Enhance User Safety in Personal Environment
When Interacting with Commodity Devices

by

Yucheng Yang

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Electrical and Computer Engineering)

at the

UNIVERSITY OF WISCONSIN–MADISON

2024

Date of final oral examination: 11/19/2024

The dissertation is approved by the following members of the Final Oral Committee:
Kassem Fawaz, Associate Professor, Electrical and Computer Engineering
Younghyun Kim, Associate Professor, Electrical and Computer Engineering,
Purdue University
Madhav Chitturi, Research Associate, Civil and Environmental Engineering
Parmesh Ramanathan, Professor, Electrical and Computer Engineering

Where there's a will, there's a way.

ACKNOWLEDGMENTS

After six and a half years in Madison, I have reached the conclusion of my Ph.D. journey—a journey of growth, learning, and discovery. Along the way, this profound experience has fostered my development and broadened my perspective, helping me become a better person. For this, I am deeply grateful to the many individuals who have supported me along the way.

First and foremost, I would like to express my heartfelt gratitude to my Ph.D. advisor, Professor Kassem Fawaz. His brilliance, resilience, and kindness have been a cornerstone of my journey. Despite the challenges he has faced, he consistently prioritizes his students, offering unwavering support and guidance. He is more like a friend to his students, and his mentorship has been the driving force throughout my Ph.D. journey. I am also deeply thankful to Professor Parmesh Ramanathan for his insightful guidance on the PedHat paper. My sincere thanks extend to Professor Younghyun Kim for his encouragement and guidance during the completion of my first paper. I would also like to thank Dr. Madhav Chitturi, Hesham Alyamani, Professor Ahn Sue, and Professor Yang Zhou (also a dear friend) for their assistance on the FHWA project. This project would not have been possible without your expertise and contributions.

To my labmates, thank you for making my academic life enjoyable and memorable. I am especially grateful to Jack West, my brilliant, focused and enthusiastic friend on the mute button project. Working with you and Professor Neil Klingensmith uncovered fascinating stories that made this project both successful and fulfilling. My gratitude also extends to Jingjie Li (now Professor Jingjie Li), Kyuin Lee (now Professor Kyuin Lee) and Di Wu (now Professor Di Wu) for their help during the early stages of my research. I would like to acknowledge both the "old faces"—Shimaa, Rishabh, Asmit, Ashish, Paul, Brian, Amrita, Varun, Harrison, and Yue—and the "new faces"—Guruprasad, Yash, Leo, and Shirley—who have brought energy and happiness to our lab. Your friendship and support have created a vibrant and welcoming environment. Special thanks to Shimaa, Rishabh, Asmit, and Ashish for their invaluable help

in solving many challenges during my thesis work.

Outside of academia, I am grateful to my friends in Madison, who made my time here truly enriching. Thank you, Zifan Liu, Hui Chen, Rui Wang and Bin Li, for the countless moments of joy and companionship. I also want to acknowledge my old friends—Weitao Wang, Fengyu Deng, Jiefeng Li, Hao Wu, Yaowei Huang, Jiachen Sun, Tianhao Huang, and Yihao Shao—for their enduring support. Special thanks to Jiefeng Li and Yaowei Huang for their daily memes that brightened my days. My gratitude also goes to my mentor during my 2024 summer internship at Meta, Xiaomin Zhang, and my peer, Lacey Liu. Your guidance was instrumental in the success of my projects and paved the way for my return to Meta.

Lastly, and most importantly, I want to thank my family and my partner, Elisa Ou, for their unwavering love and support. Elisa, since 2020, we have shared countless ups and downs, moments of happiness and sadness. Thank you for being my rock and an essential part of my life. To my father, Jianming Yang, and my mother, Liuying Xu, thank you for giving me a loving family and supporting my dreams, enabling me to embark on this journey from our small town. I am deeply grateful for the opportunities I have had to explore the world and pursue my passions. To my grandparents, who have always loved and cared for me, I will forever cherish your kindness and wish for your continued health and happiness.

This milestone is a reflection of the incredible people who have stood by me, and I dedicate this achievement to all of you.

— Yucheng Yang, December 2024

contents iv

list of tables ix

list of figures x

Abstract xiii

I Enhancing Physical Safety in Road Environment 1

1 Introduction 2
- *1.1 Sensing Platform and Data Collection* 2
- *1.2 Road Crossing Prediction* 3
- *1.3 Communication* 4
- *1.4 Presence Message Authentication* 4
- *1.5 Human Factors* 4

2 Sensing Platform & Data Collection 6
- *2.1 Design and Implementation of Data Collection App* 6
- *2.2 Data Storage* 8
- *2.3 Automatic Labeling of Data* 8
- *2.4 Data Collection Results and Pre-processing* 10
  - 2.4.1 Data Collection 10
  - 2.4.2 Trace Segmentation and Labeling 11

3 Road Crossing Prediction 13
- *3.1 Introduction* 13
- *3.2 Preliminaries* 16
  - 3.2.1 Coordinate Systems 16
  - 3.2.2 Motion Tracking 17
  - 3.2.3 Transformations 18

3.3 Part I: Pedestrian Heading Tracking 19
3.3.1 Limitations of Current Heading Tracking Methods . . 19
3.3.2 OHA Overview 20
3.3.3 Formal Analysis 22
3.3.4 OHA Evaluation 24
3.4 Part II: Pedestrian Road Crossing Prediction 28
3.4.1 *PedHat* Overview 29
3.4.2 Machine Learning Model Details 30
3.4.3 Model Development 32
3.5 Implementation Details 34
3.5.1 Android-based Implementation 34
3.6 Evaluation of Pedestrian Crossing Prediction 35
3.6.1 Q1. Crossing Identification of *PedHat* 36
3.6.2 Q2. Detailed Analysis for *PedHat* 38
3.6.3 Q3. System Overhead 44
3.7 Related Work 45
3.7.1 Infrastructure and Vehicle-based Approaches 45
3.7.2 Infrastructure-free Approaches 47
3.8 Limitations 48

4 Communication 50
4.1 Single-hop Communication 50
4.1.1 Wireless Communication Options 50
4.1.2 Single-hop Broadcasting Mechanism 51
4.1.3 Real-world Experiments 52
4.2 Multi-hop Communication 54
4.2.1 Distance based forwarding algorithms 54
4.2.2 Implementation of Wi-Fi Aware based Message Communication 56
4.3 Experiments and Simulation 57
4.4 Broadcasting Alerts Limitations 60

5 Human Factors 61

5.1 Introduction 61
5.2 Related Work 63
5.2.1 Pedestrian Crossing Prediction 63
5.2.2 Development of ADAS and Warning Modalities 63
5.2.3 Driving Simulators 65
5.3 Experiment Methods 65
5.3.1 Recruitment Procedure 65
5.3.2 Experiment Setup 66
5.3.3 Post-session Questionnaire 71
5.4 Data Analysis and Findings 72
5.4.1 Participant Reaction in Driving Session 72
5.4.2 Participant Feedback in Post-Session Questionnaire . . 77
5.5 Discussion 80
5.5.1 Driver's Excessive Anxiety 81
5.5.2 Enhancing Driver Awareness 81
5.5.3 Trust when see 82
5.6 Limitations 82
5.7 Conclusions 83

II Bridging Physical and Digital Safety 84

6 PEDRO: Secure Pedestrian Mobility Verification in V2P Communication using Commercial Off-the-shelf Mobile Devices 85
6.1 Introduction 85
6.2 System and Threat Models 87
6.2.1 System Model 88
6.2.2 Threat Model 89
6.3 Pedro Protocol 90
6.3.1 Measurement 90
6.3.2 Verification 91
6.4 Experiment and evaluation 94
6.4.1 RTT Error Model 95

6.4.2 Distance and Time Thresholds 96
6.4.3 Real-world Case Study 99
*6.5 Related work*100

III Enhancing Digital Safety in Video Conferencing Apps 101

7 Are You Really Muted?: A Privacy Analysis of Mute Buttons in Video Conferencing Apps102
*7.1 Introduction*102
*7.2 Related Work*105
*7.3 User Study*108
7.3.1 Study Design 109
7.3.2 Findings 111
*7.4 Analysis of Mute Button*115
7.4.1 Overview of VCAs and Platforms 115
7.4.2 Analysis Methodology 118
7.4.3 Findings 120
*7.5 Webex Case Study*122
7.5.1 Methodology for Traffic Interception 123
7.5.2 Findings for Traffic Interception 125
7.5.3 Classification of Background Activities 126
*7.6 Discussion*132
7.6.1 Limitations of the Study 133
7.6.2 An OS-Level Mitigation 134
7.6.3 VCA Privacy Policies 135
*7.7 Appendix*136
7.7.1 Model Architecture 136
7.7.2 Chromium API 137
7.7.3 YouTube Video List 137
7.7.4 YouTube Video List I 137
7.7.5 YouTube Video List II 138
7.7.6 Windows API 139

7.7.7 Music Mode Correlation Results 140
7.7.8 Window Length 5 and 10 141
7.7.9 Codebooks 142

IV Future work and Conclusions **143**

8 Future work144
*8.1 Privacy-Preserving Multi-Sensor Data Collection for Pedestrian Safety*144
*8.2 Generalizing Predictive Models for Diverse Pedestrian Safety Applications*144

9 Conclusion146

Bibliography148

LIST OF TABLES

3.1 Precision and Recall rate on the test dataset of models trained with varying *lookback* lengths. 33
3.2 Trade-off between time-to-crossing and false alarm rate by varying past consulted predictions n. 10 predictions equal to 1s and negative TTC represents crossing alerts happen after pedestrian has crossed the road edge. 39
3.3 Device specs, execution time and resource usage when running *PedHat* in active mode. 44
3.4 Comparison of Time-to-Crossing for pedestrian crossing detection. 46

4.1 One hop latency testing in multiple scenarios with safety message serialized over multiple packets. 53

5.1 Age distribution 66
5.2 Pedestrian jaywalking events for each driving session. 70
5.3 Comparison of Scenarios with the Base Scenario 77
5.4 36 Participants' Perceptions toward False Alarms across Modality. Abbreviations: A+V = Audio & Visual, V+T = Visual & Tactile, A+V+T = Audio, Visual & Tactile. 80

7.1 The main questions used in the user study. 109
7.2 A summary of the VCAs we studied. ✓: native app ∘: web-based app ✕: No implementation. 116
7.3 Dataset distribution, development set (training and validation) and evaluation set (subset 1 and 2). 130
7.4 Codebook for responses to Survey Question Q3 141
7.5 Codebook for responses to Survey Question Q1 142
7.6 Codebook for responses to Survey Question Q2 142

LIST OF FIGURES

1.1 Framework for exchanging safety information among road users. 3

2.1 Data collection diagram 6
2.2 Stop logging logo 7
2.3 Privacy-aware design 7
2.4 Participant walking trace and potential crossing behavior caused by GPS drift. "Crossed" twice in a short period. 12

3.1 Smartphone's LCS initially aligned with GCS, (a) roll first by ψ, (b) pitch second by θ, and (c) yaw third by ϕ, to achieve (d) last orientation. 17
3.2 Pedestrian heading θ_h represents angle between PRF and GCS. A smartphone in front of the pedestrian in LCS. This image was created with the assistance of DALL·E 2. 19
3.3 OHA processing pipeline overview 21
3.4 Overall heading estimation error CDF for three common smartphone placements 25
3.5 Heading estimation mean and interquartile range errors across 9 movement scenarios from 5 participants. 26
3.6 Participant places the smartphone in his trouser pocket and walks in "S"-shaped paths. 27
3.7 OHA heading errors related to different levels of GPS signal blockage. 28
3.8 Pedestrian Road Crossing Prediction system overview. 29
3.9 Distribution of Magnetic field magnitude measured by ten pedestrians while walking outdoors. 32
3.10 Examples of three behaviors, (a) actual crossing (ac), (b) potential crossing (pc), and (c) dangerous behavior (db) labeled by both annotators. 37
3.11 We visualized six walking traces on a map, each marked with annotated behaviors and prediction results. 42
3.12 SHAP analyzes *PedHat* model input features. 43

4.1 Broadcasting crossing alerts latency and transmission success rate using Wi-Fi Aware beacon stuffing at varying distances. 54
4.2 NLoS Scenario for multi-hop communication 56
4.3 LoS Scenario for multi-hop communication 57
4.4 Traffic simulation using SUMO. Small yellow block represents moving vehicles. 57
4.5 Network simulation with back-off algorithm. 59
4.6 Simulation of network traffic without back-off algorithm. 59

5.1 Visual warning 67
5.2 Full-scale Driving simulator 67
5.3 Overview of urban driving scenarios 68
5.4 Run Schematic: A four-loop driving route. 69
5.5 Simulated Driving Environment Scenarios 69
5.6 Sequence of events in one driving session, including six types of events. 71
5.7 (a) Stopping vehicles count and (b) brake initiation count upon road events across scenarios 73
5.8 (a) Average distance for each event when car stops and (b) average speed after each event for each scenario. 75
5.9 Q: How clear were the pedestrian crossing alerts? 78
5.10 Q: How timely were the the pedestrian crossing alerts? 78
5.11 Q: Would you recommend this pedestrian warning system to other drivers? 78
5.12 Q: How would you rate your overall confidence in the system for pedestrian crossing alerts? (4 is the highest) 78

6.1 Traveling verifiers on the road (V_1 and V_2) attempts to verify the mobility of the provers (P_1 and P_2) and verify only moving pedestrian P_1. 88
6.2 *Pedro*'s two stage verification protocol. 90
6.3 Verification stage of *Pedro* with $n_i = 3$. The prover must meet two requirements to be verified: th_d and th_t check. 92

6.4 RTT based distance measurement and Error Distributions. 95
6.5 Mean inter-region time under varying (a) verifier's speed, (b) maximum wireless range, (c) prover's speed and (d) vehicle arrival interval with different th_d. 97
6.6 (a) Distribution of inter-region time of moving prover and passive attacker. (b) EER of verification with $th_t = 13$. 98
6.7 Real-world experiment result. 99

7.1 The distribution of the codes about reasons users reported for using the mute button as extracted from answers to *Q1*. 111
7.2 The distribution of the codes about the background activities as extracted from answers to *Q2*. 112
7.3 The distribution of the codes about the users' understanding of the mute button operation from answers to *Q3*. 113
7.4 The distribution of responses to Q4. The statements *S1-S5* are defined in Table 7.1. 114
7.5 The distribution of responses to Q5. The statements *S1-S5* are defined in Table 7.1. 115
7.6 Data flow of audio bytes within a Windows 10 VCA. This pipeline is generalizable across the Windows platform. Our system attaches to the bolded modules. 122
7.7 Correlation between audio gain reported by Webex and input audio signal power level (in dbA) when noise removal mode is enabled. 125
7.8 Clusters of audio statistics data color coded by background activity type. Clusters are visually separable. 129
7.9 Background activity classifier performance with window length = 7132
7.10 The classification architecture for the background activities. . . . 136
7.11 Correlation between audio gain reported by Webex and input audio signal power level (in dbA) when music mode is enabled. 140
7.12 Background activity classifier performance with window length = 3141
7.13 Background activity classifier performance with window length = 5141
7.14 Background activity classifier performance with window length = 10142

ABSTRACT

Safety, both in physical and digital environments, is a fundamental concern for individuals, protecting them from harm, injury, or loss. In the physical realm, safety involves measures to protect individuals from real-world risks such as road crashes, where tools like traffic signs, connected vehicles, and driver alert systems play a critical role in preventing collisions. In the digital domain, safety focuses on protecting personal information and digital assets, with regulations like GDPR, secure authentication methods, and data control tools helping to mitigate privacy and security risks. Bridging these domains is the interplay between physical and digital safety, where device interactions become pivotal.

The proliferation of smart and IoT devices has made multi-sensor interactions ubiquitous. These devices collect a variety of data, such as audio, visual, and environmental information, and provide feedback to users for tasks ranging from work and study to entertainment. While these devices serve as valuable tools to enhance user safety, they may fail under certain conditions, posing risks to both physical and digital well-being.

This thesis investigates how commodity devices can be leveraged to enhance user safety in personal environments by exploring the intersection of device capabilities and user protection. Aiming to address safety challenges across physical and digital domains, this work ultimately seeks to foster a safer and more secure personal environment. Specifically, it examines user safety through three interconnected dimensions: **physical safety**, **bridging the physical and digital environments**, and **digital safety**. Each dimension highlights critical challenges and presents solutions that utilize commodity devices to enhance user safety.

In physical safety, we present PedHat [158], a real-time pedestrian road-crossing prediction system that uses commodity devices to issue crossing alerts and broadcast warnings to nearby drivers, enhancing pedestrian safety. Utilizing a full-scale driving simulator, we further analyzed the effectiveness of such crossing alerts with multi-modal warning types [11]. We found that

crossing alerts can significantly improve driver responses, but occasional false alarms posed challenges by reducing trust and inducing alert fatigue in some drivers.

To address the bridge between physical and digital environments, we introduce PEDRO [157], a secure pedestrian mobility verification mechanism for Vehicle-to-Pedestrian (V2P) communications. PEDRO authenticates senders' identities by requiring physical mobility movements, ensuring that only users actively moving in the physical environment can verify their presence. This mechanism effectively mitigates the risk of stationary adversaries impersonating legitimate users, enhancing trust and security in V2P interactions.

In the realm of digital safety, we examine the privacy risks associated with Video Conferencing Apps (VCAs), particularly during the increased reliance on these platforms in the pandemic era [159]. Our investigation uncovered how some VCAs, like older versions of Cisco Webex, accessed users' microphones without proper notification, even when the mute button was activated. By analyzing audio access practices of popular VCAs, we identified potential privacy vulnerabilities and proposed recommendations to protect users' data safety in digital environments.

Part I

Enhancing Physical Safety in Road Environment

Fatalities among vulnerable road users have been increasing over the last several years. In this part, we aim to improve the physical safety status-quo by making vehicles aware of potential conflicts with pedestrians and cyclists or the need for caution. Realizing this framework requires addressing three main challenges: detecting the intent of road users; providing real-time, secure, and infrastructure-free communication of user intent; and designing usable interfaces to make the road users aware of their surroundings. Detecting the intent of pedestrians and cyclists is challenging as they often jaywalk, cross at mid-blocks, and move unpredictably around vehicles. Communicating the inferred intent to other road users requires an ad hoc network that does not rely on infrastructure to be practical to deploy and use. A mobile ad hoc network, while providing a self-organized network architecture, exhibits challenges in coordinating transmissions across different nodes and authenticating the source of communication in the absence of a trust base. Finally, designing a usable interface requires deciding on the information to communicate to the road user, the severity of the alerts, and the presentation modality.

The overall objective was to develop and demonstrate a Mobile Ad hoc Network (MANET) for safety information exchange among road users, addressing intent prediction, networking, and user interactions. The framework, illustrated in Figure 1.1, operated in three modes: vehicles, pedestrians, and cyclists, using a smartphone app. In pedestrian mode, the app collects and broadcasts mobility information, alerting users of nearby vehicles. In vehicle and cyclist modes, the app receives broadcast alerts to avoid possible collisions. All modes include a spatio-temporal bounding box to ensure message validity.

1.1 Sensing Platform and Data Collection

We developed a smartphone app to collect movement data for developing movement models and verification functions. IRB approval and participant recruitment were obtained to ensure ethical data collection, privacy, and participant compensation. High-frequency location and sensory data were collected

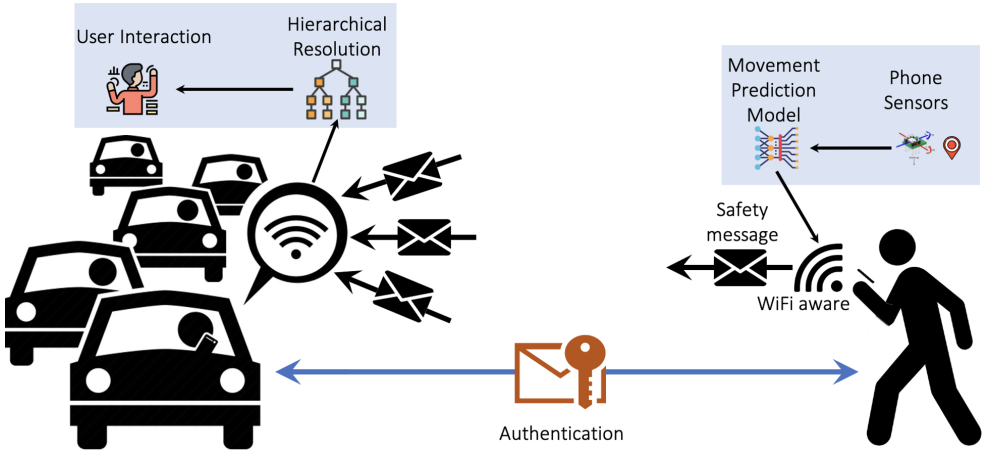


Figure 1.1: Framework for exchanging safety information among road users.

using the app to provide detailed movement information. We ensured data was anonymized and processed locally to maintain privacy. Participants were recruited from our campus, provided with privacy notices, and compensated for their participation. The app collected high-frequency location and sensory data, relying on existing libraries to identify travel modes. The data was uploaded securely for further analysis, providing a foundation for subsequent tasks.

1.2 Road Crossing Prediction

We developed models to predict pedestrian movements, alerting users of potential conflicts. General and personalized models were created to predict pedestrian crossings at mid-block sections and intersections. These models enhanced the system's ability to prevent accidents by accurately forecasting user movements. General models considered various factors, while personalized models used individual historical data to enhance accuracy. These algorithms used sensory and mobility data to classify behavior and improve system usability by reducing unnecessary alerts and focusing on genuine threats.

1.3 Communication

We designed and implemented a network to efficiently disseminate information using a spatio-temporal bounding box. Single-hop communication utilized Wi-Fi Aware for pedestrians, cyclists, and vehicles. A multi-hop network was implemented with a token-based leadership mechanism to maintain message persistence, ensuring reliable communication across different road users. WiFi Aware is used to support higher-rate connectivity for trust-related information exchange. A location-centric message persistence mechanism was implemented using a token-based leadership system among vehicles, ensuring messages remained available in specified regions until they expired.

1.4 Presence Message Authentication

We addressed authentication of message sources in a MANET using a proof-of-road-use concept and public key cryptography. This ensured that only legitimate road users could send presence messages, preventing malicious activity and enhancing trust in the system. A public key-based system was designed to identify road users anonymously. Public keys were recycled to protect privacy and maintain trust relationships, preventing long-term tracking and ensuring dynamic trust establishment. A verifier function was developed using sensory data to establish trust relationships between vehicles and cyclists. The function compared sensory sequences to determine matching movement patterns, leveraging cyclist movement properties to establish trust effectively. A blockchain-like technology was designed to build a distributed mobility log for pedestrians, verifying pedestrian-like movement and establishing trust relationships, preventing malicious stationary attackers from deceiving the system.

1.5 Human Factors

We determined in-vehicle information needs for potential conflicts with pedestrians and cyclists through a human factors study. A survey of traffic safety

experts helped prioritize conflicts and design effective communication methods. The app processed presence messages to identify potential conflicts and prioritize information for drivers. Real-time data processing and historical crash data were used to create a hierarchical map of potential conflicts, conveyed to drivers through the app's interface, ensuring drivers received clear and prioritized alerts, enhancing their ability to respond appropriately.

The first step of the project was to build a data collection platform to collect data from road users. Fig. 2.1 shows the flow of the data collection framework which we have developed. This framework includes an Android app that runs on the client side and a server component that handles the collected data. The app encrypts the collected data on the client side for privacy protection and automatically uploads them to our Amazon “Simple Storage Service" (S3) server for temporary storage. The data is stored in encrypted format on S3 storage; we download the data to our local servers for further processing/analysis after decryption.

2.1 Design and Implementation of Data Collection App

This task involved the development of an Android application to collect and upload user’s sensory information. The main functionality of the app is to collect raw position and sensory data from study participants’ smartphones, including:

- accelerometer, gyroscope, light level, proximity, pressure, and temperature which requires the user permission;
- location information comprising the latitude, longitude, and accuracy;

Figure 2.1: Data collection diagram

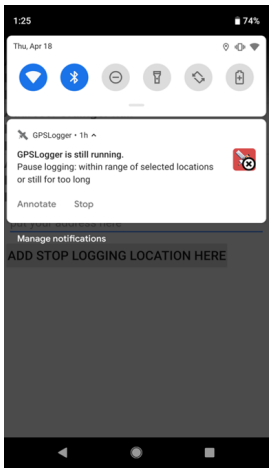


Figure 2.2: Stop logging logo

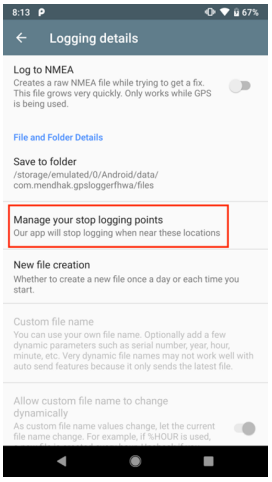


Figure 2.3: Privacy-aware design

- user's activity from the ActivityRecognition API, which can be on foot, on vehicle, still, etc.

The app then compresses and uploads the files, describing the collected sensory data, to Amazon S3 instance for temporary storage. This app, listed as "GPSLogger_FHWA"[1] in Google Play Store, serves as the main data collection tool. The source code can also be found in our GitHub repository[2].

To improve its usability, this app includes the following functionalities as evident from Fig. 2.2:

1. stop logging data when user is stationary;
2. display notification for participants;
3. log data in increments of 15 minutes;
4. restart automatically after stop logging or reboot; and
5. upload data only while connecting to Wi-Fi.

[1] `https://play.google.com/store/apps/details?id=com.mendhak.gpsloggerfhwa`
[2] `https://github.com/kmfawaz/fhwa`

We also added privacy-enhancing features to the app, as shown in Fig. 2.3. First, we identify the different participants only through the hashed serial number of their devices; we do not store direct PII that link the collected data to a participant. The app asks the participants to set their points of interest as to block logging within 150m of each point. As specified earlier, data files are encrypted during transit and cloud storage. We obtained IRB approval of the data collection app in April 2019. After completing the 2-month data collection period, we compensate each participant with an Amazon gift card for $40.

2.2 Data Storage

As we mentioned above, the compressed and encrypted raw position and sensory data will be uploaded and temporarily stored in our Amazon S3 server. The AWS "Simple Storage Service" provides better connectivity compared to uploading the data to our local servers. From 2020 to 2022, we recruited 60 participants from campus, who participated in the data collection from two weeks to two months. After data collection, we transferred all participant data collected from AWS S3 to our local servers and cleaned the cloud data storage to improve data safety.

2.3 Automatic Labeling of Data

Using the raw data collected from the 13 participants in the first period, we developed an algorithm to automatically label the road crossing events. We will use these labeled data to train the classifiers to predict road crossing events later. Due to sensory and localization errors, as well as inaccuracies in the Android APIs for activity recognition, this task proved to be more challenging than we expected. Eventually, we combined several features from Google Activity Recognition, speed, accuracy provided by GPS and the distance to the road center extracted from OpenStreetMap (OSM) to label the road crossing events. OpenStreetMap provides a topological data structure composed of nodes, ways, relations, and tags to store roads' information. After loading the

map of the city of Madison into a PostGIS-enabled PostgreSQL database, we can calculate the distance to the nearest road given a coordinate by querying the database. Generally speaking, we classify a sequence of locations as a road crossing event if a user is moving with a walking speed towards the center of the road. The details of extracting a valid road crossing event from the raw position data are elaborated in Algorithm 1.

Algorithm 1: Automatic road crossing labeling

```
Result: Labeled Road Crossing Events
Extract GPS logging points as list L_0;
for each GPS point p in L_0 do
    Calculate nearest road r and distance to road center d of p from
     OpenStreetMap using PostgreSQL;
    if d ≤ 10 & r ≠ railroad
    & speed ≤ 10 & mode ≠ DRIVING then
        add p into L_1;
    end
end
for each GPS point p in L_1 do
    if d ≤ 1 then
        set n as central point;
        extract [n − 30, n + 10] points as event_i;
    end
end
```

After identifying the the road crossing event, the next step is to extract sensor data associated with them. The head of a road crossing event is 30 location logging points before approaching the center of the road, and the tail is defined as 10 location logging points after approaching the center of the road. The tail part is needed to reduce the false positive labeling events caused by GPS error. Then, we extract all the raw sensor data between the head and tail; a valid record is defined as the raw sensor data tagged by a crossing event. The example labeled sensor data is in the form of <TIMESTAMP, SENSOR TYPE, SENSOR VALUE>. Finally, we manually check the extracted road crossing events for validity.

From June 2019 to August 2019, we automatically labeled and then manu-

ally verified approximately 1500 road crossing events in total. From September 2019 to October 2019, we added 680 labeled and verified road crossing events. More participants will be recruited to collect road traveling data for building a larger data set that can be used for building the road crossing prediction model.

Theoretically, one researcher can label road crossing events only with GPS points; errors in GPS readings, however, made this task much more challenging. We applied a Kalman filter to smooth the GPS measurements using readings from the IMU sensors. In our setting, a collected trajectory from the user might contain several noisy readings. After projecting the accelerometer readings to the geodetic coordinate system, we were able to smooth the GPS readings using a Kalman filter. This filtering made it easier to label the road crossing event semi-automatically.

2.4 Data Collection Results and Pre-processing

We curated a dataset of walking traces from 60 individuals in their daily lives, following a process of data cleaning, labeling and feature extraction. We will describe how we construct and refine our model using this dataset in Section 3.4.3.

2.4.1 Data Collection

We collected traces of IMU and GPS data when participants were walking in their daily lives. Our data collection protocol was approved by our institute's IRB. From 2020 to 2022, we recruited 60 participants from campus, who participated in the data collection from two weeks to two months. Our study began with a consent phase, where we gave an overview of the study and explained our data protection measures and participants' rights. Participants who consented to join the study installed our data collection app on their personal smartphones and kept it active in the background. We developed this app based on GPSLogger[3].

[3]https://github.com/mendhak/gpslogger

As our data collection might accidentally collect sensitive GPS information, we implemented multiple privacy protection measures. First, each participant can select points of interest (POI) to opt out of the collection, such as homes, offices, and other places. Second, our app provided an opt-out button to allow participants to manually stop data logging on demand. In addition, we removed any possible identifier from the uploaded data, associating the data only with hashed advertisement IDs.

We compensated each participant who completed the two-month session with a $40 Amazon gift card. In total, we collected around 755 hours of walking data (2.72 million GPS data points and corresponding IMU data after filtering, which will be discussed in the following).

2.4.2 Trace Segmentation and Labeling

The daily walking traces comprise pedestrians' hours-long GPS and sensor data collected when they walked. However, our GPS recording suffered from occasional GPS signal loss and was interrupted when participants opted out. Therefore, for each pedestrian, we segmented all location traces by time intervals into smaller sessions, referred to as **road-use sessions**. Each road-use session represents a complete and continuous (with signal loss shorter than 15 seconds) outdoor walking event, which consists of GPS and sensory data. In total, we extracted around 2,000 road-use sessions, and each road-use session's length ranges from 1 minute to 28.6 minutes.

The next task was labeling road crossings in the pedestrians' walking traces, which we found challenging for two reasons. First, it is difficult to determine every actual road-crossing behavior, with inherent GPS inaccuracies and unknown road width information. Second, it is difficult to differentiate participants' potential road-crossing behaviors from GPS drifts. Therefore, we cleaned and labeled the traces to minimize the influences of GPS inaccuracies and drifts that bias the model, following the following steps:

Step 1. We defined and labeled a "true crossing event" as the period from when a pedestrian **shows potential to cross** to when they **reach the road center**. Note that we do not include the time period after reaching the road

center, as doing so may delay the early detection of pedestrian road crossing. We determined the start of crossing potential by detecting turnings from GPS bearing change if the pedestrian turned to cross. For those who did not turn, we defined it as occurring five seconds before the pedestrian reached the road center. We consider five seconds to be a reasonable timeframe for a pedestrian to reach the road center once they begin crossing.

Step 2. We removed certain events where the road-crossing behavior is ambiguous, such as the example shown in Figure 2.4. Specifically, we ruled out the sessions where pedestrians returned to the original side of the road immediately after crossing. Note that our cleaning aims to label data with higher confidence. While map matching techniques [109] label two crossing events for the trace in Figure 2.4, our cleaning method will remove such trace due to low confidence.

Our processed data comprises timestamped and labeled walking traces, including when the pedestrian is not crossing and when their crossing has high confidence.

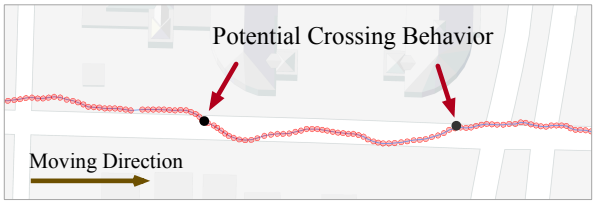


Figure 2.4: Participant walking trace and potential crossing behavior caused by GPS drift. "Crossed" twice in a short period.

3 ROAD CROSSING PREDICTION

Pedestrian heading tracking enables applications in pedestrian navigation, traffic safety, and accessibility. Previous works, using inertial sensor fusion or machine learning, are limited in that they assume the phone is fixed in specific orientations, hindering their generalizability. We propose a new heading tracking algorithm, the Orientation-Heading Alignment (OHA), which leverages a key insight: people tend to carry smartphones in certain ways due to habits, such as swinging them while walking. For each smartphone attitude during this motion, OHA maps the smartphone orientation to the pedestrian heading and learns such mappings efficiently from coarse headings and smartphone orientations. To anchor our algorithm in a practical scenario, we apply OHA to a challenging task: predicting when pedestrians are about to cross the road to improve road user safety. In particular, using 755 hours of walking data collected since 2020 from 60 individuals, we develop a lightweight model that operates in real-time on commodity devices to predict road crossings. Our evaluation shows that OHA achieves 3.4 times smaller heading errors across nine scenarios than existing methods. Furthermore, OHA enables early and accurate detection of pedestrian crossing behavior, issuing crossing alerts 0.35 seconds, on average, before pedestrians enter the road range.

3.1 Introduction

Tracking an individual's heading enables important applications, including navigation, safety, entertainment, and accessibility [29, 8, 167, 69]. For instance, accurately tracking pedestrian heading, not limited to using mobile devices, facilitates the prediction of pedestrian behavior on roads [128, 124, 70, 99, 163, 49, 77, 119]. Additionally, heading tracking methods based on inertial sensors installed on portable devices provide continuous and practical heading estimates without requiring dedicated devices such as LiDAR or stereo camera systems, garnering significant interest from researchers in both industry and academia [151, 52].

Tracking pedestrian heading involves continuously tracking an individual's facing direction on a 2-D flat plane, typically the horizontal plane of the global coordinate system (GCS). It is important to note that tracking pedestrian heading using mobile or wearable devices differs from tracking the orientation of these devices in GCS, as explored in previous works such as MUSE [127] and A^3 [169]. For example, a pedestrian could be walking from south to north on a road while swinging a smartphone. In this case, smartphone orientation estimation would indicate the device's dynamic orientation relative to the GCS, commonly represented by Euler angles (roll, pitch, yaw). On the other hand, tracking pedestrian heading should accurately show that the pedestrian is moving from south to north, regardless of how the smartphone is oriented.

Existing approaches to estimating pedestrian heading through IMU (Inertial Measurement Unit) employ a two-stage pipeline: first, they estimate the horizontal plane using gravity or magnetic fields, and then integrate the gyroscope to track relative heading changes [92, 140, 35]. These approaches hinge on a critical assumption: the phone must remain static relative to the pedestrian body. Once the phone moves, especially during swinging motions, frequent recalculations of the horizontal plane are necessary, introducing intractable errors into the system, as we describe in Section 3.3.4. Similarly, researchers have explored tracking pedestrian trajectories using IMU sensors through methods such as dead reckoning [46] or machine learning [28]. However, these methods suffer from drifts and inaccuracies inherent to on-device sensors, preventing them from accurately tracking fine-grained pedestrian heading or movements over extended periods [56].

We propose a new heading tracking algorithm, Orientation-Heading Alignment (OHA), which leverages a key insight: people tend to carry smartphones in certain attitudes due to habits, whether swinging them while walking, stashing them in pockets, or placing them in bags. These attitudes or relative orientations, defined as the smartphone's orientation relative to the human body rather than GCS, mainly depend on the user's habits, characteristics, or even clothing. For instance, regardless of which direction a pedestrian faces, they swing the smartphone in their habitual manner. For each smartphone attitude, OHA maps the smartphone orientation to the pedestrian heading.

Because the attitudes are relatively stable for each person (e.g., holding a smartphone in the right hand and swinging), it is possible to learn the mappings efficiently from coarse headings and smartphone orientation. Previous research [87, 156, 81] has noted a similar insight but adopted a different approach for heading tracking: collecting IMU and accurate heading information for multiple smartphone attitudes and training a machine learning model to predict the heading. However, due to device discrepancies and varying user behaviors, it is not feasible to construct a machine learning model that generalizes to all possible smartphone attitudes.

To anchor our heading estimation algorithm in a practical scenario, we apply OHA to a challenging task: predicting when pedestrians are about to cross the road—an important problem for improving road user safety [136, 163, 164]. This task, which requires accurate and timely predictions of pedestrian crossings, is further complicated by the diverse crossing patterns of pedestrians and the complexity of road layouts. Based on the OHA heading, we propose *PedHat*, a lightweight, infrastructure-free system that predicts when a pedestrian is about to cross the nearest road and issues crossing alerts. *PedHat* incorporates a lightweight model that accepts OHA headings as inputs and operates in real-time on user devices to predict road crossings. We developed this model using data we collected since 2020 from 60 individuals, each contributing two months of traces, covering 755 hours of walking data as we described in Sec. 2.4. We generated coarse-grained labels for these data traces, identifying and eliminating uncertain crossing behavior caused by severe GPS drifts. We integrated this functionality into a lightweight Android app that requires no special hardware, infrastructure, or user involvement. This app autonomously predicts road crossings and alerts nearby road users, making it easier to deploy and adopt than other solutions requiring dedicated infrastructure or special hardware.

We evaluated both OHA and *PedHat* in real-world settings, and here are our evaluating highlights:

• We compared OHA with two other baseline methods in outdoor settings: 1) GPS and 2) integrating horizontal components of the gyroscope. We tested them across nine distinct scenarios, which included three walking patterns

and three smartphone placements. OHA achieved 3.4 times smaller heading errors across all scenarios than the gyroscope integration method. Moreover, OHA maintains low heading errors, even when participants walked in "Multiple S-shaped Paths" while swinging their smartphones. Additionally, OHA demonstrates reliable performance even in conditions of low GPS accuracy or unavailable GPS bearing.

• We conducted a user study with 25 additional participants in a semi-controlled real-world environment to evaluate *PedHat* in two key aspects: the accuracy in identifying road crossing events and the promptness of the crossing alerts. Utilizing fine-grained labels from two annotators, we found that *PedHat* achieved a precision of 86.9% and a recall of 93.6% in identifying road crossing events. Our evaluation also shows that *PedHat* issues a crossing alert on average 0.35 seconds before a pedestrian crosses the road edge, delivering performance comparable to camera-based methods that identify pedestrian pose to predict crossings in Line-of-Sight situations [49, 163, 66, 70, 153].

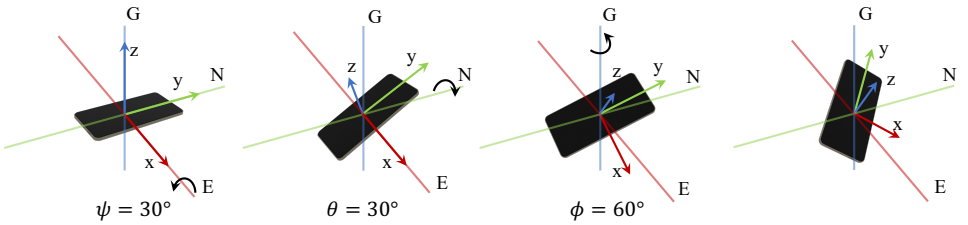
• We measured the performance overhead of *PedHat*, including OHA on multiple Android devices. We found the average execution time for each prediction is around 10 ms, and the average battery consumption over 30 minutes is about 95 mAh in idle mode and about 352 mAh in active mode.

3.2 Preliminaries

Before describing pedestrian heading tracking, we provide the necessary background on smartphone coordinate systems and how to transform one coordinate system into another.

3.2.1 Coordinate Systems

Smartphones report their dynamic state in three axes relative to the screen, which we call the local coordinate system (LCS). In this project, we follow the convention of the Android platform when describing the three axes. The X-axis is horizontal and points to the right of the screen, the Y-axis is vertical and points to the screen top, and the Z-axis points outward from the screen



(a) roll (b) pitch (c) yaw (d) final orientation

Figure 3.1: Smartphone's LCS initially aligned with GCS, (a) roll first by ψ, (b) pitch second by θ, and (c) yaw third by ϕ, to achieve (d) last orientation.

face. The global coordinate system (GCS) is a three-dimensional reference frame used to locate objects on the Earth. We use the following convention to represent axes in the GCS: the X-axis aligns with the east direction (E), the Y-axis aligns with the magnetic north direction (N), and the Z-axis points opposite to the direction of gravity (G). Due to differences in coordinate systems, raw IMU sensor measurements from a smartphone describe the phone's dynamic state in the LCS rather than the GCS.

3.2.2 Motion Tracking

Tracking the phone motion in GCS requires describing its movement along six degrees of freedom (6DOF), including translational and rotational movements. The translational degrees of freedom represent movement along the X, Y, and Z axes, while the rotational degrees of freedom correspond to rotation around these axes, often referred to as Euler angles (roll, pitch, and yaw). The phone's orientation relative to the Earth allows for transforming IMU sensor readings from LCS to GCS.

Euler angles provide an intuitive way to describe an object's orientation in 3D space. They refer to a set of three angles, ψ, θ, ϕ, that define the orientation of a rigid body in three-dimensional space relative to a fixed reference frame, in our case, the GCS. Specifically, they are the rotation angles required to transform one coordinate system to another following a specific rotating order. In our setting, Euler angles describe how we rotate a phone that aligns with the GCS (x, y, z aligns with E, N, G) to its current orientation following a

standardized order. For instance, the roll-pitch-yaw convention, also known as ZYX convention, represents a rotating sequence to rotate the GCS into LCS, following a roll (X axis, East) first by ψ degrees, pitch (Y axis, North) second by θ degrees, and yaw (Z axis, opposite of gravity) last by ϕ degrees order.

Figure 3.1 illustrates the rotation of a smartphone, initially aligned with the GCS, to its current orientation. Initially, the smartphone's screen coordinate system (LCS) is aligned with the GCS, where x corresponds to the E direction, y to the N direction, and z to the G direction. First, the smartphone rotates around the E direction by ψ (roll) degrees, following the right-hand rule. Then, the smartphone rotates around the N direction by θ (pitch) degrees. It is important to note that after the first rotation, the y axis is no longer aligned with the N direction. Finally, the smartphone rotates around the G direction by ϕ (yaw) degrees, resulting in its final orientation.

3.2.3 Transformations

Orientation information can also be represented as rotation matrices. Rotation matrices are 3×3 matrices that transform a coordinate or a vector from one frame (e.g., LCS) to another (e.g., GCS).

A rotation matrix can be expressed using the Euler angles. In the roll-pitch-yaw convention, there exists a one-to-one correspondence between Euler angles and rotation matrices under restricted domain ranges for the roll, pitch, and yaw angles. In such a case, the rotation matrix can be written in the following form:

$$R_{rpy}(\phi, \theta, \psi) = R_Z(\phi) R_Y(\theta) R_X(\psi), \tag{3.1}$$

where ϕ, θ, and ψ represent the roll, pitch, and yaw, respectively. Each rotation matrix R_X, R_Y, and R_Z represents pure rotation around that axis[148]; in GCS, axes X, Y, Z correspond to E, N, G. For example, $R_N(45^\circ)$ represents rotating the phone around the North direction for 45 degrees following the right-handed rule. Equation (3.1) indicates that any orientation can be decomposed into three rotation matrices, i.e., three pure rotations around the E, N, and G axes one by one. From this decomposition, we can compute ϕ, θ, ψ, which allows *PedHat* to estimate the user's heading, as will be evident later.

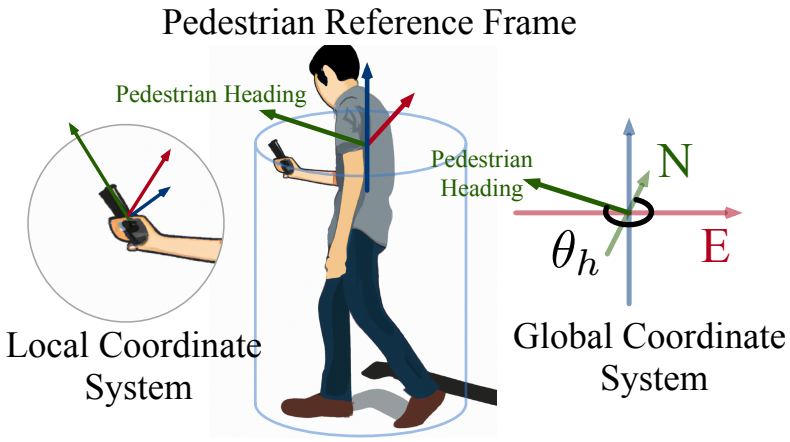


Figure 3.2: Pedestrian heading θ_h represents angle between PRF and GCS. A smartphone in front of the pedestrian in LCS. This image was created with the assistance of DALL·E 2.

3.3 Part I: Pedestrian Heading Tracking

A pedestrian's heading refers to the direction in which a pedestrian is facing projected onto a 2D horizontal plane. Essentially, we ignore any tilt of the body upwards or downwards and focus on the movement along a flat surface. Figure 3.2 illustrates an example of pedestrian heading, and the pedestrian reference frame (PRF) along with LCS and GCS. In the PRF, the Y-axis aligns with the pedestrian heading, and the Z-axis points opposite to the direction of gravity. The pedestrian can turn to change direction and move forward along the Y-axis.

3.3.1 Limitations of Current Heading Tracking Methods

There exist many methods for determining pedestrian heading, ranging from those yielding low to high precision. In outdoor pedestrian tracking, a low-precision pedestrian heading can be derived from GPS bearing, which is the direction computed from two consecutive GPS points. In indoor tracking, low-precision pedestrian heading can similarly be obtained from past trajectory points using methods such as Wi-Fi localization and Map-Matching [109]. However, such methods cannot correctly indicate the pedestrian heading

when the pedestrian turns without moving. For high-precision pedestrian heading tracking, previous researchers proposed various methods including: 1) utilizing sensors mounted on the human body, such as on foot, chest, and head [145, 167], 2) camera-based head pose estimation [115, 49, 163], 3) IMU sensor fusion-based heading tracking [92, 114, 140, 35, 155], and 4) machine-learning based heading prediction [87, 156, 81]. The first two methods require additional hardware for tracking, while the latter two can estimate pedestrian headings using smartphones carried by users.

Traditional IMU sensor fusion-based heading tracking techniques typically assume that the smartphone is fixed to the human body. These methods generally involve two-stage pipeline: first, they compute a horizontal plane using gravity, the geomagnetic field, or through smartphone orientation estimation. Then, they integrate the horizontal component of the gyroscope to compute the heading changes. An accurate initial heading and a fixed smartphone attitude are often vital for this method. Alternatively, in the second phase, they track the acceleration direction and compare with the geomagnetic north to estimate the heading direction.

With a fixed smartphone attitude, computing pedestrian heading becomes straightforward. However, heading tracking becomes more challenging when the user adopts multiple smartphone attitudes. Other researchers have explored machine-learning-based methods to address this issue [87, 81]. They begin by collecting accurate heading data and IMU sensor data across multiple phone attitudes and then train a model to understand the correlation between pedestrian heading and IMU sensor data. However, this approach may fail if the model encounters a new attitude. Overall, these two methods either do not offer a universal tracking solution for phone attitude changes or are effective only under specific phone attitudes.

3.3.2 OHA Overview

Orientation-Heading Alignment (OHA) algorithm designs improvements to pedestrian heading tracking against multiple smartphone attitudes. The key observations is that people tend to carry phones in certain ways or attitudes

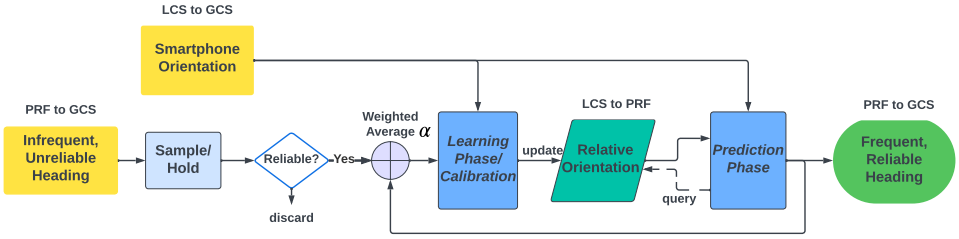


Figure 3.3: **OHA processing pipeline**: smartphone orientation data, along with unreliable and low-frequency headings (e.g. 1Hz), are processed in two phases to compute reliable and high-frequency headings (e.g., 50Hz). A weighted average is employed to merge unreliable headings with drift errors, and high-frequency headings with possible cumulative errors. **Learning Phase**: The weighted average heading and smartphone orientation data are combined to compute and calibrate the smartphone's relative orientation at current timestamp. **Prediction Phase**: High-frequency smartphone orientation data and the smartphone's current relative orientation are combined to compute a reliable and high-frequency heading.

due to habits, whether swinging them while walking, stashing them in pockets, or placing them in bags. Given the device discrepancy and user behaviors, while it's not feasible to design a system that can anticipate every potential phone attitude for all users, OHA leverages this opportunity to learn the smartphone's attitudes, referred to as the "relative orientations" to human body in the follows, for one user during movement.

Figure 3.3 illustrates the OHA heading estimation pipeline, which converts infrequent and unreliable headings into frequent and reliable ones. Essentially, OHA learns the relative orientations between smartphones and the human body (from LCS to PRF), and then combines them with smartphone orientation (from LCS to GCS) to estimate a precise pedestrian heading (from PRF to GCS). Initially, OHA lacks knowledge of any relative orientation matrices and utilizes unreliable headings to learn these matrices. As the user moves, OHA continuously calibrates the corresponding relative orientation matrices in the Learning Phase, using a weighted average of both low-frequency headings and high-frequency headings. Additionally, in the Prediction Phase, frequent updates on smartphone orientation allow for the computation of real-time and

reliable pedestrian headings. It should be noted that due to our choices for PRF and GCS, the relative orientation matrices and smartphone orientation matrices share the same roll-pitch pair, which we use to index the relative orientation matrices. Therefore, for each smartphone orientation update, OHA queries the corresponding relative orientation matrix and compute a pedestrian heading. Consequently, OHA can produce precise and frequent pedestrian headings for multiple relative orientations, even for motions such as swinging.

3.3.3 Formal Analysis

Here we provide a mathematical proof for OHA. OHA aims to predict the pedestrian heading $\theta_{h,t}$ in real time for time t. Note that PRF is the same as rotating GCS around the Z axis by $\theta_{h,t}$ in the clockwise direction, represented by rotation matrix $R_{h,t} = R_Z(-\theta_{h,t})$. In the rotation matrix, rotating in the clockwise direction is negative.

We assume that at time instant t, $O_t = R_{rpy}(\phi, \theta, \psi)$ represents the smartphone orientation, representing rotations from GCS to LCS. $R_{LP,t}$ describes the current relative orientation, representing rotations from **L**CS to **P**RF. Alternatively, this rotation $R_{LP,t}$ can be completed through two steps: (1) rotate from LCS to align with GCS O_t^{-1}, and (2) rotate from GCS to align with PRF $R_{h,t}$. Therefore, we have:

$$\begin{aligned} R_{LP,t}^{-1} &= R_{h,t}^{-1} \times O_t = R_Z(\theta_{h,t}) \times R_Z(\phi_t) \times R_Y(\theta_t) \times R_X(\psi_t) \\ &= R_Z(\theta_{h,t} + \phi_t) \times R_Y(\theta_t) \times R_X(\psi_t) \end{aligned} \tag{3.2}$$

Note that smartphone orientation O_t and relative orientation $R_{LP,t}$ share same pitch (θ_t) and roll (ψ_t) values. Therefore, we can index the relative orientations using the roll-pitch pair, and query the corresponding relative orientations using the roll-pitch pair from smartphone orientation O_t in both Learning and Prediction Phases. In the Learning Phase, we use the weight averaged headings as $\theta_{h,t}$ to learn and calibrate relative orientation $R_{LP,t}$. In the Prediction Phase, we extract heading $\theta_{h,t}$ from smartphone orientation O_t and relative orientation $R_{LP,t}$.

The equation above is also effective when the pedestrian changes the relative orientation of their smartphone. Instead of learning a single relative orientation R_{LP}, OHA can learn multiple distinguishable relative orientations, each corresponding to a common attitude where a pedestrian might carry their phone. This includes scenarios such as placement in various pockets, holding the device by one's side, or carrying it in the front. We will discuss the limitations of the OHA algorithm in Section 3.8.

3.3.3.1 Algorithm

We present the pseudo-code for the OHA algorithm in Algorithm 2. OHA processes two input streams: the coarse heading, $\theta_{h,c}$, and smartphone orientation, O_t, to produce an output stream of precise heading, $\theta_{h,p}$. Here, we briefly discuss a few details.

Initialization: OHA initiates a new relative orientation, $R_{LP,t}$, using the current smartphone orientation and a coarse heading. Alternatively, if we assume minimal change in the pedestrian's heading in the recent past, OHA could utilize the most recent estimated heading for this initialization, which we leave for future works.

Smartphone Orientation Errors: Smartphone orientation estimation typically consists of gyroscope integration and opportunistic calibration, and is inevitably prone to cumulative errors. To mitigate the impact of smartphone orientation drift and coarse heading biases, we employ a weighted average function, which works as a complementary filter, to make the estimated relative orientation more robust.

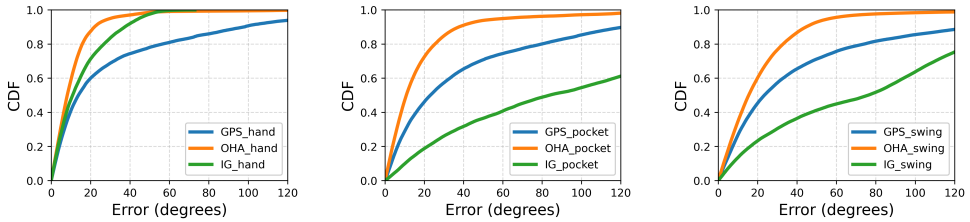
Quantization: We query the relative orientation dictionary using the quantized roll-pitch pair rather than the raw roll and pitch angles to increase query hit rate. A low hit rate generally leads to fewer opportunities to calibrate $R_{LP,t}$, thereby making it more susceptible to coarse heading bias errors. Nevertheless, setting overly large quantization factors can also induce estimation inaccuracies. We set the quantization factor to 2.

Algorithm 2: Orientation-Heading Alignment Algorithm

Result: Calibrated Relative Orientation $R_{LP,t}$
Input: coarse heading $\theta_{h,c}$, smartphone orientation O_t) $R_{rpy}(\phi, \theta, \psi)$
while O_t *updates* **do**
  Quantize θ, ψ angles;
  `// Increase query hit rate`
  if $R_{LP,t}$ ***not*** *available* **then**
    Compute $R_{LP,t}$ from $\theta_{h,c}$ and O_t for once and skip;
    `// Initialize` $R_{LP,t}$
  end
  Obtain new precise heading $\theta_{h,p}$ from Equation (3.2);
  `// Prediction Phase`
  if *coarse heading* $\theta_{h,c}$ *available* **then**
    Merge $\theta_{h,p}$ and $\theta_{h,c}$, and calibrate relative orientation $R_{LP,t}$;
    `// Learning Phase`
  end
end

3.3.4 OHA Evaluation

We evaluate OHA in an outdoor parking lot using GPS for coarse heading inputs and MUSE[127] for smartphone orientation estimation. The selection of input sources may vary depending on the scenario. In our case, MUSE is particularly suitable due to its design optimization for frequent movements and its effective use with minimal ferromagnetic interference in outdoor environment.
Experiment setup: We evaluate OHA in three typical smartphone placements: (1) handheld in front side, (2) placed in a trouser pocket, and (3) swung by body side while walking (about 60-degree swing angle). For each smartphone placement, we test three distinct walking patterns: (1) **S**traight with **O**ccasional 90-degree **T**urns (SOT), (2) **S**tationary **W**hile **R**otating to different directions (SWR), and (3) walking along a curvilinear trajectory that resembles **M**ultiple interconnected "**S**"-shaped **P**aths (MSP). We recruit five participants to walk naturally under these nine movement scenarios without further guidance. Each participant walks 3 minutes for each scenario, 9 minutes continuously for each smartphone placement and in total about 30 minutes.



(a) Handhold in front (b) In trouser pocket (c) Swing by body side

Figure 3.4: Overall heading estimation error CDF for three common smartphone placements: (1) handheld in front, (2) in a trouser pocket, and (3) swinging by the body side with approximately 60-degree swing angles. We compared OHA with GPS bearing and integrated gyroscope (IG) method, based on IMU and GPS data collected from 5 participants over a total of 135 minutes.

Baseline Comparison: We compare OHA's performance against two other techniques in outdoor settings: (1) GPS bearing, and (2) an Integrated Gyroscope (IG) approach. The IG method first utilizes MUSE-based smartphone orientation to compute the horizontal plane and then integrates the horizontal component of the gyroscope as relative heading changes, similar to previous IMU sensor fusion-based works [92, 114, 140, 35, 155]. We ensure the IG method has an accurate initial heading. All techniques employ the same IMU and GPS data collected from a Google Pixel 6 smartphone. We set gyroscope and accelerometer sampling frequency at 50 Hz, magnetic field sensor at 10 Hz and GPS updates at 1 Hz.

Ground Truth: OHA utilizes another chest-mounted smartphone to measure the ground truth heading. We first securely attach this phone to a participant's chest using a chest mount strap. We calculate the smartphone's orientation based on the IMU sensor readings, and subsequently derive the azimuth angle. We perform a one-time initial calibration of the azimuth angle by walking along a straight line from east to west, and use this calibrated azimuth angle as the ground truth heading.

Results: Figure 3.4 compares the overall heading estimation error CDF across three smartphone placements. OHA outperforms the other two techniques consistently in all tested placements. Notably, the performance of the IG method

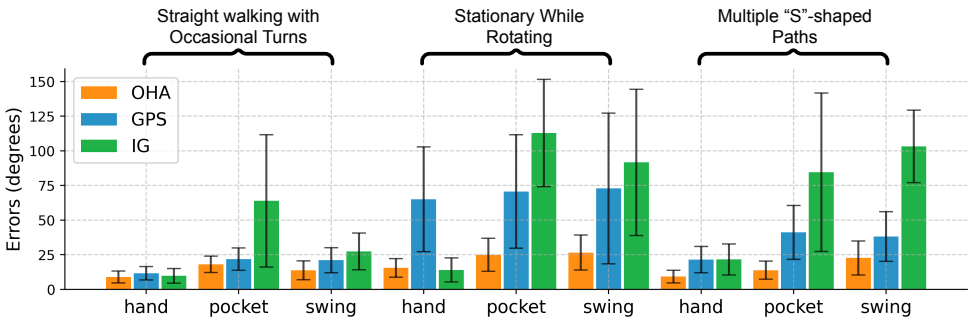
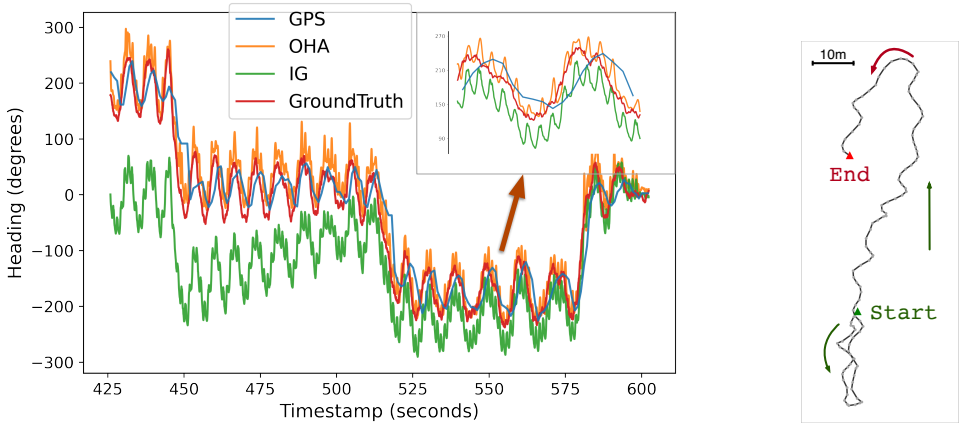


Figure 3.5: Heading estimation mean and interquartile range errors across 9 movement scenarios from 5 participants. Scenarios combine three smartphone placements: (1) hand, (2) pocket, and (3) swing, and three walking patterns: (1) Straight walking with Occasional Turns (SOT), (2) Stationary While Rotating (SWR), and (3) walking in Multiple "S"-shaped Paths (MSP).

drops significantly when the smartphone is not held in hands. Meanwhile, the GPS heading's performance is almost consistent over different smartphone placements. We observe that when the smartphone is placed in the trouser pocket, it also experiences the swinging motion due to leg movements, similar to that in the swinging case. Therefore the IG method yields higher cumulative errors for both cases.

Figure 3.5 illustrates the mean and interquartile range errors across nine movement scenarios from all five participants. Note that GPS performance drops during "MSP" and deteriorates significantly during "SWR", whereas OHA consistently maintains low heading errors, underlining its efficacy in conditions where coarse heading inputs are heavily biased. The IG method performs well when the smartphone is held in "hands", but its performance is markedly poorer for the other two smartphone placements, especially during "SWR" and "MSP". In contrast, OHA demonstrates robust accuracy and achieves the lowest error rate in seven out of nine scenarios, except when the smartphone is held in the front during "SOT" and "SWR". Under these two conditions, where minimal smartphone swinging occurs, the IG method displays similar performance with OHA. Overall, OHA maintains consistent accuracy across different participant and scenarios, outperforming the IG method with an average of 3.4 times smaller errors.



(a) (b)

Figure 3.6: Participant places the smartphone in his trouser pocket and walks in "S"-shaped paths. (a) A sample trace of heading over time for three heading estimation techniques compared with ground truth heading. (b) walking trajectory of the sample trace in (a).

Figure 3.6a compares three heading estimation techniques over a 3-minute sample trace. Before this sample trace, the participant has walked for 7 minutes with the smartphone in his trouser pocket, and continues walking in "S"-shaped paths as illustrated in Figure 3.6b. During certain short intervals (e.g., 30 seconds), the IG method can effectively mirror the heading change pattern and achieve minimal errors as in previous works [92, 140]. However, it suffers from uncontrollable cumulative errors over longer periods, which causes deviations from the ground truth headings. Meanwhile, the GPS heading, while free from drift errors, exhibits a consistent response delay of approximately 2 seconds when following the participant's turns. In the contrast, OHA demonstrates its ability to accurately track the pedestrian heading over a long time even when the user turns frequently.

GPS Accuracy: We further evaluate the impact of GPS accuracy on OHA's performance. To simulate GPS signal interference, we placed the smartphone inside a box lined with aluminum foil, and varied the degree of signal blockage by adjusting the box's gap, from fully open to completely closed. Throughout

the experiment, the smartphone was kept in the box at all times, and we carried the box while walking in two patterns: Straight and Occasional Turns (SOT) and Multiple S-shaped Paths (MSP). We observed that with the box fully closed, the GPS accuracy was consistently around 7.3 meters while occasionally reaching up to 20 meters. Figure 3.7 illustrates a comparison of OHA's performance across different GPS accuracy conditions. Despite variations in GPS accuracy, OHA maintained satisfactory performance, demonstrating its potential for real-world environments.

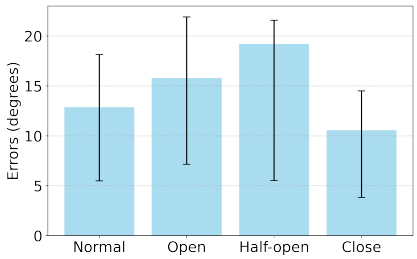


Figure 3.7: OHA heading errors related to different levels of GPS signal blockage. The smartphone was placed in a box lined with aluminum foil, with the opening controlled as follows: (1) Normal - no cover, GPS accuracy is around 4 meters, (2) Open - with an opening, (3) Half-Open - with the opening partially closed, (4) Closed - completely closed.

3.4 Part II: Pedestrian Road Crossing Prediction

Precise pedestrian heading is crucial for various applications, including enhancing crosswalk safety in urban environments, offering detailed navigation assistance in both indoor and outdoor environment, and improving the accuracy of location-based services on smartphones.

To demonstrate our technique in a practical scenario, we apply OHA in one of the more challenging tasks: predicting pedestrian's crossing behavior with **low latency** and **high accuracy** using commodity devices. GPS bearing is inadequate for this task due to its inaccuracy and inherent delays. Additionally, it fails to determine pedestrian's heading when they are stationary and rotating, a common scenario when a pedestrian decide to cross. We introduce *PedHat*,

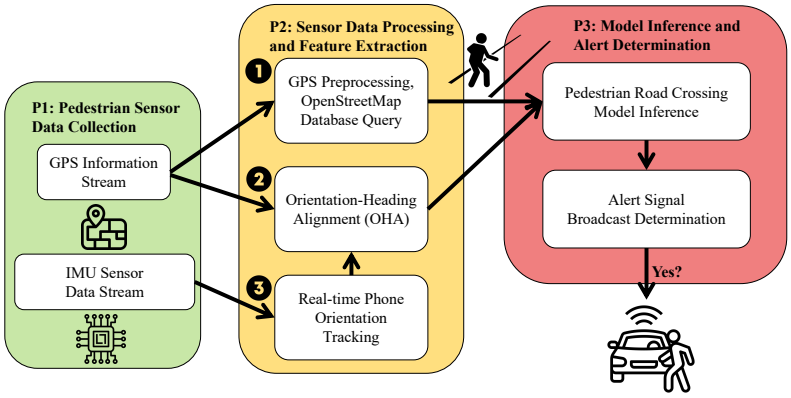


Figure 3.8: Pedestrian Road Crossing Prediction system overview. Phase 2 accepts GPS and IMU sensor data from Phase 1, and delivers Phase 3 1) general physical location and 2) OHA heading. We will introduce the road crossing model development and alert signal determination in Section 3.4.2, and implementation details in Section 3.5.

a system that predicts pedestrian's crossing behavior based on pedestrian's precise heading from OHA and pedestrian's geographical location on roads.

3.4.1 *PedHat* Overview

PedHat aims to predict instances when a pedestrian is *about to cross the road closest to them* and broadcasts alert messages to warn nearby drivers. These instances include pedestrians crossing at traffic lights or jaywalking at road mid-blocks. Figure 3.8 provides a high-level overview of *PedHat*, illustrating its main components. At a high level, *PedHat* operates in three phases. In the first phase, it samples GPS and IMU data, which it uses in phase 2 to extract fine-grained mobility features about the user. In the third phase, it feeds these features to a model that predicts when the user is about to cross the road.

Phase 1: *PedHat* utilizes GPS data and map information to identify the closest road to the user and estimate the coarse-grained distance to the road. This design makes *PedHat* resilient to on-device GPS inaccuracies, especially in urban environments [101]. It is challenging to use GPS to detect real-time user movements, such as turning towards the road or stepping towards the road edge, with a submeter accuracy.

Phase 2: *PedHat* estimates two mobility features: coarse-grained distance to the closest road and the real-time heading from OHA. As evident in Section 3.3.4, *PedHat* overcomes the challenges of estimating the fine-grained heading information from IMU. Meanwhile, *PedHat* requests the nearby road shape data, to assist in identifying the intent of pedestrian road crossing.
Phase 3: Finally, *PedHat* feeds the recent history of these mobility features to a model, which predicts whether the user is about to cross the road. Incorporating recent mobility information gives the model more context for the road crossing prediction task. Moreover, the model can uncover some pedestrian mobility patterns that rule-based approaches can not [120]. Section 3.6.2 provides more insights into how the model uses the features.

We integrate the above functionality into a standalone and lightweight user-level app on the Android platform. This app requires no special hardware, infrastructure, or user support. It autonomously predicts when the user is about to cross the road and broadcasts a wireless message to alert nearby road users about potential road crossings. We base our broadcasting mechanism on the existing Wi-Fi Aware protocol through beacon stuffing [26]. Section 3.5 contains more details about our implementation. This design makes *PedHat* easier to deploy and adopt than other solutions requiring dedicated infrastructure or special hardware.

3.4.2 Machine Learning Model Details

PedHat utilizes a machine learning model to predict whether a pedestrian will cross the road using recent mobility information as features. There are two challenges in predicting a pedestrian's road crossing: the lack of road width information and GPS inaccuracies. Both challenges prevent *PedHat* from determining the pedestrian location relative to the road. Even the fine-grained heading information from the OHA algorithm is not enough to decide on road crossing without some notion of the user's location. *PedHat* leverages an intuitive observation about pedestrian road-crossing behavior to tackle these challenges: *pedestrians turn towards the road before crossing it*. By consolidating inputs from a pedestrian's recent OHA heading, location information, and

map data, *PedHat* can assess the pedestrian's crossing potential.

Feature Extraction: *PedHat* uses three key features: the distance to the road center, OHA heading (pedestrian heading from OHA), and the road reference angle. The distance information represents how close the pedestrian is to the nearest road. The OHA heading, along with the road reference angle, helps determine the user's movements relative to the closest road. We describe each feature used as follows:

- **Distance to Road Center** d represents the perpendicular distance to the centerline of the nearest road, given the user's current GPS location. Despite imprecise GPS data and the lack of accurate road width information, the distance to the road center indicates whether a pedestrian is within a possible road crossing region.
- **Road Reference Angle** θ_{ref} provides a reference to assess if a pedestrian is approaching or departing from a road. It represents the pedestrian's heading angle when they face perpendicular to the road. *PedHat* extracts θ_{ref} using the road shape in map data.
- **OHA Heading** θ_{oha}, the pedestrian's current heading in real-time, helps reveal the pedestrian's crossing potential along with θ_{ref}. In particular, we use $\cos(\theta_{ref} - \theta_{oha})$, the cosine value between the OHA heading and road reference angle as a normalized input feature.
- **Prediction Window** represents the length of the input feature vector. In particular, the model input is a vector of the last N readings of θ_{ref}, θ_{oha}, and d. Depending on the sampling frequency in the smartphone, this vector will represent the past *lookback* seconds of mobility data.

Model Architecture: We employ a dual LSTM (Long Short-Term Memory) layer configuration to efficiently process the extracted features on daily walking trace data. Specifically, one LSTM layer analyzes distance-related sequences and the other focuses on heading-related sequences, both operating in parallel to effectively capture temporal patterns. The outputs from these layers are subsequently concatenated and fed into a fully connected layer with a sigmoid activation function, producing the crossing probability at the given timestamp.

Crossing Alert Determination: Rather than relying on the model's single prediction result, *PedHat* determines whether to broadcast a crossing alert after consulting the past n predictions. This approach reduces the occurrence of false alarms that can undermine the user's confidence in *PedHat*. If the number of "True" crossing predictions in the past n predictions surpasses a threshold, usually 50%, *PedHat* will broadcast a crossing alert to nearby vehicle drivers. With a larger n, *PedHat* provides a more reliable, yet delayed, crossing alert. On the other hand, with a smaller n, *PedHat* issues an alert earlier, but with the risk of generating more false alarms.

3.4.3 Model Development

We constructed and refined our model using this dataset described in Section 2.4.

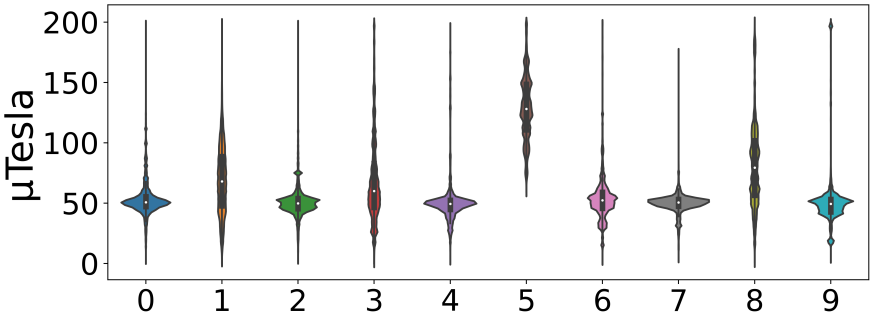


Figure 3.9: Distribution of Magnetic field magnitude measured by ten pedestrians while walking outdoors.

3.4.3.1 Model Training and Testing Results

We selected $lookback$ length as 8 seconds, extracting the input features and predicting every 100 ms ($N = 80$). We use two 64-unit LSTM layers for processing the two feature vectors and a dense layer to process the concatenated 128-long LSTM outputs. For issuing a crossing alert, we choose $n = 20$, equivalent to 2 seconds, to balance the trade-off between crossing alert timing and false alarm rate.

lookback	t = 2s	t = 4s	t = 8s	t = 16s
Precision (%)	85.40	83.97	**85.61**	82.62
Recall (%)	90.85	89.79	**89.59**	87.65

Table 3.1: Precision and Recall rate on the test dataset of models trained with varying *lookback* lengths.

We implemented our model using Tensorflow and trained it on four NVIDIA RTX 2080 Ti GPUs. During training, since temporally close samples exhibit strong similarity, we do not shuffle the samples in the training set. We set aside 10% of the training data as the validation set. We also use early stopping with patience as 3 to avoid model overfitting. Note that the samples in our daily walking traces are imbalanced, consisting of roughly 5% true crossing events and 95% non-crossing traces. As such, we employ precision and recall on true crossing events as our evaluation metrics.

Road Crossing Prediction Results: We tested the road crossing prediction model on the granularity of events instead of each input sample. Originally, the model predicts the road-crossing probability for each sample in the road-use session, resulting in a probability sequence. We determine crossing alerts on this sequence and retrieve crossing alert periods with start- and end-timestamps. We test these periods against the labeled "True" crossing events in the road-use session. We define a true alarm as an overlap between a crossing alert period and a "True" crossing event, while the absence of such an overlap is a false alarm.

For additional testing, we trained four models with different *lookback* lengths, including 2s, 4s, 8s, and 16s. We present each model's performance on the testing dataset in Table 3.1. The recall rate keeps decreasing as *lookback* length increases, while precision and recall rate drop to the lowest when the model utilizes past 16s sensor information. *PedHat* choose *lookback* length as 8s which yields highest precision rate (85.61%) among four models. We also compared with baseline models in Section 3.6.2.4.

3.5 Implementation Details

We implement *PedHat* as a standalone app on the Android platform. In the following, we briefly mention some implementation details and describe our crossing alert broadcasting mechanism.

3.5.1 Android-based Implementation

Orientation Estimation Modifications: Figure 3.9 shows the outdoor magnetic field distributions for ten users from the daily walking traces described in Section 2.4.1. We can observe significant variations in magnetometer readings both across devices and over time, which affects the performance of MUSE. To mitigate this, we empirically constrained the range of magnetometer sensor readings, which effectively reduced the maximum orientation error caused by environmental magnetic fields.

***PedHat* Running Modes:** For power-saving purposes, *PedHat* has two running modes: one is active mode, and the other is idle mode. In the active mode, *PedHat* operates IMU sensor data fetching and feature data processing at 50 Hz, GPS updates at 1 Hz, and model inference every 100 ms. In the idle mode, *PedHat* only retrieves GPS data and queries the OSM database at a relatively low frequency, such as every 2 seconds. In our implementation, *PedHat* enters the idle mode when the pedestrian is 10 meters away from the road or when the user is immobile for more than 1 minute, where the pedestrian indicate a low probability of crossing.

App Integration: We implemented *PedHat* as an Android app compatible with Android 11 to Android 13 devices. The app uses OpenStreetMap (OSM [33]) for app data containing road shape information. We load the OSM data on a local Postgres database on the phone using Termux [138]. The app queries the database for the nearest road, the distance to the road center, and the road's angle using PostGIS [108]. It utilizes TensorFlow Lite for model inference at run-time. Finally, it uses Android's WifiAwareManager [36] to broadcast the road crossing alerts.

3.6 Evaluation of Pedestrian Crossing Prediction

Evaluating pedestrian road crossing prediction includes two key aspects. First, we assess how accurately *PedHat* can **identify the crossing events** for new pedestrians. Specifically, we examine how *PedHat*-issued crossing alerts match the true crossing events. Second, we consider the promptness of these alerts, specifically how early *PedHat* can alert before a pedestrian crosses the road edge, referred to as "**time-to-crossing**" in previous research [49, 110]. We address these two key aspects respectively in Sec. 3.6.1 and in Sec. 3.6.2. We designed the evaluation to answer the following questions, and we summarize our observations:

1. **[Crossing Identification] How well can *PedHat* generalize to new participants with fine-grained labeling?** We conducted a study that monitored another 25 participants for evaluating *PedHat* in a semi-controlled real-world environment. We took videos of walking participants on a designated route and the road environment, and then we recruited annotators to label these videos independently. Our labels include both pedestrian's completed crossing and crossing potentials. Overall, *PedHat* achieved a precision of 86.9% and a recall of 93.6% in identifying the completed crossings.

2. **[Detailed Analysis] How early can *PedHat* issue alerts, what contributes to *PedHat*'s error and how is *PedHat* compared to a baseline model?** Comparing with fine-grained labels from our annotators, we observed that *PedHat* typically issues crossing alerts before a pedestrian crosses the road edge. We identified five error sources of *PedHat*, among which GPS errors and ambiguous pedestrian crossing intent contribute the most. We also illustrated six walking traces with marked predictions in Figure 3.11. After training the baseline models following the same procedures, we found that the baseline model issues a crossing alert 2.1 seconds after a pedestrian crosses the road edge.

3. **[System overhead] What is *PedHat*'s system overhead?** We measured the performance impacts of our system on multiple commodity devices

with different Android versions. We found the average execution time for each running loop is around 10 ms, and the average battery consumption over 30 minutes is around 352 mAh for active mode and 95 mAh for idle mode.

3.6.1 Q1. Crossing Identification of *PedHat*

To understand how well *PedHat* generalizes to new participants, we first conducted a user study with another 25 participants, separate from the 60 participants discussed in Sec. 3.4.3. Then we evaluated how accurately can *PedHat* identify crossing events using data collected from this user study.

3.6.1.1 Semi-controlled Walking Trace Collection

We conducted this study in a semi-controlled real-world environment and video-recorded the participants' walking behavior for labeling. We designated the start and end points on our campus for participants to walk. When a participant was walking, we recorded the pedestrian and the road environment. This data collection received an exemption from our institute's IRB. The data collection took place from May to July 2023, and we recruited 25 participants (our institute's staff and students) across campus. Of the participants, 17 identified themselves as male (8 female). Participants' ages ranged from 20 to 45 years old.

We asked each of them to follow the same walking path in the campus area, a walkable environment, while carrying a phone with *PedHat* installed. Each participant walked at their own speed and decided to cross the road when they felt comfortable. The walking took around 15 minutes for each participant. A researcher followed and recorded participants' walking behaviors without interfering with them. We compensated each participant with office supplies worth $8.

Video-based Road Crossing Labeling: We recruited two graduate students majoring in traffic safety to label and timestamp participants' walking behavior by watching the recorded videos. We identified three types of pedestrian behavior that need to be labeled: actual road crossing, potential road cross-

(a) ac: actual crossing example (b) pc: potential crossing example (c) db: dangerous behavior example

Figure 3.10: Examples of three behaviors, (a) actual crossing (ac), (b) potential crossing (pc), and (c) dangerous behavior (db) labeled by both annotators.

ing, and dangerous/unsure behavior. Both annotators followed the labeling guidelines below:

• **Completed/Actual road crossing:** Label this behavior when observing that a pedestrian has crossed the road. Mark the timestamps when the pedestrian (1) demonstrates crossing potential, (2) reaches the road edge and prepared to cross, and (3) departs from the road edge.

• **Potential road crossing:** Label this behavior when observing a pedestrian showing intent to cross but not crossing eventually. Mark the timestamps when the pedestrian (1) shows and (2) no longer shows crossing intent.

• **Dangerous/Unsure behavior:** Label this behavior when observing a pedestrian displaying dangerous behavior or when being unsure if the participant demonstrates "potential road crossing." Mark the timestamps when such behavior begins and ends.

Before labeling the whole walking traces, we asked both annotators to independently label a recorded walking video of one researcher to measure the agreement between them. We found that both annotators showed consensus on the occurrences of completed road crossing events but had minor disagreements on the other two behaviors. Their labeling results reached a high agreement with an Intersection over Union (IoU) value of 0.847. We present examples of three behaviors annotated by both annotators in Figure 3.10.

3.6.1.2 Crossing Identification Evaluation Results

We evaluated *PedHat*'s performance in identifying completed road crossings on the **semi-controlled walking traces**. Compared to daily walking traces discussed in Sec. 3.4.3, semi-controlled walking traces further recorded detailed crossing behaviors, including participants' body movements and postures, to clearly determine when pedestrians reach the road's edge and prepare to cross. Through video-recording and fine-grained labeling, we can better evaluate *PedHat*'s performance in identifying road crossings and validate our model's generalizability to new participants.

Performance Results: Based on the labeling results from both annotators, we observed 443 complete crossings in total. The annotators were in full agreement about the completed crossing events. Regarding crossing potentials, one annotator identified 63 instances while the other identified 43.

We evaluated *PedHat*'s prediction results using the complete crossings as ground truth. Specifically, we used complete road crossings as "True" crossing events and took the average of the precision and recall rates across the 25 participants. Overall, our model achieved an 86.7% precision rate and a 93.6% recall rate in identifying participants' complete road crossings. It is important to note that we labeled the semi-controlled walking traces via examining video recording, while the crossing events in the daily walking traces were labeled using GPS, which inevitably includes false crossing labels. Therefore, although our model has never encountered these semi-controlled walking traces before, it exhibits better performance in the evaluation, demonstrating its generalizability to new participants.

3.6.2 Q2. Detailed Analysis for *PedHat*

We performed a detailed analysis for *PedHat* in terms of time-to-crossing (TTC), which we define as the time from *PedHat*'s alert start timestamp to pedestrian's crossing start timestamp. Further, we analyzed how *PedHat* performs when encountering various factors, e.g., the potential crossings, in error analysis. Additionally, we compared *PedHat*'s performance against baseline models.

Past n predictions	10	**20**	30	40	50
Precision (%)	83.8	**86.7**	89.6	91.6	91.3
Time-to-crossing (s)	0.62	**0.36**	-0.43	-0.85	-1.54

Table 3.2: Trade-off between time-to-crossing and false alarm rate by varying past consulted predictions n. 10 predictions equal to 1s and negative TTC represents crossing alerts happen after pedestrian has crossed the road edge.

3.6.2.1 Time-to-crossing

Time-to-crossing indicates how early *PedHat* can alert vehicle drivers about pedestrian crossing nearby. In particular, TTC represents the time difference between when *PedHat* issues an alert and when the participant enters the road range ("reaching road edge" timestamp tagged by annotators). An earlier crossing alert provides drivers with more time to avoid potential collisions.

For each correctly identified crossing event, we calculate the time-to-crossing as We compared *PedHat*'s alert time with both annotators' results respectively. When consulting the last $n = 20$ predictions, equivalent to the past 2 seconds, *PedHat* issues an alert, on average, 0.39 and 0.32 seconds **ahead of** the "reaching road edge" event. Note that we do not include any road width information during training, and the roads traversed consist of 2-lane and 4-lane roads with varying road widths.

We also evaluated the trade-off between time-to-crossing and false alarm rate in Table 3.2. With a higher n, *PedHat* considers a longer past time period and more prediction results, leading to more confident decisions on whether to issue a crossing alert but also delaying the alert time. In summary, a higher n increases the precision rate but decreases the time-to-crossing. Furthermore, in Table 3.4, *PedHat* demonstrates performance comparable to camera-based pedestrian detection techniques, while maintaining a relatively high precision rate and functioning effectively in NLOS situations.

3.6.2.2 Error Analysis

We analyzed the root causes of false positive predictions and false negative predictions here. False positive predictions, also called false alarms in *PedHat*,

can be triggered by a variety of factors, including GPS errors, crossing potentials, road shapes, and labeling discrepancies. We visualized each participant's walking trace on a map with the three annotated behaviors (443 complete crossings), false positive and false negative predictions highlighted. We carefully examined the visualized traces and categorized the overall **64** false alarms out of the 473 "True" predictions on the semi-controlled traces based on their root causes. We illustrated six walking traces for better explanation in Figure 3.11.

- **GPS drifts**: Although *PedHat* can tolerate GPS errors to some extent, severe GPS drifts, as in Figure 3.11a and Figure 3.11c, can still mislead *PedHat*. **23** false alarms, approximately 5% of all crossing predictions, were attributed to such drifts, where the GPS data incorrectly indicated a pedestrian's entry onto the roadway.

- **Late warnings:** *PedHat* sometimes issued a crossing alert after the pedestrian has already completed the crossing. It resulted from GPS signal delays as indicated in Figure 3.11b and Figure 3.11d. Late warnings account for **8** false alarms. Additionally, these GPS delays can also result in false negative predictions.

- **Crossing potentials:** *PedHat* issued **11** "false" alarms attributed to participants' crossing potentials. We identified these alarms resulting from data being labeled as "Potential Crossing" and "Dangerous Behavior" without an actual crossing. During these events, participants' turning towards the road or approaching to the road's edge caused *PedHat* to issue crossing alerts, as shown in Figure 3.11e. For two annotators, *PedHat* triggered alarms for 5 out of 43 and 11 out of 63 labeled potential crossing events, respectively.

- **Road shape:** Of the **10** false alarms attributed to road shape, 7 occurred near a curvy road where pedestrians were heading towards the road's center. The remaining three false alarms occurred near crossings where pedestrians turned right rather than continuing straight.

- **Labeling discrepancy: 12** false alarms can be attributed to labeling discrepancies. Seven stemmed from the OSM database mistakenly identifying bike paths as roads. Four were due to annotators missing labeling two court roads. In the last false alarm, the pedestrian completed the study while standing by the road and facing it, yet neither annotator classified this as a potential crossing.

False negative predictions are caused by GPS delays and drifts. Although OHA heading indicates pedestrian facing towards the road, *PedHat* does not issue a crossing alert since pedestrian seems static (GPS delay, 8 cases) or far away from the road (GPS drift, 21 cases), constituting approximately 6% of all crossing predictions.

3.6.2.3 Model Explainability

We used SHAP (SHapley Additive exPlanations [90]) to analyze how each feature contributes to our model's predictions. We randomly selected 1200 samples from our user study, each of length 80 elements and with two features. We displayed the top 7 most important time indexes of both input features in Figure 3.12. The X-axis represents the impact on the model output, and the Y-axis denotes the element index, with 79 being the last and most important element index of the input vector. We observed that the model attributes similar importance to heading information and coarse-grained distance information. Furthermore, the model receives positive impact with high heading value (cosine value equal to 1 indicates the OHA heading and Road reference angle are aligned) and low distance value (a smaller distance to the road center suggests nearing the road center) on index 79. However, the model's behavior is the opposite for indexes around 70.

From the samples, we observed that the model may give non-crossing predictions when pedestrians turn away from the road, in which index 70 may have a high heading value. Additionally, the last 3 to 4 seconds of data usually have higher impacts on prediction results than the initial ones.

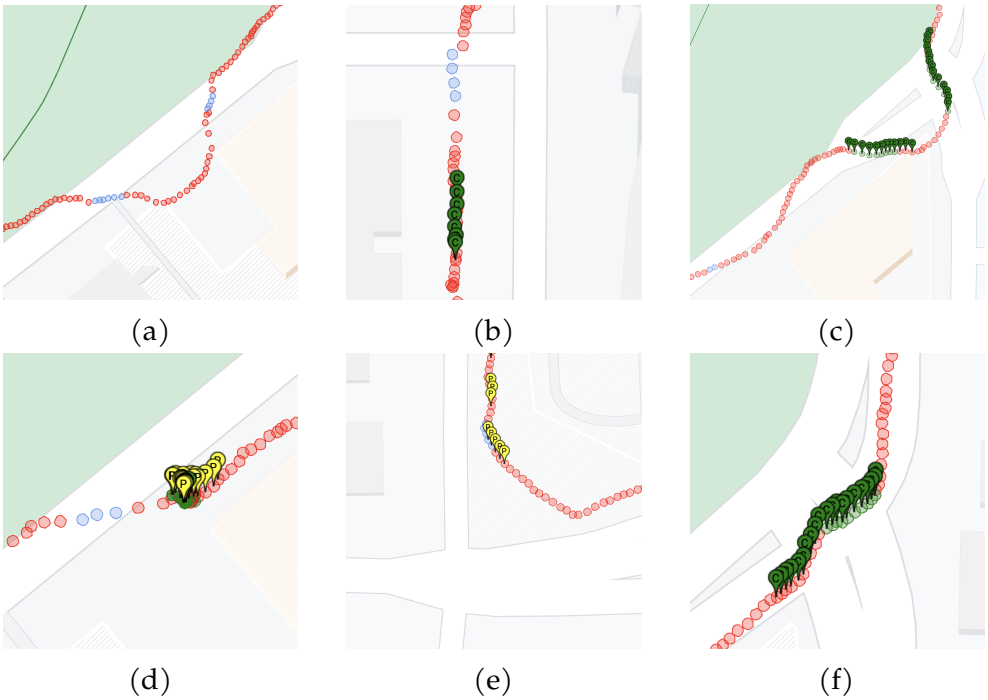


Figure 3.11: We visualized six walking traces on a map, each marked with annotated behaviors and prediction results. Red dots represent GPS coordinate points of a walking trace. Here, a true positive prediction is indicated by a green dot, and a false positive prediction by a blue dot. A green marker with "C" and a yellow marker with "P" represent actual road **C**rossings and **P**otential road crossings, respectively, as labeled by annotators. (a) The participant is not crossing, but severe GPS drift causes false alarms; (b) Significant GPS delay leads to false alarms, also called late warnings; (c) GPS drifts to the other side of the road. *PedHat* avoids initial false alarms but later issues a short-period false alarm; (d) GPS freezes while the participant is crossing. *PedHat* predicts true crossings but also issues false alarms afterward; (e) The participant shows crossing intent as labeled by annotators, and *PedHat* predicts crossing, which is also categorized as a false alarm; (f) Despite continuous GPS drift, *PedHat* successfully distinguishes between true and false crossings.

3.6.2.4 Baseline Model Comparison

We trained baseline models using alternative input features for comparative analysis. Instead of using our curated features described in Section 3.4.2, we

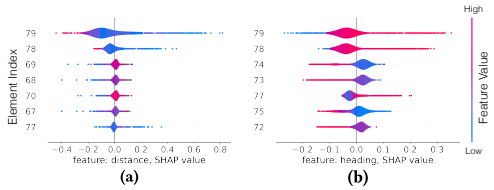


Figure 3.12: SHAP analyzes *PedHat* model input features: (a) distance and (b) heading. X axis represents SHAP value(impact on model output, positive means more likely to cross, negative means more likely to not cross), and Y axis shows top 7 important time indexes.

utilized three input feature sets: 1) distance to road center only, 2) GPS bearing only, and 3) a combination of both distance to road center and GPS bearing. It is important to note that these features are directly derivable from GPS data and road shape information, with the distance to road center same as that in our primary model. We also chose a *lookback* length of 8 seconds to facilitate a more straightforward comparison. We trained and evaluated the baseline models with same training parameters, train/test set and evaluation set.

Baseline model with feature set one achieved 78% precision rate and 86.8% recall rate on the test set of daily walking traces, and 84.4% precision and 90.7% recall on the video-recorded walking traces. In terms of early detection time, our model predicts pedestrian crossings prior to reaching the road edge. In contrast, the baseline model typically identifies a true crossing, on average, **2.1 seconds after** crossing the road edge, rendering it less effective for real-world applications. Moreover, such advance in early detection time, as well as visualized walking traces in Figure 3.11d and Figure 3.11f demonstrate that *PedHat* can mitigate the impact of GPS inaccuracies with the assistance of fine-grained real-time heading.

Furthermore, models trained with the second and third feature sets, both incorporating GPS bearing, exhibited significantly poorer performance. They failed to effectively learn from the data, regardless of variations in model architecture or adjustments in the *lookback* duration. We hypothesize that this inadequacy stems from the inherent GPS inaccuracies. Such inaccuracies might lead to false indications of pedestrian movement towards the road, even

Device specs	CPU cores	RAM (GB)	DB Query (ms)	OHA (ms)	Inference (ms)	Execution (ms)	CPU Usage % (High/Low/Avg)	MEM Usage % (High/Low/Avg)	Consumed Power(mAh)
Pixel 3, Android 11	8	4	63.91	0.068	9.93	10.24	100.0/87.3/92.3	4.5/3.8/4.25	289.2
Pixel 6, Android 12	8	8	32.92	0.054	7.01	7.63	120.0/73.0/103.0	4.0/2.5/3.03	421.1
Pixel 7, Android 13	8	8	58.04	0.088	7.72	8.27	101.0/89.0/95.0	3.1/2.7/2.93	346.4

Table 3.3: Device specs, execution time and resource usage when running *PedHat* in active mode.

when no actual road crossing occurs. Consequently, the inclusion of such features, which are not robust or reliable, misleads the model.

3.6.3 Q3. System Overhead

We evaluate *PedHat*'s system overhead in both active mode and idle mode.

Measurement Setup: Our setup for evaluating the system overhead of *PedHat* includes two Pixel 3 devices, three Pixel 6 devices, and one Pixel 7 device. We charged these smartphones to full battery capacity and had *PedHat* deployed and properly configured for each smartphone.

We first measured the program execution time. *PedHat* incorporates various functions, including database query, feature extraction, and model inference. We utilized the Android system API to record the execution time of each function in nanoseconds. We report the average execution time of each function when running *PedHat* for 30 minutes continuously. We used the Android `BATTERY PROPERTY CHARGE COUNTER` to retrieve the current battery capacity in mAh during the run time. To minimize the influence of other apps, only our app and the PostgreSQL database were active during the measurement of system overhead.

Overhead Results: We present the system overhead results of active mode in Table 3.3. The average execution time for each prediction (OHA, inference and others) is around 10 ms. Noted that database queries were conducted in a separate thread and were executed once per second. In idle mode, we observed a significant drop in average resource utility and power consumption. Specifically, the average CPU utility rate decreased to approximately 0.56%, and the average RAM usage fell to about 2.4%. Power consumption, when continuously running *PedHat* for 30 minutes, decreased to approximately 95

mAh across all devices.

3.7 Related Work

Pedestrian road crossing prediction is a popular research topic due to its potential to protect the safety of road users. Existing research in this area mainly uses cameras or LiDARs on vehicles and infrastructure [124, 70, 100, 99, 149]. For completeness, we also discuss infrastructure-free approaches that apply physical modeling or machine learning methods to predict pedestrian mobility. The latter approaches do not consider road crossing prediction per se, yet they can help such a system when combined with road map data.

3.7.1 Infrastructure and Vehicle-based Approaches

Infrastructure-based approaches utilize sensors, such as LiDARs and cameras, to first recognize pedestrians in a scene and then track their poses and positions relative to roads [128, 124]. Keller et al. estimate fine-grained pedestrian positions and poses from a camera. Then, they classify pedestrian movements, including standing, stopping, and crossing [70]. Mogelmose et al. fuse pedestrian's locations from the camera, vehicle's location, and OpenStreetMap data to decide whether a pedestrian is within a danger zone [100]. Minguez et al. provide accurate pedestrian activity recognition by estimating pedestrian gait from joint data and using a dynamic model to predict pedestrian's movements [99].

Besides detection accuracy, time-to-crossing, is another key factor in evaluating such pedestrian crossing detection systems. Zhang et al. [163] predict the crossing intent at intersections by estimating pedestrian poses captured by CCTV cameras. Their system achieves its peak performance, as measured by the F1 score when forecasting crossing intentions approximately one second prior to the pedestrian initiating the crossing. Jeong et al.[66] and Xu et al.[153] classify sudden pedestrian crossings into three danger levels—warning, caution, and normal—based on the percentage of overlap between the pedestrian's body and the road reference line. Fang et al. [49] evaluated the performance

Table 3.4: Comparison of Time-to-Crossing for pedestrian crossing detection. LOS represents Line-of-Sight, and NLOS represents Non-line-of-sight. Time-to-crossing refers to the time before crossing the road edges at which the prediction occurs. Higher time-to-crossing indicates earlier detection. Probability refers to its confidence in determining whether a pedestrian is going to cross.

Author(s)	Conditions	Sensors Used	Time-to-Crossing (s)	Precision	Recall	Probability
Xu et al. [153]	LOS	Camera (vehicle)	0	0.52	-	-
Keller et al. [70]	LOS	Camera (vehicle)	Up to 0.77s	-	-	-
Jeong et al. [66]	LOS	Far-infrared camera	0	0.85	-	-
Palffy et al. [110]	LOS, NLOS	Camera, Radar	0.3	-	-	0.8
Fang et al. [49]	LOS	Camera (vehicle)	0.43 (13 frames)	-	-	0.8
Zhang et al. [163]	LOS	Camera (road-side)	1	0.859	0.818	-
Ours: *PedHat*	LOS, NLOS	IMU, GPS	0.39	0.869	0.936	-

of recent advancements in vision-based human pose estimation for identifying pedestrian crossing intentions. They found that in scenarios where the subject continued walking to cross the road, the probability of crossing increased to 0.8 as early as 15 frames (0.5 s) ahead of the crossing events.

For pedestrians who are occluded behind objects such as parked cars, these systems cannot predict pedestrians' behavior without capturing the pedestrians through cameras. Palffy et al. [110] proposed an occlusion-aware model utilizing cameras and LiDAR reflections to help detect pedestrians earlier. Their system can capture pedestrians 0.3 seconds earlier than a camera-based detector, which detects the pedestrians after they appear in sight.

Scholler et al. [123] claimed environmental priors and long pedestrian motion history might negatively impact the generalization of these models. Other researchers track human gestures through Wi-Fi [67, 168, 47] or millimeter-wave [74, 165], which is not applicable in outdoor environments.

Compared to these approaches, *PedHat* does not require line of sight, infrastructure, or special hardware since it predicts pedestrian behavior utilizing smartphone sensors. Table 3.4 shows how *PedHat* compares in performance to prior approaches. The main takeaway from the table is that *PedHat* provides superior performance to camera-based approaches with an acceptable prediction delay.

3.7.2 Infrastructure-free Approaches

The other category of approaches uses IMU sensors from the pedestrian to predict their mobility without relying on external infrastructure.

3.7.2.1 Kinematics-based Approaches

These approaches, such as dead-reckoning, apply motion dynamics on IMU sensors and location data to model pedestrian movements [151]. Feigl et al. [52] utilize IMU sensor and deep learning models to predict the pedestrian's velocity. Kuang et al. [77] developed a pedestrian trajectory prediction approach that performs heading estimation, position estimation, and step detection using smartphone IMU sensors. Fan et al. [46] further improved the positioning accuracy by adding an adaptive Kalman filter to heading estimation. Other dead-reckoning approaches utilize a step-and-heading method involving step counting, step stride estimation, and heading estimation. Edel et al. utilize Bi-LSTM and RNN models to achieve this task from IMU sensors on the phone [40]. Xing et al. use specific sensor hardware installed on the foot to predict step length [152]. However, IMU sensors are prone to sensor drifts in the long term, and these methods did not demonstrate their performance in such situations when pedestrians keep walking outdoors. Map-matching techniques, such as open source routing machine (OSRM) [109], can match GPS coordinates of walking traces to the nearest road or pedestrian paths. However, these techniques may fail if GPS continuously drifts, potentially causing incorrect mapping to the opposite side of the road.

Related to the above, some approaches track the direction of the pedestrian using smartphone sensors. Roy et al. proposed "WalkCompass" to estimate the user's walking direction from a smartphone [119]. In particular, they compute the motion vector using the accelerometer and gyroscope, calibrate the compass sensor, and determine the user's walking direction. However, the direction accuracy drops significantly if the user swings their smartphone or changes its position. Zhou et al. proposed a smartphone orientation estimation method, A^3, using the gyroscope, accelerometer, and compass sensor on commodity smartphones [169]. Shen [127] further improved the estima-

tion accuracy by introducing the magnetometer sensor for calibration and proposed MUSE. As the state-of-the-art methods for estimating smartphone orientation, A^3 and MUSE yield high accuracy when environmental magnetic field interference is small.

These methods are limited because smartphone orientation does not provide a direct indication of the pedestrian's actual heading, and they are highly volatile in response to common human behaviors, such as the swinging of the smartphone or variations in the pedestrian's gait. *PedHat* addresses this problem by introducing a simple but practical algorithm to estimate the pedestrian's actual heading. More importantly, it introduces a new model to predict crossing based on recent mobility features.

3.7.2.2 Machine learning-based Approaches

Machine learning-based methods reconstruct the pedestrian's trajectory through training models on past IMU sensor data. Chen et al. proposed IONet to learn pedestrian trajectory from IMU measurement data through Recurrent Neural Networks and directly reconstruct the pedestrian's trajectory [28]. Although IONet shows better accuracy than traditional methods, its errors accumulate rapidly as the user continues moving. DeepIT significantly improves trajectory estimation accuracy over IONet with the assistance of earbud sensor data [56]. With sensor data from both a smartphone and an earbud, DeepIT estimated the reliability of sensor data and compensates for the angle drift caused by sensor inaccuracies. Both IONet and DeepIT achieve relatively high precision in estimating the spatial translation of a user's location while less accurate in estimating directional orientation.

3.8 Limitations

We describe the limitations in the evaluation and design of *PedHat*.

OHA Algorithm: During the initialization of R_{LP}, we approximate the pedestrian heading with GPS bearing. However, the OHA algorithm might produce distorted headings due to erroneous GPS signals encountered during this

phase. OHA's effectiveness also diminishes if the GPS heading is persistently inaccurate or if pedestrians intentionally position their phones in locations with the same roll-pitch pairs. Additionally, two relative orientation matrix could share identical θ and ψ angles, leading to OHA heading errors. However, given a large number of possible (r, p) pairs, specifically 16,200 for quantization factor as 2, the collision likelihood is minimal. To mitigate this, *PedHat* restarts the OHA algorithm whenever a pedestrian begins a new walk.

Time-to-crossing: *PedHat* achieves performance comparable to camera-based pedestrian crossing prediction techniques, while maintaining a relatively higher precision and recall rate. Additionally, *PedHat* issues alerts before a pedestrian crosses the road edge, providing a one to two-second window before the pedestrian reaches the potential collision point. However, considering the wireless broadcasting delay and the driver's response time, nearby drivers might not have sufficient time to maneuver. Assuming the vehicle travels at speeds between 25 and 40 mph on flat, dry urban roads, and considering a driver response time of 1 second, the vehicle can come to a full stop within a stopping distance ranging from 20 meters to 41.2 meters [135]. Therefore, our system can effectively alert nearby drivers out of this stopping distance. Meanwhile, our future work will focus on personalizing the prediction model to further increase time-to-crossing.

False Alarms: GPS delay causes both false alarms and missed crossing alerts since it postpones *PedHat*'s prediction on when the pedestrian should enter the road range. Future work will focus on monitoring pedestrians' motion toward the road and validating if the GPS signal has been frozen. Adding reliability metrics on coarse-grained distance and heading features should further eliminate false alarms.

4 COMMUNICATION

This chapter describes our activities related to establishing an infrastructure-free medium for communicating the safety messages between the different road users.

4.1 Single-hop Communication

We start by describing the efforts we conducted to achieve single-hop communication.

4.1.1 Wireless Communication Options

We investigated the available communication methods for peer-to-peer communication in mobile ad hoc networks. We found a few solutions: DSRC (Dedicated Short-Range Communication), Wi-Fi Direct, and Wi-Fi Aware communication. The challenge for employing DSRC is that it is not available for smartphones without support from Qualcomm, and on-board units with DSRC embedded cost about $2,000 each – not affordable for each road user. Wi-Fi Direct allows mobile devices to discover and interact with nearby devices. This technique can be used in safety message dissemination in an emergency [38]. However, the setup time in standard, autonomous, and persistent mode is over 3 seconds, which is not tolerable in the vehicle-to-pedestrian alert scenario. The Listen Channel Randomization for Wi-Fi Direct Device discovery [133] can significantly reduce the discover time during the listen and request phase. This randomization can be done by modifying the WPA supplicant source code. We implemented a Wi-Fi Direct app in two smartphones and tested the communication range and setup time. The communication range reaches 360 feet, but the setup time can take as long as 3 to 8 seconds and increases with distance. Furthermore, this communication suffers when there are obstacles between the two phones.

Beacon-Stuffing [26] provides another method for peer-to-peer communication without prior associations. The process piggybacks on the IEEE 802.11

beacons to send out additional information. However, this method requires changes to the Wi-Fi driver, which is not available on smartphones. Wi-Fi Aware [1] is another technology that has similar characteristics to Wi-Fi Direct. The difference is that Wi-Fi Aware enables the smartphone to send a limited-length message during the discovery phase. Wi-Fi Aware also allows applications to use Wi-Fi Aware to discover peers and create connections to them. This feature is supported by recent Android versions and devices including Google Pixel 2 and 3. It operates at a typical Wi-Fi range at 2.4 GHz or selective 5 GHz; it uses channel 6 on the Wi-Fi band for discovery.

After comparing it with other available short-range wireless communication methods, we decide to adopt Wi-Fi Aware as the basic message transmission method. A single-hop wireless communication using Wi-Fi Aware consists of one publisher and several subscribers. A publisher publishes services available for subscribers and a subscriber can subscribe to service by service name. In our scenario, the subscribers subscribe to the same service the publisher publishes. Setting up a publisher-to-subscriber session requires two phases, discovering other devices and creating a network connection. Subscribers within a certain range can discover the publisher over the air in the first phase and exchange a short message to establish a bi-directional Wi-Fi Aware network connection. The first phase takes hundred of milliseconds and the second takes more than 1 second. We found that messages with limited length (less than 256 bytes) can be embedded in the discover beacon packet during one discovery phase. This creates a single directional message sending method in the discovering phase from the publisher to the subscriber.

4.1.2 Single-hop Broadcasting Mechanism

We build the crossing alert broadcasting mechanism on Wi-Fi Aware protocol [1], also called Neighbor Awareness Networking (NAN). This protocol allows for peer-to-peer connections without the need for a traditional Wi-Fi network or access point. In Wi-Fi Aware, there are two key roles, publisher and subscriber, which correspond to pedestrians and drivers, respectively, in our context. A pedestrian, acting as the publisher, transmits Service Dis-

covery Frames (SDFs) containing service information. This enables drivers, as subscribers, to detect the presence of pedestrians. Instead of establishing connections, pedestrian embeds crossing alert (up to 255 bytes) inside this SDF, also called beacon stuffing [26]. Therefore, a pedestrian who is going to cross can publish a crossing alert to specific nearby drivers, who are actively listening for a matching publisher.

To minimize the discovery latency, we assign specific service identifiers and matching filters for pedestrians and drivers. Therefore, nearby users without these specific service names or matching filters will not be notified, avoiding spamming all users in the proximity.

4.1.3 Real-world Experiments

We evaluated the wireless delay, transmission range and success rate of our alert broadcasting method in real-world experiments.

4.1.3.1 Delay and Range Measurements

We measured the single-hop delay of Wi-Fi Aware using only one packet, in several scenarios. In the line of sight (LoS) scenario, we measured the range and latency with two researchers each holding an Android Pixel 3. The maximum distance for communication is 320 meters and the average latency is about 610 milliseconds. The non-line of sight (NLoS) scenario has a maximum communication range of 140 meters and an average latency of 430 milliseconds. We also measured the dynamic Wi-Fi Aware communication in an empty parking lot. One researcher approaches another in a vehicle at 30 mph and the maximum range is about 140 meters and the average discovering latency is 400 milliseconds. However, we observed that this range and latency depend on the position of phones inside the vehicle.

To transmit more data, we can partition the content of the message across several beacons after adding sequence numbers. This can be done by serializing Wi-Fi Aware beacon packets sending, which would increase the transmission delay as more packets are sent. In our later testing of one-hop latency with multiple packets in both single-hop and multi-hop scenarios, the average latency

is less than 1 second in the typical Wi-Fi range. The static LoS discovering distance is more than 140 meters and a typical Wi-Fi range of 100 meters steadily shows the latency of 400 milliseconds as shown in table 4.1.

Test date	Hops	Distance	One hop latency (ms)		
07/25	Single	90 m	456	540	-
07/25	Single	140 m	509	361	684
10/09	Multiple	75 m	480	404	581
10/09	Multiple	100 m	265	108	151
10/10	Multiple	140 m	4905	339	5676

Table 4.1: One hop latency testing in multiple scenarios with safety message serialized over multiple packets.

4.1.3.2 Broadcasting transmission success rate in single hop

We conducted real-world experiments to evaluate the transmission delay and success rate of our design on two commodity Android devices. We first synchronize the system time in milliseconds of both devices through Network Time Protocol (NTP). Then, in an open outdoor environment surrounded by buildings, we measure the Time-of-Arrival (ToA) and success rate at varying distances from 20 feet to 330 feet (100 meters). At each distance, the pedestrian device sends out a crossing alert SDF along with its current timestamp, and the driver device receives the alert and records the ToA. If the SDF is not received or ToA exceeds 1s, the transmission is marked as a lost alert. We repeated this process 50 times for each distance and present the discovery latency and success rate in Figure 4.1.

Generally, the majority of discovery latency measurements range from 200ms to 400ms over varying distances, while the transmission success rate remains above 90%. Especially at 100 meters, the median discovery latency is less than 400ms, which is notable. In our implementation, to further enhance the transmission success rate, we initiate a second SDF 400ms after the first broadcasting.

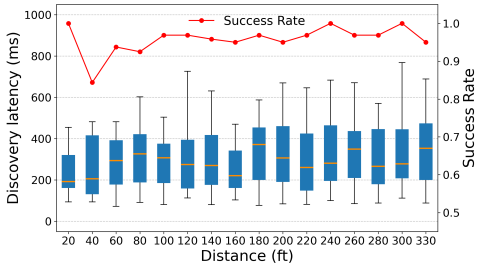


Figure 4.1: Broadcasting crossing alerts latency and transmission success rate using Wi-Fi Aware beacon stuffing at varying distances.

4.2 Multi-hop Communication

After establishing that Wi-Fi Aware is a suitable technology for single hop delay, we investigate how to realize multi-hop communication. This task includes designing a forwarding algorithm to extend the range of communication.

4.2.1 Distance based forwarding algorithms

The project plan included a milestone of simulating the multi-hop network to demonstrate the feasibility of the proposed framework. On its own, we decided that simulating the approach would not yield any interesting insights. After all, previous research has extensively simulated multi-hop vehicular networks. Instead, we decided to look at a real-world implementation of the multi-hop network using Android's Wi-Fi Aware as the underlying communication substrate. In the following, we offer a more detailed description of the multi-hop network design.

First, each mobile device in the network is assumed to support Wi-Fi Aware. The device that runs our developed app contains two modules: publisher and subscriber. The publisher module is responsible for broadcasting the available "services" through beacons. We leverage a technique call beacon stuffing to broadcast the safety messages to the nearby devices by including/partitioning the safety messages content over consecutive beacons (each supporting up to 255 bytes).

Second, the subscriber module of the app receives the beacons from nearby devices and assembles the safety message. It decides whether to re-broadcast or drop the message according to a forwarding protocol (described next). The app does not need to switch roles; it just passes the received message to simultaneously running the subscriber module.

Third, the forwarding protocol is critical to achieving latency and range requirements without causing a "broadcasting storm." We investigated several distance-based forwarding protocols to meet our objectives. These protocols attempt to extend the communication range by choosing the furthest node as the relay node. The basic idea is that the vehicles set the waiting time inversely proportional to the distance of the origin. This approach can provide the shortest end-to-end delay irrespective of the node density [51, 50, 76, 75].

After comparing several alternatives, we decided to implement the forwarding algorithm with randomly selected backoff values. The backoff value is randomly chosen from a pre-computed dataset based on the distance between the original sender and receiver. As a baseline, we compare this algorithm with one that deterministically sets the backoff value inversely proportional to the distance. While some approaches in literature [76] perform RTS/CTS handshake to mitigate the hidden-terminal problem, we decided to forgo this approach as we favor reducing the communication latency over network congestion. The approach we follow builds on the smart broadcast protocol [51, 50].

Fourth, a receiving node only forwards the message using the protocol above if (1) it is moving to the direction of the sender, (2) it is less than 1 km far from the sender, (3) the message is less than 30 seconds old, and (4) this is the very first time it receives a message with a specific ID. These constraints employ a spatio-temporal box that limits the dissemination of the message in the network.

Our current implementation to relay the messages is as the following:

1. Partition region into datasets depending on distance between transmitter and receiver.

2. Associate each dataset with a maximum back-off value, inversely related

to distance.

3. Randomly select back-off values from the appropriate pre-computed data set.

4. Only vehicles moving in the direction of the source forward.

5. Message becomes stale beyond a distance from the original transmitter and time of sending.

4.2.2 Implementation of Wi-Fi Aware based Message Communication

We implemented the forwarding approach as an Android app, following the description above. We evaluated the efficacy of the forwarding algorithm in forwarding the safety message in two scenarios. The first is an NLoS scenario with three nodes; the first and third nodes cannot establish a one-hop link as they are at the orthogonal sides of the intersection (Fig. 4.2). Adding an intermediate node between both of them at an intersection achieved two objectives: (1) node C can receive messages from A forwarded through B, and (2) the latency between A and C is around 800 ms.

Figure 4.2: NLoS Scenario for multi-hop communication

The second is an LoS scenario where two nodes are separated by a larger distance (Fig. 4.3). Note that using Wi-Fi Aware out of the box results in a very high latency between A and C (more than 4 seconds). The reason for this increase in latency is attributed to the frequency range and scanning schedule

of Wi-Fi Aware. Again, adding an intermediate node between them reduces the communication latency between the two nodes to less than 1 second.

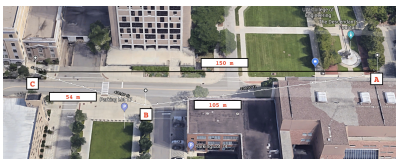


Figure 4.3: LoS Scenario for multi-hop communication

4.3 Experiments and Simulation

This section introduces the simulation scenarios and results of our proposed distance-based forwarding algorithm. We first simulated the traffic using the open-sourced traffic simulation tool, SUMO [24]. As evident from Fig. 4.4, we selected a region from the UW-Madison campus and simulated the traffic on roads with cars, buses, cyclists, and pedestrians included. The black lines in Fig. 4.4 represents the roads, and the yellow blocks within the black lines represent moving vehicles on road. SUMO simulated these road users traveling on roads without interrupting each other. From the traffic simulation, we first generated the simulated movement trace data of all the users to be used for the network simulation later.

Since we already implemented single-hop message transmission on smartphones, we decided to use the measurements to define the single-hop channel model. We use this model to simulate the message forwarding algorithm using a custom-developed Python script. Combined with the measured per-hop

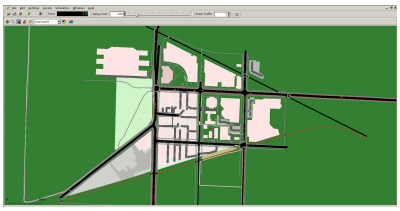


Figure 4.4: Traffic simulation using SUMO. Small yellow block represents moving vehicles.

delay from the real scenario, we simulated the transmission delay with respect to the change of back-off value in our algorithm.

As introduced in Sec. 4.2.1, the message forwarding algorithm decides on the back-off value for each node depending on its distance to the sender. This algorithm aims to address the network congestion problem in the vehicular ad hoc network. Each receiver is able to re-broadcast their received alert messages if the message is not stale beyond a distance and time. If all the receivers decide to rebroadcast the messages, the wireless channel will be congested, increasing the message error rate. This situation is more severe if there are more original publishers broadcasting their alert messages. In our algorithm, each receiver has to wait for a back-off time and check if the channel is clear to send. The back-off value significantly influences the transmission delay; a low back-off value may not relieve the network congestion problem while the large back-off value may lead to increased delay. In our algorithm, the region is partitioned into datasets depending on the distance between transmitter and receiver. In our simulation, the region is partitioned into three equal-sized sections with different back-off values datasets. The dataset of the furthest section is $\{0, 1, ..., cw - 1\}$ and the nearest is $\{2cw, 2cw + 1, ..., 3cw - 1\}$. cw, the containment window, which determines the values of the datasets, is tuned to find the optimized transmission delay in our simulation. The back-off value of each node is randomly selected in a dataset according to which section they are in.

The simulation scenario is as follows: 10 vehicles are randomly placed on a 400 meter long straight road. The main communication path takes place on this stretch of road. In our simulation, at least 4 hops transmission in the 400 meters straight road is needed, since the most efficient transmission range with low latency and reasonable range is 100 meters and vehicles are not expected to be at the transmission range boundary for each message. The node at the origin will first broadcast the alert message and each node on this road will broadcast their received alert message to the road users behind them. One-hop latency is selected as 400 ms according to our experiments, and the node with a larger cw value will be suppressed by the node with lower cw and stop broadcasting after a broadcasting collision. Collision in the scenario without

the back-off algorithm will also cause a latency increase in which one random node will broadcast again after 100 ms.

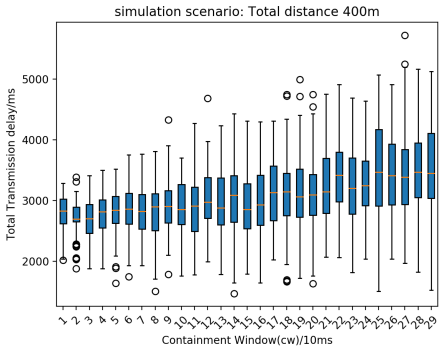


Figure 4.5: Network simulation with back-off algorithm.

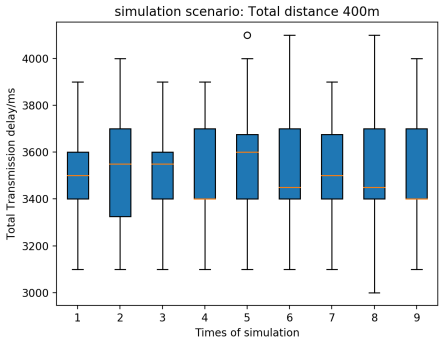


Figure 4.6: Simulation of network traffic without back-off algorithm.

Fig. 4.5 shows the results of the simulation of our distance-based algorithm. The average transmission delay increases as the containment window (cw) increases. When the cw value is less than 160 ms, the average delay varies between 2700ms and 2900ms. After that, the average delay increases when the value of cw increases. When the value of cw is equal to 100 ms, the average transmission delay is 2800 ms and the minimum is 1800 ms. As the containment window changes from 10ms to 290ms, the minimum of the transmission delay can still be as low as 1600 ms which means the vehicles are right at the maximum transmission boundary. Note that a vehicle traveling at 40 mph

needs 23 seconds to reach the source of the message. A transmission delay of 2 or 3 seconds provides the driver ample time to react.

Fig. 4.6 represents the simulation scenario without the network congestion control algorithm. The collisions in this scenario lead to an average delay of more than 3400 ms, and the minimum transmission delay is more than 3000 ms even when the vehicles are right at the maximum transmission boundary.

4.4 Broadcasting Alerts Limitations

We demonstrated a proof-of-concept for broadcasting crossing alerts to specific nearby road users. However, our method has several limitations concerning security, network congestion, and failure handling. Malicious attackers could undermine the trust in our system by sending fake crossing alerts if they obtain our Match-Filter matching sequence and service-specific string. They might also jam nearby Wi-Fi channels, causing network congestion. Additionally, since our method primarily broadcasts crossing alerts without establishing connections with the receivers, we cannot guarantee that drivers receive the alerts. Future works will focus on mitigating these problems.

5 HUMAN FACTORS

With the steadily increasing pedestrian fatalities, pedestrian safety is a growing concern, especially in urban environments. Advanced Driver Assistance Systems (ADAS) have been developed to mitigate road user risks by predicting potential pedestrian crossings and issuing timely driver alerts. However, there is limited understanding of how drivers respond to different modalities of alerts, particularly in the presence of false alarms. In this study, we utilized a full-scale driving simulator to compare the effectiveness of different alert modalities, audio-visual (AV), visual-tactile (VT), and audio-visual-tactile (AVT), in alerting drivers to various pedestrian jaywalking events. Our findings reveal that, compared to no alerts, multimodal alerts significantly increased the number of vehicles stopped for pedestrians and the distance to pedestrians when stopped. However, the false alarms negatively impacted driver trust, with some drivers exhibiting excessive caution, alert fatigue and anxiety, even including one instance where a driver fully stopped when no pedestrian was present.

5.1 Introduction

Pedestrians, being the most vulnerable among road users, face high risks of vehicular traffic, which demands proper safety measures [113]. Pedestrian crashes account for a significant portion of the total traffic-related injury and fatality globally [32]. The number of pedestrians killed in traffic crashes have been steadily increasing since 2010, reaching a high of 7,522 in 2022, according to the National Highway Traffic Safety Administration [103]. Nearly 75% of pedestrian fatalities occur at non-intersection locations. Over 75% of pedestrian fatalities occur in dawn, dusk, or night conditions [102, 104]. At the urban level, the interaction between vehicles and pedestrians is often frequent and intricate, making a timely and accurate driver responses critical [93, 60]. Advanced Driver Assistance Systems (ADAS) including pedestrian crossing alerts for drivers mitigate such hazards [78]. These systems use various alert-

ing modalities that include auditory, visual, and tactile to warn drivers and increase driver attention and responsiveness. ADAS provide additional cues to the driver in a timely pedestrian crossing alert, lowering the chances of a crash [59].

Accurate alert systems are critical for providing reliable alerts about potential hazards, minimizing false alarms, and improving driver trust. However, false alarms, or false positives, can potentially occur when the alert system mistakenly signals a potential hazard. These false alarms can negatively affect driver behavior by causing distraction or leading to habituation, where drivers become less responsive to alerts over time. Frequent false alarms erode trust in the system [112], leading drivers to ignore alerts or react less urgently to genuine warnings, thereby increasing crash risks [146]. False alarms can result from sensor inaccuracies, adverse environmental conditions, and system malfunctions. Common contributors include poor sensor calibration and unfavorable weather conditions, such as fog, rain or low-light environments [12].

We conducted a between-subjects experimental study with 48 participants (14 female and 34 male) to explore the impact of false alarms and alert modalities on driving behavior. In our study, participants operated a full-scale driving simulator to drive in a Unity-based, town-scale urban virtual environment featuring simulated pedestrians, vehicles and buildings. We designed four pedestrian road crossing events, including "Right Crossing", "Left Crossing", "Truck Blocking" and "Short Distance" described in Sec. 5.3.2.3. Additionally, we incorporated false alarms with no actual pedestrian crossings and dummy events, such as vehicles blocking lanes or pedestrians walking or running along the street. With regard to the crossing alert system, we assume a trust entity can collect pedestrian information, predict pedestrian crossing, and broadcast crossing alerts to nearby vehicles. Importantly, some crossing alerts can be false alarms due to false predictions, and the crossing pedestrian may not be visible to participants, such as behind trucks, when they receive the alerts. We also included base scenarios without alerts during driving for comparison.

We collected the participants' physical responses to these alerts, such as braking, accelerating or stopping, using sensor readings in the simulator under different road events and alert modalities. Moreover, we evaluated the partic-

ipant feedback for pedestrian crossing alert systems through a post-session questionnaire. Our findings provide insights into how alert systems and false alarms influence driver behavior, summarized as follows:

- Alert systems can enhance the driver's awareness of crossing pedestrians, especially when pedestrians are outside of their direct focus, such as on the left side of the road.
- Drivers may exhibit reduced trust in alert systems with false alarms (e.g., fewer braking actions), particularly when pedestrians are not visible at the time of the alert.
- Excessive alerts can heighten driver anxiety, potentially leading to unnecessary or dangerous behaviors.

5.2 Related Work

5.2.1 Pedestrian Crossing Prediction

Recent research on pedestrian crossing prediction focused on the prediction of the intention of pedestrian crossing through cameras or LiDAR sensors on roadside infrastructure and vehicles. Current methods for predicting pedestrian crossing intent primarily rely on visual information obtained from cameras or LiDAR sensors mounted on vehicles [154] or roadside infrastructure, such as CCTV cameras installed at intersections [163]. These methods analyze captured images to identify various aspects of the pedestrian's current state, including their motion [143, 142], past trajectory [166], skeletal structure [163, 48, 116, 88], and even demographic factors such as age and gender [162]. By applying deep learning techniques to these data [160], researchers aim to accurately determine whether a pedestrian intends to cross the street, which is vital for preventing crashes and ensuring a safer road environment.

5.2.2 Development of ADAS and Warning Modalities

ADAS have evolved to include a range of warning systems that alert drivers to potential dangers. Early systems relied primarily on auditory alerts, us-

ing sounds to capture attention effectively [126]. While these are effective in cutting through visual distractions, their impact can be diminished by environmental noise or driver habituation [19]. Visual alerts, delivered through dashboard indicators or head-up displays (HUDs), offer clear, actionable warnings without overwhelming the driver. Tactile alerts, such as vibrations in the steering wheel or seat, provide an additional layer of feedback, particularly useful when visual or auditory channels are overloaded [45].

5.2.2.1 Types of Driver Alerts

Alerting systems play a crucial role in enhancing driver awareness and ensuring timely responses.

- **Auditory Alerts:** These use sound, such as beeps or alarms, to warn drivers of hazards, with their effectiveness depending on factors like sound frequency, amplitude, and duration [54]. Properly designed auditory alerts improve safety by enhancing driver focus and response [79, 91].

- **Visual Alerts:** Visual alerts, often displayed on dashboards or HUDs, provide critical cues about potential dangers. High-contrast colors and strategic placement in the driver's line of sight enhance their effectiveness [134]. Augmented Reality HUDs further improve pedestrian awareness without adding distraction [73].

- **Tactile Alerts:** These alerts, delivered through vibrations in the steering wheel, seat, or pedals, offer a non-visual, non-auditory method of communication. They are particularly effective in noisy or visually cluttered environments, ensuring crucial alerts are noticed [37, 65].

- **Multi-Modal Alert Systems:** Modern ADAS increasingly incorporate multi-modal systems that combine auditory, visual, and tactile alerts. This redundancy increases the likelihood that drivers will notice and respond to warnings, making these systems highly adaptable to various driving conditions [39, 91].

5.2.3 Driving Simulators

Recent advancements in driving simulators have improved virtual environments for studying driver behavior and road interactions. Goedicke et al.[55] developed a mixed reality simulator where participants drive a real car while wearing a VR headset, interacting with virtual objects integrated into the real world. Bu et al.[21] extended traditional in-lab simulators to an on-road platform, enabling comparative studies between in-lab and real-world driving. Hou et al.[64] used VR to explore interactions between autonomous vehicles and cyclists, while Tateyama et al.[137] created a 180-degree simulator to study turning behaviors. Our simulator enhances the immersive experience with side and rear view mirrors and a Unity-based simulated environment.

5.3 Experiment Methods

Our study investigates two main research questions: (1) *What warning modality is most effective in alerting drivers to pedestrian jaywalking?* (2) *How do drivers react, both behaviorally and emotionally, to false alarms within pedestrian crossing alert systems?*

We conducted a two-phase experiment to address the above research questions. In the first phase, the participants completed driving sessions with different warning modalities, during which we collected the sensor data in the driving simulator. In the second phase, participants provided feedback on the warning system by completing a post-session questionnaire. Our study was approved for exemption by the university's Institutional Review Board (IRB).

5.3.1 Recruitment Procedure

Before participating in the driving simulator study, each participant read and signed a consent form, indicating their willingness to participate in the study. Next, they were introduced to the experiment details through a pre-recorded video, which provided the necessary instructions on how to follow the designated path and guidance on stopping the simulation in case of motion sickness or discomfort during driving.

Participants were not informed about the pedestrian crossing alert system or specific road scenarios prior to the experiments. They were simply instructed to follow the virtual turning arrows near intersections and drive as they would in a real-world environment. The participants had complete control over the vehicle, including speed control, avoiding other road users, and deciding when to stop.

5.3.1.1 Participant Sample Distributions

We recruited participants through our institute's research service and social networks, in an effort to achieve a diverse demographic. 48 participants participated in the study, with 14 females and 34 males. Their ages ranged from 27 to 67 years as shown in Table 5.1. All participants held a valid driver's license and were over 18 years old. One driving session took approximately 30 minutes to complete, and we compensated all participants with a $20 gift card for their participation.

Age Range	25-34	35-44	45-54	55-64	65-74
Number of Participants	25	20	1	1	1

Table 5.1: Age distribution

5.3.2 Experiment Setup

Our experiment involves four driving sessions equipped with four warning modalities. These warning modalities are 1) a base scenario condition without any kind of pedestrian crossing alerts, 2) an audio-visual (AV) warning with beeping sounds and dashboard displays as shown in Figure 5.1, 3) a visual-tactile (VT) warning with dashboard displays and seat vibration as in Figure 5.2, and 4) an audio-visual-tactile (AVT) warning combining the three modalities.

To avoid learning and carryover effects, each participant is exposed to **only one** of the four driving sessions, and therefore each driving session has 12 participants. All four driving sessions share the same sequence of road

events described in Section 5.3.2.3. In the sessions that included warnings, we introduced pedestrian crossing alerts for both actual crossing events and false alarms to assess drivers' reactions. The experiment was setup in an urban driving conditions with varying traffic volumes and pedestrian activity.

Figure 5.1: Visual warning

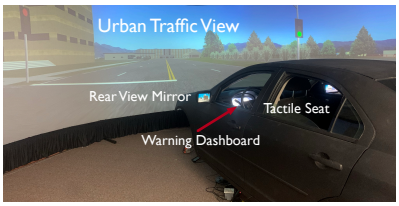


Figure 5.2: Full-scale Driving simulator

5.3.2.1 Full-Scale Driving Simulator

We conducted the experiment in a full-scale driving simulator (shown in Figure 5.2) to study driver behavior and safety within realistic traffic environments. This simulator is based on a standard Ford Fusion, equipped with real driving interfaces for operation and various sensors to monitor essential driving input. This simulator offers a virtual platform that mimics real-world driving conditions, allowing researchers to explore how drivers respond to various traffic scenarios in a controlled and safe environment. It plays a crucial role in understanding driver performance and safety, especially in simulating both daily driving environment and high-risk driving situations.

Data Collection Operating at a 60-Hz sampling rate, the driving simulator captures key metrics such as speed, lane position, steering angle, brake pedal position, and gas pedal position. This high-frequency data collection enables an in-depth analysis of driver behavior across diverse environments, from urban intersections to highways.

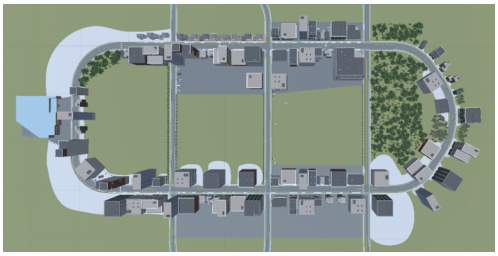


Figure 5.3: Overview of urban driving scenarios

5.3.2.2 Driving Session Design in Unity

We developed the driving sessions with multiple pedestrian jaywalking events using the Unity 3D game engine. The bird-view in Figure 5.3 shows the road map for the entire scene. We designed various buildings, multiple intersections, simulated pedestrian walking, and simulated vehicles driving randomly in a highly detailed and interactive environment that mirrors the real-world driving. We included realistic pedestrian behaviors, such as walking along the street, walking behind trucks, and abrupt jaywalking using Unity's physics engine. Our driving session provides a robust testing ground for evaluating driver reactions toward the pedestrian crossing alert system.

As shown in Figure 5.4, the driving session utilizes a closed-loop track formed by four loops connecting end to start. In other words, the end point of each loop is the start point of the next loop. The total driving route length is 4.6 miles. Each loop includes curves, tangents, and intersections to add complexity to driving environment. For each driving session, pedestrian jaywalking events are evenly distributed across four loops as described in Sec 5.3.2.3.

The simulator generates pedestrian crossing alerts based on the proximity of the vehicle. When the vehicle operated by the participant approaches the event location, a simulated pedestrian begins walking and subsequently crosses the road. The participant can visually observe the pedestrian unless obscured by a truck.

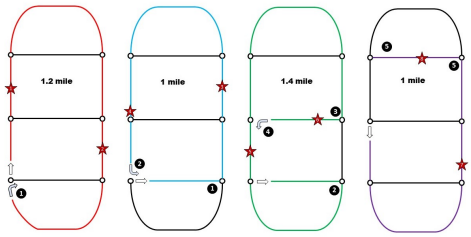


Figure 5.4: Run Schematic: A four-loop driving route. Colored lines indicated the driving path, while black lines represent paths not driven. Numbers in circles denote key connection points beween loops; for example, the number 1 in the first and second loop marks the end of first loop and the start of next loop. Numbers in stars represent actual pedestrian jaywalking events.

(a) Right Crossing (b) Left Crossing (c) Truck Blocking

Figure 5.5: Simulated Driving Environment Scenarios

5.3.2.3 Road Events

We designed different road events to test the effectiveness of pedestrian crossing alert system under various conditions. There are four road event types with actual pedestrian jaywalking that trigger pedestrian crossing alerts: right crossing, left crossing, short distance and truck blocking. False alarm event also triggers the pedestrian crossing alert, but no pedestrian crosses. Each event type occurs twice in a driving session and differs from other types in crossing side, obstacles and warning activation distance. Details of road event types are detailed in Table 5.2 and as follows:

- **Right crossing**: A pedestrian enters the road from the right side of the road, and the pedestrian crossing alert is triggered when the distance between the vehicle and the pedestrian reaches 200 feet. This serves as a basic scenario, with other crossing event type differing from it in one aspect.

- **Truck Blocking**: A pedestrian crosses behind a truck parked on the road side, and is not in the participant's line of sight when they begin crossing.
- **Left Crossing**: Pedestrian enters road from the left side of the road, offering a variation on the direction of crossing.
- **Short Distance**: The participant receives a warning when the distance between the vehicle and the pedestrian reaches 160 feet instead of 200 feet.
- **Dummy Event**: Dummy events involves multiple events that do no directly impact the vehicle's normal driving, such as stop sign runners, vehicles blocking the lane, or pedestrians walking along the street. Dummy events are intended to distract the participants from only focusing on crossing pedestrians. No pedestrian crossing alert will be triggered in a "Dummy Event."
- **False Alarm:** No pedestrian crosses during false alarms, though the participant receives a pedestrian crossing alert. In the first false alarm, a truck is parked by the right lane. In the second, pedestrians are walking along the street. These scenarios are similar to "Dummy Events", but the participant receives pedestrian crossing alert in "False Alarm".

Event name	Number of Events	Crossing Side	Obstacle	Warning Activation Distance
Left Crossing	2	Left	None	200 feet
Short Distance	2	Right	None	160 feet
Truck Blocking	2	Right	Blocked by truck	200 feet
Right Crossing	2	Right	None	200 feet

Table 5.2: Pedestrian jaywalking events for each driving session.

We carefully designed the sequence of road events to achieve a balanced distribution of different crossing events, false alarms, and dummy events. Each sequence consists of eight actual crossing events, five dummy events, and two false alarms events, as illustrated in Figure 5.6. Driving sessions with different warning modalities share the same sequence of events. Only during the crossing events in hexagon shape, there are jaywalkers crossing the road

in the middle of the block. Stop signs are placed at road intersections and they are not part of any event type.

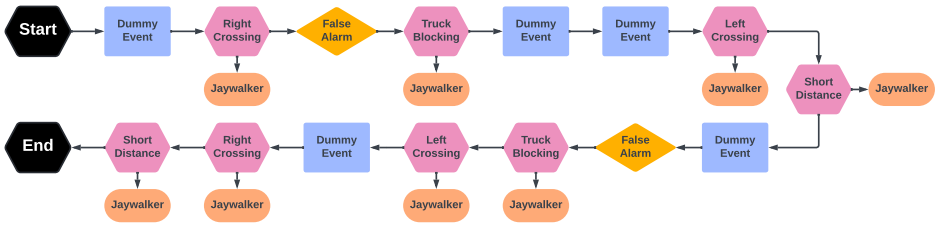


Figure 5.6: Sequence of events in one driving session, including six types of events, Right Crossing, Left Crossing, Truck Blocking, Short Distance, Dummy Event, and False Alarm. Each session will utilize one of four warning modes, 1) no warning, 2) Audio-Visual (AV), 3) Visual-Tactile (VT), and 4) Audio-Visual-Tactile (AVT).

5.3.3 Post-session Questionnaire

After completing the driving session, participants were asked to fill out a questionnaire to assess their overall experience with the pedestrian crossing alert system, as well as their attitudes toward false alarms during driving. The survey consisted of first four Likert-scale questions and one last open-ended question, as follows:

1. How clear were the system's alerts for crossing pedestrians?
2. How timely do you find the pedestrian crossing alerts?
3. How would you rate your overall confidence in the system for pedestrian crossing alerts?
4. Would you recommend this pedestrian warning system to other drivers?
5. How did the system's false alerts or failures to detect pedestrian crossing scenarios affect your driving? (you do not need to answer if you have no alert)

The last question is optional for participants in base scenarios.

5.4 Data Analysis and Findings

We comprehensively evaluated participants' responses to the pedestrian crossing alert system across four warning modalities through driving sessions and post-session questionnaire. We begin with analyzing participant reactions during the driving sessions, followed by an exploration of the participants' responses to the post-session questionnaire. Our objective is twofold: first, to examine how the pedestrian crossing alert system supports participants in making driving decisions, and second, to assess participants' reactions to false alarms when no pedestrians are present.

5.4.1 Participant Reaction in Driving Session

Using vehicle dynamics data collected from the driving simulator, we computed the following performance indicators to assess driver behavior during the pedestrian jaywalking events and false alarms:

- **Vehicle Stopping Count:** Whether vehicles fully stopped upon receiving a pedestrian crossing alert.
- **Brake Initiation Count:** Whether participants applied the brakes in response to a pedestrian crossing alert.
- **Average Stopping Distance:** The distance between the vehicle and the pedestrian when vehicle stopped.
- **Average Speed After Events:** The average vehicle speed after the road event and before the next event.

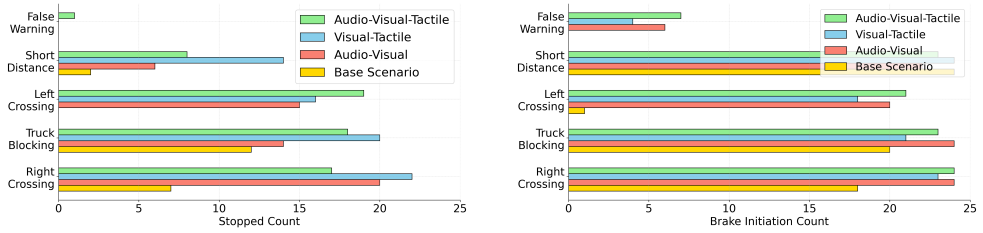
We analyzed these indicators across different road events in four driving sessions. Each of the four driving sessions had 12 participants, and during each session, participants encountered each type of crossing event twice, resulting in a total of 24 data points per event type.

5.4.1.1 Vehicle Stopping

We first evaluated the impact of the alert system on vehicle stopping behavior (Figure 5.7a). Our results indicate that the number of vehicles stopping upon receiving warnings exceeded the number of stops in the no-warning condition across all crossing event types. In the base scenario without any warning, approximately half of the participants stopped and waited for pedestrians in the "Right Crossing" and "Truck Blocking" events. However, only two participants stopped in the "Short Distance" event. For the "Left Crossing" event, no participants stopped, likely due to pedestrians from their left side may have been outside of their immediate focus.

In contrast, under conditions with alerts, around 80% of participants stopped for the "Left crossing", "Truck blocking" and "Right crossing" events, while around one-third participants stopped for "Short distance."

Interestingly, we did not observe significant differences in stopping behavior between the three warning modalities. Among the four pedestrian jaywalking events, the "VT" modality slightly exceeded the other two in terms of number of stops, except for the "Left Crossing" event. It is also worth noting that one participant stopped for a false alarm during the session with "AVT" warning modality, whereas no participants with other warning modalities stopped.



(a) (b)

Figure 5.7: (a) Stopping vehicles count and (b) brake initiation count upon road events across scenarios

5.4.1.2 Brake Initiation

We examined the number of brake initiations upon receiving pedestrian crossing alerts or observing pedestrian jaywalking across driving sessions.

In the "Short Distance," "Truck Blocking," and "Right Crossing" events, we observed similar numbers of participants applying brakes across both the warning scenarios and the baseline scenario without any warnings. However, in the baseline condition for the "Left Crossing" event, only one participant initiated braking. This aligns with our earlier observation that no participants stopped during the "Left Crossing" event in the no-warning condition. It suggests that only a small fraction of participants noticed the pedestrian jaywalking from the left or recognized it as a potential collision risk.

Conversely, when participants received alerts for the "Left Crossing" event, approximately 80% applied the brakes and came to a stop. Such difference underscores the effectiveness of the alert system in drawing participants' attention to crossing pedestrians, particularly in scenarios where pedestrians may be less noticeable.

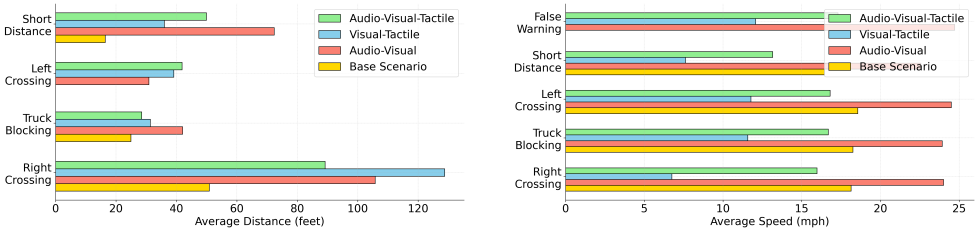
For the "False Warning" events where no pedestrians cross, no participant initiated brakes in baseline scenario. Around 20% participants applied brakes in driving sessions with warnings, indicating cautious response toward alerts.

5.4.1.3 Average Stopping Distance

The average distance between the vehicle and the pedestrian at the point of vehicle stopping varied across different scenarios (Figure 5.8a). For this analysis, we excluded the "False Alarm" events, focusing solely on pedestrian jaywalking events.

Our findings indicate that all alert modalities enhanced stopping distances compared to the baseline especially in "Left Crossing" and "Right Crossing", with the AV modality resulting in the highest increase overall. The base scenario without any warning had the shortest average distance of 31.30 feet. In contrast, the warning modalities increased the stopping distance by 60% to 90%, with the AV modality yielding the largest average distance at 59.89 feet, followed closely by the VT at 59.28 feet, and the AVT modality at 49.80 feet.

Regardless of warning modalities, participants may apply brakes differently depending on the visibility of pedestrians. Among the event types, the "Right Crossing" events resulted in the longest stopping distances, while the "Truck Blocking" events produced the shortest, even shorter than those observed in the "Short Distance" events. Since participants received alerts at the same distance from pedestrians in all scenarios, this suggests that participants applied the brakes later or less forcefully in situations where they could not clearly see the pedestrians, such as during "Truck Blocking" events. Similarly, the "Left Crossing" event, where pedestrians were less obvious, also resulted in relatively shorter distances. In contrast, in the "Right Crossing" events, where the pedestrian was clearly visible, participants responded more quickly and stopped at a greater distance.



(a) (b)

Figure 5.8: (a) Average distance for each event when car stops and (b) average speed after each event for each scenario.

5.4.1.4 Average Speed

The average speed after one event completes and before the next event starts provides insights into how pedestrian crossing alerts impact driving behavior.

As different warning modalities may influence participants' driving behavior in varying ways, the average speed after each event reflect these potential changes, such as how cautious drivers are in anticipation of future events. Our findings indicate that VT warning modality may have the strongest effect on increasing driver awareness of potential future collisions. Participants who experienced the AV warning modality maintained the highest average speed, while those in the VT modality showed the lowest. In comparison, the base

scenario, where no warning was provided, resulted in slightly higher average speeds than both the AVT and VT modalities. Note that we do not have the measurement result for base scenario in "False Warning" and leave it blank.

5.4.1.5 Statistical Analysis

We conducted an Analysis of Variance (ANOVA [106]) test to evaluate the performance measures in the scenarios and use the base scenario as the control group. ANOVA allows for the comparison of means across multiple groups to identify statistically significant differences between them and the results are presented in Table 5.3. The null hypothesis (H_0) is that there is no significant difference in the means of performance measures among the scenarios. The results included F-statistics, which represent the ratio of the variance between groups and the variance within groups. The greater the F-statistic value, the more likely that the differences found are statistically significant. From the F-statistic, a p-value is computed, based on a statistical distribution known as the F-distribution under the condition that the null hypothesis is true.

For Average Distance, the AV modality had the highest improvement compared to the base scenario, while the AVT and VT modalities also showed differences. For Stopped Count, the largest increase in stopped vehicles was observed in the VT modality followed by AVT and AV scenarios. The Enhancement is the difference between the one modality's average and the base average, with positive values representing improved performance. In the AV scenario, the enhancement for average distance was 31.92 feet, which means the vehicles stopped 31.92 feet farther from the pedestrian than in the base condition. A similar positive enhancement in the stopped count would suggest that more vehicles acted on the warning system in the test scenarios compared to the base scenario. For instance, the VT modality showed an enhancement of 11 vehicles stopped, indicating improved performance over the base condition.

Scenario	Variable (units)	F-statistic	p-value	Base Scenario Average	Scenario Average	Enhancement
AVT	Average Distance (ft)	1.686	0.251	31.11	52.91	22.8
VT	Average Distance (ft)	1.206	0.333	30.11	58.84	28.73
AV	Average Distance (ft)	2.634	0.169	30.11	62.03	31.92
VT	Average Stopped Count (number)	5.1	0.061	7	18	11
AVT	Average Stopped Count (number)	3.986	0.086	7	15.75	8.75
AV	Stopped Count (number)	2.563	0.154	7	13.75	6.75

Table 5.3: Comparison of Scenarios with the Base Scenario

5.4.2 Participant Feedback in Post-Session Questionnaire

5.4.2.1 Satisfaction toward Pedestrian Alert System

We evaluated participants' satisfaction towards the pedestrian alert system through the Likert-scale questions in the post-session questionnaire. We evaluated several key aspects, including the clarity, timeliness, and overall confidence in the system, as well as whether they would recommend the system to others. Below is a summary of the key findings from the post-session questionnaire:

- **Clarity of Alerts:** As Figure 5.9 shows, participants highly rated the clarity of the system alerts for pedestrian crossings across all scenarios. While the majority of participants in the AVT scenario found the alert to be "Extremely clear," many in the VT and AV scenarios participants considered the alert either "Very clear" or "Extremely clear."

- **Timeliness of Alerts:** Figure 5.10 shows the timeliness of pedestrian crossing alerts. Of all scenarios, the majority rated the alerts either "Quite timely" or "Very timely." It is also worth mentioning that, out of all scenarios assessed, the highest percentage rating for "Quite timely" is the AVT scenario.

- **System Recommendation:** When asked whether they would suggest the pedestrian alert system to other drivers, 75% of participants in every scenario responded positively.

- **Overall Confidence:** In general, confidence in the pedestrian warning systems was on a scale of 1 to 4, where 4 was also mapped to "Very Confident." The confidence levels were generally high for all scenarios,

while the AVT scenario received the highest level of confidence, followed by the VT scenario.

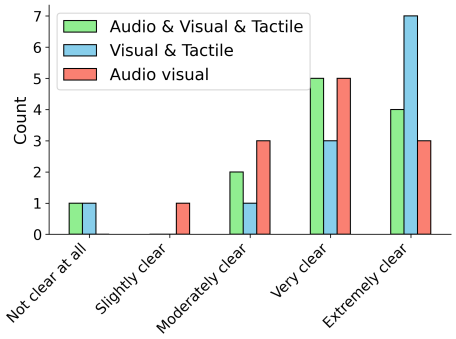


Figure 5.9: Q: How clear were the pedestrian crossing alerts?

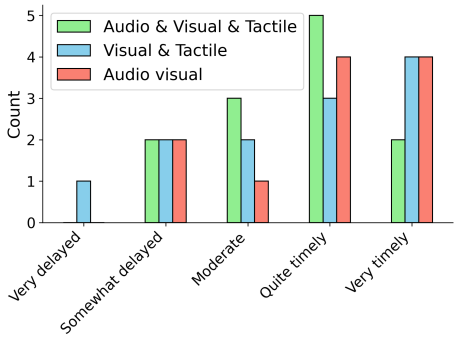


Figure 5.10: Q: How timely were the the pedestrian crossing alerts?

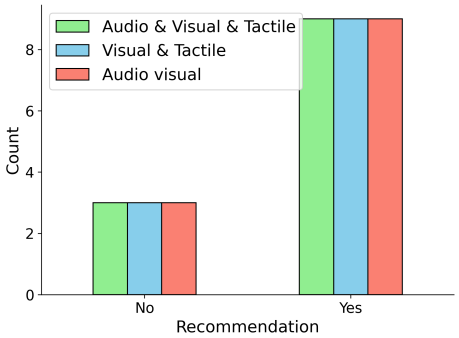


Figure 5.11: Q: Would you recommend this pedestrian warning system to other drivers?

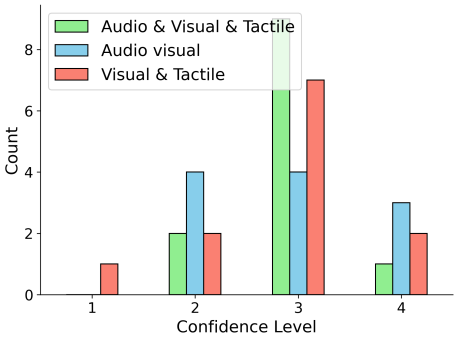


Figure 5.12: Q: How would you rate your overall confidence in the system for pedestrian crossing alerts? (4 is the highest)

5.4.2.2 Perceptions toward False Alarms

We analyzed 36 participants' perceptions towards false alarms during driving sessions through an open-ended question in the post-session questionnaire. Participants were asked one open-ended question: "How did the system's **false alerts** or **failures to detect** pedestrian crossing scenarios affect your driving?".

The purpose of this question was to assess how participants recognized and reacted to false alarms, given that they were not informed beforehand about the pedestrian alert system or the possibility of false alarms.

To understand participants' perceptions toward false alarms, two researchers first reviewed the participants' responses and identified eight common perceptions. Then each researcher independently coded each response using the perceptions as a basis. We show the results in Table 5.4 and describe each perception as follows:

- **Confusion or Uncertainty:** 13 participants expressed confusion regarding the alerts' meaning, with 10 participants unsure if they had missed a pedestrian and 3 others suspecting the system was signaling something unrelated, such as speeding.
- **Stress and Anxiety:** 3 participants who experienced multi-modal alert (including visual, auditory, and haptic cues) reported heightened stress and anxiety while driving. Participant 23 said: "It made me extremely anxious while driving that I was going to hit something." False alarms increased their stress levels, while alerts with multiple modalities amplify their feelings of anxiety.
- **Increased Caution:** 18 Participants indicated that the false alarms caused them to drive more cautiously. They mentioned checking their surroundings more frequently, slowed down, or lifted off the accelerator in response to these alerts.
- **Distracted:** 8 participants reported being distracted by the false alarms, particularly when they were unable to immediately identify the cause. This distraction occasionally shifted their focus away from driving, introducing potential safety risks.
- **Immediate Reactions to Brake:** 6 participants reported that they instinctively applied the brake upon receiving an alert, even if they did not see a pedestrian nearby.

- **Reduced Trust:** Seven participants mentioned that repeated false alerts reduced their trust in the system. Inconsistent or inaccurate alerts led them to question the system's reliability and accuracy, making them more skeptical of its usefulness in real driving scenarios.
- **Alerting Fatigue/Ignoring Alerts:** As the number of false alarms accumulated, 3 participants reported becoming desensitized to the alerts. This led them to ignore subsequent warnings, including potentially legitimate ones, commonly referred to as "alert fatigue."
- **Momentary Startle:** 3 participants reported feeling momentarily startled when the alerts first sounded, though they quickly recovered and resumed driving.

Modality	Confusion	Stress/ Anxiety	Increased Caution	Distracted	Immediate Brake	Reduced Trust	Alert Fatigue	Momentary Startle
A+V	4	0	6	3	1	1	1	2
V+T	6	0	6	2	2	3	2	1
A+V+T	3	3	6	3	3	3	0	0

Table 5.4: 36 Participants' Perceptions toward False Alarms across Modality. Abbreviations: A+V = Audio & Visual, V+T = Visual & Tactile, A+V+T = Audio, Visual & Tactile.

5.5 Discussion

We conducted a driver simulator study and post-session questionnaire to examine participants' physical and psychological responses to the pedestrian crossing alert system. We analyzed participants' immediate reactions, such as braking, stopping, and stopping distances, to understand their physical behavior. We then explored their perceptions of the warnings, particularly regarding false alarms, using the post-session questionnaire. We concluded our findings as follows.

5.5.1 Driver's Excessive Anxiety

Our findings revealed that participants who experienced multi-modal alerts reported heightened stress and anxiety. For instance, three participants explicitly mentioned feeling stressed when exposed to alerts combining all three modalities (Audio-Visual-Tactile) in the questionnaire. Meanwhile, one participant stopped for a false alarm during the session with the AVT warning modality, while no such responses were observed for other modalities. This suggests that combining all three warning modalities may excessively heighten drivers' caution and anxiety, potentially causing overreactions to false alarms and reduced trust. Additionally, when evaluating its effectiveness in alerting drivers, the AVT modality did not exhibit significant differences in stopping distance or average driving speed compared to the Audio-Visual and Visual-Tactile modalities. Therefore, we recommend minimizing over-alerting in situations where false alarms are present to enhance system reliability.

5.5.2 Enhancing Driver Awareness

The pedestrian crossing alert system demonstrated potential in improving drivers' awareness of pedestrians, particularly when the pedestrians were outside the driver's immediate focus. For instance, in the "Left Crossing" event, no participants stopped in the absence of alerts, likely because pedestrians approaching from the left were not within their primary focus. In comparison, one-third of participants without any alert stopped in the "Right Crossing" scenario, where pedestrians were more visible. In terms of braking, only one participant applied brakes in the "Left Crossing" scenario without alerts, whereas most participants braked in the "Right Crossing" scenario. In scenarios with alerts, most participants stopped and braked for both "Left Crossing" and "Right Crossing" events. These findings indicate that drivers may overlook pedestrians approaching from directions or places not in the driver's direct focus while the alert systems can effectively reduce collision risks in these cases.

5.5.3 Trust when see

Our finding also indicated that drivers may apply brakes differently depending on the visibility of pedestrians when they understand the existence of false alarms. We measured this response according to the stop distance between the stopped vehicle and the crossing pedestrian. Stop distance is a critical measure of driver response and road safety. Longer stopping distances provide greater safety margins, particularly in adverse conditions like icy roads.

In the “Right Crossing" scenario, where pedestrians were highly visible, drivers yielded the longest stopping distances. Conversely, the “Truck Blocking" and “Left Crossing" scenarios produced the shortest stopping distances, even less than those observed in the “Short Distance" events. Note that participants received alerts later in “Short Distance" than in “Truck Blocking" and “Left Crossing". This suggests that limited visibility affects driver trust in the alerts and their braking behavior. Compared to the base scenario, alerts can lead to an increase in stopping distances of around 50 feet when pedestrians were visually observable but only improved distances by approximately 10 feet when visibility was restricted, as in the “Truck Blocking" scenario.

Overall, the VT warning modality showed the greatest improvement in stopping counts, enhancing performance by 11 vehicles compared to the baseline. For stopping distances, the AV modality resulted in the most significant improvement, adding an average of 31 feet. While 18 participants reported adopting more cautious driving behaviors, such as slowing down or frequently checking their surroundings, seven participants indicated that repeated false alarms diminished their trust in the system, leading to alert fatigue and reduced responsiveness. Participants exposed to the AVT system experienced the highest levels of stress and distraction, suggesting that multi-modal alerts should be carefully calibrated to avoid negative psychological impacts.

5.6 Limitations

Although this study offers valuable insights, several limitations should be acknowledged. Motion sickness was a significant issue, with 19 out of 67

participants unable to complete the driving sessions. Additionally, the semi-controlled environment of the driving simulator may have influenced participants to behave more cautiously than they might in real-world conditions, potentially underestimating risk-taking behaviors. On the other hand, each driving session included 10 warnings, two of which were false alarms. The relatively high proportion of false alarms may have caused participants to perceive subsequent warnings as unreliable, diminishing their trust in the system. Furthermore, the study's 30-minute driving sessions, featuring 10 warnings and five dummy events, differ from typical driving environments where alerts are less frequent. Statistical analysis also presents a limitation in this study. Analysis of Variance (ANOVA) assumes continuous data with normally distributed residuals, making it inappropriate for small count data, which is typically non-normal and highly skewed. Therefore, conclusions drawn from ANOVA in our case may be unreliable. To address these limitations, we will conduct our study with larger sample sizes and real-world driving conditions. We will validate these findings and provide more generalizable insights in our future work.

5.7 Conclusions

Our driving simulator study evaluated the impact of different pedestrian crossing alert modalities on driver behavior, particularly in the presence of false alarms. Our findings indicate that all tested warning modalities (Audio-Visual, Visual-Tactile, and Audio-Visual-Tactile) can enhance driver response compared to the baseline scenarios with no warning. For instance, compared to the base scenario, the average stopping distance increased by 75% to 105% across different warning modalities, while the stopped count increased by 96% to 157%. Additionally, our study provides insights into how alert systems and false alarms influence driver behavior, including enhancing awareness, reducing trust, and causing unnecessary stopping actions. We will design improved alerting mechanisms for future work, such as tailoring alerts based on the confidence level of pedestrian crossing predictions.

Part II

Bridging Physical and Digital Safety

6 PEDRO: SECURE PEDESTRIAN MOBILITY VERIFICATION IN V2P COMMUNICATION USING COMMERCIAL OFF-THE-SHELF MOBILE DEVICES

Vehicle-to-Pedestrian (V2P) communication enables numerous safety benefits such as real-time collision detection and alert, but poses new security challenges. An imminent and probable scenario is where a malicious node claiming to be a legitimate pedestrian within the network broadcasts false observations or phenomena on the roads (e.g., traffic load, road hazard, and false road crossing alarms) in order to impede traffic flow, erode user's trust in alert messages, or even cause traffic accidents. Therefore, it is crucial to identify legitimate road users against adversaries pretending to be one. In this work, we propose *Pedro*, a **PED**est**R**ian m**O**bility verification mechanism for pedestrians using commodity hardware, where only legitimate mobile pedestrians can be admitted to the ad hoc network consisting of trustworthy vehicles and pedestrians. We leverage the round-trip time (RTT) of wireless signal between vehicle and pedestrian's devices, and verify only moving (mobile) ones while rejecting stationary ones, based on the realistic assumption that the adversaries are likely to remotely launch attacks through static malicious devices. Through an extensive analysis based on simulation as well as real-world experiments, we show that *Pedro*'s verification takes under 8 s while achieving an 8.5% Equal Error Rate (EER) under regular road environments.

6.1 Introduction

Vehicle-to-Pedestrian (V2P) communication is a networking paradigm that involves direct communication between a vehicle and pedestrians within its vicinity. It promises to improve pedestrian safety and reduce roadway fatalities and injuries through exchanging advanced warning messages between pedestrians and drivers [16]. These messages enable the different road users to cooperatively prevent collisions based on their exchanged position, speed, and direction information ahead of time [44, 83]. Systems that use V2P to improve

road safety share a common working principle: detect a road crossing event from the pedestrian's behavior and alert nearby vehicles through wireless messages [150].

Despite its safety benefits, security is one of the biggest barriers to the adoption of V2P communication in safety-critical applications [20, 141, 41]. In particular, false data injection attacks, by which an attacker broadcasts fake/-false warning messages, hinder the safety of road users and undermine the trust in the V2P system [62, 34, 20]. Here, we are concerned with an important attack scenario: an adversary can deploy a wireless node to broadcast false messages about pedestrians disregarding traffic rules (i.e., jaywalking); these messages will cause trailing vehicles to be continuously congested, trying to heed the misguided collision warnings. To prevent such an attack, it is essential to verify the authenticity of the claims of every pedestrian before its messages are accepted by vehicles.

One solution for this problem is deploying a trust infrastructure to develop and maintain trust relationships between pedestrians and vehicles in the V2P system [62]. A road user enrolls in the V2P system by registering and authenticating their identities [34] to the trust infrastructure. While seemingly straightforward, this solution suffers from scalability and usability shortcomings because: 1) the deployment of the trust verification infrastructure is costly, and 2) it requires time-consuming user involvement for registration, especially when visiting new locations [62]. In this project, we advocate for an alternative direction that does not rely on any pre-existing infrastructure and thus does not require explicit registration. We propose a novel approach to *verify the claimed behavior of a pedestrian*, where a vehicle rejects messages until it can conclusively verify that the movement pattern of the sender matches that of a pedestrian.

Existing movement or location verification schemes are limited in terms of their practical V2P use cases. For example, Schafer et al. [122] proposed to verify the sender mobility by leveraging Doppler shift measurements from multiple verifier nodes. Other methods verify the sender's location by utilizing the received signal power in an outdoor environment or a combination of GPS, ad hoc location, or dead reckoning [117, 17, 22]. However, these mechanisms

apply only in limited scenarios as they require carefully positioned verifiers, which might not be readily available in most realistic V2P scenarios [132]. They also need specialized hardware capabilities that are not available in commercial off-the-shelf (COTS) devices. A practical mobility verification scheme should avoid the above shortcomings by meeting the following conditions. First, it should utilize readily available hardware with no or minimal modification. Second, it should operate without pre-distributed verifiers that are costly to deploy and manage.

In this work, we propose *Pedro*, a new mobility verification mechanism that uses commodity smartphones without requiring pre-distributed verifiers. In *Pedro*, the vehicle receiving a message from a potential pedestrian utilizes the round-trip time (RTT) of a wireless signal to verify the movement of the sender. Realizing *Pedro* without specialized hardware or verifiers is a challenging proposition. First, measurements about the sender's location using round-trip time (RTT) tend to be inaccurate from non-specialized hardware. Second, the lack of pre-distributed verifiers makes it hard to obtain measurements from different anchor points. *Pedro* addresses these challenges through a novel concept of tracking the sender's possible region. In particular, the vehicle keeps track of the sender's possible location or *region* by mapping the measured (and often inaccurate) RTT into a disk region. Tracking these regions over a period of time allows the vehicle to conclusively verify whether the sender has moved along the side of the road, which implicitly implies that the sender is a moving pedestrian, not a stationary device. We prove that under a set of very realistic conditions, a stationary attacker cannot mimic the behavior of a moving pedestrian. Our analysis and real-world experiments show that this mechanism is simple, quick, and tolerant to noisy GPS and RTT measurements.

6.2 System and Threat Models

We consider an urban vehicular ad hoc network environment where different road users (vehicles and pedestrians) exchange messages to enhance road safety and efficiency. We adopt the notion of *mobility verification* as a proxy

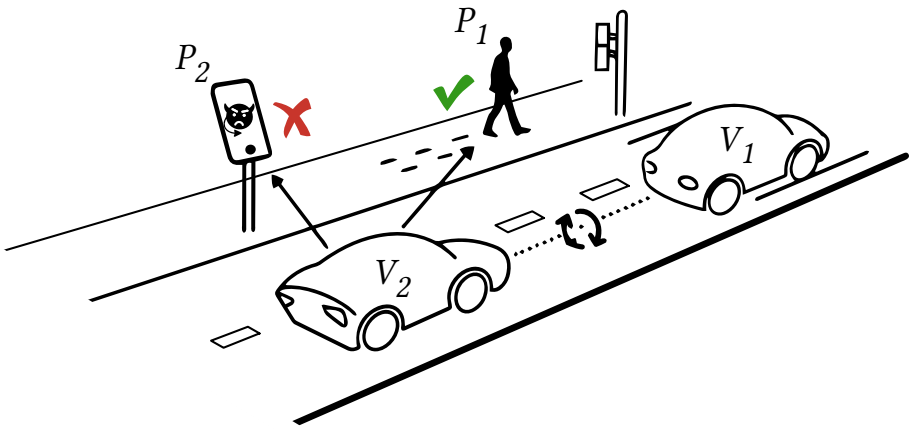


Figure 6.1: Traveling verifiers on the road (V_1 and V_2) attempts to verify the mobility of the provers (P_1 and P_2) and verify only moving pedestrian P_1.

for the authenticity of the sender as a moving pedestrian. We consider a device as belonging to a legitimate pedestrian only when it is moving, and any non-moving device is regarded as a stationary pedestrian or malicious attacker, both of which should not send warning messages. As we will prove in Section 6.4.2.2, a stationary attacker is not able to impersonate a moving pedestrian, and the only way to be able to send an accepted message is to actually move. This constraint significantly raises the bar for broadcasting fake safety transmissions.

Problem Definition: We define the *pedestrian mobility verification* problem as following: a set of moving vehicles, known as *verifiers*, $V = \{V_1, V_2, ...V_j\}$, where j denotes vehicle number, aims to verify the mobility of a *prover* (pedestrian) P. Once its mobility has been verified, the vehicles accept and process safety broadcasts from P, which is shared with all other verifiers.

6.2.1 System Model

Figure 6.1 illustrates the system model of *Pedro*. As one or more verifiers on the road (V_1 and V_2) pass by a prover (P_1 or P_2), multiple RTT measurements are taken over time to estimate the distance between the verifiers and the prover. We assume all vehicles in V to be already authenticated as trusted vehicles

based on existing state-of-the-art V2V authentication protocols [61, 131]. Thus, any information relayed among V is treated as trustworthy.

The hardware requirements for *Pedro* are minimal; all entities, V and P, utilize off-the-shelf mobile devices without further hardware modifications. More specifically, both P and V have the WiFi capability[1], and V is additionally equipped with GPS. Furthermore, each entity has a public and private key pair. An entity uses a private key to sign its messages so that other entities can track its messages and establish trust relations over time using the public key. As such, the entity's public key serves as its pseudonym and can be reset, which requires re-establishing the trust relationship with other entities.

For all wireless communications, we assume verifiers and the prover can measure RTT using line-of-sight measurements. Additionally, *Pedro* is only concerned with verifying the mobility of provers within its own wireless range. Without loss of generality, we consider the 2-D Cartesian coordinate system for simplicity; our method can be easily extended to 3-D cases.

6.2.2 Threat Model

We envision an attack scenario where an adversary deploys a malicious node on the roads or sidewalks (without any location restrictions) to broadcast false messages. An adversarial node has at least the same capabilities as a true pedestrian, except for *mobility*, i.e., the adversarial node is physically immobile. The adversary is fully aware of the verification protocol, and we do not impose any restrictions on their knowledge. It can also accurately measure the location and velocity of V at any given time instant. However, we assume that the different adversarial nodes do not collude.

The adversary can launch passive and active attacks. In the passive attack setting, the adversary is stationary and conforms with the protocol; it does not alter the RTT measurements. In the active attack setting, the adversary is also stationary and can arbitrarily alter RTT measurements aiming to be recognized as a mobile pedestrian. Under both attack scenarios, the adversary

[1]We use WiFi that is most widely available on mobile devices, but *Pedro* is not limited to a specific wireless technology.

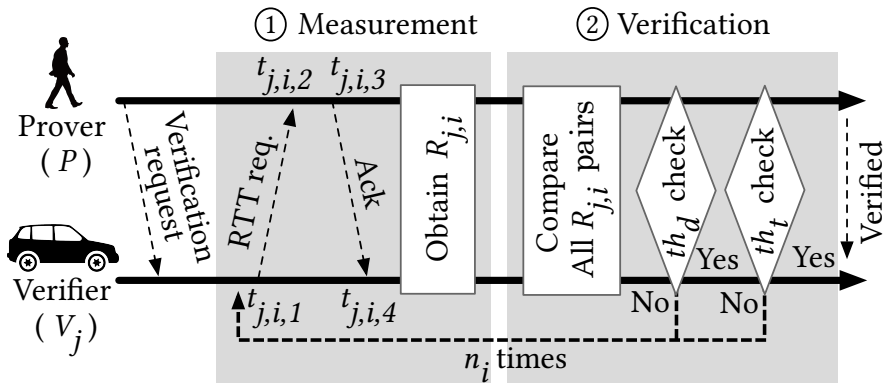


Figure 6.2: *Pedro*'s two stage verification protocol.

is not able to impersonate a trusted prover because it would require accessing their private key used for signing the safety broadcasts.

6.3 *Pedro* Protocol

This section presents the protocol of *Pedro*. As illustrated in Figure 6.2, P initiates the verification request as V enters its wireless range. The protocol consists of a sequence of cycles; each cycle involves the Measurement and Verification stages between a single prover P and a single verifier V_j. The prover, P, repeats the cycle of two stages or verification instance (denoted as i) n_i times until the verification succeeds, or until V_j exits its wireless range. If P fails to be verified by V_j, then it initiates the verification again with the next passing verifier V_{j+1}. The remainder of this section details each stage of *Pedro*'s protocol.

6.3.1 Measurement

In the Measurement stage, V estimates the location region of P by measuring the distance between itself and P by measuring the RTT of a WiFi message. As illustrated in Figure 6.2, V first initiates the RTT measurement by sending the RTT request message to P at time $t_{j,i,1}$. As soon as P receives this message at $t_{j,i,2}$, it acknowledges back to V at time $t_{j,i,3}$, which is then received back by

V at $t_{j,i,4}$. Then, the RTT is defined as the time difference measured by V, i.e., $RTT = t_{j,i,4} - t_{j,i,1}$. Note that this RTT measurement includes the time taken by P to respond to the RTT request, i.e., $t_{j,i,3} - t_{j,i,2}$, which is not proportional to the distance between V and P and thus ideally should be removed by immediately acknowledging the RTT request or measuring the response time and reporting it to V. However, since P has not been verified yet, V cannot rely on the response time reported by P. In addition, since smartphones are not equipped with a real-time operating system, time measurement of P would not be precise either, even if it is a legitimate prover. Our experiments in Section 6.4.1 show that modern smartphones are capable of almost immediately respond to the RTT request unless they intentionally delay their response. Therefore, for practicality reasons, we assume $t_{j,i,3} - t_{j,i,2} \approx 0$.

Based on the measurement, the distance between the two entities, denoted as $d_{j,i}$, can be calculated as $d_{j,i} = RTT/2 \times c$, where c is the speed of light constant. Assuming no RTT and GPS errors, if P immediately responded to the RTT request, we can conclude that P is on the rim of a circle with a radius of $d_{j,i}$ that is centered at $o_{j,i}$, which is the location of V obtained from its GPS readings. On the other hand, if P arbitrarily delays the time-to-respond to send the acknowledgement, i.e., $t_{j,i,3} - t_{j,i,2} > 0$, P can be located anywhere within the circle. Considering both cases, we consider that P can be anywhere inside the region, and we define this circle as P's *constrained region*, or $R_{j,i}$. Note that the location of P at $t_{j,i}$ is bounded by $R_{j,i}$, because P cannot arbitrarily shorten $d_{j,i}$ due to the physical constraint of the wireless signal (P cannot reduce $t_{j,i,3} - t_{j,i,2}$ to below 0). During this measurement stage, when V_j obtains a constrained region at time $t_{j,i}$, the center, radius, and timestamp information is shared with all the nearby verifiers in V for the following Verification stage.

6.3.2 Verification

After completing the Measurement stage, V proceeds to the Verification stage for verifying the mobility of P. Figure 6.3 illustrates the verification process with $n_i = 3$ as V passes P, which is moving within its wireless range. At every time instance, i, V_j obtains $d_{j,i}$ from the measurement stage. It uses this value

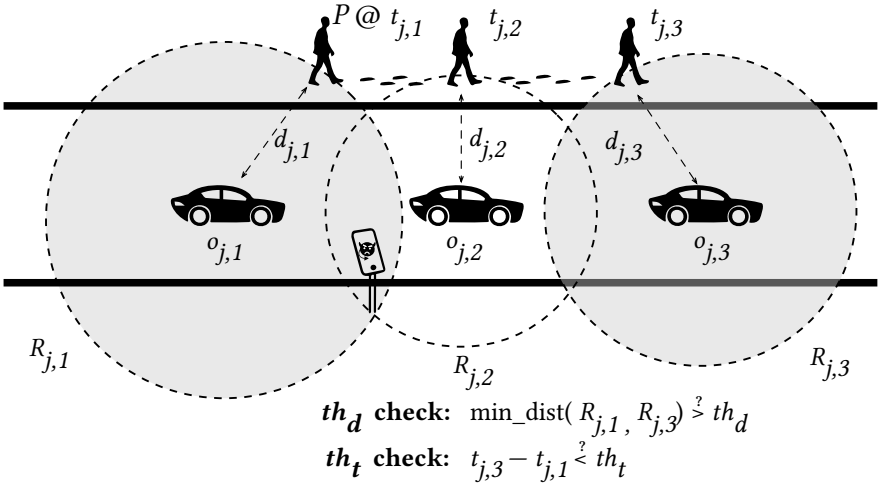


Figure 6.3: Verification stage of *Pedro* with $n_i = 3$. The prover must meet two requirements to be verified: th_d and th_t check.

to construct a constrained region $R_{j,i}$, resulting in a series of $R_{j,i}$ for each P. The core idea underlying the verification protocol is that if two $R_{j,i}$ (not necessarily consecutive) do not overlap, then P must have moved between the two measurement instances.

Recall that an adversary can make RTT arbitrary longer, but it is impossible to make it shorter than the ground-truth value; by manipulating the RTT value, the attacker can make a certain $R_{j,i}$ only larger. If two the regions overlap (e.g., $R_{j,1}$ and $R_{j,2}$), even if their centers are far apart, it is a possible that P is a stationary node located in their overlapping area (i.e., stationary node in Figure 6.3). On the other hand, if the two regions do not overlap, (e.g., $R_{j,1}$ and $R_{j,3}$), it implies that P is a mobile node because the regions represent the maximum perimeter that P can be located at each time point since P cannot intentionally reduce the radius of $R_{j,i}$ as previously mentioned. Therefore, P must have moved to be included in two non-overlapping boundaries at different times. Leveraging this physical constraint sufficiently verifies the mobility of P.

This non-overlap condition, however, is not enough to ensure mobility in the real-world due to possible measurement errors, especially in the verifier's

location; the measured region $R_{j,i}$ might be smaller than the ground truth, which results in false positive verification instances. We address this problem by two introducing two conditions for non-overlap. First, the timestamps of two consecutive $R_{j,i}$ must be less than the time threshold th_t, (i.e., $t_{j,k} - t_{j,i} < th_t$) where $k > i$. This condition prevents stale measurements from being used in the verification. Second, for P to be verified, it must have at least one $R_{j,i}$ pair with its minimum distance greater than the distance threshold, th_d. In other words, between the $R_{j,i}$ and $R_{j,k}$ with $k > i$, the minimum distance (i.e., min_dist($R_{j,i}$,$R_{j,k}$)) must be greater than th_d:

$$\sqrt{(o_{j,i,x} - o_{j,k,x})^2 + (o_{j,i,y} - o_{j,k,y})^2} - d_{j,i} - d_{j,k} > th_d, \tag{6.1}$$

where $o_{j,i,x}$ and $o_{j,i,x}$ represents the x and y coordinates of $o_{j,i}$, respectively. The two conditions are checked every time new $R_{j,i}$ is obtained, and P is verified only if it meets the above requirements. Imposing these allows the verifiers to only verify recently moved pedestrians under the RTT as well as GPS measurement errors.

Generally, *Pedro* can thwart potential attacks from passive and active attackers, described in Section 6.2.2, with a high success rate. In some rare cases, however, decreased $d_{j,1}$ or $d_{j,3}$ due to RTT error or enlarged Euclidean distance due to GPS error in Equation 6.1 may cause the minimum distance to exceed the distance threshold th_d, even when passive attackers report RTT measurements without altering. As for active attacks, altered RTT measurements $d_{j,i}$ cannot be smaller than the actual (true) distance because the active attacker can make RTT longer by delaying its response to verifier's RTT request, but cannot make it shorter, which will require impractically accurate prediction of the arrival of RTT requests. Therefore, for any $R_{j,i}$ pairs, the measured minimum distance after altering is always smaller than that without altering, meaning that active attacks achieve a lower attack success rate than passive attacks. We will evaluate the robustness of *Pedro* against the passive attacks in Section 6.4.2.2.

6.4 Experiment and evaluation

In this section, we evaluate the overall performance of *Pedro* as well as its robustness against the attack scenarios of Section 6.2.2. The experiments answer the following questions:

1. *How reliable is RTT-based ranging using COTS devices for distance estimation?*
   We report the real-world distribution of the RTT errors between a moving pedestrian and a moving vehicle using Pixel Android phones. Our results show a mean error of 0.21 m and a standard deviation of 1.87 m of the RTT-based distance compared to the ground truth distance. This small error demonstrates the reliability of using COTS devices in distance estimation.

2. *What are the optimal thresholds for Pedro that balance usability and security properties?*
   We developed a simulator to investigate the impact of different road factors on the time it takes for moving prover to obtain two minimum constrained regions. We use this result to determine the time threshold, th_t, that maximizes the robustness of our verification protocol. Based on the result, we evaluate the overall security of the verification process and find optimal th_d, by obtaining the *Equal Error Rate* (EER), which represents the intersection between FAR (False Acceptance Rate) and FRR (False Rejection Rate) intersects.

3. *What is the real-world performance of Pedro when employing the optimal thresholds?*
   We assess the usability of *Pedro* using a real-world case study between two moving nodes with two Pixel 2 devices. Our results show that the verification is successful with one verifier within 8 seconds. Further, the results from the case study are consistent with the simulation results for the same conditions. This finding suggests that the simulation platform is representative of the real-world performance of *Pedro*.

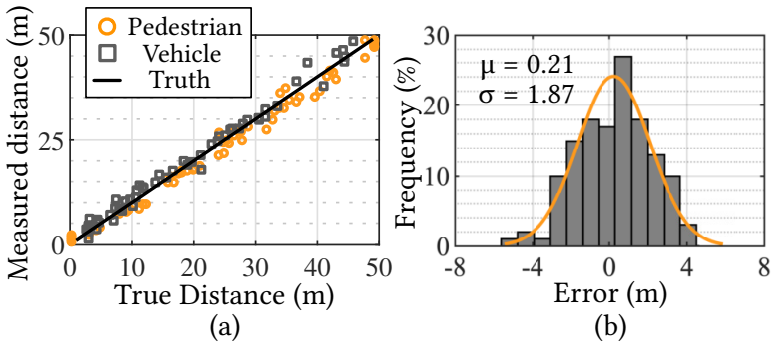


Figure 6.4: (a) RTT based distance measurement of moving pedestrian and vehicle. (b) Error distribution and its Gaussian fitted model with mean (μ) of 0.21 m and standard deviation (σ) of 1.87 m.

6.4.1 RTT Error Model

Because *Pedro* uses COTS mobile devices, RTT errors dominate its verification performance as well as the tightness of threshold values, th_d and th_t. To empirically investigate the RTT errors, we conduct real-world experiments to measure RTT-based distance on moving devices with different relative speeds representing pedestrian and vehicle. We implement the Measurement stage as an Android application on: Google Pixel 2 (Android 9.0 on a 2.35-GHz processor) and Pixel 3 (Android 9.0 on a 2.5-GHz processor). The application leverages the Wi-Fi Aware protocol; the prover acts as an active publisher while the verifier establishes the connection as a passive subscriber. We measure the ground truth distance between the two devices using a BOSCH GLM400CL laser range finder. We perform the measurements with the devices moving at the speed representing a pedestrian (1.5 m/s) and a vehicle (6.7 m/s). Figure 6.4(a) illustrates the measured RTT-based distance with respect to the ground truth distance. The result shows that the error of the RTT-based distance measurement is within 3 m. We observe that the error values remain similar regardless of the distance separating the nodes. This distribution of this error be modeled as a Gaussian fitted model with a mean (μ) of 0.21 m and standard deviation (σ) of 1.87 m, as shown in Figure 6.4(b). We use this error model in the following experiments.

6.4.2 Distance and Time Thresholds

We developed a simulation framework to model the performance of *Pedro* under different conditions. This framework incorporates the RTT error model from the previous section and represents the GPS error as a Gaussian distribution with standard deviation (σ) of 1.2 m [86]. We use this framework to estimate the time required to verify the node mobility, which we use to derive the time threshold th_t. Next, we model an attacker within the platform to derive the distance threshold, th_d, which minimizes the EER.

6.4.2.1 Verification Time

We investigate how different road conditions (i.e., verifier and prover's moving speed, maximum wireless range, etc.) affect the verification stage of a mobile prover. We quantify the performance of the verification through the *inter-region time*, defined as the time for a moving prover to obtain a pair of constrained regions with a minimum distance greater than th_d. We simulate a straight two-lane road while the prover (located within 2–5 m away from the road) is either traveling with its direction along or against the verifier. Additionally, the verification instance, i, is set to 1 second.

As illustrated in four plots in Figure 6.5, generally, as distance threshold th_d increases, the mean inter-region time increases due to the greater distance that the prover has to move to get verified. Also, we observe that the inter-region time decreases when the prover (1) moves faster, (2) is seen more frequently, and (3) is visible from further distances. In these circumstances, the verification becomes easier by taking short time since the prover will be more visible. As the verifier's speed varies from 10 m/s to 30 m/s, the inter-region time decreases as shown in Figure 6.5(a). Specifically, when $th_d = 0$, the mean inter-region time reduces from 6.1 s to 3.3 s because faster verifier speed leads to more rapidly generated constrained regions. However, starting at 30 m/s, the inter-region time starts to increase because the verifier passes by the prover too quickly and is unable to obtain enough number of regions needed for verification. In terms of varying maximum wireless range, a higher range reduces the inter-region time ranging from 25.4 s down to 3.4 s as shown

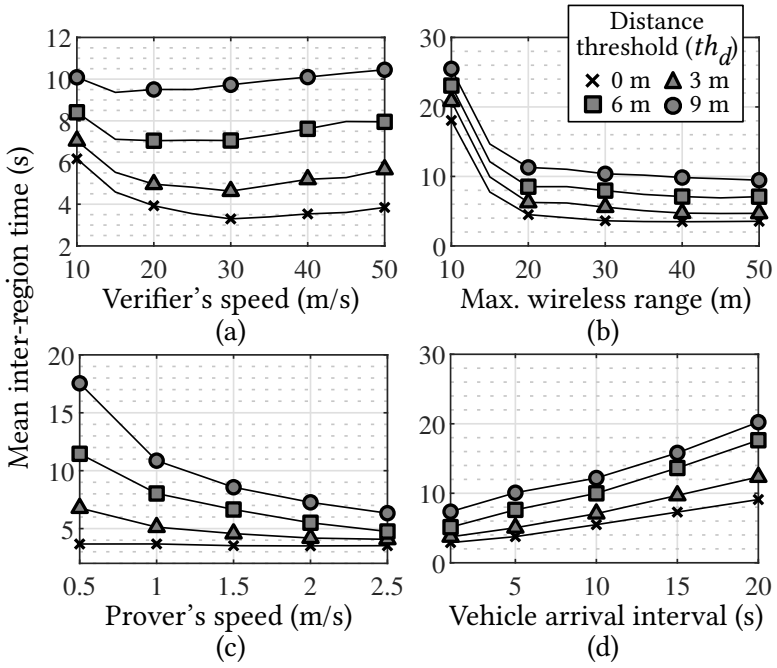


Figure 6.5: Mean inter-region time under varying (a) verifier's speed, (b) maximum wireless range, (c) prover's speed and (d) vehicle arrival interval with different th_d.

in Figure 6.5(b). This is because as the wireless range becomes higher, verifiers can obtain faster as well as greater number of constrained regions compared to lower range. Figure 6.5(c) illustrates the impact of different prover's speed ranging from 0.5 m/s to 2.5 m/s. When $th_d = 0$, varying the prover's moving speed does not significantly affect the inter-region time due to the shorter distance that the prover has to move. However, as th_d increases, the slower prover exhibits greater mean inter-region time due to the long distance it has to travel. In Figure 6.5(d), we illustrate the effect of verifier arrival interval. When interval is set to 1 s, which means verifiers are approaching the prover every 1 s, the prover's inter-region time exhibits less than 10 s. Comparatively, higher arrival interval leads to higher time due to less number of verifiers within same period of time, leading to less number of constrained regions. Specifically, at 20 s interval, the prover's inter-region time exhibits 20.4 s when $th_d = 9$. Nevertheless, under different road factors, the moving verifiers can

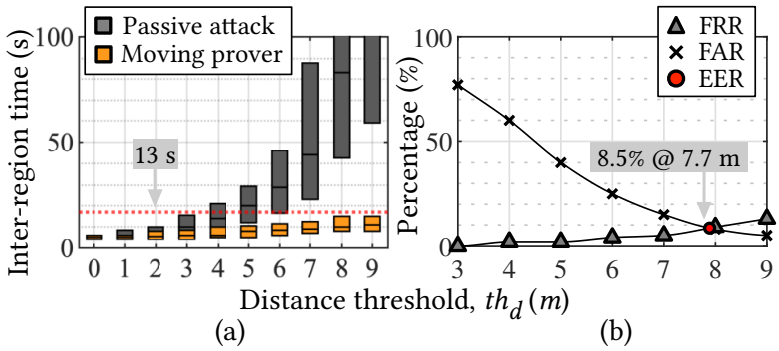


Figure 6.6: (a) Distribution of inter-region time of moving prover and passive attacker. (b) EER of verification with $th_t = 13$.

obtain a pair of regions exceeding distance threshold of 9 m under 26 s.

6.4.2.2 Attack Robustness

We evaluate the robustness of *Pedro* against the passive attack scenario. In the passive attack, the adversary is fixed to a stationary location and attempts to be verified while complying with the protocol (does not add any arbitrary timing delay during the Measurement stage). We first simulate this attack scenario 1000 times under different road factors and obtain its inter-region time. This will allow us to derive the time threshold th_t by comparing it against the inter-region of the moving prover presented in Section 6.4.2.1. Figure 6.6(a) illustrates the distribution of the inter-region times of the passive attacker and mobile prover with respect to varying th_d. When th_d is less than 5 m, the two distributions do not exhibit much difference in their inter-region times due to GPS and RTT errors that allow adversarial prover to quickly obtain pair of constrained region within small th_d. If th_d increases above 6 m, the two distributions exhibit more significant differences because the adversarial prover requires greater number of verifiers as well as constrained regions to leverage the noise/errors in its favor. From this inter-region time, we can choose th_t, which effectively distinguishes between the moving prover against the passive attacker. We experimentally choose 13 s and plot the FAR as well as FRR to obtain EER as shown in Figure 6.6(b). Generally, low ERR represents higher accuracy of distinguishing legitimate pedestrians over

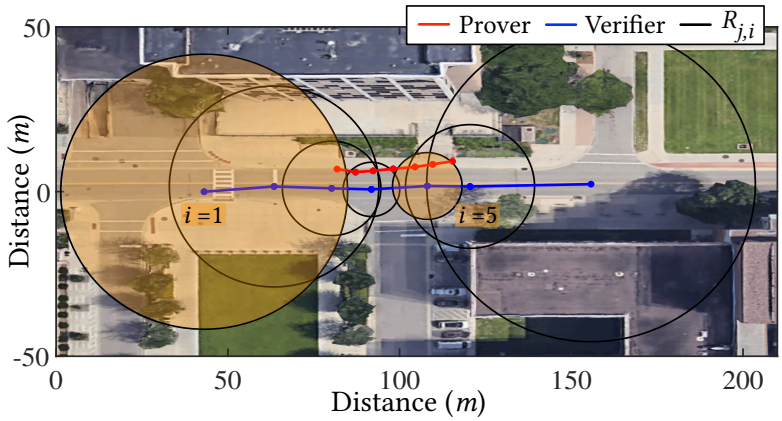


Figure 6.7: Real-world experiment result. Constrained region pairs with $i = 1$ and $i = 5$ meets the two verification requirements and the prover is verified.

adversarial nodes. When $th_t = 13$, the EER of 8.5% is achieved with th_d of 7.7 m.

6.4.3 Real-world Case Study

Using the derived thresholds, we conduct a case study under real road conditions to evaluate the usability and feasibility of *Pedro*. We utilize two Pixel 2 devices where one device is carried by moving pedestrian (prover) and the other is deployed in the vehicle (verifier). The Measurement stage is performed using the same Android application as in Section 6.4.1. The maximum wireless range is set to 50 m and the time between each verification instance, i, is set to 1 s. We conduct five experiments where the pedestrian moves at 1.2 m/s, and the vehicle travels at the speed of 8.4 m/s on average. Figure 6.7 illustrates one attempt of the verification. As the verifier and the prover move along the indicated paths (GPS-obtained) in the same direction, the verifier obtains total of 7 constrained regions ($n_i = 7$). As each region is obtained, the verifiers check the two threshold requirements on all pairs of boundaries. In this case, the verifier verifies the prover by observing a region pair with $i = 1$ and 5; the minimum distance between the two exhibits 13.1 m, which exceeds the th_d of 7.7 m and the differences in their timestamps falls under $th_t = 13$ s. In all five cases, the moving pedestrian was all verified through

single verifier. The average verification time is 7.4 s, which shows that the *Pedro* is able to quickly verify the moving pedestrian even under road conditions where there are not many readily available verifiers. Furthermore, we simulate this scenario (identical road factors) 1000 times and obtained closely matching verification time of 7.9 s. This suggests that our simulation platform represents the real-world performance of *Pedro*. Note that the result does not imply that a single verifier should be around the prover for more than 7 s as the decision can be collectively made by more than one verifier.

6.5 Related work

Several prior studies have proposed various methods to verify the location or motion of the sender (prover) to be used in many safety-critical scenarios. In order to prevent location spoofing, at least one node needs to be constantly moving, while the prover has no knowledge of the moving verifier. Similar to our work, Capkun et al. [23] proposes to leverage RTT distance measurement from at least three verifiers to cooperatively verify location claims. In motion verification domain, verifiers verifies the location, speed and direction of a moving prover or a sequence of claimed locations. For example, Schafer et al. [122] verifies the claimed movements of a mobile prover by measuring wireless signal's frequency shifts in verifier's side caused by the Doppler effect. With at least three verifiers in different locations, they can uniquely identify the position, speed and direction of any prover that cannot be faked. Based on the difference in time of arrival (DToA) from a mobile prover to a static verifier, Schafer [121] verifies a sequence of one mobile node's location claims with at least three static verifiers. Similarly, Lee et al. [82] leveraged the shared ambient electromagnetic radiation among devices located in the same space to enable wireless device verification.

Part III

Enhancing Digital Safety in Video Conferencing Apps

7 ARE YOU REALLY MUTED?: A PRIVACY ANALYSIS OF MUTE BUTTONS IN VIDEO CONFERENCING APPS

Video conferencing apps (VCAs) make it possible for previously private spaces — bedrooms, living rooms, and kitchens — into semi-public extensions of the office. For the most part, users have accepted these apps in their personal space without much thought about the permission models that govern the use of their private data during meetings. While access to a device's video camera is carefully controlled, little has been done to ensure the same level of privacy for accessing the microphone. In this work, we ask the question: *what happens to the microphone data when a user clicks the mute button in a VCA?* We first conduct a user study to analyze users' understanding of the permission model of the mute button. Then, using runtime binary analysis tools, we trace raw audio flow in many popular VCAs as it traverses the app from the audio driver to the network. We find fragmented policies for dealing with microphone data among VCAs — some continuously monitor the microphone input during mute, and others do so periodically. One app transmits statistics of the audio to its telemetry servers while the app is muted. Using network traffic that we intercept en route to the telemetry server, we implement a proof-of-concept background activity classifier and demonstrate the feasibility of inferring the ongoing background activity during a meeting — cooking, cleaning, typing, etc. We achieved 81.9% macro accuracy on identifying six common background activities using intercepted outgoing telemetry packets when a user is muted.

7.1 Introduction

As the de facto alternative for in-person meetings during the COVID-19 pandemic, the demand for online video conferencing for professional and personal use increased significantly. Video Conference Apps (VCAs), such as Zoom, Slack, Teams, and Webex, became available on all modern devices and operating systems. To support their functionality, these VCAs require access to

the device's microphone and camera. Operating systems (OSes) provide the users with permission controls that allow the app to access the microphone and camera. Once granted, the app has access to both hardware resources until the user revokes the permission.

In addition to OS-based controls, VCAs provide their users with two privacy control mechanisms during a call: turning off the camera and muting the microphone. In most OSes, such as Windows and macOS, turning off the camera from the app engages an OS-level control which prevents the app from accessing the camera. A visible hardware indicator (e.g., a light near the camera) informs the user whether an app is accessing their camera. On the other hand, the implementation of the mute button is app-dependent and rarely has a visible hardware indicator. OSes do not expose an easily accessible microphone switch to the apps without going through many steps (e.g., via a control panel).

Apart from smart speakers, which pose tangible privacy threats, the mute button has received little attention in the context of VCAs. Previous research investigates users' privacy attitudes towards VCAs and alludes to the mute button as a privacy control tool available to the users during a virtual meeting [42, 72]. However, the mute button's privacy implications during the interactions between the user and VCAs have not been adequately addressed.

This paper investigates the privacy issues associated with the mute button in VCAs, focusing on whether a mismatch exists between the user's perception of the mute button and its actual behavior. We follow a two-pronged strategy to guide our investigation. First, we design a user study to uncover what the users think the mute button does (i.e., their understanding) and what they believe it should do (i.e., their expectations). Second, we compare the user study findings against an empirical investigation of the actual behavior of the mute button across a range of VCAs and operating systems.

We conducted a user study with 223 participants recruited from Prolific. Our user study revealed that the participants perceive the mute button of VCAs as a privacy control, preventing other meeting participants from overhearing them. We observed a dichotomy in the understanding of the mute button: participants were split about whether a VCA accesses the microphone after

they click the mute button. However, most of them indicated that the VCA should access the microphone only when unmuted.

Based on the findings from the user study, we empirically characterized the conditions in which the VCA *actively* queries the microphone in different operating systems. This task was challenging because OSes only log microphone accesses for each app; they do not provide fine-grained statistics about microphone queries. We addressed this challenge by instrumenting Windows, macOS, Linux, and the Chromium browser to track the fine-grained microphone queries by popular VCAs. We conducted a set of experiments on each VCA-OS combination to monitor the API accesses of each VCA under different conditions. We discovered that *all* of the apps in our study could actively query (i.e., retrieve raw audio) the microphone when the user is muted. Interestingly, in both Windows and macOS, we found that Cisco Webex queries the microphone regardless of the status of the mute button.

We followed our instrumentation efforts with an analysis of Webex, a popular VCA for the enterprise setting. We analyzed how it processes the queried microphone data to determine whether any audio-derived data leaves the device. This analysis also proved challenging as the VCAs, such as Webex, encrypt outgoing traffic. Further, tracking the data flow within apps is not straightforward because they employ proprietary and obfuscated libraries. To facilitate tracking of audio data, we performed a backward search from the encrypted network traffic to locate the inputs to the encryption functions. This search allowed us to decrypt the contents of the network packets sent by Webex to its servers. We discovered that Webex sent periodic packets containing audio-derived telemetry data to its servers, even when the microphone was muted. Although these packets are transmitted at a low rate (once per minute), their audio-derived values correlate with the volume levels of background activities.

To verify our hypothesis, we present a classifier to fingerprint background activities from these telemetry values. Training this classifier was also challenging. Without access to the proprietary algorithm that generates the audio-derived data, it is not feasible to use existing audio datasets to create training data for the classifier. Furthermore, the training data has to represent real-

world situations, including realistic noise types and varying volume levels. We address this challenge by collecting Webex-based telemetry data corresponding to more than 200 hours of background activities. Our evaluation of the classifier with over-the-air data shows that telemetry data from Webex can conclusively fingerprint a set of popular user activities, such as music, chatting, and vacuum cleaning. We demonstrate that even with user data that is compressed and transmitted on a minute-by-minute basis, some activities have unique patterns that are discernable in Webex's telemetry data.

Our key contributions are as follows.

- *User Study*: We conduct a user study with 223 VCA participants to assess their understanding and expectations regarding the mute button (Sec. 7.3).
- *Audio Access Tracing*: We analyze VCAs' fine-grained access to the microphone; we found that most VCAs have access to audio-derived data even when the user is muted (Sec. 7.4).
- *Webex-based Case Study*: We conduct a thorough system-level study of the Webex Windows client. We discover that, in contradiction to its claims in the privacy policy, Webex sends periodic audio-derived data to its servers (Sec. 7.5).
- *Background Activity Detection*: We present a design for a machine learning model the infers background activities from Webex's audio-derived data (Sec. 7.5.3).
- *Mitigation Strategies*: We distill our findings in the form of mitigation strategies that provide users with better control over the mute button (Sec. 7.6).

7.2 Related Work

In the following, we discuss the recent results about the privacy of VCAs. While there is existing research studying possible exfiltration of audio and video

data from mobile apps [111], we focus on the research specific to VCAs. We also include related work about mute buttons in the context of smart speakers. Finally, we discuss the work surrounding background activity recognition, which is relevant for our analysis in Sec. 7.5.3.

Privacy Issues in VCAs The security and privacy of video conferencing platforms has been studied since the early 2010s. In 2013, Kilpi et al. examined privacy and security issues in future (at that time) videoconferencing technologies [72]. They discuss the mute button as a necessary privacy control for the users. More recently, Emami-Naeini performed an online user study to understand the user concerns with VCAs [42]. They found that users are concerned about the security and privacy properties of VCAs. They also found that individuals consider the mute button as a privacy control: they perceive privacy violations from forgetting to press the mute button.

During the pandemic, more people were exposed to privacy and security risks caused by VCAs [118, 98]. In 2019, Zoom fixed a camera leakage vulnerability caused by its casual use of a local web server [105]. Meanwhile, real-time background blurring for VCAs is widely adopted to protect user's privacy in an office or home environment [161, 107]. However, VCAs may leak a user's video privacy in many ways. Kagan et al. [68] demonstrated that collage images of video conference meetings posted on public websites may leak sensitive information such as users' names, ages and genders. Altschaffel et al. [10] showed that traffic patterns of encrypted metadata and multimedia data exchanged during VCA meetings, can be used to identify increased activity in front of camera or even identify users. There are also concerns with the information that VCAs collect about their users. For example, Consumer Reports identified privacy concerns with the data collection practices of popular VCAs, such as Zoom, Google Meet, Microsoft Teams, and Cisco Webex [129]. These concerns centered around the purposes of collecting metadata from the meetings.

In this paper, we follow-up on these previously-reported vulnerabilities and privacy studies. In particular, we study the users' understanding and expectation of the mute button, and whether they match the VCAs' behavior.

We focus on the interaction between the VCA and the user's microphone when the user presses the mute button, as opposed to previous research that studies the mute button in the context of protecting the user's privacy from other meeting participants.

Mute Button in Voice Assistants Researchers have also considered the privacy issues from always-listening smart home devices[80, 7]. Smart home devices continuously process the raw audio to detect a trigger word or phrase. As such, the privacy threats arise from these devices accidentally or maliciously recording the user's background activities [9]. Researchers have first discussed the efficacy of the physical mute button as a privacy control to mitigate these threats. The mute button was found to be inconvenient and suffering from user trust issues [80, 27]. Follow-up works proposed other privacy controls, such as ultrasound jamming [130, 27, 30], cutting the power [27], and employing interpersonal communication cues [96].

Contrary to the smart device case, VCA users widely utilize the mute button to prevent others from listening to their background activities (Sec. 7.3). Users trust that other meeting participants cannot hear them after applying the mute button. However, the behavior of the VCA, after applying the mute button, is less understood. In this paper, we characterize the operation of the mute button from the perspective of the interaction between the user and the VCA.

Activity Fingerprinting Finally, we discuss research about fingerprinting activities from audio-derived data. User activities and contextual information, including walking, driving, and riding, can be inferred from ambient sound. Lu et al. [89] presents an audio event classifier that identify user's current activities utilizing the microphone input of mobile phones. Not only the ambient sound, but encrypted audio traffic can be used to infer user's private information. Previous studies proved that encrypted IoT traffic might leak private information of their environment, including device status and user activities. Traffic analysis of the video streams from home security cameras enables monitoring daily activity patterns [85, 15, 31]. Li et al. [84] further demonstrated

the possibility of detecting fine-grained activities, including dressing, moving, and eating, from encrypted home security camera traffic. Similar to encrypted traffic analysis, Schuster et al.[125] performed an encrypted Video Stream Identification by analyzing bitrate burst and time interval of video streaming traffic. They utilized the segment transmission mechanism of MPEG-DASH and successfully identified Netflix video titles using a trained classifier.

Kennedy et al. and Wang et al. [71, 144] demonstrated that an attacker can infer which voice commands a user says to a smart speaker, by eavesdropping and analyzing outgoing encrypted traffic from smart speakers to a cloud server. Wang et al. [144] further manifested the incoming traffic from the server also leak voice commands information. Moreover, Bae et al. [6] presented a video streaming service identification attack by monitoring video downstreaming traffic through LTE networks with high accuracy.

These research works demonstrate that data derived from audio streams can be used to fingerprint their content and is therefore relevant to our discussion in Sec. 7.5.3 about inferring the background activities while the user is muted.

7.3 User Study

Our first objective is to study the user perceptions of the mute button along with their understanding of its functionality. Towards that end, we conduct an online user study with 230 VCA users. Our study aims to answer two questions about VCA users: (1) *When do they think the VCA accesses their microphone?* and (2) *When do they think the VCA should access their microphone?*. Answering these questions allows us to characterize the user's understanding and expectations of the mute button, respectively. In the following, we describe the design of the user study, the recruitment, and the findings.

Open Ended Questions	Multiple Choice Questions
Q1. Why do you use the mute button? **Q2.** What activities do you perform or take place in your background when you are muted? **Q3.** Please describe what does the app do when you press the mute button.	**Q4.** For your most frequently used video meeting app, when do you think it has access to your microphone? **Q5.** For your most frequently used video meeting app, when should it have access to your microphone?

Multiple Choice Answers for Q4 and Q5
S1. When the app is not running.
S2. You start the app but are not in a meeting.
S3. You're in a meeting but you apply the mute button in the app.
S4. You're in a meeting and you are unmuted.
S5. You leave the meeting while the app is still running.

Table 7.1: The main questions used in the user study. Q1-Q3 are open-ended questions with answers coded by researchers. Q4 and Q5 are multiple choice questions where the participant selects one or more statements from *S1-S5* in response. The full list of questions is available in the Appendix.

7.3.1 Study Design

We designed a Qualtrics survey[1] to help answer our research questions. We used partial disclosure to hide the fact that the study was about the privacy implications of the mute button. The description of the survey and its title focus on capturing the users' general experience with VCAs during the pandemic. The survey has four major sections; the first section collects optional demographic information. The second section collects information about the

[1]The full survey can be found here: `https://osf.io/szd4x/`. Upon completion of our study, we discovered that our Qualtrics form automatically collected IP address and location data by default. Our consent form did not account for this behavior. Nevertheless, upon discovery of this issue, we deleted all sensitive data, including data hosted at Qualtrics and locally. The IP address data and location data were never processed or analyzed in any form during our study. Our final dataset removes any potentially identifiable markers.

preferred VCA and frequency of usage.

The third section asks the respondents about their experience with the mute button. We adapt the questions from Lau et al. [80], which studies the mute button in smart speakers. In particular, we probe the users about their usage of the mute button, their reasons, and their understanding of its functionality using questions in Table 7.1. This section contains three open-ended questions and two multiple-choice questions.

The last section adopts a refined version of Internet Users' Information Privacy Concerns (IUIPC-8) from Groß [58] to measure the participants' privacy concern. This survey section contains the first mention of privacy, after the respondents have answered the questions related to the mute button. Finally, the survey includes two attention checker questions and was exempted by the IRB at our institution.

Participant Recruitment and Demographics We recruited participants from the Prolific data collection platform. We employed Prolific's prescreening criteria to enforce gender balance and to forward the survey to only those who have worked from home during the COVID-19 pandemic with 90% approval rate in previous studies. Before conducting the survey, we conducted a pilot study with 15 users to calibrate the payment and ensure that the study design is clear. Through Prolific, we were able to recruit 299 participants, where we kept 223 responses from participants who passed the attention checkers. The median completion time was 8 minutes, and we paid each participant $1.5; the median hourly rate was $11.

Among our participants, 96.8% are between 18 to 44 years old, 63.2% of them work in sales, service, management and professional industry, and 82.5% achieved at least a college degree. During COVID-19, 54.7% of our participants answered that they have used video conferencing apps more than once a day and 40% of them used once a day or once every few days. The most popular video-conferencing app among the participants is Zoom, and the other popular apps include Microsoft Teams, Google Meet and Cisco Webex.

We map the responses to the IUIPC-8 question to a score based on seven-point Likert scale, representing participant's privacy attitudes. The average

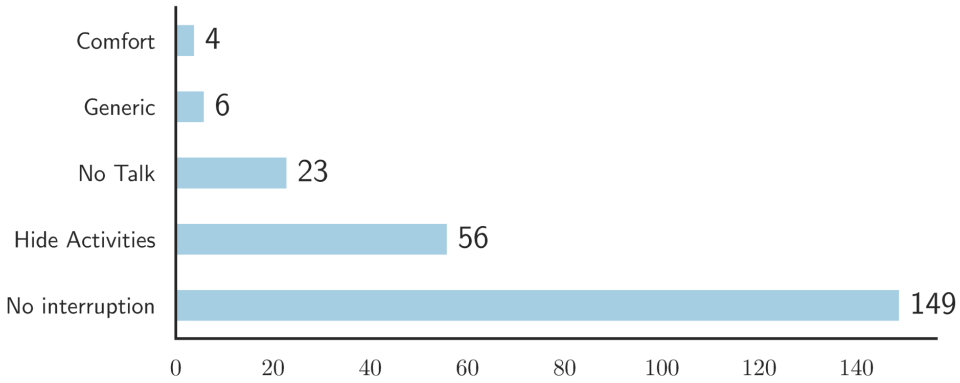


Figure 7.1: The distribution of the codes about reasons users reported for using the mute button as extracted from answers to *Q1*.

scores is 2.02 for all participants, implying that most participants are privacy-conscious in our study. The value of Cronbach Alpha Index is 0.7915 for privacy attitudes responses from 223 participants, which indicates a good internal consistency and reliability of these responses.

7.3.2 Findings

We report the key findings from our user study, through analyzing the participants' responses. We coded the responses to the open-ended questions (*Q1*, *Q2*, and *Q3*) following this procedure. For each question, two authors independently coded the responses, after which they generated a consolidated codebook describing the responses. For *Q1*, we settled on five codes about the reasons for which participants use the mute button. For *Q2*, the codebook consists of twelve codes describing the background activities. The codebook for *Q3* contains nine codes representing the participants' description of the mute button operation. Then, each coder independently coded the first 30 responses for each question; the resulting Cohen's kappa is 0.85 for *Q1*, 0.90 for *Q2*, and 0.82 for *Q3*, indicating strong agreement [94]. The coders split and coded the rest of the responses. See detailed codebooks of the open-ended questions in Appendix 7.7.9.

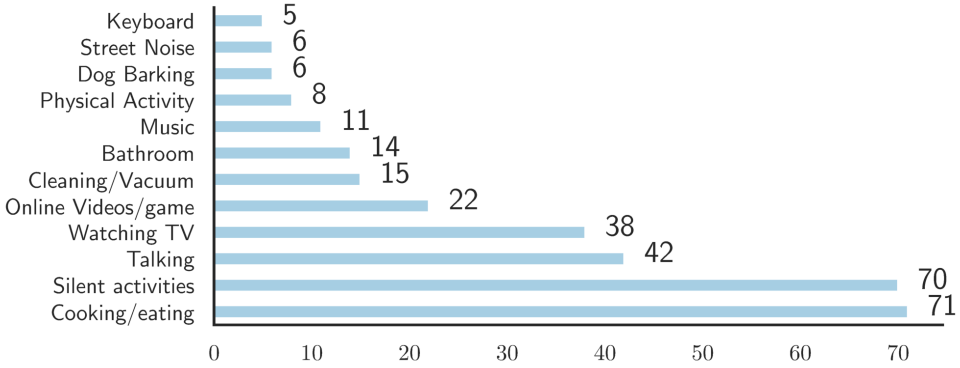


Figure 7.2: The distribution of the codes about the background activities as extracted from answers to *Q2*.

Usage Patterns: We start by analyzing the responses to *Q1*, where 214 participants out of 223 indicated that they have used the mute button before. The responses for *Q1*, as shown in Fig. 7.1, reveal two main reasons why users employ the mute button: (1) hide background activities and (2) avoid interrupting or disturbing others on the call. It is interesting that the participants regard the mute button as a privacy control measure to prevent others from hearing them. For example, *P19* mentioned the reason for using the mute button is: *"So that people won't listen to private activities or conversations."*

The responses for *Q2* indicate an array of background activities the participants perform while muted, as indicated in Fig. 7.2. Participants mentioned more than one activity in their responses; For example, *P166* mentioned: *"Talking, loud video watching, cat activity (meows, occasional falling and crashing of items), cleaning (including vacuuming)."* The most prevalent activity was related to preparing food, cooking, snacking, or eating. Other frequent activities include chatting, watching TV, cleaning, typing, or watching online videos. We elaborate more on these background activities in Sec. 7.5.3.

Understanding of the Mute Button: As indicated earlier, we asked the participants two questions (*Q3* and *Q4*) to gauge their understanding of the mute button. To gain initial insights into the participants' understanding of the mute

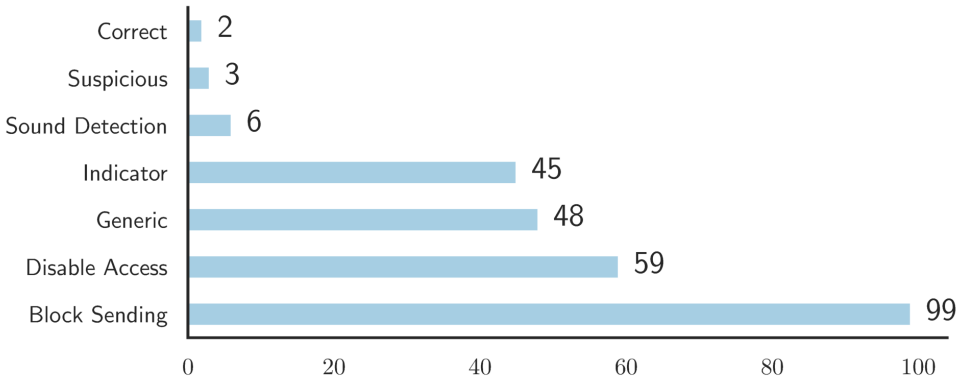


Figure 7.3: The distribution of the codes about the users' understanding of the mute button operation from answers to *Q3*.

button, we study the coded responses to *Q3*, as evident from Fig. 7.3. The most frequent response was that by using the mute button, the app prevents others in the call from hearing the user. For example, *P16* indicated that: *"It doesn't produce my audio on the other participant's platform or computer."* Moreover, other participants focused on the interface change when the mute button is pressed, as in the case of *P119*: *"It shows me the mic with a line crossing it signalling it is not working."* . Meanwhile, 59 participants mention that the mute button disables the microphone. For example, P161 mentions: *"When I press the mute button, my microphone is muted and disabled on the app from picking up any sound waves from where I am."*

For Q4, we provide five situations, *S1-S5*, in our user study as shown in Table 7.1. The responses to Q4 indicate that the participants exhibit a diverse understanding of the operation of the mute button, as shown in Fig. 7.4. Out of the 223 responses, 69 participants selected only *S4* as a response to Q4. These participants think the app only accesses the microphone when they are in the meeting and the mute button is not pressed.

Further, we found that the participants were split in their selection of *S3* as a response to Q4. Nearly half of the participants (111) did not select *S3*, indicating that the app does not access the microphone when the mute button is pressed. The other half indicated that the app accesses the microphone, even

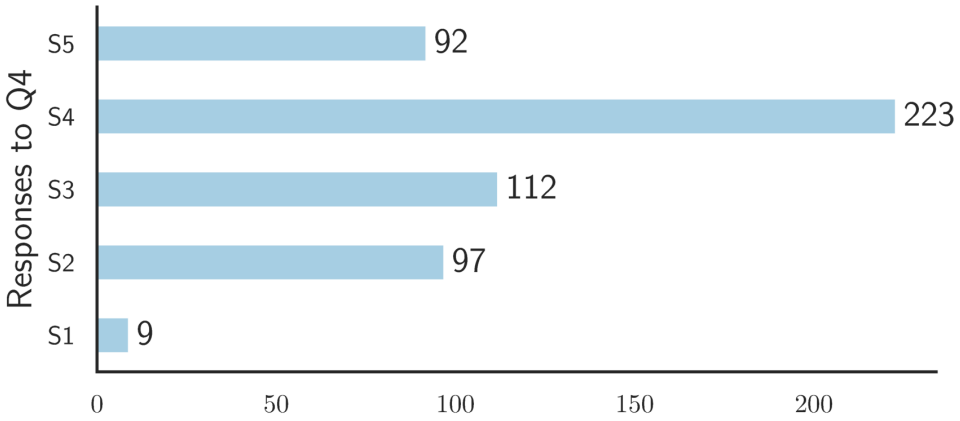


Figure 7.4: The distribution of responses to Q4. The statements *S1-S5* are defined in Table 7.1.

when muted. Interestingly, we observe that 49 participants selected *S2*, *S3*, *S4*, *S5* when responding to Q4, indicating that the app accesses the microphone as long as it is running. Also, we observe that 36 participants selected *S3* and *S4*, indicating that the app accesses the microphone as long as the user is in a meeting. In all the cases above, we found no correlation between the responses and the IUIPC-8 privacy attitude scores.

Expectations of the Mute Button: Finally, we analyze the responses to *Q5*, about when do the participants think the VCAs should access the microphone. The responses reveal that the participants have clear expectations about the operation of the mute button, as indicated in Fig. 7.5. Among the 223 responses, 173 participants selected only *S4* as a response to *Q5*. These participants indicated that the app should only access the microphone when the meeting is running and the user is unmuted. Interestingly, 27 respondents selected both *S3* and *S4* as a response to *Q5*.

In conclusion, the results from the user study suggest that the user's understanding of the mute button does not match their expectations of its behavior. In the rest of this paper, we study the actual behavior of the mute button and analyze whether it matches user understanding and expectations.

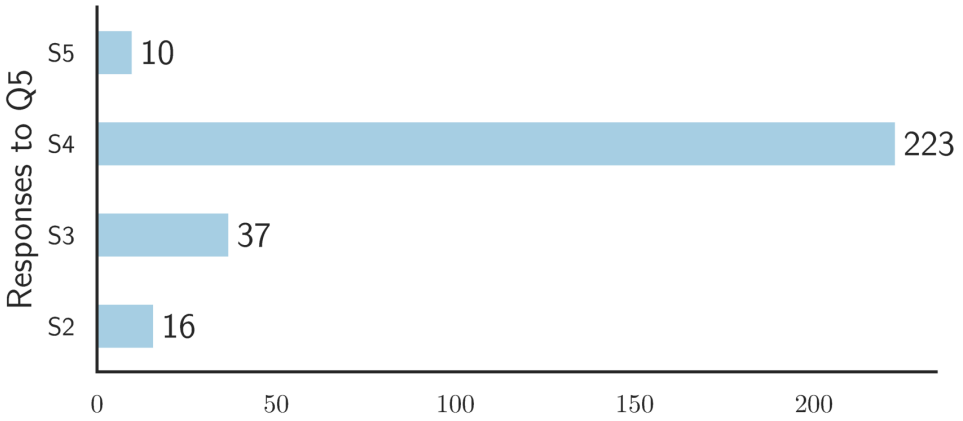


Figure 7.5: The distribution of responses to Q5. The statements *S1-S5* are defined in Table 7.1.

7.4 Analysis of Mute Button

Following the results from our user study, we investigate whether the actual behavior of VCAs matches user expectation by focusing on desktop environments. Our objectives are to determine: (1) if VCAs actively access the microphone when muted and (2) what kind of indicators (if any) they give users that the microphone is being accessed.

7.4.1 Overview of VCAs and Platforms

There are two broad categories of runtime environments in which VCAs execute: native apps that run directly in the operating system and web apps hosted by a web browser. Each has a different permission model for accessing the microphone. Most of the VCAs we study in this work have a native app implementation for the major operating systems (macOS and Windows) and a web app used on unsupported platforms (Linux and others). The VCAs that we studied (listed in Table 7.2) exhibit a consistent look and feel across platforms. Their implementation, however, on each platform is different, due to syscall interfaces and display APIs. Zoom on Windows, for example, is a self-contained Windows-specific software package. Zoom on macOS has a

App	Windows	Linux	macOS
Zoom (Enterprise)	✓	✓	✓
Slack	✓	✓	✓
MS Teams/Skype	✓	✕	✓
Google Meet	∘	∘	∘
Cisco Webex	✓	∘	✓
BlueJeans	✓	∘	✓
WhereBy	∘	∘	✓
GoToMeeting	✓	∘	✓
Jitsi Meet	✓	∘	✓
Discord	∘	✓	∘

Table 7.2: A summary of the VCAs we studied. ✓: native app ∘: web-based app ✕: No implementation.

similar user interface to its Windows counterpart, but the underlying code base appears to be different.

Native apps can collect data from the microphone with few restrictions. Web apps—implemented in JavaScript— request access to the microphone through a web browser, which generally has more restrictive policies for data collection and more tools that allow the user to control the app's access to hardware.

Browser Based Apps Browser-based VCAs rely on their host browser to mediate their interactions with the operating system and the hardware. The browser-based VCAs that we studied are implemented entirely in JavaScript, and they use a special-purpose API called WebRTC [53] for driver interactions—including microphone accesses—that are typically not available to web apps. WebRTC is a native interface written in C++ and C, acting as a driver for the hardware within the browser that can call the operating system to access the microphone. Information transferred by WebRTC is subject to controls and policies of the browser. Web-based VCAs are sandboxed inside the browser and do not circumvent WebRTC.

There are two ways a user can mute a web-based VCA: (1) using a browser-level mute button or (2) using a WebRTC software mute signal from the app.

Both techniques are more trustworthy than app-controlled mute because they are implemented and enforced by the browser, not the app.

The browser-level mute button completely disables microphone access to the VCA, as if the microphone is not active within the system. Web-based VCAs also implement an app-level mute button, which has similar functionality to the browser-level mute: it enables a software mute inside of WebRTC, disabling all audio transfers from the microphone. Users must trust the web-based VCA to use the software mute functionality rather than some internal mute button implementation. We found that all of the studied apps use the WebRTC mute functionality correctly. Furthermore, it is straightforward to verify that web-based VCAs correctly use the software mute functionality through source code audits and the WebRTC debugger built into Chromium.

Native Video Conferencing Apps Native VCAs can directly call the operating system to retrieve audio data from the microphone. Most of them abide by the operating system (OS) rules to access the microphone data, with some exceptions. The OS imposes fewer restrictions on native apps than the browser runtime environment imposes on web apps.

All operating systems utilize a permissions-based access system to retrieve data from the microphone. In most cases, apps must have explicit permission to access hardware resources such as the microphone. Each app follows three steps to configure and use the microphone: (1) user approval, (2) driver initialization, and (3) audio data retrieval. Windows and macOS require the user to explicitly provide permission for each app, which the app retains indefinitely while it runs (unless the user revokes the permission).

Once the user approves the access for the app, the app must create an interface to the audio drivers. Some OSes, like Windows, offer users a visual cue that indicates when the app is using the microphone. But unlike the WebRTC browser runtime, none of the major operating systems we are aware of support enforce a software mute. This lack of an OS-mediated software mute means each native app must implement its own internal mute functionality. Even when a software mute is active, apps can still access the microphone while the user is muted.

7.4.2 Analysis Methodology

To understand what happens when the user presses the mute button on desktop VCA clients, we utilize various OS-based tools to trace audio data as it is transferred from the operating system to the app. Our objective is not just to establish whether the app has permission to access the microphone when muted. Instead, we aim to understand whether the app actually reads microphone data when the user is muted.

Linux Audio data transfer from the Linux kernel to the VCAs is mediated through PulseAudio and ALSA. ALSA is a kernel subsystem that provides a kernel-level interface to the audio hardware, and PulseAudio is a userland process that interfaces with ALSA and provides higher-level features like mixing and multiplexing. All the VCAs we studied interface with the userland PulseAudio process.

To intercept audio data in transit from PulseAudio to a VCA, we use the DynamoRIO runtime code manipulation system [2], which allows us to inject foreign code into a running process. Our additional code, written in C, is called each time a fresh buffer of microphone data arrives from PulseAudio. We write the audio buffer's address in the process's memory space to a log file. We then trace the buffer addresses from the log using IDA Pro. The contents of the buffer are the raw audio bytes from the microphone. DynamoRIO oversees the process's execution by loading and running modified basic blocks one at a time, which substantially slows the app's execution, occasionally causing it to crash.

Windows Although it is possible to track microphone access by monitoring the system registry [63], we were not able to track transfers in real time from the microphone to the VCA. The registry only records times at which an app opens or closes a connection to an audio device. The OS registry—linked to a visual indicator in the system tray—does not distinguish detailed API calls which encode information about whether a VCA is reading audio data or accessing status flags about microphone activity.For fine-grained and detailed

information, we intercept syscalls from the VCA to the operating system.

In Windows 10, syscalls are obfuscated behind a userland API library which acts as an intermediary between the apps and the OS. The Windows API library is similar to the Linux/Unix C library syscall wrappers, except that there is no one-to-one mapping between the parameters that the app passes to the API and the parameters that the API passes to the OS. Instead, the API functions as a higher-level wrapper around system calls, and there is no official documentation available from Microsoft detailing how to call the operating system directly.

Windows implements many special-purpose API functions for actions like accessing the microphone, which in Linux and Unix are all handled as files. We develop a two-step process to trace audio data in transit from the Windows OS to the native VCAs. First, we use a tool called API Monitor [18] to instrument the userland API with hooks to log pointers to the inputs and outputs of several microphone-related API calls. We then use a live binary analysis tool called `x64dbg` [5] to read the contents of the buffers out to a log file. We utilize an anti-anti-debugging library called Scylla-Hide [4], which hides the fact that an app is being debugged to prevent the app from crashing.

Chromium Chromium acts as an intermediate layer between the operating system and the browser based VCAs. To verify whether web-based VCAs access the microphone while muted, we inject our own logging code in the source of Chromium. We instrument the following three browser functions in Chromium, which are responsible for transporting audio from the operating system to the VCA[2]. First, the browser initiates audio-related `read_data` function, which retrieves the raw microphone data from the operating system and stores it in a raw audio buffer. Then it calls `encode` and `send_stream` functions, which transforms the raw audio into an encoded stream and transfers the encoded audio stream to the web-based VCAs.

macOS An audio subsystem manages microphone data created by Apple via `AVFAudio` or the `AVAudioEngine` interfaces [14]. These interfaces have the

[2]Appendix 7.7.2 includes more details about the functions inside Chromium.

same purpose and interact with the audio hardware in userland. VCAs make a system call to `mach_msg_trap` within either an audio interface thread managed by Apple and retrieve raw audio bytes from the microphone. All of the VCAs we studied connect to the microphone using either of these interfaces and make the same system calls when reading bytes from the microphone.

We use a XCode tool called `Instruments` [13], and the standard Unix networking tool `tcpdump` to monitor VCAs' microphone accesses. `Instruments` logs all system calls and their arguments to a user interface in the Apple system log. `tcpdump` records network traffic while any of the VCAs are running. We attach `Instruments` to a live VCA and perform a `tcpdump` on the networking interface to extract and monitor the dataflow from microphone to the VCA. We then observe the results from `Instruments` to correlate behavior patterns with with Windows evaluation. VCAs in macOS behave similarly to their Windows implementations.

7.4.3 Findings

To understand how VCAs consume microphone data, we conducted experiments on each app-OS combo shown in Table 7.2. We installed all VCAs and registered two accounts for each app on each of the four operating systems. The app-OS combinations that are only accessible in a browser are tested in Linux on Chromium. We initiated the meeting app for each meeting experiment and used the techniques explained above to trace microphone data from OS to VCA under two conditions: mute button toggled on and mute button toggled off. Most platforms we studied display a visual indicator to alert the user that an app is accessing the microphone[3]. We found three broad policies that VCAs follow to read data from the microphone while muted:

1. **Continuously sampling audio from the microphone:** apps stream data from the microphone in the same way as they would if they were not muted. Webex is the only VCA that continuously samples the micro-

[3]Some Linux distributions do not provide any visual indication that the microphone is in use.

phone while the user is muted. In this mode, the microphone status indicator from an operating system remains continuously illuminated.

2. **Audio data stream is accessible but not accessed:** apps have permissions to sample the microphone and read data; but instead of reading raw bytes they only check the microphone's status flags: *silent*, *data discontinuity*, and *timestamp error*. We assume that the VCAs, like Zoom, are primarily interested in the *silent* flag to tell if a user is talking while the software mute is active. In this mode, apps do not read a continuous real-time stream of data in the same way as they would while unmuted. Most Windows and macOS native apps[4] can check if a users is talking even while muted but do not continuously sample audio in the same way as they would while unmuted. In this mode, the microphone status indicator in Windows and macOS remains continuously illuminated, reporting that the app has access to the microphone. We found that applications in this state do not show any evidence of raw audio data being accessed through the API.

3. **Software mute:** apps instruct the microphone driver to completely cut off microphone data. All of the web-based apps we studied used the browser's software mute feature. In this mode, the microphone status indicator in the browser goes away when the app is muted, indicating that the app is not accessing the microphone.

The notable exceptions to these trends are the Microsoft VCAs (Teams and Skype) and Cisco Webex. Microsoft VCAs are much more difficult to trace because they do not use the standard Windows userland API. Instead, they directly make calls to the operating system. Since the Windows syscall interface is undocumented, we could not determine how Teams and Skype use microphone data when muted. More interestingly, we observe that Cisco Webex — unlike the rest of the Windows native VCAs — continuously accesses the microphone while muted. Using `x64dbg`, we were able to trace Webex's

[4]Except Skype and Teams, which we cannot observe because they do not use the conventional Windows API.

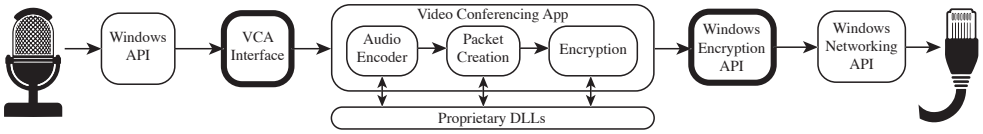


Figure 7.6: Data flow of audio bytes within a Windows 10 VCA. This pipeline is generalizable across the Windows platform. Our system attaches to the bolded modules.

copied audio buffer until that buffer reaches the stack. We discovered that while the app was muted, Webex's audio buffer contains raw audio from the microphone. In the next section, we focus our data flow analysis on Cisco Webex in Windows because of its popularity[5] in the enterprise setting and, more importantly, its unusual behavior.

Recall that our user study reveals two main observations: participants are split whether the VCAs access their microphone while muted, and expect them to access the microphone only when they are unmuted. Our results from this section indicate that the participants are largely unaware of the operation of the VCAs. More importantly, the behavior of these apps violates user expectations. This mismatch between user expectations and app behavior highlights privacy issues with the design of the mute button.

7.5 Webex Case Study

Based on our findings from the previous section, we perform an in-depth analysis of the microphone access pattern in Cisco Webex[6]. We focus on Windows 10 as it is the most widely used operating system at home and in enterprise[7]. As Webex continuously samples the user's microphone (when muted), we need to study whether audio-derived data leaves the local device.

Determining whether audio-derived data from a VCA is leaving on the network port is not a straightforward task because the network flow from VCA

[5]The monthly statistics from Cisco Webex include 600 million participants and 6 billion calls [139].

[6]We used Webex client version 41.12.3.11 through our study.

[7]90% of respondents to our user study used Windows.

to a server is encrypted. Raw dumps of the network traffic from Wireshark are not informative about precisely what the network traffic carries to the VCA's server. And we know that VCAs send and receive network packets that do not contain any audio or video data, so counting network packets cannot give us an indication of whether audio-derived data is leaving a device while the VCA is muted. Instead of directly logging network packets, we need to track how audio data is processed within a VCA.

7.5.1 Methodology for Traffic Interception

Fig. 7.6 depicts the flow of data from microphone to network in native Windows apps. Understanding how a particular VCA handles data from the microphone requires tracking the data as it traverses the chain of processing shown in Fig. 7.6. Most of the data processing in the VCAs we study is handled by proprietary DLLs[8]. Tracking data through function calls from the main VCA process to a DLL is unreliable because runtime binary analysis tools like IDA Pro [3] and `x64dbg` [5] often cause the app to crash when they single-step through function calls to a DLL. And since each VCA uses a different set of external DLLs, we could not establish a single workflow to analyze all VCAs. Existing tools such as TaintDroid [43] are able to establish the data flow within an application in older Android versions. However, in native applications designed for Windows and macOS, flow tracing is difficult and sometimes impossible.

It is easy to see when an app accesses the hardware (networking and microphone) by monitoring Windows API calls (see Sec. 7.4.2), but we are not aware of any tool that can automatically follow data through an entire Windows app. Tracking microphone data after each instruction is not straightforward. Such data initially exists inside of dynamically-allocated memory buffers. Upon each access, this data might move to a new buffer after undergoing a transformation, such as encryption, compression, or encoding. Further, the new buffers may originate from different allocator functions to be stored in the main process's memory image or in an external DLL's memory image. Race

[8]Dynamically Linked Libraries (DLLs) are the Windows implementation of shared libraries.

conditions among the threads in Webex compound the difficulty of tracing: all memory accesses at a specific address of the stack require stoppages, logging, and memory analysis, all of which take time to perform.

However, we do not necessarily need to show a linkage between every successive subroutine that handles microphone data in a VCA to demonstrate that audio-derived data leaves on the network. We can already dynamically trace the audio into the app. We need to show that data from that buffer leaves our machine and is transmitted to a Webex server.

To design such a system, we first map all of the outgoing traffic from Webex. The most efficient way of doing so is to use the Microsoft Network Monitor (MNM). We observe Webex's network traffic using the MNM while the app is muted and unmuted, and we notice a set of packets that are periodically going to a user metrics Cisco server. Now that we have our packets, we need to ensure that the audio buffer within Webex is accessed in the muted state.

Binary tracing on the audio buffer's read/write access using `x64dbg` always ends up in stack space which thwarts our further tracing. However, while following the bytes and logging API calls (from Sec. 7.4), we noticed that Webex calls encoding libraries which access in some time correlation with the audio bytes. We then trace API calls to encryption methods to verify what is happening using the API Monitor. We capture all input arguments and output buffers as a log file from these calls while the user was muted during a Webex meeting. The log contains timestamps, input parameters to the API call, and the resulting output buffer. With the results of the function logged, we compare the encrypted buffer to network traffic leaving the machine and notice a one-to-one match between the encrypted bytes (from Wireshark) and the data sections of network packets from Webex. Consequently, we link the data regions of outgoing user metrics packets to our post-encrypted output buffers. Upon observing the input, we notice that the input arguments in these cases are in plain-text where detailed data is compressed using base64 encoding. Decoding the input arguments revealed the packet content to be a JSON structure[9], which contains audio-derived data and other data elements.

[9] An example of such a structure is here: `https://osf.io/szd4x/`

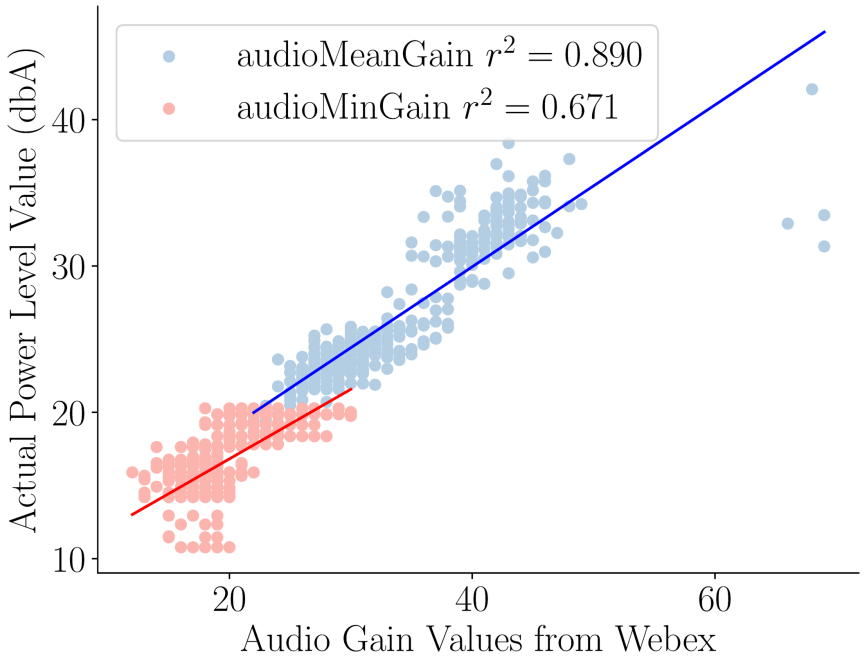


Figure 7.7: Correlation between audio gain reported by Webex and input audio signal power level (in dbA) when noise removal mode is enabled. Although we cannot observe the raw audio while muted, the statistics reported by Webex leak information about the user's background noise.

7.5.2 Findings for Traffic Interception

The data we capture from the API hook is a JSON array with unencrypted and unobfuscated attribute names such as: `audioMaxGain`, `audioMeanGain`, `audioMinGain`, and many others. These JSON arrays are transmitted by Webex once per minute to `https://tsa3.webex.com`, a telemetry server, while the user is muted. The names of these attributes suggest that the JSON array contains audio-derived statistics, most probably connected to the automatic gain control employed by Webex. Our aim is to further analyze the attributes to understand the relationship between the recorded audio levels and these attributes values when the microphone is muted.

Webex has two microphone modes: music mode and noise removal mode

(the default mode). As the name suggests, noise removal mode refers to Webex removing background noise in real-time while the user is speaking. Music mode, on the other hand, transmits audio as the microphone hears it. We perform a small-scale experiment to study whether the audio attributes from Webex network traffic are correlated with the input audio for both microphone modes. We play episodes of the U.S. TV shows "Friends" and "The Office" into a microphone during a Webex meeting while muted. To isolate environmental factors, we feed the audio from the TV shows directly into the Webex meeting through a virtual microphone interface. We repeated each experiment for both microphone modes.

We partition each audio file (corresponding to an episode) into a set of one-minute windows. We then compute the maximum and average magnitudes for each window to report their correlation with `audioMinGain` and `audioMeanGain`. Note that the `audioMinGain` value would correspond to the maximum observed audio level because it requires less gain control. Further, the minimum and mean values depend the most on the input audio. On the other hand, the maximum depends more on the input device and the amount of silent moments, which are random in each episode.

Fig. 7.7 depicts the correlation between the estimated power levels and measured gain values for noise removal mode. As evident from the figure, the measured and estimated values exhibit high correlation; the correlation with the mean gain is higher as it is a more robust metric to window shifts. Note that we do not have access to the source code when computing the gain values, so a perfect correlation is unlikely. This correlation is slightly lower than that of the music mode (Fig. 7.11 in Appendix 7.7.7), implying that the noise removal changes the input audio. Still, the measured `audioMinGain` and `audioMeanGain` are representative of the audio levels.

7.5.3 Classification of Background Activities

We established that Webex accesses the microphone while muted and sends audio statistics to their servers. Further, this data is highly correlated with the energy level received at the microphone, and appears to be indicative of

the activity happening in the background. The logical question that follows is: *is there a potential of learning the user background activities from audio statistics sent to Webex's servers?* In the following, we describe how these statistics can fingerprint the user's background activities, when they are muted.

We analyze the inference of information from user's Webex telemetry traffic while being muted. For each one-minute window, this information contains three values that change relatively : mean, min, and max audio gains. An entity with access to this information, such as Webex's cloud service or any adversary able to view this traffic in transit, can perform this analysis to infer what activity is occurring in the user's environment.

7.5.3.1 Data Collection

We focus on the background activities from our user study of Sec. 7.3. In Fig. 7.2, we highlight twelve activities that happen in the user's background. Out of these activities, we do not consider: (1) silent and physical activities as they do not result in gain changes, (2) bathroom as it is unlikely that the user's microphone will pick up bathroom noises, (3) street noise as it does not represent a private activity, and (4) diverse noise such as TV shows which may contain all of the classes in a single 30-minute instance. As such, our objective is to identify whether the gain values can fingerprint six types of activities: (1) music playing, (2) cooking or eating, (3) people talking, (4) animal sounds (especially dog barking), (5) keyboard typing, and (6) cleaning.

To simulate the real-world environment with specific background activities, we choose multi-hour long ASMR YouTube videos that consist of single background activity. Each video is different such that the videos are produced by different people (YouTube users) doing the same task. The purpose of selecting the videos in such a way is to minimize the effects from the recording environment. We play each video over the air through a Webex meeting, while muting the microphone, and log the extracted gain values.

Our data collection consists of two Windows 10 machines. The first machine plays the videos using its speaker and hosts the meeting for the other machine. The other machine runs a Webex meeting client (without any other

app running). One machine is equipped with a Logitech QuickCam Pro 9000 while the other uses a Logitech C920S Pro HD 1080p webcam for microphone input. Both machines then join the same meeting room and collect data simultaneously; on both, we mute the microphone, turn off the camera, and keep the default microphone settings.

We place the machines in a $12ft \times 7ft \times 10ft$ room. We adjust the distance from the speaker to the two microphones and generate multiple datasets based on the varying distances. Webex only allows for meetings to last for 24 hours. For each Webex meeting, we can extract around 1440 data points, stamped with the corresponding label. Each data point corresponds to three features: `audioMaxGain`, `audioMeanGain`, and `audioMinGain`, representing three user metrics values from one minute of audio. In summary, we performed data collection over the course of two months, yielding over 180 hours of data points.

We visualize the distribution of the six background activities in Fig. 7.8. This figure shows that it is feasible to fingerprint background activities by analyzing the extracted gain values from Webex. Each activity exhibits relatively consistent and distinguishable gain values, despite sampling diverse videos to represent each activity.

7.5.3.2 Classifier Training

We design a classifier to highlight how the background activities can be fingerprinted based on the observed gain values. In what follows, we describe how we curate the data for this classifier, how we design and train the classifier, and the results of the classification.

Data Preprocessing We split the YouTube videos into a development set (training and validation) and an evaluation set. The two sets have no overlaps in the videos. We set the distance from the microphone to speaker as 10 cm, 25 cm, and 50 cm for both sets of videos, whereas we added an extra distance condition, 100 cm, for the evaluation set.

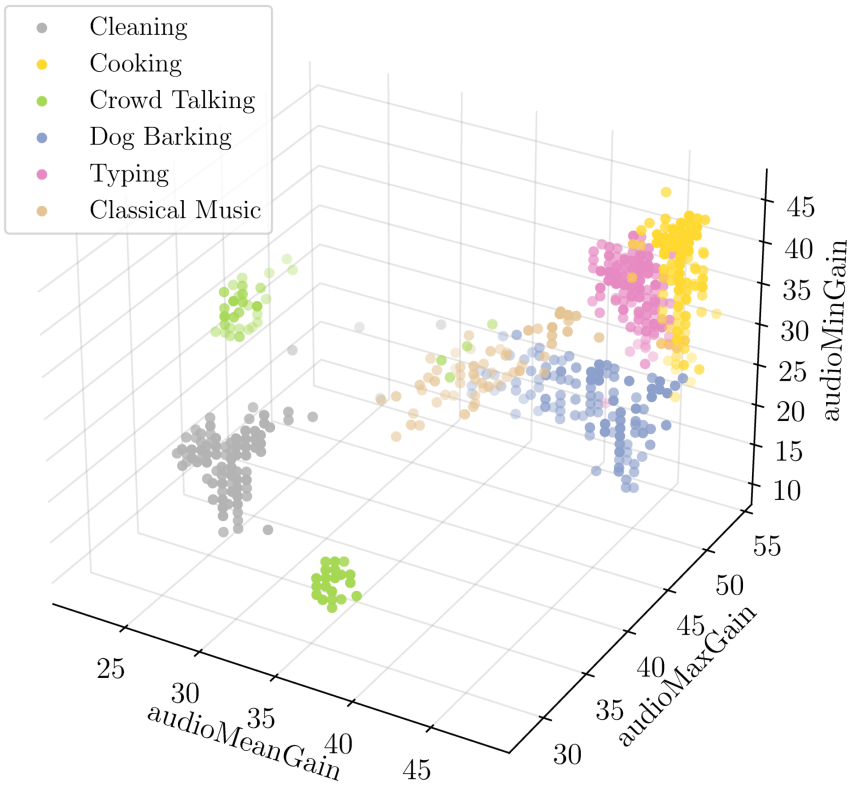


Figure 7.8: Clusters of audio statistics data color coded by background activity type. Clusters are visually separable.

Table 7.3 shows the data distribution for the development and evaluation sets. We split the development set into a training set (80%) and validation set (20%) for hyper-parameter tuning. We split the evaluation set into two subsets to study the effect of distance. The first evaluation subset is collected at distances of 10 cm, 25 cm, and 50 cm between the speaker and microphone. The second subset is collected at a distance of 100 cm; the data in the second evaluation subset has no overlap with the development set in terms of distance and source videos.

For a *t*-minutes long YouTube Video, we extract *t* data points; each data point is assigned the same label derived from the title of the video. To accommodate videos of varying lengths, we limit the input to the classifier to clips

Class	Train	Val	Eval1	Eval2
classical music	168	43	379	184
cooking/eating	500	126	486	169
crowd talking	656	164	1191	568
dog barking	408	103	726	691
keyboard	359	90	1324	580
vaccume/cleaning	544	136	668	572
total (minutes)	2637	660	4774	2764

Table 7.3: Dataset distribution, development set (training and validation) and evaluation set (subset 1 and 2).

of length n. Thus, we apply a sliding window with length n to each window and set the moving stride to be 1. We define each clip as:

$$\textsc{Clip} = \begin{bmatrix} \max_i & \max_{i+1} & ... & \max_{i+n-1} \\ \text{mean}_i & \text{mean}_{i+1} & ... & \text{mean}_{i+n-1} \\ \min_i & \min_{i+1} & ... & \min_{i+n-1} \end{bmatrix}, \tag{7.1}$$

where $\max_i$ represents the `audioMaxGain` for the i^{th} minute in the window.

Model Design and Training We train a supervised multi-class classifier to distinguish background activities given clip data of length n. Similar to Schuster et al. [125], we use a Convolutional Neural Network (Fig. 7.10 in Appendix); the network consists of two 1-dimensional convolution layers, flatten layer, three dense layers, and a softmax layer (of size 6). The design of the convolutional layer takes into account feature and temporal correlations.

We train the network using an Adam optimizer with a cross-entropy function as the loss calculation function. We set the learning rate to 0.001 and initialize model parameter weights in a random uniform distribution. As the total length of the training set and validation is around 3000, we evaluate different batch sizes: 50, 500, 1000, 1500, and 3000. We utilize early stopping to prevent over-fitting. Because the dataset is imbalanced, we calculate the precision separately for each class and compute the average precision score weighted by their proportion in the validation dataset. Then we use the

weighted average of precision score and accuracy of all classes as the early stopping criterion. We select the best-performing epoch index and batch size to train the optimal classifier for each window length.

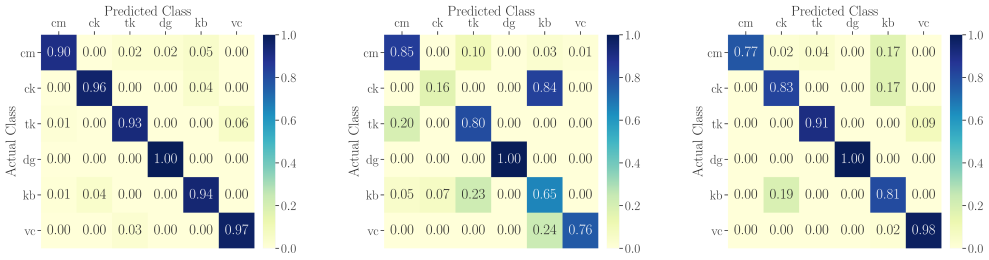
We train the network on windows of size n: $3, 5, 7, 10$. Comparing the performance of 4 optimal classifiers for each window length, we observe that $n = 7$ (96.13% precision on validation set) outperforms windows $n = 3$ (92.26%) and $n = 5$ (92.98%), in terms of the accuracy score and precision score of the validation set. We achieve 96.90% with windows length $n = 10$ but we remove it in case of over-fitting.

7.5.3.3 Classification Results

We present the per-class performance of classifier with window lengths 3 and 7 in Fig. 7.12 and Fig. 7.9. For window size of $n = 7$, we achieve 77.75% macro accuracy on evaluation set 1 and 89.03% macro accuracy on evaluation set 2. The average of per class precision for evaluation set 1 is 73.07% while that is 87.47% for evaluation set 2. Note that evaluation set 2 is collected with 100 cm microphone to speaker distance; our results suggest that the volume level and video content do not considerably hurt the classifier performance.

For window size of $n = 3$, we achieve 78.70% macro accuracy on evaluation set 1 and 78.48% macro accuracy on evaluation set 2. The average of per class precision for evaluation set 1 is 79.35% while that is 84.35% for evaluation set 2. Both classifiers follow our early stopping criteria and achieve high performance on evaluation sets. This performance indicates that, even with three-minutes worth of measurements, it is possible to infer the ongoing background activities.

For both window sizes, dog barking, crowd talking, and cleaning show high precision as well as accuracy on both evaluation datasets. Some music and people talking samples are misclassified as keyboard typing on evaluation set 1 and 2 respectively, while cooking and eating shows a lowest performance among the six background activities. On both evaluation sets 1 and 2, "cooking or eating" data points are severely mingled with "keyboard typing" as both classifiers cannot accurately classify these two classes at the same time. We



(a) Validation Set (b) Evalset1 (c) Evalset2

Figure 7.9: Background activity classifier performance with window length = 7. The six classes include Classical Music (cm), Cooking or eating (ck), Talking (tk), Dog Barking (dg), Keyboard (kb), Vacuum or Cleaning (vc).

discuss this aspect in Sec. 7.6.1.

Finally, we test whether Webex's noise-canceling feature affects the statistics reported in log packets. The results are nearly identical with noise-canceling disabled or enabled. However, there is a difference between the logged gain values from Webex when alternating between the music and noise-removal modes. Therefore, we only collect and present results based on data collected with noise-canceling enabled — the default setting — through our entire classification process.

Our classifier performs well on both evaluation sets in under various kinds of background noise, recording environments and volume levels. The gain values logged by Webex and sent to its cloud server can be used to distinguish multiple types of background activity.

7.6 Discussion

In the following, we discuss some of the limitations with our methodology. We also discuss possible mitigation strategies, including an improved OS-level permission model and user education.

7.6.1 Limitations of the Study

Using live binary analysis tools, we developed a technique to trace incoming audio data from the microphone driver to the operating system's socket API; our methods are in compliance with each app's Terms of Service (ToS). We conducted a thorough evaluation of the Webex native Windows app, demonstrating that we could distinguish a variety of background activities that were most commonly reported in our user study. We discuss limitations in (1) our binary analysis techniques, (2) our dataset and (3) our background activity classifier.

The first limitation is that our binary analysis technique does not easily generalize to other apps because different VCAs have different mechanisms for preparing and encrypting network traffic. Many of the apps we studied encrypt the outgoing data stream before passing it to the operating system's socket interface, making it impossible to search the binary's memory image for the raw microphone data. Only in Webex were we able to intercept plaintext immediately before it is passed to the Windows network socket API.

The second limitation is that the findings from the user study might not generalize to the general population. The user study participants are young and educated professionals, who are potentially more tech-savvy than the general population. However, the responses to our questions did not reveal a high level of technical sophistication when describing the operation of the mute button. Fig. 7.3 shows that handful of participants were able to correctly describe the operation of the mute button.

The third limitation is that we collected data for our Webex case study in only one room. We do not consider the impacts of the speaker's volume level or the room's acoustic properties that may affect the microphone input. It may be possible to infer a relationship between the room's acoustic properties and the audio statistics that Webex reports using raw audio data acquired while the app is unmuted.

Finally, our classifier targets single background activity at a time, and it does not perform well on all background activities. Differentiating between multiple sources is potentially possible, however, due to a limited data collection scheme

we did not evaluate multiple simultaneous events. Furthermore, the "cooking" background activity shows a low accuracy score and overlaps with "keyboard" data points in Fig. 7.8. Poor performance of the cooking class appears to be caused by inconsistent noises that are generated by different cooking activities like grilling, frying, baking, etc. Another reason for the poor performance is that cooking and typing sound similar at different distances. Also, our data does not account for noises that are short in duration. Sounds need to last at least a single minute to create a data point; our techniques cannot evaluate unique but short noises.

7.6.2 An OS-Level Mitigation

To ensure a trustworthy permission model for microphone access in VCAs, we suggest that operating systems adopt a "software mute" feature similar to the one implemented in Chromium and WebRTC. Under that model, the VCA calls an API function or syscall in the OS to disable audio traffic flowing from the microphone driver to the app, putting the OS in charge of the microphone data while the app is muted.

The OS's microphone status indicator would serve as an easy and nontechnical mechanism for users to audit VCAs, ensuring that they use the software mute correctly. The microphone status indicator should be on only when the VCA is unmuted and off otherwise. In our analysis of mute button, we found that the operating system cannot detect the state of an app-controlled mute button, and consequently the microphone status indicator does not correctly reflect whether the VCA is actively reading data from the microphone driver. Since the mute functionality is currently implemented in the VCA instead of the OS, there is no clear policy about how microphone data should be handled during mute that applies to every VCA. As we discovered, some apps read from the microphone at a lower data rate during mute, but Webex reads from the mic the same way regardless of mute button status.

An OS-mediated software mute establishes clear rules about when the VCA should be reading from the microphone, making it clear to the OS when the microphone status indicator should be illuminated and making it clear to

the user when the VCA is reading from the microphone.

7.6.3 VCA Privacy Policies

Few participants in our user study were aware of the data collection or sharing policies of popular VCAs. Around 70% of our participants believe that the mute button blocks the transmission of microphone data or disables the microphone altogether. VCA service providers should provide detailed definitions of data collection scenarios rather than generic statements about how they collect data about their users. All VCAs actively query the microphone when the user is muted, and they might have legitimate purposes. For example, Zoom alerts the user when they try to speak with their microphone muted. The privacy policies of these services need to be explicit about microphone access, which is not currently the case.

We analyzed the privacy policies of the VCAs from Table 2 to understand how do they describe their privacy practices. Other than Google [57], no privacy policy makes an explicit mention to the mute button and how microphone data is accessed when the user is muted. The mention of the mute button in Google Meet's privacy policy refers to the meeting organizer's ability to mute others. Also, the privacy policies are vague about the data collected when the user is running a VCA. Some privacy policies, such as Whereby's and Google Meet's, explicitly mention that they do not collect audio data. Other VCA privacy policies do not mention collecting audio data at all. Most policies describe their data collection, in general terms, as "depend[ing] on the context of your interactions" [97]. The common reasons that VCA service providers cite for collecting data are to improve "app performance" [97, 25], to facilitate "research" [97, 147], and for "user analytics" [97, 147, 25].

Interestingly, Cisco's privacy policy [25] mentions audio data in the context of "types of personal information that [Cisco] may process." Cisco's privacy policy is not specific about when the collection is happening and about the purposes of this collection. However, a different privacy datasheet [95] from Cisco mentions:

> Cisco Webex Meetings ***does not***:

Monitor or interfere with your meeting traffic or content.

Our findings suggest that, contrary to the statement in the privacy policy, Webex monitors, collects, processes, and shares with its servers audio-derived data, while the user is muted. To inform Cisco of our investigation results, we opened a responsible disclosure with Cisco about our findings. As of February 2022, their Webex engineering team and Privacy team are actively working on solving this issue.

7.7 Appendix

7.7.1 Model Architecture

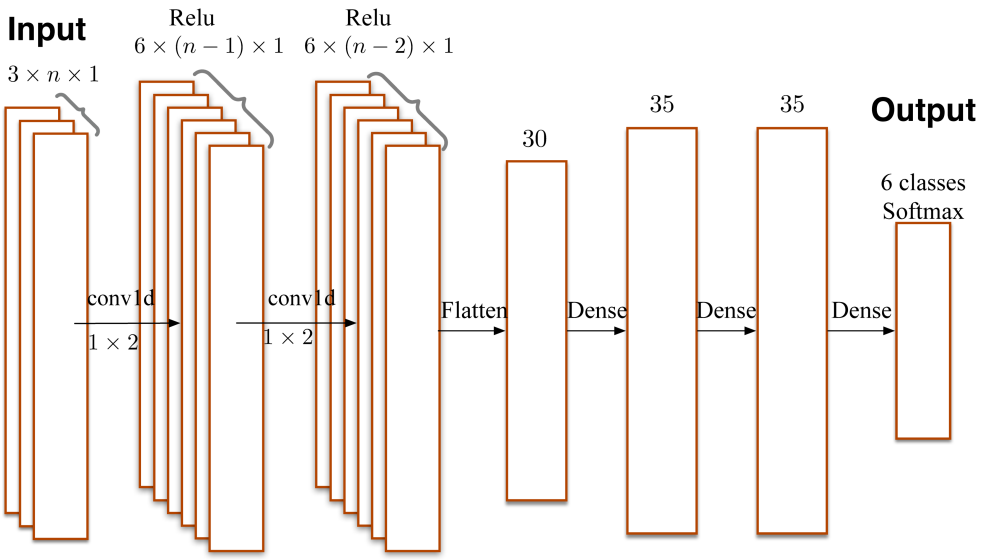


Figure 7.10: The classification architecture for the background activities.

During training, we observe that batch size affects training speed and performance. Our classifier is trained with batch size of 500, epoch of 400, window length of 7 and learning rate of 0.001 .

7.7.2 Chromium API

Chromium acted as a layer between the operating system and the browser based VCAs. To verify microphone access we injected our own logging scripts in the source code of Chromium. Knowing when an app accesses the microphone requires several functions to be monitored, the main functions we observed were:

1. `PulseAudioInputStream::ReadData()` `ReadData` indiscriminately reads audio frames from the operating system into a local buffer, regardless of the VCA's mute status (muted or unmuted).

2. `opus_encode_native()` After receiving a full microphone audio frame from the operating system, `PulseAudioInputStream::ReadData()` passes that frame to `opus_encode_native()`, regardless of the VCA's mute status.

3. `AudioSendStream::AudioSendStream()` – transfers the encoded audio stream to the web-based VCA. It is also a WebRTC API call that executing code can call. `AudioSendStream` only hands the encoded audio data to the VCA if WebRTC's software mute function is disabled.

These three functions outline a general flow of audio data within Chromium in Linux(as of writing this). Logging important variable's states within these three functions painted an accurate picture of microphone usage while the user was muted.

7.7.3 YouTube Video List

Development Set (Training set and validation set) is based on YouTube Video List I. Evaluation set is based on YouTube Video List II.

7.7.4 YouTube Video List I

- Dogs Barking for 12 hours - High Quality Sounds:
  `https://www.youtube.com/watch?v=3Go2_VXy1Tg`

- ASMR One Hour of Soothing Grill Sounds – Sizzling Meat:
  https://www.youtube.com/watch?v=NKoJDyKo1Q
- Vacuum Cleaner Sound - Extended 10 Hours | White Noise Sounds - Sleep, Study or Soothe a Baby :
  https://www.youtube.com/watch?v=Ms8oZeywjyM&t=7s
- People Talking Sound effect (10 Hours) :
  https://www.youtube.com/watch?v=y32-rwUr0Nk
- Baroque Music Collection - Vivaldi, Bach, Corelli, Telemann... :
  https://www.youtube.com/watch?v=ApSoNBu2wt8
- 10 Hours Typing | Cherry MX Blue Mechanical Keyboard | Gaming Keyboard ASMR :
  https://www.youtube.com/watch?v=h8nmVF0IDCs

7.7.5 YouTube Video List II

- ASMR Cooking No talking 5 hours deep relaxation sleeping AD free No ads:
  https://www.youtube.com/watch?v=DoRSCsrKbq8
- 1 HOUR Barbecue Sound | Soothing Grill Sounds | Sounds :
  https://www.youtube.com/watch?v=va6AOQy8sWM
- Vacuum Cleaner Sound and Video 3 Hours - Relax, Focus, Sleep, ASMR :
  https://www.youtube.com/watch?v=KilQtE5Nl9O
- Vacuum Cleaner Sound & Video 2020 Christmas Edition 3 Hours :
  https://www.youtube.com/watch?v=BFNUEVR_Ps8
- BESTE Baby Einschlafmusik Staubsauger Vacuum Cleaner Sound // 3 Hours // P :
  https://www.youtube.com/watch?v=csHiTtxDmx0

- 1 Hour of Dog Barking :
  `https://www.youtube.com/watch?v=7ej1ur8amCo`
- DOG BARKING 12 Hours Sound Effect :
  `https://www.youtube.com/watch?v=fecqn9fnG0s`
- ASMR Typing | Ducky One 2 Mini | Cherry MX Blue (1 HOUR) :
  `https://www.youtube.com/watch?v=vlgch5z4y7Y`
- 10 Hours of People talking :
  `https://www.youtube.com/watch?v=PHBJNN-M_Mo`
- Anne Pro 1 Hour Keyboard Typing Sounds ASMR (No talking, No music, No mid-roll ads) :
  `https://www.youtube.com/watch?v=qMtIOlS_WAo`

7.7.6 Windows API

We can trace the data using the following three methods, which are part of the Windows API DLLs:

1. `BCryptEncrypt` in the `ncryptsslp.dll` library for encrypting network traffic before sending.
2. `IAudioRenderClient::GetBuffer` method in the Windows 10 32-bit Audio interface which fills a local buffer with raw audio data.
3. `IAudioRenderClient::ReleaseBuffer` method in the Windows 10 32-bit Audio interface which releases the buffer space acquired in the get-buffer method call.

The BCryptEncrypt function is the method that some VCAs executes right before they send a packet over the network. After this method is executed, Wireshark captures the post-encrypted packet generated from the BCryptEncrypt function as it leaves the machine. Thus, being able to capture calls at the method before sending the packets grants us unencrypted network traffic. The GetBuffer method fills a local array in the app's memory space with raw audio

data. Using the argument's address, we can follow each call and verify if the audio buffer that an app has is changing even while the user is muted. The ReleaseBuffer method tells us how many frames the app filled their own local buffer with, which gives us a good length of what the app is seeing. Examining the data we extracted from these methods we can build a dataset that, with confidence, observes audio data from the microphone to the network.

7.7.7 Music Mode Correlation Results

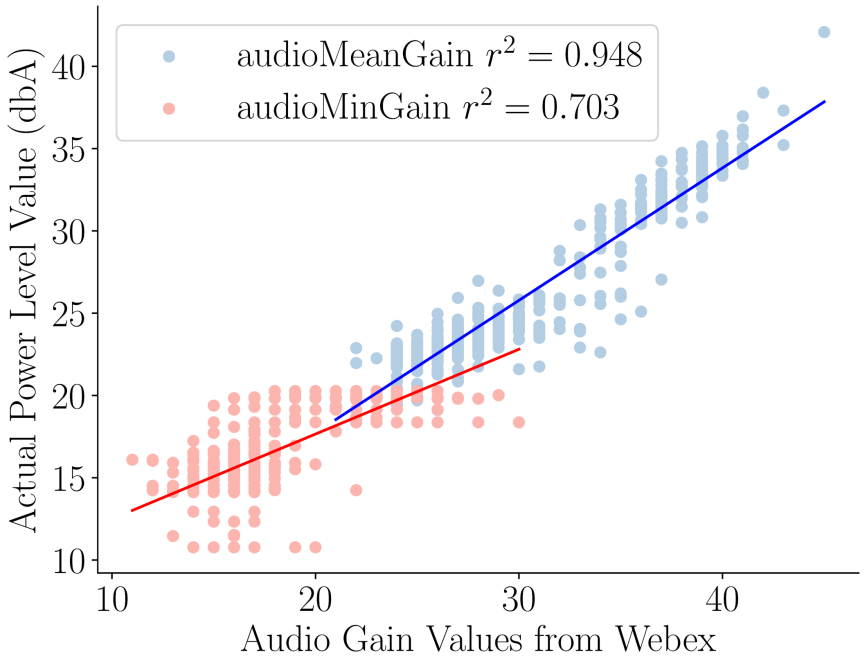
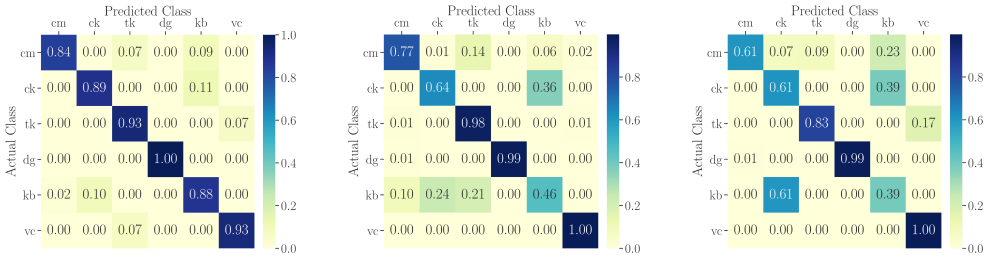
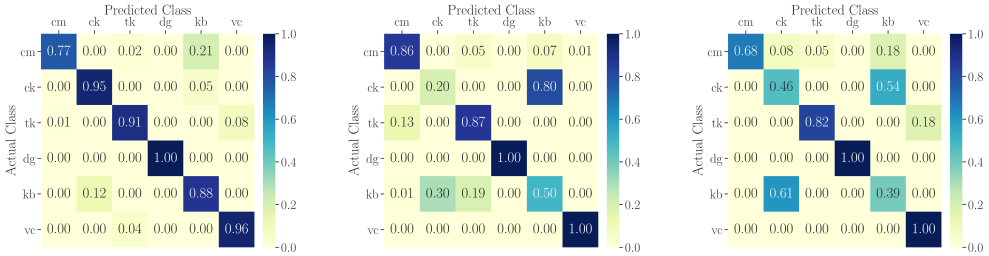


Figure 7.11: Correlation between audio gain reported by Webex and input audio signal power level (in dbA) when music mode is enabled. Although we cannot observe the raw audio while muted, the statistics reported by Webex leak information about the a user's background noise.



(a) Validation Set (b) Evalset1 (c) Evalset2

Figure 7.12: Background activity classifier performance with window length = 3. The six classes include Classical Music (cm), Cooking or eating (ck), Talking (tk), Dog Barking (dg), Keyboard (kb), Vacuum or Cleaning (vc).



(a) Validation Set (b) Evalset1 (c) Evalset2win5

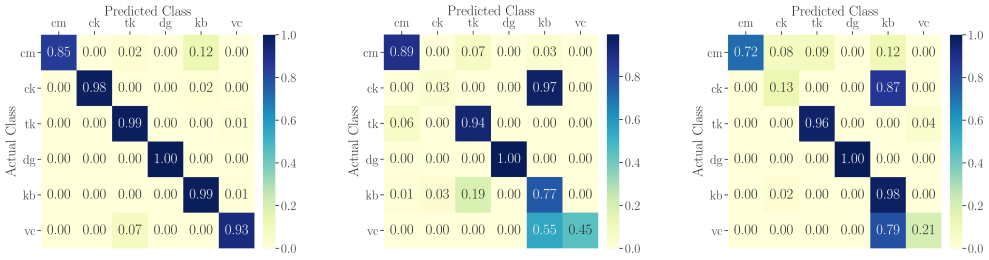
Figure 7.13: Background activity classifier performance with window length = 5. The six classes include Classical Music (cm), Cooking or eating (ck), Talking (tk), Dog Barking (dg), Keyboard (kb), Vacuum or Cleaning (vc).

CodeBook for Q3	Description
Generic	A generic description of the mute button
Indicator	Visual cue/icon notifying the user of the muting event
Block sending	User experience: block transmission of audio data to the other clients
Correct	The respondent understands the correct operation of mute button
Disable Access	The respondent mentions microphone is disabled or cut when mute button is clicked
Suspicious	The respondent suspects the app keeps recording their voice after they apply the mute button
Sound detection	The respondent mentioned the app notify them of possible speaking when muted.

Table 7.4: Codebook for responses to Survey Question Q3

7.7.8 Window Length 5 and 10

We present the confusion matrix of window length 5 and 10 in Fig. 7.14 and Fig. 7.13.



(a) Validation Set (b) Evalset1 (c) Evalset2win10

Figure 7.14: Background activity classifier performance with window length = 10. The six classes include Classical Music (cm), Cooking or eating (ck), Talking (tk), Dog Barking (dg), Keyboard (kb), Vacuum or Cleaning (vc).

CodeBook for Q1	Explanation
No Talk	No need to talk, so muted; or out of concern, in online classes no need to talk,
No interruption	Do not bother others, do not interrupt others from noise
Hide Activities	Hide private activities in the background; hide conversations in background
Generic	no reason in particular
Comfort	The participant just feels more comfortable

Table 7.5: Codebook for responses to Survey Question Q1

7.7.9 Codebooks

We present our consolidated codebooks to three open-ended questions (*Q1*, *Q2*, and *Q3*) that are independently generated by two authors in Tables 7.5, 7.6, and 7.4.

Consolidated codebook - Q2 Activities	
Music	Talking
Dog Barking	Street Noise
Watching TV	Physical Activity
Keyboard	Bathroom
Cooking/eating	Silent activities
Cleaning/Vacuum	Online Videos/game
Talking	Cleaning/Vacuum

Table 7.6: Codebook for responses to Survey Question Q2

Part IV

Future work and Conclusions

8 FUTURE WORK

This dissertation has laid the foundation for enhancing user safety through the integration of commodity devices and advanced predictive systems. Looking ahead, two key projects are proposed to expand upon this work, addressing critical gaps in pedestrian safety research and improving the generalizability of predictive safety models.

8.1 Privacy-Preserving Multi-Sensor Data Collection for Pedestrian Safety

Current pedestrian safety research lacks visual and eye-tracking data, essential for understanding road user interactions and focus. To address this, we propose a multi-sensor data collection initiative in naturalistic environments, designed to characterize pedestrian behavior comprehensively. Building on our prior success in collecting 770 hours of walking data from 85 participants, this project will develop and deploy a privacy-preserving platform integrating wearable devices. The platform will collect mobility, physiological, and visual data from diverse communities while safeguarding user privacy.

Key advancements will include automated extraction of fine-grained events, such as pedestrian heading, road crossing, and attention dynamics, using synchronized multi-sensor data. These insights will bridge the current data gap and foster innovative approaches to studying pedestrian mobility. To maximize its impact, we will make de-identified datasets publicly available, empowering stakeholders to enhance pedestrian safety further through data-driven solutions.

8.2 Generalizing Predictive Models for Diverse Pedestrian Safety Applications

To ensure broader applicability of pedestrian safety systems, this project seeks to generalize predictive models for pedestrian safety, ensuring robust perfor-

mance across diverse pedestrian behaviors, device types, and environmental conditions. The scope of this work includes expanding data collection to encompass a broader range of demographic groups and scenarios, enabling the development of a model that adapts to the complexities of real-world environments. The project will refine predictive algorithms to account for varying pedestrian crossing patterns and behaviors, ensuring adaptability to diverse settings. The system's generalizability will be evaluated through real-world trials and driving simulator studies, providing insights into driver responses and system performance.

In addition to algorithmic and data advancements, this project will develop a cross-platform mobile application to deliver real-time pedestrian safety alerts. The app will be designed with an intuitive and privacy-conscious user interface, facilitating seamless adoption by the public. Further efforts will be directed toward scaling the safety framework, with strategies to encourage widespread use across different user groups and environments. To ensure transparency and promote collaboration, the outcomes of this project, including de-identified datasets, predictive algorithms, and a comprehensive final report, will be made publicly available.

9 CONCLUSION

This dissertation explores the use of commodity devices to enhance user safety across three dimensions: physical safety, bridging physical and digital environments, and digital safety. By addressing critical challenges in these areas, the work contributes novel methodologies, systems, and insights to improve safety and security in personal environments.

In the realm of physical safety, the dissertation introduces PedHat, a smartphone-based system that predicts pedestrian road-crossing behavior to enhance the safety of vulnerable road users. By enabling early detection of crossing intentions, PedHat provides timely alerts to drivers, reducing collision risks. Furthermore, this research evaluates the impact of multimodal alerts on driver awareness, revealing their effectiveness in improving responsiveness but also identifying potential drawbacks, such as stress from redundant warnings. These findings offer a nuanced understanding of driver-pedestrian interaction and inform the design of safer alert systems.

To bridge the gap between physical and digital environments, the dissertation develops a physical mobility verification protocol for secure pedestrian-vehicle communications. This protocol authenticates pedestrian mobility through physical movements, ensuring that only legitimate users can participate in digital communications. The research establishes practical distance and time thresholds for deploying the protocol effectively in real-world scenarios, providing a secure and efficient mechanism to enhance digital safety for road users.

In the domain of digital safety, the dissertation investigates user privacy in video conferencing applications (VCAs), uncovering critical vulnerabilities in the handling of audio data. Through a detailed study of mute button behavior, the work reveals widespread user unawareness of microphone activity during mute and highlights instances of unauthorized audio data collection by certain VCAs. These findings prompted positive actions to halt such practices and contributed to raising awareness of digital privacy risks. Recommendations derived from this study offer actionable guidance for improving user safety

and transparency in digital communication platforms.

Together, these contributions demonstrate how commodity devices can be leveraged to address diverse safety challenges, fostering innovation and practical solutions to protect users in both physical and digital environments. By bridging theoretical insights with real-world applications, this dissertation advances the understanding of user safety and lays the groundwork for future research in this critical area.

[1] Wi-Fi Alliance Wi-Fi aware. `https://www.wi-fi.org/discover-wi-fi/wi-fi-aware`. Accessed: 2019-10-30.

[2] DynamoRIO. `https://dynamorio.org`, Nov 2021.

[3] IDA pro. `https://hex-rays.com/ida-pro/`, Nov 2021.

[4] Scyllahide. `https://github.com/x64dbg/ScyllaHide`, Nov 2021.

[5] `x64dbg`. `https://x64dbg.com/`, Nov 2021.

[6] Watching the watchers: Practical video identification attack in LTE networks. In *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA, Aug. 2022. USENIX Association.

[7] N. Abdi, K. M. Ramokapane, and J. M. Such. More than smart speakers: security and privacy perceptions of smart home personal assistants. In *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*, pages 451–466, 2019.

[8] K. M. Abughalieh and S. G. Alawneh. Predicting pedestrian intention to cross the road. *IEEE Access*, 8:72558–72569, 2020.

[9] S. Ahmed, I. Shumailov, N. Papernot, and K. Fawaz. Towards more robust keyword spotting for voice assistants. In *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA, Aug. 2022. USENIX Association.

[10] R. Altschaffel, J. Hielscher, S. Kiltz, and J. Dittmann. Meta and media data stream forensics in the encrypted domain of video conferences. In *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, pages 23–33, 2021.

[11] H. Alyamani, Y. Yang, D. Noyce, M. Chitturi, and K. Fawaz. Evaluating the impact of warning modalities and false alarms in pedestrian crossing alert system. `https://arxiv.org/abs/2410.06388`, 2024.

[12] M. T. H. Anik, J. L. Danger, S. Guilley, and N. Karimi. Detecting failures and attacks via digital sensors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(7):1345–1354, 2021.

[13] Apple. Measuring performance. `https://developer.apple.com/library/archive/documentation/ToolsLanguages/Conceptual/Xcode_Overview/MeasuringPerformance.html`, Oct 2016.

[14] Apple. Avaudiorecorder. `https://developer.apple.com/documentation/avfaudio/avaudiorecorder`, Apr 2021.

[15] N. Apthorpe, D. Reisman, and N. Feamster. A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic. *arXiv preprint arXiv:1705.06805*, 2017.

[16] F. Arena et al. V2X communications applied to safety of pedestrians and vehicles. *Journal of Sensor and Actuator Networks*, 9(1), 2020.

[17] R. Baker and I. Martinovic. Secure location verification with a mobile receiver. In *ACM Workshop on Cyber-Physical Systems Security and Privacy* (*CPS-SPC*), pages 35–46, 2016.

[18] R. Batra. Api monitor. `http://www.rohitab.com/apimonitor`, Mar 2013.

[19] F. Bella and M. Silvestri. Vehicle–pedestrian interactions into and outside of crosswalks: Effects of driver assistance systems. *Transport*, 36(2):146–159, 2021.

[20] K. Bian et al. Security in use cases of vehicle-to-everything communications. In *IEEE Vehicular Technology Conference* (*VTC-Fall*), pages 1–5, 2017.

[21] F. Bu, S. Li, D. Goedicke, M. Colley, G. Sharma, and W. Ju. Portobello: Extending driving simulation from the lab to the road. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2024.

[22] S. Capkun et al. Secure location verification with hidden and mobile base stations. *IEEE Transactions on Mobile Computing*, 7(4):470–483, 2008.

[23] S. Capkun and J.-P. Hubaux. Secure positioning of wireless devices with application to sensor networks. In *IEEE Computer and Communications Societies* (*INFOCOM*), volume 3, pages 1917–1928, 2005.

[24] G. A. Center. Sumo: Simulation of urban mobility. *Open Source Traffic Simulation*. Available: `https://www.eclipse.org/sumo/`. [Accessed: 30-Jul-2024].

[25] T. C. T. Center. Cisco online privacy statement. `https://www.cisco.com/c/en/us/about/legal/privacy-full.html`, May 2021. Accessed: 2021-11-28.

[26] R. Chandra, J. Padhye, L. Ravindranath, and A. Wolman. Beacon-stuffing: Wi-fi without associations. In *Eighth IEEE Workshop on Mobile Computing Systems and Applications*, pages 53–57. IEEE, 2007.

[27] V. Chandrasekaran, S. Banerjee, B. Mutlu, and K. Fawaz. PowerCut and obfuscator: An exploration of the design space for Privacy-Preserving interventions for smart speakers. In *Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021)*, pages 535–552. USENIX Association, Aug. 2021.

[28] C. Chen, X. Lu, A. Markham, and N. Trigoni. Ionet: Learning to cure the curse of drift in inertial odometry. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[29] C. Chen, P. Zhao, C. X. Lu, W. Wang, A. Markham, and N. Trigoni. Deep-learning-based pedestrian inertial navigation: Methods, data set, and on-device inference. *IEEE Internet of Things Journal*, 7(5):4431–4441, 2020.

[30] Y. Chen, H. Li, S.-Y. Teng, S. Nagels, Z. Li, P. Lopes, B. Y. Zhao, and H. Zheng. Wearable microphone jamming. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–12, New York, NY, USA, 2020. Association for Computing Machinery.

[31] Y. Cheng, X. Ji, X. Zhou, and W. Xu. Homespy: Inferring user presence via encrypted traffic of home surveillance camera. In *ICPADS*, pages 779–782, 2017.

[32] S. T. Chrysler, O. Ahmad, and C. W. Schwarz. Creating pedestrian crash scenarios in a driving simulator environment. *Traffic Injury Prevention*, 16:597–602, 2015.

[33] O. contributors. Openstreetmap is the free wiki world map., 2023.

[34] J. Cui et al. A review on safety failures, security attacks, and available countermeasures for autonomous vehicles. *Ad Hoc Networks*, 90:101823, 2019.

[35] Z.-A. Deng, G. Wang, Y. Hu, and D. Wu. Heading estimation for indoor pedestrian navigation using a smartphone in the pocket. *Sensors*, 15(9):21518–21536, 2015.

[36] A. Developers. Wi-fi aware overview: Discover and connect devices on android. `https://developer.android.com/develop/connectivity/wifi/wifi-aware`, 2023. [Online; accessed 15-November-2023].

[37] P. Di Campli San Vito, S. Brewster, F. Pollick, S. Thompson, L. Skrypchuk, and A. Mouzakitis. Purring wheel: Thermal and vibrotactile notifications on the steering wheel. In *ICMI 2020 - Proceedings of the 2020 International Conference on Multimodal Interaction*, 2020.

[38] M. Di Felice, L. Bedogni, and L. Bononi. The emergency direct mobile app: Safety message dissemination over a multi-group network of smartphones using wi-fi direct. In *Proceedings of the 14th ACM International Symposium on Mobility Management and Wireless Access*, pages 99–106. ACM, 2016.

[39] M. Dong, E. E. Etu, L. Jiang, and G. Huang. Exploring the impacts of mind wandering on driver takeover in automated vehicles: A comparative study of multimodal displays. In *ACM International Conference Proceeding Series*, 2023.

[40] M. Edel and E. Köppe. An advanced method for pedestrian dead reckoning using blstm-rnns. In *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–6. IEEE, 2015.

[41] B. M. ElHalawany et al. Physical-layer security and privacy for vehicle-to-everything. *IEEE Communications Magazine*, 57(10):84–90, 2019.

[42] P. Emami-Naeini, T. Francisco, T. Kohno, and F. Roesner. Understanding privacy attitudes and concerns towards remote communications during the covid-19 pandemic. In *Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021)*, pages 695–714. USENIX Association, Aug. 2021.

[43] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 32(2):1–29, 2014.

[44] E. C. Eze et al. Vehicular ad hoc networks (VANETs): Current state, challenges, potentials and way forward. In *International Conference on Automation and Computing (ICAC)*, pages 176–181, 2014.

[45] H. Fan, R. Wang, L. Yang, and Q. Meng. A survey on collaborative collision warning of connected vehicles at signal-free intersections. *7th IEEE International Conference on Transportation Information and Safety, ICTIS 2023*, 2023.

[46] Q. Fan, H. Zhang, P. Pan, X. Zhuang, J. Jia, P. Zhang, Z. Zhao, G. Zhu, and Y. Tang. Improved pedestrian dead reckoning based on a robust adaptive kalman filter for indoor inertial location system. *Sensors*, 19(2):294, 2019.

[47] S. Fang, S. Munir, and S. Nirjon. Fusing wifi and camera for fast motion tracking and person identification: Demo abstract. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, pages 617–618, 2020.

[48] Z. Fang and A. M. López. Is the pedestrian going to cross? answering by 2d pose estimation. In *2018 IEEE intelligent vehicles symposium (IV)*, pages 1271–1276. IEEE, 2018.

[49] Z. Fang and A. M. López. Intention recognition of pedestrians and cyclists by 2d pose estimation. *IEEE Transactions on Intelligent Transportation Systems*, 21(11):4773–4783, 2019.

[50] E. Fasolo, R. Furiato, and A. Zanella. Smart broadcast algorithm for inter-vehicular communication. In *Proceedings of the Wireless Personal Multimedia Communications Symposium (WPMC)*, 2005.

[51] E. Fasolo, A. Zanella, and M. Zorzi. An effective broadcast scheme for alert message propagation in vehicular ad hoc networks. In *2006 IEEE international conference on communications*, volume 9, pages 3960–3965. IEEE, 2006.

[52] T. Feigl, S. Kram, P. Woller, R. H. Siddiqui, M. Philippsen, and C. Mutschler. A bidirectional lstm for estimating dynamic human velocities from a single imu. In *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–8. IEEE, 2019.

[53] T. M. Foundation. WebRTC API. `https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API`, Oct 2021.

[54] C. Glatz, H. H. Bülthoff, and L. L. Chuang. Warning signals with rising profiles increase arousal. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 59, pages 1399–1403, 2015.

[55] D. Goedicke, A. W. Bremers, S. Lee, F. Bu, H. Yasuda, and W. Ju. Xr-oom: Mixed reality driving simulation with real cars for research and design. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2022.

[56] J. Gong, X. Zhang, Y. Huang, J. Ren, and Y. Zhang. Robust inertial motion tracking through deep sensor fusion across smart earbuds and smartphone. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(2):1–26, 2021.

[57] Google. Google meet security & privacy for users. `https://support.google.com/meet/answer/9852160?hl=en`, Nov 2021.

[58] T. Groß. Validity and reliability of the scale internet users' information privacy concerns (iuipc). *Proceedings on Privacy Enhancing Technologies*, 2021.

[59] F. Gu, J. Niu, L. Jiang, X. Liu, and G. P. Hancke. Safepath: Exploiting ubiquitous smartphones to avoid vehicle-pedestrian collision. *IEEE Internet of Things Journal*, 9(9):7027–7041, 2022.

[60] A. Habibovic and J. Davidsson. Requirements of a system to reduce car-to-vulnerable road user crashes in urban intersections. *Accident Analysis and Prevention*, 43(4):1570–1580, 2011.

[61] J. Han et al. Convoy: Physical context verification for vehicle platoon admission. In *International Workshop on Mobile Computing Systems and Applications (HotMobile)*, pages 73–78, 2017.

[62] M. Hasan et al. Securing vehicle-to-everything (v2x) communication platforms. *IEEE Transactions on Intelligent Vehicles*, 5(4):693–713, 2020.

[63] C. Hoffman. How to see which apps are using your microphone on windows 10, Jun 2019.

[64] M. Hou, K. Mahadevan, S. Somanath, E. Sharlin, and L. Oehlberg. Autonomous vehicle-cyclist interaction: Peril and promise. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2020.

[65] C. Jeong, E. Kim, M. H. Hwang, and H. R. Cha. Enhancing drivers' perception of traffic lights through haptic alerts on the brake pedal. In *Proceedings of the 2023 IEEE 6th International Conference on Knowledge Innovation and Invention, ICKII 2023*, 2023.

[66] M. Jeong, B. C. Ko, and J.-Y. Nam. Early detection of sudden pedestrian crossing for safe driving during summer nights. *IEEE transactions on circuits and systems for video technology*, 27(6):1368–1380, 2016.

[67] W. Jiang, H. Xue, C. Miao, S. Wang, S. Lin, C. Tian, S. Murali, H. Hu, Z. Sun, and L. Su. Towards 3d human pose construction using wifi. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pages 1–14, 2020.

[68] D. Kagan, G. F. Alpert, and M. Fire. Zooming into video conferencing privacy and security threats. *arXiv preprint arXiv:2007.01059*, 2020.

[69] R. N. Kandalan and K. Namuduri. Techniques for constructing indoor navigation systems for the visually impaired: A review. *IEEE Transactions on Human-Machine Systems*, 50(6):492–506, 2020.

[70] C. G. Keller and D. M. Gavrila. Will the pedestrian cross? a study on pedestrian path prediction. *IEEE Transactions on Intelligent Transportation Systems*, 15(2):494–506, 2013.

[71] S. Kennedy, H. Li, C. Wang, H. Liu, B. Wang, and W. Sun. I can hear your alexa: Voice command fingerprinting on smart home speakers. In *2019 IEEE Conference on Communications and Network Security (CNS)*, pages 232–240. IEEE, 2019.

[72] K. Kilpi, S. A. Elprama, and A. Jacobs. Exploring privacy and trust issues in a future immersive videoconferencing system. In *2013 46th Hawaii International Conference on System Sciences*, pages 315–324, 2013.

[73] H. Kim and J. L. Gabbard. Assessing distraction potential of augmented reality head-up displays for vehicle drivers. *Human Factors*, 64(5):757–769, 2022.

[74] H. Kong, X. Xu, J. Yu, Q. Chen, C. Ma, Y. Chen, Y.-C. Chen, and L. Kong. m3track: mmwave-based multi-user 3d posture tracking. In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*, pages 491–503, 2022.

[75] G. Korkmaz, E. Ekici, and F. Ozguner. Black-burst-based multihop broadcast protocols for vehicular networks. *IEEE Transactions on Vehicular Technology*, 56(5):3159–3167, 2007.

[76] G. Korkmaz, E. Ekici, F. Özgüner, and Ü. Özgüner. Urban multi-hop broadcast protocol for inter-vehicle communication systems. In *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*, pages 76–85. ACM, 2004.

[77] J. Kuang, X. Niu, and X. Chen. Robust pedestrian dead reckoning based on mems-imu for smartphones. *Sensors*, 18(5):1391, 2018.

[78] Y. C. Kuo, C. M. Fu, C. T. Tsai, C. C. Lin, and G. H. Chang. Pedestrian collision warning of advanced driver assistance systems. In *Proceedings - 2016 IEEE International Symposium on Computer, Consumer and Control, IS3C 2016*, 2016.

[79] S. Landry, M. Jeon, P. Lautala, and D. Nelson. Design and assessment of in-vehicle auditory alerts for highway-rail grade crossings. *Transportation Research Part F: Traffic Psychology and Behaviour*, 62:201–214, 2019.

[80] J. Lau, B. Zimmerman, and F. Schaub. Alexa, are you listening? privacy perceptions, concerns and privacy-seeking behaviors with smart speakers. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):1–31, 2018.

[81] E. Lee, K.-M. Park, B.-h. Lee, S.-C. Kim, and J. Choi. Cnn-based real-time walking direction estimation for pedestrian navigation scenarios. *IEEE Sensors Journal*, 2023.

[82] K. Lee, Y. Yang, O. Prabhune, A. L. Chithra, J. West, K. Fawaz, N. Klingensmith, S. Banerjee, and Y. Kim. Aerokey: Using ambient electromagnetic radiation for secure and usable wireless device authentication. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(1):1–29, 2022.

[83] C. Li et al. V2PSense: Enabling cellular-based V2P collision warning service through mobile sensing. In *IEEE International Conference on Communications (ICC)*, pages 1–6, 2018.

[84] H. Li, Y. He, L. Sun, X. Cheng, and J. Yu. Side-channel information leakage of encrypted video stream in video surveillance systems. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.

[85] J. Li, Z. Li, G. Tyson, and G. Xie. Your privilege gives your privacy away: An analysis of a home security camera service. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 387–396. IEEE, 2020.

[86] K. Lin et al. Energy-accuracy trade-off for continuous mobile device location. In *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 285–298, 2010.

[87] F. Liu, K. Liu, X. Guo, G. Chen, P. Zhou, and J. Yang. Pedestrian heading estimation based on f-lstm neural network. In *2023 IEEE 16th International Conference on Electronic Measurement & Instruments (ICEMI)*, pages 417–422. IEEE, 2023.

[88] J. Lorenzo, I. Parra, F. Wirth, C. Stiller, D. F. Llorca, and M. A. Sotelo. Rnn-based pedestrian crossing prediction using activity and pose-related features. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1801–1806. IEEE, 2020.

[89] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell. Soundsense: scalable sound sensing for people-centric applications on mobile phones. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 165–178, 2009.

[90] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[91] S. Ma, X. Yan, J. Yang, and R. Liu. Influence of in-vehicle audio warning on drivers' eye-movement and behavior at flashing light-controlled grade crossings. *Human Factors*, 66(3):507–518, 2024.

[92] A. Manos, I. Klein, and T. Hazan. Gravity direction estimation and heading determination for pedestrian navigation. In *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 206–212. IEEE, 2018.

[93] J. Martinez-Carballido and M. Morales-Velázquez. Using adaptive threshold to detect pedestrians crossing on a street for advanced driver assistance systems. *CONIELECOMP 2012 - 22nd International Conference on Electronics Communications and Computing*, 2012.

[94] M. L. McHugh. Interrater reliability: the kappa statistic. *Biochem Med (Zagreb)*, 22(3):276–282, 2012.

[95] C. W. Meetings. Cisco webex meetings privacy data sheet. `https://www.cisco.com/c/en/us/products/collateral/conferencing/webex-meeting-center/cisco-webex-meetings-privacy-data-sheet.html`, September 2021. Accessed: 2021-11-29.

[96] A. Mhaidli, M. K. Venkatesh, Y. Zou, and F. Schaub. Listen only when spoken to: Interpersonal communication cues as smart speaker privacy controls. *Proceedings on Privacy Enhancing Technologies*, 2020(2):251–270, 2020.

[97] Microsoft. Microsoft privacy statement. `https://privacy.microsoft.com/en-us/privacystatement`, Oct 2021.

[98] C. Mihalcik. Microsoft listened to skype calls with 'no security' to protect recordings, report says, Jan 2020. Accessed: 2021-11-28.

[99] R. Q. Mínguez, I. P. Alonso, D. Fernández-Llorca, and M. A. Sotelo. Pedestrian path, pose, and intention prediction through gaussian process dynamical models and pedestrian activity recognition. *IEEE Transactions on Intelligent Transportation Systems*, 20(5):1803–1814, 2018.

[100] A. Møgelmose, M. M. Trivedi, and T. B. Moeslund. Trajectory analysis and prediction for improved pedestrian safety: Integrated framework and evaluations. In *2015 IEEE intelligent vehicles symposium (IV)*, pages 330–335. IEEE, 2015.

[101] National Coordination Office. Official u.s. government information about the global positioning system (gps) and related topics, mar 2022.

[102] National Highway Traffic Safety Administration. Bicycle safety - countermeasures that work. `https://www.nhtsa.gov/book/countermeasures-that-work/bicycle-safety`, 2024. Accessed: 2024-09-05.

[103] National Highway Traffic Safety Administration. Pedestrian safety. https://www.nhtsa.gov/road-safety/pedestrian-safety, 2024. Accessed: 2024-09-05.

[104] National Highway Traffic Safety Administration. Pedestrian safety - countermeasures that work. https://www.nhtsa.gov/book/countermeasures-that-work/pedestrian-safety, 2024. Accessed: 2024-09-05.

[105] L. H. NEWMAN. Zoom will fix the flaw that let hackers hijack webcams. *Wired*, July 2019.

[106] M. Nishishiba, M. Jones, and M. Kraner. Comparing means of more than two groups: Analysis of variance (anova). *Research Methods and Statistics for Public and Nonprofit Administrators: A Practical Guide*, pages 193–221, 2014.

[107] L. O'Gorman. Video privacy filters with tolerance to segmentation errors for video conferencing and surveillance. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 1835–1838. IEEE, 2012.

[108] P. O. . OSgeo. Postgis: a postgresql external extension., 2023.

[109] P. OSRM. Modern c++ routing engine for shortest paths in road networks. https://project-osrm.org/, 2023. [Online; accessed 15-November-2023].

[110] A. Palffy, J. F. Kooij, and D. M. Gavrila. Occlusion aware sensor fusion for early crossing pedestrian detection. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1768–1774. IEEE, 2019.

[111] E. Pan, J. Ren, M. Lindorfer, C. Wilson, and D. Choffnes. Panoptispy: Characterizing audio and video exfiltration from android applications. *Proceedings on Privacy Enhancing Technologies*, 2018(4):33–50, 2018.

[112] Z. H. Pang, L. Z. Fan, Z. Dong, Q. L. Han, and G. P. Liu. False data injection attacks against partial sensor measurements of networked control systems. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 69(1):172–176, 2022.

[113] A. Paul, R. Chauhan, R. Srivastava, and M. Baruah. Advanced driver assistance systems. *SAE Technical Papers*, 2016.

[114] L. Pei, D. Liu, D. Zou, R. L. F. Choy, Y. Chen, and Z. He. Optimal heading estimation based multidimensional particle filter for pedestrian indoor positioning. *IEEE Access*, 6:49705–49720, 2018.

[115] M. I. Perdana, W. Anggraeni, H. A. Sidharta, E. M. Yuniarno, and M. H. Purnomo. Early warning pedestrian crossing intention from its head gesture using head pose estimation. In *2021 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, pages 402–407. IEEE, 2021.

[116] F. Piccoli, R. Balakrishnan, M. J. Perez, M. Sachdeo, C. Nunez, M. Tang, K. Andreasson, K. Bjurek, R. D. Raj, E. Davidsson, et al. Fussi-net: Fusion of spatio-temporal skeletons for intention prediction network. In *2020 54th asilomar conference on signals, systems, and computers*, pages 68–72. IEEE, 2020.

[117] K. Rasmussen et al. Realization of RF distance bounding. In *USENIX Security Symposium*, pages 389–402, 2010.

[118] J. Rosenblatt. Zoom sued for allegedly illegally disclosing personal data. `https://www.bloomberg.com/news/articles/2020-03-31/zoom-sued-for-allegedly-illegally-disclosing-personal-data`, Mar 2020. Accessed: 2021-11-28.

[119] N. Roy, H. Wang, and R. Roy Choudhury. I am a smartphone and i can tell my user's walking direction. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pages 329–342, 2014.

[120] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras. Human motion trajectory prediction: A survey. *The International Journal of Robotics Research*, 39(8):895–935, 2020.

[121] M. Schäfer et al. Secure track verification. In *IEEE Symposium on Security and Privacy (S&P)*, pages 199–213, 2015.

[122] M. Schäfer et al. Secure motion verification using the doppler effect. In *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, pages 135–145, 2016.

[123] C. Schöller, V. Aravantinos, F. Lay, and A. Knoll. What the constant velocity model can teach us about pedestrian motion prediction. *IEEE Robotics and Automation Letters*, 5(2):1696–1703, 2020.

[124] A. T. Schulz and R. Stiefelhagen. A controlled interactive multiple model filter for combined pedestrian intention recognition and path prediction. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 173–178. IEEE, 2015.

[125] R. Schuster, V. Shmatikov, and E. Tromer. Beauty and the burst: Remote identification of encrypted video streams. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1357–1374, 2017.

[126] Q. Shahab, J. Terken, and B. Eggen. Auditory messages for speed advice in advanced driver assistance systems. In *AutomotiveUI 2010 - 2nd International Conference on Automotive User Interfaces and Interactive Vehicular Applications, Conference Proceedings*, 2010.

[127] S. Shen, M. Gowda, and R. Roy Choudhury. Closing the gaps in inertial motion tracking. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 429–444, 2018.

[128] B. I. Sighencea, R. I. Stanciu, and C. D. Căleanu. A review of deep learning-based methods for pedestrian trajectory prediction. *Sensors*, 21(22):7543, 2021.

[129] A. St. John. It's not just zoom. google meet, microsoft teams, and webex have privacy issues, too. Available online: `https://www.consumerreports.org/video-conferencing-services/videoconferencing-privacy-issues-google-microsoft-webex-a7383469308/`, April 2020. Accessed: 2021-11-28.

[130] K. Sun, C. Chen, and X. Zhang. " alexa, stop spying on me!" speech privacy protection against voice assistants. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, pages 298–311, 2020.

[131] M. Sun et al. A data trust framework for VANETs enabling false data detection and secure vehicle tracking. In *IEEE Conference on Communications and Network Security (CNS)*, pages 1–9, 2017.

[132] M. Sun et al. Svm: secure vehicle motion verification with a single wireless receiver. In *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, pages 65–76, 2020.

[133] W. Sun, C. Yang, S. Jin, and S. Choi. Listen channel randomization for faster wi-fi direct device discovery. In *IEEE INFOCOM 2016, The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.

[134] P. Szagala, P. Olszewski, W. Czajewski, and P. Dabkowski. Active signage of pedestrian crossings as a tool in road safety management. *Sustainability (Switzerland)*, 13(16), 2021.

[135] B. Szyk. Stopping distance calculator. `https://www.omnicalculator.com/physics/stopping-distance`, 2023. Accessed: 2023-11-11.

[136] S. T. Overview of motor vehicle traffic crashes in 2021. Technical Report No. DOT HS 813 435, National Highway Traffic Safety Administration, 2023 April.

[137] Y. Tateyama, H. Yamada, J. Noyori, Y. Mori, K. Yamamoto, T. Ogi, H. Nishimura, N. Kitamura, and H. Yashiro. Observation of drivers' behavior at narrow roads using immersive car driving simulator. In *Proceedings of the 9th ACM SIGGRAPH Conference on Virtual-Reality Continuum and its Applications in Industry*, pages 391–396, 2010.

[138] Termux. Android terminal emulator and linux environment., 2023.

[139] The Cisco Newroom. Cisco webex powers personal well-being, higher performing teams and inclusive collaboration. `https://newsroom.cisco.com/press-release-content?type=webcontent&articleId=2151037`, March 2021. Accessed: 2021-11-28.

[140] V. Thio, K. B. Ånonsen, and J. K. Bekkeng. Relative heading estimation for pedestrians based on the gravity vector. *IEEE Sensors Journal*, 21(6):8218–8225, 2021.

[141] H. Vasudev et al. Secure message propagation protocols for IoVs communication components. *Computers & Electrical Engineering*, 82:106555, 2020.

[142] B. Völz, K. Behrendt, H. Mielenz, I. Gilitschenski, R. Siegwart, and J. Nieto. A data-driven approach for pedestrian intention estimation. In *2016 ieee 19th international conference on intelligent transportation systems (itsc)*, pages 2607–2612. IEEE, 2016.

[143] B. Völz, H. Mielenz, I. Gilitschenski, R. Siegwart, and J. Nieto. Inferring pedestrian motions at urban crosswalks. *IEEE Transactions on Intelligent Transportation Systems*, 20(2):544–555, 2018.

[144] C. Wang, S. Kennedy, H. Li, K. Hudson, G. Atluri, X. Wei, W. Sun, and B. Wang. Fingerprinting encrypted voice traffic on smart speakers with deep learning. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 254–265, 2020.

[145] Q. Wang, H. Luo, L. Ye, A. Men, F. Zhao, Y. Huang, and C. Ou. Pedestrian heading estimation based on spatial transformer networks and hierarchical lstm. *IEEE Access*, 7:162309–162322, 2019.

[146] K. Weibull, B. Lidestam, and E. Prytz. False alarm effects in early warnings for emergency vehicles: Exploring drivers' move-over behavior. *Human Factors*, 2023.

[147] Whereby. Privacy policy. `https://whereby.com/information/tos/privacy-policy/`, Apr 2021.

[148] Wikipedia. Wikipedia, rotation matrix, 2023.

[149] J. Wu, H. Xu, Y. Zheng, and Z. Tian. A novel method of vehicle-pedestrian near-crash identification with roadside lidar data. *Accident Analysis & Prevention*, 121:238–249, 2018.

[150] X. Wu et al. Cars talk to phones: A DSRC based vehicle-pedestrian safety system. In *IEEE Vehicular Technology Conference (VTC-Fall)*, pages 1–7, 2014.

[151] Y. Wu, H.-B. Zhu, Q.-X. Du, and S.-M. Tang. A survey of the research status of pedestrian dead reckoning systems based on inertial sensors. *International Journal of Automation and Computing*, 16:65–83, 2019.

[152] H. Xing, J. Li, B. Hou, Y. Zhang, M. Guo, et al. Pedestrian stride length estimation from imu measurements and ann based algorithm. *Journal of Sensors*, 2017, 2017.

[153] Y. Xu, D. Xu, S. Lin, T. X. Han, X. Cao, and X. Li. Detection of sudden pedestrian crossings for driving assistance systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(3):729–739, 2011.

[154] D. Yang, H. Zhang, E. Yurtsever, K. A. Redmill, and Ü. Özgüner. Predicting pedestrian crossing intention with feature fusion and spatio-temporal attention. *IEEE Transactions on Intelligent Vehicles*, 7(2):221–230, 2022.

[155] S. Yang, J. Liu, X. Gong, G. Huang, and Y. Bai. A robust heading estimation solution for smartphone multisensor-integrated indoor positioning. *IEEE Internet of Things Journal*, 8(23):17186–17198, 2021.

[156] Y. Yang, B. Huang, Z. Xu, and R. Yang. A wifi assisted pedestrian heading estimation method using gyroscope. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 535–544. IEEE, 2020.

[157] Y. Yang, K. Lee, Y. Kim, and K. Fawaz. Pedro: Secure pedestrian mobility verification in v2p communication using commercial off-the-shelf mobile devices. In *Proceedings of the 2th Workshop on CPS&IoT Security and Privacy*, pages 41–46, 2021.

[158] Y. Yang, J. Li, and K. Fawaz. Reliable heading tracking for pedestrian road crossing prediction using commodity devices. `https://arxiv.org/abs/2410.06400`, 2024.

[159] Y. Yang, J. West, G. K. Thiruvathukal, N. Klingensmith, and K. Fawaz. Are you really muted?: A privacy analysis of mute buttons in video conferencing apps. *22nd Privacy Enhancing Technologies Symposium (PETS2022)*, 2022.

[160] C. Zhang and C. Berger. Pedestrian behavior prediction using deep learning methods for urban scenarios: A review. *IEEE Transactions on Intelligent Transportation Systems*, 24(10):10279–10301, 2023.

[161] C. Zhang, Y. Rui, and L.-w. He. Light weight background blurring for video conferencing applications. In *2006 International Conference on Image Processing*, pages 481–484. IEEE, 2006.

[162] H. Zhang, Y. Liu, C. Wang, R. Fu, Q. Sun, and Z. Li. Research on a pedestrian crossing intention recognition model based on natural observation data. *Sensors*, 20(6):1776, 2020.

[163] S. Zhang, M. Abdel-Aty, Y. Wu, and O. Zheng. Pedestrian crossing intention prediction at red-light using pose estimation. *IEEE transactions on intelligent transportation systems*, 23(3):2331–2339, 2021.

[164] S. Zhang, M. Abdel-Aty, J. Yuan, and P. Li. Prediction of pedestrian crossing intentions at intersections based on long short-term memory recurrent neural network. *Transportation research record*, 2674(4):57–65, 2020.

[165] X. Zhang, Z. Li, and J. Zhang. Synthesized millimeter-waves for human motion sensing. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, pages 377–390, 2022.

[166] J. Zhao, Y. Li, H. Xu, and H. Liu. Probabilistic prediction of pedestrian crossing intention using roadside lidar data. *IEEE Access*, 7:93781–93790, 2019.

[167] L. Zheng, X. Zhan, X. Zhang, S. Wang, and W. Yuan. Heading estimation for multimode pedestrian dead reckoning. *IEEE Sensors Journal*, 20(15):8731–8739, 2020.

[168] Y. Zheng, Y. Zhang, K. Qian, G. Zhang, Y. Liu, C. Wu, and Z. Yang. Zero-effort cross-domain gesture recognition with wi-fi. In *Proceedings of the 17th annual international conference on mobile systems, applications, and services*, pages 313–325, 2019.

[169] P. Zhou, M. Li, and G. Shen. Use it free: Instantly knowing your phone attitude. In *Proceedings of the 20th annual international conference on Mobile computing and networking*, pages 605–616, 2014.