Sherlock: Automated Hidden Camera Detection with Shutter Speed Adaptation

Sooyoung Park,[*] Hyeongheon Cha,[*] Sriram Sami,[†] Jun Han,[‡], and Sung-Ju Lee[*]
[*]KAIST, [†]National University of Singapore, [‡]Yonsei University

ABSTRACT

With the rapid advancements in surveillance camera technology, there has been a surge in crimes involving illegal filming using hidden cameras. However, current solutions necessitate slow manual scanning processes that require thousands of workers within a city to ensure coverage. To address this challenge, we introduce *Sherlock*, a fully automated hidden camera detection system that utilizes a drone in combination with a flickering flashlight. To expedite the scanning process and accommodate the irregular movements of the drone, *Sherlock* leverages the rolling shutter effect to capture images when the flashlight is turned on and off within a single frame. This methodology enables *Sherlock* to efficiently identify reflective objects and detect hidden camera lenses among them. To demonstrate the feasibility of *Sherlock*, we present a proof-of-concept evaluation by deploying a drone in various environments to detect and locate hidden cameras.

1 INTRODUCTION

Rapid growth in surveillance camera technology has led to a proliferation of the security camera market. However, such growth also exacerbates privacy invasion using hidden cameras. To address these concerns, Airbnb, for instance, has implemented a policy that strictly prohibits using any concealed monitoring devices in its rental apartments [1]. However, enforcing compliance with such rules is practically challenging due to the difficulty in detecting hidden cameras. Hidden cameras can be easily purchased from online retailers [2, 3, 7] and installed in any private space In South Korea, there were over 30,000 reported cases of illegal filming using hidden cameras in five years, with a significant number of these incidents occurring in public restrooms [9].

Conventional camera detectors assist users in visually locating camera lenses or identifying their distinct wireless signals [16]. However, this method requires a slow, meticulous, and manual sweep of the entire area, requiring significant human resources with low guarantee of success. For example, South Korea hired over 8,000 workers in Seoul to regularly inspect public restrooms. This is clearly not scalable.

Recent approaches identify hidden cameras by scrutinizing wireless signals. However, such techniques cannot be applied to non-wireless cameras. An alternative method pinpoints the lens of hidden cameras directly. For instance,

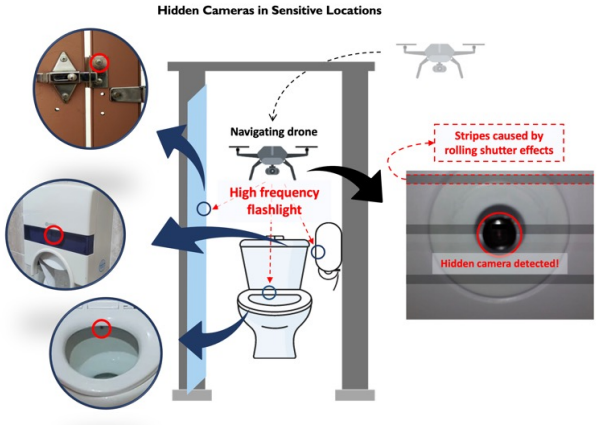

Figure 1: Figure illustrates the system overview of Sherlock in a public restroom scenario. A drone equipped with a high-frequency flashlight and a camera maneuvers within the toilet stall to detect hidden cameras.

LAPD [17] utilizes Time-of-Flight (ToF) sensors to illuminate a designated area and identify camera lenses via the *retro-reflective* quality of these lenses. However, this approach requires users to scan each suspicious object separately.

Hence, we need a novel solution to detect hidden cameras in public facilities, particularly in areas like public restrooms. The solution require minimal human effort, and precisely identify hidden cameras in unspecified areas without the need to pre-define suspicious objects. Furthermore, the solution should be cost-effective to ensure broad deployment.

To address the aforementioned requirements, we present *Sherlock*, a fully automated hidden camera detection system with an RGB camera attached to a drone, as depicted in Figure 1. By automating the scanning system using a drone, *Sherlock* scans a wide area without manual effort. *Sherlock*'s primary goal is to accurately detect the presence of hidden cameras within a minimal scanning duration using widely available commodity devices. To achieve this, *Sherlock* uses the camera lens' retro-reflective property. By flashing a light on a reflective camera lens' surface, *Sherlock*'s RGB camera can locate the unnaturally reflective hidden camera lens.

We face several technical challenges to realize our method: *(i)* Ambient light and other background reflective objects make distinguishing which reflections are hidden cameras difficult. *(ii)* Using flickering light to detect reflective objects

requires us to capture at least two static images when the light is on and off, resulting in a long scanning time. *(iii)* The drone's irregular movements cause a warped image, making detection more challenging.

To tackle these issues, we propose a technique to capture multiple instances of an object in a single frame with a flickering flashlight. The drone illuminates a target space with the flashlight, exploiting the rolling-shutter effects of an RGB camera to capture multiple instances of an object in a single frame. The sequential readout nature of the CMOS sensor array causes these rolling shutter effects; each row of the sensor array captures at a different time, resulting in a geometric distortion of the image [12]. While this distortion is often undesirable, *Sherlock* exploits these characteristics to capture multiple instances of the target area (i.e., when the flashlight is turned on and off).

We evaluate *Sherlock* in different environments, including varying distances, speeds, and ambient light conditions. To test the feasibility and generalizability of *Sherlock*, we use a cheap educational-purpose drone to detect various camera modules.

We make the following contributions: *(i)* We present a fully automated hidden camera detection mechanism; *(ii)* we design an exposure adaptation methodology that enables our system to be installed on various drones; and *(iii)* we evaluated the real-time performance of *Sherlock* in various environments.

2 BACKGROUND

We now present relevant background and a feasibility study.

2.1 Rolling-shutter Effects

Automating lens detection using a camera encounters a significant drawback of slow scanning speed. Unlike other lens detection methods, *Sherlock*'s automated lens detection approach necessitates capturing multiple distinct images with the flashlight turned on and off. To scan a designated area, the drone must maneuver through the room, capturing images at different moments when the flash is activated and deactivated. This time-consuming process delays obtaining results and increases the drone's power consumption, consequently limiting the scanning area. To address this problem, *Sherlock* capitalizes on the rolling-shutter effect of the camera [12], enabling the capture of multiple functional image fragments within a single image frame.

Another significant issue is dynamic range; while human eyes perceive brightness non-linearly, allowing for a wide range of brightness, cameras often struggle with overexposure and underexposure when dealing with scenes containing elements of varying intensity. While industrial cameras employ global shutters to simulate human eye-like perception, most consumer digital cameras, pinhole cameras, and smartphones employ the rolling shutter sampling mechanism [10], primarily responsible for their sensitivity to high-frequency flickering.

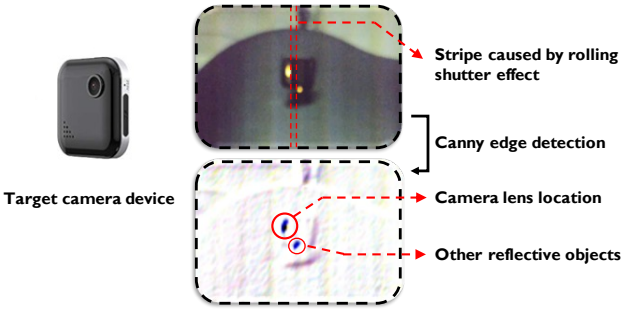

Figure 2: Figure illustrates the detected camera lens location as well as other reflective objects using canny edge detection with rolling shutter effects.

2.2 Feasibility Study

We conduct a feasibility study utilizing a commercially available hidden camera to validate our hypothesis of using the rolling-shutter effect for hidden camera detection. Figure 2 illustrates the captured frame from a CMOS camera. Due to the rolling-shutter effect, the image exhibits multiple stripe patterns: half dark and half bright. The dark stripes correspond to moments when the flashlight is turned off, while the bright stripes represent moments when the flashlight is turned on. By comparing these two images, it becomes possible to identify several areas that reflect light, which potentially include the hidden camera lens. However, differentiating between retro-reflections from the hidden camera and other high-intensity reflections poses a challenge. Additionally, depending on the widths of the stripes, it is conceivable that a single dark or bright stripe may cover the entire lens area. We describe how we overcome these challenges in Section 3.

3 SYSTEM DESIGN

We now present details on *Sherlock*'s design.

3.1 Design Overview

Our design goal is to maximize the hidden camera detection accuracy and generalizability to a wide range of low-cost drones. The major characteristics of low-cost drones are their relatively short flight time, and irregular movements during flights, causing inconsistency of the target object locations in consecutive captured frames. To solve this, our design aims to minimize the required captured image frames to complete

hidden camera inspection, increase scanning speed but also improving detection accuracy.

As depicted in Figure 3, the overall procedure is divided into two phases: exposure adaptation and image data analysis. The main goal of the exposure adaptation phase is to adjust the drone's camera settings to maximize rolling shutter effects, minimizing the number of frames necessary for analysis. After the exposure adaptation phase, the drone is capable of providing information-rich images to the image data analysis phase. During this phase, the provided image is disassembled into two images (bright and dark). These two images are then compared to detect reflective objects, namely the *candidates for hidden camera lenses*. These candidates are fed to our Siamese network image classifier for final detection of hidden camera lenses.

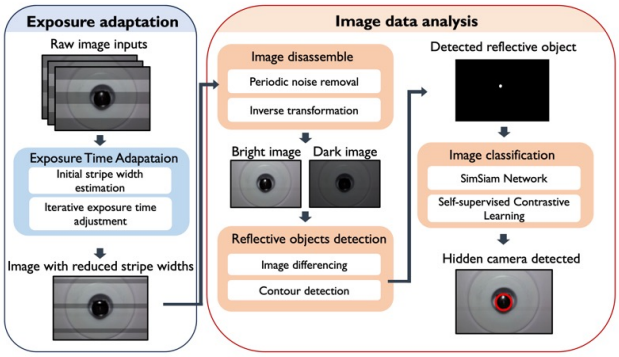

Figure 3: *Sherlock*'s system overview consists of two phases: *Exposure adaptation phase* and *Image data analysis phase*. *Exposure adaptation phase* calibrates the drone's camera to control the stripe widths caused by rolling shutter effects, after which *Image data analysis phase* inspects the images to detect hidden camera lens.

3.2 Exposure Adaptation

The exposure adaptation module takes images captured by the drone, analyzes the stripes caused by the rolling shutter effect and adapts the exposure accordingly.

For *Sherlock* to successfully capture multiple instances of the hidden camera lens, the stripes caused by the rolling shutter effect must not cover the entire lens area. For instance, if the stripes are too wide, the captured frame has the entire lens area reflecting the flashlight (i.e., bright instance). If too narrow, the camera fails to capture enough information when the flickering flashlight is turned off. Therefore, *Sherlock* requires a methodology that can adjust the stripe widths and increase the chance of capturing multiple instants of the lens area in a single frame.

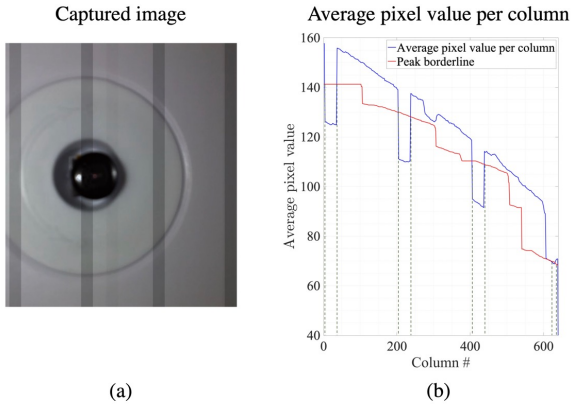

Figure 4: Figure illustrates (a) the captured image and (b) average pixel value for each image column, used for initial stripe width estimation.

We propose and apply *exposure time adaptation*. The widths of the stripes caused by the rolling shutter effects are impacted by the following components: the sampling rate and exposure time of the camera's image sensor and the frequency of the flickering flashlight as shown in the equation below [21].

$$W = \left|(t_e \mod t_{off}) - t_{on}\right| / t_s \quad (1)$$

where W represents the width of each stripe, t_e and t_s represent exposure time and image sensor's sampling rate, and t_{off} and t_{on} refer to on/off duration of the flashlight.

The sampling rate of the image sensor and the frequency of the flickering flashlight are fixed based on the camera and flashlight hardware, respectively. However, most cameras offer functions to adjust the exposure time. *Sherlock* utilizes this function to adjust the camera setting and control the widths of the stripes if needed.

For our prototype, we empirically set the target widths of the stripes as (image width in pixels) / 20. To ensure the broader applicability of our solution, we assume that the drone has no knowledge of the flickering flashlight's light duty cycle (i.e., frequency). As a result, the only known variables are t_e and t_s. To address this, *Sherlock* estimates the initial value of W by analyzing the raw image data captured.

Sherlock uses image processing to estimate the initial stripe width and leverage the periodically occurring nature of stripes caused by rolling shutter effects. As shown in the Figure 4, we first calculate the average pixel value for each column of the image. From this, we calculate the moving minimum and moving maximum (*movmin* and *movmax*) with moving window size of 200 pixels. We then compute the *peak borderline* (where *peak borderline* = $(movmin + movmax)/2$) that determines which pixel columns belong to the stripes caused by the rolling shutter effects.

As illustrated in Figure 4, the *peak borderline* does not guarantee perfect estimations of the stripe locations. However, because all stripes have the same widths and are equally spaced out, we can use these features to correctly identify the stripes. Moreover, to improve the estimation accuracy, *Sherlock* uses multiple image frames to improve its estimation of the initial stripe width. For our prototype, *Sherlock* takes 300 image frames for the estimation.

Then, using the equation above, we can also estimate the flickering flashlight's frequency and compute the required exposure time to adjust the widths of the stripes. The entire exposure adaptation phase acts as an one-time calibration for the *Sherlock*, after which only the image analysis phase is repeated for hidden camera detection.

3.3 Image Data Analysis

3.3.1 Image Disassembler. The primary goal of the "Image Disassembler" is to generate two distinct images from a single source image, which are subsequently compared to identify reflective objects. These two images are referred to as the *bright image* and the *dark image*. The bright image comprises the moments when the flickering flashlight is turned on, while the dark image comprises the moments when the flickering flashlight is turned off, (i.e., stripes caused by the rolling shutter effects).

Leveraging the periodically occurring nature of the stripes, *Sherlock* uses periodic noise removal techniques to separate these two images. A notch filter based on the fuzzy transform is proposed to smooth the spectrum and separate the original image from the periodic noise components. The filter targets and suppresses the identified noise peaks while preserving the underlying image details.

These bright image and dark images are compared via image differencing and contour detection to detect objects that reflect the flickering flashlight. With the pixel positions of the detected objects, *Sherlock* creates candidates for possible hidden camera lens.

3.3.2 Image Classification. In the previous stages, exposure adaptation and image disassembler play major roles in reducing the number of candidates, image classification in *Sherlock* plays relatively minor role: verifying whether the candidate is indeed a hidden camera lens. *Sherlock* employs the SimSiam [4] method, an unsupervised pre-training approach for deep neural networks, a popular method in identifying a specified object from images. Due to lack of camera lens open datasets, we collected over 100 images of hidden camera lens. We incorporated self-supervised contrastive learning with SimSiam model to compensate for our small dataset.

Our trained model takes crops of images scaled to 100x100, each representing a candidate for hidden camera lens, and outputs the classification results, alerting the user if the camera lens is detected.

4 PRELIMINARY EVALUATION

We now present *Sherlock*'s preliminary evaluation results.

4.1 Experimental Setup

Apparatus. We use a Tello drone, a mini drone equipped with an HD camera. For the flickering flashlight, we attached an external 80Hz flickering LED device [15], a commercially available strobe light device designed for the drone's anti-collision system at night. We use the OV2640 mini compact camera module for the hidden cameras.

Experimental procedure. Figure 5 illustrates our experimental procedures for evaluating *Sherlock*. Through this experiment, we aim to measure both accuracy and scanning duration to detect a hidden camera.

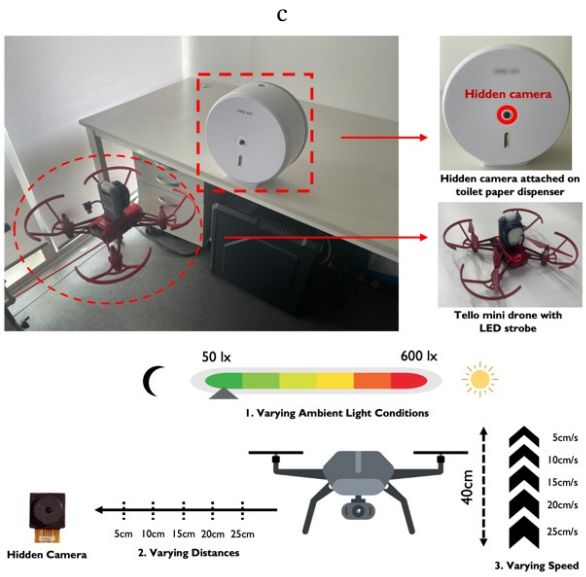

Figure 5: Figure illustrates Tello drone with a LED strobe attached, OV2640 camera module hidden in a toilet paper dispenser, and overview of various environments for *Sherlock*'s evaluations.

As discussed in Section 1, public restrooms are among the most frequently targeted locations for hidden camera installations, and the toilet paper dispenser, given the limited objects within such facilities, is a commonly chosen location for this illicit purpose. For this reason, for our evaluation, we installed the hidden camera in the center of the toilet paper dispenser to simulate a public restroom scenario.

We measure the following metrics for our evaluation:

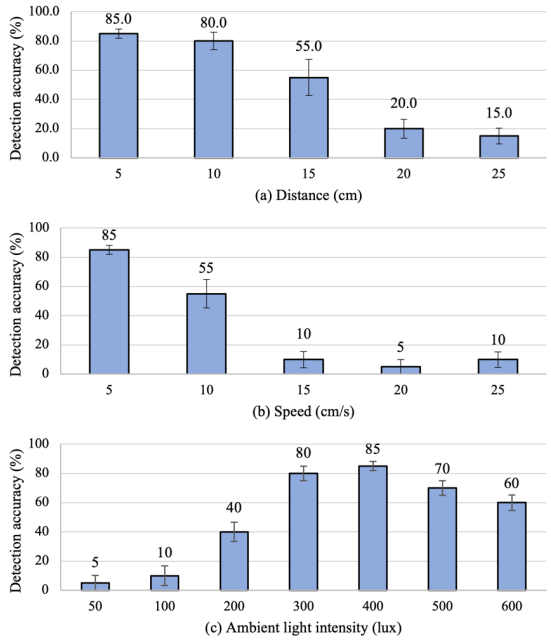

Figure 6: Figure depicts the detection accuracy of *Sherlock* with (a) varying distances, (b) varying speeds, and (c) varying ambient light intensities.

(1) **Detection accuracy:** Given a fixed time or the number of sweeps, the detection accuracy is the percentage of successful detections of the hidden camera lens.
(2) **False positive Rate:** Given a fixed time or the number of sweeps, the false positive rate is the percentage of *Sherlock*'s incorrect classifications of reflective objects that are not hidden camera lenses.
(3) **Scanning time:** Scanning time refers to the time required for a drone to successfully detect the target hidden camera.

4.2 Preliminary Results

To evaluate the overall performance, we set a scanning time limit of 2 minutes for each test run, and recorded the number of successful detection of the hidden camera lens. As a result, *Sherlock* achieves a detection accuracy of **85%**, meaning it successfully detected the presence of hidden cameras in the majority of cases. Moreover, the false positive rate was measured at a modest **5%**, indicating a relatively low rate of mistakenly identifying non-existent hidden lenses.

In addition to assessing the detection and false positive rates, we conduct an experiment to measure the scanning time required for *Sherlock* to identify the target hidden lens. The results show that *Sherlock* took approximately **121 seconds** to successfully detect the camera lens.

4.2.1 Controlled experiments. We evaluate the detection accuracy of *Sherlock* in various environments to evaluate their impacts and find ideal settings. We vary the distance between the drone and the hidden camera, speed of the drone and the ambient light intensity. The default setup is 5 cm distance, 5 cm/s speed and 400 lx ambient light intensity.

Our results indicate that the detection accuracy decreases as distance and speed increase. At a longer distance, the reflection from the flickering flashlight diminishes, and with higher speed, the quality of the capture image degrades. For the ambient light intensity, the accuracy falls when the intensity is either too high or too low. If the ambient light intensity is too high, the illuminating effects of flickering flashlight are weakened. In a dark environment, on the other hand, the accuracy falls drastically because *Sherlock* fails to capture a usable image when the flashlight is turned off.

5 DISCUSSION

We present relevant discussion points in this section.

5.1 Light Intensity and Blinking Frequency

Sherlock uses light and dark contrasts captured by a flashlight to find flickering areas (i.e., potential camera lenses). Higher flashlight brightness and blinking frequency improve lens detection. *Sherlock* uses a commercial flashlight with limitations but mitigates them with exposure time adaptation. Using a more powerful flashlight would enhance detection at greater distances. Moreover, camera specifications, such as resolution and frames per second, affect image clarity. Replacing the low-cost camera with a higher-performing one would also boost performance.

5.2 Coverage Range

Sherlock's optical scanning inspects areas sequentially, increasing the scan time as a function of area. It also relies on the reflective lens property, limiting the effective detection angle and range [17]. *Sherlock* aims for fully automated drone scanning, using multiple drones to divide large areas, making scanning manageable. Enhancing the flashlight intensity and reducing shutter intervals with external hardware can shorten scanning times and improve image quality.

6 RELATED WORK

WiFi. Many related works attempt to detect and localize hidden cameras that transmit video information *wirelessly* via two methods: analyzing wireless traffic patterns [5, 6, 19] and correlating traffic/channel state information to external stimuli [5, 8, 13, 18]. However, none can locate non-wireless hidden cameras that store video locally.

Electromagnetic Interference. A recent work detects *actively-recording* hidden cameras via the electromagnetic leakage

of their clock signals [14]. However, this technique only approximates the cameras' locations. Another work relies on the non-linear interference of such electronics on millimeter wave (mmWave) radar signals [11]. However, this is slow as it cannot differentiate hidden cameras from other devices.

Thermal. Thermal imaging could identify hidden cameras as they emit some heat [20, 22]. However, complex thermal environments (e.g., a camera hidden within another heat source, such as a WiFi router) could bring challenges, and it is unclear how generalizable it is in various environments.

Optical. Optical methods such as *Sherlock* prove the existence of cameras by detecting retro-reflections from lenses. A recent work, LAPD [17], uses commodity smartphones but works only with smartphones with time-of-flight (ToF) sensors, and its scanning process is tedious and slow. Sherlock can scan comprehensively and only requires an inexpensive RGB camera and flashlight as the core sensing modality.

7 CONCLUSION

We present *Sherlock*, a fully automated hidden camera detection system using a drone, that is able to detect and locate hidden camera lenses. *Sherlock* utilizes a flickering flashlight and rolling shutter effects to capture the image of the target area with the light on and off within a single frame. This approach enables *Sherlock* to identify reflective objects and categorize them as hidden camera lenses within a shorter scanning period compared to the state-of-the-art solutions. In our preliminary evaluation, we demonstrate that *Sherlock* achieves a reliable detection rate for hidden camera lenses.

REFERENCES

[1] Airbnb. [n. d.]. Use of cameras and recording devices - airbnb help centre. https://www.airbnb.co.uk/help/article/3061?_set_bev_on_new_domain=1697245969_ZWU3MDk4NTUxZGUw

[2] Amazon. 2019. *MHDYT Mini Spy Camera Wireless Hidden.* https://www.amazon.com/Spy-Camera-Wireless-Portable-Covert-Home-Detection-Surveillance/dp/B07X7D3BC9/.

[3] Amazon. 2022. *Hidden Camera Pen 1080P HD 2.5Hrs with 32GB SD Card.* https://www.amazon.com/Hidden-Camera-1080P-2-5Hrs-32GB/dp/B0B9G2FTX1.

[4] Xinlei Chen and Kaiming He. 2021. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 15750–15758.

[5] Yushi Cheng, Xiaoyu Ji, Tianyang Lu, and Wenyuan Xu. 2018. DeWiCam: Detecting Hidden Wireless Cameras via Smartphones. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security (ASIACCS '18)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3196494.3196509

[6] Y. Cheng, Xiaoyu Ji, Tianyang Lu, and W. Xu. 2020. On Detecting Hidden Wireless Cameras: A Traffic Pattern-based Approach. *IEEE Transactions on Mobile Computing* 19 (2020), 907–921.

[7] B & H. 2021. *Hidden & Spy Cameras.* https://www.bhphotovideo.com/c/buy/Hidden-Cameras/ci/18682/N/4045021092.

[8] Yan He, Qiuye He, Song Fang, and Yao Liu. 2021. MotionCompass: Pinpointing Wireless Camera via Motion-Activated Traffic. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, Virtual Event Wisconsin, 215–227. https://doi.org/10.1145/3458864.3467683

[9] Bicker Laura. 2021. "I was humiliated": The continuing trauma of South Korea's spy cam victims. *BBC News* (Jun 2021).

[10] Mingyang Li, Byung Hyung Kim, and Anastasios I Mourikis. 2013. Real-time motion tracking on a cellphone using inertial sensing and a rolling-shutter camera. In *2013 IEEE International Conference on Robotics and Automation*. IEEE, 4712–4719.

[11] Zhengxiong Li, Zhuolin Yang, C. Song, C. Li, Zhengyu Peng, and W. Xu. 2018. E-Eye: Hidden Electronics Recognition through mmWave Nonlinear Effects. *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems* (2018).

[12] Chia-Kai Liang, Li-Wen Chang, and Homer H Chen. 2008. Analysis and compensation of rolling shutter effect. *IEEE Transactions on Image Processing* 17, 8 (2008), 1323–1330.

[13] 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*. ACM, Munich Germany, 243–255. https://doi.org/10.1145/3210240.3210332

[14] Ziwei Liu, Feng Lin, Chao Wang, Yijie Shen, Zhongjie Ba, Li Lu, Wenyao Xu, and Kui Ren. 2023. CamRadar: Hidden Camera Detection Leveraging Amplitude-modulated Sensor Images Embedded in Electromagnetic Emanations. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 4 (Jan. 2023), 173:1–173:25. https://doi.org/10.1145/3569505

[15] Inc. Lume Cube. [n. d.]. Strobe - Anti-Collision Lighting for drones. https://lumecube.com/products/strobe

[16] Marc Roessler. 2002. How to find hidden cameras. *Accessed: May* 16 (2002).

[17] Sriram Sami, Sean Rui Xiang Tan, Bangjie Sun, and Jun Han. 2021. LAPD: Hidden Spy Camera Detection Using Smartphone Time-of-Flight Sensors. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems (SenSys '21)*. Association for Computing Machinery, New York, NY, USA, 288–301. https://doi.org/10.1145/3485730.3485941

[18] Rahul Anand Sharma, Elahe Soltanaghaei, Anthony Rowe, and Vyas Sekar. 2022. Lumos: Identifying and Localizing Diverse Hidden IoT Devices in an Unfamiliar Environment. *Proceedings of the 31st USENIX Security Symposium* (Aug. 2022).

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

[20] Zhiyuan Yu, Zhuohang Li, Yuanhaur Chang, Skylar Fong, Jian Liu, and Ning Zhang. 2022. HeatDeCam: Detecting Hidden Spy Cameras via Thermal Emissions. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22)*. Association for Computing Machinery, New York, NY, USA, 3107–3120. https://doi.org/10.1145/3548606.3560669

[21] Shilin Zhu, Chi Zhang, and Xinyu Zhang. 2017. LiShield: Privacy Protection of Physical Environment Against Photographing. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*. 522–524.

[22] Agustin Zuniga, Naser Hossein Motlagh, Mohammad A. Hoque, Sasu Tarkoma, Huber Flores, and Petteri Nurmi. 2022. See No Evil: Discovering Covert Surveillance Devices Using Thermal Imaging. *IEEE Pervasive Computing* 21, 4 (Oct. 2022), 33–42. https://doi.org/10.1109/MPRV.2022.3187464