SMART: Screen-based Gesture Recognition on Commodity Mobile Devices

Zimo Liao[1], Zhicheng Luo[2], Qianyi Huang[2,5], Linfeng Zhang[3], Fan Wu[1], Qian Zhang[4], Yi Wang[2,5]
[1]Shanghai Jiao Tong University
[2]Southern University of Science and Technology [3]Tsinghua University
[4]Hong Kong University of Science and Technology [5]Peng Cheng Laboratory
{zimo_liao, wu-fan}@sjtu.edu.cn, zhichluo.chn@gmail.com, huangqy@sustech.edu.cn,
zhang-lf19@mails.tsinghua.edu.cn, qianzh@cse.ust.hk, wy@ieee.org

ABSTRACT

In-air gesture control extends a touch screen and enables contactless interaction, thus has become a popular research direction in the past few years. Prior work has implemented this functionality based on cameras, acoustic signals, and Wi-Fi via existing hardware on commercial devices. However, these methods have low user acceptance. Solutions based on cameras and acoustic signals raise privacy concerns, while WiFi-based solutions are vulnerable to background noise. As a result, these methods are not commercialized and recent flagship smartphones have implemented in-air gesture recognition by adding extra hardware on-board, such as mmWave radar and depth camera. The question is, can we support in-air gesture control on legacy devices without any hardware modifications?

To answer this question, in this work, we propose *SMART*, an in-air gesture recognition system leveraging the screen and ambient light sensor (ALS), which are ordinary modalities on mobile devices. For the transmitter side, we design a screen display mechanism to embed spatial information and preserve the viewing experience; for the receiver side, we develop a framework to recognize gestures from low-quality ALS readings. We implement and evaluate *SMART* on both a tablet and several smartphones. Results show that *SMART* can recognize 9 types of frequently used in-air gestures with an average accuracy of 96.1%.

CCS CONCEPTS

• **Human-centered computing** → **Human computer interaction (HCI)**; **Gestural input**.

KEYWORDS

Gesture recognition; visible light sensing; device-free;non-intrusive visible communication

ACM MobiCom '21, January 31-February 4, 2022, New Orleans, LA, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-8342-4/22/01…$15.00
https://doi.org/10.1145/3447993.3483243

1 INTRODUCTION

Gesture control is a natural and user-friendly way to interact with devices. It extends the traditional keyboard/touch screen and provides users with great freedom. In home scenarios, smart TV can be directly controlled with gestures, instead of using a remote controller; when driving, the driver can adjust the volume of music using simple gestures, which is less distracting than using touch screens or buttons. Besides, gesture control prevents our hands from physically touching any devices which may carry harmful viruses. This is of vital importance for devices in public areas, such as self-service machines at the airport and vending machines at shopping malls. According to [27], the gesture recognition market is expected to grow at a compound annual rate of 27.0%, from 9.8 billion USD in 2020 to 32.3 billion in 2025.

Although prior works have implemented gesture recognition via hardware on commercial devices like cameras [8], microphones [38, 41, 42], and Wi-Fi radios [14, 31], none of them have as yet been commercialized on mobile devices. Solutions based on cameras and microphones raise privacy concerns, resulting in low user acceptance. Wi-Fi signals have low spatial resolution and thus Wi-Fi-based solutions are sensitive to background noise, such as people/object movement in users' surroundings. Furthermore, solutions based on Wi-Fi mainly rely on specialized NIC models (e.g., Intel 5300) and thus lack generality. Recently, several flagship smartphones have been released on the market and they are equipped with specialized hardware to support in-air gesture recognition. For example, Google Pixel 4 [6] relies on Soli [3], a 60GHz mmWave radar, to sense human gestures in the air; Huawei Mate 30 Pro [7] supports a similar functionality, but it relies on an extra depth camera on the front panel; similarly, LG Q8 ThinQ [9] has a ToF camera on-board to support in-air gesture recognition. This invites the question of whether we can support gesture recognition on legacy devices without any hardware modification?

We observe that we can leverage the "Screen-Hand-ALS (Ambient Light Sensor)" light path to recognize hand gestures, as shown in Figure 1. When a user is performing hand gestures over the screen, the light signal transmitted from the screen is reflected by hand to the ALS on the mobile phone. The amplitude of the reflected light signal received by ALS is relative to the position of the user's hand. Thus, it is possible to infer the hand gesture through analyzing the time-series of ALS readings. Screen and ALS are both ordinary modalities on mobile devices. The ALS is widely deployed on mobile devices (e.g., mobile phones, tablets, and smartwatches [20, 37, 39, 45]), which can sense the ambient light intensity

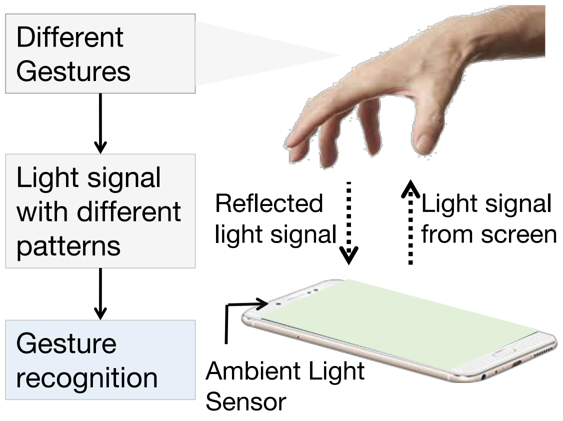


Figure 1: "Screen-Hand-ALS" light path. Light from screen is reflected by the hovering hand, and the ALS can sense the intensity of the reflected light. We analyze the received light signal and recognize different gesture.

and then adjust the brightness of the screen accordingly. Thus, the solution is compatible with commercial-off-the-shelf mobile devices. Different from cameras, ALS can only gain the intensity of ambient light, which contains little sensitive information.

Although the idea sounds straightforward, we are faced with three challenges. The first challenge is to embed spatial information into the "Screen-Hand-ALS" light path so that we can recognize the gesture direction. Since most mobile devices have only one ALS, which is a one-pixel sensor, the receiver has a low spatial resolution. To overcome the limitations of the receiver, the light emitted from the screen must provide as much spatial information as possible. To address this challenge, we model the "Screen-Hand-ALS" channel using lambert's cosine law [15]. We study the influence of the flickering block's position on the light intensity received by a light sensor. Based on the model, we arrange the position of blocks on the screen.

The second challenge is to hide the spatial light signals in the screen content to preserve the viewing experience. To hide the spatial light in the original screen content, we change the original frame into a pair of switching, complementary frames. To overcome the limitations of screen refresh rate, we use the screen's line-by-line refreshing scheme to generate high-frequency signals, which are invisible to human eyes. Besides, we select the RGB value of each frame's pixel according to the flicker fusion rule [47] to ensure the visual effect after frame fusing is nearly the same as the original frame.

The third challenge is to recognize gestures from low-quality ALS data. Since the power of the light signal is restricted by screen brightness, and the diffuse reflection on the user's hand can cause large signal loss, the signal received by ALS is weak. Furthermore, the sampling rate of ALS is even lower than the flickering frequency of the line-by-line refreshing scheme, which causes the under-sampling problem. It is challenging to extract effective features from the low-SNR ALS readings. Based on our analysis, we address this challenge through a signal segmentation and feature selection mechanism. To extract features relative to the gesture, we focus on the signal part with high SNR and carefully choose effective features for classification.

In this paper, we propose *SMART*, which leverages the screen on mobile devices for air gesture recognition. We design the screen update mechanism (the transmitter side) and the gesture recognition framework (the receiver side). We implement and evaluate *SMART* on commercial mobile devices, including a tablet and several smartphones. We evaluate the gesture recognition accuracy, human perception, and processing latency. We also compare *SMART* with a depth camera based approach. The key findings are as follows:

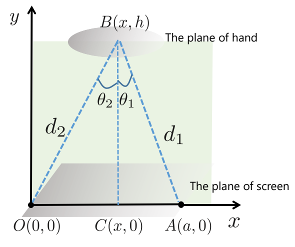
- Recognition Accuracy: We test *SMART* under 5 different lighting conditions with 8 users. *SMART* can recognize 9 types of frequently used in-air gestures with an average recognition accuracy of 96.1%.
- Subjective Viewing Experience: We invite 15 volunteers to evaluate the flickering effect and visual fatigue. We conclude that the design of *SMART* transmitter greatly relieves the flickering effect and visual fatigue. The viewing experience is quite close to the original display.
- Processing Latency: We run *SMART* on different mobile devices and find that the time to process each frame is shorter than the frame-to-frame interval for a 60 FPS (frame per second) display, which verifies that *SMART* can run in real time on these commodity mobile devices.
- Comparison with depth camera: We compare *SMART* with the gesture recognition functionality of Huawei Mate 30 Pro. Results show that *SMART* has comparable gesture recognition performance with depth camera but lower power consumption.

We highlight our main contributions as follows:

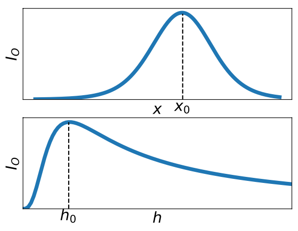
- We propose a new design paradigm based on screen and ALS for gesture recognition on mobile devices. Using a screen as the transmitter, we design a frame switching mechanism to embed spatial information into the original screen content. We also develop a gesture recognition framework based on specific features extracted from the light intensity data collected by ALS.
- We model the "Screen-Hand-ALS" channel to explore the theoretical relationship between the received light power and hand gesture. The model guides us to determine the position of the flickering blocks on the screen.
- We implement and evaluate *SMART*. Evaluation results demonstrate that *SMART* can recognize nine types of frequently used in-air gestures with an average accuracy of 96.1%. We expect that *SMART* provides the legacy device with the same in-air gesture control capability as the expensive flagship smartphones.

2 MODELING "SCREEN-HAND-ALS" CHANNEL

In this section, we model the "screen-hand-ALS" channel, where light emitted by the screen is reflected by the hand and then the reflected light goes to the ALS. This is the fundamental working principle under *SMART*.



(a) Geometric model



(b) Mathematical result

Figure 2: "Screen-Hand-ALS" light path modeling. (a) The geometric illustration of the "Screen-Hand-ALS" model. 'A' denotes a point light source (a pixel); 'B' denotes a reflection point on hand; 'O' denotes the ALS and 'C' is the projection of 'B' on the screen plane. (b) The mathematical relationship between the intensity of the reflected light (I_O) and hand position (x and h).

To achieve gesture recognition based on the "Screen-Hand-ALS" light path, we want to know the relationship between the received light power and hand gesture. *SMART* uses a model to estimate the intensity of reflected light when a hand is placed at a specific position over the screen. As the screen is a combination of discrete light sources (i.e., a large number of pixels) and the shape of hand is complicated, the mathematical model of the light propagation and reflection procedure in the system is intractable. Thus, we build a simplified model to illustrate the basic mechanisms.

In this model, we use the coordinate system shown in Figure 2(a) to illustrate the power loss. Here we assume that the transmitter is a point light source. Although the screen is apparently not a point light source, we can take it as a combination of many discrete points (pixels). The original point O and A are two points on the screen plane and represent the ALS and the point light source (i.e., a pixel), respectively; B is a reflection point on the user's hand, and C is the projection of B on the screen plane.

We first describe the light traveling process: the light from A propagates to B is then reflected by B and received by O. The traveling path can be decomposed into 4 parts: the screen-to-hand path ($A \longrightarrow B$) in free space, reflection by hand (point B), hand-to-sensor ($B \longrightarrow O$) path in free space, and reception at the receiver (point O). Next, we calculate the propagation loss for each part based on the Lambertian radiation pattern [15].

We denote $\angle ABC$ by θ_1 and $\angle OBC$ by θ_2. For light power loss in the free-space, the illuminating path follows the inverse-square law for visible light propagation. The loss from A to B, denoted by l_{AB}, is inversely proportional to $|AB|^2$ (i.e., $l_{AB} \propto \frac{1}{|AB|^2}$) and the loss from B to O follows $l_{BO} \propto \frac{1}{|BO|^2}$. When an area element is radiating as a result of being illuminated by an external source, the irradiance landing on that area element will be proportional to the cosine of the angle between the illuminating source and the normal. A Lambertian scatter will then scatter this light according to the same cosine law as a Lambertian emitter. Thus, the loss at point B (i.e., l_r) caused by reflection follows $l_r \propto (\cos\theta_1 * \cos\theta_2)$, with the Loss at receiver O follows $l_O \propto \cos\theta_2$.

We denote $\overrightarrow{BC} = (0,-h), \overrightarrow{BA} = (a-x,-h), \overrightarrow{BO} = (-x,-h), d_1 = |\overrightarrow{BA}|, d_2 = |\overrightarrow{BO}|$. According to the calculation above, the light intensity of signal from A to O is

$$\begin{aligned} I_O &= I_A \cdot l_{AB} \cdot l_r \cdot l_{BO} \cdot l_O \\ &= I_A \cdot c \cdot \frac{\cos\theta_1}{d_1^2} \cdot \frac{\cos\theta_2^2}{d_2^2} \\ &= I_A \cdot c \cdot \frac{h^3}{((x-a)^2+h^2)^{\frac{3}{2}}(x^2+h^2)^2}, \end{aligned} \tag{1}$$

where c is a constant. Although c may depend on the user's skin color, its effect can be eliminated by some normalization techniques.

Now we know the light signal's amplitude at point O is related to x and h, that is, the hand's position. Particularly, we analyze the relationship between the amplitude and x, h. When $h > 0.5a$[1], for the same value of h, when x is becoming larger, I_O is first monotonically increasing and then monotonically decreasing. I_O has the maximum value when $x = x_0$[2], $x_0 \in (0,a)$. For the same value of x, I_O increases first and then decreases with the increasing h. When $h = h_0$[3], the value of I_O is maximum. How I_O changes with x and h is shown in Figure 2(b).

According to our analysis above, the hand's position affects the path loss during the propagation of light signal from screen to ALS. Therefore, we can exploit the "Screen-Hand-ALS" light path to design *SMART*.

3 OVERVIEW

In this section, we provide the system overview of *SMART*. As we illustrated above, *SMART* leverages the "Screen-Hand-ALS" light path to implement in-air gesture control on legacy mobile devices. As shown in Figure 3, the design of *SMART* mainly contains the transmitter side and the receiver side.

For the transmitter side (the screen display), *SMART* embeds spatial information into the light signal while preserving the viewing experience. *SMART* designs a mechanism to decouple the original frame into a pair of switching, complementary frames. As human eyes have persistence-of-vision effect, it looks the same as the original frame. Different blinking blocks are arranged at different positions on the screen to convey spatial information. In order to overcome the restrictions of the screen refresh rate, *SMART* exploits the line-by-line screen refresh mechanism [40] to provide high-frequency signals, which are above the frequency that human eyes can perceive.

For the receiver side (the ALS), *SMART* proposes a framework to recognize gestures from low-quality ALS data. The signal quality of the received light is poor, as the light from the screen is attenuated during propagation and reflection, while the noise level is high. We carefully analyze the sensor data and identify distinguishing features for the gesture recognition task. After feature extraction, *SMART* builds a lightweight classifier for gesture recognition.

[1]The condition is always true in our system since the largest a value in *SMART* represents the width of our mobile devices. h is the height of the hand above screen, which is about 10cm, while the width of screen is usually less than 20 cm.
[2]The analytical solution of x_0 is unsolvable.
[3]$h_0 = ((\frac{x^4}{64} + \frac{3}{4}x^2(x-a)^2)^{\frac{1}{2}} - \frac{x^2}{8})^{\frac{1}{2}}$

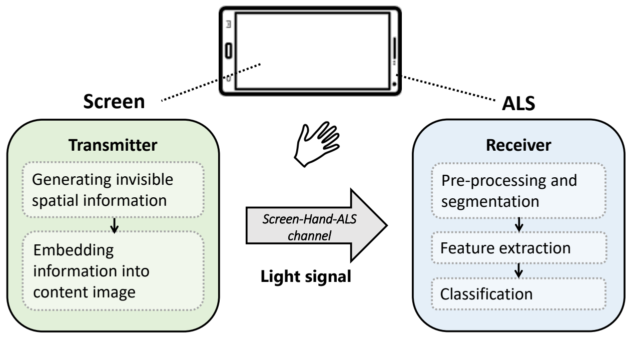

Screen
ALS
Transmitter
Generating invisible spatial information
Embedding information into content image
Screen-Hand-ALS channel
Light signal
Receiver
Pre-processing and segmentation
Feature extraction
Classification

Figure 3: Overview of *SMART* .

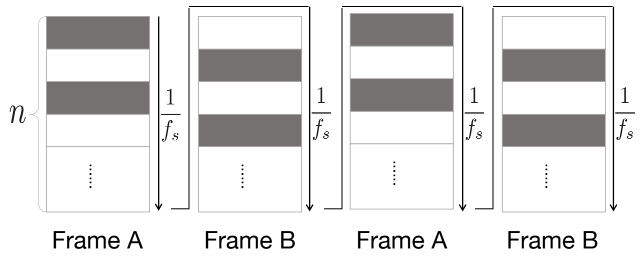


Figure 4: Complementary frames.

4 *SMART* TRANSMITTER

In this section, we present the design of *SMART* transmitter. The main challenge is how to embed spatial information into the screen so that such changes remain invisible to human eyes while remaining obvious to an ALS. It is challenging for two reasons. First, the screen has a limited refresh rate and thus it can only generate low-frequency light signals perceivable by humans. Second, the ALS is a one-pixel hardware. It can only sample the combined light intensity from all light sources around with no spatial resolution. To address these challenges, we first analyze the main differences between light sensors and human eyes.

4.1 Comparison between light sensors and human eyes

To guide the design of *SMART*, we consider the differences between light sensors and the naked eye by focusing on two aspects: 1) what kind of information can be sensed by light sensors? 2) how can it be ignored by human eyes?

The human eye's structure is sophisticated and can perceive abundant information of the images displayed on screens. However, light sensors can simply convert light signal into electrical signals, which means that they can only sense light intensity. The following are the two main differences between them. First, a human eye can be seen as a linear low-pass filter and averages the high-frequency blinking light [35]. In other words, when a light source is flickering at a frequency above a certain threshold, the human eye will not perceive the flickering. However, light sensors can sample at a frequency higher than eyes can perceive. Second, human eyes can discriminate colors with different chromaticities, even if they have the same brightness. However, ALSs on mobile devices can usually only sense the brightness of light. We try to enlarge the signal that can be sensed by light sensors, and diminish the flickering effect that can be perceived by human eyes.

Based on the analysis, we design complementary frames to maximize the light signal received by the ALS while ensuring the signal is unobtrusive to human eyes. The design of *SMART* transmitter has the following three main components:

(1) To avoid the flickering effect, we take advantage of the line-by-line screen display refresh principle, so that switching complementary blocks provides high-frequency light signals, which is above the frequency that human eyes can perceive (Section 4.2);
(2) To keep the perceived colors of complementary frames look the same as the original image, we use the color mixture principle to hide the blocks in the screen content (Section 4.3);
(3) We smooth the edges of complementary blocks to relieve the phantom array effect (Section 4.4).

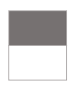
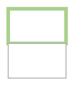
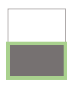
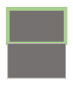
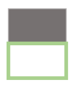
4.2 Complementary block structure design

Specifically, as a low pass filter, human eyes are sensitive to low-frequency light signals (the frequency below 50Hz [46]). When the refreshing rate of the screen is f_s, switching between two complementary frames can provide a light source with a maximum frequency of $\frac{f_s}{2}$. Assume that the refresh rate of the screen is 60Hz, the maximum light frequency can be 30Hz. The screen can also generate the light signal with frequency 20Hz, 15Hz, 10Hz, which are all lower than the threshold. In order to overcome this frequency constraint, we take advantage of the line-by-line refreshing principle.

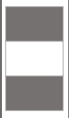
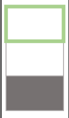
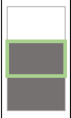
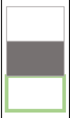
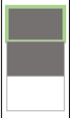
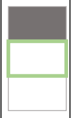
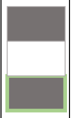
Similar to the rolling shutter effect of a camera, screens on the mobile devices are updated line by line [40]. The pixels in a frame are updated from top to bottom. We design a pair of n-line complementary frames (frame A and frame B) as shown in Figure 4, and switch them continuously. The luminance of the whole screen alternates every $\frac{1}{n \cdot f_s}$, which is shorter than the original refreshing cycle $\frac{1}{f_s}$.

We find that only when n is odd, by switching between two complementary frames, we can get a reliable high-frequency flickering light. The explanation is shown in Figure 5. If n is an odd number, the number of bright blocks constantly increases and decreases alternatively in a consistent pattern, generating a light signal with frequency $\frac{n \cdot f_s}{2}$. However, when n is even, the change from frame A to frame B is not symmetric with the change from frame B to frame A. Thus, the signal with frequency $\frac{n \cdot f_s}{2}$ does not exist. Therefore, an odd n should be chosen for *SMART*.

In addition to the light signal frequency, the value of n also affects the power of the flickering light. With a smaller n, the flickering area is larger, which brings a higher light intensity, as the light intensity is proportional to the flickering area. Thus, the power of the light signal is larger with a smaller n. We prefer a large signal energy to get a higher SNR. Thus, we choose $n = 3$.

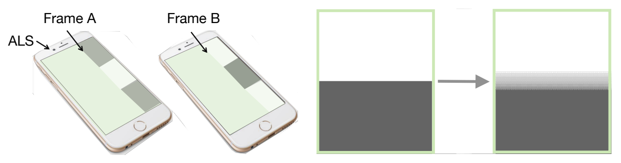
Frame	Frame A		Frame B		Frame A	
Time	0	$\frac{1}{2f_s}$	$\frac{1}{f_s}$	$\frac{3}{2f_s}$	$\frac{2}{f_s}$	...
Frame Content						...
N	1	2	1	0	1	...
$\triangle N$	+1	-1	-1	+1	+1	...

(a) $n = 2$

Frame	Frame A			Frame B			Frame A	
Time	0	$\frac{1}{3f_s}$	$\frac{2}{3f_s}$	$\frac{1}{f_s}$	$\frac{4}{3f_s}$	$\frac{5}{3f_s}$	$\frac{2}{f_s}$	...
Frame Content								...
N	1	2	1	2	1	2	1	...
$\triangle N$	+1	-1	+1	-1	+1	-1	+1	...

(b) $n = 3$

Figure 5: Illustration of the high-frequency light signal generation based on line-by-line refreshing scheme of the screen in detail. Here N denotes the number of bright blocks in the current frame.



(a) Block's positions (b) Edge smoothing

Figure 6: The specific complementary frames on the screen. (a)Positions of blocks and ALS on the mobile device. (b)The edge smoothing scheme to attenuate phantom array effect.

Flickering blocks are arranged on one side of the screen, as shown in Figure 6(a). We design the complementary frames in this way, as we expect that the following three frequency components in the received light signal can be used in gesture recognition:

(1) $f_0 = 0$ Hz: ambient light has a low frequency and the main power falls around 0 Hz [26]. When a hand approaches a light sensor, the DC current from the light sensor decreases under bright conditions or increases under dark conditions. This is because if the user is in a bright room, low-frequency light is mainly from the ambient light, which will be blocked by the palm. However, if the room is dark, low-frequency light signals come mainly from the screen backlight reflected by the user's palm.

(2) $f_1 = \frac{f_s}{2}$ Hz: the two complementary frames alternate at f_s results in f_1. According to the model we build in Section 2, the power of f_1 becomes larger if the hand is moving closer to the whole flickering zone.

(3) $f_2 = \frac{n \cdot f_s}{2}$ Hz: this frequency component is caused by the line-by-line refreshing mechanism. If the hand is approaching the middle part (vertically) of the complementary block area, more light generated by the complementary blocks will be reflected. Thus, the power of f_2 will also change with the hand position.

4.3 Hiding complementary blocks into screen content

In this subsection, we present how we hide the complementary blocks so that users will not perceive the differences from the original display. According to the analysis in Section 7.4, our goal is to maintain chromaticities while maximizing the luminance difference between complementary frames.

RGB color space is widely used on mobile devices, however, it is not designed according to human color vision. Considering human eye perception, we need to convert RGB images into another space that suits the human vision system[47] . Several color spaces are created to quantify human color vision. The CIE 1931 XYZ color space is one of the first defined quantitative links between distributions of wavelengths in the electromagnetic visible spectrum, and physiologically perceived colors in human color vision, which has a linear relationship with RGB color space. Specifically, the color of each pixel is converted from (R, G, B) into (X, Y, Z), in which Y parameter determines the luminance of a color. The chromaticity can be specified by the two derived parameters $x = \frac{X}{X+Y+Z}$ and $y = \frac{Y}{X+Y+Z}$ [5] .

As mentioned before, light sensors can only sense light intensity while ignoring chromaticity. However, human eyes are sensitive to chromaticity change. In order to minimize the eye's perception of the image distortion caused by mixing two frames, we keep the chromaticity (x, y) of the complementary blocks the same as the original pixel (x_0, y_0); on the other hand, we maximize the luminance change of complementary pixels so that the light sensor can receive a stronger flickering light. We model the problem as an optimization problem. Here we denote colors of complementary pixels as (x_1, y_1, Y_1) and (x_2, y_2, Y_2) and the original color is (x_0, y_0, Y_0). ΔY is the luminance difference between a pair of pixels. The optimization problem is shown as follows:

$$\max \quad \Delta Y = |Y_1 - Y_2|$$

$$s.t. \quad \begin{cases} x_1 = x_2 = x_0, \\ y_1 = y_2 = y_0, \\ Y_0 = \frac{Y_1+Y_2}{2} \end{cases}$$

We formulate the problem as a linear optimization problem. Given x_0 and y_0, we can find the best combination of Y_1 and Y_2. In order to reduce the processing latency, for a given pair of x_0 and y_0, we can compute the optimal value of Y_1 and Y_2 offline and store the results in a lookup table. When running, *SMART* can directly search for the optimal value from the lookup table, instead of solving the optimization problem for each pixel. As a frame usually contains a number of pixels with the same color and illuminance, the lookup table can significantly reduce the processing time.

4.4 Edge smoothing

After hiding the well-designed spatial information into the screen content, we find that the high-frequency flickering at the edge of the flickering blocks is still visible when the eyes are blinking or moving. This phenomenon has been mentioned in [35], and it is called the phantom array effect. It is because human eyes are more sensitive to the flickering of moving light sources than the static light source. To mitigate the effect, we smooth the edges of the flickering blocks by scattering the switching pixels near the edges. As shown in Figure 6(b), when approaching to the boundary of the flickering blocks, the density of complementary pixels gradually decreases. Then the abrupt change between two different blocks becomes smooth. In this way, *SMART* relieves the phantom array effect.

5 RECEIVER

SMART receiver extracts features from ALS readings and recognizes different gestures. Our goal is to distinguish 9 in-air gestures that are frequently used in commodity mobile devices. The gestures are illustrated in Figure 7. Specifically, "Left-Right", "Right-Left", "Top-Bottom", "Bottom-Top" can be used for page turning. "Fist" and "Open hand" are commonly used for screenshots and zooming.

The main design challenge is to extract distinguishing features from the light signal. The strength of the signal from the screen becomes weak after the propagation and reflection loss. Besides, the noise is large since the sampling rate of the ambient light sensor is limited [20, 39]. The receiver needs to extract reliable features for gesture recognition from the low-quality, down-sampled signal. To address the challenge, we design strategies to segment the exact gestures from the time-series of signals and extract distinguishable features strongly related to gestures.

5.1 Pre-processing and Segmentation

As present in Section 4.2, we are interested in f_1 and f_2, which are produced by the complementary blocks displayed on screen. Since the sampling rate of ALS (denoted by f_l) on mobile devices is usually low, e.g., 100Hz, according to the Nyquist–Shannon sampling theorem, it can only sample up to $\frac{f_l}{2}$. In order to recover frequency above $\frac{f_l}{2}$, we use frequency aliasing [44].

When sampling a high frequency signal at sub-Nyquist rate, the high frequency component will be aliased to low frequency spectrum as follows:

$$f_a = \begin{cases} (N+1)f_l - f, & f_l/2 < f - N \cdot f_l < f_l \\ f - N \cdot f_l, & 0 \le f - N \cdot f_l \le f_l/2 \end{cases} \tag{2}$$

where f is the signal frequency and f_a is the aliasing frequency, $N = 0, 1, 2, \cdots$. According to Equation (2), we can get the aliasing frequency of f_1 and f_2, denoted by f_1^a and f_2^a, respectively. We use FFT to extract signal energy of f_1 and f_2 (f_1^a and f_2^a actually). Besides, we also utilize power on f_0 to show the relative hand position with ALS. The energy of f_0, f_1 and f_2 are denoted by E_0, E_1 and E_2, respectively.

We segment light signal based on the FFT results. A gesture is detected when the power of f_1 and f_2 is large. This is because only when a user is making a gesture, the "Screen-Hand-ALS" channel can be built and the light signal from screen can be reflected to ALS. Our target is to cut and analyze the segment of light signal with high power (which also means high SNR). As shown in Figure 8, for each gesture, we can get two observations: 1) the power of f_1 is always larger than f_2 during the relatively high light signal period; 2) the time period with high power is nearly the same for both f_1 and f_2. Thus, f_1's power is a reliable indicator for the signal power. Next, we detect the gesture and segment the light signal according to f_1's power with an empirical threshold.

5.2 Feature Analysis and Selection

As mentioned in Section 4.2, the time-series of 0 Hz, f_1, f_2 energy are closely related to hand position. The 3 sequences after segmentation are shown in Figure 10. However, these time series cannot be directly used as features for gesture recognition. According to our observation, we need to address the following two problems.

First, as we have mentioned before, $f_1 = \frac{f_s}{2}$, $f_2 = \frac{n \cdot f_s}{2}$. Since f_2 is f_1's harmonic frequency, a part of f_2's energy is from f_1. Thus, E_1 and E_2 are coupled and E_2 depends on E_1. How to decouple E_1 and E_2 and amplify the difference between E_1 and E_2? We have an observation that the ratio of E_1 and E_2, $R_{12} = \frac{E_1}{E_2}$, is relevant to the hand position. When the hand is approaching the middle of the complementary block area (for example, when the hand is performing "Top-Bottom" gesture) , R_{12} increases; on the contrary, R_{12} decreases if the hand is moving reversely. To illustrate the observation, Figure 9(c) and Figure 9(d) show the pattern of R_{12} for the gestures "Top-Bottom" and "Bottom-Top" as two examples. Thus, we use R_{12} as a feature.

Second, different gestures may have similar patterns of E_0, E_1. For example, for both gestures "Left-Right" and "Right-Left", each pair of patterns are similar, but temporal relationship between E_0 and E_1 is different. For gesture "Left-Right", E_0 decreases first and then E_1 increases. For gesture "Right-Left", E_0 decreases after E_1 increasing. Figure 7 shows the phenomenon directly.

The temporal relationship between E_0 and E_1 can reflect the relative position among the user's hand, the light sensor, and the blinking block, which is an important feature for gesture recognition. In order to capture the inherent temporal relationship, We take the derivative of E_0 and E_1 with respect to time, to show the changing trend. As shown in Figure 9(a) and Figure 9(b), $E_0' \cdot E_1'$ can clearly discriminate between "Left-Right" and "Right-Left" gestures.

To sum up, four feature are used in classification: E_0, E_1, R_{12} and $E_0' \cdot E_1'$. The effect of the later two features on *SMART* is further evaluated in Section 7.1.

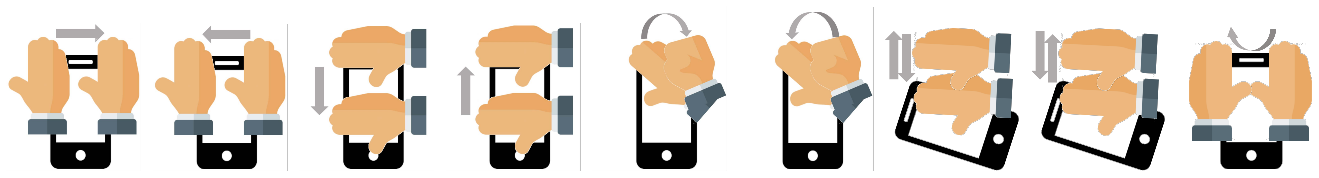
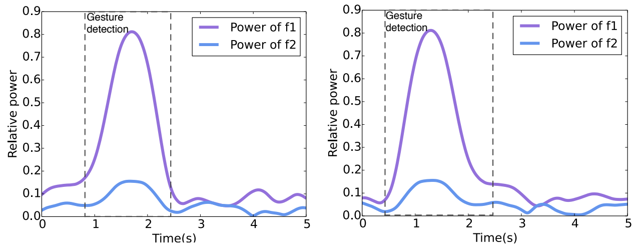

(a) Left-Right (b) Right-Left (c) Top-Bottom (d) Bottom-Top (e) Fist (f) Open hand (g) Up-Down (h) Down-Up (i) FlipPalm

Figure 7: Nine gestures of *SMART* .



(a) Left-Right (b) Top-Bottom

Figure 8: f_1 power and f_2 power throughout the duration of different gestures. Here we use "Left-Right" and "Top-Bottom" gestures as two examples to compare power of f_1 and f_2.

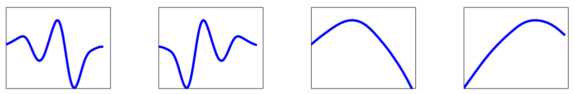

(a) Left-Right (b) Right-Left (c) Top-Bottom (d) Bottom-Top

Figure 9: The patterns of additional features added according to our observation. (a) The pattern of $E_0' \cdot E_1'$ during "Left-Right" gesture. (b) The pattern of $E_0' \cdot E_1'$ during "Right-Left" gesture. (c) The pattern of R_{12} during "Top-Bottom" gesture. (d) The pattern of R_{12} during "Bottom-Top" gesture.

5.3 Classification

After obtaining four feature series, we build a classification model for gesture recognition. Our method is similar to [26]. First, we apply Z-score normalization on features such that every feature stream has zero mean and unit variance. After that, we train a k-nearest neighbour (kNN) classifier using DTW distance as the distance metric in order to eliminate the effect of different gesture speeds. To be specific, the distance from the test point to its neighbouring points is the sum of DTW distances between each pair of feature series. In Section 8, we compare the performance of different models, including recurrent neural networks (RNN), long short-term memory networks (LSTM), multilayer perceptrons (MLP), convolutional neural networks (CNN) and gated recurrent networks (GRN) and kNN.

Table 1: Experiment setting

Item	Number	Value
User	8	5 males, 3 females
Gesture	9	LeftRight, RightLeft, TopBottom, BottomTop, Fist, Openhand, UpDown, DownUp, Flip
Environment	5	0lux, 150lux, 350lux, 700lux, 2000lux

6 PROTOTYPE

We implement *SMART* on a commercial off-the-shelf tablet, i.e., iPad Pro with an 11-inch screen. As the operating system restrains the operation access to the screen driver, we use pre-processed videos to emulate the switching between complementary frames. The blinking blocks are positioned on the right side of the screen. The width of the blinking zone is about 5cm, which can fit onto the screens of the majority of mobile phones[11]. Thus, *SMART* can not only be implemented on tablets, but also on smartphones. By default, the brightness of the screen is 100% and the screen displays a coffee shop picture.

We use a standalone ambient light sensor (i.e., TEMT6000) as the receiver since the operating system also restricts the sampling rate of light sensors on commercial off-the-shelf devices [20]. ALS is connected to an Arduino DUE micro-controller as the receiver. We place the ALS just above the screen as shown in Figure 11 to emulate the relative position between screen and light sensor on commercial devices. The distance between the light sensor and the blocks' left edge is 2.5cm. The default sampling rate of ALS is set to 250Hz, since the integration time of most ALSs are below 4ms [2, 12, 13]. Users perform gestures at approximately 10cm above the screen.

7 EVALUATION

In this section, we evaluate *SMART* in terms of gesture recognition accuracy, user perception, and processing latency. We also conduct an in-depth comparison between *SMART* and the gesture recognition functionality on COTS smartphones.

We test *SMART* in 5 environments with 8 users (5 males and 3 females) in the age range of 20 to 30. Our experiments are conducted in five typical environments. Table 1 summarizes the experiment settings.

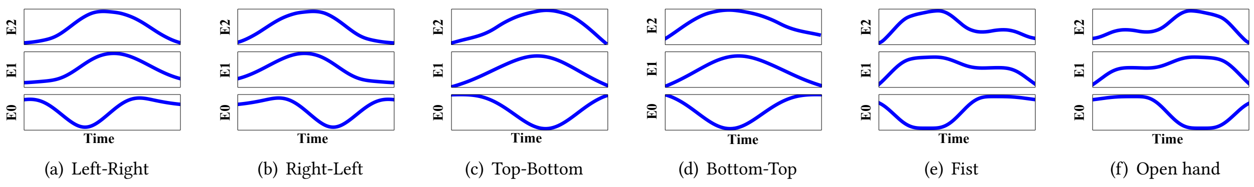


Figure 10: Signal patterns for several gestures.

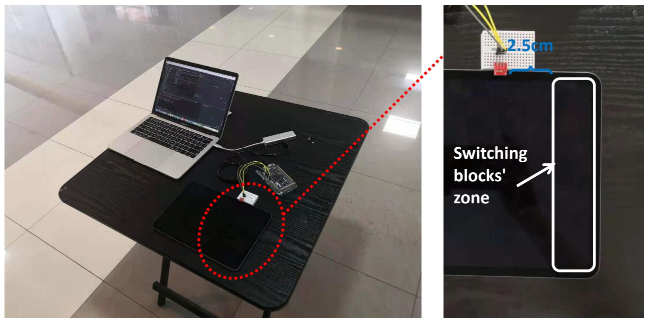


Figure 11: Experiment setup.

7.1 Recognition Accuracy

We evaluate the classification accuracy of *SMART* with different design choices and different environment settings. We ask eight users to perform each gesture 20 times. Besides, to investigate the robustness of *SMART* for various environments, one user is asked to perform each gesture 50 times in each environment. By default, we use the average of 10-fold cross-validation as the final result.

Different Feature Sets. In order to show the effectiveness of the features we have selected in Section 5.2, we compare the recognition accuracy when the system is trained with different sets of features. Feature set 1 only includes the time series of E_0, E_1 and E_2, which are the power of DC, f_1, f_2, respectively. Feature set 2 contains the four features present in Section 5.2. Figure 12 and Figure 13 show the confusion matrices of the recognition framework trained with the two feature sets, separately. We can see that feature set 2 achieves 96.1% accuracy compared to that of 87.3% for the feature set 1. Especially, for the four gestures "TopBottom", "BottomTop", "Openhand" and "Fist", the accuracy is improved from 79.6% to 95.3% with the two carefully designed features, i.e., R_{12} and $E_0' \cdot E_1'$.

Different Lighting Environments. We test 5 static environments that correspond to common lighting conditions:

(1) A completely dark room, where the light intensity is 0 lux.
(2) A conference room with the lighting infrastructure on at night. The average light intensity is about 150 lux.
(3) A lounge environment in the day time, where the average light intensity in the room is about 350 lux.
(4) A normal office in the day time with sunlight and lighting infrastructure. The average light intensity is about 700 lux.
(5) A bright corridor besides a window in the afternoon. The average light intensity is about 2000 lux.

To examine the influence of light fluctuations on recognition accuracy, we also investigate two common dynamic light environments:

(1) Human interference: We ask one subject to perform the nine gestures and another subject is commanded to walk around the place. Each type of gesture is tested for 20 times in 4 light environments (except for the 700lux normal office, since there is no space around the testbed to allow a subject to walk around).
(2) Global light intensity variation: We conduct the experiment in the office with multiple light sources. A user performs each gesture 20 times, while one lamp, on the same desk as the testbed, is switched on/off every 3s. The ALS measures the light intensity changes between 600lux and 750lux.

Figure 14 presents the recognition accuracy under the different light conditions. We can observe that 1) the recognition accuracies under the static environments range from 94.3% to 96.9%, which means that *SMART* works well under static environments. 2) the accuracies in the two dynamic light environments are above 93%. Thus, *SMART* is able to work at various ambient light intensities, from a dark (0lux) to a bright (2000lux) indoor environment, and is robust under dynamic changing light conditions.

User diversity. To investigate the robustness of *SMART* for unseen users, we use both leave-one-out and 10-fold cross validation to evaluate the accuracy of each user. With leave-one-out, the test user's samples are excluded from the training set. The results are shown in Figure 15.

The leave-one-out and 10-fold cross validation results of each user are similar, which means that *SMART* is a generic rather than a personalized model. This is because although the gesture amplitude and velocity are diverse for different users, we apply normalization techniques (Z-score and dynamic time warping introduced in Section 5.3) to cancel out the interference of personal habits and focus on the features that are related to hand gestures.

Unseen Scenarios. We consider the performance of *SMART* for unseen environments. We use leave-one-out cross validation. As shown in Figure 16, we find that we can achieve 96% accuracy with kNN if tested environment's samples are included in the training set, while we achieve 88.7% accuracy for unseen environments.

To improve the performance of unseen scenarios, we further propose to replace the KNN classifier with a gated recurrent neural network (GRN) to achieve better performance. This model is built with two bi-directional gate recurrent layers with dropout for feature extraction and one fully connected layer for classification. Our experiments show that it achieves 93.45% average accuracy on "unseen" environments. Besides, the performance of GRN can be

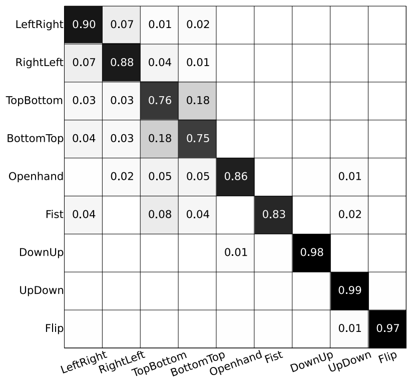


Figure 12: The confusion matrix for feature set 1.

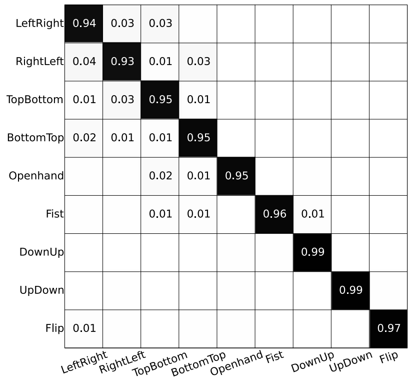


Figure 13: The confusion matrix for feature set 2.

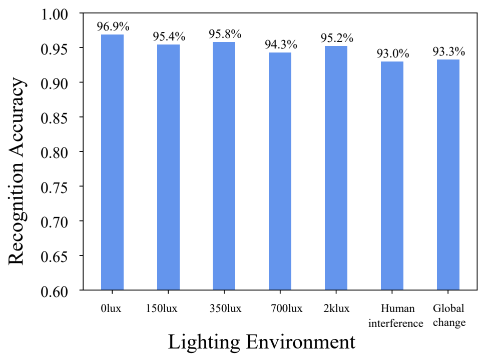


Figure 14: Recognition accuracy in different lighting environments.

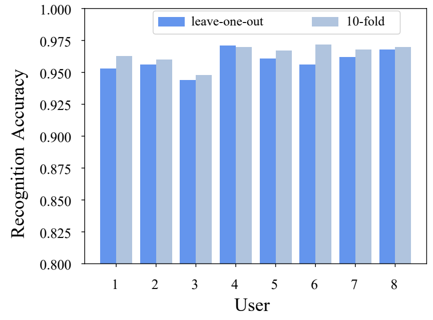


Figure 15: Recognition accuracy of different users.

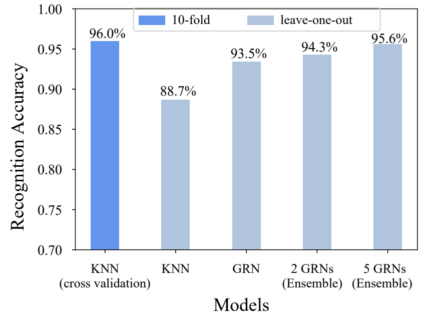


Figure 16: Different models' accuracies under unseen scenarios.

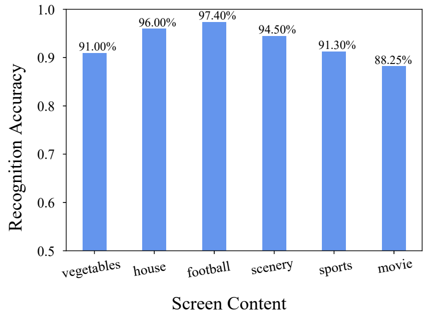


Figure 17: Evaluation with different contents.

improved with model ensemble, which jointly considers the output of multiple models and determines the final label. Usually, model ensemble can promote accuracy at the price of more computation and storage consumption. Our experiments demonstrate that the ensemble of 2 GRNs and 5 GRNs achieve 94.27% and 95.61% average accuracy on "unseen" scenarios, respectively. The results of different models' accuracies are shown in Figure 16.

Different Screen Contents. (1) Static contents: We test the gesture recognition accuracy of 3 different static contents (vegetables, coffee shop and football field). The three contents separately corresponds to three levels of average ΔY: $(20, 40)$, $(40, 60)$, $(60, 80)$. As shown in Figure 17, we can observe that with a larger ΔY, the recognition accuracy becomes higher. It is easy to understand since a larger ΔY means higher SNR of light signals from the screen, leading to more distinguishable features. (2) Dynamic contents: We also test the gesture recognition accuracy of 3 types of dynamic contents including scenery video, sports, and movies. They respectively represent videos with minor, medium, and drastic frame transition. For each video type, we choose 3 video clips, each about 30-90s. During the test for each video clip, we play the video clip on a loop and the subjects perform each gesture 10 times at random moments. As shown in Figure 17, we can see that the gesture recognition accuracy of *SMART* is acceptable when the screen is displaying dynamic content. Although the dynamic content changes the light intensity, for the majority of time, it changes smoothly and slowly. Furthermore, the duration of a gesture is usually short (around 1-2s [26]) and screen light will not change significantly within such a short interval. Thus, hand gestures play a the dominant role in the received light intensity.

7.2 Subjective Perception

We conduct a user study to evaluate the subjective perception quality of the screen display. We invited 15 volunteers (5 female and 10 male) in the age range of 18 to 28 to evaluate 6 different images. The images belong to various types, including natural scenery, sports, food, and buildings. They can be classified according to the range of ΔY: (1)ΔY of cliff and vegetables belongs to $(20, 40)$; (2)ΔY of coffee shop and grassland belongs to $(40, 60)$; (3)ΔY of a football field and tennis court belongs to $(60, 80)$. Each image is shown to the participants in 3 versions. The first version is the original static image without any modification; the second version is dynamically switching between two complementary frames as we designed in Section 4 (without smoothing); the third version adds the edge smoothing scheme (Section 4.4) to the second version. We showed the three versions of each image one by one to each volunteer and asked them to rate each version from three aspects: image content difference, continuous flicker, and visual fatigue. To be specific, we use scores 1 to 4 to indicate the degree of each aspect. Table 2 shows

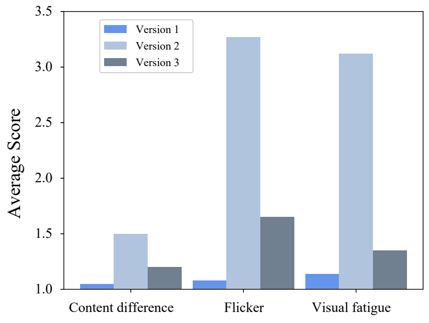


Figure 18: Perception scores of three image versions.

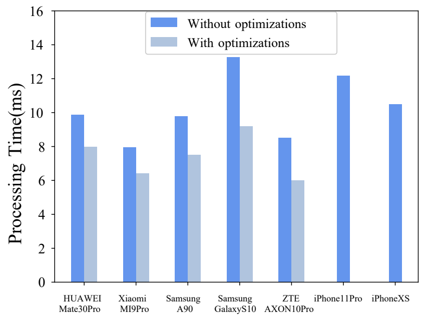


Figure 19: *SMART*'s transmitter processing latency per frame on different commercial devices.

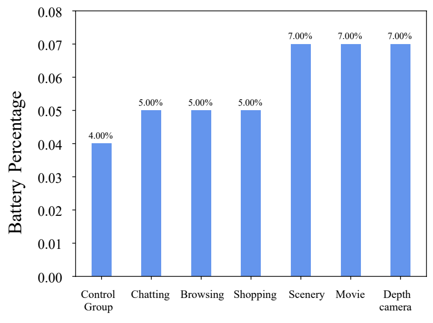


Figure 20: Power consumption of *SMART* and depth camera.

Table 2: Perception scores.

Score	Image content difference	Flicker	Visual fatigue
1	Completely the same	Completely no flicker	No visual fatigue
2	Almost the same	Almost no flicker	A little visual fatigue
3	A little different	A little flicker	Evident visual fatigue
4	Evidently different	Evidently flicker	Strong visual fatigue

the meaning for each score. We treat 1 and 2 as satisfactory scores, which means an acceptable viewing experience.

Figure 18 shows the average scores rated by the volunteers for each version. We arrive at two conclusions. First, from the score result of version 3, we conclude that the visual effect of *SMART* transmitter is acceptable since the average scores for the three aspects are all between 1 and 2. Second, comparing version 2 and version 3, we notice that the visual effect of version 3 is much better than version 2 especially in the respect of "flicker" and "visual fatigue". Without the smoothing scheme, the screen flickering is mostly evident, causing obvious or even strong visual fatigue. After edge smoothing, the scores improve dramatically. This result demonstrates that the smoothing scheme apparently relieves the continuous flickering and visual fatigue.

7.3 Frame Processing Latency

To evaluate *SMART*'s ability to support real-time display, we deployed the frame processing algorithm on both the Android and iOS platform. We run *SMART* transmitter on 5 Android devices (Xiaomi MI9 Pro, Samsung A90, Samsung Galaxy S10, ZTE AXON10Pro), and 2 iOS devices (iPhone 11Pro, iPhone XS) and measure the processing time for each frame.

We test 10 1080p images and 2 videos on different devices. Each image/video is tested on each device 10 times. We perform some simple optimizations to reduce the computation load, including both the spatial domain and the time domain:

(1) Spatial domain: if a block in the frame is of single color (same RGB values), *SMART* does the processing (Section 4.3) only once;
(2) Time domain: if pixels in a frame share the same color with the previous frame, *SMART* reuses the results from the previous frame.

The average result of the processing time for each device is calculated and shown in Figure 19. We can observe that the average processing time of different devices is 6-9ms after optimizations. Thus, it is possible for each frame to be processed and rendered in real time to support 60 FPS dynamic displaying.

7.4 Comparison with depth camera

We compare *SMART* with depth camera in terms of both accuracy and power consumption.

Accuracy. We test the gesture recognition of Huawei Mate 30 Pro, which has a gesture sensor (i.e. a depth camera) on the front panel. As Huawei Mate 30 Pro supports 6 gestures (i.e., "LeftRight", "RightLeft", "TopBottom", "BottomTop", "UpDown", "Fist"), we test each gesture for 30 times in a static light environment. The average accuracy is 93.8%. For *SMART*, the average accuracy for recognizing 9 gestures is 93.0%- 96.9%. Thus, *SMART* has comparable accuracy with the commercial system.

Power consumption. To evaluate the power consumption, we run *SMART* on Huawei Mate 30 Pro. The power consumption of *SMART* comes from two parts:

(1) *SMART* transmitter: It mainly refers to the power consumption for frame processing (Section 4.3). The power consumption for screen display is not included in the measurement, as the screen is always on when the smartphone is in use, whether *SMART* is running or not.
(2) *SMART* receiver: It mainly refers to the the power consumption for running the gesture recognition algorithm. Similar

to screen, ALS is always on when the smartphone is in use, and thus we do not include the power consumption of ALS in the measurement.

We tested 5 types of display content: online chatting, web browsing, online shopping, playing scenery videos and watching movies. We also set a control group (without running the algorithm, but with the screen and ALS on). By looking at the difference between the experimental group and the control group, we can measure the power consumption of *SMART*.

For Huawei Mate30 Pro, we use BatteryManager.BATTERY _PRO PERTY_CAPACITY[10] for reading the battery percentage. In order for the results to be accurate, we let *SMART* run for 1.5 hours for each test. Each case is repeated 3 times. The average battery drop of each type of scene is shown in Figure 20. To measure the power consumption of depth-camera, we use the API function CameraManager.open()[4] to keep the depth-camera on for 1.5 hours and examine the battery drain of the mobile phone. We repeat the experiment 3 times and the battery drop is 7%.

Comparing the power consumption of SMART and depth-camera, we have two observations. First, we found that the power consumption of *SMART* is lower than depth-camera in most cases. It mostly benefits from the time domain optimization, as a large portion of pixels in subsequent frames share a lot of similarity. Second, we found that the power consumption for more drastic frame transition is higher. The reason is that drastic transition leads to more different pixels between the adjacent frames, which means more pixels in the new frames need to be processed.

Jointly considering accuracy and power consumption, *SMART* has comparable gesture recognition performance with depth camera but lower power consumption.

8 DISCUSSION

Degradation in the perception scores. In *SMART*, although we make great effort to reduce the influence on the user's viewing experience, there is still a little degradation in the perception scores. To further minimize the perception degradation, we propose two possible solutions. First, *SMART* can be triggered when detecting hand proximity. This can be achieved with proximity sensor, which is widely available on smartphones. When detecting hand proximity, the system triggers *SMART* to run the screen modulation algorithm, preventing long-term perception degradation. Second, users can choose whether to run *SMART* or not in different application scenarios, so that users can balance between visual perception and convenience. For example, when users are driving, they may put driving safety in the first place and choose in-air gesture control instead of using touch screen or buttons. In this scenario, a little degradation of visual effect is acceptable since driving safety is more important.

Implementation via frame buffer. *SMART* can be implemented via the frame buffer device driver on COTS mobile devices. Frame buffer is the display memory, containing the image that is displayed on the screen. Once the image is mapped to the process address space, through reading and writing the corresponding memory address, users can control what is displayed on the screen. *SMART* can be implemented by processing the data in the frame buffer. For example, on the Android platform, we can use open("/dev/graphics/fb0") to visit the frame buffer and map the address space through the mmap function. Then the frame buffer can be modified by writing the mapped memory space and the new pixels will be displayed on the screen [1].

Model	Accuracy	Parameters	FLOPs
RNN	88.62	135690	13365760
LSTM	97.50	535050	53660160
GRN	97.85	401930	40296960
CNN	89.18	1482	82176
MLP	97.60	323206	322400
KNN	96.00	–	–

Table 3: Experiments of neural network models and the KNN classifier on 10-fold cross validation. "Parameters" and "FLOPs" indicate the number of parameters and the floating point operations in the model.

Other possible classification algorithms. In order to compare KNN against other state-of-the-art models, we have tested with 5 different neural network models, including recurrent neural networks (RNN), long short-term memory networks (LSTM), multilayer perceptrons (MLP), convolutional neural networks (CNN) and gated recurrent networks (GRN). Their 10-fold cross validation accuracies and FLOPs have been shown in Table 3. Our conclusion is that several neural networks achieve higher accuracy than the KNN classifier, but the gap is not very large. We think that is because neural networks are good at extracting representative features from high dimensional data whereas the feature dimension in our task is low. Thus, neural networks do not show their full potential in this task.

9 RELATED WORK

Device-free in-air hand gesture recognition. Most existing in-air gesture recognition systems use customized hardware. In industry, some companies start to manufacture mobile devices supporting gesture recognition such as Huawei [7], LG[9], Google [6], and so on. However, most of them need customized hardware such as radar and depth camera. Soli [3] is a gesture recognition system developed by Google based on a 60GHz wireless signal with mm-level wavelength. Leap Motion [8] uses infrared cameras to sense hand gestures.

In academia, different sensing media are used to sense human hand gestures. Cameras are widely used in the field of gesture recognition [17, 30, 36]. However, such systems may cause privacy problems and heavy computation overhead. Since WiFi signals are ubiquitous in our daily environment, some prior works have also studied the use of WiFi to sense hand gestures [14, 19, 31, 33, 41, 48]. They can be easily affected by electromagnetic interference and can not be used in some RF-inappropriate environments, like hospitals, underground mines and gas stations. Besides, most of them are based on the measurement of CSI to gain accurate gesture recognition. However, only specific Wi-Fi NIC models provide CSI information, while the majority of mobile devices do not provide such information. WiGest [14] uses RSS information which can be achieved on commercial devices. However, it needs a special preamble gesture for gesture detection. Acoustic signal is also commonly used to recognize in-air hand gestures [16, 29, 38, 42]. They also have privacy problems and can be effected by ambient sound interference. LLAP [38] is a device-free gesture tracking system that can be deployed on mobile devices. However, it can be interfered by nearby moving objects and other devices deployed with LLAP.

Visible light based human gesture sensing. Existing studies have explored various gesture sensing modalities based on visible light[21, 23–26, 32, 43]. Okuli[43] is proposed to realize fine-grained finger tracking with an LED and two light sensors. Some human sensing systems [21, 23, 24] based on visible light can reconstruct human gestures. Specifically, LiSense[21] can reconstruct human skeleton postures using several LEDs and photodiodes embedded in the floor and StarLight [23] can gain a more fine-grained sensing posture. Aili [24] can reconstruct hand poses with a table lamp with an LED panel and an array of photodiodes. Some works propose visible light human gesture recognition methods. LiGest[32] uses a grid of light sensors deployed on the floor to build an ambient light based gesture recognition system. [25] develops a gesture recognition system using small, low-cost photodiodes for both energy harvesting and sensing. SolarGest [26] is designed to recognize hand gestures near a solar-powered device. However, they can not be used on mobile devices directly since they all need customized devices or previous deployment in the environment.

Hidden Screen-Camera Communication. Prior work [18, 22, 28, 34, 35, 46] utilizes the gap of perception ability between human eyes and cameras to hide unobtrusive information in a given screen content while realizing the screen-camera communication. InFrame++ [35] convert barcodes into complementary frames to enable screen-camera communication. They leverage the flicker fusion property of HVS to keep the visual effect of switching complementary frames seems like the original content. HiLight[22] encodes data into translucency change and enables any-scene communication. Since HiLight changes the translucency instead of the RGB value of each pixel, it realizes real screen-camera communication. Chromacode[46] excels in its adaptive embedding in uniform color space and realizes imperceptible, high rate, and reliable communication. Our techniques are relevant to screen-camera communication methods, but also different from them. While existing methods realize hidden screen-camera communication, we try to realize the unobtrusive communication between screen, hand and light sensor. Unlike cameras collecting complete vision information, ambient light sensors can only sample light intensity, which makes unobtrusive communication non-trival.

10 CONCLUSION

In this paper, we proposed *SMART*, a screen-based gesture recognition scheme, which enables in-air gesture control on legacy mobile devices. *SMART* exploits the "Screen-Hand-ALS" light path to recognize the in-air gesture. We have implemented and evaluated *SMART* on both a tablet and several smartphones. Results show that *SMART* can recognize 9 frequently used gestures with 96.1% accuracy, while it still preserves the viewing experience of the screen. We expect that *SMART* provides legacy devices with the same in-air gesture control capability as flagship smartphones.

ACKNOWLEDGMENTS

This work was supported in part by National Key R&D Program of China No. 2019YFB2102200, in part by China NSF grant No. 62002150, 62025204, and 61972252, in part by the Key-Area Research and Development Program of Guangdong Province No. 2020B0101390001 and 2020B010164001, in part by Alibaba Group through Alibaba Innovation Research Program. The opinions, findings, conclusions, and recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies or the government. Q. Huang is the corresponding author of this paper.

REFERENCES

[1] 2018. Introduction and use of Android Framebuffer. https://kaustav.space/2018-09-22/introduction-to-android-framebuffer

[2] 2019. APDS-9253-001. https://docs.broadcom.com/docs/APDS-9253-001-DS

[3] 2019. Google project soli. https://www.google.com/atap/project-soli/

[4] 2021. Camera2. https://developer.android.com/training/camera2

[5] 2021. CIE 1931 color space. https://en.wikipedia.org/wiki/CIE_1931_color_space

[6] 2021. Google pixel 4. https://store.google.com/product/pixel_4a

[7] 2021. Huawei mate30. https://consumer.huawei.com/en/phones/mate30/

[8] 2021. Leap Motion,. https://www.leapmotion.com/

[9] 2021. LG G8 ThinQ. https://www.lg.com/us/cell-phones/lg-LMG820UM1-att-g8-thinq

[10] 2021. Measuring Device Power. https://source.android.com/devices/tech/power/device

[11] 2021. Smartphone unit shipments worldwide by screen size from 2018 to 2022. https://www.statista.com/statistics/684294/global-smartphone-shipments-by-screen-size/

[12] 2021. TSL2540 Ambient Light Sensor. https://ams.com/zh/tsl2540

[13] 2021. TSL2740 Ambient Light Sensor. https://ams.com/tsl2740

[14] Heba Abdelnasser, Moustafa Youssef, and Khaled A Harras. 2015. Wigest: A ubiquitous wifi-based gesture recognition system. (2015), 1472–1480.

[15] J. R. Barry. 1997. Wireless Infrared Communications. *PROCEEDINGS- IEEE* 85, 2 (1997), 265–298.

[16] Sidhant Gupta, Daniel Morris, Shwetak Patel, and Desney Tan. 2012. Soundwave: using the doppler effect to sense gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1911–1914.

[17] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, and Andrew W. Fitzgibbon. 2011. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Acm Symposium on User Interface Software & Technology*.

[18] M. Izz, Z. Li, H. Liu, Y. Chen, and F. Li. 2016. Uber-in-light: Unobtrusive visible light communication leveraging complementary color channel. In *IEEE INFOCOM 2016*.

[19] B. Kellogg, V. Talla, and S. Gollakota. 2014. Bringing gesture recognition to all devices. In *Usenix Conference on Networked Systems Design & Implementation*.

[20] L. Li, P. Hu, C. Peng, J. Shen, and A. F. Zhao. 2014. Epsilon: A Visible Light Based Positioning System. *USENIX Association* (2014).

[21] Tianxing Li, Chuankai An, Zhao Tian, Andrew T Campbell, and Xia Zhou. 2015. Human sensing using visible light communication. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. 331–344.

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

[23] T. Li, Q. Liu, and X. Zhou. 2017. Practical Human Sensing in the Light. *GetMobile: Mobile Computing and Communications* 20, 4 (2017), 28–33.

[24] Tianxing Li, Xi Xiong, Yifei Xie, George Hito, Xing-Dong Yang, and Xia Zhou. 2017. Reconstructing hand poses using visible light. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 3 (2017), 1–20.

[25] Yichen Li, Tianxing Li, Ruchir A Patel, Xing-Dong Yang, and Xia Zhou. 2018. Self-powered gesture recognition with ambient light. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. 595–608.

[26] Dong Ma, Guohao Lan, Mahbub Hassan, Wen Hu, Mushfika B Upama, Ashraf Uddin, and Moustafa Youssef. 2019. Solargest: Ubiquitous and battery-free gesture recognition using solar cells. In *The 25th Annual International Conference on Mobile Computing and Networking*. 1–15.

[27] Markets and Markets. 2020. *Gesture Recognition and Touchless Sensing Market - Global Forecast to 2025.* Technical Report.

[28] V. Nguyen, Y. Tang, A. Ashok, M. Gruteser, and N. Mandayam. 2016. High-rate flicker-free screen-camera communication with spatially adaptive embedding. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications.*

[29] W. Ruan, Q. Z. Sheng, Y. Lei, G. Tao, and L. Shangguan. 2016. AudioGest: enabling fine-grained hand gesture detection by decoding echo signal. In *Acm International Joint Conference on Pervasive & Ubiquitous Computing.*

[30] T. Sharp, Y. Wei, D Freedman, P. Kohli, E. Krupka, A. Fitzgibbon, S. Izadi, C. Keskin, D Robertson, and J. Taylor. 2015. Accurate, Robust, and Flexible Real-time Hand Tracking. In *Acm Conference on Human Factors in Computing Systems.* 3633–3642.

[31] Li Sun, Souvik Sen, Dimitrios Koutsonikolas, and Kyu-Han Kim. 2015. Widraw: Enabling hands-free drawing in the air on commodity wifi devices. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking.* 77–89.

[32] Raghav H Venkatnarayan and Muhammad Shahzad. 2018. Gesture recognition using ambient light. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 1 (2018), 1–28.

[33] A. Virmani and M. Shahzad. 2017. Position and Orientation Agnostic Gesture Recognition Using WiFi. *International Conference on Mobile Systems* (2017).

[34] D Wan, J. C. Liando, and L. Mo. 2016. SoftLight: Adaptive visible light communication over screen-camera links.. In *IEEE INFOCOM 2016 - IEEE Conference on Computer Communications.*

[35] A. Wang, Z. Li, C. Peng, Guobin Shen, and Z. Bing. 2015. InFrame++: Achieve Simultaneous Screen-Human Viewing and Hidden Screen-Camera Communication. In *13th ACM Annual International Conference on Mobile Systems, Applications, and Services.*

[36] R. Y. Wang, S. Paris, and J. Popovic. 2011. 6D Hands: Markerless Hand Tracking for Computer Aided Design. In *Acm Symposium on User Interface Software & Technology.*

[37] Z. Wang, Z. Yang, Q. Huang, L. Yang, and Q. Zhang. 2019. ALS-P: Light Weight Visible Light Positioning via Ambient Light Sensor. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications.*

[38] W. Wei, A. X. Liu, and S. Ke. 2016. Device-free gesture tracking using acoustic signals. In *International Conference on Mobile Computing & Networking.*

[39] L. Yang, Z. Wang, W. Wang, and Q. Zhang. 2018. NALoc: Nonlinear Ambient-Light-Sensor-based Localization System. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 4 (2018), 1–22.

[40] Z. Yang, J Zhang, Z. Wang, and Z. Qian. 2018. Lightweight Display-to-device Communication Using Electromagnetic Radiation and FM Radio. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* (2018).

[41] N. Yu, W. Wang, A. Liu, and L. Kong. 2018. QGesture: Quantifying Gesture Distance and Direction with WiFi Signals. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* (2018).

[42] S. Yun, Y. C. Chen, H. Zheng, L. Qiu, and W. Mao. 2017. Strata: Fine-Grained Acoustic-based Device-Free Tracking. In *International Conference on Mobile Systems.*

[43] Chi Zhang, Josh Tabor, Jialiang Zhang, and Xinyu Zhang. 2015. Extending mobile interaction through near-field visible light sensing. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking.* 345–357.

[44] Chi Zhang and Xinyu Zhang. 2016. LiTell: Robust indoor localization using unmodified light fixtures. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking.* 230–242.

[45] Chi Zhang and Xinyu Zhang. 2017. Pulsar: Towards ubiquitous visible light localization. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking.* 208–221.

[46] K. Zhang, C. Wu, C. Yang, Y. Zhao, and Z. Yang. 2018. ChromaCode: A Fully Imperceptible Screen-Camera Communication System. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking.*

[47] Lan Zhang, Cheng Bo, Jiahui Hou, Xiang-Yang Li, Yu Wang, Kebin Liu, and Yunhao Liu. 2015. Kaleido: You can watch it but cannot record it. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking.* 372–385.

[48] Yue Zheng, Yi Zhang, Kun Qian, Guidong Zhang, Yunhao Liu, Chenshu Wu, and Zheng Yang. 2019. Zero-effort cross-domain gesture recognition with Wi-Fi. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services.* 313–325.