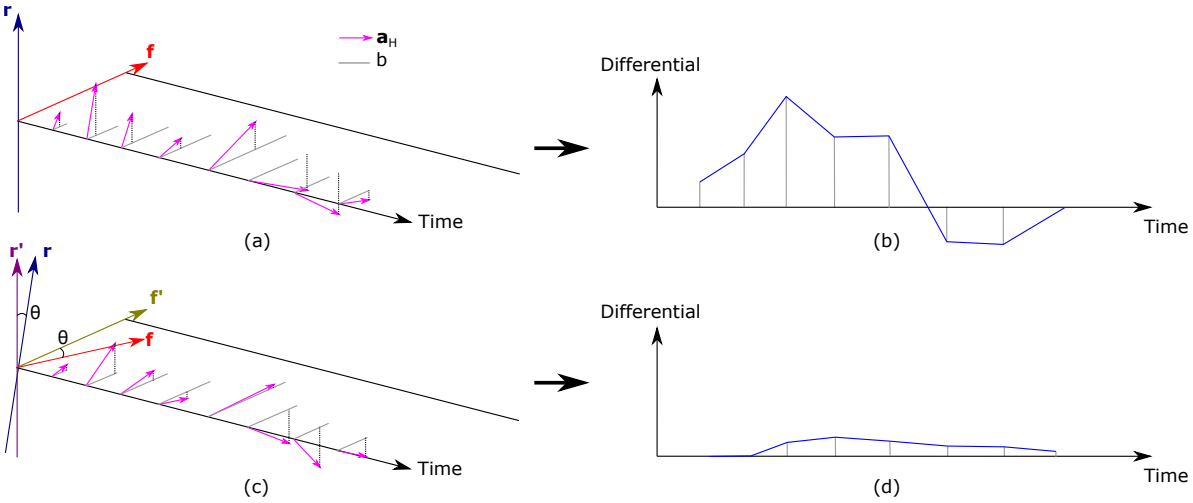


Figure 4.8: Detect the vehicle's forward direction by the smartwatch

4.4.5 Coordinate System Alignment on Smartwatch

After receiving the vehicle's motion detected by the smartphone, the smartwatch uses the data to align the coordinate systems between the vehicle and itself. A key observation is since the smartwatch stays in the vehicle, its speed should be similar to the vehicle's speed when the smartwatch is in the "stable" state. If the smartwatch has prior knowledge of the vehicle's forward direction, i.e. a set $\{\boldsymbol{f}\}$ containing the recognized forward direction of the vehicle under its coordinate system for the previous moments, it is possible to calculate the smartwatch's acceleration along with the vehicle's forward axis at each moment i as $\textsc{Project}(\boldsymbol{a}_H, \boldsymbol{f}_i)$. Moreover, if each $\boldsymbol{f}$ is recognized correctly at each moment during T,

the accumulation of the horizontal acceleration, i.e. the smartwatch's speed along with the vehicle's forward axis, must be the same value as the vehicle's speed at the end of the hand movement. Based on this observation, we have,

$$\epsilon = \left| \sum_{i=0}^{n} (\textsc{Project}(\boldsymbol{a}_{H,i}, \boldsymbol{f}'_i) - b_i \boldsymbol{f}'_i) \right|, \quad \epsilon \to 0 \ , \tag{4.10}$$

where $\boldsymbol{a}_{H,i}$ is the horizontal acceleration sensed by smartphone at the moment i, $\boldsymbol{f}'_i$ is the unit vector correctly pointing to the forward direction of the vehicle at the moment i, and b_i is the vehicle's acceleration at the moment i. Here, for samples at moment 0 and moment n, the smartwatch is in the "stable" state. The differential ϵ is generally close to 0. Our goal is to search for a series of $\boldsymbol{f}'_i$ that can minimize the differential based on a set of priorly known $\{\boldsymbol{f}\}$ on the smartwatch and the received data from the smartwatch containing vehicle's turning angles $\{\beta\}$ as well as the vehicle's acceleration $\{b\}$.

Since the motion sensors on the smartwatch and the smartphone may have different sampling rates, the first step of this process is to resample the motion data that represents the vehicle's motion by linear nearest-neighbor interpolation. Fig. 4.9 shows an example of the process of resampling.

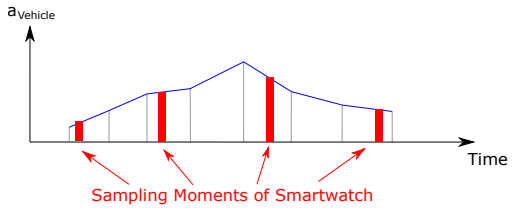


Figure 4.9: An example of resampling

Since the correct forward direction of the vehicle $\boldsymbol{f}'$ exists on the same horizontal plane with the pirorly known one $\boldsymbol{f}$, our goal can be simplified to be searching for θ, which is the angle between $\boldsymbol{f}$ and $\boldsymbol{f}'$, and we have,

$$\boldsymbol{f}'_i = \textsc{Rotate}(\boldsymbol{f}_i, \boldsymbol{u}_i, \sum_{j=0}^{i} \beta_j + \theta). \tag{4.11}$$

where function ROTATE($\boldsymbol{v}$,$\boldsymbol{s}$,α) returns a vector that is calculated by rotating vector $\boldsymbol{v}$ with angle α counter-clockwise around axis $\boldsymbol{s}$. Fig. 4.8 shows an example of the selection of θ and how the differential ϵ changes by using $\boldsymbol{f'}$ instead of $\boldsymbol{f}$. According to Equation (4.10) and (4.11), we need to solve the value of θ that minimizes ϵ. However, this formula is a high-degree polynomial equation with trigonometric functions, and a general algebraic solution does not exist according to the Abel-Ruffini theorem [3]. In the following, we propose a gradient descent method to solve θ. Specifically, on the first attempt, we start with the $\theta = 0$, which is the previous alignment between the coordinate systems. Then, we search for θ with 0.01 learning rate in a range between -90° to 90°. The advantage is that not only it requires only a small memory footprint, but also the expectation of the error can be limited within 1° after around 200 trials. This algorithm can be executed within 50ms by an off-the-shelf smartwatch.

At the startup, $\boldsymbol{f}$ on the smartwatch is set to $< 0, 1, 0 >$. With this method, every time the smartwatch receives the information about the vehicle's movement, we rotate the current $\boldsymbol{f}$ on the smartwatch with angle $\hat{\theta}$ around $\boldsymbol{u}$ in order to align it to the vehicle's actual forward direction.

4.4.6 Motion Tracking and Gesture Recognition

After a set of unit vectors $< \boldsymbol{r}, \boldsymbol{f}, \boldsymbol{u} >$ are found and maintained by the previous procedures, the smartwatch's attitude is known under the vehicle's coordinate system. Since the smartwatch's motion is related to the user's action, the information of its attitude in the vehicle can be used to recognize the user's hand gesture in a 3-D space. According to the design of the motion sensors, the X-axis of the smartwatch is always parallel to the user's forearm, the user's gesture can be defined as how the direction of its X-axis is represented under the coordinate system of the vehicle, i.e. where the hand points to. We denote this vector as $\boldsymbol{x'}$, and we have

$$\begin{aligned} \boldsymbol{x}' &= < \textsc{Project}(\boldsymbol{x}, \boldsymbol{r}), \textsc{Project}(\boldsymbol{x}, \boldsymbol{u}) >, \\ \boldsymbol{x} &= < 1, 0, 0 > . \end{aligned} \tag{4.12}$$

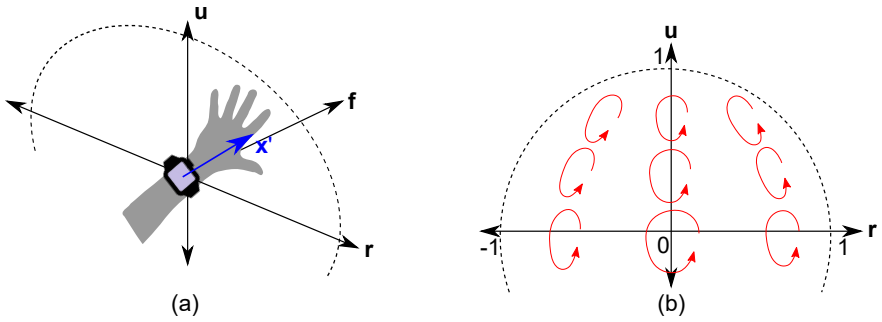


Figure 4.10: Basic idea of tracking-based gesture recognition

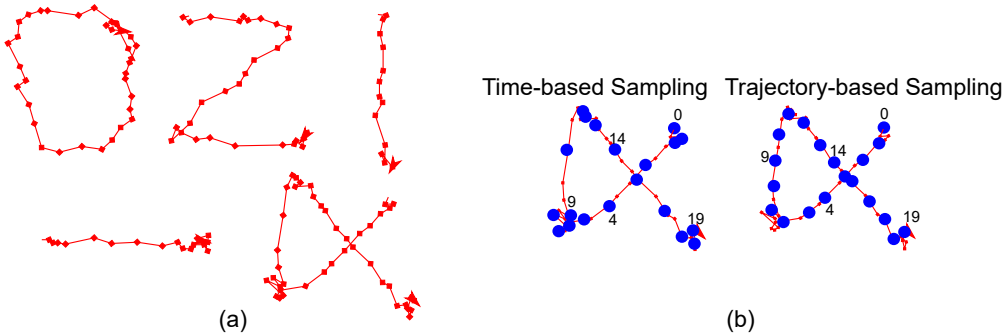


Figure 4.11: Representation of a gesture by samples

Note that we only need to know the projection of $\boldsymbol{x}$ to $< \boldsymbol{r}, \boldsymbol{u} >$ plane, because two coordinate values are enough to determine a 3-D unit vector. We developed an application to recognize the gesture based on the result by a Convolutional Nerual Network (CNN) deep learning model. In our experiment, when the watch face is tilt to upward, the sampling of a gesture starts [80]. The gesture can be represented as a series of $\boldsymbol{x}'$ as shown in Fig. 4.10(a). To keep the consistency of size of the data that represents a gesture, we take another resampling step as shown in Fig. 4.11(b). We pre-defined five basic and easy-to-understand shapes of the gestures, which are the circle like "O", the letter "Z", the vertical down stroke like "I", the horizontal right stroke like "-", and the cross like "X". Each gesture can be performed at one of the nine positions as shown in Fig. 4.10(b).

The pipeline of the design of the gesture recognition is shown in Fig. 4.13. We recorded 20 sample-series of each of the 45 combinations and trained the classifier by TensorFlow [1] on a laptop. The classifier uses a CNN model as shown in Fig. 4.12 with the input as a series of both time-based and trajectory-based sampled $\boldsymbol{x}'$. The model is strong in capturing the individual connectivity patterns for each gesture according to the design of CNN. The training can be done with those pre-defined gestures as well as additional customized ones. After this offline training, the generated model is downloaded to the smartphone for real-time gesture recognition with TensorFlow Lite.

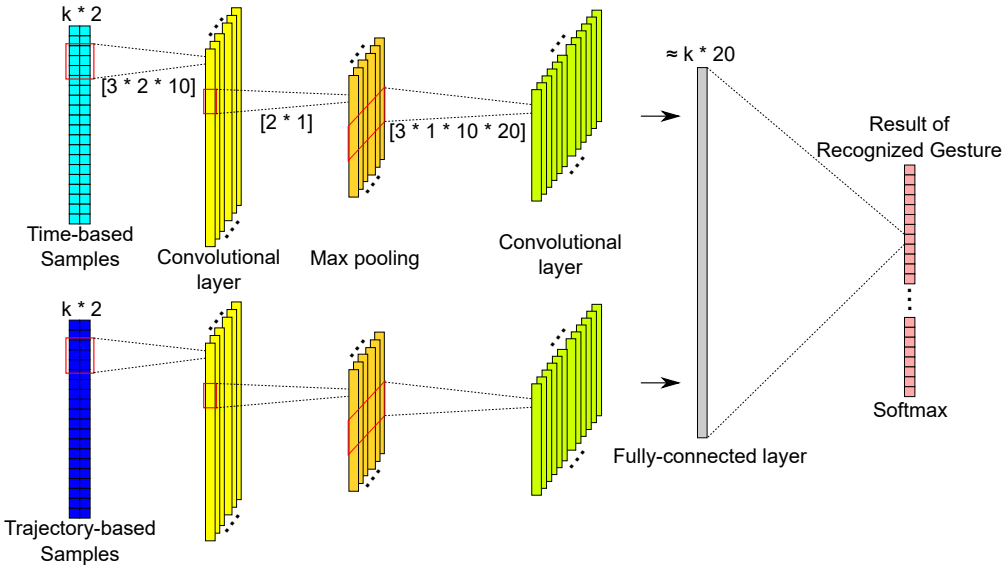


Figure 4.12: The structure of the CNN deep learning model

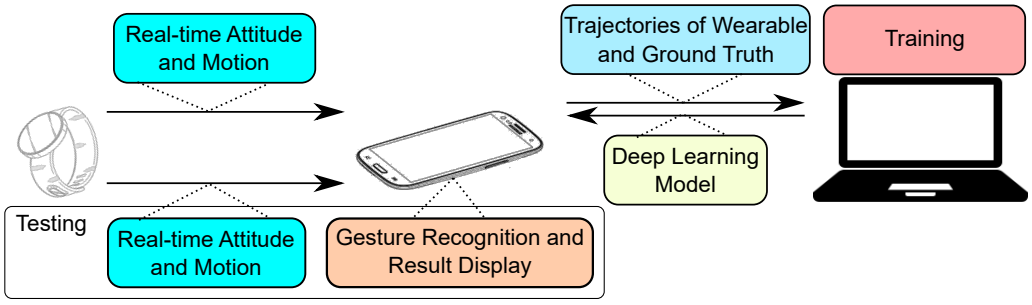


Figure 4.13: The pipeline of training and testing process of gesture recognition

4.5 Evaluation

In order to evaluate the performance of RMAT, We have implemented it on a smartphone (Moto G 2nd Gen.) and a smartwatch (Moto G 360 2nd Gen.). To evaluate the performance of the alignment, the recognized coordination system of the vehicle is displayed on both devices in real time. The displayed result can be switched between the RAMT and GPS with compass. For gesture recognition, the smartphone runs the CNN model. The recognized gesture is displayed on the smartphone. In addition to our pre-defined gestures, each subject is allowed to record two more customized gestures during the experiment, and a new CNN model will be trained immediately on a laptop.

We have recruited 10 subjects to test RAMT under the approval by the Institutional Review Board (IRB) of the author's institution. The experiment is done in a moving vehicle. In the vehicle, the ground truth of the coordinate system of the vehicle is indicated as a preset cotton string, and the gesture performed by the subject is captured by a camera above the seat as shown in Fig. 4.14. Each subject is asked to wear the smartwatch (in our experiment, all of the 10 subjects choose to wear the smartwatch on the left hand) and carry the smartphone. For safety concerns, in each experimental session, the subject performing the experiments is a passenger during a 30-min driving trip. During this trip, the subject is responsible to regularly check the accuracy of the alignment, and he/she is also required to perform 3 times of each customized gestures and 30 randomly chosen gestures, which comes from the 45 pre-defined shape-position combinations, then records the errors of the gesture recognition.

4.5.1 Computational Performance

We uses devices listed in Table. 4.1 for this experiment. As shown in Fig. 4.17, RMAT is able to run on multiple threads on each mobile device. The processing time of each sample is sufficiently short comparing with the sampling rate of the motion sensors. The details of

Table 4.1: Information of the devices

Device	CPU	RAM	TensorFlow
Moto G 2nd Gen.	Qualcomm Snapdragon 400, 1.2GHz, quad-cores	1GB	TensorFlow Lite
Moto G 360 2nd Gen.	Qualcomm Snapdragon 400, 1.2GHz quad-core	512MB	N/A
Lenovo ThinkPad	Intel Core i5-7300, 2.6GHz / 2.71GHz, quad-core	8GB	TensorFlow 1.8.1

Table 4.2: Information of the deep learning model

Number of Samples	Model Size	Time of Training	Time of Prediction	Accuracy of Validation
10	45KB	2min	310ms	87.3%
20	85KB	3min	430ms	95.8%
30	125KB	6min	535ms	98.3%
40	165KB	9min	625ms	98.4%

Figure 4.14: Setup of experiment

the average processing time of each step is listed in Table. 4.3.

The size and the processing time of the CNN model depend on the number of samples we extracted from the gestures as shown in Fig. 4.12. In our study, we use a 0.05 learning rate and 50000 rounds of the gradient descents to train the model. The outcomes of each setup are summarized in Table. 4.2. In the experiment, we choose 30 samples for each gesture as a balance between the processing time and accuracy.

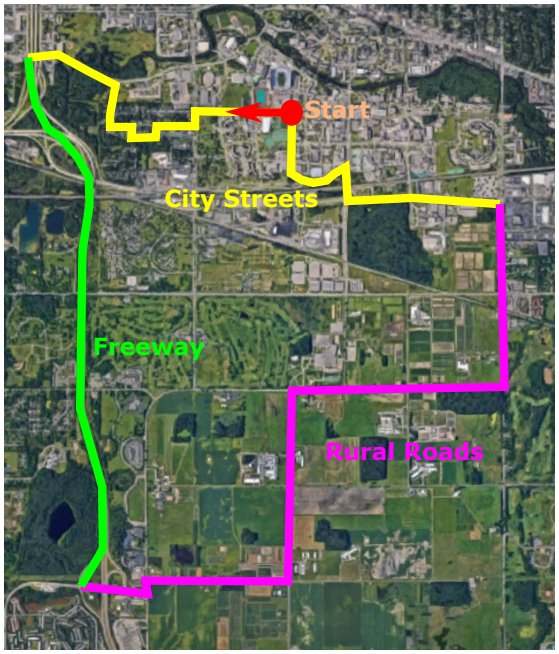


Figure 4.15: The driving route of experiment

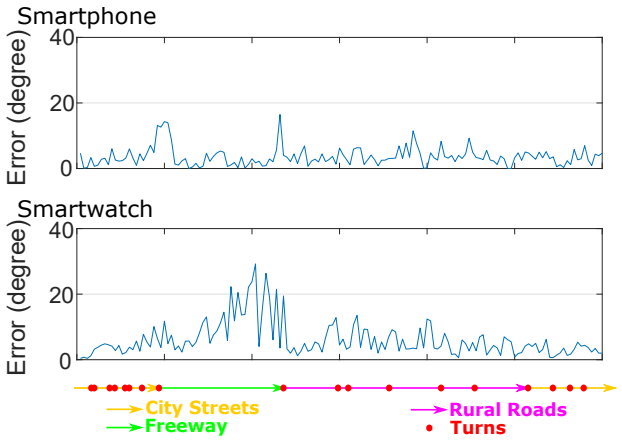


Figure 4.16: The error of alignment

Table 4.3: Processing time of each step

Smartphone		Bluetooth	Smartwatch		
Sampling, Gravity Sensing	Vehicle Motion Extraction	Transmission	Sampling, Gravity Sensing, etc.	Resampling	Gradient Descent
1.32ms	4.18ms	257ms	3.77ms	6.79ms	38.6ms

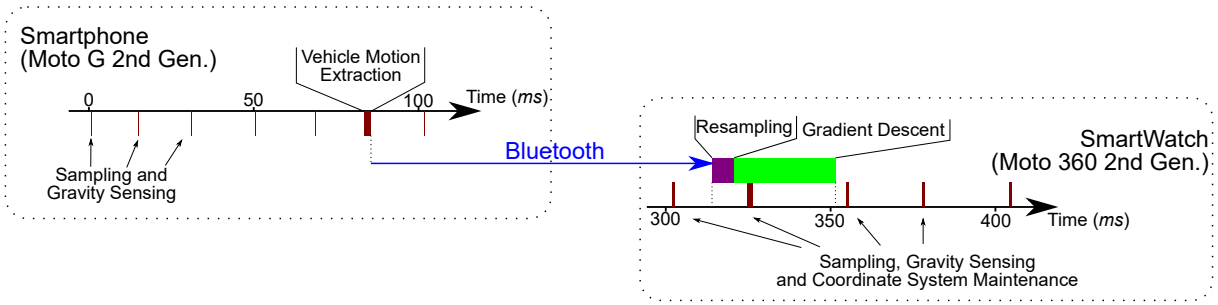


Figure 4.17: The timeline of processing

4.5.2 Coordinate System Alignment and Maintenance

As mentioned in Section 4.3, RAMT searches for a set of unit vectors $\boldsymbol{r}$, $\boldsymbol{f}$, and $\boldsymbol{u}$ under the native coordinate system of each device, which represents the coordinate system of the vehicle (the forward, right, and up directions). Since $\boldsymbol{u}$ can be calculated via gravity sensing

and $\boldsymbol{r}$ can be calculated by a cross product with $\boldsymbol{f}$ and $\boldsymbol{u}$, we only focus on the horizontal accuracy of $\boldsymbol{f}$. To this end, the subject is asked to compare a horizontal cotton string in the vehicle with the detected forward direction, which is displayed on the mobile device's screen. The angle between those two directions is the error of the coordinate system alignment in the horizontal panel.

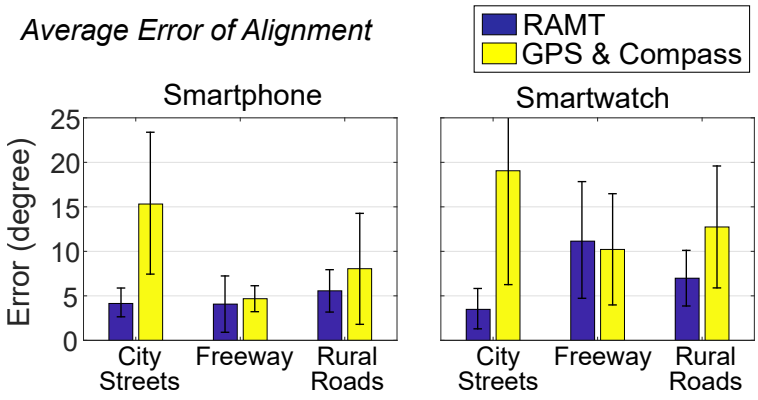


Figure 4.18: The average error of alignment

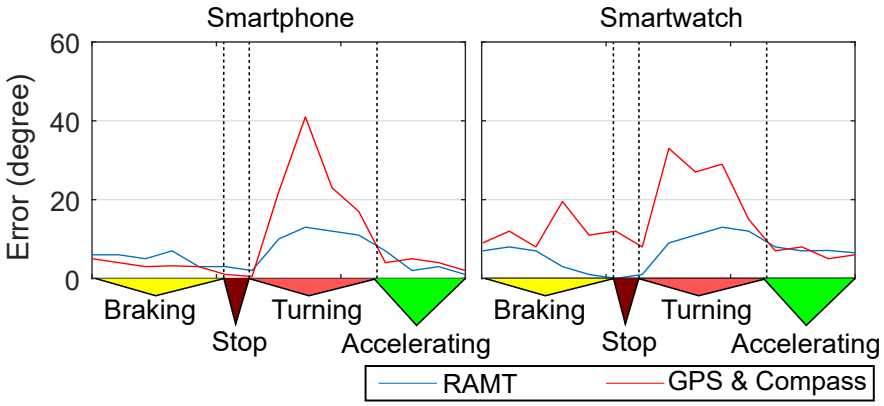


Figure 4.19: The error of alignment for turning

The smartphone captures the vehicle's horizontal motion and recognizes the forward direction of the vehicle. According to the design of RAMT, the alignment will be performed after a significant change in the vehicle's speed. These often occur before or after the vehicle's turning. As shown in Fig. 4.16 and Fig. 4.18, the average error is around 5° throughout the trip. The error is lower on the city streets, where the vehicle stops and turns frequently.

However, an error of around 15° can be observed when the vehicle enters or exits the freeway. The long-lasted turning leads to a significant centripetal force, which is wrongly detected as an acceleration. The error can be fixed soon when the vehicle leaves the ramp. The smartwatch receives the extracted vehicle's motion from the smartphone and calculates the forward direction of the vehicle. The average error is around 10° all through the trip. In addition to the errors occurring on the ramp, the error gradually increases while the vehicle is driving straightly for a long distance, e.g. on the freeway or the rural road, where the vehicle's speed does not vary significantly. Our study shows that the error comes from the zero drifting of the gyroscopes on both of the smartphone and the smartwatch.

As shown in Fig. 4.18, the solution based on GPS and compass is relatively better for the smartphone when the vehicle is moving straightly. However, when the vehicle turns or shifts a lane, the error appears in a burst. Fig. 4.19 shows a detailed result of how the accuracy varies when the vehicle enters or exits the turn. For both the smartphone and the smartwatch, the errors are reduced when the vehicle was braking or accelerating, and the errors while turning are less than 20°. The GPS and compass solution fails to maintain accuracy while turning because the GPS must process a series of samples to estimate the new direction, which leads to long latency. The compass and GPS solution has a relatively low accuracy for the smartwatch. The attitude estimation by the magnetometer is not robust for an object under frequent motion. Moreover, due to the irregular hand motion, the smartwatch constantly travels across the magnetic field from the vehicle or other devices [87]. As a result, the measurements of the magnetometer are not reliable for detecting a moving device's attitude relative to the earth.

4.5.3 Hand Gesture Recognition

Based on the result of hand gesture recognition, we build an error matrix as Fig. 4.20 to show the accuracy. Here we have 45 rows and columns (5 shapes multiplies 9 positions) for the pre-defined gestures and 20 rows and columns (2 customized gestures for each of 10 subjects)

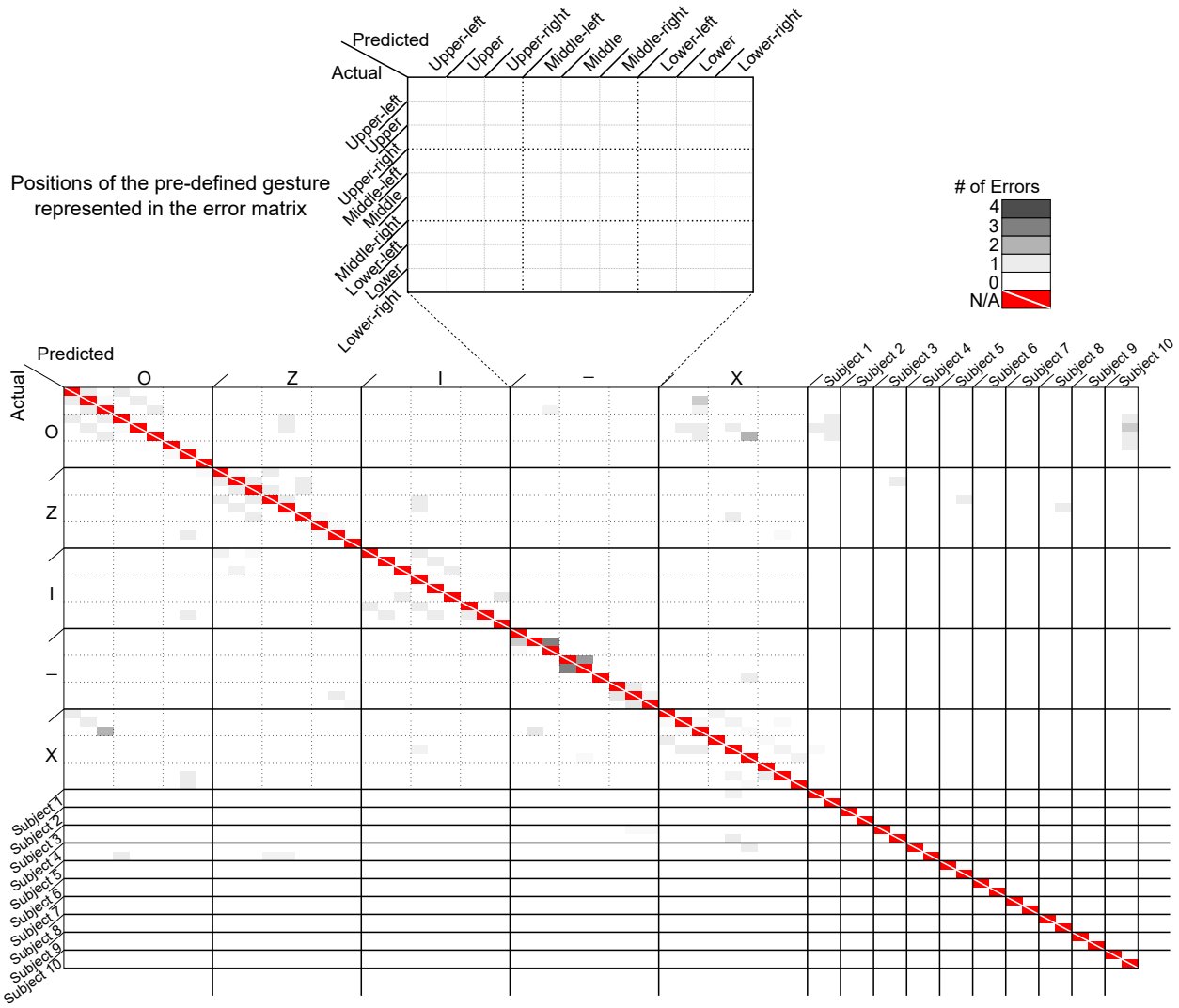


Figure 4.20: The error matrix for detecting all gestures

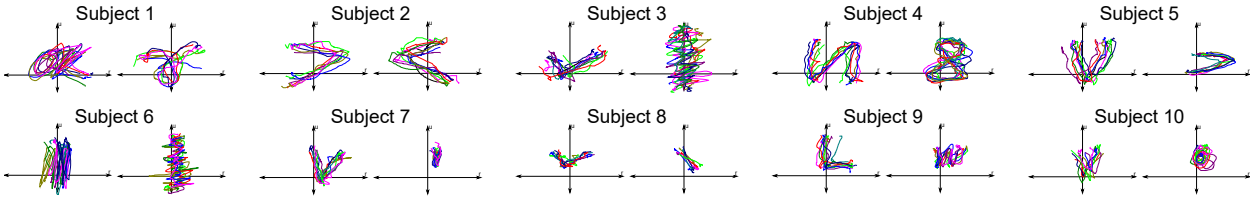


Figure 4.21: The trajectories of all the customized gestures

for the customized gestures. Each cell represents the number of errors in recognition, which corresponds to the truth on the row and predicted result on the column. In total, 360 gestures by the subjects are recognized by our application. The number of total errors is 57, which means approximately 84% accuracy. Among the predefined gestures, "z" has the

least errors of both the recognition of the position and the shape. Gesture "o" and "x" are often misclassified with each other, because "x" often contains a similar shape as the circle as shown in Fig. 4.11. The recognition of the vertical position of gesture "I" has lower accuracy, because the start and the end of this gesture have a large variety among the subjects. This issue also happens to the horizontal position of the gesture "-". Overall, most of the errors appear at the upper and upper right position. This is related to the basic idea of RAMT and the special posture of the user in the vehicle. When the user sits in the vehicle and reaches to the upper-right position with the left hand, the lower arm actually has more freedom. Thus, although the subject supposes that he/she moves the hand to the upper-right position, the forearm posture, i.e. the attitude of the smartwatch, has a large variety.

For customized gestures, most of the subject only observe two or three errors, including false alarm and missed detection. However, the accuracy is relatively low for some customized gestures, which are shown in Fig. 4.21. Specifically, gesture "o" is often misclassified as the second one of the subject 1 and the second customized one of subject 10 (double circles), because a circle exists in those customized gestures. The second gesture of subject 4 does not only contains a circle but also contains a "z" shape. This causes some difficulty for our system to accurately recognize it. We observe that shape "v" is a preferred choice among most of the subjects. The result also shows that postures with "v" shape are not likely to be confused with other gestures.

4.6 Discussion

Our results suggest that the accuracy of the alignment should be higher in a well-controlled situation. Once the smartphone is put in a vehicle and it is not touched or moved, the alignment of the coordinate system between the smartphone and the vehicle only needs be executed once. However, in our experiment, it is required to check the error of the alignment by manual operation. Thus, the alignment must constantly run on the

smartphone to maintain accuracy. A possible improvement is only executing the alignment when the smartphone is moved by the user. This requires RMAT to distinguish the vehicle's motion (e.g. accelerating, turning, braking, vibration, etc.) and the activities by the user. The solution is left for future work.

Although deep learning has a good performance in gesture recognition, its computing overhead is still heavy for a mobile device. A recommended technique to recognize gestures is decomposing the gesture into multiple segments, then recognizing the whole gesture by analyzing the combination of the segments [55]. This technique is proved to be effective with computer vision, but it is not designed for motion-based gesture recognition, which contains significantly more unpredictable errors in data sampling. It is hence important to develop a better machine learning methodology for gesture recognition or motion tracking solely based on motion signals.

Once the coordinate systems on the smartphone and the smartwatch are correctly aligned, RAMT can maintain an accurate alignment following the procedure in Section 4.4.4. The only factor that affects the accuracy is the accumulated zero drifting on the gyroscopes. Several studies have been conducted to predict the value of drifting based on various factors, such as ambient temperature [28] and the device's attitude [90]. However, these methods are not designed to work in real time and hence are ill-suited for our scenario. One possible solution in the future is to periodically use the GPS and compass alignment as a reference to mitigate the zero drifting, especially on the highway, where the acceleration and braking are rare.

4.7 Conclusion of Study

RAMT monitors low-level activities. It describes the result of monitoring with continuous numbers rather than discreate classes. It satisfies the **Three Rules**. Specifically, RAMT

- **is sensitive to context**. The state of each device is constantly switched between moving or still. The position of the device at the next moment is totally decided by

the position at the previous moment and the motion signal.

- **is adaptive to dynamic condition**. RAMT can leverage each significant change of the vehicle's speed to calibrate its model. This grants it high accuracy along the trip.
- **processes readable features**. It shares rotation and acceleration of the vehicle between multiple devices. Those features are readable by human.

CHAPTER 5

CONCLUSION

This dissertation uses the experience from three studies to introduce **Three Rules** of monitoring human activities with mobile devices. These rules are summarized from the successful designs of those systems, which are used to monitor the family mealtime activities and driver's behavior. In conclusion, again, this dissertation argues that such a system should,

- **be sensitive to context**. Human activities can be allocated into time slots. One activity has a very strong relation to the ambient condition as well as the activities before and after it. The basic context-sensitive model like HMM is still a good choice to recognize activities like speech. This kind of model inspires the new design of deep learning models in some recent studies [86] [98].

- **be adaptive to dynamic condition**. The system must be able to calibrate itself, update the training data set, or accept corrections to maintain high accuracy over time. Different from recognizing the image or translating the text, a human may change the habit or routine constantly. This rule has attracted attention since the concept of reinforcement learning is widely studied [102].

- **process readable features**. It is preferred to use readable features and predictable classifiers in monitoring human activities rather than directly applying the neural network. The reason is these systems may relate to the clinical studies or safety issue, where the performance of the system must be carefully maintained and under control. Some biology studies already express concern about the hidden parameters of deep learning at the current stage [27]. However, deep learning is still a good approach for monitoring human activities because of its outstanding performance. Some recent studies try to integrate this rule into deep learning, like neural network with HMM for speech recognition [2].

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: a system for large-scale machine learning.

[2] O. Abdel-Hamid and H. Jiang. Fast speaker adaptation of hybrid nn/hmm model for speech recognition based on discriminative learning of speaker code. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7942–7946. IEEE, 2013.

[3] N. H. Abel. *Mémoire Sur Les Équations Algeébriques Où on Démontre L'impssibilité de la Résolution de L'equation Générale Du Cinquième Dégré... Christiania, Groendahl, 1824*. University of Olso, 1824.

[4] E. Aidman, C. Chadunow, K. Johnson, and J. Reece. Real-time driver drowsiness feedback improves driver alertness and self-reported driving performance. *Accident Analysis & Prevention*, 81:8–13, 2015.

[5] T. A. A. Association. Avoiding crashes & emergency maneuvers. 2012. available at `http://seniordriving.aaa.com/improve-your-driving-skills/handle-unexpected-situations/avoiding-crashes-emergency/`.

[6] S. Avancha, A. Baxi, and D. Kotz. Privacy in mobile technology for personal healthcare. *ACM Comput. Surv.*, 45(1):3:1–3:54, Dec. 2012.

[7] A. Bandura. Health promotion by social cognitive means. *Health education & behavior*, 31(2):143–164, 2004.

[8] D. E. Baxa. Noise control in internal combustion engines. *JOHN WILEY & SONS, INC, 605 THIRD AVE., NEW YORK, NY 10158, 1982, 520*, 1982.

[9] V. Belagiannis, S. Amin, M. Andriluka, B. Schiele, N. Navab, and S. Ilic. 3d pictorial structures revisited: Multiple human pose estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):1929–1942, Oct 2016.

[10] N. Bellissimo, P. B. Pencharz, S. G. Thomas, and G. H. Anderson. Effect of television viewing at mealtime on food intake after a glucose preload in boys. *Pediatric Research*, 61(6):745–749, 2007.

[11] A. Ben-Yaacov, M. Maltz, and D. Shinar. Effects of an in-vehicle collision avoidance warning system on short-and long-term driving performance. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 44(2):335–342, 2002.

[12] N. Bendickson, E. F. Harper, CSP, DABFE, DABFET, CFC, and T. C. Healey. Tips to avoid distracted driving. In *Driver Safety*, 2012.

[13] T. R. Bennett, R. Jafari, and N. Gans. Motion based acceleration correction for improved sensor orientation estimates. In *Wearable and Implantable Body Sensor Networks (BSN), 2014 11th International Conference on*, pages 109–114. IEEE, 2014.

[14] C. Bi, J. Huang, G. Xing, L. Jiang, X. Liu, and M. Chen. Safewatch: A wearable hand motion tracking system for improving driving safety. In *Internet-of-Things Design and Implementation (IoTDI), 2017 IEEE/ACM Second International Conference on*, pages 223–232. IEEE, 2017.

[15] C. Bi and G. Xing. Ramt: Real-time attitude and motion tracking for mobile devices in moving vehicle. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 3(2):38:1–38:21, June 2019.

[16] C. Bi, G. Xing, T. Hao, J. Huh, W. Peng, and M. Ma. Familylog: A mobile system for monitoring family mealtime activities. In *2017 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 21–30. IEEE, 2017.

[17] S. T. Birchfield and R. Gangishetty. Acoustic localization by interaural level difference. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, volume 4, pages iv–1109. IEEE, 2005.

[18] H. Blunck, S. Bhattacharya, A. Stisen, T. S. Prentow, M. B. Kjærgaard, A. Dey, M. M. Jensen, and T. Sonne. Activity recognition on smart devices: Dealing with diversity in the wild. *GetMobile: Mobile Comp. and Comm.*, 20(1):34–38, July 2016.

[19] A. Bonde, S. Pan, Z. Jia, Y. Zhang, H. Y. Noh, and P. Zhang. Vvrrm: Vehicular vibration-based heart rr-interval monitoring system. In *Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications*, HotMobile '18, pages 37–42, New York, NY, USA, 2018. ACM.

[20] L. Boon-Leng, L. Dae-Seok, and L. Boon-Giin. Mobile-based wearable-type of driver fatigue detection by gsr and emg. In *TENCON 2015-2015 IEEE Region 10 Conference*, pages 1–4. IEEE, 2015.

[21] W. T. Boyce, E. W. Jensen, J. C. Cassel, A. M. Collier, A. H. Smith, and C. T. Ramey. Influence of life events and family routines on childhood respiratory tract illness. *Pediatrics*, 60(4):609–615, 1977.

[22] W. K. Bryant and C. D. Zick. An examination of parent-child shared time. *Journal of Marriage and Family*, 58(1):pp. 227–237.

[23] P. Burns, A. Parkes, S. Burton, R. Smith, and D. Burch. *How Dangerous is Driving with a Mobile Phone?: Benchmarking the Impairment to Alcohol*. Transport Research Laboratory Berkshire,, United Kingdom, 2002.

[24] J. M. Carmona and J. Climent. A performance evaluation of hmm and dtw for gesture recognition. In L. Alvarez, M. Mejail, L. Gomez, and J. Jacobo, editors, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 236–243, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[25] D. Chen, K.-T. Cho, S. Han, Z. Jin, and K. G. Shin. Invisible sensing of vehicle steering with smartphones. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 1–13. ACM, 2015.

[26] K. Chen and H. Shen. Roadaware: Learning personalized road information on daily routes with smartphones. In *Computer Communication and Networks (ICCCN), 2016 25th International Conference on*, pages 1–9. IEEE, 2016.

[27] T. Ching, D. S. Himmelstein, B. K. Beaulieu-Jones, A. A. Kalinin, B. T. Do, G. P. Way, E. Ferrero, P.-M. Agapow, M. Zietz, M. M. Hoffman, et al. Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 15(141):20170387, 2018.

[28] S. Chong, S. Rui, L. Jie, Z. Xiaoming, T. Jun, S. Yunbo, L. Jun, and C. Huiliang. Temperature drift modeling of mems gyroscope based on genetic-elman neural network. *Mechanical Systems And Signal Processing*, 72:897–905, 2016.

[29] H. Chung, M. Iorga, J. Voas, and S. Lee. Alexa, can i trust you? *Computer*, 50(9):100, 2017.

[30] T. L. Curtright, D. B. Fairlie, and C. K. Zachos. A compact formula for rotations as spin matrix polynomials. *arXiv preprint arXiv:1402.3541*, 2014.

[31] E. R. Dahlen, R. C. Martin, K. Ragan, and M. M. Kuhlman. Driving anger, sensation seeking, impulsiveness, and boredom proneness in the prediction of unsafe driving. *Accident Analysis & Prevention*, 37(2):341–348, 2005.

[32] J. W. Deng and H. T. Tsui. An hmm-based approach for gesture segmentation and recognition. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 3, pages 679–682 vol.3, Sep. 2000.

[33] S. A. Denham. Relationships between family rituals, family routines, and health. *Journal of Family Nursing*, 9(3):305–330, 2003.

[34] N. Deo, A. Rangesh, and M. Trivedi. In-vehicle hand gesture recognition using hidden markov models. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2179–2184, Nov 2016.

[35] T. J. Dishion, S. E. Nelson, and K. Kavanagh. The family check-up with high-risk young adolescents: Preventing early-onset substance use by parent monitoring. *Behavior Therapy*, 34(4):553–571, 2003.

[36] C. Dong, M. C. Leu, and Z. Yin. American sign language alphabet recognition using microsoft kinect. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 44–52, 2015.

[37] C. J. Dunst, D. Hamby, C. M. Trivette, M. Raab, and M. B. Bruder. Everyday family and community life and children's naturally occurring learning opportunities. *Journal of Early Intervention*, 23(3):151–164, 2000.

[38] N. Fakotakis, A. Tsopanoglou, and G. Kokkinakis. Text-independent speaker recognition based on vowel spotting. In *Digital Processing of Signals in Communications, 1991., Sixth International Conference on*, pages 272–277, Sep 1991.

[39] B. H. Fiese, A. Hammons, and D. Grigsby-Toussaint. Family mealtimes: a contextual approach to understanding childhood obesity. *Economics & Human Biology*, 10(4):365–374, 2012.

[40] K. Gade. The seven ways to find heading. *The Journal of Navigation*, 69(5):955–970, 2016.

[41] D. Gebre-Egziabher, G. H. Elkaim, J. Powell, and B. W. Parkinson. A gyro-free quaternion-based attitude determination system suitable for implementation using low cost sensors. In *Position Location and Navigation Symposium, IEEE 2000*, pages 185–192. IEEE, 2000.

[42] D. Gebre-Egziabher, G. H. Elkaim, J. D. Powell, and B. W. Parkinson. A gyro-free quaternion-based attitude determination system suitable for implementation using low cost sensors. In *IEEE 2000. Position Location and Navigation Symposium (Cat. No.00CH37062)*, pages 185–192, 2000.

[43] P. Gilbertson, P. Coulton, F. Chehimi, and T. Vajk. Using tilt as an interface to control no-button 3-d mobile games. *Comput. Entertain.*, 6(3):38:1–38:13, Nov. 2008.

[44] K. R. Ginsburg et al. The importance of play in promoting healthy child development and maintaining strong parent-child bonds. *Pediatrics*, 119(1):182–191, 2007.

[45] K. P. Goebel and C. B. Hennon. Mother's time on meal preparation, expenditures for meals away from home, and shared meals: Effects of mother's employment and age of younger child. *Home Economics Research Journal*, 12(2):169–188, 1983.

[46] P. D. Groves. *Principles of GNSS, inertial, and multisensor integrated navigation systems*. Artech house, 2013.

[47] S. Gupta, M. S. Reynolds, and S. N. Patel. Electrisense: Single-point sensing using emi for electrical event detection and classification in the home. In *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, Ubicomp '10, pages 139–148, New York, NY, USA, 2010. ACM.

[48] T. Hao, G. Xing, and G. Zhou. isleep: Unobtrusive sleep quality monitoring using smartphones. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, pages 4:1–4:14, New York, NY, USA, 2013. ACM.

[49] H. Harrison. Quaternions and rotation sequences: a primer with applications to orbits, aerospace and virtual reality, kuipers jb, princeton university press, 41 william street, princeton, nj 08540, usa. 1999. 372pp. illustrated.£ 35.00. isbn 0-691-05872-5. *The Aeronautical Journal*, 103(1021):175–175, 1999.

[50] A. M. Hendawi, A. Rustum, A. A. Ahmadain, D. Hazel, A. Teredesai, D. Oliver, M. Ali, and J. A. Stankovic. Smart personalized routing for smart cities. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 1295–1306, April 2017.

[51] B.-J. Ho, P. Martin, P. Swaminathan, and M. Srivastava. From pressure to path: Barometer-based vehicle tracking. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, pages 65–74. ACM, 2015.

[52] D. Hogg. Model-based vision: a program to see a walking person. *Image and Vision Computing*, 1(1):5 – 20, 1983.

[53] A. Honkela. *Nonlinear switching state-space models*. PhD thesis, Citeseer, 2001.

[54] Y. Hou, A. Gupta, T. Guan, S. Hu, L. Su, and C. Qiao. Vehsense: Slippery road detection using smartphones. 05 2017.

[55] F.-A. Huang, C.-Y. Su, and T.-T. Chu. Kinect-based mid-air handwritten digit recognition using multiple segments and scaled coding. In *Intelligent Signal Processing and Communications Systems (ISPACS), 2013 International Symposium on*, pages 694–697. IEEE, 2013.

[56] H. Huang and S. Lin. Toothbrushing monitoring using wrist watch. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, pages 202–215. ACM, 2016.

[57] Z. Huang and T. Zhu. Sbd: A signature-based detection for activities of appliances: Poster abstract. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, BuildSys '14, pages 222–223, New York, NY, USA, 2014. ACM.

[58] D. A. James, N. Davey, and T. Rice. An accelerometer based sensor platform for insitu elite athlete performance analysis. In *Proceedings of IEEE Sensors, 2004.*, pages 1373–1376 vol.3, Oct 2004.

[59] J. W. Jenness, R. J. Lattanzio, M. O'Toole, and N. Taylor. Voice-activated dialing or eating a cheeseburger: which is more distracting during simulated driving? In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 46, pages 592–596. SAGE Publications, 2002.

[60] L. Jiang, X. Lin, X. Liu, C. Bi, and G. Xing. Safedrive: Detecting distracted driving behaviors using wrist-worn devices. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(4):144, 2018.

[61] D. A. Johnson and M. M. Trivedi. Driving style recognition using a smartphone as a sensor platform. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pages 1609–1615. IEEE, 2011.

[62] A. Kadomura, C.-Y. Li, Y.-C. Chen, H.-H. Chu, K. Tsukada, and I. Siio. Sensing fork and persuasive game for improving eating behavior. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, pages 71–74. ACM, 2013.

[63] S. Kaplan, M. A. Guvensan, A. G. Yavuz, and Y. Karalurt. Driver behavior analysis for safe driving: a survey. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):3017–3032, 2015.

[64] C. Karatas, L. Liu, H. Li, J. Liu, Y. Wang, S. Tan, J. Yang, Y. Chen, M. Gruteser, and R. Martin. Leveraging wearables for steering and driver tracking. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, April 2016.

[65] H. B. Kekre, A. Athawale, and M. Desai. Speaker identification using row mean vector of spectrogram. In *Proceedings of the International Conference & Workshop on Emerging Trends in Technology*, ICWET '11, pages 171–174, New York, NY, USA, 2011. ACM.

[66] S. G. Klauer, F. Guo, B. G. Simons-Morton, M. C. Ouimet, S. E. Lee, and T. A. Dingus. Distracted driving and risk of road crashes among novice and experienced drivers. *New England journal of medicine*, 370(1):54–59, 2014.

[67] S. G. Klauer, F. Guo, B. G. Simons-Morton, M. C. Ouimet, S. E. Lee, and T. A. Dingus. Distracted driving and risk of road crashes among novice and experienced drivers. *New England journal of medicine*, 370(1):54–59, 2014.

[68] M. Klausner and W. Grimm. Method for detecting the position of hands on a steering wheel, Mar. 28 2006. US Patent 7,019,623.

[69] S. Kopparapu and M. Laxminarayana. Choice of mel filter bank in computing MFCC of a resampled speech. In *Information Sciences Signal Processing and their Applications (ISSPA), 2010 10th International Conference on*, pages 121–124, May 2010.

[70] C. J. Kulas. Visual indicators on vehicle steering wheel displayed in response to hand position, Oct. 20 2009. US Patent 7,605,693.

[71] M. P. Kumar, A. Zisserman, and P. H. S. Torr. Efficient discriminative learning of parts-based models. In *2009 IEEE 12th International Conference on Computer Vision*, pages 552–559, Sept 2009.

[72] I. lain Seymour-Hart. Road traffic accident reconstruction: Vision, alertness and reaction relating to driving. Technical report, SAE Technical Paper, 2000.

[73] L. Leavitt. Sleep-detecting driving gloves, Jan. 18 2000. US Patent 6,016,103.

[74] B.-G. Lee, B.-L. Lee, and W.-Y. Chung. Smartwatch-based driver alertness monitoring with wearable motion and physiological sensor. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 6126–6129. IEEE, 2015.

[75] B. G. Lee, B. L. Lee, and W. Y. Chung. Wristband-type driver vigilance monitoring system using smartwatch. *IEEE Sensors Journal*, 15(10):5624–5633, Oct 2015.

[76] L. B. Leng, L. B. Giin, and W.-Y. Chung. Wearable driver drowsiness detection system based on biomedical and motion sensors. In *SENSORS, 2015 IEEE*, pages 1–4. IEEE, 2015.

[77] Y. Li, F. Xue, L. Feng, and Z. Qu. A driving behavior detection system based on a smartphone's built-in sensor. *International Journal of Communication Systems*, 30(8), 2017.

[78] J. F. Lichtenauer, E. A. Hendriks, and M. J. Reinders. Sign language recognition by combining statistical dtw and independent classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):2040–2046, 2008.

[79] L. Liu, C. Karatas, H. Li, S. Tan, M. Gruteser, J. Yang, Y. Chen, and R. P. Martin. Toward detection of unsafe driving with wearables. In *Proceedings of the 2015 workshop on Wearable Systems and Applications*, pages 27–32. ACM, 2015.

[80] G. LLC. Use wrist gestures on wear, 2019.

[81] J. Lobo and J. Dias. Vision and inertial sensor cooperation using gravity as a vertical reference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1597–1608, 2003.

[82] H. J. Luinge and P. H. Veltink. Measuring orientation of human body segments using miniature gyroscopes and accelerometers. *Medical and Biological Engineering and computing*, 43(2):273–282, 2005.

[83] J. Ma and Y. Du. Study on the evaluation method of in-vehicle gesture control. In *Control Science and Systems Engineering (ICCSSE), 2017 3rd IEEE International Conference on*, pages 145–148. IEEE, 2017.

[84] X. Ma, C. Fellbaum, and P. R. Cook. Soundnet: Investigating a language composed of environmental sounds. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1945–1954, New York, NY, USA, 2010. ACM.

[85] A. Makni, H. Fourati, and A. Y. Kibangou. Adaptive kalman filter for mems-imu based attitude estimation under external acceleration and parsimonious use of gyroscopes. In *Control Conference (ECC), 2014 European*, pages 1379–1384. IEEE, 2014.

[86] S. Mao, D. Tao, G. Zhang, P. Ching, and T. Lee. Revisiting hidden markov models for speech emotion recognition. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6715–6719. IEEE, 2019.

[87] S. V. Marshall. Vehicle detection using a magnetic field sensor. *IEEE transactions on vehicular technology*, 27(2):65–68, 1978.

[88] B. Mortazavi, E. Nemati, K. VanderWall, H. G. Flores-Rodriguez, J. Y. J. Cai, J. Lucier, A. Naeim, and M. Sarrafzadeh. Can smartwatches replace smartphones for posture tracking? *Sensors*, 15(10):26783–26800, 2015.

[89] S. Nirjon, R. F. Dickerson, P. Asare, Q. Li, D. Hong, J. A. Stankovic, P. Hu, G. Shen, and X. Jiang. Auditeur: A mobile-cloud service platform for acoustic event detection on smartphones. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '13, pages 403–416, New York, NY, USA, 2013. ACM.

[90] O. Nonnarit, A. Barreto, et al. Gyroscope drift correction algorithm for inertial measurement unit used in hand motion tracking. In *SENSORS, 2016 IEEE*, pages 1–3. IEEE, 2016.

[91] U. B. of Labor Statistics. *American Time Use Survey*. 2015.

[92] T. Oron-Gilad, A. Ronen, and D. Shinar. Alertness maintaining tasks (amts) while driving. *Accident Analysis & Prevention*, 40(3):851–860, 2008.

[93] D. O'Shaughnessy. *Speech communication: human and machine*. Addison-Wesley series in electrical engineering: digital signal processing. Universities Press (India) Pvt. Limited, 1987.

[94] D. E. Phillips, R. Tan, M.-M. Moazzami, G. Xing, J. Chen, and D. K. Y. Yau. Supero: A sensor system for unsupervised residential power usage monitoring. *2014 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 0:66–75, 2013.

[95] C. A. Pickering, K. J. Burnham, and M. J. Richardson. A research study of hand gesture recognition technologies and applications for human vehicle interaction. In *2007 3rd Institution of Engineering and Technology Conference on Automotive Electronics*, pages 1–15, June 2007.

[96] A. Prasad, J. Sorber, T. Stablein, D. Anthony, and D. Kotz. Understanding sharing preferences and behavior for mhealth devices. In *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*, WPES '12, pages 117–128, New York, NY, USA, 2012. ACM.

[97] M. Rabbi, S. Ali, T. Choudhury, and E. Berke. Passive and in-situ assessment of mental and physical well-being using mobile sensors. In *Proceedings of the 13th International Conference on Ubiquitous Computing*, UbiComp '11, pages 385–394, New York, NY, USA, 2011. ACM.

[98] A. Ramachandran, S. Lumetta, E. Klee, and D. Chen. A recurrent markov state-space generative model for sequences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3070–3079, 2019.

[99] K. Rathi, D. Patil, S. Bhavsar, K. Jadhav, and S. V. Thakur. Gesture human-machine interface (ghmi) in home automation. 2017.

[100] U. Review. Safedrive: Detecting distracted driving behaviors using wrist-worn devices.

[101] L. Reyner and J. Horne. Falling asleep whilst driving: are drivers aware of prior sleepiness? *International journal of legal medicine*, 111(3):120–123, 1998.

[102] N. Rhinehart and K. M. Kitani. First-person activity forecasting with online inverse reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3696–3705, 2017.

[103] F. Sagberg, P. Jackson, H.-P. Krüger, A. Muzet, and A. Williams. Fatigue, sleepiness and reduced alertness as risk factors in driving. *Project Report, Transport RTD*, 2004.

[104] M. Sahidullah and G. Saha. Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition. *Speech Communication*, 54(4):543 – 565, 2012.

[105] S. Shen, H. Wang, and R. Roy Choudhury. I am a smartwatch and i can track my user's arm. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '16, pages 85–96, New York, NY, USA, 2016. ACM.

[106] J. Sterkenburg, S. Landry, and M. Jeon. Eyes-free in-vehicle gesture controls: Auditory-only displays reduced visual distraction and workload. In *Proceedings of the 9th International Conference on Automotive User Interfaces and Interactive Vehicular Applications Adjunct*, pages 195–200. ACM, 2017.

[107] S. S. Stevens. A Scale for the Measurement of the Psychological Magnitude Pitch. *Acoustical Society of America Journal*, 8:185, 1937.

[108] D. Tannen. *You just don't understand: Women and men in conversation*. Virago London, 1991.

[109] D. Tharini and J. Kumar. 21 band 1/3-octave filter bank for digital hearing aids. In *Pattern Recognition, Informatics and Medical Engineering (PRIME), 2012 International Conference on*, pages 353–358, March 2012.

[110] 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*, pages 1029–1040. ACM, 2015.

[111] E. Thomaz, C. Zhang, I. Essa, and G. D. Abowd. Inferring meal eating activities in real world settings from ambient sounds: A feasibility study. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, IUI '15, pages 427–431, New York, NY, USA, 2015. ACM.

[112] H. Truong, P. Nguyen, N. Bui, A. Nguyen, and T. Vu. Demo: Low-power capacitive sensing wristband for hand gesture recognition. In *Proceedings of the 9th ACM Workshop on Wireless of the Students, by the Students, and for the Students*, S3 '17, pages 21–21, New York, NY, USA, 2017. ACM.

[113] A. B. Ünal, D. de Waard, K. Epstude, and L. Steg. Driving with music: Effects on arousal and performance. *Transportation research part F: traffic psychology and behaviour*, 21:52–65, 2013.

[114] United States Department of Transportation. Using efficient steering techniques. available at `https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/steeringtechniques.pdf`.

[115] C. Wang, Z. Liu, and S.-C. Chan. Superpixel-based hand gesture recognition with kinect depth camera. *IEEE transactions on multimedia*, 17(1):29–39, 2015.

[116] T. S. Weisner, C. Matheson, J. Coots, and L. P. Bernheimer. Sustainability of daily routines as a family outcome. In *Learning in cultural context*, pages 41–73. Springer, 2005.

[117] B. L. Welch. The generalization ofstudent's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947.

[118] C. Xu, S. Li, G. Liu, Y. Zhang, E. Miluzzo, Y.-F. Chen, J. Li, and B. Firner. Crowd++: Unsupervised speaker count with smartphones. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 43–52, New York, NY, USA, 2013. ACM.

[119] C. Xu, P. H. Pathak, and P. Mohapatra. Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, pages 9–14. ACM, 2015.

[120] J. Yu, H. Zhu, H. Han, Y. J. Chen, J. Yang, Y. Zhu, Z. Chen, G. Xue, and M. Li. Senspeed: Sensing driving conditions to estimate vehicle speed in urban environments. *IEEE Transactions on Mobile Computing*, 15(1):202–216, 2016.

[121] Y. Zhao, P. H. Pathak, C. Xu, and P. Mohapatra. Demo: Finger and hand gesture recognition using smartwatch. In *MobiSys*, page 471, 2015.