APPLYING MULTIMODAL SENSING TO HUMAN MOTION TRACKING IN MOBILE SYSTEMS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Siyuan Cao

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

I dedicate this dissertation to the people who have supported me,
especially to my parents.

ACKNOWLEDGMENTS

I am deeply indebted to many: To my advisor, Professor He Wang, for his continuous support and guidance throughout my entire Ph.D life. His enthusiasm, sincerity, vision and motivation have deeply inspired me. He gave me the precious opportunity to do research on mobile sensing and has generously provided all possible resources to support my work. I will never forget the time that he spent with me in our lab discussing new ideas and testing designed systems. I remember when I was submitting my first paper in which he was the only coauthor, he even came to guide me through the experiments. He joked "I'm the most supportive coauther you would have." — which I think is true. I would also like to thank him for his patience,understanding, and encouragement, when I was confused and hesitant. He kindly share his thoughts and experience to help me make important decisions. Without these, this dissertation would not have been possible.

To Purdue University and the Department of Computer Science for all the support during my Ph.D. study. I especially thank Professor Chunyi Peng and Professor Kihong Park for serving on my doctoral committee and for their insightful comments. I also thank Professor Ninghui Li and Professor Ming Yin for serving on my examining committee.

To my past and present labmates. I especially thank Habiba Farrukh, Zhiyuan Cheng, and Xin Xie for their collaboration.

To all the volunteers who participated my user study.

To my parents and Chengyuan for their endless love, belief, and support.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

ABBREVIATIONS

ABT	association based tracking
BLE	bluetooth low energy
CDF	cumulative distribution function
CFT	category free tracking
FFT	fast fourier transform
FMCW	frequency-modulated continuous wave
HMD	head-mounted display
IMU	inertial measurement unit
PCA	principal component analysis
RFID	radio-frequency identification
SCTU	single camera tracking unit
SLAM	simultaneous localization and mapping
VR	virtual reality

ABSTRACT

Cao, Siyuan Ph.D., Purdue University, August 2020. Applying Multimodal Sensing to Human Motion Tracking in Mobile Systems. Major Professor: He Wang.

Billions of “smart” things in our lives have been equipped with various sensors. Current devices, such as smartphones, smartwatches, tablets, and VR/AR headsets, are equipped with a variety of embedded sensors, e.g. accelerometer, gyroscope, magnetometer, camera, GPS sensor, etc. Based on these sensor data, many technologies have been developed to track human motion at different granularities and to enable new applications. This dissertation examines two challenging problems in human motion tracking. One problem is the ID association issue when utilizing external sensors to simultaneously track multiple people. Although an “outside” system can track all human movements in a designated area, it needs to digitally associate each tracking trajectory to the corresponding person, or say the smart device carried by that person, to provide customized service based on the tracking results. Another problem is the inaccuracy caused by limited sensing information when merely using the embedded sensors located on the devices being tracked. Since sensor data may contain inevitable noises and there is no external beacon used as a reference point for calibration, it is hard to accurately track human motion only with internal sensors.

In this dissertation, we focus on applying multimodal sensing to perform human motion tracking in mobile systems. To address the two above problems separately, we conduct the following research works. (1) The first work seeks to enable public cameras to send personalized messages to people without knowing their phone addresses. We build a system which utilizes the users’ motion patterns captured by the cameras as their communication addresses, and depends on their smartphones to locally compare the sensor data with the addresses and to accept the correct messages. To protect user privacy, the system requires no data from the users and transforms the motion patterns into low-dimensional codes to prevent motion leaks. (2) To enhance distinguishability and scalability of the camera-to-human communication system, we introduce context features which include both motion patterns and ambience features (e.g. magnetic field, Wi-Fi fingerprint, etc.) to identify people. The enhanced system achieves higher association accuracy and is demonstrated to work with dense people in a retailer, with a fixed-length packet overhead. The first two works explore the potential of widely deployed surveillance cameras and provide a generic underlay to various practical applications, such as automatic audio guide, indoor localization, and sending safety alerts. (3) We close this dissertation with a fine-grained motion tracking system which aims to track the positions of two hand-held motion controllers in a mobile VR system. To achieve high tracking accuracy without external sensors, we introduce new types of information, e.g. ultrasonic ranging among the headset and the controllers, and a kinematic arm model. Effectively fusing this additional information with inertial sensing generates accurate controller positions in real time. Compared with commodity mobile VR controllers which only support rotational tracking, our system provides an interactive VR experience by letting the user actually move the controllers' positions in a VR scene. To summarize, this dissertation shows that multimodal sensing can further explore the potential power in sensor data and can take sensor-based applications to the next generation of innovation.

1. INTRODUCTION

Smart devices are changing our lives. Many mobile devices that we use every day, such as smartphones, smartwatches, fitness bands, tablets, and VR/AR headsets, are functioning more and more intelligently. Equipped with a variety of sensors and powerful processors, these devices can sense human-centered environments, process collected data, retrieve useful information, communicate with other devices, and adapt its operation autonomously. Billions of smart mobile devices existing in our surroundings [1] help us to better understand human behavior from multiple aspects and enhance human-device interaction. Moreover, they open up new possibilities and applications, for example, smart home, smart city, cashierless store, self-driving car, virtual reality, etc.

As smart devices are becoming a new trend, utilizing embedded sensors on them to understand human behaviors is at the frontier of the related research. Accurate human motion tracking at different granularities can enable various applications. Tracking a person's walking trajectory [2, 3] can enable navigation, provide location-aware service, and record its shareable moments. Activity detection [4–6] can help to monitor people's sleep patterns and conduct fall detection for seniors. Gesture detection [7,8] allows device-free remote control. Hand or hand-held device tracking [9,10] improves user experience in interactive gaming and virtual reality applications. By tracking a user's finger positions [11,12], a virtual keyboard can be developed for televisions to remotely input or for smartwatches to expand their screens of limited size. Fine-grained expression detection [13] has also been explored for emotion prediction and analysis. As the above applications are innovative and promising, many works have explored the possibilities — from both research and product perspective.

In this chapter, we first describe the existing human motion tracking technologies and two main problems in Chapter 1.1. One problem is the ID association issue when relying on external stationary sensors to track multiple people at the same time. Another problem is the inaccuracy caused by limited sensing information when relying on sensors located on the objects being tracked. We discuss when and why these problems arise and possible solutions to them. Then we briefly describe specific scenarios where we aim to solve these problems and our contributions in Chapter 1.2. Finally, Chapter 1.3 provides the organization of this dissertation.

1.1 Problems in Human Motion Tracking Technologies

Current human motion tracking systems fall into two categories depending on where the sensors used for tracking locate. One category utilizes external sensors in surroundings to track a person's motion in a certain area; while the other category relies on sensors attached to a human body or built-in sensors on the smart devices carried by a person for tracking. Based on many aspects of a specific application scenario, such as performance requirement, computation power, portability, motion range, etc., either method or a combination of them is chosen accordingly.

1.1.1 ID Association when Using External Sensors on Multiple People

One category of tracking systems utilizes stationary sensors placed in the environment to track objects' movement in a designated area. Some early VR systems, such as Oculus Rift, rely on infra-red markers on devices and cameras placed in a room to perform positional tracking for the headset and controllers. Other external sensors like Wi-Fi access points [7] and sound transceivers [14], are also used to track human gestures or detect respiration. With additional hardware, these systems are able to provide high-accuracy and high-resolution tracking results. But meanwhile, they require extra setup effort, for example, room calibration after placing the external sensors and strict clock synchronization between the sensors and the devices. Moreover, these systems always rely on a computer or server where they stream the data collected by the external sensors and aggregate the data in real time. All of these make this type of tracking system suitable for long-term and low-portability usage scenarios, especially where external sensors already exist in the environment and can be borrowed.

One advantage of the systems relying on external sensors is that they can track the movement of multiple devices or even multiple people simultaneously. However, this introduces a practical issue of *how to digitally associate each tracking trajectory with the corresponding device/person.* Such a problem happens in various scenarios. For example, when using external sensors to track multiple users playing VR games in the same area, a system can locate the devices all at once and generate real-time VR scenes for each user. But when sending the scenes to the corresponding headset for rendering, it needs a way to match the headsets captured in the sensor data with their communication addresses. Otherwise, the system won't know to which device it should send the computed scenes. Another example would be providing navigation service in public areas or deliver promotion information to a retailer customer's smartphone based on the customer's real-time shopping behavior. Existing surveillance cameras in these environments can be adopted to conduct pedestrian tracking and behavior analysis, while we still need a way to associate each person captured in the images with the smartphone that it carries to deliver the customized information to the correct person.

An intuitive solution to this problem is adding a pre-registration stage before usage to digitally associate a tracking trajectory from the "outside" with a person (or the smart device that it carries). For example, each user follows the instructions given by the system to perform a series of actions like waving hands or simply uploads a face photo for recognition. This may be possible in some scenarios like a VR party — each user gets registered only once and plays for a few hours. However, for applications in public areas, it is impractical to ask each person to do pre-registration since it requires user involvement and raises privacy concerns. Therefore, it is necessary to solve the ID association issue to fully leverage the power of the tracking systems based on external sensors. In this dissertation, we would like to explore the possibility of using motion patterns as a temporary identifier for a person. Using this unique identifier as the communication address, we will be able to send the customized information (e.g. navigation guidance and targeted ads) based on the locations and gestures obtained from the tracking results to the user's smartphone. Since both the "outside" tracking system and the smart device carried by the user can sense how the user moves, we want to utilize the consistency between the two sides for ID association.

1.1.2 Inaccuracy with Limited Sensing Information from Internal Sensors

The other category of tracking schemes utilizes sensors located on the devices being tracked to perform positional tracking relative to the environment. In contrast to the aforementioned schemes, they do not rely on stationary external sensors for tracking. Some recent VR headsets, such as Microsoft HoloLens and Oculus Quest, leverage vision-based solutions. They are equipped with a set of built-in cameras and leverage simultaneous localization and mapping (SLAM) [15] algorithm to track their 3D positions based on the visual feature points extracted from the environment. Smart devices, like smartphones and smartwatches, have a variety of built-in sensors, such as accelerometer and gyroscope. They can use the IMUs to track their users' hand movement for air-handwriting [16] or to imply what a person types on a keyboard [17]. In addition to the motion sensors, the smart devices are also equipped with many context sensors that sense the environment. The context sensing information, such as magnetic field [2], light [18, 19], sound [20, 21], Wi-Fi signals [2, 3], cellular signals [22], etc., have all been used in motion tracking of different granularities, for example, indoor localization [2, 3, 18, 22], gesture detection [19], and virtual keyboard [20, 21]. Without the need of setting up external sensors, these systems can also offload computation tasks to remote servers as cloud computing is becoming a trend. This enables applications that require high portability like mobile VR.

But merely relying on embedded sensors on commodity devices means that the information that we obtain can be limited regarding both its amount and quality.

For instance, the images captured by the cameras on the VR headsets can be blurry in the cases of imperfect light conditions and fast movement. Sometimes sensor data is noisy due to either the sensor quality (e.g. sound distortion at microphone) or the changing environment (e.g. multipath effect on wireless signals). Without any external beacons, we have no reference points to correct sensor drift like in gyroscope readings. These present challenges to *achieving accurate tracking results with limited sensing information from internal sensors.*

Many things can be done from various aspects to achieve this goal: using a large set of sensors (HoloLens has 6 cameras on the headset and Quest has 4), bringing richer information to the tracking scheme by using sensors of different types [2, 23], and designing new algorithms to effectively fuse the collected data to generate better tracking results [16, 24]. When designing a motion tracking system with internal sensors for a specific application scenario, we must make a trade-off between the performance (e.g. accuracy, resolution, delay, and update rate, etc.) and the cost (e.g. the number and type of the sensors, computation power, etc.); and an optimal solution is usually a combination of multiple technologies. In this dissertation, we would like to introduce additional information, such as acoustic ranging and a kinematic arm model, to refine position ranges and simultaneously track multiple objects. Without stationary external beacons, we want to explore the possibility of using the moving objects as the reference points for each other, and provide real-time accurate tracking for each object from their relative location information.

1.2 Our Contributions

This dissertation aims to address the above two problems separately. Now we define the concrete problems and describe the specific application scenarios that we consider.

The first problem is to enable surveillance cameras in public areas to send personalized messages to people. Although cameras have been widely deployed and used for video analysis to understand human behaviors, their potential for providing customized service to users is not fully explored due to the lack of direct communication from the cameras to people. Since the cameras cannot get any communication addresses (e.g. IP/MAC address of a user's phone) from their perspective, we need to find a way to digitally associate each person captured in the camera view to its smartphone. If such an association exists, it will open up new applications, such as sending alerts to pedestrians on a street, providing automatic audio guide based on a visitor's gestures in a museum, and enabling accurate indoor localization for real-time navigation. To build an affiliation between a person and its phone, we take advantage of the consistency between motion and context features extracted from the video side and the embedded sensor side. There are two main challenges that we need to overcome. First, the system should protect the users' privacy by not directly revealing their sensing information to the public. Since sensing data may indicate a person's walking history and even important personal information such as gait and health condition, we should avoid transmitting raw sensing data and prevent possible information leaks. Second, when designing new communication addresses, the system needs to maintain high distinguishability among a group of people with a limited packet payload overhead. It is inefficient to send a packet of which only a small part is useful data. Our goal is to limit the payload overhead to be the same size as a standard IPv6 header.

The second problem is to track the positions of two motion controllers in a mobile VR system without using external sensors. In contrast to high-end standalone VR headsets, mobile VR has merely a smartphone mounted as its headset, therefore has no dedicated cameras or sensors to rely on. As such, current mobile VR systems only track their controllers' rotations by applying dead reckoning on accelerometer and gyroscope readings, but cannot locate the controllers' positions. This makes the current mobile VR controllers function as a laser pointer — it is acceptable for simple actions like selecting a menu, but will fall short in playing interactive games like archery and slingshot. Therefore, we introduce two types of additional information as further constraints to realize positional tracking on the two controllers. One is pair-wise distances among the three moving devices (i.e. the headset and the two controllers) obtained from ultrasonic ranging. Another one is a kinematic arm model — it demonstrates that when a user is holding the controllers, their orientations and positions are strongly coupled. With this insight, building such a tracking system is still challenging from the following perspectives. First, ultrasonic ranging between two unsynchronized and fast-moving devices is not trivial. We need to consider various factors that may affect the distance measurements, for example, imperfect sound signals due to the lack of dedicated hardware, propagation delay offset caused by separate clocks, and possible errors introduced by the Doppler effect. Second, since there will be sensing information provided by different types of sensors and all sensors may have occasional and inevitable noises in their readings, we need to carefully design a fusion algorithm to fully leverage the power of all types of sensors and meanwhile ensure that the tracking result is robust to the sensor noises.

In our works, we focus on *applying multimodal sensing to human motion tracking in mobile systems*. We believe that carefully fusing multiple types of sensor data can further exploit the power of the sensors and can take the sensor-based mobile systems to the next generation of innovation. In particular, we aim to explore how multimodal sensing can benefit various real-world systems which involve real-time interaction between the system and users. This dissertation includes the design of three systems related to human motion tracking, among which the first two aim to realize ID association to enable direct communication from surveillance cameras to people, and the third one designs a positional tracking system to enable six-degree-of-freedom controllers for mobile VR. Here we briefly introduce each of the three systems.

1.2.1 Camera-to-Human Communication using Motion Pattern

In Chapter 2, we try to answer the following question: Is it possible for cameras in public areas, say ceiling cameras in a museum, to send personalized messages to people without knowing any addresses of their phones? We define this kind of problem as Private Human Addressing and develop a real-time end-to-end system called *PHADE* to solve it. Unlike traditional data transmission protocols that need to first learn the destination's address, our system relies on viewing users' motion patterns through surveillance cameras and uses the uniqueness of these patterns as the address for communication. Once receiving the wireless broadcast from the cameras, a user's phone can locally compare the "motion address" of the packet against its own motion sensor data, and accept the packet upon a "good" match.

This system contributes to protecting user privacy. *PHADE* not only requires no data uploaded by users, but also transforms the motion patterns into low-dimensional codes using principal component analysis (PCA) to prevent leakage of user's walking behaviors. Thus, a hacker who collects all the broadcast messages would still not be able to infer the motion patterns of users. Real-world experiments show that *PHADE* discriminates 2, 4, 6, 8, 10 people with 98%, 95%, 90%, 90%, 87% correctness and about 3 seconds constant delay. Since abundant and accurate information can be extracted from videos, *PHADE* would find applications in various contexts. Extended to localization system and audio guide, *PHADE* achieves a median error of 0.19m and 99.7% matching correctness, respectively. We also demonstrate in our experiments that *PHADE* can deliver messages based on human gestures. This system utilizes the existing surveillance cameras that are already deployed in public areas and commodity smartphones that now almost everyone is carrying every day. Thus there is no need to deploy any extra infrastructures or to require users to rent any dedicated device.

1.2.2 Camera-to-Human Communication using Context Address

However, since *PHADE* only uses motion features as a person's communication address, its distinguishability and scalability are limited. Another issue is the large packet overhead. *PHADE* broadcasts packets to avoid asking the users to upload their sensor data. But including the large coefficient matrix generated by PCA in the packet introduces a large packet payload overhead. When working with 10 people, the overhead in each packet is 800 to 1000 bytes. So we continue working on this system to solve the two above problems. In Chapter 3,we introduce a new concept called "context features", which is extracted from videos and used as a person's sole address. The context address consists of: motion features, e.g. walking velocity; and ambience features, e.g. magnetic trend and Wi-Fi signal strengths. To reduce the payload overhead, we first discretize the features along the time axis and then select the ones that can effectively distinguish the target from others. Finally, the selected features are encoded into a fixed-length header as the communication address of the broadcast packet.

We highlight three novel components in our system: (1) definition of discriminative and noise-robust ambience features; (2) effortless ambient sensing map generation; (3) a context feature selection algorithm to dynamically choose lightweight yet effective features which are encoded into a fixed-length header. Real-world and simulated experiments are conducted for different applications. The enhanced system achieves a sending ratio of 98.5%, an acceptance precision of 93.4%, and a recall of 98.3% with ten people. A simulated experiment conducted in a real retailer demonstrates the system's feasibility and scalability when extended to a complex scenario with denser people (about 50). We design the system as a general framework so that it can be extended to include more types of features according to specific application scenarios. We believe this is one more step towards direct camera-to-human communication and will become a generic underlay to various practical applications.

1.2.3 Mobile VR Controller Tracking using Inertial-Ultrasonic Sensor Fusion

In Chapter 4, we present *VIU*, an inside-out controller tracking system designed for mobile VR, which utilizes existing on-chip sensors. The key idea is as follows: When a user is wearing a smartphone-mounted HMD and holding two controllers in its hands, *VIU* obtains the controllers' orientations via inertial sensing and pair-wise distances among the three devices through ultrasonic ranging. By fusing this information with an anatomical arm model, it restricts the search space and precisely locates the two motion controllers in real time. Compared with commodity controllers which only performs 3DoF rotational tracking, *VIU* introduces real 3D position information, therefore it provides a more realistic and interactive experience where the user can actually move the controllers' positions in the VR scene.

To implement such a system, we develop a series of innovative techniques. (1) We design a two-way FMCW algorithm to accurately measure the triangular distances and meanwhile avoid clock synchronization among the devices. (2) We deal with the Doppler effect caused by fast movement by adding additional sine waves to the ultrasound tones and correcting the detected Doppler shifts in FMCW frequency peaks. (3) We systematically design a statistical model to fuse the various types of sensing information. It accurately tracks the controllers' locations while remaining tolerant of inevitable sensor noises. We implemented a prototype and show that *VIU* achieves a tracking accuracy of 9.59cm in real experiments. When used in various VR applications,*VIU* outperforms prior work in a standard object selection test for input devices, and enables new types of game (e.g. slingshot) on mobile VR systems.

1.3 Organization

The rest of the dissertation is organized as the following. We discuss camera-to-Human communication using motion pattern in Chapter 2; camera-to-Human communication using context address in Chapter 3; and mobile VR controller tracking using inertial-ultrasonic sensor fusion in Chapter 4. Finally, we summarize our contributions and conclude the dissertation in Chapter 5.

2. ENABLING PUBLIC CAMERAS TO TALK TO THE PUBLIC

2.1 Introduction

Surveillance cameras are pervasively deployed in public areas, such as shopping malls, museums, galleries and so on [25]. Although videos captured by these cameras are widely used to identify people for security purposes, their utility is limited by the unavailability of any direct communication between people and the camera. To the best of our knowledge, there is no existing end-to-end and real-time system which digitally associates people in the camera view with their smartphones. If one develops such a system, it may be possible to deliver customized messages to a user's smartphone, even if no IP/MAC address of that smartphone is known. In this chapter, we define this problem as *Private Human Addressing* and aim to design a practical system that realizes it.

Fig. 2.1 illustrates an application scenario of the system. In a museum, people are walking around or standing to look at the exhibits. Although standing near the right side, the man may still receive an introduction to a painting on the left wall, which he is looking at. The messages are customized for some targets with certain external characteristics and transmitted using people's *behavioral addresses* as their identifiers, which are extracted from their movements in the video. For example, the introduction to a painting may be sent to those who show interest to it (the camera would know the user is looking or pointing at it); some activity information may be sent to those who are with their children (the camera can recognize children through their heights).

One may ask: why not attach a Bluetooth beacon to each exhibit and send messages when people approach it? One issue with this method is that it's hard to adjust

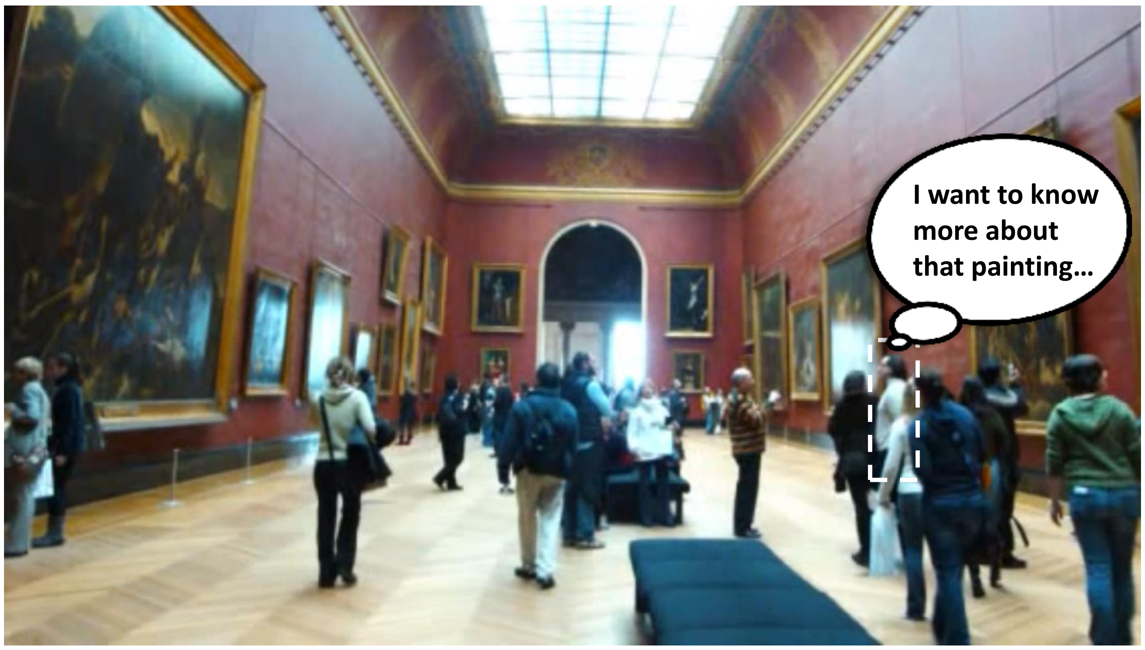


Fig. 2.1. A camera is able to send context-aware messages to a targeted user, without knowing any addresses of their phones.

the range of transmission, which should be considered case by case based on the distance between exhibits. Another issue is that we intend to enable context-aware messaging. The message content is not only based on the user's location but also related to other context attributes (e.g. human behaviors like pointing, gazing or hesitating; appearance features like height or clothing; something that is happening and requires the user's attention). Bluetooth-based beacons fall short in capturing this contextual information, while surveillance camera provides the opportunity of sending context-aware messages in many application scenarios. For instance, a system based on surveillance cameras can warn a pedestrian if it's texting while crossing the road. It can also enable superstore chains like Walmart to send the customers coupons, specific to products of their interest, in real time.

Prior works have presented some schemes for human ID association using cameras and sensors. ID-Match [26] uses RFID tags worn by people to assign a unique ID to each person in the camera view. The use of such tags requires user registration and is not suitable for public areas where not everyone might wear these tags. Insight [27, 28] demonstrates that motion patterns and clothing colors can be used as a temporary fingerprint for recognizing a person. [29] asks people to wear sensors on their belts and then associates people in the camera view with the accelerometer readings. However, none of these works implement a scalable and practical real-time system for applications requiring communication between the camera and humans. Moreover, the latter two require the users to upload sensor data to the server, which raises privacy concerns. A system solving the human addressing problem should ideally be *robust*, *real-time* and *privacy-protecting*, so that it can be extended to many context-aware applications.

The key ideas of our work are (1) to leverage the diversity in human motion features to distinguish individuals, and (2) to exploit the consistency between motion features extracted from video and sensors. The server receives streamed videos and performs pedestrian detection on each frame. Once the total amount of frames is large enough to make up a tracking window, the server runs pedestrian tracking schemes to associate a person in each frame to a continuous tracklet. Motion features, e.g. how the user moves and turns, are then extracted from the tracklet. After that, the server creates a packet with the application-centric content and inserts the motion features as the destination address; this message is broadcast to all clients over the wireless medium, say Wi-Fi. Upon the receipt of a message, the client compares its own sensor motion feature with the included motion address to determine if this message is for itself.

Implementing the above idea into a real system presents a number of challenges. First, today's existing pedestrian tracking procedure is not in real time. Initially, the tracking procedure takes about 5.3 times longer than receiving the video frames. This is not acceptable in a real system as the delay will accumulate and the messages generated for users may become stale. Second, there may be privacy concerns if the server requires the users to upload their sensor data, or if the server just broadcasts all motion patterns that are captured by the camera towards the public. Under the former condition, users can't keep anonymous if identified through sensor fingerprinting [30, 31]. Private information may also be referred from the uploaded data. In the latter case, users can easily know how others are behaving. Hackers can even recover users' walking traces if walking directions are included in the motion address.

To tackle these two challenges, we first accelerate the overall procedure on the server side. Overlapping communication and computation largely helps us to make our system real-time. Besides that, we also optimize the pedestrian tracking algorithm from several aspects to considerably speed up the tracking process. Secondly, to deal with the privacy concerns, we keep the users' personal sensing data within their smartphones and let the clients distributively make their own decisions based on received video motion features, which naturally keeps the users anonymous. Moreover, to prevent users' walking behavior from being revealed to the public, we transform the raw features via principal component analysis (PCA) [32], a commonly used technique to reduce the dimension of features. This blurs partial details about the walking patterns, ultimately preserving the main characteristics of users. We name our system as *PHADE* since the blurring processing *"fades" people's motion details out.*

PHADE is implemented into a real-time end-to-end system using Samsung Galaxy S4 as clients and S5 as IP cameras. Two PCs with dual NVIDIA GTX 1080 Ti SLI are used as a server running processes in a pipelined and parallel manner; Wi-Fi is used for video streaming from the cameras to the server and for broadcasting messages from the server to the clients. Evaluations from real world demonstrate the ability to discriminate 2, 4, 6, 8, 10 people with 98%, 95%, 90%, 90%, 87% correctness and about 3 seconds constant delay. When extending our human addressing system to do video-based localization, the median localization accuracy is $0.19m$, while for 99 percentile, the error is $0.65m$. When used as an automatic audio guide, the matching results are correct in 99.7% cases. Based on gestures detected via OpenPose [33], our system can also deliver messages according to arm pointing directions.

Our main contributions are summarized as follows.

- We propose a new notion called Private Human Addressing and design an end-to-end system to solve this problem in the real world.
- From a technical perspective, we accelerate the tracking process on the server and make it real-time; we design packet structure by selecting simple but effective motion features for computing.
- We investigate the possibility of motion leakage with different forms of broadcast features and protect user privacy by conducting PCA transformation on features and requiring no sensing data from users.
- We demonstrate the performance of *PHADE* in three application examples, i.e. indoor localization, automatic audio guide, and gesture-based messaging.

2.2 System Overview

Fig. 2.2 illustrates a functional overview of *PHADE*. Multiple cameras are continuously monitoring parts of a public area (e.g. museum, gallery or shopping mall) with some overlaps, and meanwhile streaming the recorded videos to a server. Once receiving a new video frame, the server conducts pedestrian detection [34] on it and caches the frame with its detection results into a buffer for further processing. Then the tracking unit associates pedestrian detection responses in continuous frames from each camera into local tracklets, where these tracklets represent various individuals in a camera view. After that, these local tracklets from different cameras are stitched in both spatial and temporal space to generate global tracklets representing individuals in the entire area.

Once the global tracklets have been generated, several types of motion features (e.g. isWalking, walking-direction, etc.) are extracted and reorganized into a set of

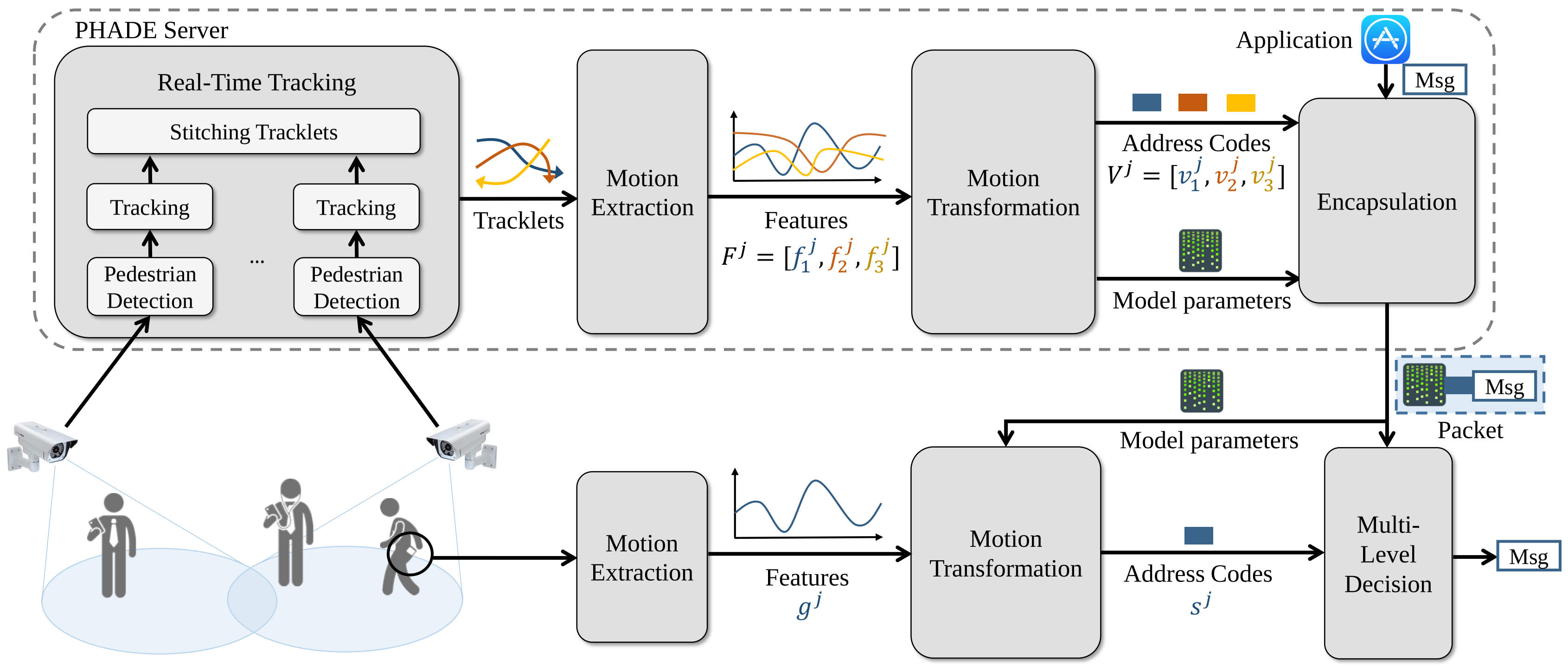


Fig. 2.2. System overview. Videos are streamed to a server to track people in each camera view. Local tracklets from various cameras are then stitched into global tracklets. Motion features for each tracklet are extracted and transformed into address codes to protect user behavior details. The server broadcasts messages labeled with address codes and model parameters. And the client will duplicate the same transformation on sensor motion features and locally decide which message to accept by comparing with the address codes.

feature vectors, $F_i = [f_i^1, f_i^2, ...]$, for user i, where f_i^j denotes the feature vector of type j. Since raw motion features contain abundant information about walking behaviors and may also reveal walking path histories, simply sending raw features to the public may cause privacy concerns among users. Therefore, feature vectors of the same type are integrated into a feature matrix $F^j = [f_1^j, f_2^j, ...]$, which will be transformed into address codes $V^j = [v_1^j, v_2^j, ...]$ using Principal Component Analysis (PCA) to diminish walking details and meanwhile preserve the distinguishable characteristics. The address code tuple $< v_i^1, v_i^2, ... >$ is used as the communication address for user i. To ensure the same transformation process can be duplicated by the clients on sensor-based features, the transformation-related parameters (e.g. feature timestamps, feature preprocessing parameters and coefficient matrix, etc.) are also saved. The address code tuple and model parameters are both sent to the users together with a piece of application-customized message in a structured packet. In order to run in real time, jobs conducted on the server are separated into several stages and accomplished in a pipelined and parallel manner. Messages are guaranteed to be sent to targeted users with a constant and short delay.

Although the clients are passively listening to all broadcast messages, they are given the freedom to choose whether to accept the messages and which one to accept. Once a smartphone carried by a user hears a broadcast packet, the smartphone will extract motion features $G = [g^1, g^2, ...]$ from its sensor readings. By utilizing the received model parameters, these sensor motion features are then transformed in the same way as on the server side to obtain the corresponding sensor address code $S = [s^1, s^2, ...]$. During the hierarchical comparison between tuples $< s^1, s^2, ... >$ and $< v_i^1, v_i^2, ... >$ for $i = 1, 2, ...$, the candidate message Msg_i for user i is passed with its video address code to the next decision level if the similarity is higher than a threshold. The multi-level matching decider finally comes up with a decision of whether to accept each received message Msg_i. After going through the entire matching levels, if only a single video address code fulfills all the threshold requirements, the decider will convey the corresponding message to the application level for further usage. Otherwise, the decider will generate a final decision of "Unsure" and discard all received messages.

2.3 Design Details

2.3.1 Multi-camera Real-time Human Tracking

This section first describes the tracking scheme cooperatively used on multiple cameras. Optimizations on both the tracking scheme and the server structure are then detailed.

Tracking Scheme

As shown in Fig. 2.3, the tracking scheme is composed of three functional modules. *Single Camera Tracking Unit* (SCTU) is applied to each sliding window [35] (W_i, with a length of 8 seconds and a step of 2 seconds) of a video stream and outputs local tracklets for people in an individual camera view. It adopts a unified framework in [36], combining two mainstream tracking approaches, i.e. Association Based Tracking (ABT) [36–40] and Category Free Tracking (CFT) [41–43]. Based on a pre-trained detector [44,45], ABT conservatively associates detection responses from neighboring frames into short low-level tracklets. Then CFT relies on the immediate detected regions to extend the head or tail of the low-level tracklets. This partial recovery of ends helps to reduce the gap between tracklets which represent an identical person. The extended tracklets are further associated using Hungarian algorithm [38,46] and smoothed by Kalman filter [47] to obtain high-level local tracklets. Finally, the local tracklets are *spatially stitched* between cameras and *temporally stitched* with tracklets in W_{i-1}, to generate latest global tracklets. Here we also add a latest human pose detection scheme called OpenPose [33,48,49], as a complement to the existing pre-trained detector. OpenPose not only improves the detecting and tracking accuracy

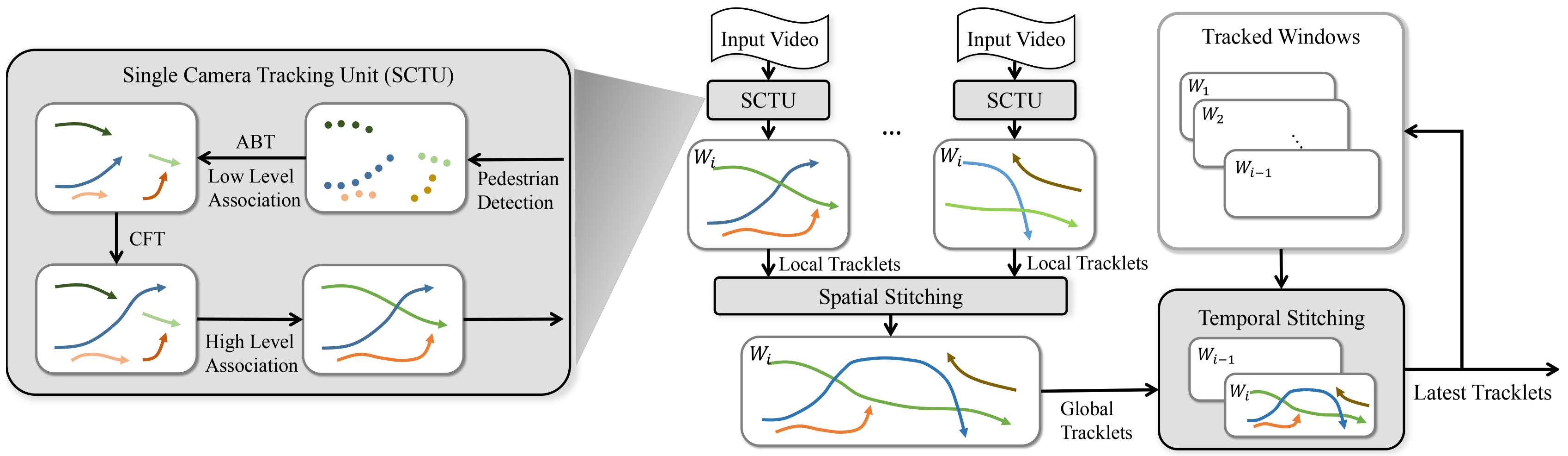


Fig. 2.3. Tracking scheme. Single Camera Tracking Unit is applied to each video stream to obtain local tracklets in the current sliding window. After going through four steps of pedestrian detection, low-level association, Category Free Tracking, and high-level association, the generated local tracklets for each camera are then spatially stitched into global tracklets in the entire region covered by camera views. To ensure tracklet IDs are inherited along the time, temporal stitching is then performed between current and previous windows. Finally, the fully stitched tracklets in the current window are stored for further feature extraction.

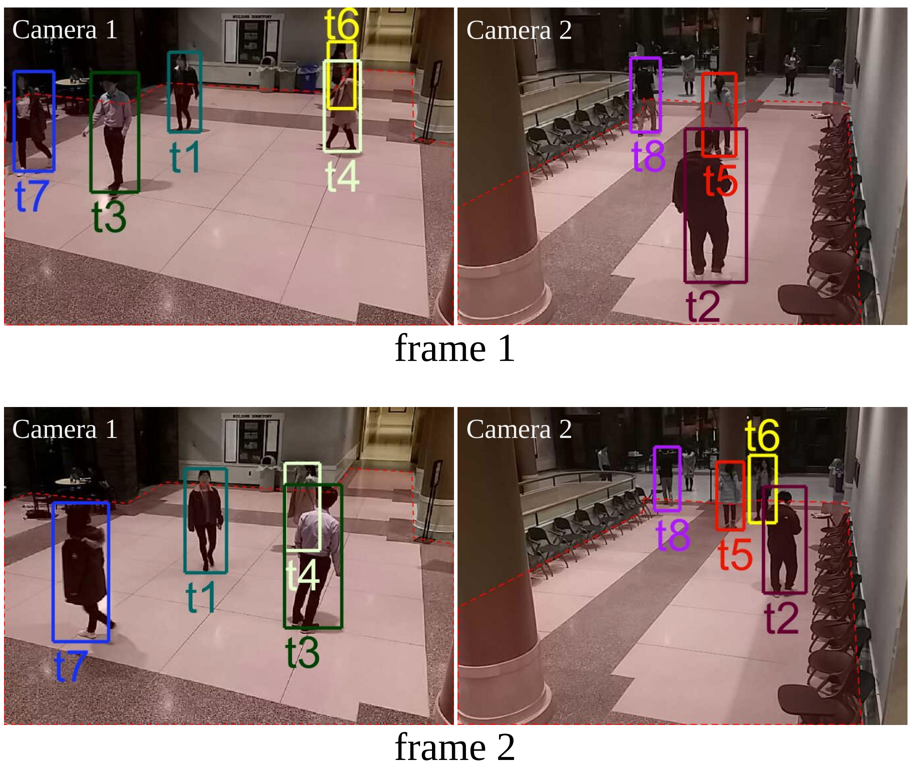


Fig. 2.4. Tracking results in a real scenario. With two cameras cooperatively monitoring a lobby, people in the camera views are labeled with their tracklet IDs. Frame 2 is around 2 seconds after frame 1. t_6 is correctly and stably tracked even if it is occluded or it crosses various camera views and sliding windows.

in cases where a person is not fully visible because of occlusion, but also provides the possibility of analyzing each person's gestures and activities.

Fig. 2.4 illustrates an example of tracking results in our experiments. With two cameras cooperatively monitoring parts of a lobby, people in the camera views are labeled with their tracklet IDs. Frame 2 is around 2 seconds after frame 1. Even if people are occluded or cross various camera views and sliding windows (e.g. t_6 in Fig. 2.4), the tracking results are still correct and stable.

Optimizations for Real Time

Although the above tracking scheme is feasible in offline scenarios, computation speed and tracking accuracy still present critical barriers to realizing it in a real-time system. Optimizations in several aspects must be made in both tracking algorithms and server structure.

Pruning sample space in CFT: The low-level tracklets obtained from associating detection responses in neighboring frames are often fragmented because of missing detections in some frames. Fig. 2.5(a)-(c) show consecutive detection responses which are able to form a low-level tracklet, while Fig. 2.5(d) shows a missing detection in the next frame. Therefore CFT is used to narrow the gaps between fragmented tracklets. The common approach to performing CFT on a frame is to generate a set of samples with randomly disturbed heights around the predicted response by linear motion model [36]. Then a potential extension is chosen from the samples by extracting appearance features (e.g. color histogram, texture [50], and Histogram of Oriented Gradient descriptors [51]) from each sample and comparing the similarity one by one with the existing tracklet.

Because of the intensive computation cost of extracting and comparing appearance features, we prune the sample searching space using two methods. (1) *Height-aware CFT*. Real-world heights of bounding boxes can be estimated using projective geometry [52, 53], we reversely calculate the image-plane height for each sample according to its location and use the determined height for further searching instead of the randomly disturbed ones. The dot array in Fig. 2.5(e) shows centroids of a 9×11 sample space, which has been pruned by the determined heights. The dashed rectangle illustrates a sample with a reasonable and fixed height at $(1, 1)$. (2) *Gradient-search CFT*. One of our empirical observation is that the chosen potential sample often locates close to the center of sample space. Instead of traversing all samples, we enhance searching efficiency by performing gradient search from the central sample at $(5, 6)$ to obtain a local optimum. Fig. 2.5(f) shows a heat map of similarity between the

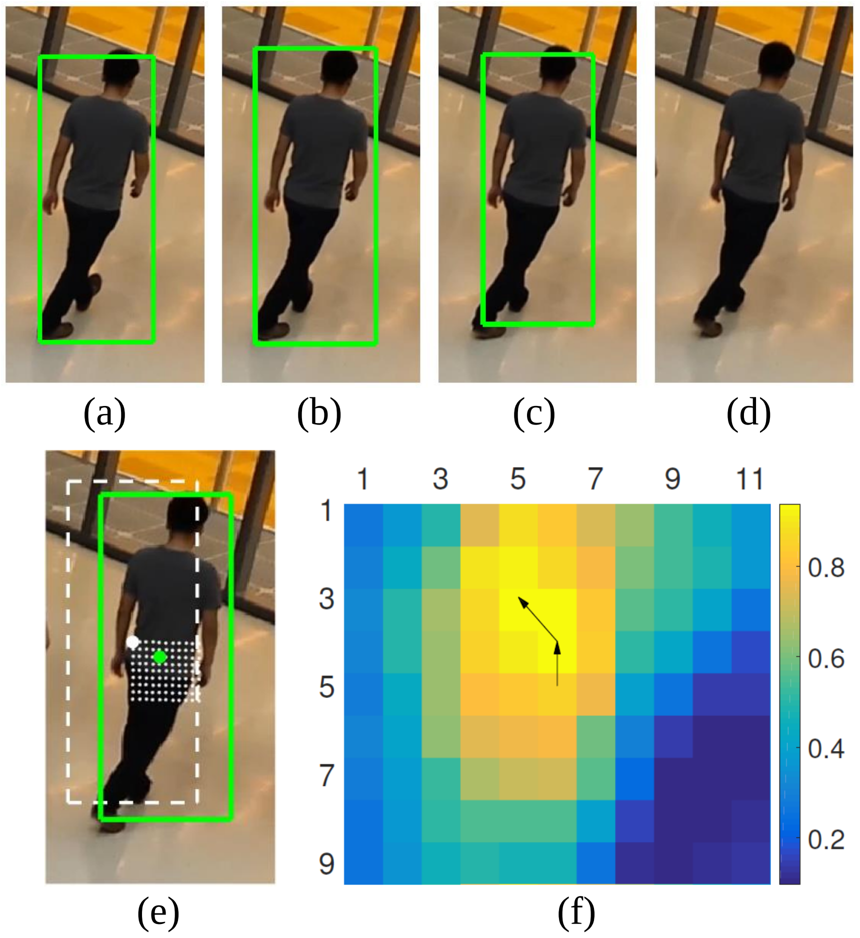


Fig. 2.5. Pruning CFT sample space. (a)-(c) are detection responses on consecutive frames. (d) shows a missing detection on the next frame. (e) shows a height-aware sample space for the frame in (d). (f) shows the similarity heat map for the sample space in (e) and a gradient search path for the local optimal sample.

tracklet and each sample. An example of gradient search path is shown with arrows, in which the optimum sample locates at $(3, 5)$.

Pipelined and parallel scheme: As videos are continuously streamed to the server for tracking, a serial processing scheme cannot well balance multiple tasks including receiving frames, tracking, and communicating with clients (shown in Fig. 2.2).

Therefore, we devise a pipelined and parallel scheme (shown in Fig. 2.6) which uses three types of separated processes to handle different sets of tasks. In the *camera process* (P_{Ci}), pedestrian detection is performed for each new frame on its arrival at the server. The frame is then cached into a buffer together with its detection results. Once the buffer is filled with frames to compose a sliding window (with a total video length of two seconds as set in Sec. 2.3.1), the *tracking process* (P_{Ti}) conducts pedestrian tracking on the cached frames, while the camera process waits for the frames for next window. The tasks on the tracking process include low-level association, CFT, and high-level association, among which CFT occupies most of the computation time. Once pedestrian tracking for the same window has been finished on all tracking processes, the *stitching process* (P_S) merges local tracklets into global ones and then completes rest jobs related to feature processing and packet transmission.

When using pipelining, it is necessary that the computation time of each stage is hard bounded [54]. If pedestrian tracking within a sliding window is not finished before the next window totally arrives, the delay will accumulate over time, resulting in frame loss. Therefore, a hard deadline is set to terminate CFT early if no time is left at this stage, while other tasks can always be finished in a negligible period of time. Within the limited time, CFT is conducted in a token-like manner to equally extend each tracklets.

2.3.2 Motion Extraction

We now describe how we define motion features and how they are extracted from both videos and sensors.

Motion features qualified for matching and comparing between videos and sensors ought to meet the following requirements. First, an efficient motion feature should have high distinguishability, which means that it holds rich diversity among different people and can be used to easily discriminate their walking behaviors. Second, reliable motion features extracted from the two sides need to be consistent, which validates

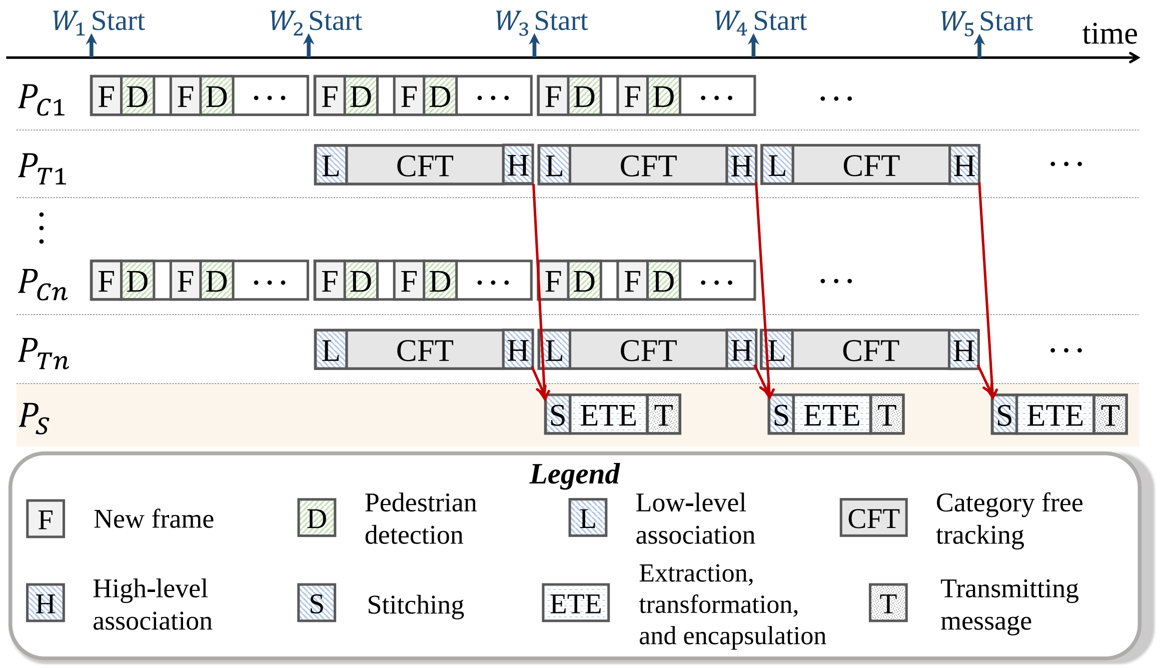
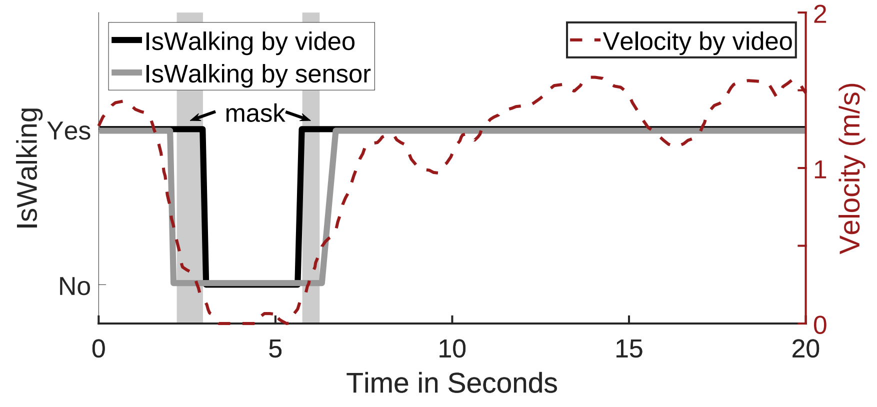


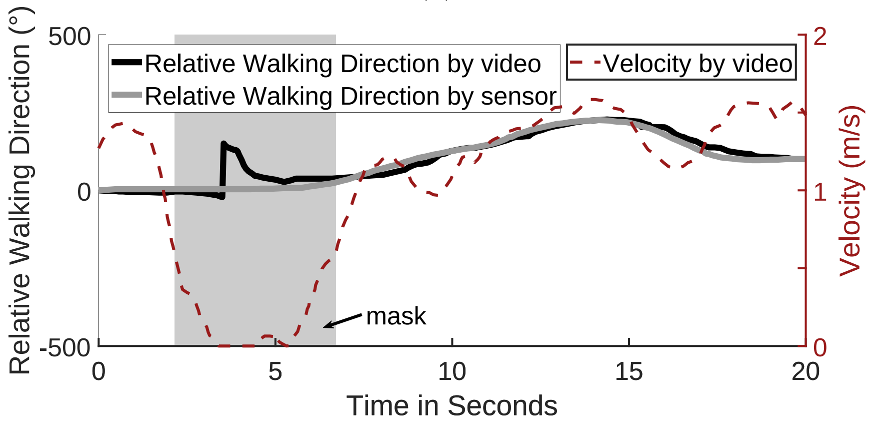
Fig. 2.6. Pipelined and parallel scheme. Three types of separated processes are assigned to work cooperatively on different tasks. P_{Ci} receives videos and performs pedestrian detection on each frame. P_{Ti} performs pedestrian tracking in a sliding window. P_S stitches local tracklets from different videos, processes features and transmits packets.

comparison. Third, due to the concerns about power consumption on smartphones and limited computation time for each stage on the server, features are required to be easily extracted with modest computation cost.

Based on the above requirements, some intuitive options are considered unsuitable for our system. Extracting step-related motion features (e.g. step phase and gait, etc.) from videos suffers from intensive computation cost and therefore poor scalability to a large number of people [55]. In contrast, walking speed can be easily calculated from videos whereas it is challenging to obtain it from sensors. The performance of approaches like integrating accelerometer readings, combining step count with stride length, or feature-based estimating, depends on the orientation and position of the



(a)



(b)

Fig. 2.7. Motion features from video and sensor. (a) IsWalking with masks for transition period. (b) Relative Walking Direction with masks for choppiness.

phone. Also, walking direction attained from the compass is not suitable for many indoor scenarios because the compass itself is susceptible to ferromagnetic interference [56].

Therefore, we choose to utilize two types of motion features named as *IsWalking* and *Relative Walking Direction* for the benefit of simpleness and robustness.

IsWalking: To determine whether a person in the video is walking, we check the velocity of bounding boxes generated by Kalman filter during the tracking process. On the user side, we first project sensor acceleration onto gravity and calculate the variance of the projected values within two seconds. Then we mark IsWalking as "Yes" if the variance is above a pre-set threshold. The decisions on both sides along a period of time are shown in Fig. 2.7(a). However, because the process of starting or stopping walking lasts for a while, a mask on the derived features is added to serve as a cushion against the uncertainty during this transition period (speed is $0.2 \sim 0.5$ m/s). Only features which are not masked are taken into comparison.

Relative Walking Direction: Relative Walking Direction is defined as the direction in which a person moves in reference to his/her initial direction at the beginning of a *motion period*. Relative direction can also be derived from the velocity of bounding boxes without extra computation cost. On the user side, we first project rotation rate onto gravity and then integrate it to get relative rotation. As shown in Fig. 2.7(b), Relative Walking Direction estimated from the video rapidly jumps about 180 degrees and returns back to normal. This choppy direction on the video side is caused by subtle waggles of the human body when the person is actually stationary. Similarly, we cope with this exception with a mask and cover up the direction features when the person is static or moving below a speed threshold (0.6 m/s).

2.3.3 Motion Transformation

Broadcasting packets with raw video motion features attached as communication addresses will cause severe information leaks. By continuously capturing these packets, a hacker can recover people's walking traces and know how they behaved. This does harm in many scenarios, for example, supermarkets or shopping malls definitely don't want consumer behaviors (e.g. where the customers walk, and where they stop to touch or pick up items) to be revealed to their competitors - imagine browsing histories on Amazon are leaked to Netflix. Even worse, this information may be manipulated by criminals for illegal activities like tailing. It's natural to ask: Why not pick some fragments from the raw motion features to use as addresses? One reason is that, although this method partially hides the shape of walking traces, each fragment

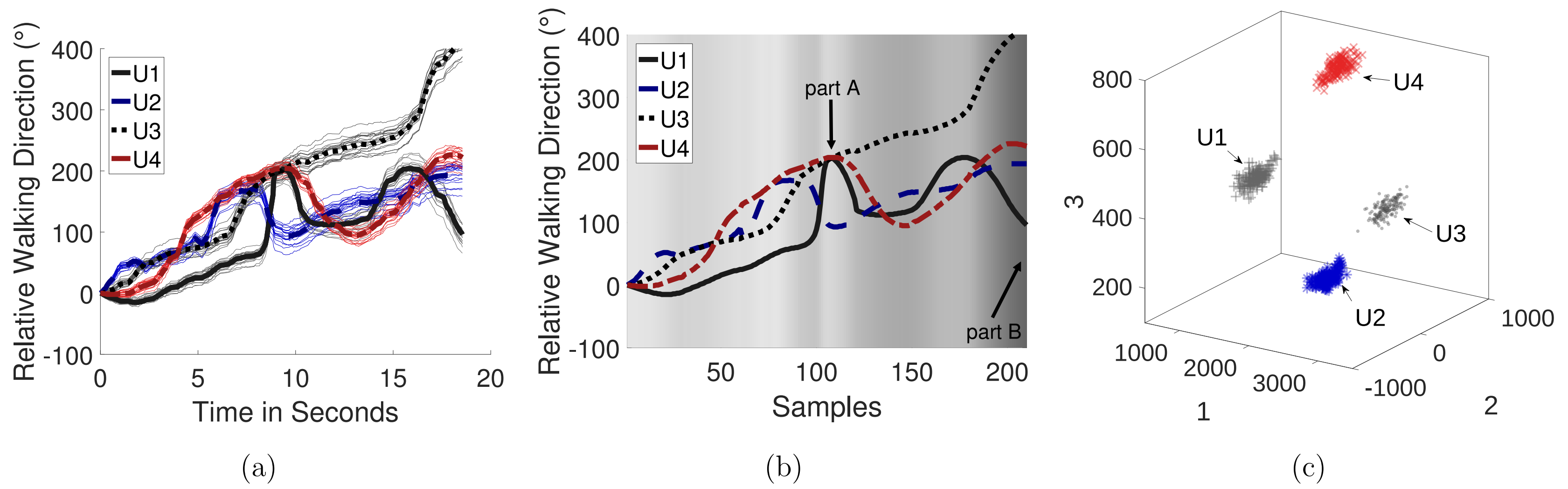


Fig. 2.8. Motion transformation. (a) shows Relative Walking Direction and its simulated samples generated by random walks. (b) shows how PCA emphasizes parts of Relative Walking Direction with large variance. The first principal component is visualized in the background, where darker shade means more emphasis on the corresponding part. Part A is less emphasized due to high similarity between user 1, 3, and 4. Part B is more emphasized due to large diversity among all users. (c) shows how feature vectors with noises are distributed on the K-dimensional space ($K = 3$) after transformation.

still leaks behavior details which may imply important information such as human health conditions [57]. More importantly, searching for an effective subset of fragments is essentially a feature selection problem, which is known as NP-hard [58, 59]. Even the approximate solutions still suffer from high time complexity.

Based on these above thoughts, we choose principal component analysis (PCA) to transform raw motion features to low-dimensional address codes and therefore alleviate motion leaks. As a widely used statistical procedure, PCA aims to compress information by projecting a set of high-dimensional components onto a low-dimensional space and meanwhile maximally reserve the variance of the projected data. PCA is appreciated in our case for two advantages: (1) It highlights the most distinguishable parts among motion features without large computation effort such as model training; (2) The number of principal components (K) can be used as a knob to tune the amount of diversity reserved after transformation, which is traded off against the amount of motion leak.

Before conducting PCA on the video-based motion features, one problem still needs to be solved. Considering that different approaches are used to extract features from videos and sensors, there should be a certain amount of tolerance for minor and inevitable inconsistency. Therefore we add pepper noise and random walks onto IsWalking and Relative Walking Direction, respectively, to generate a set of simulated feature vectors $p_i^j = [f_i^j + n_{i,1}^j, f_i^j + n_{i,2}^j, ..., f_i^j + n_{i,R}^j]$ for each feature vector f_i^j, where R is the number of simulated feature vectors and $n_{i,r}^j$ is the rth noise. Simulated features of type j for all users are denoted as $P^j = [p_1^j, ..., p_N^j]$, where N is the number of users. We illustrate P^j which is generated by adding random walks to Relative Walking Direction in Fig. 2.8(a) and omit pepper noise in the interest of space. Note that, for the features covered up by masks, PCA treats them as missing data and runs the standard process [60].

Now PCA runs on P^j and generates a $K \times L_j$ coefficient matrix $Coeff^j$, where L_j is the feature vector length of type j. Fig. 2.8(b) is an intuitive illustration of how PCA emphasizes parts of motion features with significant diversity. The first principal component is visualized in the background, where darker shade means more emphasis on the corresponding part. Note that part A is less emphasized due to the high similarity among Relative Walking Direction of user 1, 3 and 4. In contrast, part B is more emphasized due to the large diversity. After the PCA transformation, P^j is converted into K-dimensional codes $H^j = [h_1^j, ..., h_N^j]$, where $h_i^j = Coeff^j * p_i^j$. Fig. 2.8(c) illustrates how H^j is distributed in the K-dimensional space. Four groups representing h_i^j $(i = 1, 2, 3, 4)$ are well separated even though we just use the first three principal components $(K = 3)$. Note that the address codes v_i^j which will be actually used in packets are transformed from the original feature vectors without noise via Equation 2.1.

$$v_i^j = Coeff^j * f_i^j \tag{2.1}$$

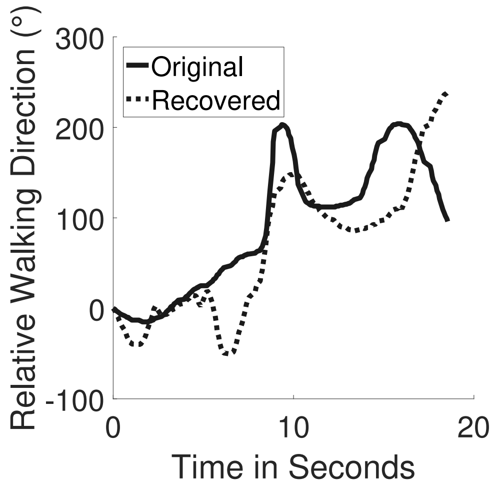
Recall that $Coeff^j$ is sent in packets to ensure the same transformation on the sensor side (See Fig. 2.2). The original motion feature can be partially recovered to f'^j_i via Equation 2.2.

$$f'^j_i = (Coeff^j)^T * v_i^j \tag{2.2}$$

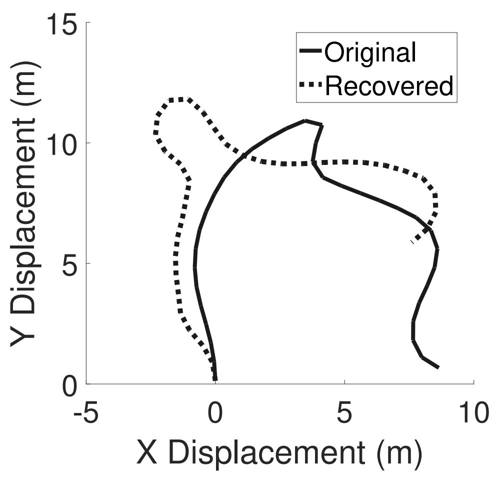
To demonstrate how much motion information can be hidden by the transformation, we choose user 1 as an example and show how much Relative Walking Direction and the walking trace can be recovered from her address code. Although the partial trend is reserved in the recovered feature vector (shown in Fig. 2.9(a)), locations within the trace cannot be precisely reproduced (shown in Fig. 2.9(b)). This largely prevents motion leaks and avoids the walking history and details being inferred from the addresses. We will elaborate more detailed evaluation later in Sec. 2.4.2. By decreasing K, more motion information can be hidden but meanwhile this may affect the matching performance.

2.3.4 Packet Encapsulation

Now, the server organizes all data to broadcast into a uniform format. As shown in Fig. 2.10, each packet is labeled with an application ID, which represents the function



(a)



(b)

Fig. 2.9. Motion and trace recovery. Using PCA ($K = 3$) on user 1's features, (a) the feature vectors (e.g. Relative Walking Direction), and (b) the traces cannot be precisely recovered.

of customized messages. Since the server is aware of current network conditions, it specifies a Time Shift Range to let the clients search for corresponding sensor readings with tolerance for various network delays. In each model generated for the feature of type j, a series of feature timestamps are shared among all users to extract corresponding motion features. PCA Coefficient Matrix and other optional parameters (e.g. matching thresholds) are also sent as parts of the model. In the field for each user i, a pair of address code v_i^j and mask m_i^j is included for each model j.

2.3.5 Multi-Level Decision

Once receiving a broadcast message, the user will first extract corresponding sensor-based motion features for each model j according to the received feature timestamps by using the methods introduced in Sec. 2.3.2. The sensing motion feature g^j is then transformed into sensing address code s^j by duplicating the same transformation process via PCA (Equation 2.3).

$$s^j = Coeff^j * g^j \tag{2.3}$$

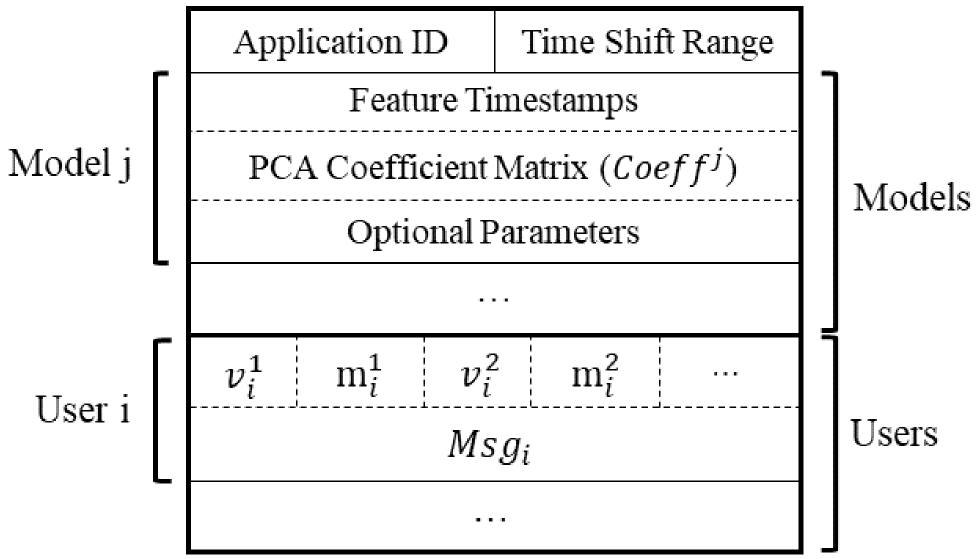


Fig. 2.10. Packet format. The packet is composed of application ID, Time Shift Range regarding the network delay, model parameters, and fields (e.g. address codes, masks and messages) for users.

Then the tuples $< s^1, s^2, ... >$ and $< v_i^1, v_i^2, ... >$ are hierarchically compared for all received Msg_i. The similarity between each pair is measured by their Euclidean distance. The candidate message Msg_i is passed to the next decision level if the similarity is higher than a certain threshold. The multi-level decider finally comes up with a decision of whether to accept each received message. After going through all matching levels, if only a single video address code fulfills all the threshold requirements, the decider will convey the corresponding message to the application level for further usage. Otherwise, the decider will generate a final decision of "Unsure" and discard all received messages.

2.4 Evaluation

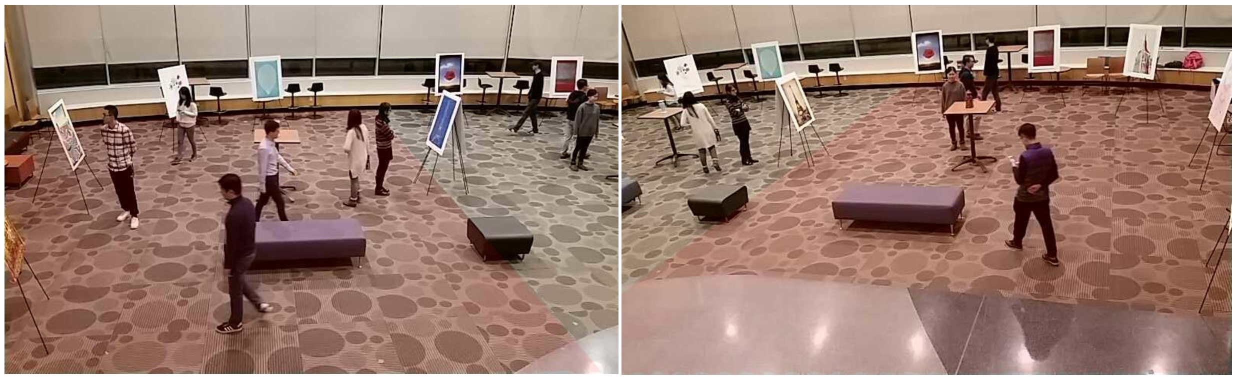
This section discusses the experiment methodology and performance results of *PHADE*.

2.4.1 Implementation and Methodology

The client side of *PHADE* is implemented on the Samsung Galaxy S4 smartphone, which logs accelerometer, gyroscope, and gravity readings at $100Hz$. The smartphone runs our *PHADE* App to match with address codes and receive application-customized messages in real time. We set up a server using two PCs with dual NVIDIA GTX 1080 Ti SLI, and run MATLAB and C++ programs on each. Two Samsung Galaxy S5 smartphones are used as IP cameras to record and continuously stream videos to the server. The video is recorded at a frame rate of 15 fps, a bit rate of 2000 kbps, and a resolution of 800×480 [1]. Wi-Fi is used for video streaming and message transmission. And the number of principal components K is set to 3.

PHADE is evaluated via real-life experiments with 17 volunteers in three different university lobbies (A, B, and C) which covers around $192m^2$, $100m^2$, and $270m^2$ respectively. The experiments were executed in five sessions. (1) We arranged some paintings and sculptures in lobby A and set it as a mock museum. The volunteers put a phone in pocket, and naturally walk and pause as they pleased. Five times of 10-minute experiment are conducted with 2, 4, 6, 8, and 10 users in the scene. (2) 6 users walked in lobby B with the phones put in their pockets. (3) A similar experiment was conducted in lobby C with four users, where the phone is put in each volunteer's coat or pant pocket. (4) We pre-labeled ten points on the ground of lobby C and set the minimum distance between two points to 0.5 m. Then two volunteers were asked to walk and deliberately pause at these points. With the phones in their pockets, the volunteers wore earphones to listen to audio clips representing each point. This is to mimic an application of *PHADE* as an automatic audio guide. (5) Three people walked in lobby A, pretending the left wall, the right wall and the roof of the mock museum have one mural on each of them. The users could point at any of these three sides and would expect to receive messages about the corresponding mural. Each session last for 50, 8, 10, 5, and 10 minutes, respectively.

[1]These video settings are based on a trade-off between video processing speed and tracking accuracy. We omit the details in the interest of space.


(a)

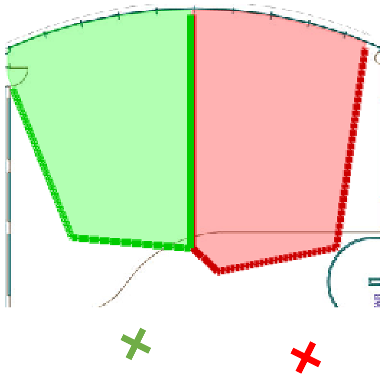

(b)

Fig. 2.11. Experiment scenario. (a) shows a pair of example frames with ten users walking in lobby A. (b) shows areas cooperatively covered by two cameras. where each camera is marked with a cross.

2.4.2 Performance Results

The following questions are of our interests:

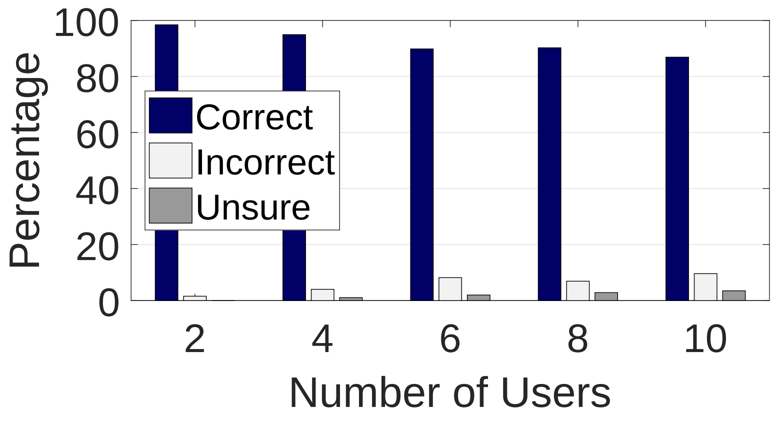
- How well does *PHADE* perform overall?
- How does *PHADE* achieve real-time tracking?
- How much is motion history blurred?
- How does *PHADE* perform in applications?

(1) How well does *PHADE* perform overall?

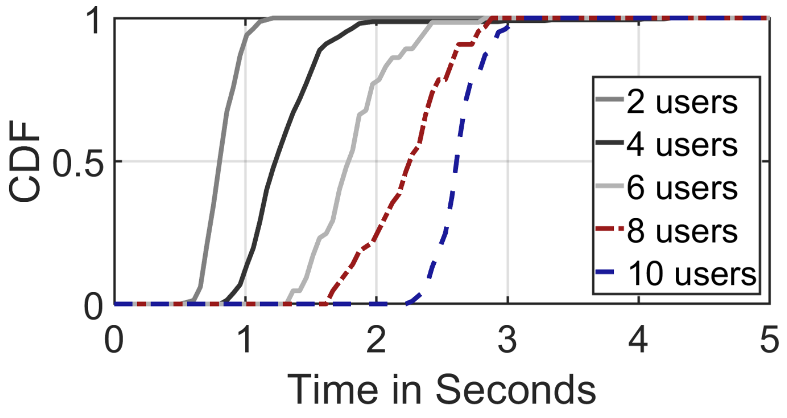
Here we use the first experiment session to demonstrate the overall performance with different numbers of users. Tracklet IDs generated from the latest sliding window are sent right away in a message. Fig. 2.11(a) illustrates the example frames with ten users in lobby A. Fig. 2.11(b) shows the covered area of each camera with shades and the camera positions with crosses.

By comparing tracklet IDs from accepted messages, we obtain the matching performance, shown in Fig. 2.12(a). Our system achieves 98%, 95%, 90%, 90%, 87% matching correctness for 2, 4, 6, 8 and 10 users respectively using 18 seconds of motion features. The performance is degraded as the number of users increases because the occlusion between people happens more frequently as there are more users walking in the limited area. This can be mitigated by put the cameras higher or on the ceiling.

Fig. 2.12(b) plots the CDF of computation time for each sliding window on the server side. The computation time on the server contains the time cost for tracking, motion extraction, motion transformation, and encapsulation. Note that the time is bounded around 3 seconds, which is caused by the hard boundary for each stage in the pipeline. On the client side, the computation time is less than 0.4 seconds at 99 percentile.



(a)



(b)

Fig. 2.12. (a)Matching performance. Occlusion partially affects the matching results as the number of users in a limited experiment area increases. (b) Computation time on server. The time is bounded around 3 seconds, which is caused by the hard boundary for each stage in the pipeline.

(2) How does *PHADE* achieve real-time tracking?

Based on the video captured in the second session, we evaluate how the tracking scheme is optimized and accelerated with the techniques described in Sec. 2.3.1. Evidently, compared with processing videos from multiple cameras one by one, the parallel design guarantees the tracking process to be linearly sped up. So we use the video captured by only one camera in the second experiment session for fairness.

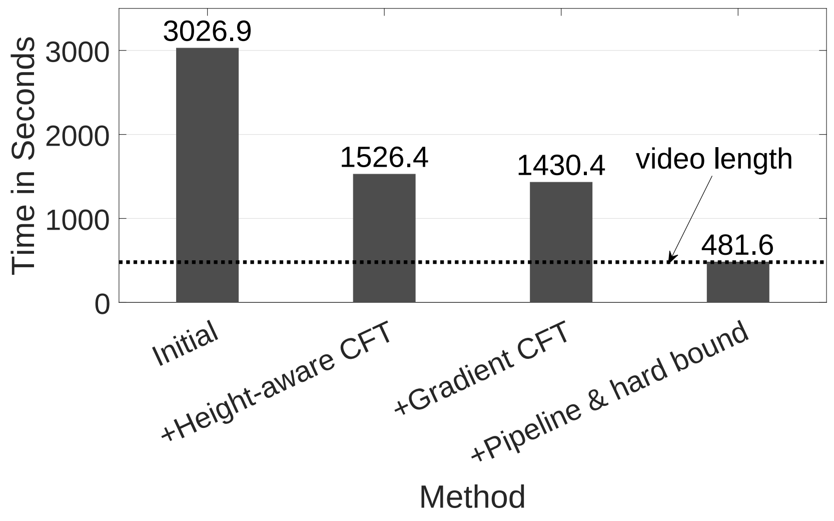
Fig. 2.13(a) and Fig. 2.13(b) show how much the tracking time is compressed and how the tracking performance changes while incrementally adding each technique. The height-aware CFT benefits the tracking performance regarding precision and the number of ID switches. And all the optimizations jointly contribute to accelerating the tracking scheme and make the system real-time. Originally, the delay of generating tracking results is accumulated as the video length increases. Now the tracking delay for the last sliding window is shortened to a constant value of around 2 seconds. One observation is that adding the hard bound to each stage increases the number of ID switches a bit. However, compared with its improvement to the computation time, it's still necessary to use the pipeline.

(3) How much is motion history blurred?

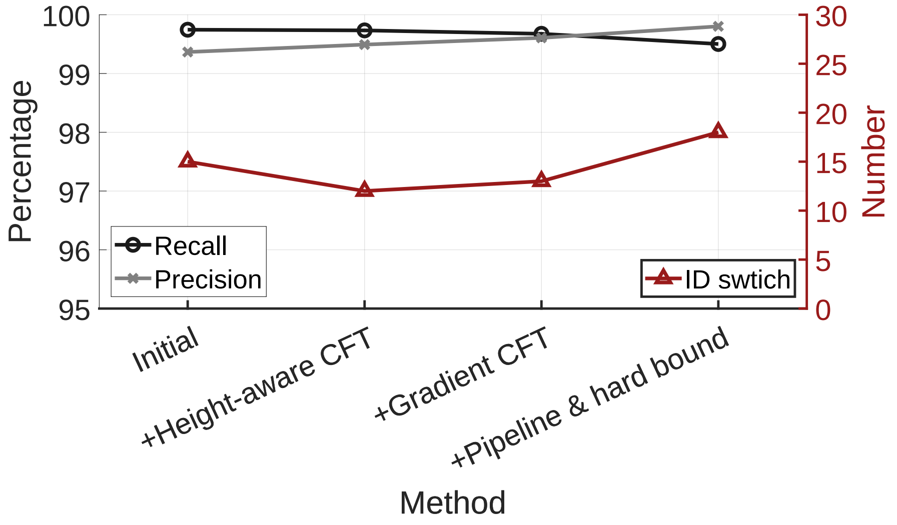
Here we assume that a hacker is listening to broadcast packets. Combining this motion features with the current position and walking direction of a user, the hacker is able to trace back the user's walking history.

We first manually label the ground truth positions of users in session three. Walking traces are recovered from one of the following three types of motion features with various motion periods: (1) velocity (speed and corresponding direction); (2) raw motion features (i.e. IsWalking, Relative Walking Direction); (3) address codes after transformation by PCA. The distance between the starting point of a recovered trace and the ground truth is used as the metrics to measure the amount of motion leak.

These distances are statistically shown in Fig. 2.14(a) for the recovered traces tracing back to various time points. When trying to recover traces back to 5 minutes ago, the above three methods generate the starting points with a median distance of 0.21 m, 2.7 m, and 13.7 m from the ground truth. Since the velocity from the video is computed from each frame via the Kalman filter, it can always precisely recover

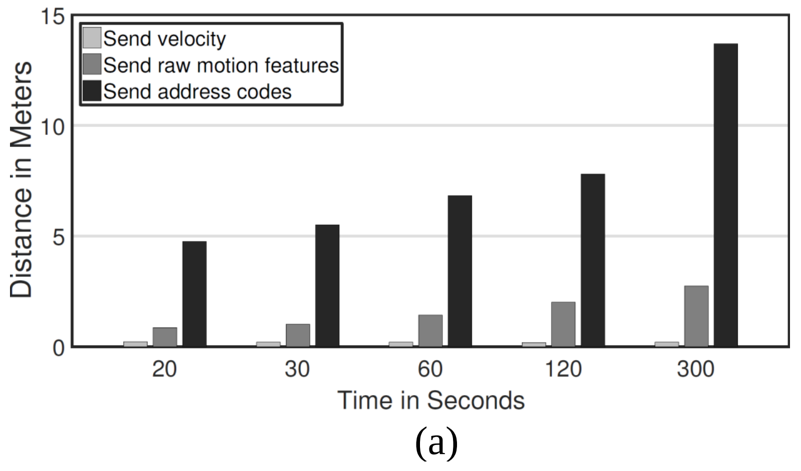


(a)



(b)

Fig. 2.13. Improvements in tracking time and performance. (a) shows that the tracking delay is shortened to a constant time. (b) shows that the tracking performance while incrementally adding the optimization techniques.



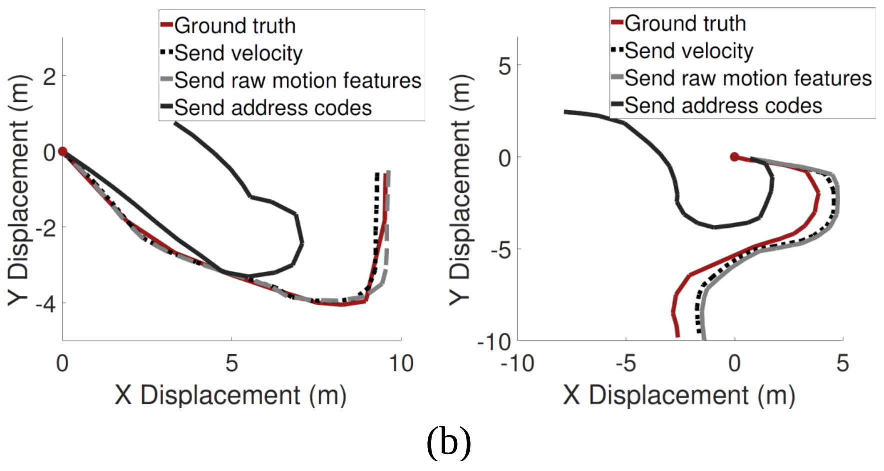


Fig. 2.14. Motion blurring. (a) shows the distances between the starting point of a recovered trace and the ground truth largely increase when using transformed address code, which implies modest motion leak in *PHADE*. (b) shows two examples of recovered traces using various motion features.

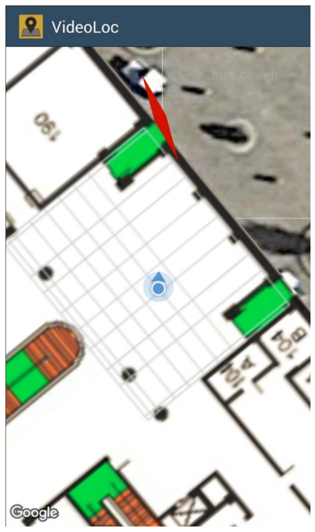
traces with a constant distance from the ground truth and thus causes severe motion leaks. Considering raw motion features, Relative Walking Direction is integrated with a fixed average walking speed of 1.5 m/s [61] during the period when a user is walking (indicated by IsWalking). Motion is still largely leaked due to the high precision of video-based Relative Walking Direction. However, the distance from ground truth rises to a median of 13.7 m when using transformed address codes, which means the hacker can hardly infer the walking histories.

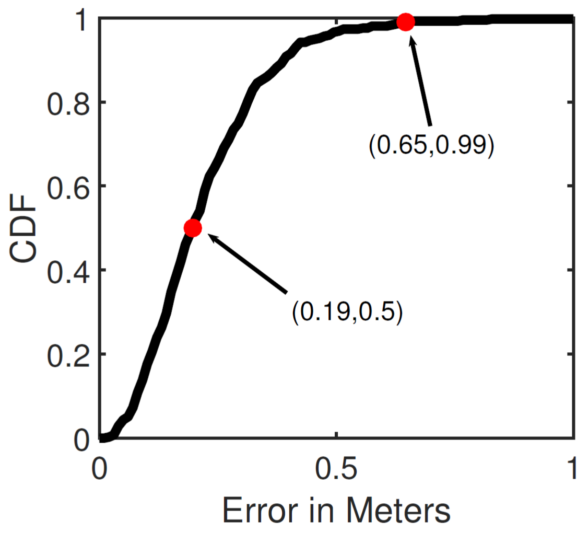
Fig. 2.14(b) shows two examples of 20-second recovered traces, which are derived from the above three motion features, comparing with the ground truth. The recovered traces are greatly distorted when using address codes after transformation. This implies a modest motion leak in *PHADE*.

(4) How does *PHADE* perform in applications?

Indoor localization. Simply using location information obtained from the video [62,63] as the customized messages, *PHADE* can be easily adopted into a localization application. Fig. 2.15(a) illustrates the app we implemented to receive and show real-time locations for the phone user. Fig. 2.15(b) shows the localization error with four users walking in lobby C. The median error is 0.19 m and the error is 0.65 m at 99 percentile, which makes *PHADE* amenable to most location-based services [64, 65]. The localization accuracy is barely affected by the number of users since the locations don't rely on any wireless signal.

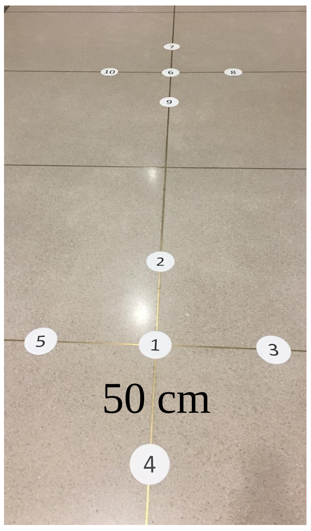
Automatic audio guide. We also evaluate *PHADE* when it is used as an automatic audio guide. Fig. 2.16(a) shows how we set up for the experiment, where the minimum distance between two points is 0.5 m. Fig. 2.16(b) reports the confusion matrix for those ten points. The point matching is accurate in 99.7% cases, which implies that our system is capable of distinguishing exhibits which are immediately close to each other. This advantages *PHADE* over other schemes such as Bluetooth.

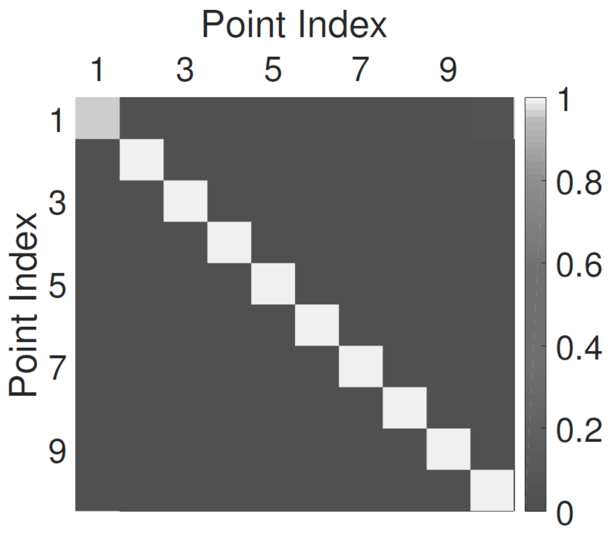




(a) (b)

Fig. 2.15. Indoor localization. (a) The app with user position and direction on the map. (b) The localization errors.





(a) (b)

Fig. 2.16. Automatic audio guide. (a) Ten points setup for audio guide experiments. (b) The confusion matrix for ten pre-labeled points.

Gesture-based messaging. In some scenarios such as requesting information about unreachable displays, simply pointing at it to send the request may be a good alternative to searching for it with its name or index. To demonstrate an example of context-aware messaging, we set up an experiment in which users imagined that there were murals on the walls and they could point at them to get introductions. We detect three gestures (i.e. pointing to the left wall, the right wall, or the roof) using the body parts generated by OpenPose, and send customized messages to the user according to the "mural" that it is pointing at. Two examples of detected gestures are illustrated in Fig. 2.17(a) and (b). Taking the direction of gravity as the reference direction, these three different gestures are classified according to the user's arm angles. The angle ranges for pointing left, right and up are set to $[-135, -80)$, $(80, 135]$ and $(-135, -180] \cup (135, 180]$, respectively. In Fig. 2.17(c), the dashed boxes show three examples of pointing gestures and the corresponding arm angles over time. Here we are just presenting a proof of concept and don't claim a contribution to that. The detection performance could be improved with more complex algorithms. Table 1 and Table 2 show the confusion matrix for gesture detection results and message receiving performance. Except for the cases of misdetection, most gesture-based messages are received correctly. And the message sending delay (from when the gesture starts until the corresponding message is sent out) is shown in Fig. 2.17(d), where the median sending delay is 3.2 seconds and it's 4.6 seconds at 97 percentile.

2.5 Discussion

In this section, we discuss several limitations and untapped opportunities with *PHADE*.

Similar motion patterns. While *PHADE* works in most cases, one exception is when two or more users are walking in similar patterns. One opportunity is to dynamically decide whether to extract more sophisticated motion features (such as step phase) to increase distinguishability. Note that real-time human pose estimation has

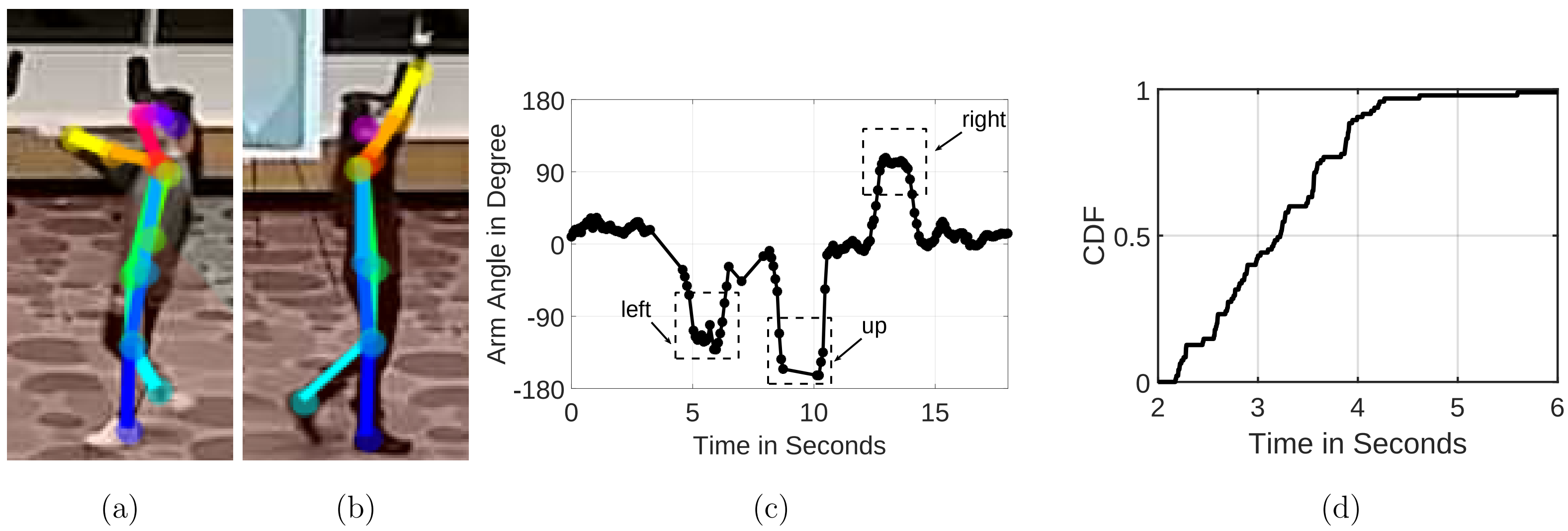


Fig. 2.17. Gesture-based messaging. (a) Pose when pointing to the left wall. (b) Pose when pointing to the roof. (c) shows examples of arm angles during each type of pointing gesture. The three dashed boxes represent pointing to the left wall, the roof and the right wall, respectively. (d) The time intervals since the pointing gesture starts to the corresponding message is sent.

Table 2.1.
The confusion matrix for detected pointing gestures.

	up	left	right	others
up	42	6	5	4
left	0	32	0	6
right	1	0	42	0
others	4	4	1	433

Table 2.2.
The confusion matrix for received gesture-based messages.

	up	left	right	others
up	38	6	5	7
left	0	26	1	10
right	2	1	38	3
others	6	3	2	432

been accomplished by [33], it's feasible to extract more fine-grained motion features as addresses. However, this trades off the timeliness of message delivery for matching accuracy.

Cooperation with sensors. *PHADE* requires no sensor data from users and protects user privacy from this aspect. However, if there are some volunteers willing to upload their sensor data to the server, it is feasible to build a digital map with ambient sensing data, e.g. magnetic fluctuation. Seeing a targeted user walking across certain positions, the camera may conjecture how the sensing patterns may look like on this user's smartphone and uses them as "sensing addresses". In addition to the motion addresses that we currently adopt, these sensing addresses can improve the distinguishability among users.

Message encryption. To protect user privacy, *PHADE* applies PCA transformation to partially hide walking behaviors from the public. However, users may still feel uncomfortable as the messages for them are broadcast to the public. To cope with this concern, one possible solution is to design a symmetric key to encrypt the messages. Note that although the blurred motion features have been broadcast, the rest part is kept as a private and shared knowledge between the server and the client, which may be suitable as a symmetric key. We leave this to our future work.

2.6 Related Work

Camera-based communication. Traditionally, cameras are used as a receiver for information in visual communication. For example, HiLight [66] encodes data into pixel translucency changes atop any screen content to realize real-time screen-camera communication. InFrame++ [67] enables simultaneous communication for both users and devices on full-frame video contents. ARTcode [68] preserves both image and code features in one visual pattern. [69] creates a model of a screen-to-camera communication system to predict the information capacity based on the receiver perspective. In contrast, *PHADE* enables cameras to talk to users, which is in a reversed direction.

Motion information leaks. Recently, motion leakage is brought to researchers' attention. PowerSpy [70] shows that aggregate power consumption implies user's location. MoLe [17] leverages the pattern in English words to infer what a user is typing on the keyboard. [71] uses embedded sensors on wearable devices to capture inputs on ATM keypads. [72] studies the privacy bound of human mobility and reports that four spatial-temporal points are enough to identify 95% of individuals. While *PHADE* demonstrates the motion leak from videos and proposes PCA transformation as a solution.

Camera sensor fusion. Several works exist which utilize a fusion of camera and mobile sensors with a wide variety of applications. Overlay [73] uses a combination of the smartphone camera and various sensors to build a geometric representation of an environment to enable augmented reality on the phone. Gabriel [74] employs image capturing and mobile sensing to develop a cognitive assistance system. Authors in [75] have used smartphone's motion and light sensors together with the camera to allow enhanced biometric authentication on phones through facial recognition. Compared with these prior works in which cameras and sensors are always on the same device and complementary to each other in the same task, our work introduces the novel concept of using sensors to allow communication between the camera and people in the camera view.

ID association. A key contribution of our work is the ability to identify and associate individuals in the camera view with their smartphones. Some schemes use spatial location information as an identifier for mobile devices. For example, Tracko [76] tracks the relative 3D locations between multiple devices by fusing Bluetooth low energy signals, inaudible stereo sounds and inertial sensors, and uses these locations as the destination identifiers to send data. Compared with Tracko, *PHADE* uses human motion patterns as addresses instead of spatial information and is not restricted by the spreading range of Bluetooth signal, which makes *PHADE* more suitable for communicating with moving people. Other schemes for user ID association within an environment exist which use various techniques and devices for identification. ID-Match [26] uses both RFID tags worn by people and a 3D depth camera to recognize and assign IDs to individuals. In [77], RFID and BLE are combined with a stereo-based identification system to recognize individuals in outdoor environments. For such approaches, the identification relies on BLE beacons or the users wearing RFID tags. This makes them infeasible for public areas with a large number of people since many of them might not be carrying previously registered tags. Moreover, if a user switches its tag with another user, these systems will not be able to associate the IDs correctly. Our system, on the other hand, temporarily associates a user in the camera view with its smartphone and requires no preregistration. The use of motion addresses allows it to work in public areas. And it can still correctly identify an individual even if she changes her phone. Insight [27] recognizes people through their motion patterns and clothing colors serving as a temporary fingerprint for an individual. [29] has developed an ID matching algorithm for associating people, detected by the camera, to the accelerometer readings from a sensor worn on their belts. However, none of [27, 29] is implemented into a real-time end-to-end system. Besides, these schemes require users to upload their sensor data. In comparison, this chapter presents a system in which the individual identifying process is not dependent on uploading data. Moreover, our system uses transforms raw motion features to blur behavior details thus protects user privacy.

2.7 Conclusion

This chapter proposes a problem called Private Human Addressing and develops a fully operational real-time prototype named *PHADE* to solve this problem. Without knowing users' smartphone addresses, *PHADE* is able to communicate with them relying on the motion patterns captured by cameras and using these patterns as destination addresses. *PHADE* transforms the raw patterns using principal component analysis to diminish motion details and meanwhile preserves their distinguishable characteristics. The smartphones then locally make their own decisions on whether to accept a message or not. *PHADE* achieves reasonable address matching performance and also provides privacy protection mechanisms to prevent motion leaks.

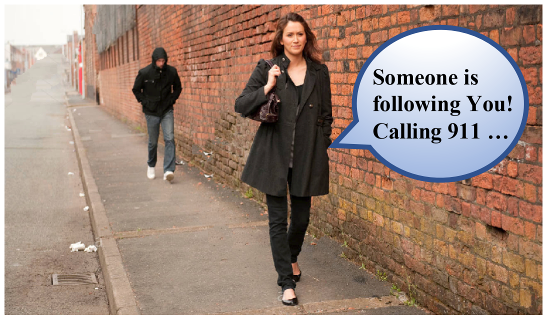
3. TOWARDS CONTEXT ADDRESS FOR CAMERA-TO-HUMAN COMMUNICATION

3.1 Introduction

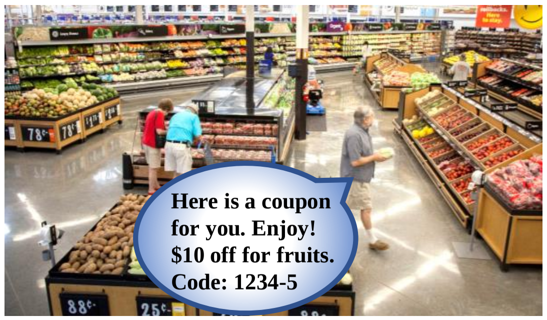
Video analytics has enabled widespread applications ranging from security surveillance to business intelligence [25]. Although with existing analytic algorithms, a camera can identify and track people under surveillance, its potential is not fully explored without any direct communication from the camera to people. Such communication requires an affiliation between a person and its phone, which serves as the person's unique identity for personalized message delivery from the camera. In this chapter, we aim at solving the problem of digitally associating people in the camera view with their smartphones without knowing the phones' IP/MAC addresses.

The capability of sending customized messages to a specific person in a camera view can intelligently enhance public safety and daily life quality. Imagine a person on a street is being followed by someone with a suspicious behavior (shown in Figure 3.1(a)). Potential crimes can be prevented by informing the person about the threat. As shown in Figure 3.1(b), retailers like Walmart, Target, etc. can improve customers' experience by delivering targeted ads and coupons in real time, according to their interests or in-store behavior. Similarly, museums or galleries (Figure 3.1(c)) can provide an interactive experience to visitors by introducing interesting facts relevant to the exhibits of their interests, e.g. when a visitor points to an exhibit, through customized messages. Despite having an operator (be it a human or AI agent) monitoring the surveillance feed, the aforementioned benefits can happen only if a person can receive messages from the camera.

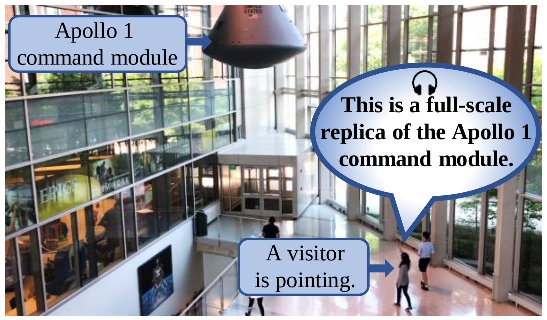
One may argue: Why not simply ask people to register with a face photo and then employ face recognition on the surveillance video? The main reason is that faces



(a)


Here is a coupon for you. Enjoy! $10 off for fruits. Code: 1234-5

(b)


Apollo 1 command module
This is a full-scale replica of the Apollo 1 command module.
A visitor is pointing.

(c)

Fig. 3.1. Application scenarios. (a) Send alerts to a pedestrian about potential threats. (b) Deliver ads or coupons to a customer based on his in-store shopping behavior. (c) Introduce an exhibit to a visitor when she points to it.

are not always visible due to facing direction or limited camera resolution. Moreover, many people express discomfort with uploading their profile photos, given that it may become their permanent identifier [78]. Some cities even banned the use of face recognition [79]. Another way of targeted message delivery is to add short-range communication links (e.g. acoustics, light, and Bluetooth 5.1) and send messages once people approach a beacon. The deployment and maintenance of the beacons are costly. Also, message sending is simply triggered by relative distances and is not related to any contextual information (e.g. a person's behavior or surrounding events), which means these methods are not able to pinpoint an individual in a group.

Prior works have explored some schemes for human ID association involving cameras and sensors. ID-Match [80] requires wearing RFID tags and assigns a unique ID to each individual in a Kinect camera view. [29] associates people in a camera view with accelerometer readings from sensors worn on their belts. Insight [27] demonstrates that a person can be recognized using its motion patterns and clothing colors as a temporary fingerprint. However, [29,80] require extra hardware and [27,80] need the users to register beforehand. Also, neither of [27,29] implements a real-time system for applications requiring direct camera-to-human communication. PHADE [81] uses walking behavior as people's temporary communication address, suffering from large packet overhead since it transmits large coefficient matrices. TAR [82] uses Bluetooth proximity sensing to associate IDs and deliver targeted advertisements in retailers. It requires Bluetooth on all users' smartphones to be on and continuously broadcast BLE signals, which raises severe privacy concerns. Moreover, [29,82] rely on a single type of feature which is not suitable for various scenarios.

The key idea of our work is enabling camera-to-human communication using a person's *context features* as its address. The context address consists of two types of features: (1) *motion features*, e.g. walking velocity; and (2) *ambience features*, e.g. magnetic trend and Wi-Fi signal strengths in user's trajectory history. This chapter pursues to utilize the diversity in these context features as well as the consistency within these features and mobile sensor data. We design an addressing scheme such that on the server side, based on pedestrian tracking results and ambient sensing maps (containing magnetometer and Wi-Fi data), the context features are determined for each individual in the region covered by cameras. Among these extracted features, the ones that maximize the differentiation between the target individual and the rest of the people are selected to serve as the target's context address. This context address is compressed and added as a new header in the application layer and is used to determine the destination of the packet. The server then broadcasts the packet. On the client side, upon receiving a broadcast packet, a user's phone generates corresponding features from its sensor data and compares them with the context address in the packet. If the matching score is above a threshold, the message is indeed targeted for that particular user and is relayed to the application on this phone.

When translating this idea into a functional system, we face three challenges. (1) Defining context features is nontrivial since they need to maintain both distinguishability among a group of people and some tolerance to inevitable signal noises. (2) It is hard to build and update ambient sensing maps efficiently. Simple methods, like war-driving, are not feasible due to extra and repeated human effort. (3) Selecting an optimal set of features that is discriminative and with a limited payload overhead is challenging. Ideally, these features should fit into a fixed-length header, without affecting the space available for data within a normal packet.

This chapter tackles these challenges one step at a time. We present an end-to-end real-time system for camera-to-human communication based on context address, using Google Pixel XL as clients and Samsung Galaxy S5 as IP cameras. The server is designed in a pipelined and parallel manner, running on three PCs with dual NVIDIA GTX 1080 Ti SLI. We evaluate the utility and accuracy of context-based addressing from a real-world experiment in a mimic art gallery. Messages are broadcast to ten users with a sending ratio of 98.5%, an acceptance precision of 93.4%, and a recall of 98.3%. A simulated experiment in a retail store shows that the system is also feasible when scaling to a practical scenario with denser people (about 50) and more complicated environments. The context address header is always compressed to under 40 bytes, same as the length of an IPv6 header.

The main contributions are summarized below:

- Develop a novel context-based addressing scheme for camera-to-human communication, using motion and ambience features. This enables discriminating an individual from a group and sending targeted messages to it.
- Define noise-robust ambience features and an effortless way to generate ambient sensing maps with no war-driving.
- Introduce an effective context selection algorithm to dynamically choose discriminative and low-cost features.
- Implement and evaluate our system in both real-world and simulated experiments. It runs in real time and achieves high performance.

3.2 System Overview

Figure 3.2 depicts an overview of our system. Multiple cameras continuously monitor a public area and stream the video feed to a server. Upon receiving a video frame, the server conducts pedestrian detection [33, 34, 48, 49] and stores the frame with its detection responses into a buffer. Once enough frames are accumulated, the detection responses in consecutive frames are associated into tracklets, each representing an individual in the camera view.

Context features are then extracted for each person from these tracklets. Each feature is either based on the person's motion pattern (e.g. whether it is walking or not at a certain timestamp) or ambience (e.g. magnetic trend in its trajectory history). The motion pattern can be directly generated from the visual tracklets, while ambience relies on maps built from magnetometer reading in gravity direction and Wi-Fi data contributed by volunteer users. Dependent on the target of the application-specific message, the context features, which can effectively distinguish the target

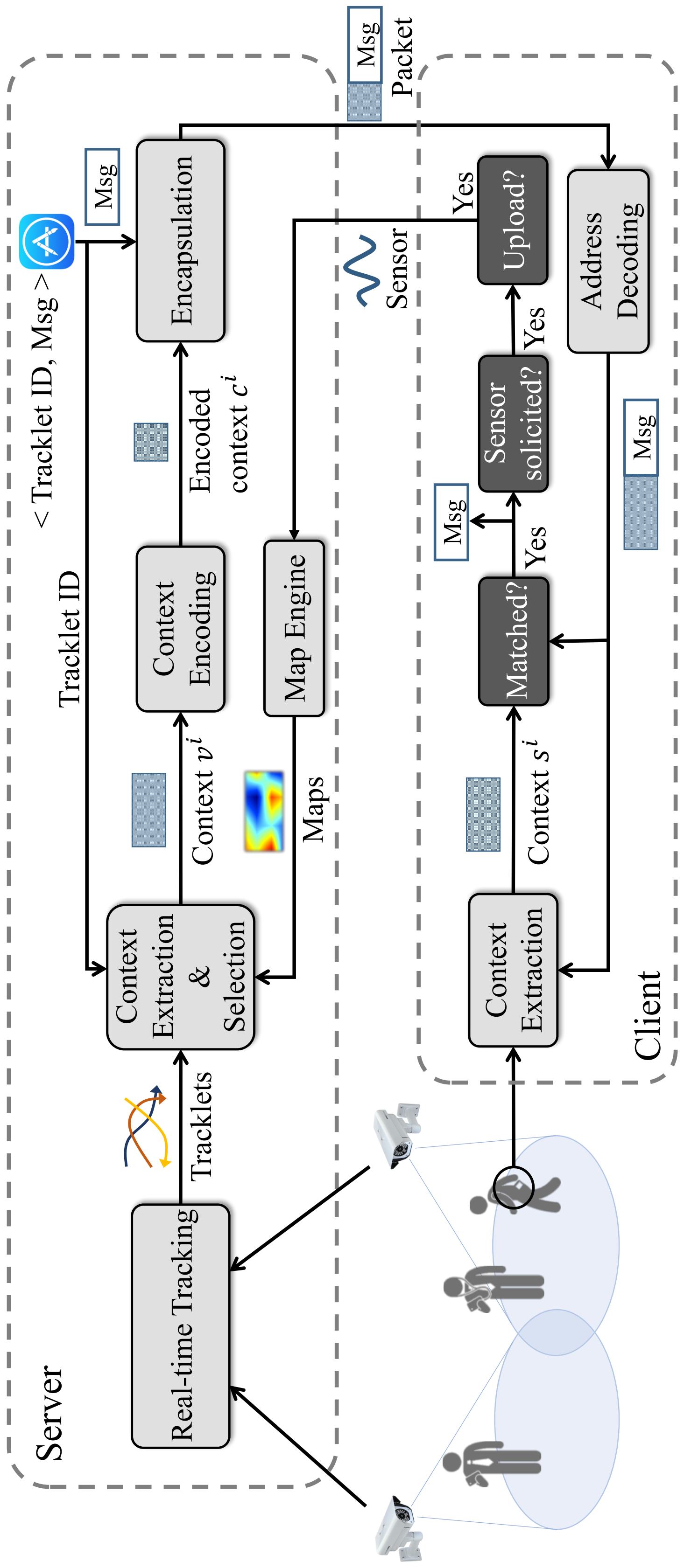


Fig. 3.2. System overview.

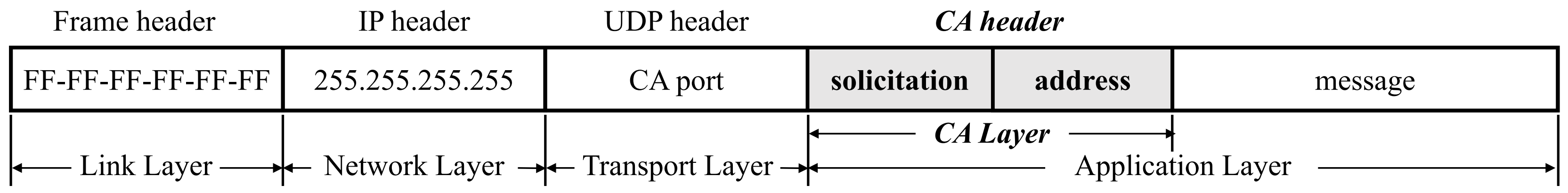


Fig. 3.3. Network packet with proposed context address (CA) layer.

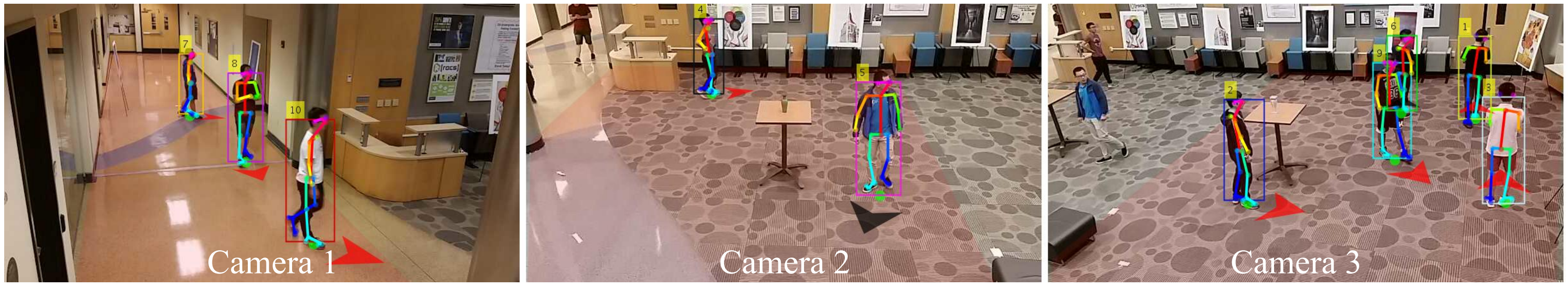


Fig. 3.4. Tracking result in a real-life scenario with three cameras. It is stable and accurate despite having mutual occlusions (e.g. tracklet 6).

from the rest, are selected as the context address for that person. The selected features have various lengths and formats according to their types. To reduce the payload overhead and maintain consistency in packet structure, the context features are organized and encoded into a fixed-length header. As shown in Figure 3.3, the context address header may also include solicitation, which requests the target user to voluntarily upload its recent magnetometer and Wi-Fi data. This data is later used to update the ambient sensing maps. The context address header is combined with a message and put into the application layer of a network packet while the destination IP/MAC address is set to all one's. The packet is then broadcast through UDP over Wi-Fi. The jobs conducted at the server are separated into stages and accomplished in a pipelined and parallel manner [81] for the system to work in real time. This guarantees the messages to be sent out with a short and constant delay.

On the other hand, clients passively listen to all broadcast messages and locally decide whether a message is destined for them or not. Once a smartphone carried by a person receives a broadcast packet, it extracts the corresponding motion and ambience features from its sensor readings according to the decoded context address in the packet. By comparing each feature in the context address with the smartphone's sensor readings, an overall matching score is calculated and used to decide if the message should be accepted. If accepted, the message is passed on to the upper-level applications. Also, if there is a solicitation in the header, the client may voluntarily upload the requested sensor readings to the server. This uploaded data facilitates magnetic trend and Wi-Fi map generation in the map engine on the server.

3.3 System Design

This section describes the design details of each component of the system beginning with the pedestrian tracking across multiple cameras, followed by the context extraction and selection process. Then it describes the structure of the context header and the matching schemes on the client side for receiving messages. Finally, the ambient sensing map generation is explained.

3.3.1 Real-time Multi-camera Human Tracking

To track people through multiple cameras in real time, we employ a pipelined and parallel scheme proposed in [81]. A state-of-the-art human pose detector, OpenPose [49], is first used for pedestrian detection. *Association Based Tracking* (ABT) [36–40] conducts low-level association between detection responses from neighboring frames of the streamed video. The tracklets are extended via *Category Free Tracking* (CFT) [41–43] and Kalman filter is applied to form local tracklets representing each person in a camera view. The local tracklets from all cameras are eventually merged into global tracklets in the entire covered area. An example of tracking results from our experiments is shown in Figure 3.4.

3.3.2 Context Extraction

Context features that qualify for address matching between videos and sensors should be: (1) distinguishing, i.e. having rich diversity among different people; (2) reliable, i.e. being consistent between the two sides to validate matching. We define the context features, which consist of motion and ambience features. Since the context address is added into the packet payload, we encode the features to reduce the overhead.

Motion Features

Since the tracking process generates locations of each person, a person's velocity can be computed by applying Kalman filter on its locations [47, 83], and further extracted into motion features with no extra computation cost. We adopt the motion features defined in [81] and briefly describe them for the sake of completeness.

Moving Or Not: On the video side, from the velocity magnitude, it is straightforward to determine whether a person is *Moving Or Not*. On the sensor side, sensed acceleration is first projected onto gravity and its variance, within two seconds, is calculated. If the variance is above a predefined threshold, we can mark Moving or Not as "Yes".

Relative Rotation: We define *Relative Rotation* as the difference between a person's walking directions at the beginning and the end of a motion period. On the sensor side, rotation rates obtained from gyroscope are first projected onto the gravity and then integrated into Relative Rotation. For comparison, we define an adaptive threshold ($= Kl + B$) to compensate gyroscope drift, where l is the time length of the motion period and K, B are parameters preset to $1°/s$ and $25°$, respectively. If the rotation difference is within the threshold, we take it as a match.

Ambience Features

The visual tracking incidentally generates a user's location at each timestamp, which can be used with ambient sensing maps to extract location-related ambience features. For now, we assume that the server has a Magnetic Trend map and a Wi-Fi Fingerprint map, and will explain why we select these feature types and how we generate the maps in Section 3.3.6.

Magnetic Trend: *Magnetic Trend* represents the difference between magnetometer readings in gravity direction at any two locations. For each pair of different locations, the difference is represented by a normal distribution using its mean (μ) and standard deviation (σ) and stored in Magnetic Trend map. Figure 3.5 shows an example of increasing Magnetic Trend (i.e. $\mu > 0$) from location A to B, where the blocks are different locations and red indicates larger projected magnetometer readings. When a person moves from one location to another during a motion period, a (μ, σ) pair is extracted by looking up Magnetic Trend map with its two locations and used as a candidate feature.

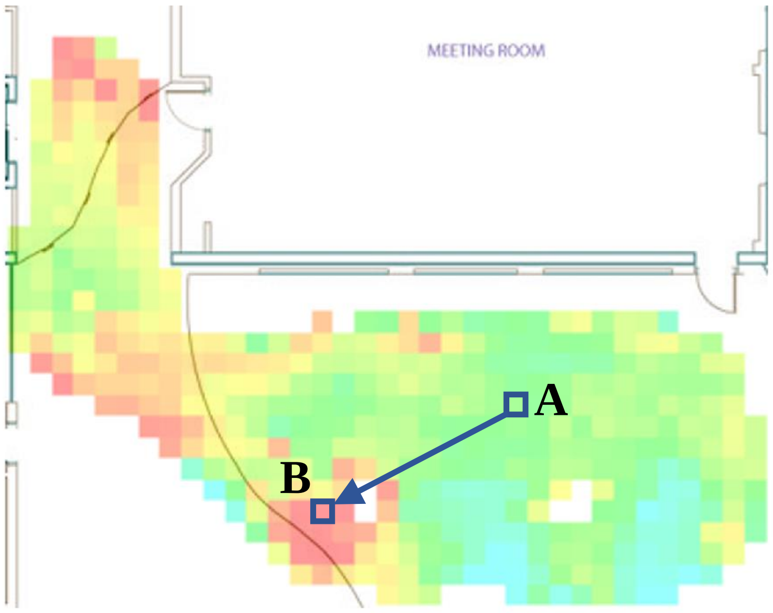


Fig. 3.5. Increasing Magnetic Trend between two locations.

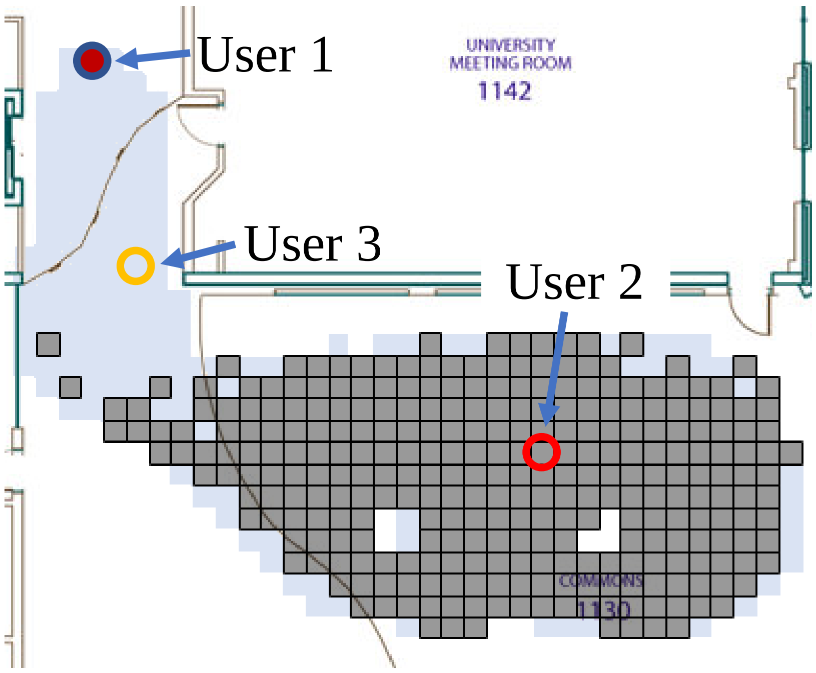


Fig. 3.6. Wi-Fi Fingerprint map for a reference position (shown by solid red).

On the sensor side, the phone periodically samples the magnetometer and gravity readings. The 3D magnetometer readings are first projected to the current gravity to eliminate the influence of the phone pose. The projected magnetometer readings are then used to calculate the difference and to compare with (μ, σ) from the server side. If the difference from the sensor side lies in the range $\mu \pm \lambda\sigma$ ($\lambda = 2.5$, i.e. 98.8% confidence interval), we consider the sensor readings match with this Magnetic Trend feature.

Wi-Fi Fingerprint: In Wi-Fi Fingerprint map, each location in the area contains a series of Wi-Fi signal strength readings of N_w (preset to 15) different MACs, i.e. *Wi-Fi Fingerprint*, as well as a distinguishable region. For a specific location (defined as *reference position*), its *distinguishable region* represents the locations with Wi-Fi fingerprints that have large Euclidean distances from the Wi-Fi fingerprint of the reference position. Based on the tracking results at a certain timestamp, if a target user is tracked at a reference position with a valid term in Wi-Fi Fingerprint map while some other users are in the distinguishable region, Wi-Fi Fingerprint can be extracted as a candidate feature to distinguish the target. We carefully selected N_w Wi-Fi's that are stable in each block and have the most distinguishability among different locations. Please refer to Section 3.3.6 for more details.

Figure 3.6 shows an example of a Wi-Fi Fingerprint map. Suppose that user 1, at the reference position, is our target to send a message. User 1 can be distinguished from user 2 using Wi-Fi Fingerprint since user 2 presents in the distinguishable region of the reference position, while user 1 cannot be distinguished from user 3. Note that the distinguishable region is around 5 meters away from the reference position, which is larger than the resolution of some existing Wi-Fi based localization scheme. This is to keep Wi-Fi Fingerprint noise-robust and ensure its reliability.

Table 3.1.
Payload cost of each feature in terms of number of bits.

Feature	**Type**	**Δt_1**	**Δt_2**	**Content**	**Total**
Moving Or Not	3	5	-	1	9
Relative Rotation			5	18	31
Magnetic Trend			5	18	31
Wi-Fi Fingerprint			-	~75	~83

Context Encoding

To minimize the payload overhead, each feature is compressed into a bit string. The encoded feature structure and corresponding payload cost in bits are shown in Table 3.1. 3 bits represent the feature *type*. 5 bits represent each timestamp (Δt_1 or Δt_2), which is used by the client to search for corresponding sensor readings. Either one or two timestamps are needed, depending on whether the feature contains an absolute or relative value. The *content* length varies among different features. *Moving Or Not* needs 1 bit to represent two states. *Relative Rotation* needs 9 bits to represent an angle (0 - 360°). *Magnetic Trend* uses 18 bits – 9 bits for μ and σ each. *Wi-Fi Fingerprint* uses, on average, 75 bits to specify 15 Wi-Fi signal strength values with different MAC addresses. Note that this is not a fixed cost since we encode the Wi-Fi signal strengths using a variant of Huffman coding [84] based on empirical frequencies. Details are omitted in the interest of space. The MAC addresses are sent only once when a user enters the covered area so the cost is not included.

Table 3.2.
Example feature table T for target user 1 and payload cost for each feature. Selected features marked by ticks.

	Feature	User 2	User 3	User 4	Cost
✓	$f^{Moving\ Or\ Not}(\overrightarrow{t_1})$	1	0	0	9
✓	$f^{Magnetic\ Trend}(\overrightarrow{t_2})$	0	1	1	31
	$f^{Relative\ Rotation}(\overrightarrow{t_1})$	0	0	1	31
	$f^{WiFi\ Fingerprint}(\overrightarrow{t_3})$	1	1	0	46
✓	$f^{WiFi\ Fingerprint}(\overrightarrow{t_4})$	0	1	0	83
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

3.3.3 Context Selection

Once the context features are extracted and encoded into the format specified above, the next task is to select the optimal set of features capable of distinguishing an individual from other people in the video.

Feature Table Construction: We define binary function D, where $D(f', f'') = 1$ if two features f' and f'' are different. $f'(= f^{type_k}_{person_i}(\overrightarrow{t}))$ and $f''(= f^{type_k}_{person_j}(\overrightarrow{t}))$ are features with the same type and timestamps, but for different people. We use the same comparison in Section 3.3.2 to evaluate D.

Based on D, we build a feature table T for each target that we want to send messages to. T is of size $m \times n$, where each row is for one feature with a certain type and some timestamps, and each column is for a person besides the target. Since in our experiment (Section 3.4.1), we use the features from last 30 seconds and distinguish among ten users, a typical size of T is 876×9. Each entry T_{ij} is 1 only if the jth user can be discriminated from the target by using the ith feature. Each feature is also associated with a pre-defined payload cost, C_i. An example feature table T is shown in Table 3.2. $T_{11} = 1$ means that user 2 can be distinguished from user 1 by *Moving Or Not* at time $\vec{t_1}$.

To determine the value for T_{ij}, first we need to check whether the features f^i for both the target and the other user is *valid*. For a motion feature, we follow the criteria in [81]. For an ambience feature, we define "valid" as that a corresponding value can be found for both users from the map.

Secondly, due to time delay inherited from video streaming and packet propagation, a recorded timestamp may shift by a small amount from its actual value. We want to consider only those features which are *stable* for the target across a shift period Δs $(= 0.5s)$. Namely, a feature f^i at time $\overrightarrow{t} = (\Delta t_1, \Delta t_2)$, is considered stable for T if

$$\forall s \in [0, \Delta s],\ D(f^i_{target}(\overrightarrow{t}), f^i_{target}(\overrightarrow{t+s})) = 0. \tag{3.1}$$

Finally, we also consider the time shift when comparing the target and other users. A feature is discriminative between the target and user j when

$$\forall s \in [0, \Delta s],\ D(f^i_{target}(\overrightarrow{t}), f^i_j(\overrightarrow{t+s})) = 1. \tag{3.2}$$

As a result, $T_{ij} = 1$ only if all the above three conditions are satisfied. All features in T compose a set F $(= \{f^1, \ldots, f^m\})$.

Selection Algorithm: To select the most effective features, a naive way is to decreasingly sort all candidate features according to their distinguishability/cost ratio, $H_i = \sum_j T_{ij}/C_i$. Then the features are chosen in this order until the total cost reaches a limit C_m (= 40 bytes). However, this method may fail when a specific user cannot be discriminated from the target by these selected features, even if $\sum_i H_i$ is large.

Therefore, we define *distinguishing power vector* P as the sorted sum of selected rows in feature table T, and lexicographically maximize P in a greedy manner under the limit of total payload cost. This is formulated as:

$$\max P = sort(\sum_{i \in I} T_{i\cdot}),\ \text{s.t.} \sum_{i \in I} C_i \le C_m, \tag{3.3}$$

where $sort()$ ascendingly sorts the elements of a vector, and $T_{i\cdot}$ is the i-th row of matrix T. $I \subseteq F$ is the selected feature set.

Algorithm 1: Context Selection

```
Initial selected feature set, $I \leftarrow \{\}$
Initial distinguishing power, $P \leftarrow \vec{0}$
Number of iterations, $n_{iter} \leftarrow 0$
while $n_{iter} < n_{max}$ do
  for each feature $k \in I \cup \{\emptyset\}$
    for each feature $l \in \overline{I} \cup \{\emptyset\}$
      $I' \leftarrow (I \setminus k) \cup l$
      $P' \leftarrow sort(\sum_{i \in I'} T_{i\cdot})$
      if $P$ lexicographically smaller than $P'$ and $\sum_{i \in I'} C_i \leq C_m$
        $I'' \leftarrow I', P \leftarrow P'$
  $I \leftarrow I'', n_{iter} \leftarrow n_{iter} + 1$
```

Lexicographical maximization of this sorted vector P guarantees that we have high distinguishability even for the least distinguishable user j, where j is the index of the smallest element in $\sum_{i \in I} T_{i\cdot}$. We can successfully send the packet only when the *normalized distinguishing power* $\hat{P} = P_1/|I|$ (note that P is already sorted and P_1 is the first element in P) is above a threshold (0.1). Otherwise, the attempt of sending the packet fails. A *sending ratio* is defined as the number of packets successfully sent over the total number of attempts.

We formulate a local search strategy (Algorithm 1) to solve this computationally hard optimization problem. It begins with an empty set and keeps applying local changes to the selected feature set I by adding, removing, or substituting one feature at a time. The iteration stops when n_{iter} reaches a predefined limit n_{max}. For each iteration, we greedily maximize the increase of P by enumeration. This converged set I is used as the context address for the target user.

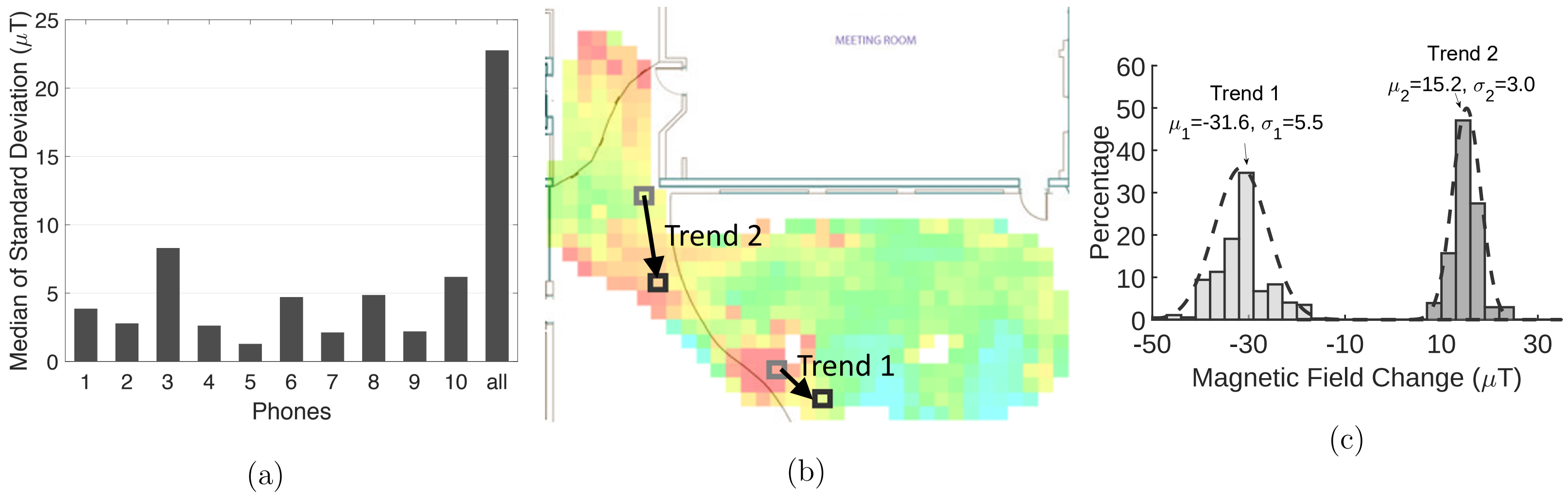


Fig. 3.7. Magnetic Trend map. (a) Median of standard deviations of projected magnetometer readings among all blocks. (b) Two pairs of locations with different Magnetic Trends. (c) Normal distributions representing the two Magnetic Trends labeled in (b).

3.3.4 Packet Encapsulation

In the context address header, some other fields are required. The header also includes the normalized distinguishing power $\hat{P}$ (7 bits) as a threshold for context matching on the client side. Moreover, depending on the completeness of stored maps and recent locations of the target user, the server occasionally requests the target to voluntarily upload its magnetometer data and/or scanned Wi-Fi signal strengths. The solicitation for this data uses 2 bits to convey whether each type is needed. Alongside this, the transaction ID of this request (8 bits) is used to keep track of the sensor data received from the users later on. The context address header, containing all the above fields as well as the selected context features, is organized and encoded. It is put into the application layer of a packet along with an application message (as shown in Figure 3.3). The server then broadcasts the packet to all the users in the area.

3.3.5 Packet Processing on the Client Side

Upon receiving a broadcast packet, the user's phone decodes the context address header and extract all fields described in Section 3.3.2 and Section 3.3.4. The phone extracts its corresponding sensor data for each feature at the time computed by subtracting Δt_1 and Δt_2 from the current time on the phone. Note that we do not need to consider the packet propagation delay here since it has already been dealt with during context selection (Equation 3.1 and 3.2). Each sensor-based feature is then extracted and compared with the video-based feature as discussed in Section 3.3.2. The matching scores are averaged over all the features to obtain the overall matching score for this packet. If the matching score is greater than $\hat{P}/2$ in the received context address header, the client accepts the message in the packet and forwards it to upper-level applications.

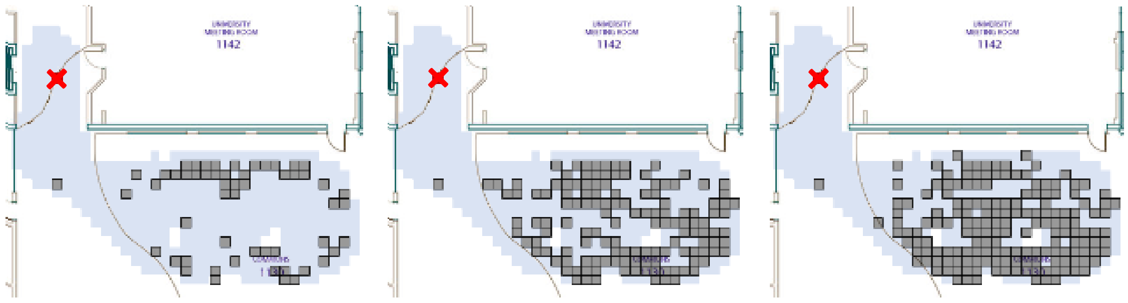
Upon accepting a message, the client also checks the solicitation fields in the packet and may volunteer to upload the requested sensor data, e.g. magnetometer readings or scanned Wi-Fi signal strengths.

3.3.6 Map Generation

Now we come back to the two ambient sensing maps and discuss how they are generated efficiently. The first step is to divide the area covered by the camera views into a grid of small blocks. Each block is $0.5m \times 0.5m$, which determines the spatial resolution of the maps.

Magnetic Trend map: Upon receiving voluntarily uploaded sensing data, we first project the magnetometer readings to the gravity to eliminate the influence of the phone pose. As the time and location series can be easily obtained from the visual tracking process, a straightforward way is to directly use the magnetometer reading in the gravity direction as a fingerprint. The problem is that phone models and sensor quality may affect the absolute sensor readings, which is a non-negligible source of errors. Figure 3.7 (a) shows the median of standard deviations of projected magnetometer readings among all blocks. We observe that in the same block, the projected readings collected from the same user are consistent while the entire dataset from all users lies scattered. It inspires that, if we use a relative trend between the magnetometer readings from two different blocks collected by the same device, it is more stable than using the absolute values from different devices.

Therefore, we compute the difference between the projected magnetometer readings from one user and add it to the map, only when the user walks from one block to another. For each pair of blocks, we approximate these differences into a normal distribution, which is represented by its mean (μ) and standard deviation (σ). In Figure 3.7 (b), there are two pairs of blocks, whose Magnetic Trends are labeled as trend 1 and trend 2, respectively. Figure 3.7 (c) shows the normal distributions representing these two Magnetic Trends, where the two 98.8% confidence intervals do not overlap.



(a) After 2 minutes. (b) After 4 minutes. (c) After 6 minutes.

Fig. 3.8. Wi-Fi Fingerprint maps for a certain block (marked by red cross).

Therefore, when using Magnetic Trend features as described in Section 3.3.2, a person who passes the block pair of trend 1 can be distinguished with high confidence, from another person who passes the block pair of trend 2 at the same timestamps. Using a distribution instead of a single value to represent Magnetic Trend provides tolerance to possible fluctuations caused by sensor noises.

Wi-Fi Fingerprint map: Wi-Fi Fingerprint map is generated through a multi-step process. Similar to Magnetic Trend, we may receive scanned Wi-Fi signal strengths from volunteer users. First, for each MAC address and each block, we compute the median of the Wi-Fi signal strengths from all users. Secondly, we calculate the variance of the medians, var, for each MAC address. var represents how the Wi-Fi signal strength differs across the blocks. Thirdly, the MAC addresses are then sorted decreasingly by var and the top N_w $(= 15)$ MACs are selected for higher distinguishability among different locations. Finally, Wi-Fi Fingerprint map is generated for all blocks in the grid. It stores the medians of the Wi-Fi signal strengths of the selected MACs. All other blocks with large Euclidean distances to the *reference position* (defined in Section 3.3.2), i.e. over a threshold of 10 dB, are marked as the distinguishable region of the reference position.

Figure 3.8 shows an example of a Wi-Fi Fingerprint map for a reference position (marked by a red cross) at 2, 4 and 6 minutes. The walkable region is shown by light shade and the distinguishable region by dark shade. With time, as more and more data is contributed by users, the distinguishable region grows, showing that the difference between the reference position and the other blocks becomes clearer. Thus, if a target user is tracked located at the reference position in this map while another user is in the distinguishable region, this Wi-Fi Fingerprint feature can be selected to distinguish these two users.

In real cases, the server does not need to frequently send solicitation requests. Once the collected dataset is large enough, the server holds back on solicitation for that area. An expiration time can also be set to void outdated sensing data. In our experiments, we ignore these two factors due to the short experiment period.

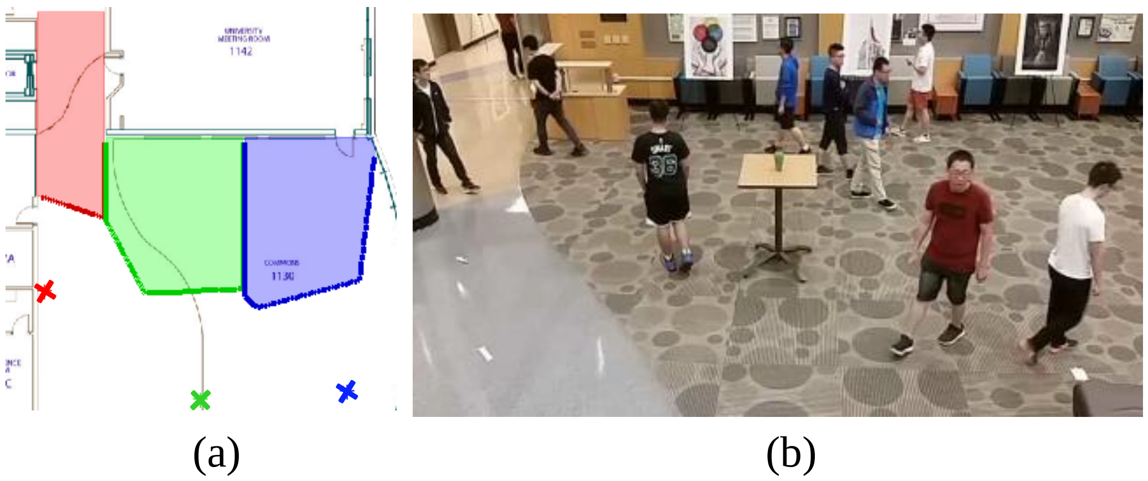


Fig. 3.9. Experiment scenario. (a) Areas covered by three cameras and corresponding camera positions. (b) An example frame in a mimic gallery.

Table 3.3.
End-to-end computation time.

Stage	**Time in Seconds**
Tracking	1.5
Context extraction	1.3
Context selection and packaging	0.2
Client-side processing	0.2
Total	**3.2**

3.4 Evaluation

3.4.1 Experimental Setup

In our real-world experiment, three Samsung Galaxy S5 smartphones are used as IP cameras to capture and stream videos at a frame rate of 13 fps, a bit rate of 2000 kbps, and a resolution of 800×480. We set up our server on three PCs with dual NVIDIA GTX 1080 Ti SLI, and run MATLAB and C++ programs on each. A software called ClockSynchro [85] is used to synchronize these computers. Google Pixel XL smartphones are employed as clients, which log accelerometer, gyroscope, gravity, magnetometer readings, and Wi-Fi scan results at $400Hz$, $400Hz$, $200Hz$, $50Hz$ and $1Hz$ respectively. They also run our Android client app to receive packets.

We evaluated our system in a real-world scenario of an "art gallery" in a university lobby with a walkable area of $107m^2$. The area covered by each camera is shown in Figure 3.9(a) with shades and the camera positions are marked with crosses. We invited 10 volunteers to naturally walk around or stop by at the paintings as they pleased, with a smartphone put in their pockets. Figure 3.9(b) shows an example frame from one camera, with 10 users in the gallery. We tested the utility of the context address by sending messages to all 10 users every two seconds, i.e. 6000 messages sent in a 20-minute session.

3.4.2 Performance Results

We intend to concentrate on the following aspects:

(1) System Overall Performance

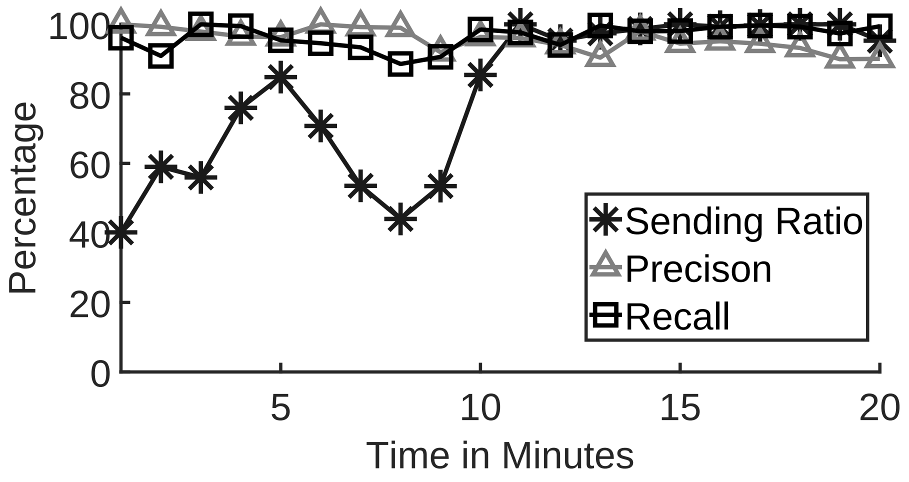


Fig. 3.10. Overall precision, recall and sending ratio of the system.

We evaluate the overall precision and recall rate of our system. The precision represents the ratio of the messages accepted by a user which are actually targeted for it. The recall is the ratio of the messages targeted for a specific user which are successfully accepted by it. Figure 3.10 shows that, as magnetometer and Wi-Fi readings gradually contribute to the map generating process, the performance starts to improve after about 10 minutes. After the cold start period, the average precision of our system is 93.4% throughout the last 10 minutes while the recall rate is 98.3%. Moreover, the sending ratio (defined in Section 3.3.3) increases sharply and reaches an average of 98.5%. The combination of motion and ambience features leads to an overall high precision and recall rate, showing that the context features used have a high distinguishability and the maps are stable over time.

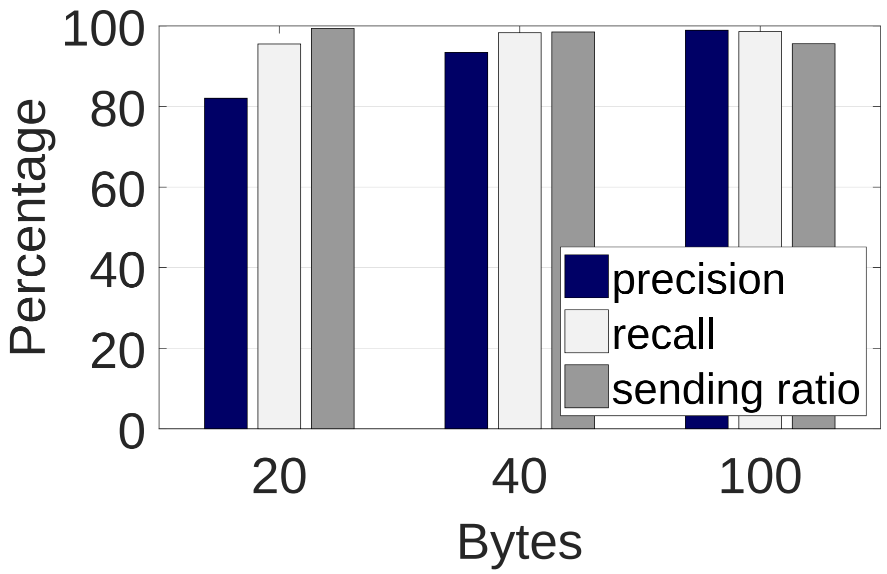


Fig. 3.11. Performance when using the context address header of various maximum length.

Table 3.3 shows the median of computation time for different processing stages through 20 minutes. The total computation time is 3.2 seconds, in which the tracking process takes the largest portion, i.e. 1.5 seconds. The tracking time could be shrunk if a faster and more accurate tracking scheme can be introduced into our system. And other stages, i.e. context extraction, context selection and packaging, and client side processing, take 1.3, 0.2, and 0.2 seconds, respectively. This demonstrates the efficiency of our context selection algorithm.

(2) Packet Overhead

We set the maximum number of bytes for the context address header to 20, 40 and 100, and evaluate how it affects the performance during the last 10 minutes. Figure 3.11 shows the results. When the limit is set to 40 bytes (i.e. same as an IPv6 header), it already achieves similar performance as it's extended to 100 bytes. In PHADE [81], the packet overhead is a severe problem since the large coefficient

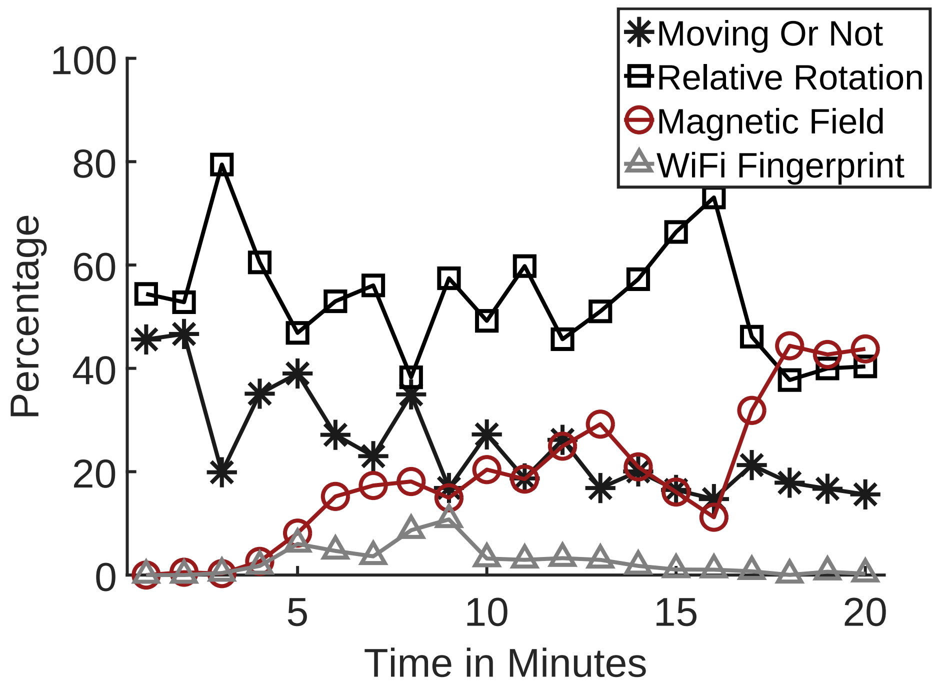


Fig. 3.12. The proportion of different types of features selected for the context address.

matrix needs to be sent out, no matter how many users that the system is trying to communicate with. For example, same as in our experiment, if PHADE is sending messages to 10 users at the same time, the average packet overhead for each user is 200 floats, i.e. 800 bytes. In contrast, the packet overhead in our system is always less than 40 bytes regardless of the number of users, only 5% of PHADE.

(3) Selected Context Features

The types of context features selected over time are shown in Figure 3.12. During the initial period, only *Move or Not* and *Relative Rotation* are selected. This is because the system was at a start-up stage, waiting for users to voluntarily upload their sensor data. With more and more valid sensor data contributed to building the ambient sensing maps, *Magnetic Trend* and *Wi-Fi Fingerprint* features start to

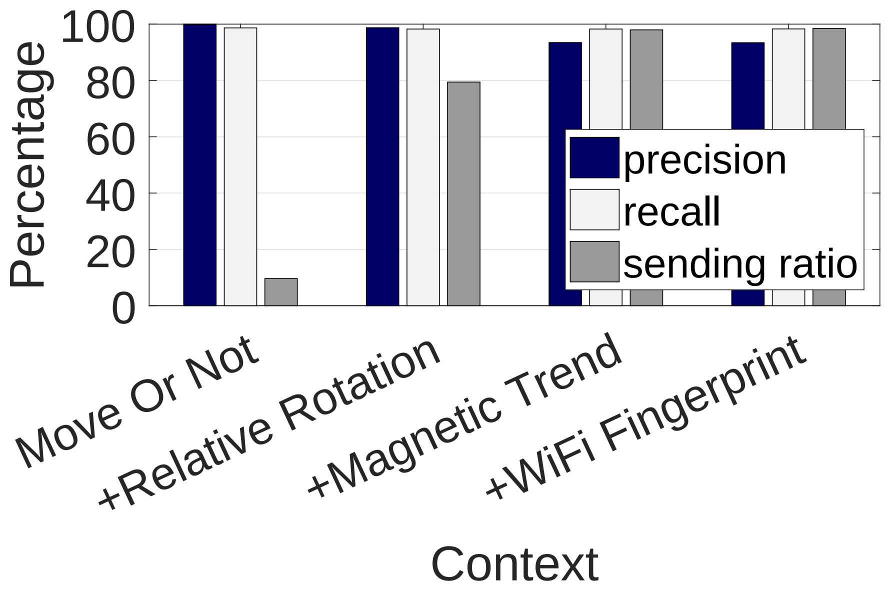


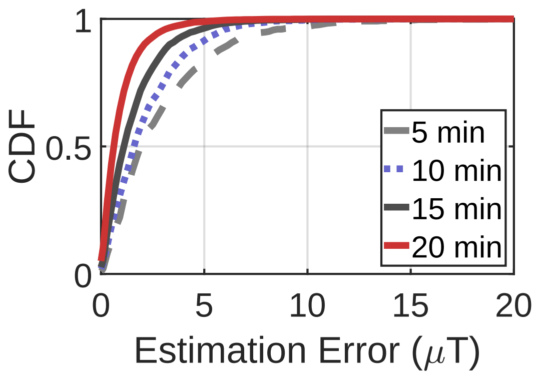
Fig. 3.13. Performance by adding different types of context features.

be selected. The percentage of *Magnetic Trend* increases greatly after the first 4 minutes. The *Wi-Fi Fingerprint* has limited contribution because its payload cost is larger than the other three types.

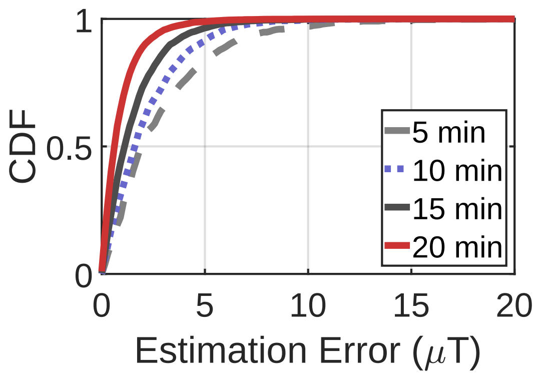
Figure 3.13 shows how the system performance changes when adding different types of context features. When adding more types of features, the sending ratio is largely enhanced while the precision and the recall remain high.

(4) Maps Generated Over Time

To build the ground truth maps, we use the ground truth messaging destinations to get all available sensor data from the users. We evaluate Magnetic Trend map in terms of errors of the mean and the standard deviation of the difference in each pair of blocks. Figure 3.14 shows that as there are more and more sensor data, these two kinds of errors gradually and steadily decrease over time. After about 10 minutes, the median of both errors drops below $1.5\mu T$. This implies that the map generation



(a)



(b)

Fig. 3.14. Error in the difference of magnetometer readings over time. (a) and (b) show the distribution of errors in estimated mean and standard deviation, respectively.

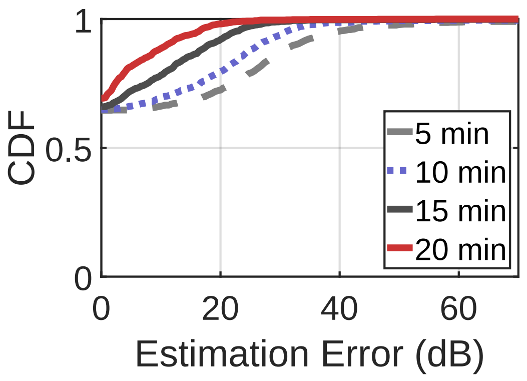


Fig. 3.15. Error in Wi-Fi Fingerprint map over time.

converges in a short time. Similarly, we evaluate the errors of Wi-Fi Fingerprint map by calculating the distances from the medians of Wi-Fi signal strengths. Figure 3.15 shows that with more Wi-Fi readings uploaded, the Wi-Fi Fingerprint map gradually approaches the ground truth.

3.4.3 Video Simulation

We also want to gain insight into the scalability and feasibility of our system in a retailer scenario with dense people. Since we want to observe the natural behavior of human, we run an alternative simulation to approximate our system's performance when deployed in a real retail store. Instead of obtaining real sensor readings from the smartphones carried by people, we synthesize sensor features by injecting statistical noises into video-based features, to simulate the inconsistency between the video and the sensor sides.

To conduct the simulation, we first record videos in a retail store during its busy hours at 5-6pm. Two cameras cooperate to cover a walkable area of $105m^2$ and each video is 6 minutes long. The cameras capture up to 13 people simultaneously and the entire videos include 52 distinct people. Figure 3.16 is an example frame illustrating the typical people density in the video. We then conduct pedestrian tracking on these videos and extracted the video-based context features for each person. To synthesize sensor-based motion features, we analyze the error distributions using the data collected in Section 3.4.1 and inject noises based on these distributions into the video-based features. For synthesizing sensor-based ambience features, we wardrive this area for 20 minutes, and use this data for both building the maps and fitting the error distributions for magnetometer and Wi-Fi data. The wardriving is also conducted between 5-6pm to ensure the collected Wi-Fi signal strengths reflect the actual readings with crowds in the surrounding. Finally, we simulate the message sending process and each person accepts messages by comparing the synthesized sensor features with the actual video-based features.

The simulation demonstrates that our system can distinguish a person in practical and dense scenarios, reaching a sending ratio of 90.0%, a precision of 99.7%, and a recall of 95.3%.

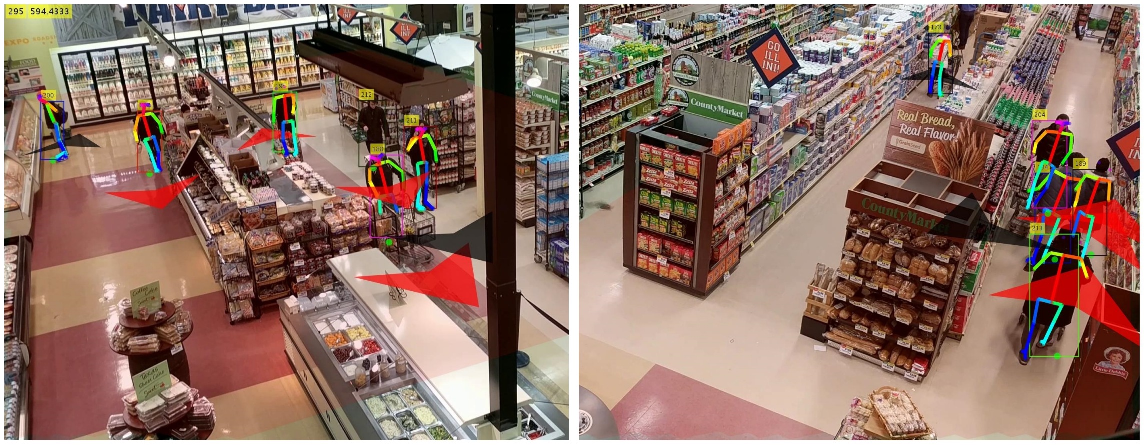


Fig. 3.16. Video simulation scenario.

3.5 Discussion

Limitations: As our system highly depends on the performance of pedestrian detection and tracking schemes, it may not work well in scenarios with lots of obstacles in the environment and human mutual occlusion. If the tracking fails occasionally, the user is treated as a new person just entering the area. Therefore, the chance of successfully delivering the messages is affected.

Privacy protection: Recall that in Chapter 2, *PHADE* uses PCA to transform the raw motion features to prevent motion leakage to the public. However, the enhanced system in this chapter protects user privacy from another perspective. Instead of utilizing all the continuous motion features extracted from recent history, now we discretize the features along the time axis into separate feature points and only select a small set of effective ones as the address. In our experiment with 10 people, during an attempt of sending a message, 800–900 candidate features are included in the feature table, among which only about 10 are selected and put into the communication address — that is merely 1% of them. An example of the selected feature set would contain 2 *Move or Not*, 5 *Relative Rotation*, 2 *Magnetic Trend*, and 1 *Wi-Fi Fingerprint* features. It is difficult to recover any meaningful walking history from such a small amount of information.

Other possible features: Since our system has explored multiple ways to represent features (e.g. single data point, trend, and fingerprint), it can also be extended to use other features, such as light intensity, walking direction, step phase, Bluetooth, 5G signals, etc. They may contribute differently in various use cases, for example, step phase can be used to distinguish people walking along a similar trajectory.

Difference with indoor localization: One may wonder if we built another indoor localization system. The short answer is "No". Our system, as a general framework for direct camera-to-human communication, can be adapted into more various application scenarios, as discussed in Section 3.1, including indoor localization. Some systems with similar communication purposes [81] have demonstrated these applications with real-world evaluations. Even if we had a perfect indoor localization system, this location information is not suitable for a communication system like ours. One main reason is the potential privacy leakage from broadcasting location information.

Broadcast methodology: In our system implementation, we use Wi-Fi as the broadcast media. But there are also other options, such as LTE Direct, BLE advertisement, etc.

3.6 Related Work

Message delivery based on visual tracking. Recent works have built some communication paths to send messages to a targeted person in surveillance camera views. PHADE [81] uses people's walking behaviors as their temporary communication addresses. However, in some crowded scenarios, merely using motion features does not provide enough distinguishability to represent each person. Also, PHADE transmits large coefficient matrices along with address codes, which introduces a non-negligible packet overhead. On the other hand, our system utilizes both motion and ambience features to obtain higher distinguishing ability and introduces a context address header with a small fixed length while providing accurate message targeting.

Another work, TAR [82] uses a combination of multi-camera human tracking and Bluetooth proximity sensing to conduct ID association and deliver targeted advertisements. When some people are in close proximity, TAR needs BLE readings for a longer period to identify a person among them. However, the ability to discriminate a user from its companions is insufficient since Bluetooth proximity is the only feature used. Our system can distinguish a person from a denser group relying on richer context features, even if some people locate closely or behave similarly. Also, it requires Bluetooth on all users' smartphones to be on and continuously broadcast BLE signals, which raises severe privacy concerns.

Human ID association. Existing schemes for human ID association use various techniques and devices for identification. [29] has used the accelerometer readings from a sensor worn on a person's belt to develop an ID matching algorithm for associating people. Another work, Insight [27], uses the motion patterns and clothing colors to recognize people. These patterns serve as a temporary fingerprint for an individual. Both of these schemes depend on the users to upload their sensor data while we can still correctly identify a user even if it does not upload any data. Also, in contrast to [27, 29], we have implemented our idea into a real-time end-to-end system. Among other approaches, ID-Match [80] can recognize and correctly assign IDs to individuals using relative motion paths of RFID tags worn by people and 3D camera. For outdoor environments, RFID and BLE are combined with a stereo-based identification system in [86]. In these approaches, the identification relies on users wearing RFID tags or BLE beacons. It is hard to ensure everyone carries its tag in a large public area, hence rendering these schemes infeasible for such environments. Our system associates a user in the camera view with its smartphone without requiring tags or preregistration.

Camera sensor combination. Research based on combining cameras and sensors has been popular in recent past with widespread applications. Gabriel [87] uses image capturing and mobile sensing to develop a cognitive assistance system. Smartphone's motion and light sensors combined with cameras allow authors in [75] to enhance the biometric authentication process through facial recognition. Overlay [73] uses a fusion of a smartphone camera and sensors to enable augmented reality on the phone via building a geometric representation of the environment. We introduce the novel concept of using cameras and smartphone sensors to allow communication between the camera and people in the camera view with applications in public safety and other day-to-day activities.

3.7 Conclusion

This chapter solves the problem of digitally associating people in a camera view to their smartphones without knowing their IP/MAC addresses. A fully operational real-time prototype system is developed, which utilizes a context address consisting of motion patterns and ambience to identify each person. We deploy an efficient context selection algorithm to choose discriminative features and fit them into a fixed-length header. We also generate ambient sensing maps in an effortless way. Our system achieves a sending ratio of 98.5%, an acceptance precision of 93.4%, and a recall of 98.3%.

4. VIU: CONTROLLER TRACKING FOR MOBILE VR USING INERTIAL-ULTRASONIC SENSOR FUSION

4.1 Introduction

Since Google Cardboard was released in 2014 [88], mobile Virtual Reality (VR) headsets have enabled people to dive into the virtual world, anytime and anywhere. Compared with PC/console VR and standalone VR headsets, mobile VR headsets provide reasonable VR experience with lower prices and better portability. However, merely inserting a smartphone as a head-mounted display (HMD) without extra peripherals, e.g. external cameras and LEDs, mobile VR suffers from poor positional tracking on its motion controller. There is a need of enhancing mobile VR controller tracking with existing on-chip sensors to improve users' VR immersion and interaction experience.

To solve this problem, main commercial mobile VR devices, e.g. Google Daydream View and Samsung Gear VR, conduct 3 degree-of-freedom (DoF) rotational tracking using embedded inertial sensors and query a mathematical approximation of the controller's positions based on the rotations and a simplified arm model. Without any actual 3D position information, these systems simply distribute the measured rotations among several virtual joints (e.g. elbow, wrist, etc.) to approximate the controller's positions. Moreover, they tend to restrict the rendered controller into a small range in VR scenes to blur obvious errors, thus predict larger movements with less accuracy [89]. This makes the controller function as a laser pointer, which is good for simple actions like selecting a menu but fails in interactive gaming like slingshot (shown in Figure 4.1(c)). Someone may ask: Why not use cameras? Vision-based solutions are indeed popular among console and standalone VR for 6DoF controller tracking. However, it's hard to transplant these solutions to mobile VR due to the

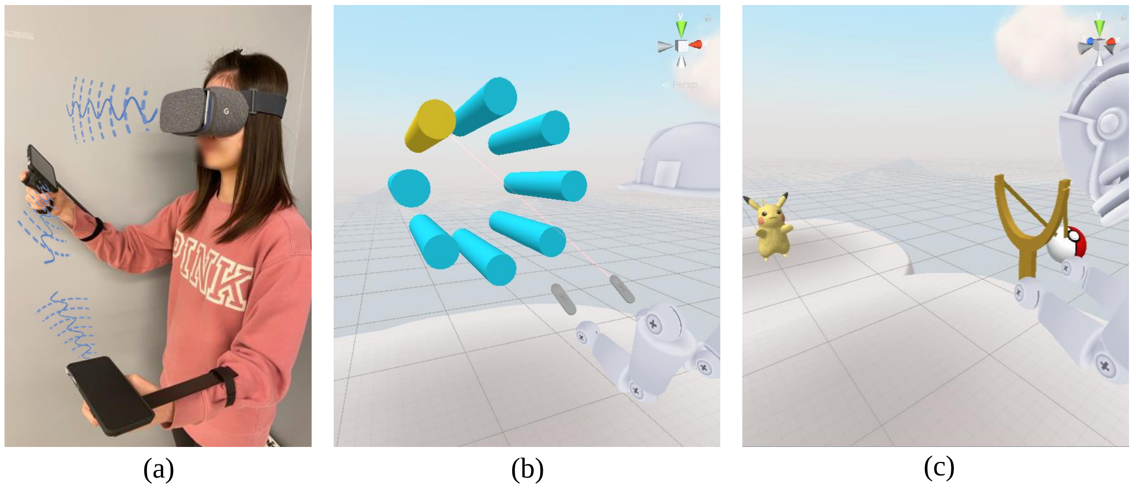


Fig. 4.1. (a) *VIU* prototype, and its usage in different VR applications: (b) object selection, (c) slingshot.

lack of dedicated hardware on smartphones. They rely on either external infrared (IR) laser emitters [90] or multiple cameras embedded on the headset. the headset to perform controller tracking. They also suffer from tracking failure in blocking or dark cases [91, 92] and potential privacy concerns [93].

In research space, some works [9, 94] explore the possibility of using the smartphone's existing camera and IMU to track a VR controller, which demands heavy computation and high power consumption. Aura [10] is an inside-out electromagnetic controller tracking system achieving millimeter accuracy while it requires precise pretraining and dedicated hardware in both the HMD and the controller. There are some other techniques that track a person's hand or a worn device although not explicitly for controller tracking. Placing multiple sensors on the human body ensures precise tracking but is not suitable for daily use like VR [95, 96]. [16, 97] track the position of a smartwatch by combining IMU like a magnetometer with a kinematic model. Some systems also leverage Wi-Fi signal for gesture recognition [7] or capture human skeleton [98]. Such systems are often sensitive to wireless signal noises and magnetic interference, etc. Acoustic solutions [99–102] usually requires external sound beacons as reference points and fall short when tracking multiple devices at the same time. In contrast, we aim to design an accurate and robust VR controller tracking system using on-board sensors, which can be used as a generic controller tracking solution to mobile VR.

This chapter presents *VIU*, a novel inside-out mobile VR controller tracking system without using external beacons. Based on anatomical models of arm joints, the controllers' orientations as well as pairwise distances among the two controllers and the headset are strongly coupled to the controllers' positions. Inspired by this key idea, *VIU* fuses inertial sensing and ultrasonic ranging to conduct positional tracking of the controllers. However, We face critical challenges when realizing the above idea. (1) Compared with correlation-based methods [54, 103], frequency-modulated continuous wave (FMCW) has been proved to be more effective for ultrasonic ranging [104]. However, as FMCW assumes that a transmitter and a receiver share the same clock, synchronization remains an unsolved problem when applying FMCW on separate devices. Solutions such as Bluetooth and Wi-Fi, fail in this case — even a synchronization error of 1ms will cause a distance offset of 0.34m. (2) When playing VR games, a user's two hands can move as fast as 2m/s, which introduces non-negligible Doppler shifts in the received sound signals and affects the distance estimates. (3) With the controller orientations and the pair-wise distances, it is challenging to effectively fusing these measurements that may contain occasional noises to generate accurate location estimates in real time.

We tackle these challenges as follows. (1) We propose a novel two-way FMCW (T-FMCW) scheme to avoid clock synchronization and concurrently measure three pairwise distances among the devices. Since FMCW measures the difference in frequency between two identical signals, T-FMCW obtains two frequency differences and derives them into two time differences where each contains the time offset between the two devices with different signs. The sum of these two time differences eliminates the time offset and results into the distance without synchronization. (2) We take the Doppler effect into consideration and re-derive the FMCW algorithm under the condition where both the transmitter and the receiver are moving. Based on the new algorithm, the Doppler shift is detected via an additional sinusoid signal and the FMCW peak frequency is corrected accordingly. (3) We systematically design a statistical model to infer the probabilities of candidate locations obtained from a kinetic model. The orientations and the distances are combined through this model and used to estimate the real-time positions of the two controllers through a particle filter, which leverages temporal consistency.

Based on these innovative technologies, we implement a real-time end-to-end prototype (Figure 4.1) and systematically evaluate its performance. Since our algorithms are software-based and we want to focus on demonstrating their validity, we skip hardware implementation and use a Samsung Galaxy S10 as the HMD and two LG G7's as the controllers. The three devices are connected through Wi-Fi for data exchange. All the devices sample the IMUs (e.g. accelerometer, gyroscope and magnetometer) to calculate the orientations and meanwhile continuously emit ultrasound (16-22.5kHz) chirp signals of 1.5kHz bandwidth. To distribute the workload, each device takes charge of calculating the distances between one pair of devices by conducting Doppler-corrected T-FMCW on received sound signals. Finally, the controllers' poses and inter-device distances are all uploaded to the HMD for final location prediction by a particle filter. We evaluated our system with 5 users in various real-world settings including natural movements. A subgroup of 4 further tested the system's usability in pre-designed VR games, which shows *VIU*'s advantages over existing solutions. *VIU* can support a position update rate of about 30Hz and an accuracy of less than 10cm, which is adequate for most use cases. Since only using commodity hardware (e.g. IMU's, speaker, and microphone), *VIU* makes it possible to experience interactive VR anywhere with low-cost lightweight controllers. Our contributions are as follows:

- Designing a 6DoF controller tracking system suitable for mobile VR headsets, without using cameras or external beacons.

- To our knowledge, the first ultrasonic ranging system measuring the distances between two fast-moving objects. We propose (1) an innovative two-way FMCW to avoid clock synchronization and (2) correct the FMCW frequency peaks by taking the Doppler shift into consideration.
- Using particle filter to fuse the orientation and distance measurements into the position estimates.
- Systematic evaluation regarding tracking accuracy and runtime latency. Extensive user studies demonstrating the use of *VIU* in various VR applications, e.g. object selecting and slingshot.

4.2 Related Work

4.2.1 Positional Tracking in Virtual Reality

High-quality HMD and controller tracking is always a common challenge that all VR systems are facing. Most of the current solutions are based on vision, inertial sensing or a combination of both.

Optical tracking. Since the release of the Oculus DK2 in 2014 [105], outside-in optical tracking [106, 107] is widely utilized by several mainstream products, e.g. original Oculus Rift and PlayStation VR. It relies on tracking sensors placed in a stationary location to observe the tracked device with a set of sensors/markers on it. Constellation [108] is a outside-in system used on Oculus Rift. Rift embeds IR LEDs onto each tracked device and uses external Oculus sensors wiredly connected to a powerful host PC to detect the device movement. The process is computationally demanding since a large number of pixels need to be transmitted and processed, and meanwhile suffers from the need for heavy calibration and precise synchronization among the sensors and devices [109]. These prevent similar solutions to be applied on mobile VR HMDs and controllers.

SteamVR Lighthouse [90,110,111] used by HTC Vive is another popular tracking solution. Multiple base stations are not sensors but serve as reference points by constantly emitting wide-angle IR laser beams across a room. Each tracked device (i.e. HMD or controller) is equipped with photodiodes to measure the time of flight (ToF) of each laser beam. From the ToF the device determines its own position in the room. Lighthouse is an inside-out tracking system. In contrast to outside-in systems, the cameras/sensors used to determine a device's locations are placed on the device itself instead of stationary beacons. However, this approach relies on prior setup, and requires line of sight (LoS) thus is vulnerable to occlusion.

Another inside-out vision-based approach is simultaneous localization and mapping (SLAM) [15]. Many newer VR kits (e.g. Oculus Quest, Oculus Rift S and HTC Vive Focus) have multiple built-in cameras on their HMDs. Based on the frames obtained from these cameras, an HMD is able to build a map of the surrounding, locate itself by matching feature points with the map and meanwhile track its accessory controllers which are usually equipped with IR LEDs. To improve accuracy, tracking systems based on SLAM always combine data from inertial sensors like accelerometer and gyroscope. Although SLAM outperforms the above methods by getting rid of external beacons and trivial setup, it still does not work in dark environments and when the controller is occluded by the human body [91]. Moreover, using cameras in private environments presents privacy concerns among users although it is claimed that the captured frames are not uploaded to the server [93].

Inertial tracking. Due to hardware limitations, most mobile VR systems, for example Google Daydream View and Samsung Gear VR, track its HMD and controllers using inertial sensing data obtained from accelerometer, gyroscope and magnetometer [112]. Lack of any true 3D position information, these controllers act like a laser pointer with limited positional movement, which affects their usage in complex and interactive application scenarios like shooting, archery and using virtual tools, etc. Integrating noisy acceleration and gyroscope drift introduce accumulated errors to the tracking process, which is hard to be corrected using the unstable magnetic field in indoor environments.

4.2.2 Acoustic Tracking and Ranging

Acoustic tracking and ranging in VR/AR can be traced back to 1965, when the first prototype VR system "the Ultimate Display" [113] is introduced. By transmitting ultrasonic continuous waves from transmitters attached to the HMD to receivers mounted in a square array in the ceiling, the system keeps track of motions from the measured phase shift for each ultrasonic path. Recently in early 2019, TDK announces Chirp SonicTrack inside-out ultrasonic controller tracking solution for standalone VR [114,115]. The CH-101 sound sensors in both HMD and the controllers are omni-directional and providing relative position information in a wide field of view (FoV). Immigrating the same technique to mobile VR kits may face hardware issues, e.g. the fixed locations and spotty quality of phone speakers/microphones.

There are other prior work on acoustic tracking and ranging. [103] measures distances between two static devices by having both of them transmit and receive audible beeps to cancel out propagation delay and clock difference. [54] adopts and improves [103] for mobile gaming, also using audible sound. [99, 104] apply distributed FMCW with inaudible sound to track moving devices and meanwhile consider Doppler shifts during the motion, which requires multiple speakers on a single device. All these solutions cannot be directly borrowed and adopted to simultaneously track a mobile VR HMD and its controllers because of additional issues like sharing the narrow inaudible sound frequency band (from 15 kHz to 23 kHz) [116] among three devices.

4.3 System Overview

VIU is an inside-out tracking system which aggregates inertial sensing and ultrasonic ranging to estimate the positions of a pair of controllers in mobile VR system. Our target usage scenario is that: A user sits or stands at a spot (since mobile VR

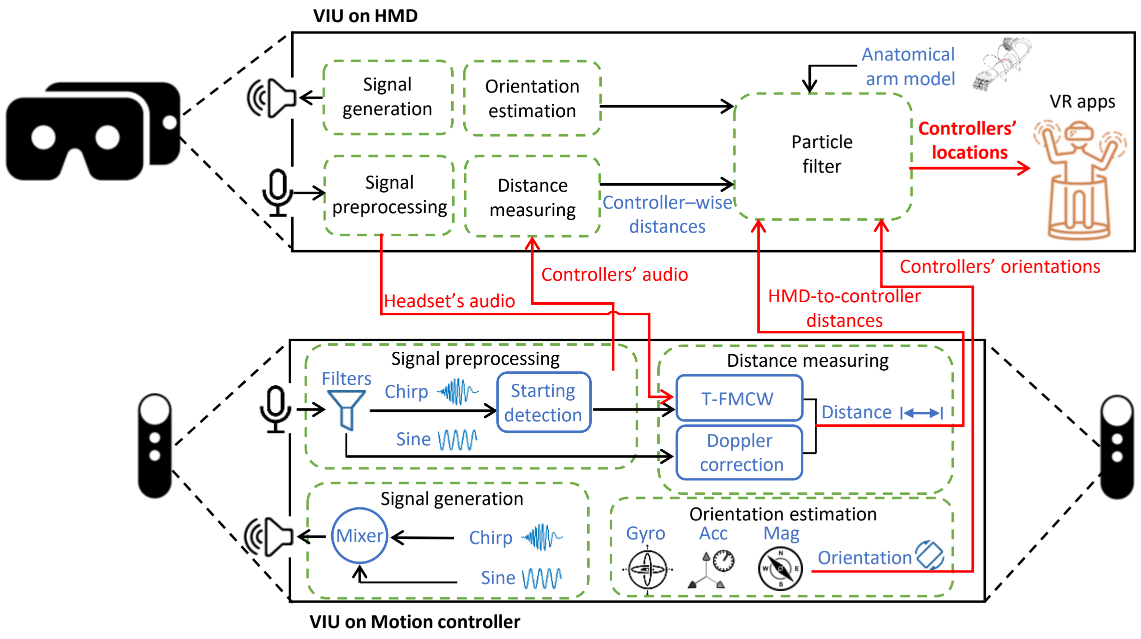


Fig. 4.2. System overview.

obstructs the view of the outside world, it's not safe to walk while wearing it) with a controller held in each hand. The user can turn its head to change the view and move the two hands to interact with the VR scene.

Figure 4.2 illustrates a functional overview of *VIU*. The end-to-end system consists of three devices, i.e. one HMD and two motion controllers. Each device is equipped with inertial sensors and a pair of general-purpose speaker and microphone. The two controllers are connected to the HMD through Wi-Fi. The flow of operation composes of the following steps. (1) To measure three pair-wise distances among the devices, each device continuously emits an ultrasonic tone on different frequency bands, which is a sum of a chirp signal and a sinusoid signal, and meanwhile records received audio. To decentralize workload, the devices exchange their audio and each will take on the distance measuring between one pair. (2) Each device pre-processes the recorded audio from two devices. It extracts the signal components and detects the starting points of the latest chirp signals using a novel FMCW-based method (Section 4.4.1).

(3) After fetching the signals, *VIU* conducts a two-way FMCW (T-FMCW) (Section 4.4.1) on the chirps to estimate the distance without synchronizing the transmitter and the receiver. Since the speed of natural hand movement can be as high as 2m/s [117], *VIU* utilizes the sinusoid signals to estimate Doppler shifts and corrects the distance estimates accordingly (Section 4.4.1). (4) Based on an anatomical arm model, *VIU* introduces a particle filter (Section 4.4.3) on the HMD to fuse the devices' poses obtained from IMU readings (e.g. accelerometer, gyroscope and magnetometer) and the triangular distances to derive the controllers' positions. It computes Bayesian probabilities as the weights of possible positions and takes temporal relationship with tracking history into consideration. The real-time tracking results are finally used as input to various VR applications for controlling and interacting.

4.4 System Design

In this section, we first present the design of our multi-device acoustic ranging algorithm. It includes how we introduce a two-way FMCW to avoid synchronization and how we correct Doppler shifts caused by the transmitter and the receiver's concurrent movement. Then we describe the mapping between controllers' locations and their orientations based on a kinematic arm model. Finally, we introduce a particle filter which effectively fuses all the measurements into Bayesian probabilities and derives the controllers' positions.

4.4.1 Multi-device Acoustic Ranging

Motivation

Although there are many prior works on acoustic ranging, few fits into the scenario of tracking VR motion controllers. Correlation-based methods [54, 103] relies on the integrity of received signals thus are not robust to environment noises and signal distortion caused by hardware imperfection. They are hard to be adopted to ultrasounds, especially with general-purpose commodity speakers and microphones. FMCW is more accurate and reliable using the same amount of bandwidth. When used like a radar [14], it tracks an object through sound signals reflected off it. It not only requires the sound transmitter and the receiver to share the same clock, but also suffer from complex signal reflections caused by multiple moving human body parts. To support separate and unsynchronized transmitter and receiver, prior works [99,104] require multiple synchronized speakers for calibrating initial distance and then measure distance changes instead of absolute distances. However, this method is not suitable for our scenario since the speakers on different devices cannot be synchronized. Moreover, as each speaker occupies a certain frequency range, it is hard to assign the narrow ultrasound band to more speakers and extend these methods to simultaneously track multiple objects.

This motivates us to develop an ultrasonic ranging approach which: (1) simultaneously measures the absolute distances (not distance changes) among *multiple moving devices* using limited ultrasound band; (2) solves the *synchronization issue* without using extra hardware or external beacons; (3) is *robust* to high mobility. From this end, we design a novel ultrasonic ranging algorithm leveraging two-way Doppler-assisted FMCW.

Existing FMCW-based ranging

We first review FMCW with a synchronized transmitter and receiver [118]. The transmitter emits periodic chirp signals whose frequency sweeps linearly over time as $f = f_{min} + \frac{Bt}{T}$, where f_{min} is the starting frequency, B is the bandwidth, and T is the signal period. By integrating the frequency, the transmitted signal is

$$v_t = \cos(2\pi f_{min} t + \frac{\pi B t^2}{T}). \quad (4.1)$$

Thus the received signal is $v_r = \alpha \cos(2\pi f_{min}(t - \tau) + \frac{\pi B (t-\tau)^2}{T})$, where α is the amplitude attenuation and τ is the propagation delay. By mixing the transmitted and the received signal, we get the signal $v_m = v_t \cdot v_r$. The mixed signal is then simplified by filtering out the lower-frequency component. So v_m becomes

$$v_m = \alpha \cos(2\pi f_{min}\tau + \frac{\pi B(2t\tau - \tau^2)}{T}). \tag{4.2}$$

Ignoring the transmitter and the receiver's moving speed (it's very small compared with sound speed), the propagation delay can be derived based on the first peak f_p in the spectrum of the mixed signal according to

$$\tau = \frac{f_p T}{B}. \tag{4.3}$$

Meanwhile we can get the distance R based on

$$R = c \cdot \tau = \frac{f_p c T}{B}. \tag{4.4}$$

Limitation. In our usage scenario, the HMD and the controllers are separate devices without a shared clock, which makes it difficult to directly adopt the traditional FMCW. Even if the clock synchronization error is just 1ms, the distance error can be as large as 0.34m. Also, the traditional FMCW ignores the effect of movement thus introduces non-negligible errors when tracking the fast-moving controllers. Therefore, we develop a two-way FMCW to avoid synchronization and take Doppler shift into account to correct the FMCW peak frequency. Below we elaborate on these techniques.

Starting Point Detection on Chirps

Without loss of generality, we consider one pair-wise distance from now. To get an accurate result, FMCW requires the chirps heard by the transmitter and the receiver to be almost complete and to have enough overlap (i.e. small propagation delay) with each other. Thus, we need to first detect the starting points of the chirp signals. Correlation has been used in some works [99] but it generates multiple peaks with similar amplitudes on noisy signals (shown in Figure 4.3 (c)). Therefore, we propose an FMCW-based method for starting point detection. This procedure is

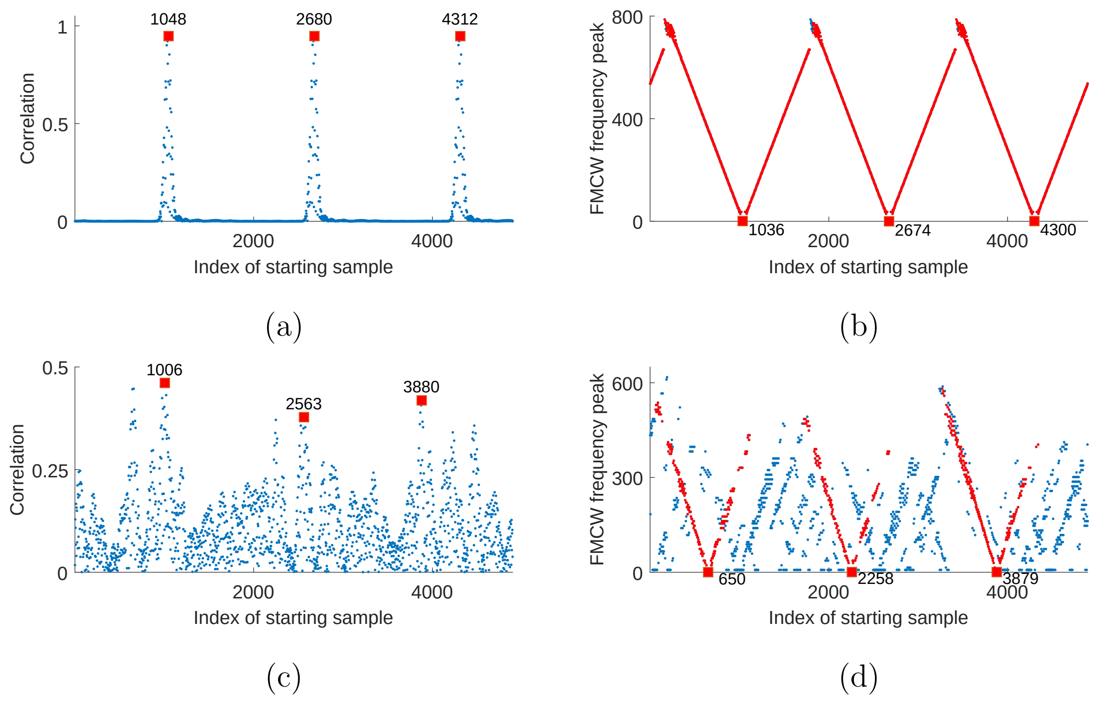


Fig. 4.3. Detecting signal starting point using correlation ((a) and (c)), and FMCW frequency peaks ((b) and (d)). When used on noisy signals ((c) and (d)), correlation generates multiple peaks with similar amplitudes while FMCW obtains a more reliable result with constant intervals between the starting points.

automatically conducted immediately after the system is turned on — when a user is wearing the HMD and naturally holding the controllers in its two hands but hasn't started moving. After passing through the recorded signals through a seventh-order band-pass Butterworth filter, we shift the template along with the signal and perform FMCW, the FMCW frequency peaks form a "V" shape as shown in Figure 4.3 (b). Based on Equation 4.3, the slope of the "V" shape is $\frac{B}{F_s T}$, where $F_s (= 48\text{kHz})$ is the sampling rate,. By fitting the frequency peaks into this "V" shape line, we choose the time with the most inliers as the best fit and use it as the starting point. When applied to the same noisy signal as in Figure 4.3 (c), our method can achieve a more stable detection result compared with correlation. As each chirp signal has $L(= 1632)$ samples, the consecutive starting points have the index intervals of 1608 and 1621 with the FMCW-based method (shown in Figure 4.3 (d)), in contrast to 1557 and 1317 with correlation.

Note that the starting point detection is a one-time effort. Afterward, we will fetch the following signals every L samples. Also, when actually using the above results, we intentionally chop the signals recorded by the receiver at 200 samples before the detected starting point. This is to create a "propagation delay" between the signals from the two devices. It ensures at least a 75% signal overlap for FMCW when the distance between the devices is in a range of 0-2m and meanwhile maintains a constant temporal relationship which is critical in the two-way FMCW. Finally, although more accurate, the FMCW-based method approximates the starting points with some possible errors. We will explain how the two-way FMCW utilizes the 200-sample offset and automatically compensates this approximate starting point in Section 4.4.1.

Two-way FMCW

After extracting the signals to be processed, we elaborate on the two-way FMCW (T-FMCW) to estimate distances under the following assumptions. (1) We assume the speaker and the microphone on the same device are co-located and ignore the distance between them - this assumption can be easily relaxed using the methods in [103]. (2) In this subsection, we assume that the moving speed of the two devices is close to zero and will explain how we deal with fast movement in Section 4.4.1.

The basic procedure of T-FMCW is illustrated in Figure 4.4. Two devices, named A and B, emit chirp signals 1 and 2, respectively. Each ultrasonic chirp has a sampling frequency of $F_s = 48\text{kHz}$, a signal length of $L = 1632$ samples, a signal duration of $T = L/F_s = 0.034\text{s}$) and a frequency band of $B = 1500\text{Hz}$. Therefore each device can hear two kinds of chirps in its recorded audio. Two time lines are shown in the figure presenting the local time of each device. We denote the actual local time of chirp 1 arriving at device A and B as t_{A1} and t_{B1}, respectively. Similarly, the actual local

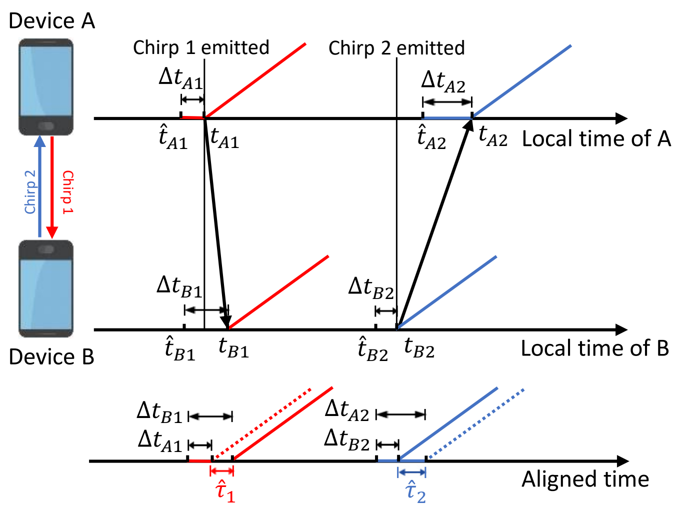


Fig. 4.4. Example event sequences in two-way FMCW.

time of chirp 2 arriving at device A and B is represented by t_{A2} and t_{B2}, respectively. Since these actual arriving times are unknown to us, we employ the aforementioned method to roughly detect the approximate starting point of each chirp (i.e. $\hat{t}_{A1}$, $\hat{t}_{B1}$, $\hat{t}_{A2}$, and $\hat{t}_{B2}$). With a pair of starting times, e.g. $\hat{t}_{A1}$ and $\hat{t}_{B1}$, two signal segments of length L are extracted from the audio recorded by A and B and are aligned at their first samples. Figure 4.4 shows how the extracted signals from the two devices are aligned, where the dashed lines represent the chirps from A and the solid lines are from B. By multiplying the two chirp signals in each pair, i.e. the signals starting from $\{\hat{t}_{A1}, \hat{t}_{B1}\}$ and $\{\hat{t}_{A2}, \hat{t}_{B2}\}$, we can get the FMCW frequency peaks $\hat{f}_{p1}$ and $\hat{f}_{p2}$, respectively. Based on Equation 4.3, the "propagation delay" ($\hat{\tau}_1$ and $\hat{\tau}_2$ in Figure 4.4) introduced by the alignment can be inferred as

$$\hat{\tau}_1 = \frac{\hat{f}_{p1}T}{B} \tag{4.5}$$

$$\hat{\tau}_2 = \frac{\hat{f}_{p2}T}{B}. \tag{4.6}$$

From the temporal relationship in Figure 4.4, we know that

$$t_{A1} - \hat{t}_{A1} + \hat{\tau}_1 = t_{B1} - \hat{t}_{B1} \tag{4.7}$$

$$t_{A2} - \hat{t}_{A2} - \hat{\tau}_2 = t_{B2} - \hat{t}_{B2}. \tag{4.8}$$

Note that the signs before the two $\hat{\tau}_i$ values are opposite. The reason is that in the case shown by Figure 4.4, the signals in the two pairs have an opposite relative temporal relationship, i.e. chirp 1 fetched from device A precedes the one from device B and it is the opposite regarding chirp 2. Since FMCW cannot reflect which signal is at the front, we define a directional "prorogation delay" as

$$\hat{\tau}_i' = \begin{cases} \hat{\tau}_i, & \text{if aligned chirp } i \text{ from } A\text{'s audio precedes the one from } B \\ -\hat{\tau}_i, & \text{otherwise.} \end{cases} \tag{4.9}$$

Now we discuss how we decide the signs in practical use. Recall that the signals recorded by the receiver are chopped at 200 samples before the approximate starting point that is detected. We observed that: when the user is naturally holding the controllers at the beginning, the pair-wise distances among the HMD and the controllers are always in a range of 0.5–1m; when using the VR system, the distances are in a range of 0–2m. That means, the distances can change in a range of -1–1.5m, which can cause a signal offset of -140–210 samples given $\tau = R/c$. Thus, during the movement, the signals received by the transmitter is always 60–410 samples in front of the other signal.

So we get

$$t_{A1} - \hat{t}_{A1} + \hat{\tau}_1' = t_{B1} - \hat{t}_{B1} \tag{4.10}$$

$$t_{A2} - \hat{t}_{A2} + \hat{\tau}_2' = t_{B2} - \hat{t}_{B2}. \tag{4.11}$$

We denote the distance estimated from chirp 1 and chirp 2 as R_1 and R_2, respectively; the actual interval between the arrival of each chirp at the two devices as τ_1' and τ_2'; and the time difference between the two devices as Δt (i.e. local time on device B is ahead of A by Δt). Thus, the distance between the two devices can be derived by

$$\begin{aligned} D &= \frac{1}{2}(R_1 + R_2) \\ &= \frac{c}{2}(\tau_1 + \tau_2) \\ &= \frac{c}{2}[(t_{B1} - t_{A1} - \Delta t) + (t_{A2} + \Delta t - t_{B2})] \\ &= \frac{c}{2}[(t_{B1} - t_{A1}) + (t_{A2} - t_{B2})]. \end{aligned} \tag{4.12}$$

By plugging in Equation 4.10 and Equation 4.11, we get

$$D = \frac{c}{2}[(\hat{t}_{B1} - \hat{t}_{A1} + \hat{\tau}'_1) + (\hat{t}_{A2} - \hat{t}_{B2} - \hat{\tau}'_2)]. \tag{4.13}$$

Finally, by combining Equation 4.13 with Equation 4.5, 4.6 and 4.9, the distance can be computed as following

$$\begin{aligned} D &= \frac{c}{2}[(\hat{t}_{B1} - \hat{t}_{A1} + \hat{\tau}_1) + (\hat{t}_{A2} - \hat{t}_{B2} + \hat{\tau}_2)] \\ &= \frac{c}{2}[(\hat{t}_{B1} - \hat{t}_{A1} + \frac{\hat{f}_{p1}T}{B}) + (\hat{t}_{A2} - \hat{t}_{B2} + \frac{\hat{f}_{p2}T}{B})]. \end{aligned} \tag{4.14}$$

Note that it does not rely on any time information between the two separate devices, so that we can conduct T-FMCW without clock synchronization. Moreover, the final distance estimate is actually independent of the approximate starting times. Minor shifts of the starting points will result in a corresponding minor difference in FMCW frequency peaks. In other words, the inaccuracy in starting point estimation will be compensated by the FMCW results and won't harm the distance measurement.

Propagation delay on consecutive chirps. When extending the above algorithm to the actual cases with consecutive chirps, one intuitive concern is if this method is robust to propagation delay. For example, as shown in Figure 4.5, at one time of periodic distance computing, the latest chirp is not completely received by device A due to acoustic propagation delay. Since we always use the latest signals for distance computing, this causes a mismatch between the chirp 2 signals from the two devices. T-FMCW would wrongly choose the previous chirp 2 received by A and use $\hat{t}'_{A2}$ instead of $\hat{t}_{A2}$. Since $\hat{t}_{A2}$ is the only term in Equation 4.13 affected by the

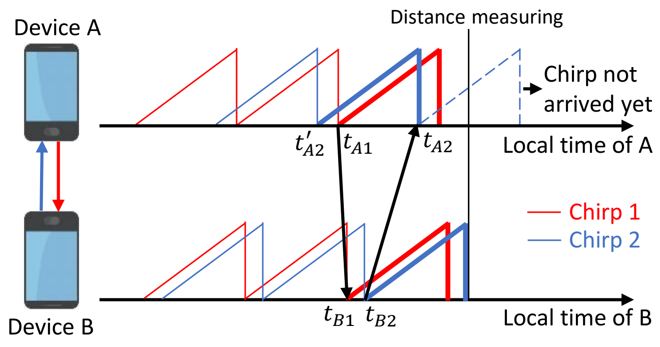


Fig. 4.5. Ranging error when a chirp is not completely received.

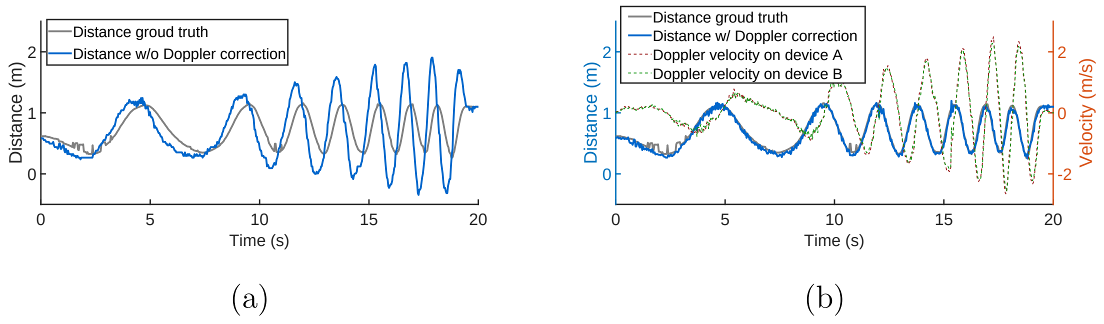


Fig. 4.6. Distance measurement (a) before Doppler correction, (b) after Doppler correction.

mismatch and it is about T ahead of the actual value, this introduces a distance error of about 5.8m ($\Delta D \approx cT/2$). With a prior knowledge that the distance between two devices (i.e. the HMD and one of the controllers, or the two controllers) is always less than 2m, the unreasonable distance estimates can be easily detected and discarded.

Doppler Correction

So far we assume that the velocity of the devices is close to 0. However in practice, fast hand movement can greatly affect the distance measurement. Let's move to how the assumption is relaxed. We did a simple experiment by repeatedly moving two hands close to and away from each other and the movement gradually becomes faster over time. In Figure 4.6(a), the gray line shows the ground-truth distances between the two hands and the blue line is the measured distances using the aforementioned schemes in Section 4.4.1. The measurements fluctuate more and more sharply as the hands move faster, although the hands actually move in a fixed range.

To further examine the Doppler effect in this situation, we generalize it as the following. During a short period, two objects A and B move at v_1 and v_2, respectively; and they share a global clock (synchronization solved in Section 4.4.1). When B transmits a chirp signal at time t, A will receive it at

$$t' = t + \frac{R + (v_1 - v_2)t}{c - v_1}. \tag{4.15}$$

Thus,

$$t = t' - \frac{R + (v_1 - v_2)t'}{c - v_2}. \tag{4.16}$$

Plugging Equation 4.16 into Equation 4.1, the received signal at A can be represented as

$$v_r = \alpha \cos(2\pi f_{min}(t - \frac{R + (v_1 - v_2)t}{c - v_2}) + \pi B(t - \frac{R + (v_1 - v_2)t}{c - v_2})^2/T). \tag{4.17}$$

Similar to Equation 4.2, the mixed signal is generated and simplified as

$$v_m = \alpha \cos(2\pi f_{min} \frac{R + (v_1 - v_2)t}{c - v_2} + \frac{\pi B}{T}[2t\frac{R + (v_1 - v_2)t}{c - v_2} - (\frac{R + (v_1 - v_2)t}{c - v_2})^2]). \tag{4.18}$$

By differentiating the phase of v_m, we get the frequency of v_m as

$$f_p = \frac{1}{2\pi}\frac{\partial Phase(v_m)}{\partial t} = \frac{f_{min}(v_1 - v_2)}{c - v_2} + \frac{BR}{(c - v_2)T} + \frac{2B(v_1 - v_2)t}{(c - v_2)T}. \tag{4.19}$$

Thus the distance between A and B can be obtained by

$$R = \frac{T}{B}[f_p(c - v_2) - f_{min}(v_1 - v_2)(1 + \frac{2Bt}{f_{min}T})] \tag{4.20}$$

$$= \frac{f_p cT}{B} - \frac{f_p v_2 T}{B} - \frac{f_{min}(v_1 - v_2)T}{B} - 2(v_1 - v_2)t. \tag{4.21}$$

Since the speed of natural hand movement is much smaller than the sound speed, we approximate $c - v_2$ to c for the first component in Equation 4.20. In the last component in Equation 4.20, B is a small portion of f_{min} while t is a small portion of T. So we simplify the distance measurement as

$$R = \frac{f_p cT}{B} - \frac{f_{min}(v_1 - v_2)T}{B}. \tag{4.22}$$

Regarding the experiment above, we use Figure 4.7 to illustrate the amplitude order of the four components in Equation 4.21. $\frac{f_p cT}{B}$ and $\frac{f_{min}(v_1-v_2)T}{B}$ dwarf the distance errors caused by the other two terms, which demonstrates the simplification in Equation 4.22.

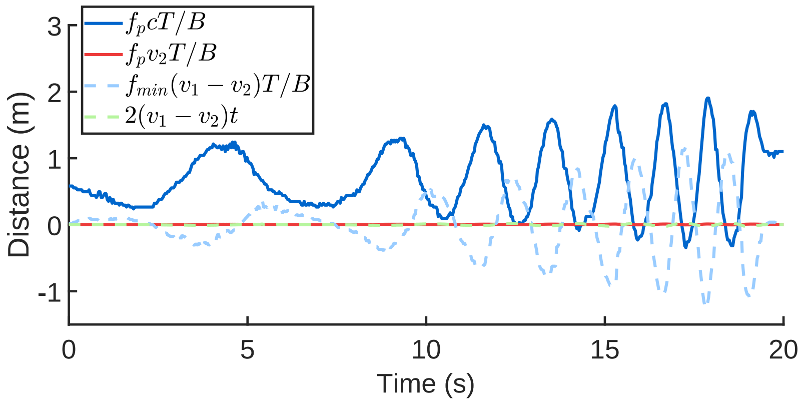


Fig. 4.7. Distances contributed by different components in Equation 4.21.

Based on the previous analysis, we need to know the relative velocity for Doppler correction. Thus, besides the chirp signals, we let each device play continuous ultrasonic sinusoid signals at different frequency F with a 0.5 kHz guard band on both sides. The recorded audio is passed through a seventh-order Butterworth band-pass filter to get the sine waves transmitted from the peer in distance measurement. We use the filtered sine waves in the same extracted chirp duration to perform Fast Fourier Transform (FFT). Based on the frequency peak found in FFT, we can easily get the Doppler frequency shift F^s and the relative velocity $v_1 - v_2 = (F^s/F)c$. Finally, we have

$$R = (f_p - f_{min}\frac{F^s}{F})\frac{cT}{B}. \tag{4.23}$$

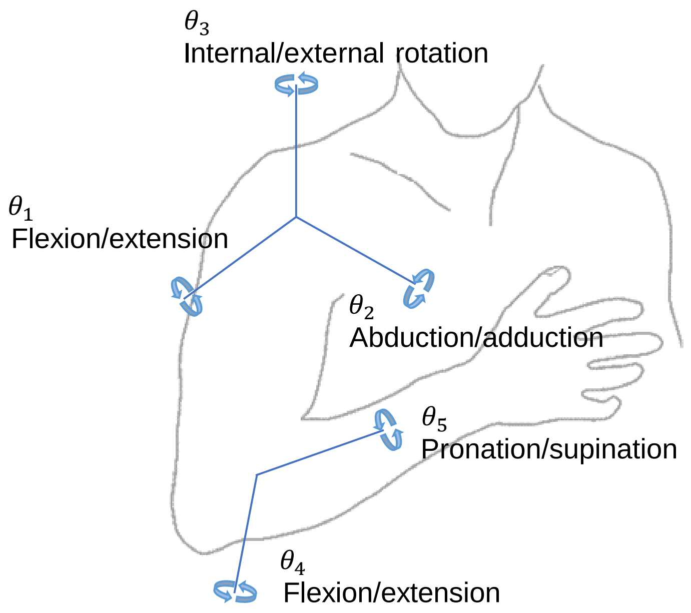


Fig. 4.8. 5-DoF kinematics arm model.

Unlike the prior works who aim to estimate the velocity [100,104], we infer how much the FMCW frequency peaks deviate due to the Doppler effect and correct the distance accordingly. Given a relative moving speed between the two devices, an FMCW frequency peak shifts proportionally to the original chirp frequency. Figure 4.6(b) shows the distance measurements after the Doppler correction, which is consistent with the ground truth.

4.4.2 Mapping Location to Orientation

Human arm movements are highly constrained in a range according to the anatomical arm model [119]. As shown in Figure 4.8, the arm model consists of 5 joint DoFs. θ_{1-3} represents the 3 DoFs of the shoulder and θ_{4-5} represents the 2 DoFs of the elbow. The DoFs of the wrist are not contained in our arm model since the controller is tightly attached to it. For a given wrist location, the corresponding wrist orientation (i.e. the controller orientation) can only vary in a limited range. Thus, if we define the locations of the two controllers as the state of a particle and represent it

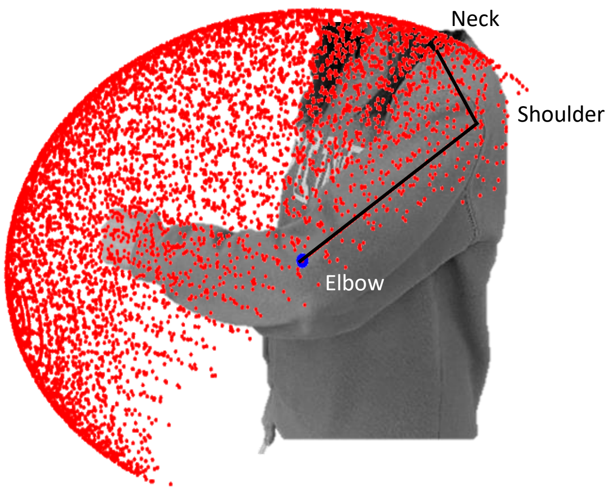


Fig. 4.9. An example elbow location and the end points of its corresponding forearm vectors.

with a 6D tuple, we can compare the possible space of controller orientations with the orientation measurements to infer the likelihood of this particle state.

With 6D tuples, we need a large number of particles to cover the entire location space. But actually with more prior knowledge, we can reduce the state dimension for each particle. With a known length of the upper arm, each elbow location is determined merely by θ_1 and θ_2. And with a known length of the lower arm and the lower arm direction computed from the measured wrist orientation, a wrist location can be inferred by shifting the corresponding elbow location along the lower arm vector. In this way, the state of one particle is defined as the locations of the two elbows and is represented by a 4D tuple ($A_i =< \theta^i_{l,1}, \theta^i_{l,2}, \theta^i_{r,1}, \theta^i_{r,2} >$) instead. Here in Figure 4.9 we show an example elbow location, along with the corresponding wrist orientations. For visualizing simplicity, we omit one dimension of the wrist orientations and represent them by the end points of the corresponding forearm vectors. We can see that the possible space of wrist orientations is reasonably restricted.

Now, we mathematically model the mapping from each elbow location to the possible wrist orientations. Based on some medical works [119–121] and the average range of motion (ROM) for each joint angle summarized in [16], we enumerate all possible combinations of the 5 joint angles with a resolution of 7.5 degrees for each DoF. From each combination, we can derive an elbow location loc_{elbow} $(= f(\theta_1, \theta_2))$ and a wrist orientation ori_{wrist} $(= h(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$ based on the Denavit-Hartenberg transformation [122]). We further convert the wrist orientations to a uniform partition of SO(3) using the Hopf Fibration [123,124], to avoid repeated counting of orientations around the polar regions. By grouping the combinations according to loc_{elbow}, we can obtain a mapping $HashMap < loc^j_{elbow}, < \mathbb{O}^j, \mathbb{P}^j >>$, where $\mathbb{O}^j$ is the list of the possible wrist orientations ori_{wrist} and $\mathbb{P}^j$ is the corresponding probabilities for each ori_{wrist}.

4.4.3 Fusion using Particle Filter

With the intermediate results from preceding modules: pair-wise distance mesurements among the three devices $dist^m =< dist^m_{l,h}, dist^m_{r,h}, dist^m_{l,r} >$ where l, r and h denotes *left controller*, *right controller* and *HMD*, respectively; measured controller orientations $ori^m =< ori^m_l, ori^m_r >$ obtained from inertial sensors using the technique in [125,126]; an arm model containing the mapping from each elbow location to the correponding controller orientations with the possibilities $Map_l = \{loc^j_{elbow,l}, < \mathbb{O}^j_l, \mathbb{P}^j_l >\}$ and $Map_r = \{loc^j_{elbow,r}, < \mathbb{O}^j_r, \mathbb{P}^j_r >\}$, $1 \leq j \leq M$. *VIU* further fuses these values via a particle filter to estimate the controller locations. We denote the state of a particle as $A_i =< loc^i_{elbow,l}, loc^i_{elbow,r} >$ and the set of particles as $\mathbb{A} = \{A_1, A_2, ..., A_N\}$.

Particle weight assignment. Initially, the $N(= 100)$ particles are randomly distributed in the space of possible elbow locations. To update particle locations, we need to calculate particle weights according to the measured orientations and distances. Based on Bayes' theorem, the probability of ith particle is derived as

$$\begin{aligned}
&P(A_i|dist^m, ori^m)\\
=&\frac{P(dist^m, ori^m|A_i)P(A_i)}{P(dist^m, ori^m)}\\
=&\frac{\sum_{ori_l^{i,u}\in\mathbb{O}_l^i, ori_r^{i,v}\in\mathbb{O}_r^i} P(dist^m, ori^m|A_i, ori_l^{i,u}, ori_r^{i,v})P(ori_l^{i,u}, ori_r^{i,v}|A_i)P(A_i)}{P(dist^m, ori^m)}\\
=&\frac{\sum_{ori_l^{i,u}\in\mathbb{O}_l^i, ori_r^{i,v}\in\mathbb{O}_r^i} P(dist^m|A_i, ori_l^{i,u}, ori_r^{i,v})P(ori^m|A_i, ori_l^{i,u}, ori_r^{i,v})P(ori_l^{i,u}, ori_r^{i,v}|A_i)P(A_i)}{P(dist^m, ori^m)}\\
=&\frac{\sum_{ori_l^{i,u}\in\mathbb{O}_l^i, ori_r^{i,v}\in\mathbb{O}_r^i} P(dist^m|dist^{i,u,v})P(ori^m|ori^{i,u,v})P(ori_l^{i,u}, ori_r^{i,v}|A_i)P(A_i)}{P(dist^m, ori^m)},
\end{aligned} \tag{4.24}$$

where the tuples $dist^{i,u,v} =< dist_{l,h}^{i,u,v}, dist_{r,h}^{i,u,v}, dist_{l,r}^{i,u,v} >$, $ori^{i,u,v} =< ori_l^{i,u}, ori_r^{i,v} >$. Assuming the measured distances and orientations are independent, Equation 4.24 can be further derived using

$$P(dist^m|dist^{i,u,v}) = P(dist_{l,h}^m|dist_{l,h}^{i,u,v})P(dist_{r,h}^m|dist_{r,h}^{i,u,v})P(dist_{l,r}^m|dist_{l,r}^{i,u,v}), \tag{4.25}$$

$$P(ori^m|ori^{i,u,v}) = P(ori_l^m|ori_l^{i,u})P(ori_r^m|ori_r^{i,v}), \tag{4.26}$$

$$P(ori_l^{i,u}, ori_r^{i,v}|A_i) = P(ori_l^{i,u}|A_i)P(ori_r^{i,v}|A_i). \tag{4.27}$$

Since random measurement errors are always normally distributed [127], each term on the right side of Equation 4.25 fit into $\mathcal{N}(\mu_{dist} = 0, \sigma_{dist} = 0.1m)$ and each term on the right side of Equation 4.26 fits in to $\mathcal{N}(\mu_{ori} = 0, \sigma_{ori} = 12°)$. And each term on the right side of Equation 4.27 can be retrieved from the maps generated from the arm model. For example, to retrieve $P(ori_l^{i,u}|A_i)$, we first find the closest elbow location to $loc_{elbow,l}^i$ from Map_l (say $loc_{elbow,l}^p$) and then find the closest orientation to $ori_l^{i,u}$ from $\mathbb{O}_l^p$ (say $ori_l^{p,q}$). Thus the term value is the probability corresponding to that closest orientation, i.e. $P(ori_l^{i,u}|A_i) = P_l^{p,q}$. We omit $P(dist^m, ori^m)$ during the weight calculation since it is the same for all particles. The weights are then normalized across all particles for further processing, denoted by $\mathbb{W} = \{w_1, w_2, ..., w_N\}$. Overall, particles that can yield pair-wise distances and controller orientations that are closer to the measurements with high probability will have higher weights.

Moreover, along with the process of particle weight assignment, a weighted average of possible controller orientations (denoted as $< \overline{ori}_l^i, \overline{ori}_r^i >$) can also be computed based on the probability of each pair of the two orientations $< ori_l^{i,u}, ori_r^{i,v} >$. It will be used to transit the elbow location estimates to the corresponding controller locations.

Position estimation. The distribution of the particles and their weights reflect the likelihood of the real elbow positions. We use a weighted average of all particles as the two elbows' locations. We also compute a weighted average of the weighted-average wrist orientations and then derive the pointing directions of the two forearms. As each controller is tightly attached to the wrist, the controller locations are generated by shifting the elbow locations along the forearm vectors.

Particle re-sampling and state transition. With the updated particle weights, the particles are re-sampled accordingly. Among all the current particles, we randomly select one using the weights as the probability distribution. The re-sampling process leverages inverse transform sampling [128] and is as follows:

1. Generate the cumulative distribution function (CDF) for the particles where $F_{\mathbb{A}}(x = A_i) = \sum_{j<i} w_j$.
2. Compute the inverse of the CDF, i.e. $F_{\mathbb{A}}^{-1}(x)$.
3. Generate a random number r from the standard uniform distribution in the interval $[0, 1]$.
4. Find $A_i = F_{\mathbb{A}}^{-1}(r)$ and use A_i as one new particle candidate.
5. Repeat all previous steps for N times to get a new set of particle candidates.

Via this re-sampling process, the particles with smaller weights are gradually discarded. Then taking the continuous movement into consideration, we transit the particle states by randomly move each selected particle according to normal distributions of θ_1 and θ_2 ($\sigma = 90°/s \cdot \mathfrak{t}$, where $\mathfrak{t}$ is the time passed since last location update). With this particle filter, *VIU* fuses the distance estimates, the controller orientations and the arm model into the accurate and smooth controller locations.

4.5 Evaluation

4.5.1 Implementation and methodology

We implement *VIU* using Samsung Galaxy S10 as the HMD and two LG G7 ThinQ's as the controllers. Each G7 runs a controller emulator written in Java, and the S10 runs a modified version of Google VR service emulator [129] in Java and a Unity app in C#. Each device continuously plays ultrasonic tones containing chirps and sine waves. The speaker volume is set to 80% of its maximum to ensure sound quality. Since the sound between 14.5 kHz and 22.5 kHz is virtually inaudible to most people [99, 116], we assign this frequency band among the three devices as follows: The right controller occupies 16 kHz to 17.5 kHz for chirps and 14.5 kHz for sine waves; similarly, the HMD occupies $18.5 - 20$ kHz and 15 kHz; and the left controller occupies $21 - 22.5$ kHz and 15.5 kHz. The length of each mixed signal is set to 1632 samples to ensure no phase misalignment and no frequency leak when the signal is played repeatedly. The accelerometer and gyroscope on each controller are sampled at 500Hz while the magnetic field sensor is sampled at 100Hz. The two controllers are connected to the HMD via Wi-Fi for transmitting recorded audio, orientations and distance estimates. The distance between the left controller and the HMD is computed on the left controller; similarly, the right controller computes another controller-to-HMD distance. The HMD computes the controller-wise distance and runs a particle filter to estimate the controller locations. Considering the length of the chirp signals, we set *VIU* to generate final controller locations at about 30 Hz.

We evaluate *VIU* with 5 volunteers, including 4 males and 1 female. Each volunteer was asked to wear a Google Daydream View headset with the S10 inserted as

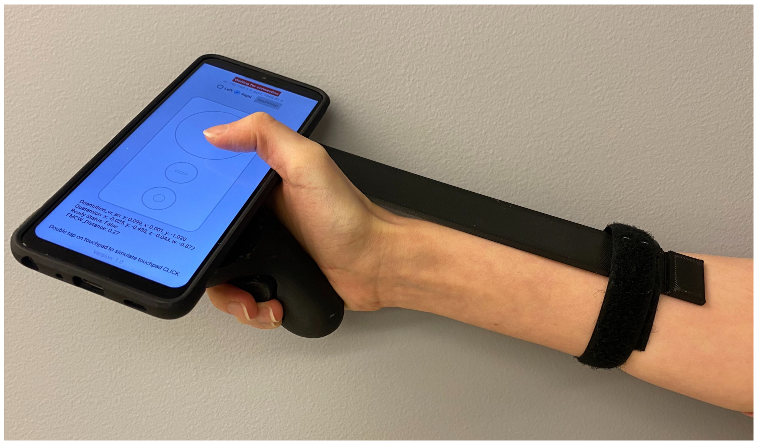


Fig. 4.10. 3D printed controller used in *VIU*.

the VR HMD, and hold two 3D printed controllers with a G7 attached to each one (shown in Figure 4.10). Each controller's long handle is fastened to the volunteer's forearm using nylon tapes to avoid wrist movement. We measure the upper arm and lower arm lengths, the shoulder width, and the HMD speaker position relative to the neck for the volunteers beforehand. Our experiments contain four parts:

1. Each volunteer participated in two 5-minute experiment sessions. They are asked to use our system in a normal indoor setting, with furniture and a noise level of 45 dB. At the beginning of each session, we calibrate our system by putting the HMD and the controllers to the volunteer's facing direction and record their orientations. The volunteer is then asked to keep her torso static while move her head and arms naturally at her will, i.e. along any trajectories and at any speed. Their movement is captured by a Kinect 2.0 in front of them for ground truth. In total, our collected trace is 50 minutes long. This is to evaluate the overall performance of *VIU* including accuracy, computation time, and power consumption.

2. We asked the volunteers to participate other two 5-minute sessions. The settings are the same as before, except that we played loud music at about 70 dB and had two people walk and talk in the room. This is to mimic a normal VR

gaming environment where the user is surrounded by some friends. In total, our collected trace is 50 minutes long. This is used to verify the robustness of *VIU*.

3. With the same setting as in the first experiment, we asked one person to use the system and naturally move the hands for 10 minutes. This is to evaluate the performance of the distance measuring module.

4. 4 people were asked to use two VR apps that we designed. One is to select 3D objects in the VR view, which demonstrates the basic functions of our system as an input device. The other one is a slingshot game, which shows how *VIU* performs in a scenario requiring two-hand collaboration. We will discuss more details in Section 4.5.7.

4.5.2 Overall tracking performance

We first show how *VIU* performs on controller tracking. Since Kinect can only track human body joints and hands are occluded by the controllers, we compare the

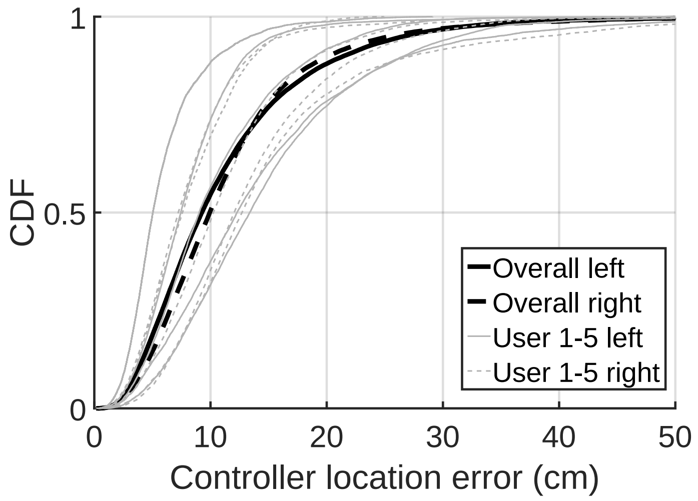


Fig. 4.11. Overall performance on controller tracking.

wrist locations from our system with those from Kinect. Figure 4.11 shows the CDF of the controller location errors. The median location error of the left and the right controller is 9.25 cm and 9.93 cm, respectively. There is no obvious difference between the tracking accuracy of the two controllers. In most cases, the tracking performance is consistent among all the volunteers with a 10 ± 5 cm range of median errors.

4.5.3 Computation time

Figure 4.12 shows the median of computation time for different processing stages in *VIU*. Each controller performs orientation estimation, whose computation time is very short and can be ignored. All the three devices conduct distance measuring which takes 2.0 ms for each calculation. On HMD, besides distance measuring, it also runs a particle filter to fuses intermediate results from all devices. Each stage, namely particle preprocessing, weight assignment, position estimation, particle re-sampling and state transition, takes 0.03 ms, 26 ms, 0.05 ms, 0.04 ms and 0.05 ms, respectively. The most computationally heavy device, i.e. the HMD, uses less than 29 ms to update a pair of controller locations. This ensures *VIU* to operate at a frame rate of 30 Hz.

Stage	**Time in milliseconds**
Orientation estimation	~0
Distance measuring	2.00
Particle preprocessing	0.03
Weight assignment	26.00
Position estimation	0.05
Particle resampling	0.04
State transition	0.05
Total	**28.17**

Fig. 4.12. Computation time on different stages.

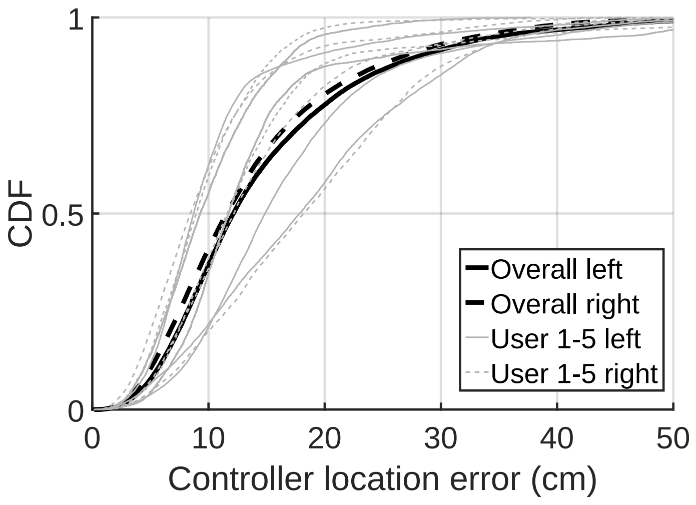


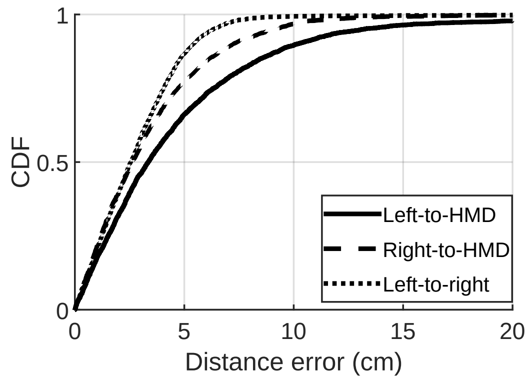
Fig. 4.13. Overall performance on controller tracking with multi-path effect and noises.

4.5.4 Effect of multi-path and noise

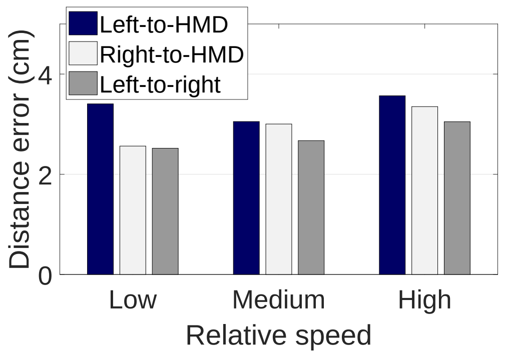
During the second part of our experiments, we demonstrate how *VIU* performs under the condition with multi-path and loud music. Figure 4.13 shows the CDF of the controller location errors with multi-path and noises. The median location error of the left and the right controller is 12.14 cm and 11.55 cm, respectively. The accuracy slightly degrades within a reasonable range.

4.5.5 Distance measuring accuracy

Figure 4.14 shows how the distance measuring module in *VIU* performs. Figure 4.14 (a) is the CDF of the distance errors between the left controller and the HMD, between the right controller and the HMD, and between the two controllers, respectively. The median errors are 3.35 cm, 2.66 cm and 2.55 cm. According to the relative speed between the two devices, each distance measurement fall into one of the categories — low speed (less than 0.5m/s), medium speed (between 0.5m/s and 1m/s),



(a)



(b)

Fig. 4.14. Performance of distance measuring. (a) Errors of pair-wise distances. (b) Errors with different relative moving speed.

and high speed (larger than 1m/s). As in Figure 4.14 (b), as the controllers move faster, the errors slightly increase. But the median errors are always below 4cm.

4.5.6 Comparison with ArmTrak

To provide a better sense of the performance of *VIU*, we save the orientation data from our first experiment session and run the online version of ArmTrak [16]. The comparison between ArmTrak and *VIU* is in Figure 4.15. Compared with ArmTrak's accuracy of 10.21 cm and 11.25 cm on each hand, *VIU* is about 1 cm more accurate than ArmTrak. Although the overall tracking errors are similar when conducting random hand movements, our system performs better in real VR apps which relies on the interaction of two hands. We will provide a more detailed comparison in specific scenarios in Section 4.5.7.

4.5.7 Applications

We also evaluate *VIU* via two VR applications.

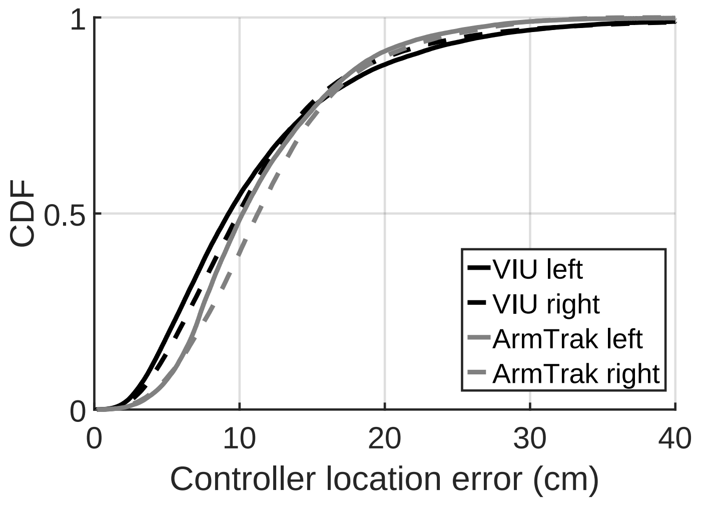


Fig. 4.15. Comparison between ArmTrak and *VIU* on controller tracking.

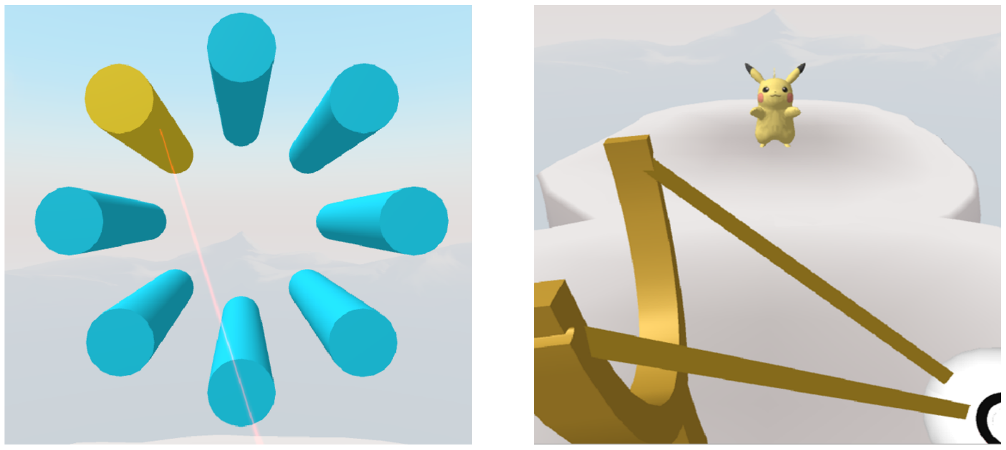


Fig. 4.16. *VIU* in VR applications. (a) Object selection, (b) slingshot.

Object selection. Object selection is a basic test for input devices [130]. We put 8 3D objects (as shown in Figure 4.16 (a)) in to the VR scene. One object is highlighted each time and the user is supposed to use the controller laser to point at the object and tap to confirm the selection. We asked each user to use this app for 2 min, and recorded the number of successful selections and the durations between each selection. The four users successfully selected 83, 92, 84, 60 objects using *VIU*, while they selected 66, 63, 70, 49 objects using ArmTrak. The average selection count and interval are 79.75 times and 1.505s among all users when using *VIU*, in contrast to 62 times and 1.935s when using ArmTrak. This demonstrates that *VIU* can enable basic operations like precise pointing and selection, meanwhile outperforms ArmTrak.

Slingshot. We also design a VR app to do a simple slingshot which requires the interaction of the two controllers. The user moves her two hands to control the position and the direction of the slingshot. She can increase the power used on the slingshot by move her hands further. The task is to use the Poke ball to catch Pikachu in the VR scene. For each Pikachu, the user can make five attempts. Pikachu will change its location if not caught in five attempts. There 6 Pikachus in total. We count the number of Pikachus that a user catches when she plays the game using *VIU* and ArmTrak. When playing this game using *VIU*, each user successfully caught 6, 4, 3, 6 Pikachus, respectively, with an average of 4.75. In contrast, all users failed to catch any Pikachu with ArmTrak. The main reason is that it is hard for ArmTrak to distinguish controller positions when it moves along users' facing direction.

4.6 Discussion

Acoustic ranging limitation. Our system falls short in the following aspects. The smartphone built-in speakers and microphones are not tailored for transceiving ultrasonic waves. For instance, their frequency response is non-linear [103] and the peripherals are not omni-directional, which harms the acoustic ranging performance. In the case when two speakers turn back to each other, the accuracy significantly drops. Another limitation of acoustic-based ranging is the interfere from multi-path reflection, which leads to a limited performance in a small room. One possible solution to this issue is MUSIC [131].

Sensor limitation. The prototype is implemented on commercial smartphones which have no dedicated sensors for being adopted as VR controllers. This may cause inaccuracy in orientation estimations. Moreover, relying on magnetometers prevents the usage of our system in certain environments with magnetic interference, for example, near an elevator. We also explored the possibility of utilizing acceleration as another factor in the particle filter. Although it shows a good trend in suggesting motion directions, the wide fluctuation makes its reading at each timestamp to be not precise.

Movement limitation. The current system only works under the assumption that the transformation between the hand and the lower limb is rigid, which means users cannot rotate their wrists freely. Relaxing the assumption will increase the arm model in our current system from 5-DoF to 7-DoF, consequently ending up with a higher dimension of the search space. With only one device attached to each arm, this would introduce more uncertainty to location estimates, degrading the system's performance. Another constraint is that we require the user to keep facing the same direction. In order to translate the hands' orientation into the torso coordinates system, the user's facing direction is required.

4.7 Conclusion

We propose *VIU*, a self-contained, inside-out controller tracking system for mobile VR. Our system is built on off-the-shelf commercial phones without the need for extra hardware. It composes of inertial sensing and ultrasonic ranging to precisely locate the two motion controllers in real time, with a frame rate of 30 Hz. We design a novel two-way FMCW ultrasonic acoustic ranging algorithm and a particle filter to fuse multiple measurements including orientations and distances. It achieves a tracking accuracy of less than 10 cm, which enables several VR applications, such as object selection and slingshot.

5. CONCLUSION AND FUTURE WORK

In this dissertation, our works have focused on applying multimodal sensing to human motion tracking in mobile systems. Classifying the tracking systems into two categories, we have explored the main challenging problem in each category and presented our works that address these problems separately.

For the tracking systems based on external stationary sensors, we point out the ID association issue when the external tracking system wants to provide customized service for each specific user being tracked. Although the "outside" system can tracks multiple people simultaneously, it cannot know each user's phone address from its perspective. Driven by this, our first work enables public surveillance cameras to send targeted messages to people's smartphones. To solve the ID association issue, the system extracts motion features from videos and uses them to compare with the sensor data collected by the users' smartphones. Based on the consistency between the two sides, walking patterns are used as a temporary identifier for a person. To prevent motion leaks, the system requires no sensor data from the users' phones and further protects user privacy by transforming the motion features in communication addresses into low-dimensional codes. We have demonstrated the performance of this system with both experiments (87% matching accuracy and 3-second delay) and three real-world applications (i.e. indoor localization, automatic audio guide, and gesture-based messaging). Then to further improve the distinguishability of this human identifier, our second work enhances the previous system by introducing context features into the addresses. The new system presents a context selection algorithm to dynamically select lightweight yet effective features to reduce packet overheads, as well as an effortless way to generate ambient sensing maps. It not only shows better scalability in a real retailer scenario with dense people but also fits the communication address into a fix-length header (i.e. 40 bytes) regardless of the number of users that are supported simultaneously. Designing it as a framework, we believe this system can become a generic underlay to various practical applications.

The motion tracking systems without external sensors suffer from inaccuracy caused by limited sensing information. In our third work, we focus on designing a controller tracking scheme for mobile VR systems. For easy setup and high portability, mobile VR systems do not rely on external beacons as reference points or embedded cameras to provide rich environmental information. Our key idea is utilizing multiple types of light-weight embedded sensors to provide information from various aspects, e.g. controller orientations obtained from inertial sensing data, pairwise distances among the headset and the two controllers based on ultrasonic ranging, and a kinematic arm model. Relying on a well-designed statistical model, we are able to have tolerance for inevitable sensor noises and fuse this information into accurate controller positions in real time. Our system opens up new types of interactive gaming on mobile VR systems, such as slingshot.

In a word, our works take a step towards realizing the potential of multimodal sensing in human motion tracking. In the long term, we see closer interactions between human and smart mobile devices as a future trend. Motion tracking using multimodal sensing will help provide more detailed information about human behaviors and forge new possibilities. In the future, we plan to continue our work in the following directions.

Dynamically switching among multiple solutions. Since there have been many tracking schemes based on various hardware that are designed for different application scenarios, we plan to explore a high-level framework which contains these existing schemes as functional blocks and dynamically calls one or several of the blocks according to current sensor data. For example, when vision-based tracking is suffering from occlusion, motion blur, or dim light, IMU-based schemes can take over the tracking tasks; in the outdoor environment, the framework can put more weight onto magnetometer readings since the magnetic field tends to be stable. Such a solution requires fast assessment on sensor data quality and a careful tradeoff between switching overhead and tracking performance.

Using vision to get keyframes. Cameras can provide tracking results that are more accurate, but demand high power consumption. To save power, we propose a solution that relies on vision-based tracking to provide accurate location information used as keyframes. As inertial sensors are quite accurate in the short term with small integration drifts, these lightweight sensors can be used to interpolate the location estimates between the keyframes.

In the future, we will continue exploring unsolved problems and new opportunities in mobile systems, especially in the VR area, and propose practical solutions by fusing data from multiple types of sensors. We believe sensor-based solutions will perfectly fit into many use cases due to its lightweight, low cost, and growing popularity.

REFERENCES

[1] Internet of things — sbir.gov. [Online]. Available: https://www.sbir.gov/sbirsearch/detail/1483095

[2] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: Unsupervised indoor localization," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*, 2012, pp. 197–210.

[3] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: Zero-effort crowdsourcing for indoor localization," in *Proceedings of the 18th annual international conference on Mobile computing and networking*, 2012, pp. 293–304.

[4] J. Sung, C. Ponce, B. Selman, and A. Saxena, "Unstructured human activity detection from rgbd images," in *2012 IEEE international conference on robotics and automation*. IEEE, 2012, pp. 842–849.

[5] H. Kwon, G. D. Abowd, and T. Plötz, "Adding structural characteristics to distribution-based accelerometer representations for activity recognition using wearables," in *Proceedings of the 2018 ACM international symposium on wearable computers*, 2018, pp. 72–75.

[6] W. Wang, A. X. Liu, M. Shahzad, K. Ling, and S. Lu, "Device-free human activity recognition using commercial wifi devices," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 5, pp. 1118–1131, 2017.

[7] Q. Pu, S. Gupta, S. Gollakota, and S. Patel, "Whole-home gesture recognition using wireless signals," in *Proceedings of the 19th annual international conference on Mobile computing & networking*, 2013, pp. 27–38.

[8] W. Ruan, Q. Z. Sheng, L. Yang, T. Gu, P. Xu, and L. Shangguan, "Audiogest: enabling fine-grained hand gesture detection by decoding echo signal," in *Proceedings of the 2016 ACM international joint conference on pervasive and ubiquitous computing*, 2016, pp. 474–485.

[9] T. Babic, H. Reiterer, and M. Haller, "Pocket6: A 6dof controller based on a simple smartphone application," in *Proceedings of the Symposium on Spatial User Interaction*, 2018, pp. 2–10.

[10] E. Whitmire, F. Salemi Parizi, and S. Patel, "Aura: Inside-out electromagnetic controller tracking," in *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, 2019, pp. 300–312.

[11] R. Nandakumar, V. Iyer, D. Tan, and S. Gollakota, "Fingerio: Using active sonar for fine-grained finger tracking," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 2016, pp. 1515–1525.

[12] L.-Y. Chi, L. R. P. Gomez, R. A. Ryskamp, and S. G. Mavinkurve, "Wearable heads-up display with integrated finger-tracking input sensor," Jun. 19 2012, uS Patent 8,203,502.

[13] M. Zhao, F. Adib, and D. Katabi, "Emotion recognition using wireless signals," in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, 2016, pp. 95–108.

[14] T. Wang, D. Zhang, Y. Zheng, T. Gu, X. Zhou, and B. Dorizzi, "C-fmcw based contactless respiration detection using acoustic signal," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 4, p. 170, 2018.

[15] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality.* IEEE Computer Society, 2007, pp. 1–10.

[16] S. Shen, H. Wang, and R. Roy Choudhury, "I am a smartwatch and i can track my user's arm," in *Proceedings of the 14th annual international conference on Mobile systems, applications, and services.* ACM, 2016, pp. 85–96.

[17] H. Wang, T. T.-T. Lai, and R. Roy Choudhury, "Mole: Motion leaks through smartwatch sensors," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking.* ACM, 2015, pp. 155–166.

[18] L. Li, P. Hu, C. Peng, G. Shen, and F. Zhao, "Epsilon: A visible light based positioning system," in *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*, 2014, pp. 331–343.

[19] T. Li, Q. Liu, and X. Zhou, "Practical human sensing in the light," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, 2016, pp. 71–84.

[20] J. Wang, K. Zhao, X. Zhang, and C. Peng, "Ubiquitous keyboard for small mobile devices: harnessing multipath fading for fine-grained keystroke localization," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, 2014, pp. 14–27.

[21] J. Liu, Y. Wang, G. Kar, Y. Chen, J. Yang, and M. Gruteser, "Snooping keystrokes with mm-level audio ranging on a single phone," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, 2015, pp. 142–154.

[22] V. Otsason, A. Varshavsky, A. LaMarca, and E. De Lara, "Accurate gsm indoor localization," in *International conference on ubiquitous computing.* Springer, 2005, pp. 141–158.

[23] S. H. Yoon, K. Huo, V. P. Nguyen, and K. Ramani, "Timmi: Finger-worn textile input device with multimodal sensing in mobile interaction," in *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction*, 2015, pp. 269–272.

[24] 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*, 2014, pp. 605–616.

[25] N. Jenkins, “245 million video surveillance cameras installed globally in 2014,” *IHS Technology.*

[26] H. Li, P. Zhang, S. Al Moubayed, S. N. Patel, and A. P. Sample, “Id-match: a hybrid computer vision and rfid system for recognizing individuals in groups,” in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems.* ACM, 2016, pp. 4933-4944.

[27] H. Wang, X. Bao, R. R. Choudhury, and S. Nelakuditi, “Insight: recognizing humans without face recognition,” in *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications.* ACM, 2013, p. 7.

[28] H. Wang, X. Bao, R. Roy Choudhury, and S. Nelakuditi, “Visually fingerprinting humans without face recognition,” in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services.* ACM, 2015, pp. 345–358.

[29] D. Jung, T. Teixeira, and A. Savvides, “Towards cooperative localization of wearable sensors using accelerometers and cameras,” in *INFOCOM, 2010 Proceedings IEEE.* IEEE, 2010, pp. 1–9.

[30] H. Bojinov, Y. Michalevsky, G. Nakibly, and D. Boneh, “Mobile device identification via sensor fingerprinting,” *arXiv preprint arXiv:1408.1416*, 2014.

[31] S. Dey, N. Roy, W. Xu, R. R. Choudhury, and S. Nelakuditi, “Accelprint: Imperfections of accelerometers make smartphones trackable.” in *NDSS*, 2014.

[32] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning.* Springer series in statistics Springer, Berlin, 2001, vol. 1.

[33] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, “Realtime multi-person 2d pose estimation using part affinity fields,” in *CVPR*, 2017.

[34] P. Dollár, R. Appel, and W. Kienzle, “Crosstalk cascades for frame-rate pedestrian detection,” in *ECCV*, 2012, pp. 645–659.

[35] B. Yang and R. Nevatia, “Multi-target tracking by online learning a crf model of appearance and motion patterns,” *International Journal of Computer Vision*, vol. 107, no. 2, pp. 203–217, 2014.

[36] ——, “Online learned discriminative part-based appearance models for multi-human tracking,” in *European Conference on Computer Vision.* Springer, 2012, pp. 484–498.

[37] A. A. Perera, C. Srinivas, A. Hoogs, G. Brooksby, and W. Hu, “Multi-object tracking through simultaneous long occlusions and split-merge conditions,” in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1. IEEE, 2006, pp. 666–673.

[38] J. Xing, H. Ai, and S. Lao, "Multi-object tracking through occlusions by local tracklets filtering and global tracklets association with detection responses," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on.* IEEE, 2009, pp. 1200–1207.

[39] A. Andriyenko and K. Schindler, "Multi-target tracking by continuous energy minimization," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on.* IEEE, 2011, pp. 1265–1272.

[40] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes, "Globally-optimal greedy algorithms for tracking a variable number of objects," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on.* IEEE, 2011, pp. 1201–1208.

[41] S. Wang, H. Lu, F. Yang, and M.-H. Yang, "Superpixel tracking," in *Computer Vision (ICCV), 2011 IEEE International Conference on.* IEEE, 2011, pp. 1323–1330.

[42] B. Liu, J. Huang, L. Yang, and C. Kulikowsk, "Robust tracking using local sparse appearance model and k-selection," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on.* IEEE, 2011, pp. 1313–1320.

[43] H. Grabner, J. Matas, L. Van Gool, and P. Cattin, "Tracking the invisible: Learning where the object might be," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on.* IEEE, 2010, pp. 1285–1292.

[44] P. Dollár, R. Appel, and W. Kienzle, "Crosstalk cascades for frame-rate pedestrian detection," in *Computer Vision–ECCV 2012.* Springer, 2012, pp. 645–659.

[45] P. Dollár, "Piotr's Computer Vision Matlab Toolbox (PMT)," https://github.com/pdollar/toolbox.

[46] C.-H. Kuo and R. Nevatia, "How does person identity recognition help multi-person tracking?" in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on.* IEEE, 2011, pp. 1217–1224.

[47] R. E. Kalman *et al.*, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[48] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping," in *CVPR*, 2017.

[49] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *CVPR*, 2016.

[50] O. Tuzel, F. Porikli, and P. Meer, "Region covariance: A fast descriptor for detection and classification," in *European conference on computer vision.* Springer, 2006, pp. 589–600.

[51] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[52] O. Faugeras, *Three-dimensional computer vision: a geometric viewpoint.* MIT press, 1993.

[53] H. W. Eves, *A survey of geometry.* Allyn and Bacon, 1972, vol. 1.

[54] Z. Zhang, D. Chu, X. Chen, and T. Moscibroda, "Swordfight: Enabling a new class of phone-to-phone action games on commodity phones," in *Proceedings of the 10th international conference on Mobile systems, applications, and services.* ACM, 2012, pp. 1–14.

[55] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on.* IEEE, 2005, pp. 65–72.

[56] 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.* ACM, 2014, pp. 329–342.

[57] D. Kelly, S. Donnelly, and B. Caulfield, "Smartphone derived movement profiles to detect changes in health status in copd patients-a preliminary investigation," in *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE.* IEEE, 2015, pp. 462–465.

[58] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[59] E. Amaldi and V. Kann, "On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems," *Theoretical Computer Science*, vol. 209, no. 1-2, pp. 237–260, 1998.

[60] M. Scholz, F. Kaplan, C. L. Guy, J. Kopka, and J. Selbig, "Non-linear pca: a missing data approach," *Bioinformatics*, vol. 21, no. 20, pp. 3887–3895, 2005.

[61] R. C. Browning, E. A. Baker, J. A. Herron, and R. Kram, "Effects of obesity and sex on the energetic cost and preferred speed of walking," *Journal of Applied Physiology*, vol. 100, no. 2, pp. 390–398, 2006.

[62] A. Goshtasby, "Piecewise linear mapping functions for image registration," *Pattern Recognition*, vol. 19, no. 6, pp. 459–466, 1986.

[63] ——, "Image registration by local approximation methods," *Image and Vision Computing*, vol. 6, no. 4, pp. 255–261, 1988.

[64] J. Schiller and A. Voisard, *Location-based services.* Elsevier, 2004.

[65] I. A. Junglas and R. T. Watson, "Location-based services," *Communications of the ACM*, vol. 51, no. 3, pp. 65–69, 2008.

[66] T. Li, C. An, X. Xiao, A. T. Campbell, and X. Zhou, "Real-time screen-camera communication behind any scene," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services.* ACM, 2015, pp. 197–211.

[67] A. Wang, Z. Li, C. Peng, G. Shen, G. Fang, and B. Zeng, "Inframe++: Achieve simultaneous screen-human viewing and hidden screen-camera communication," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2015, pp. 181–195.

[68] Z. Yang, Y. Bao, C. Luo, X. Zhao, S. Zhu, C. Peng, Y. Liu, and X. Wang, "Artcode: preserve art and code in any image," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2016, pp. 904–915.

[69] A. Ashok, S. Jain, M. Gruteser, N. Mandayam, W. Yuan, and K. Dana, "Capacity of pervasive camera based communication under perspective distortions," in *Pervasive Computing and Communications (PerCom), 2014 IEEE International Conference on*. IEEE, 2014, pp. 112–120.

[70] Y. Michalevsky, A. Schulman, G. A. Veerapandian, D. Boneh, and G. Nakibly, "Powerspy: Location tracking using mobile device power analysis."

[71] C. Wang, X. Guo, Y. Wang, Y. Chen, and B. Liu, "Friend or foe?: Your wearable devices reveal your personal pin," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*. ACM, 2016, pp. 189–200.

[72] Y.-A. De Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel, "Unique in the crowd: The privacy bounds of human mobility," *Scientific reports*, vol. 3, p. 1376, 2013.

[73] P. Jain, J. Manweiler, and R. Roy Choudhury, "Overlay: Practical mobile augmented reality," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2015, pp. 331-344.

[74] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*. ACM, 2014, pp. 68–81.

[75] S. Chen, A. Pande, and P. Mohapatra, "Sensor-assisted facial recognition: An enhanced biometric authentication system for smartphones," in *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '14. New York, NY, USA: ACM, 2014, pp. 109–122. [Online]. Available: http://doi.acm.org/10.1145/2594368.2594373

[76] H. Jin, C. Holz, and K. Hornbæk, "Tracko: Ad-hoc mobile 3d tracking using bluetooth low energy and inaudible signals for cross-device interaction," in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, 2015, pp. 147–156.

[77] D. F. Llorca, R. Quintero, I. Parra, and M. Sotelo, "Recognizing individuals in groups in outdoor environments combining stereo vision, rfid and ble," *Cluster Computing*, vol. 20, no. 1, pp. 769-779, 2017.

[78] K. W. Bowyer, "Face recognition technology: security versus privacy," *IEEE Technology and society magazine*, vol. 23, no. 1, pp. 9–19, 2004.

[79] "San Francisco Banned Facial Recognition. Will California Follow?" https://www.nytimes.com/2019/07/01/us/facial-recognition-san-francisco.html.

[80] H. Li, P. Zhang, S. Al Moubayed, S. N. Patel, and A. P. Sample, "Id-match: A hybrid computer vision and rfid system for recognizing individuals in groups," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI '16. New York, NY, USA: ACM, 2016, pp. 4933–4944. [Online]. Available: http://doi.acm.org/10.1145/2858036.2858209

[81] S. Cao and H. Wang, "Enabling public cameras to talk to the public," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 2, no. 2, pp. 63:1–63:20, Jul. 2018. [Online]. Available: http://doi.acm.org/10.1145/3214266

[82] X. Liu, Y. Jiang, P. Jain, and K.-H. Kim, "Tar: Enabling fine-grained targeted advertising in retail stores," in *MobiSys*, 2018.

[83] N. Peterfreund, "Robust tracking of position and velocity with kalman snakes," *IEEE transactions on pattern analysis and machine intelligence*, vol. 21, no. 6, pp. 564–569, 1999.

[84] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[85] "Time synchronization in a local network," http://clocksynchro.com/.

[86] D. F. Llorca, R. Quintero, I. Parra, and M. A. Sotelo, "Recognizing individuals in groups in outdoor environments combining stereo vision, rfid and ble," *Cluster Computing*, vol. 20, no. 1, pp. 769–779, Mar. 2017. [Online]. Available: https://doi.org/10.1007/s10586-017-0764-0

[87] G. Takacs, V. Chandrasekhar, and Others, "Outdoors augmented reality on mobile phone using loxel-based visual feature organization," in *ACM ICMR*, 2008.

[88] Google cardboard. [Online]. Available: https://en.wikipedia.org/wiki/Google_Cardboard

[89] A. Robertson. (2017) The gear vr's new controller makes it more expensive, but a lot more useful. [Online]. Available: https://www.theverge.com/2017/3/29/15076978/samsung-gear-vr-motion-controller-announced-vs-oculus-touch

[90] D. C. Niehorster, L. Li, and M. Lappe, "The accuracy and precision of position and orientation tracking in the htc vive virtual reality system for scientific research," *i-Perception*, vol. 8, no. 3, p. 2041669517708205, 2017.

[91] B. Lang. (2019) Quest and rift s update brings controller tracking improvements. [Online]. Available: https://www.roadtovr.com/update-brings-controller-tracking-imr/

[92] D. Heaney. (2019) How vr positional tracking systems work. [Online]. Available: https://uploadvr.com/how-vr-tracking-works/

[93] B. Lang. (2019) Here's what facebook says about camera privacy on quest and rift s. [Online]. Available: https://www.roadtovr.com/oculus-quest-camera-privacy-rift-s-facebook/

[94] R. Pandey, P. Pidlypenskyi, S. Yang, and C. Kaeser-Chen, "Egocentric 6-dof tracking of small handheld objects," *arXiv preprint arXiv:1804.05870*, 2018.

[95] H. Zhou and H. Hu, "Upper limb motion estimation from inertial measurements," *International Journal of Information Technology*, vol. 13, no. 1, pp. 1–14, 2007.

[96] A. G. Cutti, A. Giovanardi, L. Rocchi, A. Davalli, and R. Sacchetti, "Ambulatory measurement of shoulder and elbow kinematics through inertial and magnetic sensors," *Medical & biological engineering & computing*, vol. 46, no. 2, pp. 169–178, 2008.

[97] 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*, 2018, pp. 429–444.

[98] F. Adib, C.-Y. Hsu, H. Mao, D. Katabi, and F. Durand, "Capturing the human figure through a wall," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, pp. 1–13, 2015.

[99] W. Mao, J. He, and L. Qiu, "Cat: high-precision acoustic motion tracking," in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking.* ACM, 2016, pp. 69–81.

[100] S. Yun, Y.-C. Chen, and L. Qiu, "Turning a mobile device into a mouse in the air," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services.* ACM, 2015, pp. 15–29.

[101] W. Mao, M. Wang, W. Sun, L. Qiu, S. Pradhan, and Y.-C. Chen, "Rnn-based room scale hand motion tracking," in *The 25th Annual International Conference on Mobile Computing and Networking*, 2019, pp. 1–16.

[102] H. Zhou, Y. Gao, X. Song, W. Liu, and W. Dong, "Limbmotion: Decimeter-level limb tracking for wearable-based human-computer interaction," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 3, no. 4, pp. 1–24, 2019.

[103] C. Peng, G. Shen, Y. Zhang, Y. Li, and K. Tan, "Beepbeep: a high accuracy acoustic ranging system using cots mobile devices," in *Proceedings of the 5th international conference on Embedded networked sensor systems.* ACM, 2007, pp. 1–14.

[104] W. Mao, Z. Zhang, L. Qiu, J. He, Y. Cui, and S. Yun, "Indoor follow me drone," in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services.* ACM, 2017, pp. 345–358.

[105] D. K. Michael Abrash, "Why virtual reality isn't (just) the next big platform : Michael abrash and dov katz of oculus vr," Video, 2014. [Online]. Available: https://www.youtube.com/watch?v=dxbh-TM5yNc

[106] A. Medien, "Implementation of a low cost marker based infrared optical tracking system," 2006.

[107] M. Ribo, A. Pinz, and A. L. Fuhrmann, "A new optical tracking system for virtual and augmented reality applications," in *IMTC 2001. Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference. Rediscovering Measurement in the Age of Informatics (Cat. No. 01CH 37188)*, vol. 3. IEEE, 2001, pp. 1932–1936.

[108] E. Foxlin, M. Harrington, and G. Pfeifer, "Constellation: A wide-range wireless motion-tracking system for augmented reality and virtual set applications," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques.* Citeseer, 1998, pp. 371–378.

[109] S. Islam, B. Ionescu, C. Gadea, and D. Ionescu, "Full-body tracking using a sensor array system and laser-based sweeps," in *2016 IEEE Symposium on 3D User Interfaces (3DUI).* IEEE, 2016, pp. 71–80.

[110] P. Dempsey, "The teardown: Htc vive vr headset," *Engineering & Technology*, vol. 11, no. 7-8, pp. 80–81, 2016.

[111] A. Borrego, J. Latorre, M. Alcañiz, and R. Llorens, "Comparison of oculus rift and htc vive: feasibility for virtual reality-based exploration, navigation, exergaming, and rehabilitation," *Games for health journal*, vol. 7, no. 3, pp. 151–156, 2018.

[112] P. Lang, A. Kusej, A. Pinz, and G. Brasseur, "Inertial tracking for mobile augmented reality," in *IMTC/2002. Proceedings of the 19th IEEE Instrumentation and Measurement Technology Conference (IEEE Cat. No. 00CH37276)*, vol. 2. IEEE, 2002, pp. 1583–1587.

[113] I. E. Sutherland, "A head-mounted three dimensional display," in *Proceedings of the December 9-11, 1968, fall joint computer conference, part I.* ACM, 1968, pp. 757–764.

[114] Tdk announces new chirp sonictrack inside-out 6-dof ultrasonic controller tracking solution for all-in-one vr. [Online]. Available: https://www.invensense.com/news-media/TDK-announces-new-Chirp-SonicTrack-inside-out-6-DoF-ultrasonic-controller-tracking-solution-for-all-in-one-VR/

[115] Tdk chirp sonictrack power 6-dof in the best vr systems. [Online]. Available: https://www.ubergizmo.com/2019/03/tdk-chirp-sonictrack-6-dof-vr/

[116] P. G. Kannan, S. P. Venkatagiri, M. C. Chan, A. L. Ananda, and L.-S. Peh, "Low cost crowd counting using audio tones," in *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems.* ACM, 2012, pp. 155–168.

[117] L. A. Jones and S. J. Lederman, *Human hand function.* Oxford University Press, 2006.

[118] A. G. Stove, "Linear fmcw radar techniques," in *IEE Proceedings F (Radar and Signal Processing)*, vol. 139, no. 5. IET, 1992, pp. 343–350.

[119] D. C. Boone and S. P. Azen, "Normal range of motion of joints in male subjects." *JBJS*, vol. 61, no. 5, pp. 756–759, 1979.

[120] R. L. Gajdosik and R. W. Bohannon, "Clinical measurement of range of motion: review of goniometry emphasizing reliability and validity," *Physical therapy*, vol. 67, no. 12, pp. 1867–1872, 1987.

[121] N. B. Reese and W. D. Bandy, *Joint range of motion and muscle length testing-E-book*. Elsevier Health Sciences, 2016.

[122] R. S. Hartenberg and J. Denavit, "A kinematic notation for lower pair mechanisms based on matrices," *Journal of applied mechanics*, vol. 77, no. 2, pp. 215–221, 1955.

[123] J. C. Mitchell, "Discrete uniform sampling of rotation groups using orthogonal images," 2007.

[124] A. Yershova, S. Jain, S. M. Lavalle, and J. C. Mitchell, "Generating uniform incremental grids on so (3) using the hopf fibration," *The International journal of robotics research*, vol. 29, no. 7, pp. 801–812, 2010.

[125] Android sensor fusion tutorial. [Online]. Available: http://plaw.info/articles/sensorfusion/

[126] S. Colton and F. Mentor, "The balance filter," *Presentation, Massachusetts Institute of Technology*, 2007.

[127] Nist/sematech e-handbook of statistical methods. [Online]. Available: https://www.itl.nist.gov/div898/handbook/pmd/section2/pmd214.htm

[128] C.-l. Shen, *Fundamentals of the theory of inverse sampling.* University of Michigan, 1936.

[129] "Google vr services emulator for non android n to get working daydream development kit hardware," 2016. [Online]. Available: https://github.com/domination/gvr-services-emulator

[130] D. Natapov, S. J. Castellucci, and I. S. MacKenzie, "Iso 9241-9 evaluation of video game controllers," in *Proceedings of Graphics Interface 2009.* Canadian Information Processing Society, 2009, pp. 223–230.

[131] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE transactions on antennas and propagation*, vol. 34, no. 3, pp. 276–280, 1986.

VITA

Siyuan Cao is a Ph.D. student at Purdue Univerisity. She received her B.S. degree from Shanghai Jiao Tong University in 2015. She worked as an intern at Microsoft Research — Asia (2016), Microsoft Research AI (2017), and Facebook (2019).

Her research focuses on techniques and applications that use sensor fusion to support real-world mobile systems.