Enabling Public Cameras to Talk to the Public

SIYUAN CAO, Purdue University, USA
HE WANG, Purdue University, USA

This paper asks: 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 cameras rely on viewing user's motion patterns, and use the uniqueness of these patterns as the address for communication. Once receiving the wireless broadcast from the cameras, the 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.

In addition to requiring no data from users, our system transforms the motion patterns into low-dimensional codes 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. *PHADE* can also deliver messages based on human gestures. There is no need to deploy any extra infrastructures or to require users to rent any dedicated device.

CCS Concepts: • **Security and privacy → Privacy protections**; • **Networks → Naming and addressing**; *Mobile networks*; • **Software and its engineering** → Real-time systems software;

Additional Key Words and Phrases: Human addressing, motion features, camera, communication, principal component analysis

ACM Reference Format:
Siyuan Cao and He Wang. 2018. Enabling Public Cameras to Talk to the Public. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 2, Article 63 (June 2018), 20 pages. https://doi.org/10.1145/3214266

1 INTRODUCTION

Surveillance cameras are pervasively deployed in public areas, such as shopping mall, museum, galleries and so on [24]. 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 paper, we define this problem as *Private Human Addressing* and aim to design a practical system that realizes it.

Authors' addresses: Siyuan Cao, cao208@purdue.edu, Purdue University, 305 N University St, West Lafayette, IN, 47907, USA; He Wang, hw@purdue.edu, Purdue University, 305 N University St, West Lafayette, IN, 47907, USA.

2474-9567/2018/6-ART63 $15.00
https://doi.org/10.1145/3214266

Fig. 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).

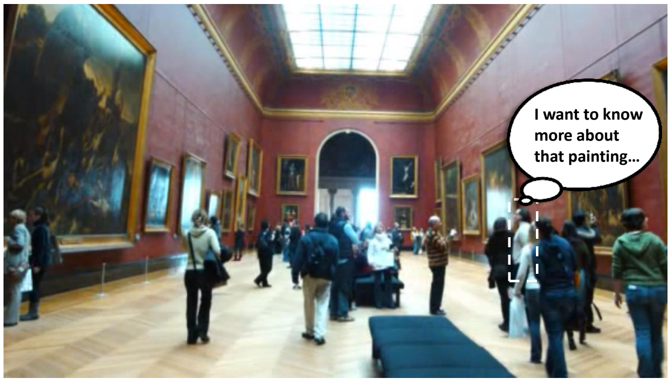


Fig. 1. A camera is able to send context-aware messages to a targeted user, without knowing any addresses of their phones.

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 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 [31] uses RFID tags worn by people to assign a unique ID to each person in the camera view. Use of such tags requires user registration and is not suitable for public areas where not everyone might wear these tags. Insight [45, 46] demonstrates that motion patterns and clothing colors can be used as a temporary fingerprint for recognizing a person. [26] 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 [4, 10]. 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) [17], 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[6], 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.

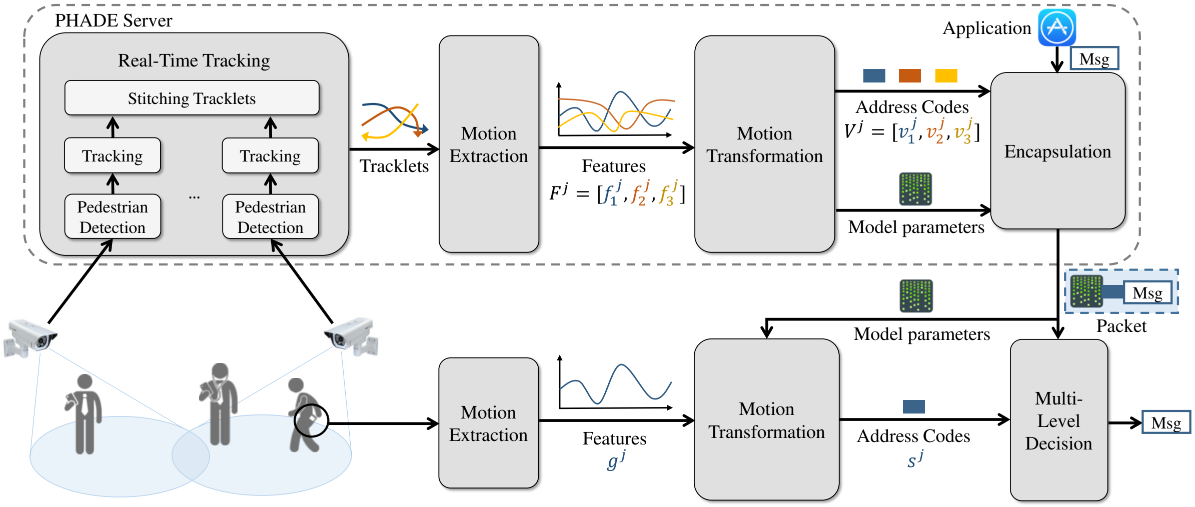


Fig. 2. System overview. Videos are streamed to a server to track people in each camera views. Local tracklets from various cameras are then stitched into global tracklets. Motion features for each tracklet are extracted and transformed into address codes to protect user behavior details. The server 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.

2 SYSTEM OVERVIEW

Fig. 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 [12] 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 feature vectors, $F_i = [f_i^1, f_i^2, \ldots]$, 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 $F^j = [f_1^j, f_2^j, \ldots]$, which will be transformed into address codes $V^j = [v_1^j, v_2^j, \ldots]$ using Principal Component Analysis (PCA) to diminish walking details and meanwhile preserve the distinguishable characteristics. The address code tuple $< v_i^1, v_i^2, \ldots >$ 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 $G = [g^1, g^2, \ldots]$ 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 $S = [s^1, s^2, \ldots]$. During the hierarchical comparison between tuples $< s^1, s^2, \ldots >$ and $< v_i^1, v_i^2, \ldots >$ for $i = 1, 2, \ldots$, 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.

3 DESIGN DETAILS

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.

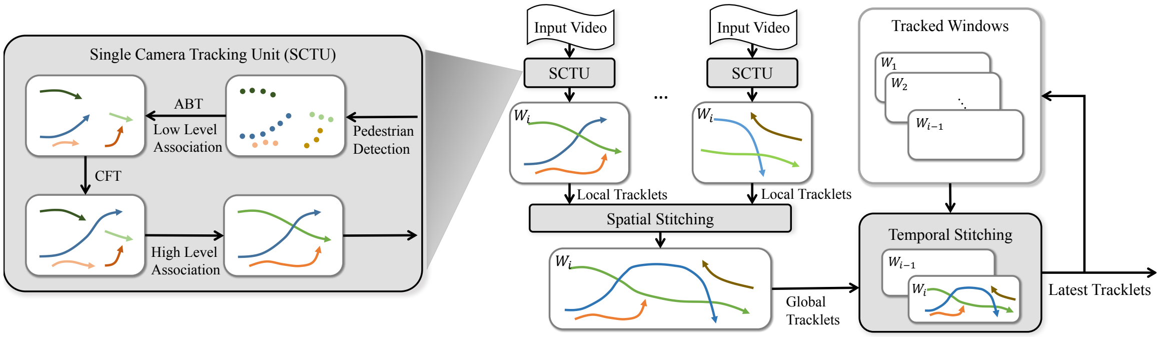


Fig. 3. Tracking scheme. Single Camera Tracking Unit is applied to each video stream to obtain local tracklets in the current sliding window. After going through four steps of pedestrian detection, low-level association, Category Free Tracking and high-level association, the generated local tracklets 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 current window are stored for further feature extraction.

3.1.1 *Tracking Scheme.*

As shown in Fig. 3, the tracking scheme is composed of three functional modules. *Single Camera Tracking Unit* (SCTU) is applied to each sliding window [52] (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 individual camera view. It adopts a unified framework in [51], combining two mainstream tracking approaches, i.e. Association Based Tracking (ABT) [2, 36, 37, 50, 51] and Category Free Tracking (CFT) [20, 33, 48]. Based on a pre-trained detector [11, 13], 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 [30, 50] and smoothed by Kalman filter [28] to obtain high-level local tracklets. Finally, the local tracklets are *spatially stitched* between cameras and *temporally stitched*

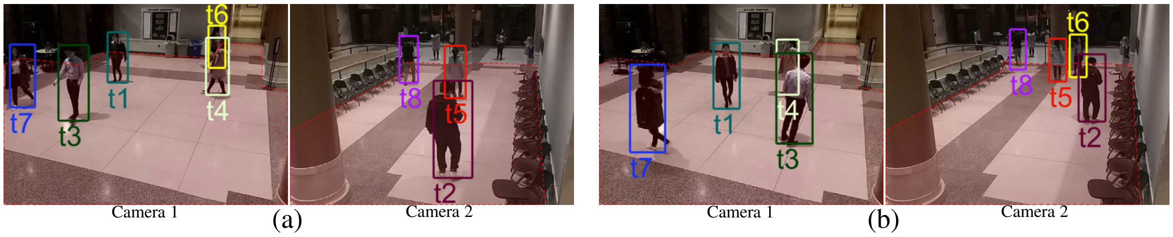


Fig. 4. Tracking results in a real scenario. With two cameras cooperatively monitoring a lobby, people in the camera views are labeled with their tracklet IDs. t_6 is correctly and stably tracked even if it is occluded or it crosses various camera views and sliding windows.

with tracklets in W_{i-1}, to generate latest global tracklets. Here we also add a latest human pose detection scheme called OpenPose [6, 41, 49], as a complement to the existing pre-trained detector. OpenPose not only improves the detecting and tracking accuracy 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. 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. Even if people are occluded or cross various camera views and sliding windows (e.g. t_6 in Fig. 4), the tracking results are still correct and stable.

3.1.2 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. 5(a)-(c) show consecutive detection responses which are able to form a low-level tracklet, while Fig. 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 [51]. Then a potential extension is chosen from the samples by extracting appearance features (e.g. color histogram, texture [42], and Histogram of Oriented Gradient descriptors [8]) from each sample and comparing the similarity one by one with the existing tracklet.

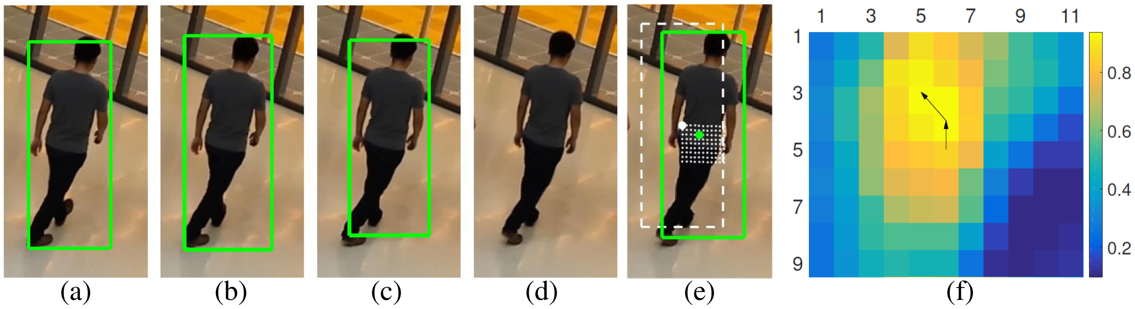


Fig. 5. Pruning CFT sample space. (a)-(c) are detection responses on consecutive frames. (d) shows a missing detection on the next frame. (e) shows a height-aware sample space for the frame in (d). (f) shows the similarity heat map for the sample space in (e) and a gradient search path for the local optimal sample.

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[15, 16], 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. 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. 5(f) shows a heat map of similarity between the 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). Therefore, we devise a pipelined and parallel scheme (shown in Fig. 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. 3.1.1), 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.

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 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 above requirements, some intuitive options are considered as 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 [14]. 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 phone. Also, walking direction attained from the compass is not suitable for many indoor scenarios because the compass itself is susceptible to ferromagnetic interference [38].

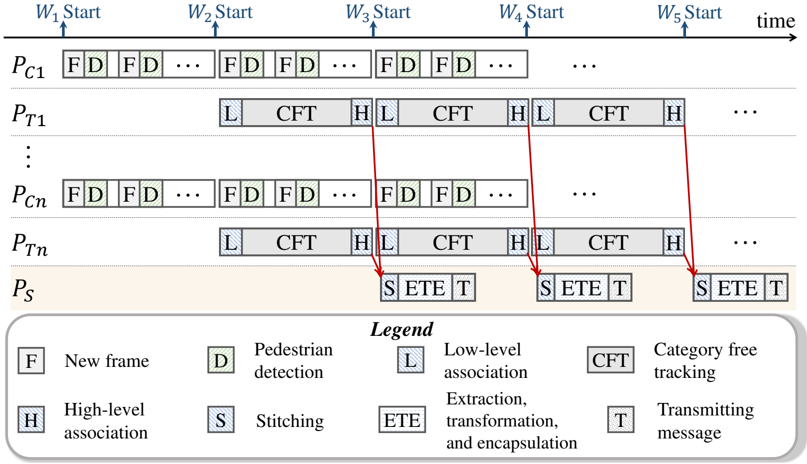


Fig. 6. Pipelined and parallel scheme. Three types of separated processes are assigned to work cooperatively on different tasks. P_{Ci} receives videos and performs pedestrian detection on each frame. P_{Ti} performs pedestrian tracking in a sliding window. P_S stitches local tracklets from different videos, processes features and transmits packets.

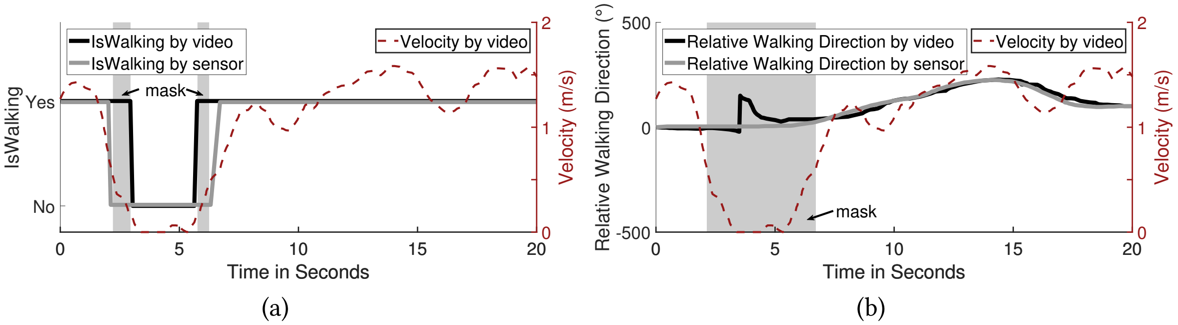


Fig. 7. Motion features from video and sensor. (a) IsWalking with masks for transition period. (b) Relative Walking Direction with masks for choppiness.

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. 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. 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).

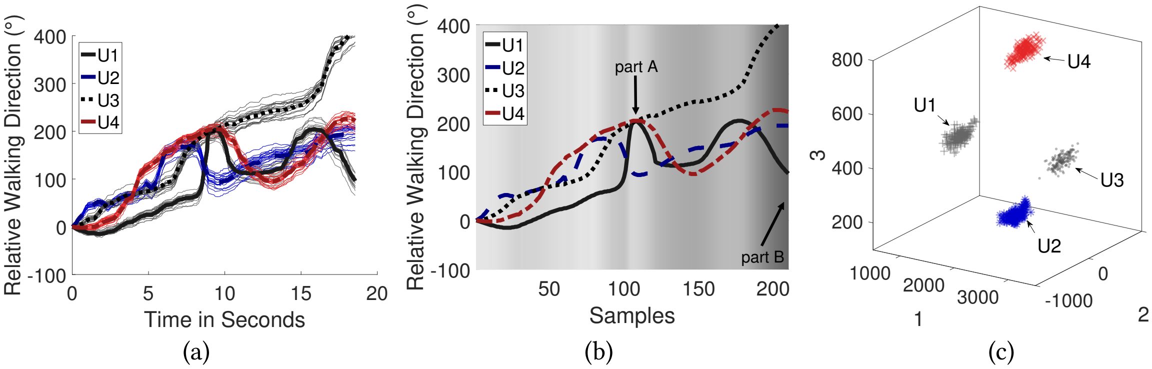


Fig. 8. Motion transformation. (a) shows Relative Walking Direction and its simulated samples generated by random walks. (b) shows how PCA emphasizes parts of Relative Walking Direction with large variance. The first principal component is visualized in the background, where darker shade means more emphasis 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.

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 still leaks behavior details which may imply important information such as human health conditions [29]. More importantly, searching for an effective subset of fragments is essentially a feature selection problem, which is known as NP-hard [1, 21]. 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. 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 [40].

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. 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. 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 1.

$$v_i^j = Coeff^j * f_i^j \tag{1}$$

Recall that $Coeff^j$ is sent in packets to ensure the same transformation on sensor side (See Fig. 2). The original motion feature can be partially recovered to f'^j_i via Equation 2.

$$f'^j_i = (Coeff^j)^T * v_i^j \tag{2}$$

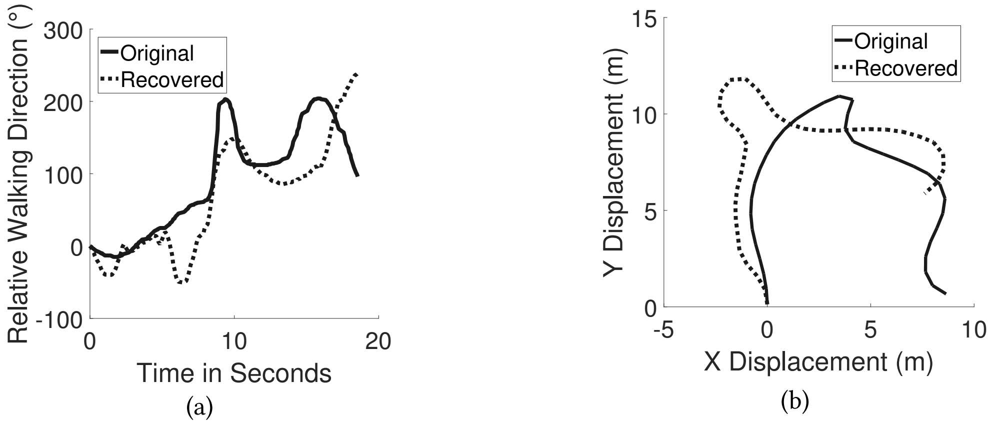
To demonstrate how much motion information can be hidden by the transformation, we choose user 1 as an example and show how much Relative Walking Direction and the walking trace can be recovered from her address code. Although the partial trend is reserved in the recovered feature vector (shown in Fig. 9(a)), locations within the trace cannot be precisely reproduced (shown in Fig. 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. 4.2. By decreasing K, more motion information can be hidden but meanwhile this may affect the matching performance.

3.4 Packet Encapsulation

Now, the server organizes all data to broadcast into a uniform format. As shown in Fig. 10, each packet is labeled with an application ID, which represents the function 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 a tolerance to various network delay. 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.

3.5 Multi-Level Decision

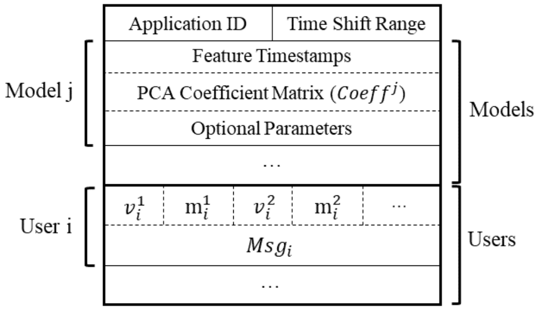
Once receiving a broadcast message, the user will first extract corresponding sensor-based motion features for each model j according to the received feature timestamps by using the methods introduced in Sec. 3.2. The



(a)

(b)

Fig. 9. Motion and trace recovery. Using PCA ($K = 3$) on user 1's Relative Walking Direction as examples, the feature vectors and the traces cannot be precisely recovered.



Application ID | Time Shift Range

Model j

Feature Timestamps

PCA Coefficient Matrix ($Coeff^j$)

Optional Parameters

...

Models

User i

v_i^1 | m_i^1 | v_i^2 | m_i^2 | ...

Msg_i

...

Users

Fig. 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.

sensing motion feature g^j is then transformed into sensing address code s^j by duplicating the same transformation process via PCA (Equation 3).

$$s^j = Coeff^j * g^j \tag{3}$$

Then the tuples $< s^1, s^2, \ldots >$ and $< v_i^1, v_i^2, \ldots >$ 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.

4 EVALUATION

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

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.

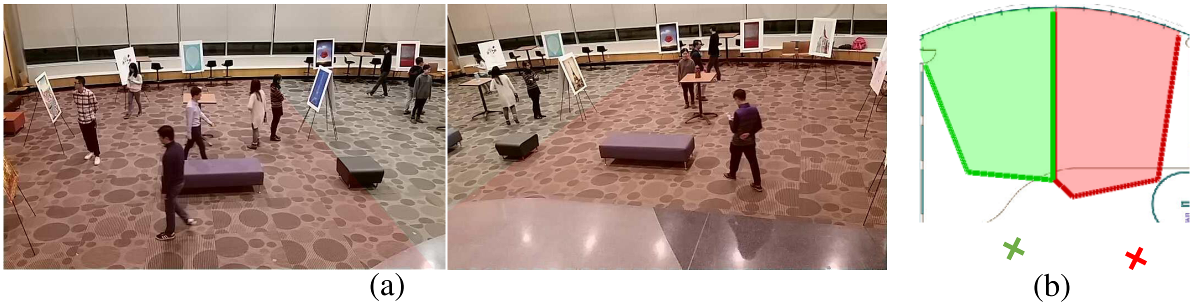


Fig. 11. Experiment scenario. (a) shows a pair of example frames with ten users walking in lobby A. (b) shows areas cooperatively covered by two cameras. where each camera is marked with a cross.

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.

4.2 Performance Results

The following questions are of our interests:

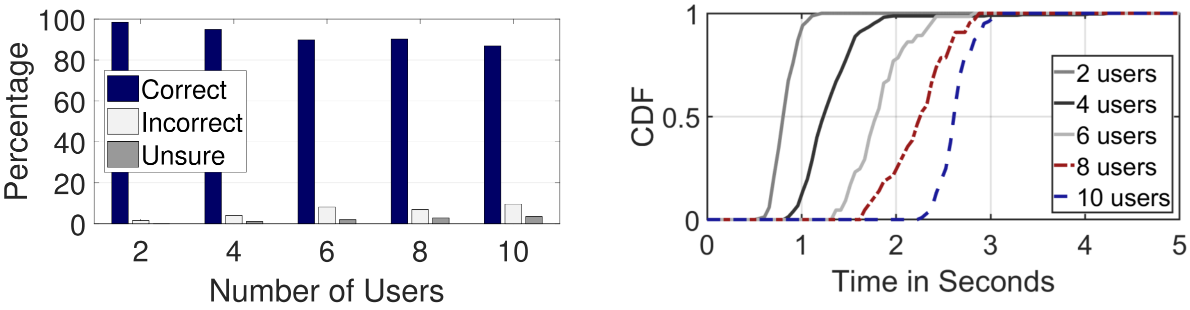
- How well does *PHADE* perform overall?
- How does *PHADE* achieve real-time tracking?
- How much is motion history blurred?
- How does *PHADE* perform in applications?

[1] 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.

(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. 11(a) illustrates the example frames with ten users in lobby A. Fig. 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. 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.



(a)

(b)

Fig. 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.

Fig. 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.

(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. 3.1.2. 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. 13(a) and Fig. 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.

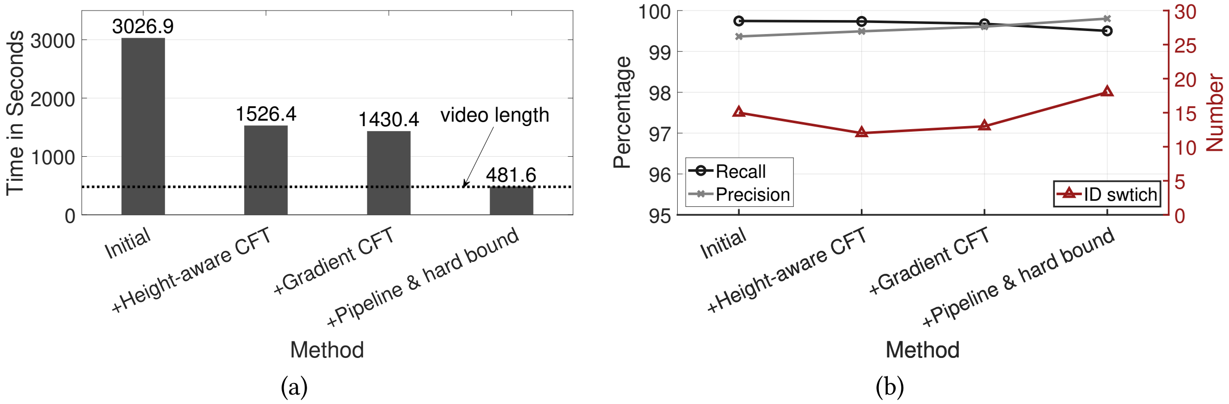


Fig. 13. Improvements in tracking time and performance. (a) shows that the tracking delay is shortened to a constant time. (b) shows that the tracking performance while incrementally adding the optimization techniques.

(3) How much is motion history blurred?

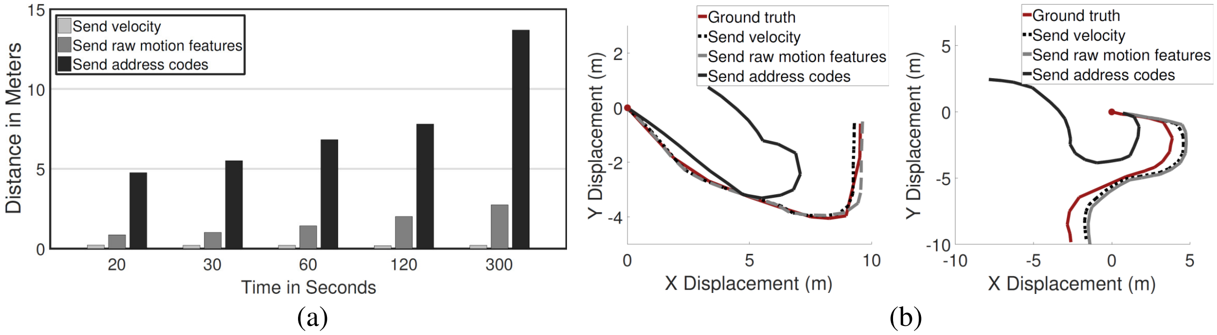


Fig. 14. Motion blurring. (a) shows the distances between the starting point of a recovered trace and the ground truth largely increase when using transformed address code, which implies modest motion leak in *PHADE*. (b) shows two examples of recovered traces using various motion features.

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.

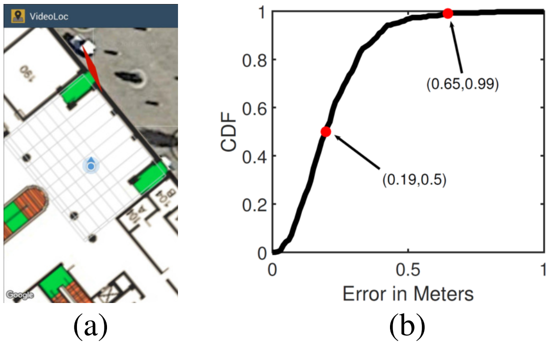
We first manually label the ground truth positions of users in session three. Walking traces are recovered from one of the following three types of motion features with various motion periods: (1) velocity (speed and corresponding direction); (2) raw motion features (i.e. IsWalking, Relative Walking 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. 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 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 [5] 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. 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 modest motion leak in *PHADE*.

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

Indoor localization. Simply using location information obtained from the video [18, 19] as the customized messages, *PHADE* can be easily adopted into a localization application. Fig. 15(a) illustrates the app we implemented to receive and show real-time locations for the phone user. Fig. 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 [27, 39]. The localization accuracy is barely affected by the number of users since the locations don't rely on any wireless signal.



(a) (b)

Fig. 15. Indoor localization. (a) The app with user position and direction on the map. (b) The localization errors.

Automatic audio guide. We also evaluate *PHADE* when it is used as an automatic audio guide. Fig. 16(a) shows how we set up for the experiment, where the minimum distance between two points is 0.5 m. Fig. 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 others schemes such as Bluetooth.

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. 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. 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

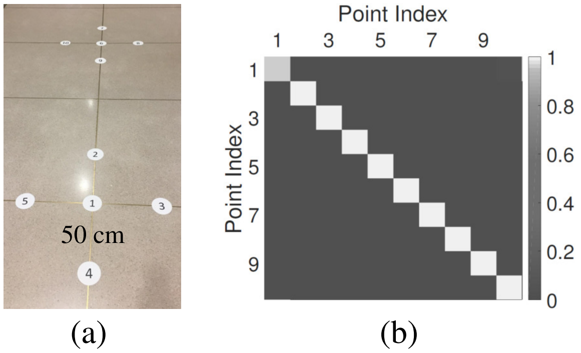


Fig. 16. Automatic audio guide. (a) Ten points setup for audio guide experiments. (b) The confusion matrix for ten pre-labeled points.

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. 17(d), where the median sending delay is 3.2 seconds and it's 4.6 seconds at 97 percentile.

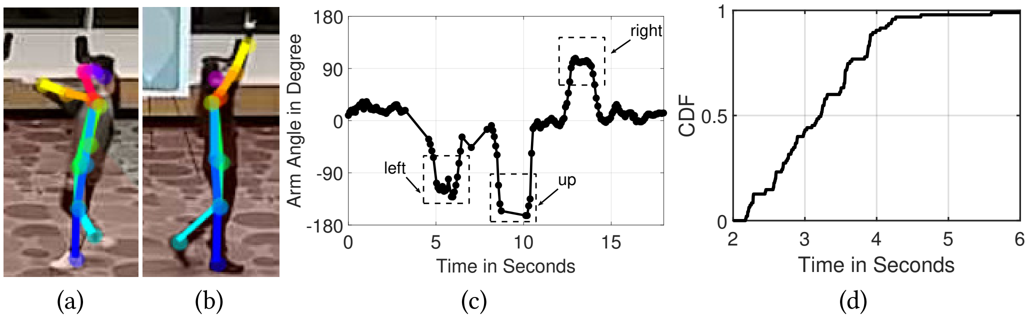


Fig. 17. Gesture-based messaging. (a) Pose when pointing to the left wall. (b) Pose when pointing to the roof. (c) shows examples of arm angles during each type of pointing gesture. The three dashed boxes represent pointing to the left wall, the roof and the right wall, respectively. (d) The time intervals since the pointing gesture starts to the corresponding message is sent.

Table 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. 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

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 been accomplished by [6], 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.

6 RELATED WORK

Camera-based communication. Traditionally, cameras are used as a receiver for information in visual communication. For example, HiLight [32] encodes data into pixel translucency changes atop any screen content to realize real-time screen-camera communication. InFrame++ [43] enables simultaneous communication for both users and devices on full-frame video contents. ARTcode [53] preserves both image and code features in one visual pattern. [3] creates a model of a screen-to-camera communication system to predict the information capacity based on 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 [35] shows that aggregate power consumption implies user's location. MoLe [47] leverages the pattern in English words to infer what a user is typing on the keyboard. [44] uses embedded sensors on wearable devices to capture inputs on ATM keypads. [9] 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 [23] 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 [22] employs image capturing and mobile sensing to develop a cognitive assistance system. Authors in [7] 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 camera 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 [25] 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 [31] uses both RFID tags worn by people and 3D depth camera to recognize and assign IDs to individuals. In [34], 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 camera view with its smartphone and requires no preregistration. Use of motion address allows it to work in public areas. And it can still correctly identify an individual even if she changes her phone. Insight [45] recognizes people through their motion patterns and clothing colors serving as a temporary fingerprint for an individual. [26] 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 [26, 45] is implemented into a real-time end-to-end system. Besides, these schemes require users to upload their sensor data. In comparison, this paper 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.

7 CONCLUSION

This paper 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.

REFERENCES

[1] Edoardo Amaldi and Viggo Kann. 1998. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science* 209, 1-2 (1998), 237–260.

[2] Anton Andriyenko and Konrad Schindler. 2011. Multi-target tracking by continuous energy minimization. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 1265–1272.

[3] Ashwin Ashok, Shubham Jain, Marco Gruteser, Narayan Mandayam, Wenjia Yuan, and Kristin Dana. 2014. Capacity of pervasive camera based communication under perspective distortions. In *Pervasive Computing and Communications (PerCom), 2014 IEEE International Conference on*. IEEE, 112–120.

[4] Hristo Bojinov, Yan Michalevsky, Gabi Nakibly, and Dan Boneh. 2014. Mobile device identification via sensor fingerprinting. *arXiv preprint arXiv:1408.1416* (2014).

[5] Raymond C Browning, Emily A Baker, Jessica A Herron, and Rodger Kram. 2006. Effects of obesity and sex on the energetic cost and preferred speed of walking. *Journal of Applied Physiology* 100, 2 (2006), 390–398.

[6] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2017. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In *CVPR*.

[7] Shaxun Chen, Amit Pande, and Prasant Mohapatra. 2014. 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 (MobiSys '14)*. ACM, New York, NY, USA, 109–122. https://doi.org/10.1145/2594368.2594373

[8] Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, Vol. 1. IEEE, 886–893.

[9] Yves-Alexandre De Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. 2013. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports* 3 (2013), 1376.

[10] Sanorita Dey, Nirupam Roy, Wenyuan Xu, Romit Roy Choudhury, and Srihari Nelakuditi. 2014. AccelPrint: Imperfections of Accelerometers Make Smartphones Trackable.. In *NDSS*.

[11] Piotr Dollár. [n. d.]. Piotr's Computer Vision Matlab Toolbox (PMT). https://github.com/pdollar/toolbox.

[12] P. Dollár, R. Appel, and W. Kienzle. 2012. Crosstalk Cascades for Frame-Rate Pedestrian Detection. In *ECCV*. 645–659.

[13] Piotr Dollár, Ron Appel, and Wolf Kienzle. 2012. Crosstalk cascades for frame-rate pedestrian detection. In *Computer Vision–ECCV 2012*. Springer, 645–659.

[14] Piotr Dollár, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. 2005. 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, 65–72.

[15] Howard Whitley Eves. 1972. *A survey of geometry*. Vol. 1. Allyn and Bacon.

[16] Olivier Faugeras. 1993. *Three-dimensional computer vision: a geometric viewpoint*. MIT press.

[17] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2001. *The elements of statistical learning*. Vol. 1. Springer series in statistics Springer, Berlin.

[18] Ardeshir Goshtasby. 1986. Piecewise linear mapping functions for image registration. *Pattern Recognition* 19, 6 (1986), 459–466.

[19] Ardeshir Goshtasby. 1988. Image registration by local approximation methods. *Image and Vision Computing* 6, 4 (1988), 255–261.

[20] Helmut Grabner, Jiri Matas, Luc Van Gool, and Philippe Cattin. 2010. Tracking the invisible: Learning where the object might be. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 1285–1292.

[21] Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. *Journal of machine learning research* 3, Mar (2003), 1157–1182.

[22] Kiryong Ha, Zhuo Chen, Wenlu Hu, Wolfgang Richter, Padmanabhan Pillai, and Mahadev Satyanarayanan. 2014. Towards wearable cognitive assistance. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*. ACM, 68–81.

[23] Puneet Jain, Justin Manweiler, and Romit Roy Choudhury. 2015. Overlay: Practical mobile augmented reality. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 331-344.

[24] Niall Jenkins. [n. d.]. 245 million video surveillance cameras installed globally in 2014. *IHS Technology* ([n. d.]).

[25] Haojian Jin, Christian Holz, and Kasper Hornbæk. 2015. 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, 147–156.

[26] Deokwoo Jung, Thiago Teixeira, and Andreas Savvides. 2010. Towards cooperative localization of wearable sensors using accelerometers and cameras. In *INFOCOM, 2010 Proceedings IEEE*. IEEE, 1–9.

[27] Iris A Junglas and Richard T Watson. 2008. Location-based services. *Commun. ACM* 51, 3 (2008), 65–69.

[28] Rudolph Emil Kalman et al. 1960. A new approach to linear filtering and prediction problems. *Journal of basic Engineering* 82, 1 (1960), 35–45.

[29] Daniel Kelly, Seamas Donnelly, and Brian Caulfield. 2015. 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, 462–465.

[30] Cheng-Hao Kuo and Ram Nevatia. 2011. How does person identity recognition help multi-person tracking?. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 1217–1224.

[31] Hanchuan Li, Peijin Zhang, Samer Al Moubayed, Shwetak N Patel, and Alanson P Sample. 2016. 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, 4933-4944.

[32] Tianxing Li, Chuankai An, Xinran Xiao, Andrew T Campbell, and Xia Zhou. 2015. Real-time screen-camera communication behind any scene. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 197–211.

[33] Baiyang Liu, Junzhou Huang, Lin Yang, and Casimir Kulikowsk. 2011. Robust tracking using local sparse appearance model and k-selection. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 1313–1320.

[34] David Fernández Llorca, Ricardo Quintero, I Parra, and MA Sotelo. 2017. Recognizing individuals in groups in outdoor environments combining stereo vision, RFID and BLE. *Cluster Computing* 20, 1 (2017), 769-779.

[35] Yan Michalevsky, Aaron Schulman, Gunaa Arumugam Veerapandian, Dan Boneh, and Gabi Nakibly. [n. d.]. PowerSpy: Location Tracking Using Mobile Device Power Analysis.

[36] AG Amitha Perera, Chukka Srinivas, Anthony Hoogs, Glen Brooksby, and Wensheng Hu. 2006. 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, 666–673.

[37] Hamed Pirsiavash, Deva Ramanan, and Charless C Fowlkes. 2011. Globally-optimal greedy algorithms for tracking a variable number of objects. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 1201–1208.

[38] Nirupam Roy, He Wang, and Romit Roy Choudhury. 2014. 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, 329–342.

[39] Jochen Schiller and Agnès Voisard. 2004. *Location-based services*. Elsevier.

[40] Matthias Scholz, Fatma Kaplan, Charles L Guy, Joachim Kopka, and Joachim Selbig. 2005. Non-linear PCA: a missing data approach. *Bioinformatics* 21, 20 (2005), 3887–3895.

[41] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. 2017. Hand Keypoint Detection in Single Images using Multiview Bootstrapping. In *CVPR*.

[42] Oncel Tuzel, Fatih Porikli, and Peter Meer. 2006. Region covariance: A fast descriptor for detection and classification. In *European conference on computer vision*. Springer, 589–600.

[43] Anran Wang, Zhuoran Li, Chunyi Peng, Guobin Shen, Gan Fang, and Bing Zeng. 2015. 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, 181–195.

[44] Chen Wang, Xiaonan Guo, Yan Wang, Yingying Chen, and Bo Liu. 2016. 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, 189–200.

[45] He Wang, Xuan Bao, Romit Roy Choudhury, and Srihari Nelakuditi. 2013. InSight: recognizing humans without face recognition. In *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*. ACM, 7.

[46] He Wang, Xuan Bao, Romit Roy Choudhury, and Srihari Nelakuditi. 2015. Visually fingerprinting humans without face recognition. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 345–358.

[47] He Wang, Ted Tsung-Te Lai, and Romit Roy Choudhury. 2015. Mole: Motion leaks through smartwatch sensors. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. ACM, 155–166.

[48] Shu Wang, Huchuan Lu, Fan Yang, and Ming-Hsuan Yang. 2011. Superpixel tracking. In *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 1323–1330.

[49] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. 2016. Convolutional pose machines. In *CVPR*.

[50] Junliang Xing, Haizhou Ai, and Shihong Lao. 2009. 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, 1200–1207.

[51] Bo Yang and Ram Nevatia. 2012. Online learned discriminative part-based appearance models for multi-human tracking. In *European Conference on Computer Vision*. Springer, 484–498.

[52] Bo Yang and Ramakant Nevatia. 2014. Multi-target tracking by online learning a CRF model of appearance and motion patterns. *International Journal of Computer Vision* 107, 2 (2014), 203–217.

[53] Zhe Yang, Yuting Bao, Chuhao Luo, Xingya Zhao, Siyu Zhu, Chunyi Peng, Yunxin Liu, and Xinbing Wang. 2016. ARTcode: preserve art and code in any image. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 904–915.

[54] Zengbin Zhang, David Chu, Xiaomeng Chen, and Thomas Moscibroda. 2012. 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, 1–14.

Received August 2017; revised February 2018; accepted April 2018