MicroPrivacy: Detection of Visual Eavesdroppers on Smartphones

***Abstract*—In this paper, we define a 'micro-privacy in macro-place' privacy violation issue and proposed a computer vision (CV) based method to address this problem. Specifically, when a user is using his phone, another person close to him tries to read from his phone screen. We refer to this person as 'a visual eavesdropper.' Our idea is to use the front camera of smartphone as the eye on the back to preserve phone screen privacy in the public place. We present *MicroPrivacy*, the first computer vision based system that integrates image and video processing techniques on smartphones to effectively detect visual eavesdropping. MicroPrivacy deals with the limited camera view issue and the constrained computation power of smartphones. We validate the effectiveness of MicroPrivacy on CV public data and private data that we collected from real world. Experimental results on the datasets demonstrate that MicroPrivacy achieves proper precision and recall in detection of visual eavesdroppers.**

***Index Terms*—Smartphone privacy; Mobile applications; Face detection**

I. Introduction

As the usage of smartphone becomes part of our daily lives and our public surroundings are increasing, people are more eager than ever to create their own personal space using the small and micro-private smartphone screen. However, light-emitting phone screens are naturally attention grabbing, easily seen by people around, so they can easily become the target of privacy violation. For instance, on a subway train, Alice is reading an instant message from her intimate friend, people behind her maybe eavesdrop the screen leading to leakage of the private information. Alice might feel uncomfortable about it whether eavesdropper is innocuous or hostile.

In this paper, we address this micro-privacy issue with smartphone screen in a public space, e.g., subway station, train, bus, office, cafe, classroom, or plaza. We consider the person who watches another person's smartphone screen without the user's awareness as a visual eavesdropper and define this behavior as visual eavesdropping. We found that it is a new privacy issue only proposed recently and in urgent need of a protection or warning strategy to help a phone user to avoid being visually eavesdropped. Hand-shielding gesture is used to prevent password from being seen by others [**?**], but it is obtrusive and it is hard to require a user to protect his screen all the time while he is busy texting or browsing. There, we aim to design an automatic method to solve this privacy problem.

To avoid visual eavesdropping, it is better not to rely on any other additional device other than the user's phone being used. Most modern commodity smartphones have embedded a front camera, we propose to use that to serve as the eye at the user's back and then use a computer vision (CV) based method for detection of visual eavesdropping.

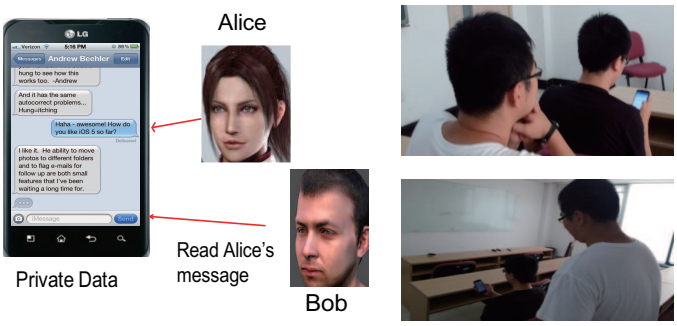


Fig. 1: An example of visual eavesdropping

While the idea of using the CV(Computer Vison) based method with the front camera to detect visual eavesdropping seems straightforward, many interesting challenges arise in practice. First, the camera field of view is limited. Evidently, the necessary prerequisite of detection is that a visual eavesdropper should be captured by the front camera. However, due to the limited camera view, a visual eavesdropper might fall outside of the camera view but still can read the phone screen. Also, a user may read his messages on his phone while walking. The front camera can hardly see the object clearly because of the motion blur. Second, the phone user's face always blocks the camera view. This blocking essentially exacerbates the limited camera view problem. Last, a fast real-time visual detection algorithm typically requires lots of computation resources that a smartphone may not afford to, so it is hard to achieve both high detection accuracy and speed in visual detection.

In this paper, we these challenges as follows. To address the problem of limited field view of front camera, we divide the privacy area into two parts: the area inside the camera view and the area outside of the camera view. For the area inside, we propose a visual detection algorithm to identify visual eavesdroppers; for the area outside, we pre-estimate the movement speed and direction of the visual eavesdropper while he is inside the area, and then predict the probability of the area that visual eavesdroppers may appear. To address the problem of the phone user blocking camera, we replace the area blocked with a consistent background. We calculate the size of the blocked area to predict the probability of the visual eavesdropper may appear in this area. To address the problem of motion blur caused by moving, we adapt the motion deblur

technique [**?**]. To address the limited computation capabilities of smartphones, we include a strategy that removes the blocked area before visual detection.

In brief, we have made the following contributions in this work.

- We present a visual privacy issue on the smartphone screen that easily leaks out private information.
- We propose a new method to use the front camera of a smartphone to act as a human eye at the back.
- We implement MicroPrivacy, the first computer vision based system to detect visual eavesdroppers under the constraints of limited camera view and computational power of smartphones. We integrated several CV based techniques for handling different situation in real world.
- We evaluate MicroPrivacy using both public datasets. Experimental results show that MicroPrivacy achieves high precision and recall in detecting visual eavesdroppers. It also requires 20% less computational power while still guaranteeing the same detection accuracy.
- We conduct real world experiments and results show that MicroPrivacy is practical for preserving people micro-privacy on smartphone screen.

The rest of the paper is structured as follows. We present related work about privacy in Section II and show the difference between them and our approach. We describe the system design considerations of MicroPrivacy in Section III. We then present details of MicroPrivacy, including three parts: privacy definition, visible human detection and invisible human detection in Section IV. In Section V, we report our experimental results. Finally, we make final remarks and discuss the limitations of our approach in Sections VI.

II. Related Work

Privacy is recognized as a dominant concerning factor for a mobile device, in particular for smartphones. Several pieces of work have studied how to protect user privacy by detecting behaviors accessing sensitive data. TapPrints [1] uses motion sensors (i.e, accelerometer and gyroscope) to infer the location of taps on the touch screens and detect smartphone-sensitive data such as password. TaintDroid [2] provides an Android's virtualized execution environment to analyze and determine whether third-party applications leak out user privacy data or not. Similarly, PiOS [3] conducts a static analysis over iOS application intermediate code to discover any possible leaks of sensitive information by third parties. EnCore [4] introduces a privacy preserving approach for mobile social applications, where people encounter strangers within a wide range of communication. In contrast, our work focuses on a totally different privacy on smartphones, micro-privacy on screen. A light-emitting phone screen maybe leak out phone user's privacy information in the public area.

To protect user privacy, it is important to distinguish from a malicious behavior. RiskRanker [5] identifies and assesses potential security risks from untrusted apps to defend against malware. MAdFraud [6] studies advertisement fraud issues that is displayed on app's GUI and is embedded with Ad libraries from the ad provider on mobile phones. Differently, our work proposes to preserve user privacy via issuing an alert/alarm to users, where the front camera acts as an eye on the back. The front camera identifies the scene that a phone user's private information might be leaking out.

Similar to our work, several smartphone applications also adopt the front camera for different purposes. Carsafe [7] protects a driver safety by alerting drivers to dangerous driving conditions and behavior. It uses the front camera to monitor the driver status and the rear camera to detect road conditions. Our work also uses the front camera, but we use it to detect visual eavesdroppers to protect user privacy. ViRi [8] restores the front-view effect at the slanted viewing angle to help users see smartphone screen content conveniently. It uses the front camera to determine whether the screen is needed to slant a certain angle and adds a commercial off-the-shelf fisheye lens to address the limited field view of the front camera. Our work comes from a different perspective. it is common that people carry and use their phones when they are in public areas. It is impractical and inconvenient to require a user to pre-setup a fisheye lens on the front camera to detect visual eavesdroppers. Therefore, our approach does not depend on the addition of an extra hardware.

III. Design Considerations

In this section, we discuss the technical considerations that underpin the design of our approach while the detailed design is presented in Section IV.

A. Definition of Privacy-Impraied Area

Detection of visual eavesdropping on smartphones essentially relies on the real-time processing of the front camera video streams. A proper pre-processing techniques can reduce computation cost and helps estimate the probability of potential privacy leakage. Therefore, we propose a preprocessing strategy that takes into account the following issues.

Limited Camera Field of View. Ideally detecting visual eavesdropping requires a wide camera angle. However, the front camera on modern smartphones usually has a rather limited field of view. Therefore, a visual eavesdropper may fall outside of the camera monitoring area, rendering traditional detection method helpless. Requesting users to add an additional device such as fisheye lens to extend the field of view is impractical since it reduces user mobility and incurs additional cost. Currently there is no good solution to the limited field of view problem yet unless mobile phone manufacturers expand the angle of the camera. To address this issue, we provide a proper definition of the privacy impaired area. With the knowledge of the privacy impaired area and camera monitoring area, we can predict the probability of the privacy leakage outside the camera view by using the historical human detection results. Besides, a proper definition of the privacy impaired area can also be used to determine the detection range and degree of protection, thereby saving the computation cost of the privacy leakage detection.

Blocked Camera Field of View. Considering that a smartphone user always stays in front of the front camera, we can identify a center area blocked by the user's face and body most of the time. Detection of a visual eavesdropper in this blocked area is unnecessary. Generally, the user's face is the largest object in the image. We detect the face in a full image and record the blocked area every few seconds. When the visual eavesdropping detection method tries to detect a human, it skips the blocked area. Further, the blocked camera monitoring area limits the field of view again and significantly affects the performance of the visual eavesdropping detection method. Therefore, the definition of the privacy impaired area needs to consider the user blocked area to help better predict the privacy leakage probability.

B. Detection of Human

Human detection is an important part of visual eavesdropping detection. It tries to detect the human and record its motion trajectory in order to determine whether the person is a visual eavesdropper or not. An intuitive solution for human detection is to detect the face and eyes. Advances in face detection and increasing availability of face detection SDKs and source codes make it practical to explore camera-based mobile apps. However, the accuracy of the face detection method decreases as the background scene gets more complicated. A face detection method typically only uses a single static image and ignores the rich video information. Comparing to the exact matching required by a face detection method, motion detection only needs rough matching by using video to track the object. This not only helps detect the human without face detection, but also makes full use of the video information. We hence combine the face detection method and motion detection method for improving the robustness of the human detection.

IV. System Detailed Design

In this section, we first present the overview of MicroPrivacy, then describe the details of each key component.

Figure 2 shows the proposed MicroPrivacy framework. In Layer 1, six modules are used: smartphone tilt angle, camera view angle, and angle of potential eavesdroppers, motion detection, face detection, and body detection. We assume that three types of sensors are embedded into smartphones to act as input data for Layer 1, i.e, accelerator and gyroscope are used to estimate smartphone tilt angle, and camera sensor is used to capture image frames. These modules provide input data for Layer 2. In Layer 2, there are three key components: (1) *Privacy area Estimation* determines the area where potential visual eavesdropping may occur. (2) *Visual detection inside the camera view* combines face detection and motion detection to detect human visible by the camera. (3) *Prediction outside the camera view* predicts the probability of visual eavesdroppers based on the estimated movement speed and direction of the eavesdropper while he's inside the camera monitoring area. In Layer 3, the visual eavesdropper detection algorithm determines whether to alert the user or not.

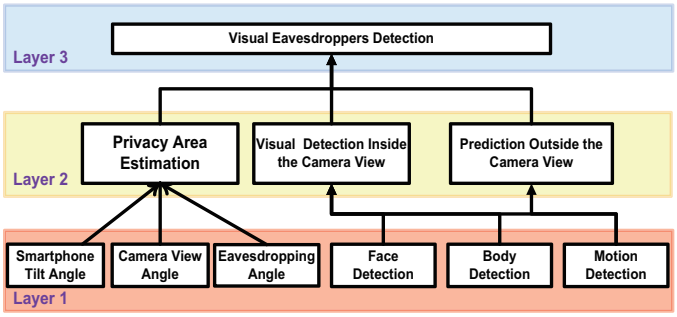


Fig. 2: Architecture of visual eavesdroppers detection

A. Privacy Area Estimation

The size of the privacy area determines the scope the front camera is able to cover. First, we define four types of areas to represent the situation that the visual eavesdropper detection algorithm works differently. We then derive a formula to estimate the size of the privacy area taking into account the visual distance, and the phone tilt angle, and the area blocked by user.

1) Definition of the Privacy Area: As Fig. 3 shows, the red area (i.e., the monitoring area) represents the front camera field of view and the blue area (i.e., the privacy impaired area) shows the range that the visual eavesdroppers can see the information on smartphone screen. The overlap of the two areas is denoted as A_{11}, the part of the impaired area that is out of the monitoring area is denoted as A_{10}, the part of the monitoring area that falls out of the privacy impaired area is denoted as A_{01}, and the area that is outside both monitoring and privacy impaired areas is A_{00}. Typically, A_{01} is very small and A_{00} is the unconcerned area. We further use α_h, α_v to represent the horizontal and vertical angle of the camera view respectively. The angle of potential eavesdropping is defined as β. Therefore, at distance d, the monitoring area (i.e., $(A_{11} + A_{01})_d$) can be expressed as $d^2 \tan^2 \beta$ and the privacy impaired area (i.e., $(A_{11} + A_{10})_d$) is $d^2 \tan \alpha_v \tan \alpha_h$.

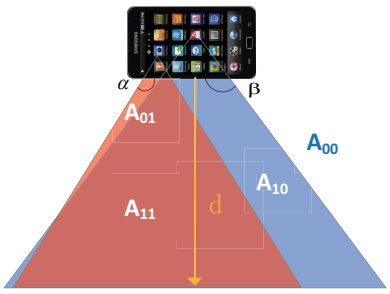


Fig. 3: Definition of different areas

2) Issues for Determining Area Size: The size of privacy impaired area is affected by three factors: visual distance d, phone tilt angle θ, and size of the area blocked by user.

Visual distance. To estimate the privacy area, we should measure the maximum visual distance that a person can see the text on smartphone (Fig. 4). In our daily life, we use the eye chart to measure the visual acuity (i.e., how far a

person can see clearly). We determine the visual distance by considering the phone screen as an eye chart with the international definition of visual acuity [9]. The visual distance is expressed as:

$$d = \frac{acuity * fontsize}{gapsize * tan\theta^*} \quad (1)$$

where $\theta^* = 0.0167$ is a visual angle of 5 arc minutes and $gapsize$ is 5 based on the design of a typical optotype. *Acuity* is the value used in the international definition of visual acuity and we set acuity value to 20/20=1.0, the standard visual range of the human. *fontsize* represents the height of the font on the phone screen. The screen size of the smartphone ranges from 3 inches to 6 inches and font size is no larger than 10.7 mm. This gives the maximum visual distance of 5 meters.

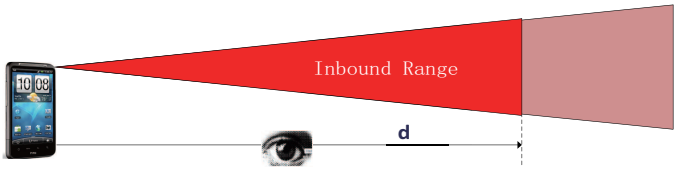


Fig. 4: Estimation of visual distance

Tilt angle of the phone. The phone rotation affects the area size (Fig. 5). With the angle of the phone rotation θ and distance d, the monitoring area is $d^2 \tan^2 \beta \tan\theta$ and privacy impaired area is $d^2 \tan\alpha_w \tan\alpha_h \tan\theta$.

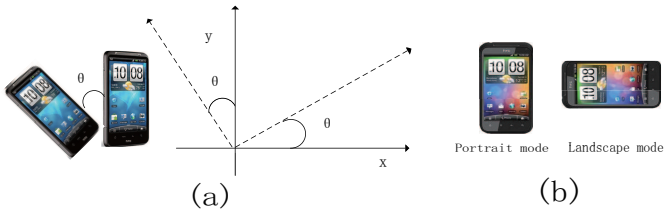


Fig. 5: (a) The angle of the phone rotation (b) The camera view mode

Size of user-blocked area. When the camera view of the smartphone is blocked by the user's face, the blocked area has no use for face detection and affect the performance of the visual eavesdroppers, even though it is part of the monitoring area, so we need to determine the size of the blocked area (Fig. 6).

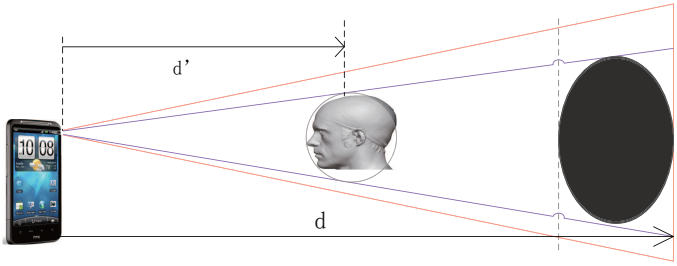


Fig. 6: The user's face blocks the camera view

Assume that the face height in the camera is defined as f_h and the camera focus is expressed as fos (available through Android API), we can derive the distance d' between face and camera by using the pinhole camera model [10]. The pinhole camera model describes the mathematical relationship between the coordinates of a point and its projection onto the image plane of an ideal pinhole camera.

$$d' = \frac{fos * face\_height}{f_h} \quad (2)$$

where $face\_height$ is the actual face height in millimeters, $f_h = \frac{f_h(p) * res_h}{s_h * dpi * 0.3937}$, note that 1 cm $\approx$ 0.3937 inches. $f_h(p)$ is the face height measured in pixels on smartphone, dpi is dots per inch, res_h is screen height resolution and s_h is image height size. They can all be obtained via Android APIs. The range of the face size is limited and the estimation of the face size is difficult. Therefore, we set the face height to the average size of 220mm instead of estimating one's actual face height. Therefore, we can use Eq. 2 to calculate the distance d' between camera and face. The aspect ratio of face width and height in an image is the same as in reality, so with the face width measured in pixels available via Android API, we can compute f_w in a similar way as computing f_h. Therefore,

$$\text{size of blocked area} = f_w * f_h \frac{d}{d'} \quad (3)$$

3) Computation of Privacy Area Size : To calculate size of the monitoring and privacy impaired area, we use the tilt angle of the phone (θ), camera view angle (α_v and α_h), and the angle of a potential visual eavesdropper (β). In addition, we take into account the area blocked by the phone user, which is determined by face height (f_h), face width (f_w), and distance (d'). Based on the discussions previously, we have the size of the monitoring area and privacy impaired area at distance d as follows.

$$\begin{aligned}(A_{11} + A_{01})_d &= d^2 \tan^2\beta \tan\theta - f_w * f_h \frac{d}{d'} \\ (A_{11} + A_{10})_d &= d^2 \tan\alpha_w \tan\alpha_h \tan\theta - f_w * f_h \frac{d}{d'}\end{aligned} \quad (4)$$

B. Detection inside the camera view

When an eavesdropper stays inside the camera view, we use a visual detection algorithm to discover him. Given two image frames, visual human detection aims to detect human in the monitoring area by combining face detection and body detection.

Face detection can help detect human and plays a important role in detecting the visual eavesdropper. Many face detection methods have been proposed [11] and shown their effective using public datasets. The face detection APIs or libraries are available from opencv [12], android [13], and other open source projects [14]. In MicroPrivacy, we integrate different APIs to run our system on different platforms. The basic face detection algorithm is based on the Haar cascade algorithm [11].

Many computer vision algorithms have been designed to track and detect objects such as human body [15] [16].

However, we adopt a motion detection method to track the body. Even with limited computation power of mobile phones, motion detection is still fast and effective for camera-based apps. As Fig. 7 shows, the proposed method uses two frames to get the object and considers the center of the object as the key point for tracking. Specifically, we use frame difference method [17] that subtracts two frame for detecting the moving object. The centroid of the moving object is considered as a key point. Motion detection can generate human trajectories that can be very handy for potential eavesdropping even when the person is invisible in the camera.

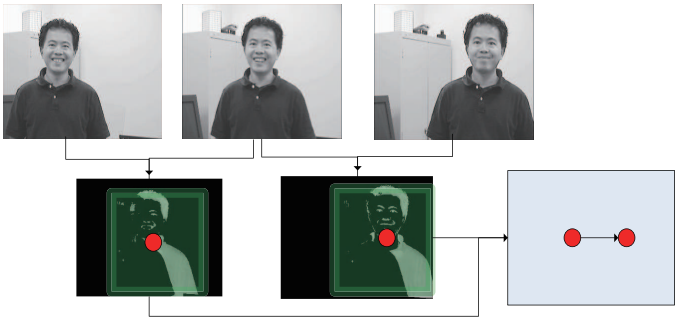


Fig. 7: Motion detection using the Mhyang tracking database

We propose to speed up the visual detection algorithm. Intuitively, we reduce the size of privacy area which reduces the computation of visual detection algorithm. We first remove the user-blocked area from the monitoring area of the front camera. Usually, the smartphone user face takes a large space in the image size. Removing it will speed up visual detection algorithm. We then process the frames with the face-blocked area removed. In this way, face detection can be significantly accelerated. Fig. 8 is an example of this process using the Caltech dataset [18]. The left is the original image that consumes more computation than the right with the face removed from the image.

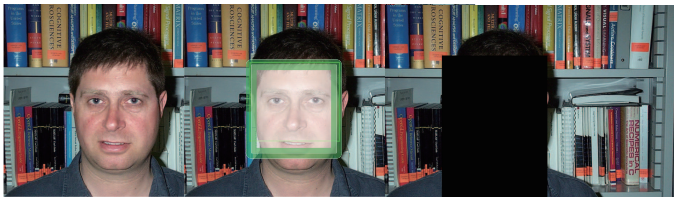


Fig. 8: Remove the blocked area for fast face detection (the original image is from the Caltech dataset)

C. Prediction outside the camera view

A visual eavesdropper may move out of the camera's monitoring area but can still see the phone's screen. To detect this, we estimate the probability of this situation. The basic idea is to use the historical human trajectory to predict its next location. If the predicted location is still in the privacy impaired area, the invisible eavesdropper is considered to be detected.

We use maximum likelihood estimation to build a probability model for this detection:

$$P(t, \tau|M) = P(t|M)P(\tau|M), \tag{5}$$

where t is the duration of the detected face that disappears from the monitoring area and τ is the duration of the detected body that disappears from the monitoring area. M is the parameter to be be estimated for maximizing the likelihood probability $P(M|t, \tau)$. Three aspects of the parameter M can be used for determining the data distribution and they are direction, speed, and area size. We use these three elements to build a probability model to determine M. Instead of obtaining M from training data as a typical maximum likelihood estimation method, we get it from the human detection model; the goal is the same, i.e., to maximize the likelihood probability $P(M|t, \tau)$. Specifically, we use face detection and motion detection to record the human's trajectory. Once the detected human disappears from the camera view, we calculate the motion direction and speed from the records, predict the possible location of the human. We use the area size to estimate the possible time duration that the invisible person can still read the phone screen. The probability of the invisible person eavesdropping is expressed as follows (Fig. 9).

$$P(t|M) = P(\tau|M) = 1 - \frac{v * t}{d^* \cos\gamma(\tan\beta - \tan\alpha) + \epsilon} \tag{6}$$

where v is the motion speed, γ is the motion direction, d^* is the distance between object's face and camera, and ϵ is a smoothing factor to avoid the exception of division by zero. α is set to α_v or α_h while the phone is in vertical ro horizontal.

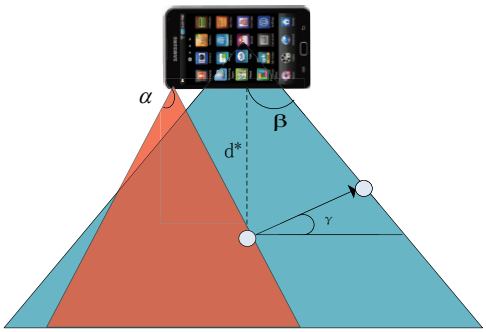


Fig. 9: Prediction of eavesdropper outside camera view

There is another situation for predicting the location of the visual eavesdropper, i.e., the visual eavesdropper moves into the user blocked area. This area is inside the monitoring view but is blocked by phone user's face (Fig. 10).

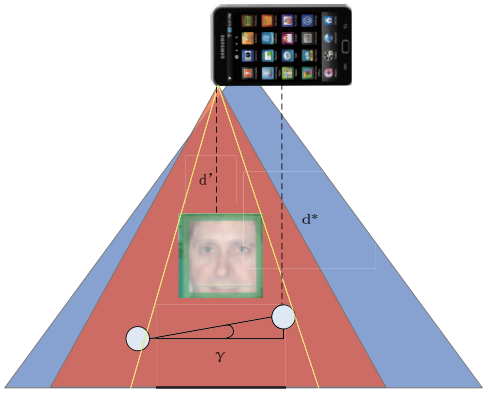


Fig. 10: Human prediction in area blocked by user face

In this case, the probability of the invisible human eavesdropping is expressed as:

$$P(t|M) = P(\tau|M) = 1 - \frac{v * t}{d^* \cos\gamma * (f_w * \frac{d^*}{d'}) + \epsilon} \quad (7)$$

D. Visual Eavesdropper Detection Algorithm

A visual eavesdropper is a person who stays around the user and watches the user's phone screen for a time duration longer than the threshold. Since we use face detection and motion detection to detect human, it is necessary to record the face detection time t_h and the motion detection time τ_f. If we denote $f(\tau_f, t_h)$ as the visual eavesdropper detection result, then

$$f(\tau_f, t_h) = \begin{cases} 1 & \text{if } \tau_f > \tau(s) \text{ or } t_h > t(s) \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

where t_h represents the time of the human body in the privacy impaired area, τ_f represents the time that the person looks at the phone screen, $\tau(s)$ and $t(s)$ are the thresholds to indicate time duration that a normal person stays. A person is considered as a visual eavesdropper if detected face time τ_f is larger than the threshold $\tau(s)$ or detected body time t_h is larger than the threshold t(s). The threshold is sensitive to the environments around the smartphone user. For instance, the time threshold on the bus station should be different from the classroom. According to the experience or the feedbacks of smartphone users, we set the time threshold $\tau(s)$ and t(s) to empirical values.

Algorithm 1 describes the overall algorithm of our visual eavesdropper detection. We first calculate the privacy area using Eq. (4) based on camera angles (α_v, α_h), title angle of the phone (θ), and visual distance (d) (Line 1). If the front camera detectes the face of a visual eavesdropper, we add one to the time variable τ_f that represents the storage time of face detection (Line 13). If the body of the eavesdropper is detected by front camera, the time variable τ_h needs to increment by one, which denotes the storage time of body detection (Line 16). When an eavesdropper falls outside the camera view, we use Eq.(7) to calculate the time variables τ_h and τ_f (Line 21).. Finally our algorithm determines whether there is a visual eavesdropper or not (Line 23).

Algorithm 1 Visual Eavesdroppers Detection

Input: face detection $t(s)$, body detection threshold $\tau(s)$, camera angles (α_v, α_h), title angle of the phone (θ), visual distance (d)

Output: Visual eavesdropper detection result; alerts to user;

1: Calculate privacy area using Eq.(4);
2: Connect to the front camera initially;
3: Initialize face detection time $\tau_f = 0$;
4: Initialize body detection time $t_h = 0$;
5: **repeat**
6: Get the current frame from the front camera;
7: Run face detection algorithm;
8: Run human motion detection algorithm;
9: **if** *detected human in previous frame* **then**
10: Calculate motion direction γ and motion speed v
11: **end if**
12: **if** *detected human face* **then**
13: $\tau_f = \tau_f + 1$; Calculate total face detection time
14: **end if**
15: **if** *tracked motion of human body* **then**
16: $t_h = t_h + 1$; Calculate total motion detection time
17: **end if**
18: **if** *face detection failed* and *motion detection failed* **then**
19: Estimate R_h and R_f based on Eq. 7:
20: $t_h = t_h + P(t|M)$;
21: $\tau_f = \tau_f + P(t|M)$;
22: **end if**
23: **if** $t_h \geq t$(s) or $\tau_f \geq \tau(s)$ **then**
24: Label the human as a visual eavesdropper;
25: $f(\tau_f, t_h) = 1$;
26: $\tau_f = 0$ and $t_h = 0$;
27: Alert the user about the privacy leakage;
28: **end if**
29: **until** User terminates the application

V. Performance Evaluation

In this section, we present the performance evaluation of our proposed approach. We first evaluate privacy area estimation using real world measurement data. For detection of eavesdroppers in the front camera view, we use publicly available datasets to evaluate the accuracy and speed of our face detection and motion detection. For prediction of eavesdroppers outside the camera view, we have conducted controlled experiments to evaluate the accuracy of our prediction. Finally, we run a prototype system in a real-world setting to evaluate the effectiveness of our overall approach to visual eavesdropping detection.

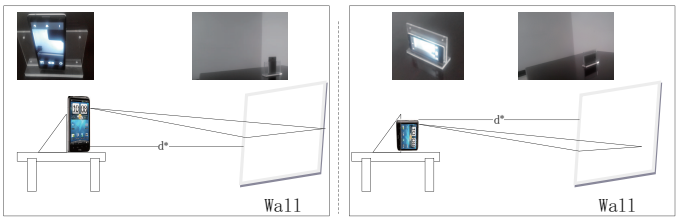
A. Experiments on Privacy Area Estimation

To evaluate our ideas on privacy area estimation, we tie a smartphone to a tablet stand so that we can keep the phone in vertical and horizontal positions on the desk as shown in Fig. 11. The phone faces a wide wall at distance d^*. We use the front camera of the phone to capture images of the wall in order to get the monitoring area, and we use tape to measure

TABLE I: Monitoring range of different smartphones.

Phone make/model	R_h	R_v	α_h	α_v	areaSize
HuaWei honor 3C	1.77m	1.31m	$36.501°$	$28.692°$	$2.318m^2$
Nexus4	1.43m	1.02m	$30.795°$	$23.001°$	$1.456m^2$
Nexus3	1.28m	0.92m	$28.058°$	$21.057°$	$1.177m^2$
lenovo S810t	1.31m	1.03m	$28.369°$	$23.268°$	$1.349m^2$
lenovo p780	1.31m	0.99m	$28.679°$	$22.49°$	$1.297m^2$

the actual size of monitoring area. In this way, we can calculate the camera angle of different smartphones in both vertical and horizontal positions at distance d^*. We then derive the error of our distance estimation that deviates from the actual distance and the error of our size area estimation that deviates from the actual area size.



(a) Phone in Vertical (b) Phone in Horizontal

Fig. 11: Experimental setup for evaluation of area estimation

In these experiments, $d^* = 2.4$m. R_h and R_v are the horizontal length of the monitoring area measured on the wall when the smartphone is put on the desk in horizontal and vertical positions respectively. The view angle of the camera in horizontal is $\alpha_h = arctan(\frac{d^*}{2*R_h})$ and the view angle of the camera in vertical $\alpha_v = \arctan(\frac{d^*}{2*R_v})$. The area size is $R_v * R_h$. Table I shows our experimental results using phones of different vendors.

For privacy area estimation, the phone needs to know the distance d^* between the wall and the camera. As explained in the previous section, we can use detected face size in the image captured by the camera to calculate the distance. To mimic the situation, we ask a person to stand by the wall and use the front camera to take a picture of him. We marked the face black in Fig. 12 to preserve anonymity of the work. We next use a concrete example to explain the whole process. First, we use the front camera to capture the image of the person, then detect the face in the captured image and calculate its face height ($51px$). We then use the Android API *getFocallength()* to get camera focal length (3.5mm) and use $Bitmap.getHeight()$ to get the captured image height 964. In the next, we use *getDisplayMetrics()* to get dpi(dot per inch) values ($320dpi$) and phone screen resolution (1280×720). Finally, we calculate the distance according to the eq.(2) and estimate the area.

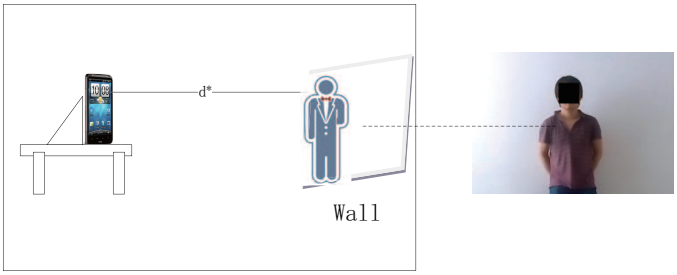


Fig. 12: Experimental setup for evaluation of distance estimation based on face detection

Fig. 13 shows that the estimated distance is close to the actual distance using different phones. The accuracy of distance estimation achieves nearly 82%. The error is caused by deviation in mapping image pixel count to actual distance. We also observe that different phones have similar degree of accuracy. By assuming the real face height as 22cm (i.e., 8.734 inches), we use Eq.(2) to calculate the distance between camera and the face, which is 2.938m, very close to the actual distance of 2.4m. We also calculate the area size to be $3.407m^2$ while the actual size is $2.318m^2$. This area estimation error is acceptable for detecting visual eavesdroppers.

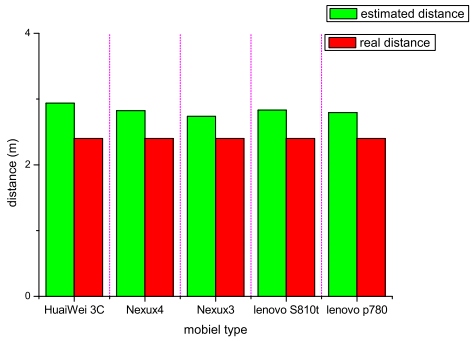


Fig. 13: Estimated and actual distance of different phones

B. Experiments on Face and Motion Detection

We use publicly available face datasets such as BioID [19], Caltech [18], and FDDB [20] datasets to evaluate the performance of our face detection, use the motion detection or tracking dataset such as PETS [21] and part of the Yhyang tracking datasets to evaluate the performance of motion detection.

Our face detection achieves an accuracy of 81.7%, 95.3% and 98.5% using the FDDB, Caltech, and BioID datasets respectively. Fig. 14 is a sample of the face detection results. We observe the decrease of face detection accuracy with the lower clarity of the face in the image either due to presence of multiple faces in the images or low resolution of the image. We also notice that the face detection method always fails while the face does not directly face the camera.

Fig. 14: Face detection results using publicly available datasets

The motion detection method achieves a high accuracy of more than 90% using the classical datasets PETS2001. Fig. 15 is a sample result of motion detection. However, the method detects any moving object, not limited to moving person. Since we are only interested in human, we combine face detection and motion detection.

Fig. 15: The experiments on motion detection in the PETS2001 datasets.

To evaluate the efficiency of our face detection which first removes the user-blocked area, we compare it against the method without blocked area removal using images in the Caltech face datasets. The resolution of each image is 800×600, which is high enough to test the speed of face detection. Fig. 16 shows that our face detection method saves almost 20% of the computation.

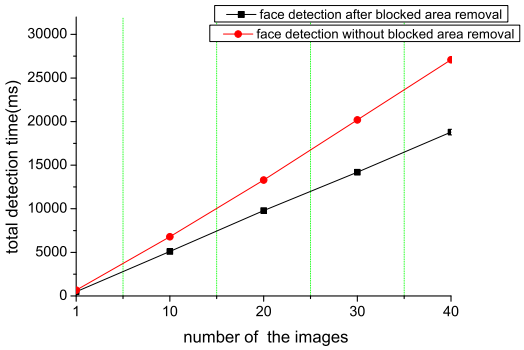


Fig. 16: Face detection time

C. Experiments on Prediction of Visual Eavesdropper Location

For predicting the location of a visual eavesdropper, motion speed and direction is important to estimate the probability of the existing visual eavesdropping. In most situations, motion direction takes a simple form, i.e., moving either left or right. Compared with motion direction, motion speed is more important in predicting the location of the visual eavesdropping. Therefore, we conduct the experiments on estimating motion speed to evaluate the accuracy of our method.

As Fig. 17 shows, we let a person stand by the wall and go from left to right. We record the time from the motion start to the motion end for several times. The width of the wall is 6.5 m and the speed can be calculated by dividing the length by the time. The average time is 7.5 seconds and the speed is 0.86 m/s. For our motion speed estimation, we use face detection method to locate the initial human position and consider the area that is three times of the face size, below the face is the human body. We then use motion detection to track the human that includes the human face and body for getting the trajectories.

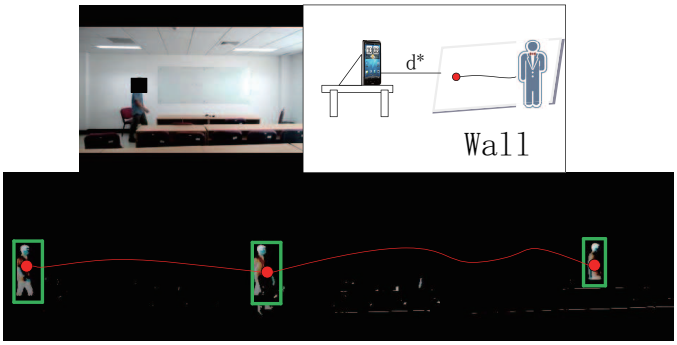


Fig. 17: Experimentsal setup for evaluating motion speed estimation.

Considering that the distance d^* is important for estimating the motion speed, we estimate the motion speed with known distance d^* and also estimate motion speed in a designated area using time. Fig. **??** shows the estimated motion speed is close to the actual speed and achieves nearly 91.3%.

D. Experiments on Overall Visual Eavesdropper Detection

All the experiments discussed previously demonstrate that each component of our proposed visual eavesdropper detection method (i.e., area estimation, visible human detection, and location prediction) performs well using public datasets. To validate that our overall approach works well in practice, we conducted the following experiments. We ask one person to use his phone as usual and another person tries to read his phone screen from behind while walking slowly behind him from left to right. we set the angle of the camera view α_h, α_v to $30°$, $22°$ and set the angle of potential eavesdropping β to $60°$. In this experiment, the time threshold t(s) and $\tau(s)$ are set to be $12s$ and $8s$ respectively. Fig. 18 shows the results of

these experiments. The black area is the area blocked by phone user's face. The left figure shows that the person behind the user is detected by face detection where the small rectangle is the detected human face; the right figure shows that the person is tracked by motion detection although the face is not detected where the large rectangle is the tracked human body. When the person moves out of the monitoring area, we get the probability to visual eavesdropping. When the time of the detected face is more than threshold $\tau(s)$ or the time of the detected body is more than threshold *t*(s), the person behind is considered as a visual eavesdropper. The result of this experiment validates the effectiveness of our proposed visual eavesdropper detection.

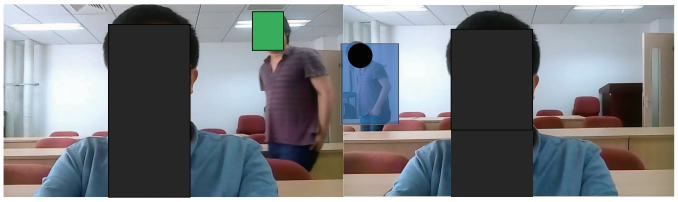


Fig. 18: Experiments on visual eavesdropper detection

VI. Conclusion

In this paper, we define the problem of visual eavesdropping and propose a method to detect it for preserving user privacy on smartphone. We conduct a variety of experiments to evaluate the effectiveness of the proposed method public datasets and real-world experiments. This work can help alert people of potential eavesdropping. In the future, we will complement this work by protecting user privacy via image distortion and masking when potential privacy leakage is detected.

Since MicroPrivacy is a CV based approach, its performance is limited by the performance of CV techniques. When there are multiple people walking around the phone user or the background has significant light variation, the accuracy of face detection drops. Nevertheless, MicroPrivacy can work well in many scenarios, especially in a spacious place or room with few people. Our work shows that CV based methods have a great potential especially in the future when CV techniques become more robust and efficient. Further, there is a special scenario that a person may be looking in the direction of the smart-phone, he may not be looking at the smartphone. In this case, MicroPrivacy takes a conservative approach by alerting the phone user anyway, without trying to differentiate between the two cases. High energy consumption is another problem in CV based approaches on smartphones. To address this problem, a few pieces of work have already started investigating how to reduce the computation [**?**]. We believe these efforts along with the advances in smartphone hardware will make it acceptable to have CV based apps on smartphones.

References

[1] E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. R. Choudhury, "Tapprints: your finger taps have fingerprints," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 2012, pp. 323–336.

[2] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "Taintdroid: an information flow tracking system for real-time privacy monitoring on smartphones," *Communications of the ACM*, vol. 57, no. 3, pp. 99–106, 2014.

[3] M. Egele, C. Kruegel, E. Kirda, and G. Vigna, "Pios: Detecting privacy leaks in ios applications." in *NDSS*, 2011.

[4] P. Aditya, V. Erdélyi, M. Lentz, E. Shi, B. Bhattacharjee, and P. Druschel, "Encore: private, context-based communication for mobile social apps," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*. ACM, 2014, pp. 135–148.

[5] M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, "Riskranker: scalable and accurate zero-day android malware detection," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 2012, pp. 281–294.

[6] J. Crussell, R. Stevens, and H. Chen, "Madfraud: investigating ad fraud in android applications," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*. ACM, 2014, pp. 123–134.

[7] C.-W. You, N. D. Lane, F. Chen, R. Wang, Z. Chen, T. J. Bao, M. Montes-de Oca, Y. Cheng, M. Lin, L. Torresani *et al.*, "Carsafe app: alerting drowsy and distracted drivers using dual cameras on smartphones," in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. ACM, 2013, pp. 13–26.

[8] P. Hu, G. Shen, L. Li, and D. Lu, "Viri: view it right," in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. ACM, 2013, pp. 277–290.

[9] "Roger n. clark. notes on the resolution and other details of the human eye." http://www.clarkvision.com/articles/eye-resolution.html.

[10] "Pinhole model," http://en.wikipedia.org/wiki/Pinhole_camera_model.

[11] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

[12] "Opencvface," http://opencv.willowgarage.com/wiki/FaceDetection.

[13] "Android face detetor," http://developer.android.com/reference/android/media/FaceDetector.html.

[14] "Microsoft face sdk." http://research.microsoft.com/en-us/projects/facesdk/.

[15] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[16] X. Wang, T. X. Han, and S. Yan, "An hog-lbp human detector with partial occlusion handling," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 32–39.

[17] D. Lee and Y. Park, "Vision-based remote control system by motion detection and open finger counting," *Consumer Electronics, IEEE Transactions on*, vol. 55, no. 4, pp. 2308–2313, 2009.

[18] O. Jesorsky, K. J. Kirchberg, and R. W. Frischholz, "Robust face detection using the hausdorff distance," in *Audio-and video-based biometric person authentication*. Springer, 2001, pp. 90–95.

[19] "Bioid face database," http://www.bioid.com/.

[20] V. Jain and E. G. Learned-Miller, "Fddb: A benchmark for face detection in unconstrained settings," *UMass Amherst Technical Report*, 2010.

[21] "Pets 2001 dataset." ftp://ftp.pets.rdg.ac.uk/pub/PETS2001.