CSI:DeSpy - Enabling Effortless Spy Camera Detection via Passive Sensing of User Activities and Bitrate Variations.

MUHAMMAD SALMAN*, Inha University, South Korea
NGUYEN DAO, Inha University, South Korea
UICHIN LEE†, KAIST, South Korea
YOUNGTAE NOH†, KENTECH, South Korea.

Recently, the spy cameras spotted in private rental places have raised immense privacy concerns. The existing solutions for detecting them require additional support from synchronous external sensing or stimulus hardware such as on/off LED circuits, which require extra obligations from the user. For example, a user needs to carry a smartphone and laboriously perform preset motions (e.g., jumping, waving, and preplanned walking pattern) for synchronous sensing of acceleration signals. These requirements cause considerable discomfort to the user and limit the practicability of prevalent solutions. To cope with this, we propose *CSI:DeSpy*, an efficient and painless method by leveraging video bitrate fluctuations of the WiFi camera and the passively obtained Channel States Information (CSI) from user motion. *CSI:DeSpy* includes a self-adaptive feature that makes it robust to detect motion efficiently in multipath-rich environments. We implemented *CSI:DeSpy* on the Android platform and assessed its performance in diverse real-life scenarios, namely; (1) its reliability with the intensities of physical activities in diverse multipath-rich environments, (2) its practicability with activities of daily living, (3) its unobtrusiveness with passive sensing, and (4) its robustness to different network loads. *CSI:DeSpy* attained average detection rates of 96.6%, 96.2%, 98.5%, and 93.6% respectively.

CCS Concepts: • **Security and privacy → Privacy protections**.

Additional Key Words and Phrases: Channel State Information (CSI), Access Point (AP), Person in Line of Sight (PLoS), Person in None Line of Sight (PNLoS)

ACM Reference Format:
Muhammad Salman, Nguyen Dao, Uichin Lee, and Youngtae Noh. 2022. *CSI:DeSpy* - Enabling Effortless Spy Camera Detection via Passive Sensing of User Activities and Bitrate Variations.. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 2, Article 72 (June 2022), 27 pages. https://doi.org/10.1145/3534593

1 INTRODUCTION

With technological advancement, it is possible to make a tiny camera which can be easily implanted in everyday objects (e.g., power outlets, USB chargers, and smoke detectors, etc. [29]). Unfortunately, this advancement helped to increase the voyeurism crime [36, 37], *i.e.*, recording the individuals private zone without their consent. It was reported that over 25,000 spy cameras were seized by police in China in 2021 [56]. These spy cameras are generally installed in places where the individuals have a reasonable expectation of privacy [48]. According to an IPX1031 survey of 2000 US candidates who stayed in Airbnb, 58% of them showed their concern about spy camera, whilst 11% of them actually found spy cameras in Airbnb rental [9, 21]. Recently, the Korean National Police Agency conducted a crackdown against the spy cameras across the country and they found spy cameras in 30 different hotels, filming around 1,600 hotel guests whose footage were live streamed over the internet [14]. Moreover, there are numerous incidences reported about the spy camera spotted in the restroom, for instance, 60 minor females were filmed on a spy camera spotted in a girl's bathroom at Premier Athletics, Franklin [51]. In addition, spy cameras have also been spotted in the dressing room for changing clothes in shopping malls [33]. It

† Corresponding Authors.
Authors' addresses: Muhammad Salman, salman@inha.edu, Department of Electrical and Computer Engineering, Inha University, South Korea; Nguyen Dao, nguyen@nsl.inha.ac.kr, Department of Electrical and Computer Engineering, Inha University, South Korea; Uichin Lee, uclee@kaist.edu, School of Computing, KAIST, South Korea; Youngtae Noh, ytnoh@kentech.ac.kr, Energy AI, KENTECH, South Korea.

2474-9567/2022/6-ART72 $15.00
https://doi.org/10.1145/3534593

is heartbreaking that several victims, including celebrities, committed suicide because of losing their self-esteem due to unwanted exposures [5, 35, 50].

The commodity spy camera integrates various advanced features such as support for High Definition (HD) audio and video, night vision, and WiFi connectivity. The WiFi connectivity further reinforces the privacy invasion in two ways. First, the footage can be sent to cloud storage, which allows the attacker to record the streaming of the target scene for unlimited time without worrying about the storage issue. And second, streaming can be broadcast in real-time remotely over the internet. The attacker uses private streaming for illegal purposes, for instance, selling it over the internet [53]. Such an example is further illustrated in Fig. 1. At one end, the camera is placed in a hidden place (*i.e.*, dress stand), and the victim is unaware of its placement. This camera (hereafter spy camera) surreptitiously records the target area and keeps streaming the video to the remote (*i.e.*, cloud) through a WiFi connection. Typical users lacked adequate knowledge with poor mental models about how home networking works [66], leaving most users vulnerable to this kind of attack. At the other end, an offender can access the real-time video streaming anytime, and can further use it for blackmailing or other illegitimate purposes. Such a high privacy concern [9–11, 29, 33] has spiked the endeavor for a suitable, affordable, and reliable solution for detecting spy cameras.

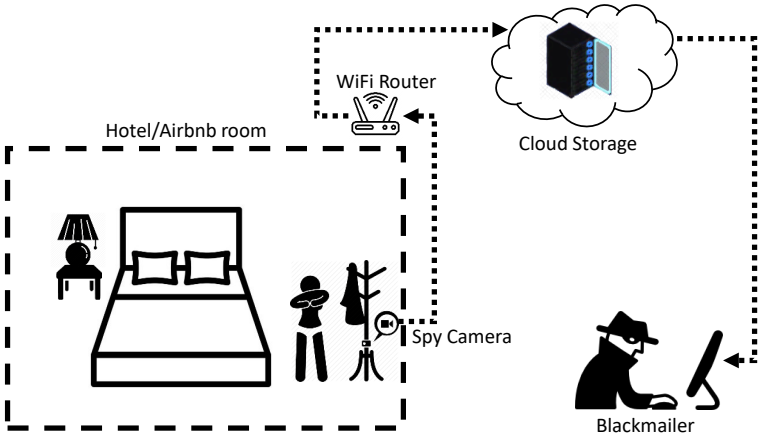


Fig. 1. An example of invading privacy by a spy camera

To ease the concerns, hardware and software-based endeavors have been made to detect the spy cameras. Commercial hardware-based solutions are on the market but they are not satisfactory due to their frequent distractions from nearby RF devices [2, 3], bulkiness [6], and prohibitive cost [49]. A handful of related works [24, 29, 57] detect the spy camera by exploiting its bitrate. Unfortunately, the bitrate-based camera detection requires a strong stimulus in the target scene [24, 29]. In addition to that, the network jitters caused by other flows in the same WLAN [29], and the repetitive spikes caused by the compression algorithm (e.g., H.264 [47]) result in unusual fluctuations in the bitrate, which consequently mislead the camera detection process. Recently, Cheng *et al.* proposed a human-assisted identification model known as DeWiCam [13] for spy camera detection inside a room. This approach identifies the existence of a spy camera based on the correlation between its unique traffic pattern (*i.e.*, bitrate distribution) and accelerometer readings from the smartphone. Although DeWiCam clearly shows a novel direction for camera detection, because it is solely a smartphone-application based solution and does not require support from additional hardware for the spy camera detection. However, it inherently includes three critical drawbacks which make the solution less practical. Firstly, vigorous physical activities (e.g., waving hands, jumping, and walking for a longer duration) are obligated to correlate accelerometer readings over the activities, and video streaming traffic for the camera detection. Secondly, preplanned actions are required where the individual has to stay still for some pre-defined time and then move to distinguish between static and dynamic states. Lastly, the user is required to carry the smartphone to sense their motion via an accelerometer.

To cope with the aforementioned problems, we propose *CSI:DeSpy* which tracks video bitrate fluctuations from WiFi camera and by using the passively obtained Channel States Information (CSI) from user movements that happen in everyday activities [28, 30]. In other words, we examine the variation for both video bitrate and CSI, and exploit their correlation to detect the existence of a spy camera.

CSI:DeSpy has practical advantages over the existing solutions. Let us consider an example of a user staying in a hotel room. To detect the spy camera within that room with the existing solution, the user would either require to perform some laborious activities [13][1], carry collection of bulky and expensive devices for the camera detection [6, 49], or use specialized scene illumination tools for bitrate stimulation [29, 57]. All these obligations are burdensome for an ordinary individual. The *CSI:DeSpy* omits them by allowing the camera detection with a higher comfort level, *i.e.*, no laborious activities, carrying bulky devices, and using scene illumination tools are desired. Nonetheless, the choice of CSI adaptation in *CSI:DeSpy* introduces inherent challenges as follows:

Effect of multipath. The size of the room and the scatters (e.g., furniture) highly affect the signatures of CSI due to the multipath effect [63]. Thus, a hard-coded threshold or supervised machine learning-based classification does not work well for diverse environmental settings. To remedy this, *CSI:DeSpy* introduces a self-adaptive method by merging density-based clustering called DBSCAN [17] with stochastic process controlling called the control chart [52] that dynamically adjusts its threshold with regard to changing environment and detects motion with high precision. We evaluated the performance of *CSI:DeSpy* in different room sizes and confirmed the average detection rate of around 99%.

Sensitivity gain. The intensity of physical activity (PA) performed by humans influences the sensitivity gain. Thanks to the aforementioned self-adaptive method, the threshold which distinguishes the sedentary (static state) and PA or non-sedentary (dynamic state) is auto-adjusted to a quasi-optimal value. Thus, PA of any intensity from light to vigorous can be detected effectively. We examine the performance of *CSI:DeSpy* at different intensities of PA and achieve 96.6% of the average detection rate for a light PA (e.g., changing a jacket), which is only 3% lower than vigorous PA (e.g., jumping).

Preplanned actions. The activities of daily living (ADL), such as changing dresses, pressing clothes, and brushing hairs include both sedentary behavior and physical activities. However, the duration and instances of these states occur randomly. Unlike the existing solution [13], which requires discomforting preplanned actions (*i.e.*, staying static and then moving for an unnaturally large amount of time) from the user, *CSI:DeSpy* automatically tracks the user's sedentary behavior from the static cluster's formation and works effectively with any ADL. *CSI:DeSpy* shows an average detection rate of 96.2% for diverse ADL cases.

Passive sensing. Carrying a smartphone for camera detection is quite burdensome. It is highly desirable for such a system to support passive sensing mode. There are two related effects to be considered. Firstly, the communication range between the camera and smartphone: nevertheless, the smartphone belonging to the user and the camera are co-located, the RSSI link between them is either strong ($\leq$ -50 dBm) or good ($\leq$ -60 dBm). Secondly, the ambient CSI fluctuation is caused by the motion of Person-in-Non-Line-of-Sight (PNLoS) to the camera. Fortunately, the effect of PNLoS is much lower than that of the person in a target area, namely Person-in-Line-of-Sight (PLoS) to the camera. *CSI:DeSpy* attains an average detection rate of 98%, and 98.5% with and without PNLoS while placing the smartphone at a different position relative to the camera.

To our best knowledge, there is no reliable solution to detect a spy camera without pain. To cope with this, we put the above inherent challenges into the design and propose *CSI:DeSpy*. To make the system reliable, practical, and unobtrusive, *CSI:DeSpy* gets rid of the obligations from users, namely laborious physical activity, preplanned actions, and carrying the smartphone. The major contributions of *CSI:DeSpy* are summarized as follows:

- To minimize the user's effort, we adapt the CSI readings and check the correlations of fluctuation with the bitrate observed by the WiFi camera during the user's motion. Compared to the existing solution, we justify that it is effortless (*i.e.*, works with light activity), needs no preplanned actions (works with any ADL), works passively (works without carrying), and is effective (high detection rate and lower false alarm rate).
- To show the feasibility of *CSI:DeSpy*, we incorporate the feature extraction from CSI and video traffic, self-adaptive motion detection, and camera detection algorithm. We built an Android app named *CSI:DeSpy*.
- To thoroughly assess the performance of *CSI:DeSpy* in abundant real-life scenarios, we deliver the results to verify its reliability assessment (assessing the reliability with the intensities of physical activities, and

[1]Which might not be feasible for a special group of individuals, such as pregnant ladies, and elderly people.

diversity of environment), practicability (assessing the performance with activities of daily living), and unobtrusiveness (assessing the performance with passive sensing). We also examine *CSI:DeSpy* on the smartphone as an Android application for camera detection in real-time.

2 RELATED WORK

Recently, on the gloomy side, the spy camera voyeurism has raised serious privacy concerns. On the bright side, this has led to increased endeavors for finding a reliable defensive solution against the privacy invasion. Below are the existing solutions accompanied by the endeavors.

Commercial solutions available in the market (*i.e.*, Wireless Camera RF Detector [2] and Spy RF Signal Detector [3]) are designed for detecting the RF signal from the wireless camera. However, these devices are unreliable and can easily be interfered by other RF devices working on 2.4 GHz or 5 GHz. Some existing commercial services reported by BBC [6] employ a collection of devices for spy camera detection (e.g., illumination and wireless signal checking), which are difficult to operate by typical users. Another device referred to the Spy camera hunter [49] detects and displays the video streaming by a wireless camera. This device works over a range of frequencies such as 1.2 GHz, 2.4 GHz, and 5.8 GHz. However, it only identifies streaming from an analog video source such as PAL, NTSC, and SECAM, which are not widely used these days. To sum up, two reasons make the hardware-based solution infeasible to use—they are expensive and cumbersome.

Recently, several attempts have been made for solutions based on traffic analysis to detect the spy camera. Wu *et al.* [57] presented a camera detection technique called similarity of simultaneous observation—correlating the traffic patterns of a known wireless camera with the other network devices to detect whether they are observing the same environment simultaneously. However, this work does not consider various parameters that can mislead the camera detection; for instance, resolution difference, codec difference, variation in a complex scene observed from a different angle, and the difference in the protocol used for the data transmission (TCP or UDP). Another approach based on the traffic analysis tracks the changes in the physical environment based on the light stimuli and observes the response in terms of bandwidth usage. Using this approach, Liu *et al.* [29] used two ways to detect the spy camera: firstly, they manually turn on/off the lights and recorded the bitrate response, and secondly, they leverage a flashing LED circuit called Flicker to probe the camera and examine the camera's bitrate in response to it. Lagesse *et al.* [24] also used the same approach by using a smartphone with external WiFi in promiscuous mode to sniff the traffic from a web camera (*i.e.*, spy camera) that was recording a scene. They stimulated the bitrate of the spy camera on the smartphone's flashlight so that the camera traffic can be distinguished from the mixed traffic. Nevertheless, camera detection based on scene illumination has three crucial limitations. Firstly strong stimulus is required to induce the bitrate of the camera. However, regardless of the strong stimulus provision, the bitrate encounters periodic spikes that are cause by the compression algorithm (e.g., H.264 [47] which periodically transmits MTU frames). Such spikes mislead the classification. Secondly, if there are multiple concurrent flows available in the same network, they cause network jitters and consequently cause fluctuation in the bitrate despite a non-flashing and static scene. Lastly, their model requires a stable target environment (i.e., no motion in the scene) to precisely track the pixel changes that correspond to the flashing stimulus. More recently, Cheng *et al.* proposed DeWiCam [13] which requires the smartphone in the monitoring mode and captures the wireless signals from the air. Based on the unique traffic patterns (*i.e.*, packet length distributions) and smartphone acceleration changes over the movements, it analyzes the correlation between the two and identifies the existence of wireless spy cameras. However, the DeWiCam has three critical drawbacks which make it impractical: (1) use accelerometer sensor to sense the user motion, hence the user needs to perform labor-intensive motions (e.g., jumping and waving hands), 2) do preplanned actions (e.g., staying still for a certain period of time and then moving), and (3) carry the smartphone to record accelerometer readings over the movements.

CSI:DeSpy gets rid of all the obligations introduced by the aforementioned works, and offers a painless solution with the following advantages: (1) no labor-intensive motion (*i.e.*, reliable sensitivity) required, (2) no preplanned actions (*i.e.*, working with ADL) needed, and (3) no need to carry the *CSI:DeSpy*'s-enabled smartphone (*i.e.*, passive-mode detection) for the spy camera detection.

3 ADVERSARY MODEL AND PROTECTION GOALS

Now consider an attacker who wants to hijack the privacy of the user staying in a private place as in the prior studies [13, 24, 29, 57, 68]. The adversary knows the credentials of WLAN and associates the spy camera to the

AP with reasonable signal strength. The adversary may use a single or multiple spy cameras in order to cover the entire target scene for real-time video streaming. Moreover, the adversary has also control over the spy camera parameters such as controlling the video resolution and turning on/off its audio. In addition, the adversary has control to view the monitored area remotely in real-time and also stores the streaming from the camera to cloud storage. This adversary model appears to be quite invasive, but a recent privacy research [66] revealed that most users did not have proper knowledge about how home networking works, and further lacked concerns of security and privacy risks of home IoT devices despite potential security attacks.

The goal of *CSI:DeSpy* is to prevent the privacy invasion of the user in any private place. *CSI:DeSpy* should be sensitive enough to detect light daily living activities such as changing jacket, washing hands, etc. To make our problem settings more realistic, for example, we can even assume that there is a spy camera hidden in an unsuspected package and the user is unaware of its placement. The spy camera coexists with the abundant wireless traffic generated from various devices (e.g., laptop, IPTV, and smartphone). To get rid of the dependency on connecting to a specific access point where the spy camera is transmitting, we further consider that the *CSI:DeSpy*-enabled smartphone supports traffic monitoring. Moreover, we confirm that the spy camera can be detected despite the video encryption. In order to achieve spy camera detection with a user's minimum effort, *CSI:DeSpy* should not require the user to perform labor-intensive physical activities, preplanned action, and carry the smartphone.

4 OUR METHODOLOGY

In this section, we discuss *CSI:DeSpy*'s methodology, which includes *CSI:DeSpy*'s background, data pre-processing, feature extractions from both CSI and video traffic, motion detection, and its implementation.

4.1 Background of the *CSI:DeSpy*

Herein, we discuss basics on major building-blocks of *CSI:DeSpy* namely Channel State Information (CSI) and wireless camera traffic.

4.1.1 Channel State Information. In modern wireless networks such as 802.11g/n/ac, CSI is used to ensure reliable communication with high data rates in MIMO systems [60], [20]. The wireless signals when propagating in a rich multipath environment suffer from multiple reflected copies of a signal, which finally degrade the channel quality. The wireless channel is further distorted with motions in the background. The distortion caused by motion includes signal's Doppler frequency spread, amplitude attenuation and phase shift [63]. As shown in prior studies, this additive distortion caused by motion can positively be exploited for enabling various applications such as fall detection [55], crowd counting [58], activity recognition in indoor environment [12] and decimeter level indoor localization [41, 61].

4.1.2 Wireless Camera Traffic. The WiFi cameras first encode their videos by using variable bitrate, where the coding rate is adapted to the scene's complexity [29]. Then, a compression mechanism is applied to the encoded video for the sake of saving the bandwidth [13]. The commodity WiFi cameras commonly use H.264 compression techniques [47], which include three types of pictures: Intra-coded (I), Predicted (P), and Bidirectional predicted (B) pictures [23, 64]. Based on Group of Pictures (GoP), the video data is compressed. The I-picture serves as a reference point for the other pictures in the GoP, because it encodes the complete scenes independently. In contrast, the B- and P-pictures encode the inter-scene variations [29]. Thus, the GoP leads to a lower bitrate transmission when there is no change in the scene (*i.e.*, still scene), and on the contrary, a higher bitrate is transmitted when there is variation in the scene (*i.e.*, dynamic scene).

In summary, the human movement in the target scene causes fluctuation in both the CSI and camera's bitrate. Such fluctuations are correlated in both worlds and these can be exploited for the detection of a spy camera.

4.2 Raw Data Pre-processing

The received CSI and bitrate data in the raw form contain superfluous distortions including short spikes and zero bins. Thus, the received data needs to be sanitized before feature extraction.

4.2.1 CSI Cleaning. The recent network standards such as 802.11 g/n/ac use the OFDM modulation to transmit data. The data or payload field of a packet is composed of numerous subcarriers (*i.e.*, 64, 114, 242, and 484 for channel width of 20MHz, 40MHz, 80MHz, and 160MHz respectively) [8]. They have been classified into three

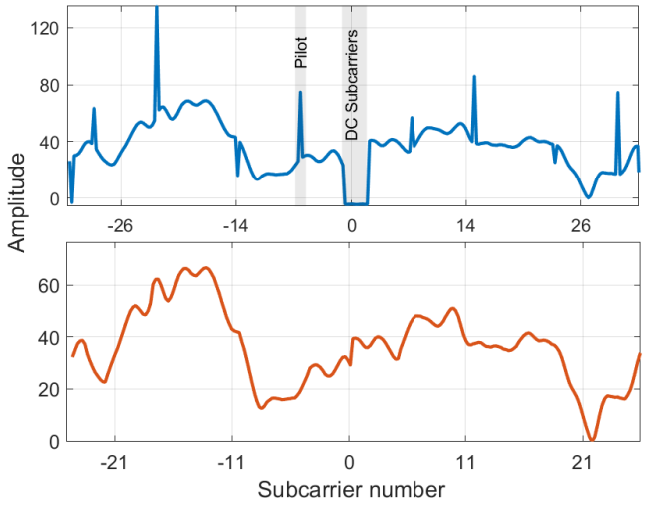


Fig. 2. CSI signal with (top) and without (bottom) pilot and DC subcarriers.

categories: 1) the data subcarriers which carry the modulated data, 2) the pilot subcarriers which track amplitude, frequency, and phase fluctuations, and 3) the DC (or null) subcarriers which do not carry any data but are used as a guard carrier to avoid interference from the adjacent channels. On receiving, the pilots form short spikes[2] while the DC subcarriers make zero bin as shown in Fig. 2. Thus, the presence of pilot and DC subcarriers will introduce errors in motion detection. To improve the accuracy of motion detection, these subcarriers should be discarded based on their indices [18]. For instance, 802.11ac standard has a total of 64 subcarriers in case of 20MHz channel width[3], which are composed of 48 data, 8 pilots, and 8 DC subcarriers. Fig. 2 (bottom) shows the data subcarriers after filtering pilots and DC subcarriers.

In addition to the errors caused by pilots and DC subcarriers, the noise added to the CSI due to the complex indoor environment (e.g., the room size, obstacles like furniture, and so forth) must be eliminated as it contributes to further errors in motion detection. However, such noise is distributed randomly in all the CSI's bands, therefore the conventional time and frequency domain filters are not very effective to mitigate it. For instance, leveraging a time domain filter (e.g., mean or median filters) can further distort the signal by replacing the information in the noise-free band with that of the noisy one [31]. Additionally, frequency-domain filter, e.g., Butterworth filter has a slow fall off[4] that incorporates the stopband's residual noise into the passbands [38]. To overcome these limitations, we use the in-band noise filter, namely Discrete Wavelet Transform (DWT). The DWT transforms the noisy signal into the wavelet domain by decomposing it into multiple wavelet basis signals. The wavelet level shows the time behavior of the signal expressed in a time series of coefficients. To use DWT efficiently for the in-band noise removal whilst preserving the desired human motion information in the CSI, the parameters such as wavelet type, sampling rate, cut-off threshold, and order, should be chosen carefully. More specifically, the noise from the CSI should be removed in such a way that the information belonging to the human motion is preserved while the unwanted harmonics due to complex indoor environment are removed. For this reason, the cut-off threshold for the DWT is computed by choosing the Nyquist sampling rate as 500Hz as in [27]. This is sufficient enough to accurately estimate the maximum Doppler frequency f_d from the speed v of human walking and running, which is on average 1.5 and 5m/s respectively [7]. This f_d for the human motion can be derived as

[2]Pilots have almost double the amplitude of the data subcarriers
[3]Note that the number of subcarriers is different for the different channel width. e.g., for 40MHz the number of subcarriers is 114.
[4]Although Butterworth filter is widely used [40] due to its maximum flat amplitude response in the passband, and out-of-band noise removal.

follows:

$$f_d = \frac{2v}{c} \times f_c. \tag{1}$$

Using the commodity WiFi frequency (f_c) of 2.4 GHz, we achieve f_d of 24 Hz for walking and 80Hz for running. Finally, by choosing the 80 Hz as cutoff threshold for DWT [42], the noise levels above this cutoff can be efficiently eliminated.

4.2.2 Camera Traffic Identification and Bitrate Smoothing. The commodity WiFi cameras adopt H.264 compression technique [47]. This technique generates small-sized data packets when there is no change in the scene in order to minimize the data transferring size. However, when there is motion in the target scene both full-sized and small-sized packets are transmitted. The camera traffic can be identified from the mixed traffic (e.g., laptop, IPTV, and smartphone, etc.) by investigating the two-dimension feature vector [13]: (1) Transient Packet length distribution (PLD) feature. As a result of H.264 compression, the traffic of the WiFi camera is composed of a group of pictures (GoP) that are the combination of a specific pattern of full-sized and small-sized packets. Such GoP based traffic is distinct from mixed traffic. Thus, this can be extracted easily by using the CDF over n number of packets. (2) Bandwidth stability feature. Since the camera traffic is relatively stable than those of other devices, this feature can be computed from the standard deviation of the instantaneous bandwidth of each flow.

After the camera traffic is identified from the mixed traffic, its payload part is read. However, during live streaming, the I-frames are recurrently transferred and thus can cause false motion detection [29]. To mitigate the unnecessary spikes due to I-frame transmission, we use a sliding window of 250ms over the received packets and apply a moving median filter to achieve a smoothed bitrate.

4.3 Feature Extraction

4.3.1 Feature Extraction from CSI. The pre-processed CSI can be represented as:

$$H = [H_1, H_2, ...H_i, ..., H_N]^T, i \in [1, 48], \tag{2}$$

where N is the number of subcarriers. The H_i subcarrier can be defined as:

$$H_i = |H_i| e^{j \sin\{\angle H_i\}}, \tag{3}$$

where $|H_i|$ and $\angle H_i$ denote the amplitude and phase responses for the i_{th} subcarrier respectively. Diverse approaches can be applied to detect a user's motion in the target area by utilizing the amplitude [59], phase [62] or both terms [39]. We use Principal Component Analysis (PCA) for extracting the $CSI_{features}$ by leveraging the amplitude's variation of the CSI. To do so, a window W of length n is defined, which slides over the received packets. While a slide happens, the subcarriers of each packet are correlated (using Pearson correlation [59]) with the rest of the packets in the same W. The correlation result is stored in a form of $n \times n$ matrix. This matrix is further normalized by the size of n, and then the eigenvector is computed from the normalized correlation matrix. Finally, The $CSI_{feature}$ is derived from the eigenvector as follows:

$$CSI_{feature} = max(eigenvector). \tag{4}$$

This $CSI_{feature}$ shows the maximum variance of the principal component (*a.k.a.* first principal component). In particular, this feature identifies the amount of maximum variation in CSI data due to motion [22]. For the static case (*i.e.*, no scene variation) the $CSI_{feature}$ is equal or very close to 1. Conversely, it starts to fluctuate drastically between 0 and 1 in the dynamic case (*i.e.*, motions in the scenes).

4.3.2 Features Extraction from Network Traffic. We define two windows of 250ms which are sliding back-to-back (*i.e.*, successive) over the smoothed bitrate. The feature value is computed as follows: (1) the covariance matrix is computed between them. (2) the eigenvector is derived from the covariance matrix, and finally (3) the maximum eigenvalue (*i.e.*, first principal component) is computed. This feature value for the bitrate corresponds to the variation in the bitrate values between the two successive windows (*i.e.*, 0.5 second). For instance, they experience distinct rising and falling trends with a high correlation between successive windows when there is motion in the target scene (*i.e.*, dynamic). On the contrary, during no mobility in the target scene (*i.e.*, static), the smoothed bitrate retains a lower value, with slight variations due to periodic occurrence of I-frame. Thus, the successive windows show lower covariance and thereby a lower feature value.

4.4 Motion Detection

The complexity of the indoor settings (e.g., room size, furniture, and so forth) could highly influence CSI due to the multipath effect. A hard-coded threshold or supervised machine learning does not fit into every setting to classify the static and mobile cases robustly. For instance, a model trained for a room without scatters (e.g., furniture) will not work effectively for that room with scatters. To address this problem, we need a classification method that adapts to the changes in the environment.

Herein, we introduce a stochastic process control method called the control chart (a.k.a., Shewhart chart) method, which is widely used in the field of industrial engineering to manufacture products with high quality [52]. This method works robustly for detecting motion in real-time in a time series $CSI_{feature}$. Nevertheless, it requires a self-adaptive method for adjusting its threshold[5] (*i.e.*, control limit) according to the variation in the indoor environment. We first introduce a lightweight variant of *CSI:DeSpy* which adjusts its control limit based on historical data. We then present a more advanced self-adaptive method that leverages auto-tuned density-based clustering (or DBSCAN [17]) for computing the control limits.

4.4.1 Control Chart Method. In the control chart method, three lines can be determined based on historical data. The central line is the average line of the historical data and control lines, namely upper control limit (UCL) and lower control limit (LCL) [52]. The data can be treated as an outlier when it surpasses either the UCL or LCL. Algorithm 1 thoroughly explains the flow of events for spy camera detection.

Algorithm 1: PSEUDOCODE FOR CONTROL CHART METHOD

Input: n is the number of data sampling, $Csi_{RECENT}[n] \leftarrow 0$, $Bitrate_{RECENT}[n] \leftarrow 0$, $Bitrate_{STATIC}[n] \leftarrow 0$;
Output: $SpyCamList[]$;

begin
  Find candidate spy cameras $Mac_{CAM}[]$ *by inspecting the Packet Length Distribution*
  foreach $i \in Mac_{CAM}[]$ **do**
    while $isSpyCam = true$ *or* $t >$ *time limit* **do**
      Sniff the $Packets_{CAM}(t)$ *from* $Mac_{CAM}[i]$ *and calcutate the* $Bitrate_{CAM}(t)$;
      Extract the $Csi_{CAM}(t)$ *CSI data from the* $Mac_{CAM}[i]$;
      `// Detect the motion by analyzing the the bitrate and CSI`
      $isSpyCam \leftarrow$ `ControlChart(`$Bitrate_{CAM}(t), Csi_{CAM}(t)$`)` ;
    if $isSpyCam = true$ **then**
      Add $Mac_{CAM}[i]$ *to the* $SpyCamList[]$;
  return $SpyCamList[]$;

Subroutine `ControlChart(`$Bitrate$,Csi`)`**:**
  Update Csi *in* $Csi_{RECENT}[]$ *in n-sized buffer*;
  Update $Bitrate$ *in* $Bitrate_{RECENT}[]$ *in n-sized buffer*;
  $LCL \leftarrow max(Csi_{RECENT}[]) - \alpha$;
  if $Csi > LCL$ **then**
    Add $Bitrate$ *to* $Bitrate_{STATIC}$;
  else
    $UCL \leftarrow 3\times$ std(Bitrate$_{STATIC}$);
    if $Bitrate > UCL$ **then**
      return true;
  return false;

The flow of the algorithm is summarized as follows: Firstly, the smartphone's WiFi is set to monitoring mode to inspect the Packet Length Distribution (PLD) of all the flows. If the camera's PLD is found according to the format explained in [13], its MAC address is extracted from that flow. As multiple cameras could be available in the target location; hence all the camera flows are subjected to the spy camera inspection to verify whether

[5]The threshold is for accurately discriminating different states. For example, distinguishing $CSI_{feature}$ belonging to the static state from the mobility state

or not they are recording the user in the monitored area. Then the bitrate and CSI are extracted from the MAC address of the spy camera stored in a buffer called Mac_{Cam} as shown in lines (5)~(6).

Both the features (CSI and bitrate) are analyzed in (7) by the $ControlChart()$ subroutine to examine the availability of a spy camera. The lower control limit (LCL) in (14) is computed for CSI where the universal value for α is exhaustively searched in the range $(0\% \sim 20\%) \times CSI_{feature}$. We determined 5% of the $CSI_{feature}$ as the universal value for α based on its reasonable performance over diverse indoor scenarios[6]. Thus, as long as the instantaneous $CSI_{feature}$ value is greater than the LCL, it is regarded as a static condition. When it falls below this threshold, the state becomes dynamic (*i.e.*, mobility is sensed in the target scene). At that instance, the control limit is computed for the bitrate. The camera detection is anticipated if the instantaneous bitrate value also crosses the control limit (UCL).

The control chart-based method is a lightweight method for detecting the camera in real-time. However, the control limits threshold is not very robust to discriminate static and dynamic states effectively in diverse indoor environmental settings. To make the control limits self-adaptive to the variation in the indoor environment, we integrate an unsupervised based clustering method called DBSCAN [17] in the control chart method. In the following, we explain the DBSCAN method and auto-tuning for its input parameters. We then explain the *CSI:DeSpy* method, which merges the control chart and auto-tuned DBSCAN clustering.

4.4.2 DBSCAN Clustering. The key insight of DBSCAN is that a neighborhood of a particular radius is defined for each data point, which becomes a core point if it contains at least a minimum number of other data points in its neighborhood. Thus, DBSCAN has two important input parameters. (1) epsilon (ε): the radius of the data point, (2) MinPts: the minimum number of points to be contained in ε-neighborhood. This type of clustering method has two main benefits. Firstly, clusters are formed based on their density, and hence no prior knowledge is required for the numbers of cluster formation. Secondly, clusters can be formed with arbitrary shapes as long as parameters are properly configured.

We compute the Euclidean distance between the $CSI_{feature}$ values. This represents how far one feature value f_i is from another feature value f_j. We then implement the DBSCAN method for the CSI. For the cluster formation, the following conditions must hold.

- **Condition 1:** The feature value is said to be a core point, if it occupies at least MinPts in its radius ε-neighborhood.
- **Condition 2:** If any two given core points p_1 and p_2 are directly density reachable either with symmetry between pairs of core points, or asymmetry between one border point and one core point; they belong to the same cluster.
- **Condition 3:** A feature value f_i is said to be an outlier, if it occupies less than MinPts and does not share any core point[7] with a cluster.

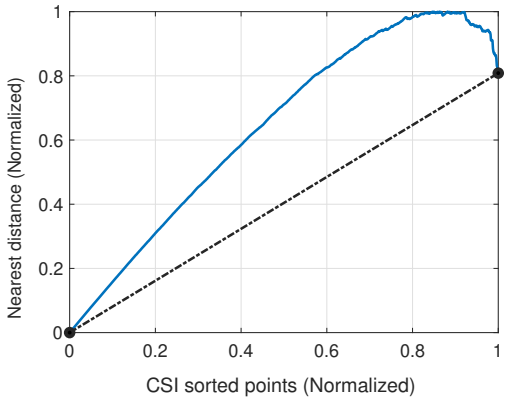
Because of the multipath effect, the CSI value changes dramatically with the variations in the environmental settings. This effect causes the distance variation between the feature values. More specifically, the distance between the $CSI_{feature}$ values in a small room will be different from a big room. Owing to this property, ε and MinPts threshold also vary with the change in a given environment.

Traditionally the value of ε is determined by manually finding the first "valley" [17] or appropriate "knee" [19] in the sorted k-dist graph. Such a valley or knee is the bifurcation point that separates the cluster from the noise. Moreover, MinPts is chosen by setting it equal to the number of dimensions in the data set [19]. However, the valley or knee keeps changing for the CSI with variation in the indoor environment due to the multipath effect as mentioned above. Hence, using fixed values for the parameters (ε and MinPts) do not work for the DBSCAN in diverse environmental settings. Furthermore, in most cases the valley or knee point is not clearly identifiable in the sorted k-dist graphs [45]. To address these issues, we devised an auto-tuned way for determining the ε and MinPts. Firstly we use the kneedle approach [44] to estimate the knee point automatically, and then we perform auto-tuning to further adapt the ε and MinPts thresholds to different environmental settings.

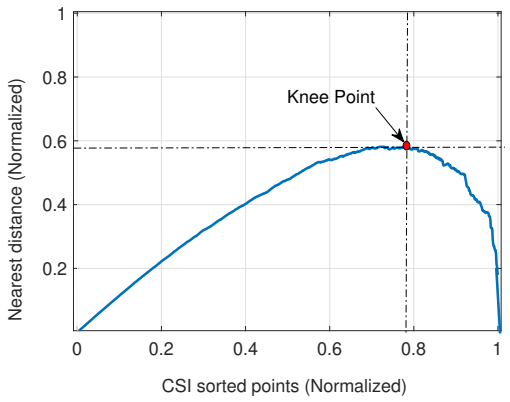
Finding the knee point. We leverage the kneedle approach to determine the ε of the data set in stochastic environments. The main idea of kneedle is to find the point of maximum curvature in a data set that deviates from a straight line. The knee point determined with this procedure lies in between the values which are slowly

[6]This has been explained in detail in Section 5.1.2.

[7]Conversely, if it shares a core point then it is called a border point



(a) Negative concave form of the sorted k-dist graph (normalized curve)



(b) Finding the knee point in maximum curvature (normalized curve).

Fig. 3. (a) Forming negative concavity, (b) Adjusting the curve to determine the local maximim.

changing (congested core points) and the values which are drastically changing (*i.e.*, dispersed core points or outliers). Intuitively, this is the bifurcation point in a data set that separates two clusters (e.g., static and dynamic clusters). Finding the "knee point" involves the following steps:

(1) The k-dist graph is flipped by 180 degree about (x_{min}, y_{min}) along the line formed by the endpoints—(x_{min}, y_{min}) and (x_{max}, y_{max}). This transforms it into negative concavity. The purpose of the line between the endpoints is to preserve the overall behavior of the data set. Moreover, in order to maintain the original trend or shape, the data points on both axes have been normalized, as shown in Fig. 3(a).
(2) The curve is rotated by θ degrees clockwise about (x_{min}, y_{min}) until the line formed by the endpoints overlaps the x-axis.
(3) Since the data points have been sorted in increasing order, therefore there is always a single global maximum. Thus, the peak of the curve is the desired knee point as shown in Fig. 3(b).

DBSCAN parameter auto-tuning. Although kneedle provides a good estimate of ε which works robustly when reasonable data points of $CSI_{feature}$ are available for both static and dynamic states. However, there are two critical issues where the default kneedle fails to work. (1) When the duration of one of the two states is very brief, the untuned parameters obtained from kneedle would mix the data points belonging to the brief state with that of a longer duration state. (2) The variation in an indoor environment causes different multipath effect, and thus severe noise. As a results, the static data points could be identified as dynamic. Due to these issues, a single cluster is obtained for two different states (*i.e.*, static and dynamic). This leads us to two conclusions. First, the value of ε is so large that all the data points qualify the MinPts criteria for the core point, and each point in the cluster becomes direct density reachable. Second, the MinPts value is considerably smaller than the desired value that all the data points become the core points, and consequently, each point becomes direct density reachable in spite of smaller ε. To overcome these issues, we introduce an auto-tuning mechanism for the MinPts and the ε obtained from kneedle, as illustrated in Algorithm 2. This algorithm works as follows: The parameters estimated by the kneedle *i.e.*, ε-neighborhood value for the data point, and MinPts for the core point condition are provided as input parameters to the DBSCAN. Based on these input parameters the DBSCAN computes the number of clusters from the data set. The goal is to determine distinct clusters (≥ 2) for each state (*i.e.*, static and dynamic).[8] Initially, the MinPts is set equal to the size of the data set dimension as in [19]. If this value is not sufficient, it keeps increasing up to a threshold σ. The σ has been chosen as 5 times the size of the dimension of the data set. We use different ranges of thresholds for σ, ranging from 1 to 10 times the size of dimensions of the data set.

[8]If by default the number of clusters is already greater than or equal to 2, then the auto-tuning algorithm will be skipped. In that case, the clusters belonging to different states are distinguishable with the default parameters obtained from the kneedle.

Algorithm 2: DBSCAN's parameters Fine-tuning

Input: data: CSI or bitrate feature data set;
ε: knee point determined by the kneedle;
MinPts: the size of data set dimension
Output: clusters

```
begin
    clusters ← DBSCAN(data, ε, MinPts);
    // To ensure that we have more than 1 cluster
    while |clusters| < 2 do
        clusters ← DBSCAN(data, ε, MinPts);
        MinPts ← MinPts+1;
        if MinPts>σ then
            ε ← ε-1
```

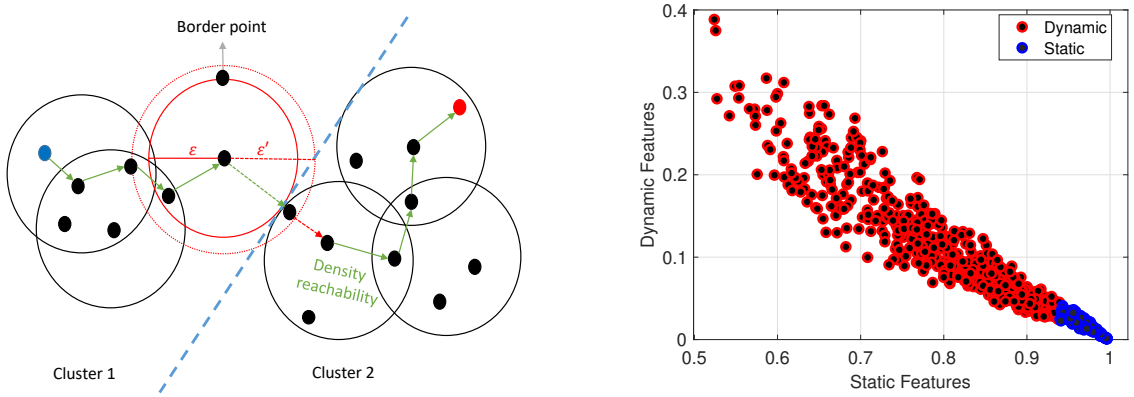
We found that for the same value of ε further increase in σ (*i.e.*, more than 5 times) does not have a significant impact on the number of clusters. Thus, when the σ threshold is reached, the ε is slightly reduced, and reiterate the process until the numbers of clusters greater than or equal to 2 are obtained. This finally sets the tuned values for ε and MinPts.

We illustrate it with an example shown in Fig. 4(a). Initially, the untuned parameters (*i.e.*, improper ε and MinPts' size) are used for the data points belonging to static and dynamic states. As a result, a single cluster is formed, where every data point is density reachable (e.g., the leftmost blue data point is density reachable to the rightmost red data point). To separate the clusters belonging to these two different states (*i.e.*, static and dynamic state), the auto-tuning method is applied, which automatically adjusts a proper radius (*i.e.*, from ε' – dotted-red circle to ε – solid-red circle) and MinPts' size for the data points. The tuned parameters finally separate the two different clusters as shown in Fig. 4(b).

Note that this method makes the clustering process fully automated and does not require any extra parameters or active input from the user. It is also worth mentioning that our auto-tuning method is not influenced by the device type used for collecting the CSI data points or the router type used for the spy camera connectivity. As discussed in Section 4.5, the CSI is extracted from a particular location in the frame, which remain the same regardless of the device used for CSI collection. Moreover, in our setting, the CSI data is collected from the physical layer, where the source for CSI collection is the WLAN device of the spy camera. Thus, the router type and its lower layer's security mechanism do not affect the CSI collection. The auto-tuning method is insusceptible to any unexpected changes taking place in the target scenario. For instance, if the router place is changed (within the communication range). The individual's motion in that area would still be causing fluctuation to the CSI data points. Therefore, this method would effectively detect the individual movements from the passively collected CSI data points.

Identifying the static cluster. As the fine tuning ensures at least two clusters formation, *i.e.*, static and dynamic clusters. The static cluster is formed when $CSI_{feature}$ value is close to 1. On the contrary, a dynamic cluster is formed when $CSI_{feature}$ drops to a lower value—up to 0.5 in Fig. 4(b). We illustrate the cluster formation as a result of the static and dynamic states with an example shown in Fig. 5. We deliberately performed a predicted pattern of motion (staying still and moving for some time) in order to show the coherence of user motion with the CSI and bitrate fluctuation over time. In the beginning, the user is static from 0~10s. During this time, the clustering algorithm will detect the static conditions such as $CSI_{feature}$ close to 1 and forms a cluster with more congested data points. As the user moves from 10~30s, both The CSI and bitrate features fluctuate vigorously. During the dynamic duration, $CSI_{feature}$ are varying and they have more dispersed data points in their cluster. Based on these static and dynamic clusters formation at common timestamps, the spy camera is detected. It is noteworthy that the bitrate traffic and CSI are collected from a single camera source (*i.e.*, same MAC address) at a time[9]. This omits the possibility of coincidence that a camera might be detected based on using the CSI fluctuation from one camera while the bitrate traffic from another one. More specifically, let us suppose that

[9]As there might be multiple cameras available in the target place, However, *CSI:DeSpy* considers one camera for spy camera inspection at a time and then check for the others.



(a) fine-tuning (b) DBSCAN classification for CSI data

Fig. 4. (a) Auto-tuning the knee point, (b) Clustering classification of static and dynamic data.

there is a non-target outdoor wireless CCTV camera, and an individual is moving inside a room located in the vicinity of that camera. In this case, the individual could cause some fluctuation to the CSI (as part of CSI might penetrate the wall) but could not affect the bitrate because of not being in the camera's visible range. Therefore, *CSI:DeSpy* could not find any correlation in both quantities and does not anticipate it a spy camera.

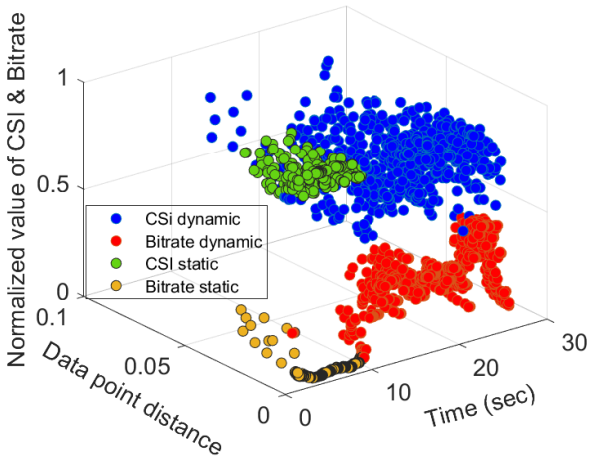


Figure 5. Cluster formation from CSI and bitrate features at common timestamps.

4.4.3 CSI:DeSpy's Self-adaptive Classification. *CSI:DeSpy*'s self-adaptive classification method leverages the fine-tuned DBSCAN method to intelligently and accurate compute the threshold for the control chart. The control chart takes this threshold as its control limits and hence reliable performance is achieved regardless of the variation in indoor environmental settings, and the intensity with which the activity has been performed.

The pseudocode of *CSI:DeSpy*'s self-adaptive classification is illustrated in Algorithm 3. During the initialization, the DBSCAN method collects samples in batch mode to compute the thresholds from the static clusters of the CSI.

Algorithm 3: *CSI:DeSpy*'s self-adaptive method

Input: $CSI_{feature}$, $Bitrate_{feature}$, $C_{limit} \leftarrow 0$, k$\leftarrow$0, timeout$\leftarrow$10s, $B_{static}[\,] \leftarrow 0$
Output: $isSpyCam$

begin
  if $C_{limit} == 0$ **then**
    // Determine the control limits for the control chart using the DBSCAN method
    $C_{limit} \leftarrow$ DBSCANClustering($CSI_{feature}$);
  timer$\leftarrow$0;
  while $isSpyCam$ ==*false* $\vee$ $timer \leq timeout$ **do**
    $\tau \leftarrow$ ControlChart(C_{limit}, $CSI_{feature}$, $Bitrate_{feature}$);
    // τ is true if detection happens
    if $k < \delta$ **then**
      // k is the number of successive detections for δ amount
      if $\tau ==$ *true* **then**
        $k \leftarrow k + 1$;
        break;
      else
        $k \leftarrow 0$;
        break;
      $isSpyCam \leftarrow$false;
    else
      $isSpyCam \leftarrow$true;
  return $isSpyCam$;

Subroutine DBSCANClustering($Data$):
  // Find the ε using kneedle and apply the fine-tuning Algorithm 2 to determine $\varepsilon_{finetuned}$, $MinPts_{finetuned}$
  Data$_{StaticCluster}\leftarrow$ DBSCAN(Data, $\varepsilon_{finetuned}$, $MinPts_{finetuned}$);
  // $Data_{StaticCluster}$ is the data belonging to the static cluster
  $C_{limit} \leftarrow$ Minimum($Data_{StaticCluster}$);
  return C_{limit};

Subroutine ControlChart(C_{limit}, $CSI_{feature}$, $Bitrate_{feature}$):
  $LCL \leftarrow C_{limit}$;
  if $CSI_{feature} > LCL$ **then**
    Add bitrate to the $B_{static}[n]$ *buffer of size n*;
    $UCL \leftarrow 3 \times std(B_{static})$;
    if $Bitrate_{feature} > UCL$ **then**
      $\tau \leftarrow$ true;
    else
      $\tau \leftarrow$ false;
  return τ

This threshold is used by the control chart method for its control limit in real-time[10]. During the static period, the instantaneous $CSI_{feature}$ stays above the control limit. At the same time, $Bitrate_{feature}$ is collected in a buffer called B_{static}. As the $CSI_{feature}$ violates the control limit (LCL), $Bitrate_{feature}$'s violation is also checked for its control limit. When a motion is detected in both features simultaneously, the $ControlChart()$ subroutine returns a true value. When there are several successive detections from the control chart, then the camera detection is anticipated.

[10]The motion is detected in both CSI and bitrate based on common timestamps

4.5 Implementation

In our settings, we used the Nexus 5 smartphone. We use two wireless cameras named DEATTI (2.4GHz) and VSTAR (dual band) for generating the WiFi camera traffic. We choose these cameras based on their rich features such as high-resolution support (1080p), latest compression support for video (H.264, and H.265), and audio (AAC). The AP used for the camera connection were NETGEAR N600, ASUS AC-2900, and TP-LINK AC-1750 (all of them support the dual mode operation).[11]

We tested our system both in real-time and offline mode. For real-time mode, we build an Android application named *CSI:DeSpy*. *CSI:DeSpy* sweeps all the available WiFi channels to obtain the MAC addresses and channel information of the available cameras. After that, the OFDM-modulated CSI are extracted from the WiFi camera's frames which are collected by listening on UDP socket 5500. These frames have a variety of information in addition to the CSI data. The bytes reserved in the header and payload field of the frame holds the following information.

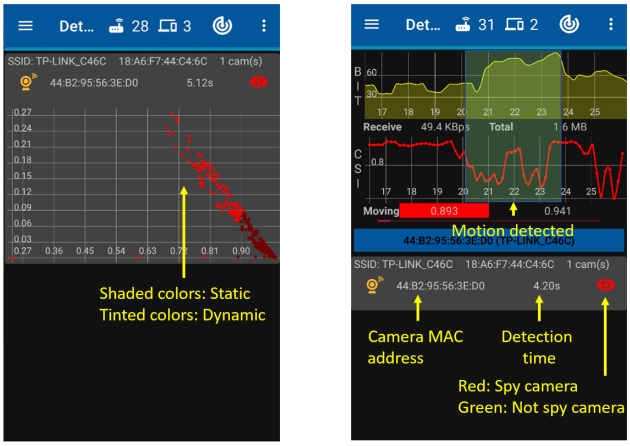
- **Header:** (1) The first 6 bytes and the following 6 bytes of the header field are reserved for the destination IP and the source IP addresses respectively. (2) The next 20 bytes are reserved for IPv4, and (3) the following 8 bytes hold the UDP information.
- **Payload:** The data field starts after the above 42 bytes of the header field. The payload field contains (1) 4 magic bytes used for distinguishing the CSI frames from others, (2) followed by 6 bytes reserved for the source mac (*i.e.*, camera's mac), (3) the following 2 bytes are reserved for the WiFi sequence number which is used for triggering the collection of CSI, (4) then next 2 bytes are reserved for antenna core and spatial stream information, (5) following 2 bytes contain the information related to the channels specifications, and (6) then next 2 bytes are reserved for the chip version. (7) The actual CSI data follows after the abovementioned 60 bytes (including the header bytes).

We extract the OFDM-modulated CSI from the correct location of the frame. The received UDP packets on the listening port (*i.e.*, 5500) has a source address as 10.10.10.10 and a broadcasting destination address (*i.e.*, 255.255.255.255). Note that the CSI can be received on multiple spatial streams if the receiving device has multiple antennas (*i.e.*, MIMO support). However, the Nexus 5 smartphone has a single receiving antenna, thereby we were receiving the WiFi frames on a single spacial stream. Alongside the CSI, we also collected the bitrate in raw form from the given camera's transmission packets through a UDP socket. The raw CSI and bitrate are firstly preprocessed, and then features are extracted. The features extracted from CSI and bitrate are buffered for a given window size (e.g., 8 seconds in our setting), and then the fine-tuned DBSCAN method is applied over the window to determine the static and dynamic clusters as shown in Fig. 6(a). The control limit is computed from the static cluster shown with shaded data points. The control chart method exploits the control limit to distinguish the static and dynamic states. Fig. 6(b) illustrates the real-time camera detection using the control chart method, where the upper chart labeled as BIT represents the bitrate and the lower chart represents the CSI. The $CSI_{feature}$ shows a static state if the instantaneous $CSI_{feature}$ remains above the lower control limit. During this state, the bitrate features for the static condition are stored in the buffer. The motion is detected in the target area when the lower control limit is crossed by the $CSI_{feature}$. During the dynamic state, the control chart checks the violation for the bitrate feature at the same time. When the control limit is violated by both the CSI (*i.e.*, crossing the lower control limit) and bitrate (*i.e.*, exceeding the upper control limits) for a consecutive amount of time, the camera is detected.

For the offline data collection, we connected the Nexus-5 smartphone via a USB cable with MacBook Pro (13 inch 2018), and collected the CSI and bitrate through the Android Debug Bridge (ADB). To ensure a reliable connection between camera and AP we considered strong ($RSSI \leq -50dBm$) and good ($RSSI \leq -60dBm$) signal strength between them. In contrast, the poor RSSI level ($RSSI \leq -70dBm$) is inefficient for data transfer [67], and therefore we did not consider it. Moreover, the smartphone belonging to the user in the target area (where the spy camera is situated) makes a strong or good RSSI link with the spy camera. Owing to a better link quality between them, the *CSI:DeSpy*-enabled smartphone effectively collects the CSI and bitrate from the camera.

Discussion: Note that our implementation can be generalized to any other smartphone or device (such as raspberry pi) by enabling the monitoring mode feature on its WLAN card. Nexmon's open-sourced GitHub

[11]Note that the AP type is not a strict requirement, as aforementioned, our system works fine regardless of any AP used. In some experiment we had used APs of different brands available at the university campus, as discussed in Section 4.1.1.



(a) Batch mode threshold computation by the DBSCAN method.

(b) Real-time detection by *CSI:DeSpy*'s self-adaptive method.

Fig. 6. (a) Header and payload fields of CSI' frame, (b) The snapshot of *CSI:DeSpy* app. (c) Camera detection using control chart mode. Detection with clustering mode.

repository [46] provide this feature for a diverse variety of smartphone and devices that has Broadcom or Cypress WiFi chips. It is noteworthy that we leverage the CSI from the WiFi transmission for motion detection, and thus, the AP must support OFDM symbol transmission. Such a transmission is supported by recent WiFi standards, such as IEEE 802.11g or the latest (such as IEEE 802.11 n/ac, etc.) standards.[12] Note also that thanks to the monitoring mode feature, our system bypasses the security protocols used by the router. More specifically, it sniffs the wireless network traffic in the physical layer, and thereby, neither it needs to associate with a particular WLAN nor required to decode or hack the transmission packets. Nevertheless, it observes the fluctuation of channel state information (extracted from the OFDM symbol's transmission) and the length of the camera packet (bitrate) caused by someone moving in the target area.

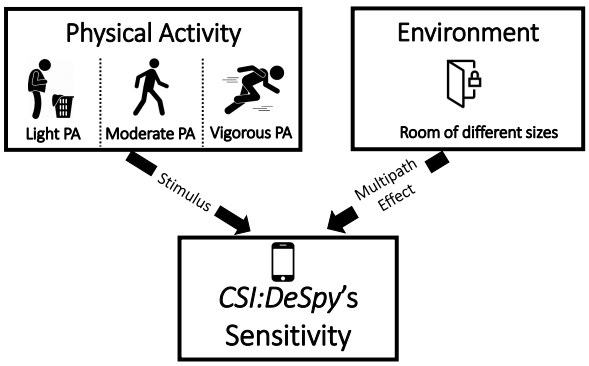
5 EXPERIMENTAL SETTINGS AND RESULTS

In this section, we explain experimental settings and the practical scenarios which evaluate the system parameters of the *CSI:DeSpy* with: (1) reliability, (2) practicability, and (3) unobtrusiveness.

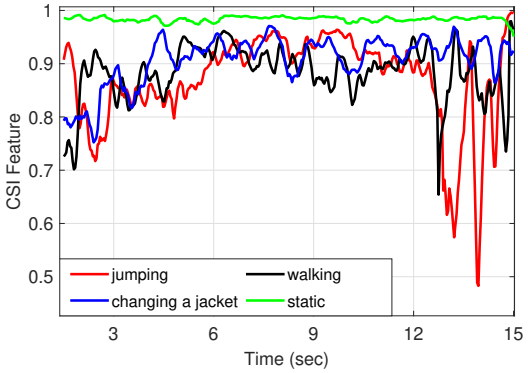
5.1 Reliability Assessment

5.1.1 Testbed Settings. To test the reliability of *CSI:DeSpy*, we measure its sensitivity with regard to the intensity of physical activity (PA) performed and the diversity of indoor settings as shown in Fig. 7(a). The former factor is crucial as this can degrade the reliability when the high intensity of PA is required for spy camera detection. The latter factor can also crucially affect the reliability of the system if it requires specific environmental settings (e.g., large room). To assess the performance of *CSI:DeSpy* with regards to PAs, we considered three types of PAs, namely light PA where the individual is changing the jacket, moderate PA where the user is walking around the room at a moderate pace ($\approx$ 1.3 m/s), and lastly, the vigorous PA in which the user is skipping in the target space ($\approx$ 3 skips/sec). This experiment aims to measure the robustness of detection accuracy in terms of various PA intensities in an uncontrolled environment. In particular, the self-adaptability feature of *CSI:DeSpy* can be evaluated by the effect of different multipath, whilst the robustness of classification can be assessed by the activity of different intensities.

[12]Majority of the WiFi devices are supporting the OFDM symbol transmission. For instance, it has been reported in [54] that over 70% of the WiFi devices are using IEEE 802.11g standard since 2017, and nowadays (after the year 2020) IEEE 802.11ac is evolving very fast (e.g., 70% office devices are using this standard).



(a) Impact of physical activity and environment on *CSI:DeSpy*'s sensitivity.



(b) Fluctuation in CSI caused by different PAs.

Fig. 7. (a) *CSI:DeSpy*'s sensitivity (b) $CSI_{feature}$ for diverse PA.

We consider different uncontrolled environmental factors that can affect the performance of *CSI:DeSpy*. Firstly, we consider three rooms with different sizes, namely small, medium, and large sizes to assess the impact of multipath. Secondly, we consider the available wireless networks (SSIDs) in the surroundings to account for the co-channel interference, which can be caused by different APs transmitting on the same channel that eventually weakens the RSSI. Note that we conducted our experiment at a large university, where the small room was the restroom with maximum of 10 accessible SSIDs, the medium room was an office with 15 SSIDs, and the large room was the research lab with 15 SSIDs. These factors are presented in Table 1. We recorded each data sample for a duration of 30 seconds. The initial 10 seconds were recorded in a static condition and after 10~30 seconds, PA of diverse intensities (such as light, moderate and vigorous) were performed.

Table 1. Various factors affecting the experiments:

Factors	small	medium	large
Room size (m)	3.5×3.5	4×5	8×8
Available SSIDs	10	15	15

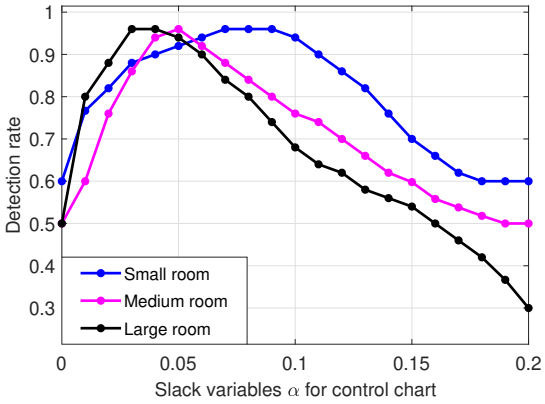
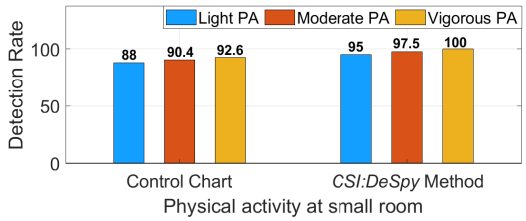
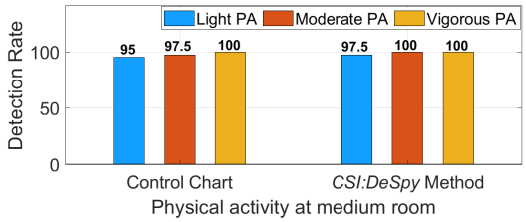


Fig. 8. Slack variable determined for different environment.

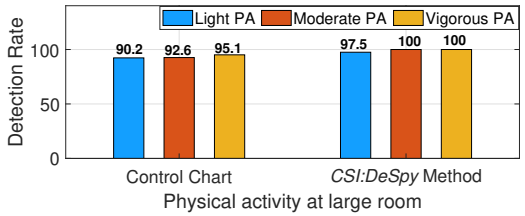
5.1.2 Performance Evaluation. We noticed that during the static state, the CSI in a small room experiences an unstable behavior because of a more severe multipath effect, which is exacerbated by the smaller spacing between walls. However, the CSI in larger rooms has a more stable behavior (*i.e.*, milder multipath effect) owing to the increased spacing between walls. Moreover, for the stimulus, we conduct the physical activity of variable



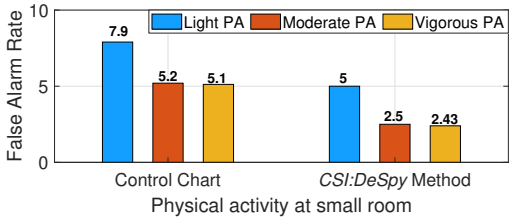
(a) Detection rate of different physical activities in a small room.



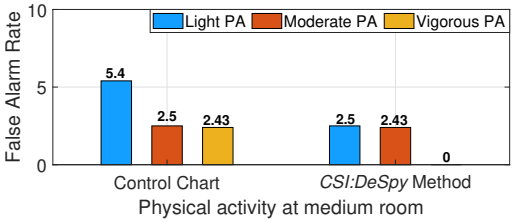
(b) Detection rate of different physical activities in a medium room.



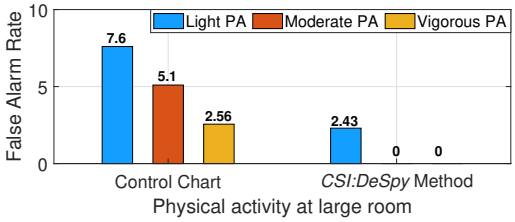
(c) Detection rate of different physical activities in a large room.



(d) False alarm rate of different physical activities in a small room.



(e) False alarm rate of different physical activities at medium room.



(f) False alarm rate of different physical activities at large room.

Fig. 9. **(a)**~**(c)** Detection rate of different intensities of PA performed at diverse indoor environments. **(d)**~**(f)** False alarm rate of PA at variable instances performed at diverse indoor environments.

intensities, namely light, moderate, and vigorous PA. From Fig. 7(b) we also noticed that the fluctuation in CSI caused by the PA is proportional to its intensity, whereby light PA leads to lower fluctuations in the CSI, while vigorous PA results in higher fluctuations. Considering the effect of both factors, the CSI experiences higher fluctuations in a small room even in the static state whilst lower variation with a light PA. Both factors in a setting lead to the worst case for motion detection. On the contrary, the CSI in a larger room shows a more stable behavior in the static state and exhibits distinct variations with any physical activity performed. In the followings, we study the combined impact of the aforementioned factors:

We exploit two variants of *CSI:DeSpy* to evaluate the reliability. The first one is a plain control chart method that does not contain the automatic threshold adjustment. However, the second one is a control chart equipped with an automatic threshold adaptation feature (hereafter, *CSI:DeSpy* method). In the plain control chart method, the performance is significantly affected by the threshold choice as shown in Fig. 8. We present a range of slack variable α for computing the control limit threshold for CSI in different indoor environments. It is evident that the maximum detection rate is attained at different threshold values for different indoor settings. For instance, the small room is more susceptible to multipath than the medium and large room, thereby experiencing more fluctuation in the static case. Consequently, the small room has a maximum detection rate (around 96%) at α values of 0.07~0.09 whereas the CSI in the medium and large room has better stability in their static case, and thus they have a maximum detection rate at α value of 0.05 and 0.03 respectively. Based on the observation of slack variables with diverse room-size settings, we select a well-fit threshold value (*i.e.*, α value of 0.05) owing to

its reasonable detection rates for all the cases. However, this hard-coded threshold results in slightly degrading the performance in some rooms (*i.e.*, small and large rooms). In contrast, the *CSI:DeSpy* method leverages the auto-tune DBSCAN to overcome the inadaptability of the plain control chart. This method robustly adapts its threshold to the desired value regardless of the room size and PA intensity.

We deliver the performance comparison of the plain control chart and *CSI:DeSpy* method in Fig. 9, where the impact of both the physical activities and variation in indoor settings have been demonstrated. As evident from Fig. 9(a)~9(c), both the intensities of PA, and indoor settings have obvious implications on the detection rate of the control chart and *CSI:DeSpy* method. The control chart method has the worst performance in the small room, where the detection rate is 88%, 90.4%, and 92.6% with the light, moderate and vigorous PAs respectively. This is due to the unstable CSI behavior in static state (*i.e.*, sedentary of the individual) and non-optimal threshold usage for motion detection. On the contrary, the control chart in the medium room outperforms the small and large rooms in terms of detection rate due to the usage of an accurate threshold for the control limit as depicted in Fig. 8. It achieves 95%, 97.5%, and 100% of detection rate with the light, moderate, and vigorous PA, respectively. Finally, the control chart in the large room also shows slight poor performance than that of the medium room, however it has a slight improved performance than the small room owing to the stable static state of CSI. The control chart in the large room achieves 90.2%, 92.6%, and 95.1% of detection rate with the light, moderate, and vigorous PA, respectively. The *CSI:DeSpy* method outperforms the control chart method in all types of indoor environments and PA intensities. The detection rate of *CSI:DeSpy* method in the small room is 95%, 97.5%, and 100% for light, moderate and vigorous PAs, respectively. As the room size increases, the static state becomes more stable.

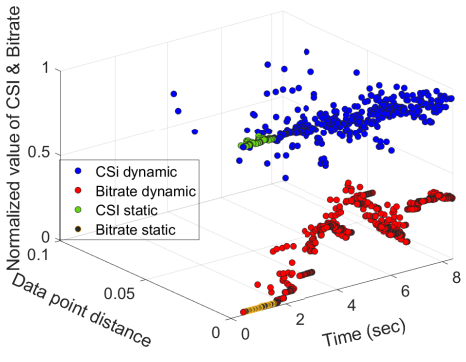
The performance comparison of the control chart and *CSI:DeSpy* method is more evident with the false alarm rate shown in Fig. 9(d)~9(f). We see that the control chart has the highest false alarm rate in the small room, where it is around 7.9%, 5.2%, and 5.1% with a light, moderate and vigorous PA, respectively. However, the control chart shows a better performance in the medium room in terms of false alarm rate, where it is around 5.4%, 2.5%, and 2.43% with the light, moderate and vigorous PA. Lastly, the control chart in the large room has a slightly lower false alarm rate than the small room, and it is 7.6%, 5.1%, and 2.56% for the corresponding light, moderate and vigorous PAs. In contrast, the false alarm rate for the *CSI:DeSpy* method is significantly lower than the control chart method. *CSI:DeSpy* method in the small room has the false alarm rate of 5% for light PA, 2.5% for moderate, and 2.43% for vigorous PA. However, in the medium room, the false alarm rate is 2.5%, 2.43%, and 0% for light, moderate and vigorous PA respectively. Finally, the false alarm rate recorded by *CSI:DeSpy* method in the large room for light PA is 2.43% and 0% for moderate and vigorous PAs.

In summary, it is evident from the above analysis that regardless of the multipath rich environment, *CSI:DeSpy* method can efficiently exploit its self-adaptive feature and performs effectively even in the worst case scenario (*i.e.*, detecting light physical activity in the small room). This proves that *CSI:DeSpy* shows reliable performance in diverse environments and its robustness to multipath effects.

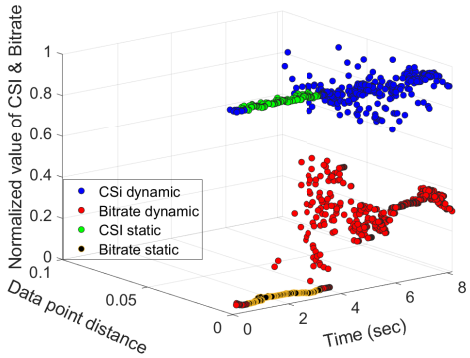
5.2 Practicability Assessment

5.2.1 Testbed Settings. The existing solutions [13] impose some preplanned obligations to the users to stay still for a particular amount of time and then move. This series of actions (*i.e.*, pre-training) enables them to easily distinguish the static and dynamic states. Unfortunately, such a requirement causes discomfort to an individual and finally degrades the practicability of the systems. *CSI:DeSpy*, however, does not impose any preplanned actions and conveniently work with the physical activities of daily living (ADL) such as changing clothes, trimming the beard, and pressing clothes etc. [25]. To evaluate the performance of *CSI:DeSpy* without preplanned actions, we set up the following four scenarios.

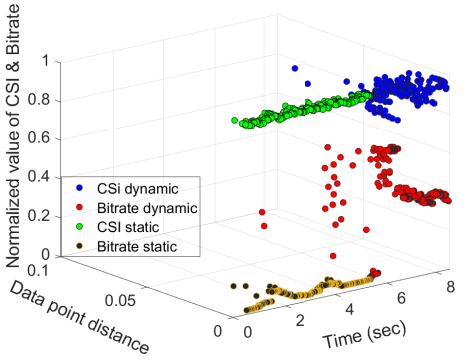
In the first scenario, we consider a smaller duration for the static period (2 seconds) than that of the dynamic (6 seconds). It is quite natural in an ADL such as changing clothes where sedentary (or static) actions like opening the button or zipper require a very little amount of time than the physical activity (or dynamic) such as taking off the clothes and hanging them on stands or wardrobe. In the second scenario, the duration of sedentary behavior is equal to that of physical activity. Such behavior is also common in ADL cases, like pressing the clothes, getting in and out of bed, and so forth. In contrast to the first scenario, we also consider the third one where the dynamic instance (2 seconds) is lower than that of the static (6 seconds). Such activity includes trimming the beard and using the toilet etc. Finally, we consider the fourth one where the user is sedentary for the entire duration such as an individual using a phone or laptop, nail care, or sitting on the chair, etc.



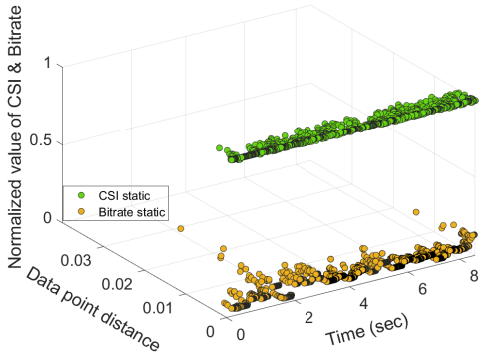
(a) Scenario 1: 2 seconds of static duration.



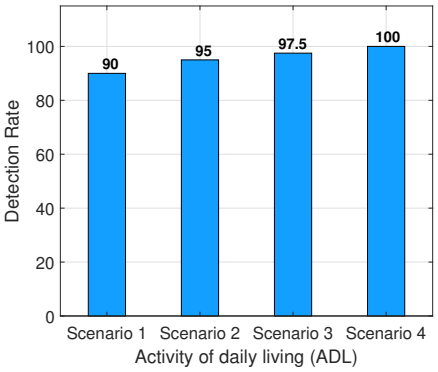
(b) Scenario 2: 4 seconds of static duration.



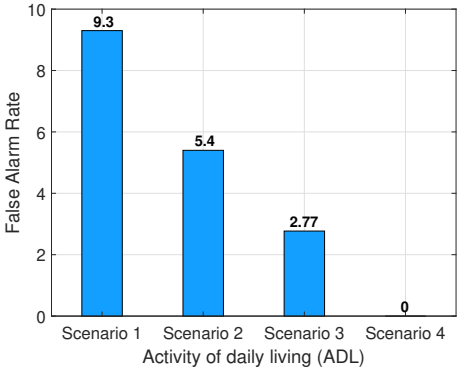
(c) Scenario 3: 6 seconds of static duration.



(d) Scenario 4: fully static.



(e) Detection rate of CSI:DeSpy method on ADL.



(f) False alarm rate of CSI:DeSpy method on ADL.

Fig. 10. **(a)**~**(d)** Cluster formation from datasets of physical activities performed at variable instances. **(e)**~**(f)** Performance evaluation of *CSI:DeSpy*

It is noteworthy that the ADL cases are natural as they are mixed up with sedentary and physical activities in our regular lives. In real-life indoor settings, 2 seconds of sedentary duration is also acceptable for the *CSI:DeSpy* method to determine the control limit of the control chart. Less than 2 seconds leads to a higher false alarm rate

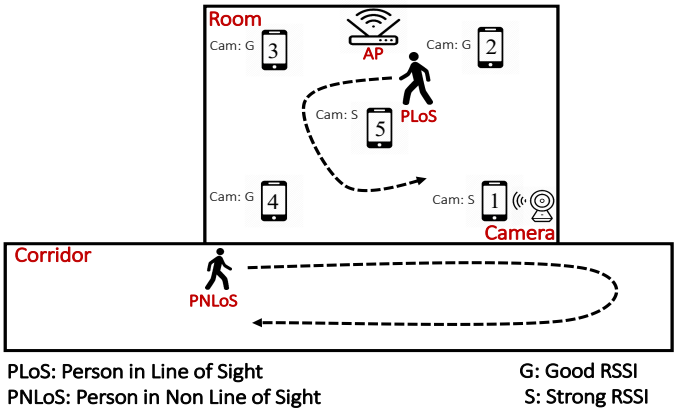


Fig. 11. *CSI:DeSpy*-enabled smartphone at different locations relative to the camera

due to the errors caused by the noise in video bitrate measurement. The bitrate encounters three types of errors in detecting the shorter sedentary duration. First, the network jitters vary the bitrate measurement regardless of a stable scene [29]. Second, the transmission of the I-frame (belonging to GoP) during the brief sedentary interval can also eliminate that static instance. Third, the packet loss during the brief sedentary interval also distorts its occurrence. Owing to these noises brief sedentary interval (< 2 seconds) is not observed in bitrate measurement, which causes high false alarm rate in the overall system.

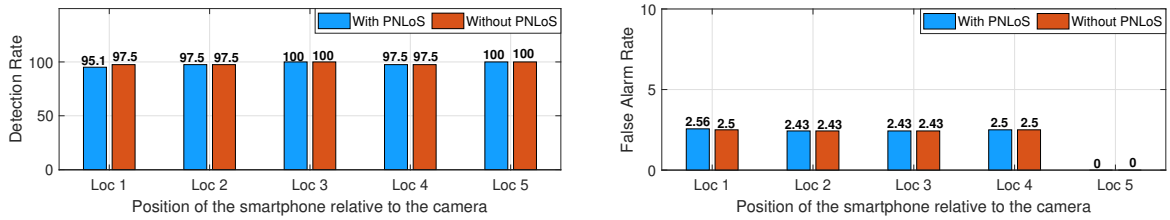
5.2.2 Performance Evaluation. In this experimental setting, we assess the practicability of *CSI:DeSpy* working with the ADL. As the occurrence of sedentary behavior and physical activity in ADL is irregular in nature, therefore, we consider the diverse scenarios with a combination of variable sedentary behavior (static) and physical activity (dynamic) as shown in Fig. 10(a)∼10(d). *CSI:DeSpy* records the minute static cluster formed as a result of the sedentary behavior from the user, and computes a quasi-optimal control limit for the control chart. The control limits are violated by the user, when performing some physical activity, which results in anticipation of spy camera detection. A careful reader might notice that we omitted the fully dynamic scenario as the minimum duration of static is necessary to update online threshold. To get rid of this requirement, we can use a default threshold which will be discussed in the discussion Section 6 in details.

We deliver the performance evaluation of the *CSI:DeSpy* method on the aforementioned ADL cases in Fig. 10(e)∼10(f). As scenario 1 has the lowest static duration, it is more prone to the errors in bitrate caused by network jitters, I-frame transmission, and packet loss. Thus, compared to other scenarios it has a lower detection rate (*i.e.*, 92.3%) and higher false alarm rate (*i.e.*, 9.75%). However, when the duration of static instance increases, the performance improves as depicted in Cases 2, 3 and 4 which have 95%, 97.5%, and 100% detection rate, whilst 5.4%, 2.77%, and 0% false alarm rate, respectively.

5.3 Unobtrusiveness Assessment

5.3.1 Testbed Settings. To make *CSI:DeSpy* easy to use, it is crucial that the user is not obligated to carry the *CSI:DeSpy*-enabled smartphone for camera detection as well as the system should work regardless of its positions in the target area. These are closely related to unobtrusiveness and can be achieved by passive detection. As a matter of fact, The user, the spy camera and the user's smartphone are always co-located. We consider the link quality between camera and smartphone either strong ($RSSI \leq -50$ dBm) or good ($RSSI \leq -60$ dBm), because an individual can use the smartphone for detecting a spy camera in a private rental space to avoid the privacy invasion. It is not possible to have a poor RSSI link (≤ -70 dBm) between two devices communicating in the same room. To measure the feasibility of passive sensing, we place the smartphone at different distances relative to the camera position in the room as shown in Fig. 11.

To distinguish the CSI fluctuation from a person in the target area and another person out of the target area (*i.e.*, ambient noise), we schedule the walking of two persons, namely Person in Line-of-Sight (PLoS) to the camera



(a) Detection rate of *CSI:DeSpy*-enabled smartphone placed at different locations with/without PNLoS.

(b) False alarm rate of *CSI:DeSpy*-enabled smartphone placed at different locations with/without PNLoS.

Fig. 12. Performance comparison of passive camera detection with and without PNLoS

and the Person in None-Line-of-Sight (PNLoS) to the camera. The walking schedule of PLoS was carried out at 15∼30 seconds while the walking of PNLoS was scheduled consistently for the entire 30 seconds. The PLoS completes its motion in a loop by passing from each corner of the room as shown in Fig. 11. Furthermore, the PLoS keeps the same starting point of walking in the considered smartphone placements. In contrast, the PNLoS is moving in the corridor adjacent to the room where the camera is situated. For each smartphone location in the room relative to the camera position, we record 200 data samples, where 100 of them were taken with PNLoS and the remaining 100 data samples were recorded without PNLoS. The duration of each data sample was 30 seconds.

5.3.2 Performance Evaluation. To measure the feasibility of passive camera detection, we evaluate the performance of *CSI:DeSpy* by placing the smartphone at different positions relative to the camera as in Fig. 11. Fig. 12(a) depicts results of *CSI:DeSpy* placed in different locations of the room and the existence of PNLoS. Firstly, blue shaded bars show the results of different positions with PNLoS walking. It is evident that the impact of PNLoS is negligible, owing to the fact that the signal power is reduced by three to five orders of magnitude after traversing the wall twice [1, 15]. Note that concrete wall type is commonly used in building construction due it its strong load-bearing capabilities and good sound immune performance [65]. Concrete walls attenuate the WiFi signal by 9 dB after one time passing through the wall [1]. Thus the signal reflected off the PNLoS is too weak to interfere with the signal of PLoS. The attenuation caused by different wall types is different and has been discussed in Section 6. In this experiment, the wall type obstructing the PNLoS signal is a 14 cm thick concrete wall.

Although the smartphone at *Loc 1* has the strongest signal strength between the smartphone and the camera, yet its performance is slightly lower (95.1% of detection rate) than the other locations (*i.e.*, *Loc 2*∼*5*). The reason for this is the PLoS does not obstruct the link between the smartphone and camera directly, thereby causing a lower variation in CSI. *CSI:DeSpy* placed at other locations *i.e.*, *Loc 2* ∼*5* achieves 97.5%, 100%, 97.5%, and 100% respectively with the presence of PNLoS. As the link between smartphone and camera at *Loc 3* and *Loc 5* is hindered by the PLoS during motion, which results in higher fluctuation in CSI, and therefore a detection rate of 100% is achieved. Secondly, red bars show the results of different positions without PNLoS in Fig. 12(a). We notice that except *Loc 1*, where the detection rate of *CSI:DeSpy* improves by around 2%, all other locations (*i.e.*, *Loc 2*∼*5*) share the same performance with PNLoS.

We also deliver the performance comparison in terms of false alarm rate in Fig. 12(b). It is evident that the PNLoS does not impact *CSI:DeSpy*'s performance. We see this in the case of *Loc 2* and *Loc 3*, where the false alarm rate is 2.43% with and without PNLoS. Even when *CSI:DeSpy*-enabled smartphone placed close to the motion of PNLoS, the overall effect of PNLoS is negligible. For instance, in *Loc 1* and *Loc 4* the false alarm rate is 2.56%, and 2.5 respectively with PNLoS while that without PNLoS is 2.5%. It is because the PNLoS motion has a lower impact on the CSI fluctuation than the PLoS motion. Lastly, as like detection rate, *CSI:DeSpy* has the best performance at *Loc 5* and achieves 0% of false alarm rate. It is because of: (1) the strongest RSSI link between the smartphone and camera, and (2) the link is well obstructed by the motion of PLoS.

5.4 Robustness Assessment

In real-life environments such as hotels or Airbnb, the WLAN may contain abundant data traffic generated by other users in addition to the associated spy camera. Severe data traffic may cause packet loss and jitters in the

camera's bitrate [29]. As a result, the spy camera's bitrate suffers random clumping and dispersion, which may mislead the camera detection process. In the following experiment, we evaluate the impact of traffic load on data transmission of the spy camera and the detection accuracy of *CSI:DeSpy*.

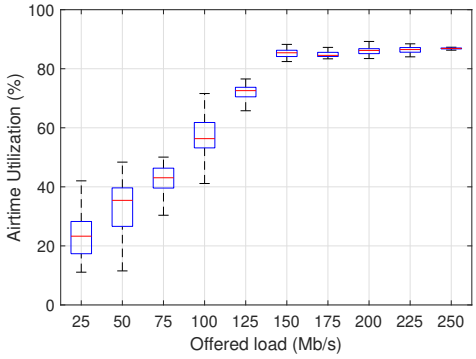
5.4.1 Testbed Settings. To measure the robustness of *CSI:DeSpy* to the network jitters, we designed the following testbed setups.

(1) To study the AP's channel saturation due to the induced load, we deliberately generated a deterministic traffic over five stations (STAs). Firstly, we set the STAs load at a minimum (*i.e.*, 5Mbps on each STA) and then discretely increased its load by 5 Mbps in each trial until the channel became fully-saturated. In addition to the 5 STAs, the wireless camera was also connected to the same network. Each STA was generating UDP traffic using iPerf tool and sending it to the iPerf server.[13] Note that we added the traffic loads to increase airtime utilization and observed its effect on the timing jitters of the camera's transmission. The AP and the iPerf servers were connected to the same Ethernet switch via Ethernet cable, however, the STAs and camera were associated to the AP using channel 11 of 802.11n. For the STAs we used 3 MacBook pro and 2 Windows based laptops, whereas for AP we used TP-Link Archer C7 router (AC1750) operated on 2.4 GHz band (with a bandwidth of 20 MHz).[14] We used Metageek's WiFi spectrum analyzer tool known as Wi-Spy DBx [34], connected to an additional laptop for monitoring the airtime utilization of the considered channel.
(2) To determine the effect of the network jitters with less-loaded and heavily-loaded network traffic on the spy camera's bitrate, we subjected the camera to capture a neutral target zone (i.e., in our setting, the camera lens was facing the wall). This neutral target area was important because we wanted to prevent the variation in bitrate caused by the changes in the target scene. Furthermore, we placed the camera in such a way that the lighting conditions were stable during the sample data collection. Firstly, we recorded 50 data samples of the camera's bitrate without offered load and then we recorded 50 data samples for each offered load. The duration of each recorded sample was 10 seconds.
(3) Finally, we used *CSI:DeSpy*-enabled smartphone both in real-time and offline mode for camera detection. In the real-time mode, we conducted 50 trials for each offered load. For each trial, we deliberately perform an activity containing 5 seconds of sedentary behavior followed by 5 seconds of walking at a moderate pace ($\approx$ 1.3 m/s) and noticing the camera detection result from the *CSI:DeSpy* application (whether the camera was found or not). In the offline mode, we recorded the data samples of the same experiment by connecting the Nexus-5 smartphone via a USB cable with MacBook pro, and collected the CSI and bitrate through Android Debug Bridge utility. These data samples were then used for interpreting the detection performance after applying *CSI:DeSpy* algorithm.

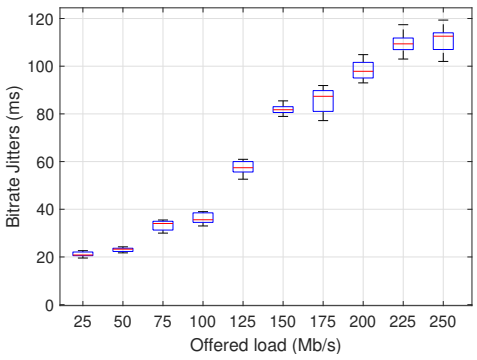
5.4.2 Performance Evaluation. It is evident from Fig. 13(a) that at lower load (e.g., 5 Mbps on each STA – or 25 Mbps aggregated load), the airtime utilization is lower (*i.e.*, around median of 28%). This means that 72% of the time, the channel is free, and each STA has enough time to transmit its data rate. As the load increases to 150 Mbps, we see an increase in airtime utilization. The median airtime utilization is around 35%, 43%, 56%, 72%, and 85%, for 50 Mbps, 75 Mbps, 100 Mbps, 125 Mbps, and 150 Mbps offered loads respectively. Further increase in offered load beyond 150 Mbps does not show a significant increase in airtime utilization. In particular, this non-increasing trend beyond 150 Mbps shows the channel saturation. The median airtime utilization is around 84.5%, 86.2%, 85.5%, and 86.7% for 175 Mbps, 200 Mbps, 225 Mbps, and 250 Mbps offered load respectively.

To measure the jitters in the bitrate of camera connected to the same network, we computed the inter I-frame interval of the reference data samples (I_frame_R) which were collected without offered load. Then we computed the inter I-frame interval of the data samples collected in the presence of offered load (I_frame_L). Finally, the jitter was computed as the deviation of I-frame periodicity of the I_frame_L from the I_frame_R. Fig. 13(b) depicts the impact of network load on the camera's bitrate traffic. Below the channel saturation (*i.e.*, < 150 Mbps offered load), there is a slight increase in camera's bitrate jitter. The median bitrate jitter is 20.82 ms, 23.27 ms, 34.04 ms, 35.60 ms, and 57.42 ms for 25 Mbps, 50 Mbps, 75 Mbps, 100 Mbps, and 125 Mbps respectively. However, when

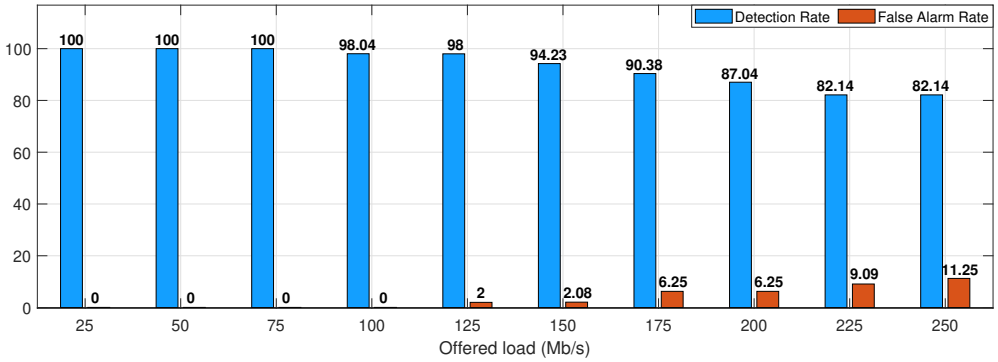
[13] This tool is used for actively measuring the maximum achievable bandwidth on AP [16]
[14] Although the AP's saturation depends on various factors, such as the wireless standard (e.g., ieee 802.11g/n/ac), band (*i.e.*, 2.4 GHz and 5 GHz), and the bandwidth (*i.e.*, 20~80 MHz). However, the overall trend of a channel's saturation with increasing traffic load is the same.



(a) Airtime utilization of the AP channel in response to the offered load.



(b) Impact of load on the camera's bitrate.



(c) Performance of camera detection over various offered loads.

Fig. 13. **(a)** Observation of AP's airtime utilization in response to increase in traffic load. **(b)**Impact of jitters on camera's bitrate. **(c)** *CSI:DeSpy* robustness to the offered load.

the channel becomes saturated then a significant increase in jitter occurs (e.g., the bitrate jitter is increased by around 24% when the offered load is changed from 125 Mbps to 150 Mbps). This trend shows that the STAs in a fully saturated channel (*i.e.*, > 150 Mbps offered load) experience more severe contentions with the camera's stream, which consequently results in higher packet jitters in the transmission of camera videos.

As in the real-life scenario such as hotel or Airbnb, the channel saturation may not be very common. For instance, 5 users watching a high motion 4k video with 60 fps at a time on separate devices can generate an average throughput of around 125 Mbps [32], which is still below the channel saturation of 802.11n. Furthermore, the capacity of the channel further increases if advanced WiFi standards (such as 802.11 ac/ad) are available. Fig. 13(c) illustrates the robustness of *CSI:DeSpy* to the noise caused by the jitters. During the non-saturated channel condition, the detection rate is 100% up to 75 Mbps of offered load (or 44% of the median airtime utilization), and the false alarm rate is 0%. A slight degradation happens when the offered load is increased to 150 Mbps (or 70% of median airtime utilization), where the detection rate is 98.04%, 98% and 94.23% respectively whilst the false alarm rate is 0%, 2%, and 2.08%. It is noteworthy that even in the saturated channel condition, *CSI:DeSpy* is working robustly and its performance has degraded slightly. For instance, in the worst case when the channel is fully saturated (*i.e.*, at 250 Mbps offered load) the detection rate is 82.14% and the false alarm rate is 11.25%. However, for such situation, the sedentary and PA behavior in activities of daily living should be considerably longer (5 seconds each in our evaluation). This is because, if the network jitters corrupts a static behavior of the bitrate for

a short duration, the slightly longer activity (e.g., ≥ 5 seconds) will enable *CSI:DeSpy* to detect the individual's activity from the subsequent frames and accurately predict the availability of a spy camera.

6 FUTURE WORK AND DISCUSSION

This work evaluated *CSI:DeSpy* in diverse and uncontrolled environments. Based on considering the realistic factors, such as room sizes, irreverent people's mobility in the vicinity of target space, and channel occupancy by mixed traffic, etc., we believe that *CSI:DeSpy* is likely to be effective in various indoor environments. In the future, we plan to deploy *CSI:DeSpy* in the wild by considering different real-world target scenarios, e.g., hotel's room, Airbnb, shopping mall's fitting room, and public restrooms, to see the real-time performance of our system whilst exploring more systemic challenges and uncontrolled environmental factors. In the following discussion, we have shed light on the shortcomings of our approach.

Fully dynamic ADL. In the practicability section, we did not consider a fully dynamic ADL scenario, as *CSI:DeSpy* requires a minimum static duration to update the online threshold. However, when there is a fully dynamic case, we handle it by using a default threshold (*i.e.*, α=0.05). This default threshold is updated to a quasi-optimal value when *CSI:DeSpy* opportunistically finds a sedentary behavior (*i.e.*, ≥ 2 *seconds*) from the user.

Spy cameras with micro SDcard support. Some cameras, such as [4] do not use Wi-Fi connectivity and have built-in support for a micro SDcard, that can store around one week of videos in loop recording mode. Such cameras have several limitations: (1) lack of remote control capabilities, (2) consistently operating around the clock for capturing the target footage, and (3) periodically require unplugging memory cards from the camera to download the recorded streamings. These limitations make them less feasible for attackers. Moreover, since such cameras do not generate Wi-Fi traffic, therefore *CSI:DeSpy* cannot detect them.

Handling of buffering or delaying video upload. It is still possible the camera can take advantage of buffering which can store the video clip for a given time. *CSI:DeSpy* cannot detect a camera if it missed the correlation of bitrate distribution and CSI fluctuations. However, the camera needs to consistently upload its captured video to the cloud, which can be accessed later on or stream remotely through an online application. In those cases, *CSI:DeSpy* can detect the spy camera by extending monitoring window size.

Different wall types. As discussed in Section 5, the PNLoS does not impact the signal of PLoS if the wall type is concrete. However, with a different wall type such as plywood, cloth, and glass partition, the impact of PNLoS will be different. For instance, the WiFi signal at 2.4 GHz central frequency has two-way traversal losses of 2.8 dB, 2.58 dB, and 1.2 dB in material type plywood, cloth, and glass respectively [43]. Due to the reduced loss, the interference from PNLoS will be higher. In future work, we aim to tackle the effect of PNLoS with different material partitioning.

Knowledgeable adversary. Recently, the advanced video coding (or H.264) has been adopted by 92% of the developers in the video streaming industry as a standard compression technique [26]. The commodity WiFi cameras also leverage this technology owing to its highest quality video transmission at a much lower bit rate. It is likely that the compression algorithm in WiFi cameras is implemented on the system on chip (SoC), whose interface does not allow any modifications. However, the adversary who is an expert in the network breaching can alter the camera's inter-frame compression technique (e.g., H.264 or MPEG-2/4, etc., compression) to a constant bitrate or intra-frame compression technique. Such an alteration would make *CSI:DeSpy* ineffective for identifying the camera based on its unique traffic pattern.

Monitoring mode requirement. The existing solutions [13, 24, 29, 57] for camera detection require the monitoring mode support to inspect the surrounding wireless traffic. *CSI:DeSpy* also adapts this feature for analyzing the transmission frames of a spy camera without associating with WLANs. Currently, we have adopted this feature by rooting the smartphone to build a proof-of-concept of *CSI:DeSpy*. As an alternative approach, smartphone rooting could be avoided by using the monitoring mode feature of an additional monitoring-mode supported WLAN adaptor. This add-on device could be attached to any smartphone via a USB OTG cable (e.g., Sabrent NT-WGHU [24]).

7 CONCLUSION

We propose *CSI:DeSpy*, a practical and painless approach implemented on the smartphone for detecting a spy camera. The key idea is to exploit the fluctuation patterns of CSI and bitrate features for spy camera detection. *CSI:DeSpy* is equipped with a robust automatic parameter tuning mechanism that can deal with heterogeneous behavioral and environmental settings. We perform extensive evaluations under diverse settings to validate the

robustness of the proposed approach. *CSI:DeSpy* attains an average detection rate of 96.6% for light physical activity in diverse indoor settings, which is only 1.7% lower than vigorous physical activity. Despite different ADL cases, *CSI:DeSpy* achieves an average detection rate of 95.62%, which obviates the necessity of the pre-training. *CSI:DeSpy* works passively and achieves an average detection rate of 98%, and 98.5% with and without ambient noise while placing the smartphone at different positions relative to the camera. Lastly, *CSI:DeSpy* shows a robust performance in the presence of network jitters caused by busy channel.

ACKNOWLEDGMENTS

This research was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education under the grant numbers NRF-2020R1A4A1018774, NRF-2019R1F1A1059898, and NRF-2021M3A9E4080780.

REFERENCES

[1] Fadel Adib and Dina Katabi. 2013. See through walls with WiFi!. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*. 75–86.
[2] Amazon. 2012. *Wireless Camera RF Detector with Built-in Flashlight.* Retrieved Nov 03, 2020 from https://tinyurl.com/y3hn87sz
[3] Amazon. 2019. *Anti Spy RF Signal Detector.* Retrieved Nov 03, 2020 from https://tinyurl.com/y5wjsbgp
[4] Amazon. 2021. *1080P Super Night Vision Hidden Camera.* Retrieved Oct 10, 2021 from https://tinyurl.com/ch75rjs
[5] B. Baird. 2016. *Woman commits suicide after sex tape leaked on internet.* Retrieved May 26, 2021 from https://tinyurl.com/35n2j2xp
[6] BBC. 2020. *How many spycams can Stacey Dooley find in a love motel bedroom?* Retrieved Nov 01, 2021 from https://youtu.be/ggYIsnUgUdU
[7] BBC.com. 2020. *Everyday motion.* Retrieved April 27, 2021 from https://tinyurl.com/2humk9u2
[8] Apurv Bhartia, Yi-Chao Chen, Swati Rallapalli, and Lili Qiu. 2011. Harnessing frequency diversity in wi-fi networks. In *Proceedings of the 17th annual international conference on Mobile computing and networking*. 253–264.
[9] A. Carter. 2015. *Couple Wakes Up in Airbnb to Find Hidden Camera Watching Them.* Retrieved Oct 20, 2020 from https://tinyurl.com/y7ukmt9r
[10] A. Carter. 2019. *Study says guests are finding hidden cameras inside rental properties, hotel rooms.* Retrieved Oct 23, 2020 from https://tinyurl.com/y7ukmt9r
[11] A. Castelan and J. Treanor. 2016. *Hidden cameras found inside a Las Vegas Airbnb rental recording naked people.* Retrieved Oct 23, 2020 from https://tinyurl.com/zrtysgx
[12] Zhenghua Chen, Le Zhang, Chaoyang Jiang, Zhiguang Cao, and Wei Cui. 2019. WiFi CSI Based Passive Human Activity Recognition Using Attention Based BLSTM. *IEEE Transactions on Mobile Computing* 18, 11 (2019), 2714–2724.
[13] Yushi Cheng, Xiaoyu Ji, Tianyang Lu, and Wenyuan Xu. 2019. On detecting hidden wireless cameras: A traffic pattern-based approach. *IEEE Transactions on Mobile Computing* 19, 4 (2019), 907–921.
[14] Emily DeCiccio. 2019. *Hidden cameras: Are you being watched?* Retrieved Nov 5, 2021 from https://tinyurl.com/pjdrapc2
[15] Daniel Dobkin. 2012. *The rf in RFID: uhf RFID in practice.* Newnes.
[16] Jon Dugan. 2016. *iPerf - The ultimate speed test tool for TCP, UDP and SCTP.* Retrieved Dec 23, 2021 from https://github.com/esnet/iperf
[17] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *Kdd*, Vol. 96. 226–231.
[18] Matthew S Gast. 2013. *802.11 ac: a survival guide: Wi-Fi at gigabit and beyond.* " O'Reilly Media, Inc.".
[19] Michael Hahsler, Matthew Piekenbrock, and Derek Doran. 2019. dbscan: Fast density-based clustering with R. *Journal of Statistical Software* 91, 1 (2019), 1–30.
[20] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. 2010. Predictable 802.11 packet delivery from wireless channel measurements. *ACM SIGCOMM Computer Communication Review* 40, 4 (2010), 159–170.
[21] IPX1031. 2019. *Survey: Do Airbnb Guests Trust Their Hosts?* Retrieved Nov 10, 2021 from https://www.ipx1031.com/airbnb-guests-trust-hosts/
[22] J Edward Jackson. 2005. *A user's guide to principal components*. Vol. 587. John Wiley & Sons.
[23] Minqiang Jiang, Xiaoquan Yi, and Nam Ling. 2004. Improved frame-layer rate control for H. 264 using MAD ratio. In *2004 IEEE International Symposium on Circuits and Systems (ISCAS)*, Vol. 3. IEEE, III–813.
[24] Brent Lagesse, Kevin Wu, Jaynie Shorb, and Zealous Zhu. 2018. Detecting spies in iot systems using cyber-physical correlation. In *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 185–190.
[25] M Powell Lawton and Elaine M Brody. 1969. Assessment of older people: self-maintaining and instrumental activities of daily living. *The gerontologist* 9, 3_Part_1 (1969), 179–186.
[26] Stefan Lederer. 2018. *Video Developer Report!* Retrieved Feb 12, 2022 from https://tinyurl.com/2btpmtku
[27] Adrian Lin and Hao Ling. 2007. Doppler and direction-of-arrival (DDOA) radar for multiple-mover sensing. *IEEE transactions on aerospace and electronic systems* 43, 4 (2007), 1496–1509.
[28] Jian Liu, Hongbo Liu, Yingying Chen, Yan Wang, and Chen Wang. 2020. Wireless Sensing for Human Activity: A Survey. *IEEE Communications Surveys Tutorials* 22, 3 (2020), 1629–1645. https://doi.org/10.1109/COMST.2019.2934489
[29] Tian Liu, Ziyu Liu, Jun Huang, Rui Tan, and Zhen Tan. 2018. Detecting wireless spy cameras via stimulating and probing. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. 243–255.
[30] Yongsen Ma, Gang Zhou, and Shuangquan Wang. 2019. WiFi sensing with channel state information: A survey. *ACM Computing Surveys (CSUR)* 52, 3 (2019), 1–36.

[31] Rachna Mehta and N Kumar Aggarwal. 2014. Comparative analysis of median filter and adaptive filter for impulse noise–a review. *International Journal of Computer Applications* 975 (2014), 8887.

[32] Jiayi Meng, Qiang Xu, and Y Charlie Hu. 2021. Proactive {Energy-Aware} Adaptive Video Streaming on Mobile Devices. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. 303–316.

[33] C. Mengxiao. 2017. *Hidden camera found in H&M store's changing room.* Retrieved Oct 21, 2020 from https://tinyurl.com/y7psdcxr

[34] metageek. 2020. *Professional WiFi Tools.* Retrieved Feb 02, 2022 from https://www.metageek.com/enterprise-wi-fi/complete/

[35] BBC News. 2019. *Goo Hara and the trauma of South Korea's spy cam victims.* Retrieved Nov 6, 2021 from https://www.bbc.com/news/world-asia-50582338

[36] Sandra Norman-Eady. 2003. *OLR research report.* Retrieved Nov 10, 2021 from https://tinyurl.com/b8rxjebr

[37] Maya Oppenheim. 2018. *From changing rooms to public toilets: The dark trend of hidden spy cameras filming women.* Retrieved Nov 10, 2021 from https://tinyurl.com/5fdk8u6v

[38] Sameera Palipana, David Rojas, Piyush Agrawal, and Dirk Pesch. 2018. FallDeFi: Ubiquitous fall detection using commodity Wi-Fi devices. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 4 (2018), 1–25.

[39] Kun Qian, Chenshu Wu, Zheng Yang, Yunhao Liu, Fugui He, and Tianzhang Xing. 2018. Enabling contactless detection of moving humans with dynamic speeds using CSI. *ACM Transactions on Embedded Computing Systems (TECS)* 17, 2 (2018), 1–18.

[40] Kun Qian, Chenshu Wu, Zheng Yang, Yunhao Liu, and Kyle Jamieson. 2017. Widar: Decimeter-level passive tracking via velocity monitoring with commodity Wi-Fi. In *Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. 1–10.

[41] Kun Qian, Chenshu Wu, Yi Zhang, Guidong Zhang, Zheng Yang, and Yunhao Liu. 2018. Widar2. 0: Passive human tracking with a single wi-fi link. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. 350–361.

[42] Shobha Sundar Ram, Yang Li, Adrian Lin, and Hao Ling. 2008. Doppler-based detection and tracking of humans in indoor environments. *Journal of the Franklin Institute* 345, 6 (2008), 679–699.

[43] Ahmad Safaai-Jazi, Sedki M Riad, Ali Muqaibel, and Ahmet Bayram. 2002. Ultra-wideband propagation measurements and channel modeling. *Report on Through-the-Wall Propagation and Material Characterization* (2002).

[44] Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. 2011. Finding a" kneedle" in a haystack: Detecting knee points in system behavior. In *2011 31st international conference on distributed computing systems workshops*. IEEE, 166–171.

[45] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. 2017. DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems (TODS)* 42, 3 (2017), 1–21.

[46] Matthias Schulz, Daniel Wegemer, and Matthias Hollick. 2017. Nexmon: The C-based Firmware Patching Framework. https://nexmon.org

[47] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. 2007. Overview of the scalable video coding extension of the H. 264/AVC standard. *IEEE Transactions on circuits and systems for video technology* 17, 9 (2007), 1103–1120.

[48] Brick House Security. 2021. *Understanding the legal ramifications of using hidden cameras.* Retrieved Nov 3, 2021 from https://www.brickhousesecurity.com/hidden-cameras/laws/

[49] Signal. 2021. *The Signal Jammer.* Retrieved July 17, 2020 from https://tinyurl.com/y2hlznnw

[50] N. Smith. 2019. *South Korean woman commits suicide after doctor filmed her using spycam, reports say.* Retrieved Oct 18, 2020 from https://tinyurl.com/y52x98pk

[51] WKRN Web Staff. 2021. *Police: 60 females, mostly minors, recorded on hidden camera found in Franklin cheer studio bathroom.* Retrieved May 26, 2021 from https://tinyurl.com/3jj4m5c

[52] NR Tague. 2004. The Quality Toolbox, ASQ (American Society for Quality).

[53] Global Times. 2021. *Media investigation exposes thousands of candid shots shared and sold via social media platforms.* Retrieved Nov 5, 2021 from https://tinyurl.com/23xcz4hv

[54] A. Vikulov and A. Paramonov. 2020. Practical retrospective of 5-year evolution of the IEEE 802.11 client device capabilities. In *2020 12th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. 296–300. https://doi.org/10.1109/ICUMT51630.2020.9222427

[55] Yuxi Wang, Kaishun Wu, and Lionel M Ni. 2016. Wifall: Device-free fall detection by wireless networks. *IEEE Transactions on Mobile Computing* 16, 2 (2016), 581–594.

[56] Zhang Wanqing. 2021. *China Cracks Down on Hidden Cameras and Clandestine Filming.* Retrieved Nov 11, 2021 from https://tinyurl.com/35jfux7d

[57] Kevin Wu and Brent Lagesse. 2019. Do You See What I See? Detecting Hidden Streaming Cameras Through Similarity of Simultaneous Observation. In *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 1–10.

[58] Wei Xi, Jizhong Zhao, Xiang-Yang Li, Kun Zhao, Shaojie Tang, Xue Liu, and Zhiping Jiang. 2014. Electronic frog eye: Counting crowd using WiFi. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 361–369.

[59] Jiang Xiao, Kaishun Wu, Youwen Yi, Lu Wang, and Lionel M Ni. 2012. FIMD: Fine-grained device-free motion detection. In *2012 IEEE 18th International conference on parallel and distributed systems*. IEEE, 229–235.

[60] Yang Xiao. 2005. IEEE 802.11 n: enhancements for higher throughput in wireless LANs. *IEEE Wireless Communications* 12, 6 (2005), 82–91.

[61] Yaxiong Xie, Jie Xiong, Mo Li, and Kyle Jamieson. 2019. mD-Track: Leveraging multi-dimensionality for passive indoor Wi-Fi tracking. In *The 25th Annual International Conference on Mobile Computing and Networking*. 1–16.

[62] Jie Xiong and Kyle Jamieson. 2013. Arraytrack: A fine-grained indoor location system. In *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation NSDI 13*. 71–84.

[63] Zheng Yang, Zimu Zhou, and Yunhao Liu. 2013. From RSSI to CSI: Indoor localization via channel response. *ACM Computing Surveys (CSUR)* 46, 2 (2013), 1–32.

[64] Xin-Wei Yao, Wan-Liang Wang, Shuang-Hua Yang, Yue-Feng Cen, Xiao-Min Yao, and Tie-Qiang Pan. 2014. Ipb-frame adaptive mapping mechanism for video transmission over IEEE 802.11 e WLANs. *ACM SIGCOMM Computer Communication Review* 44, 2 (2014), 5–12.

[65] Chia-Jen Yu and Jian Kang. 2009. Environmental impact of acoustic materials in residential buildings. *Building and environment* 44, 10 (2009), 2166–2175.
[66] Eric Zeng, Shrirang Mare, and Franziska Roesner. 2017. End User Security and Privacy Concerns with Smart Homes. In *USENIX Symposium on Usable Privacy and Security (SOUPS)*. 65–80.
[67] Xu Zhong and Yu Zhou. 2012. Maintaining wireless communication coverage among multiple mobile robots using fuzzy neural network. In *Proceedings of 2012 IEEE/ASME 8th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*. IEEE, 35–41.
[68] Shilin Zhu, Chi Zhang, and Xinyu Zhang. 2017. Automating visual privacy protection using a smart LED. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*. 329–342.