StrLight: An Imperceptible Visible Light Communication System with String Lights

Huanle Zhang, *Student Member, IEEE,* Wan Du, *Member, IEEE,* Mo Li, *Member, IEEE,* Kaishun Wu, *Member, IEEE,* and Prasant Mohapatra, *Fellow Member, IEEE*

Abstract—This paper presents StrLight, the first practical VLC system that leverages widely-deployed string lights to transmit data. The data transmission is imperceptible to human eyes. Users can decode the data by mobile devices (e.g., smartphones) equipped with cameras. StrLight primarily differs from existing VLC systems in using string lights which are composed of a large number of small LEDs and thus the unique design to address practical issues including a special data modulation/encoding scheme, a data representation with unstructured/unknown topologies of LEDs in the string light, and a fault tolerance against broken and blocked LEDs. To the best of our knowledge, StrLight is the first practical VLC system of its kind. We build several prototypes of string light transmitters and test with different smartphone models and a customized mobile device as receivers. The experiment results show that StrLight provides an efficient and robust data broadcasting. A string light of 100 LEDs working in $450Hz$ and a camera with a capture rate of $30Hz$ and an image resolution of as low as 320×240 pixels, delivers data rate of $\sim1kbps$, without observable light flickers.

Index Terms—Visible Light Communication, String Light, Mobile System, Data Broadcasting.

1 Introduction

Digital signages are ubiquitous in people's daily lives. They have widespread applications in broadcasting public information, advertising and promotion, enhancing customer experience, to name a few [1]. According to the industry report that is jointly conducted by five companies (i.e., Cisco, HP, LG, 3M, Samsung an Panasonic), global digital signage market is expected to grow to 21.92 billion dollars in 2020, with a compound annual growth rate of 8.04% during the period 2015 to 2020 [2].

Traditional digital signages are formed into rectangular displays and thus very often difficult to blend into surrounding environments, preventing them from further improving user experience. At a fundamental level, displays are just a collection of individually controllable pixels, fixed into two-dimensional grid (i.e., row-column addressing architecture) [3], [4]. Recently, a Microsoft research group advocates the idea of *autonomous pixels* (spatial pixels) in which pixels are scattered over space [4], [5]. Compared to rectangular displays, spatial pixels have the advantages of flexible display geometry, flexible display density and generality [3], [4], [5]. However, existing works on spatial pixels are detached from reality in that researchers "forcefully" group LEDs that are intrinsically not related to each other.

This paper presents StrLight, the first practical VLC system that leverages string lights to transmit data. The data are encoded by turning *on* or *off* LEDs. The LEDs flash fast to avoid light flickers that may cause stress and anxiety [6], Thus, the data transmission is imperceptible to humans but a mobile device (e.g., a smartphone) can decode the data carried by the LED flashes.

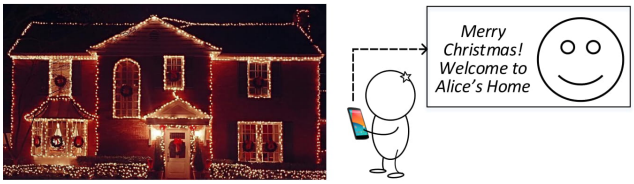


Fig. 1: One application of StrLight in personal use case. StrLight leverages string lights to transmit data, and visitors use mobile devices equipped with cameras

String lights are widely used. They are composed of a large number of small Light Emitting Diodes (LEDs) that are connected by a wire. A string light system has the following advantages compared to physically displaying messages using string lights: (1) *Flexibility*: overhead of changing the transmission message is negligible, whereas it is normally overwhelming to manually redeploy the string lights for the new message; (2) *Multimedia*: it allows owners to broadcast multimedia messages such as images and videos, while the counterpart can only display texts or simple lines and curves; (3) *Privacy* [7], [8]: encryption to the message is support, whereas the counterpart cannot hide/protect the message from non-target people [9]. A string light system could increase revenue for store owners by improving customer experience [10], [11], boost tourism revenue by creating smart cities [12], [13], and amuse people. Figure 1 illustrates one application of StrLight in personal use case. In addition to the above advantages, a string light system achieves much higher data rates than regarding all the LEDs as "one" LED. Therefore, a string light system can be applied directly to boost the data rates of mainstream lighting panels that are composed of many small LEDs.

Translating such an idea of string lights based communication into a system, however, entails several challenges. *First*,

- *Huanle Zhang and Prasant Mohapatra are with the Department of Computer Science, University of California, Davis, USA 95616. E-mail: dtczhang@ucdavis.edu; pmohapatra@ucdavis.edu.*
- *Wan Du is with the Department of Computer Science and Engineering, University of California, Merced, USA 95343. E-mail: wdu3@ucm.edu.*
- *Mo Li is with the School of Computer Engineering, Nanyang Technological University, Singapore 639798. E-mail: limo@ntu.edu.sg.*
- *Kaishun Wu is with the School of Computer Science and Engineering, Shenzhen University, Shenzhen, China 518061. E-mail: kwinson@cse.ust.hk.*

to avoid harmful light flickers and imperceptibly transmit data, LEDs are required to flash in hundreds of hertz. To recover signal waveform (Nyquist sampling theorem) and thus decode the transmitted data, either photodiodes with high-sampling-rate Analog-to-Digital Converters (ADCs) [14], [15] or complicated design based on rolling shutter effect of Complementary Metal-Oxide Semiconductor (CMOS) cameras [16], [17], [18], [19] is required. However, neither of them can be applied to a string light system due to the large number of small LEDs (we further clarify this argument in §2.1). *Second*, LEDs in a string light are connected by a wire, which facilitates a structured data representation as in screen-to-camera systems [20], [21]. However, in everyday usage, string lights may be formed into different shapes allowing freedom of LED arrangements. Existing structured data representation schemes for screen-to-camera systems cannot be applied to a string light system due to the unknown linking relationship of LEDs on the captured images at the receiver side. *Third*, some LEDs in a string light may be blocked from time to time. Fault tolerance mechanisms are required to guarantee transmission correctness and efficiency in the presence of the disrupted linking relationship of LEDs caused by the blocked or broken LEDs.

To tackle the above challenges, we design and implement StrLight. *First*, to avoid using rolling shutter effect in demodulation, we adopt On-Off Keying (OOK) modulation, in which LED status on/off represents bit 1/0. It is easy for receivers to demodulate the bit value in the capture image. However, if the data need to transmit the same bit value by a LED for a relatively-long duration, OOK causes severe light flickers for user eyes. To increase the on/off frequency of LEDs and in turn mitigate light flickers, we transmit one frame in two consecutive time slots, in which the status of every LED is toggled. A new OOK encoding and decoding method is proposed to flexibly use both on/off to present the same bit value and to handle the frame mixture problem caused by rolling shutter effect. *Second*, in StrLight, the transmitter controls LEDs by the LEDs' indexes in the string light, e.g., turn on the 1st LED, turn off the 2nd and 3rd LEDs, etc. However, the receiver cannot directly identify the sequence of LEDs in the captured images, as the wire of the string light cannot be efficiently detected. To correctly demodulate the transmitted data in the images, we design a topology discovery, which leverages a new frame format and some unique characteristics of the above proposed encoding and decoding method to help the receiver to identify the sequence of the captured LEDs. *Finally*, a new link transmission scheme is developed to enable the receiver to identify the missing LEDs by processing a batch of received frames and in turn recover the missing bits in frames.

To the best of our knowledge, StrLight is the first practical VLC system that leverages low-profile string lights to imperceptibly transmit data. StrLight is a ready-to-use system with extremely low cost. In our prototype system, the transmitter includes a string light (each LED costing $\$0.008$), a customized LED driving board ($\$4.0$) and a low-cost micro-controller (Raspberry Pi II). We exhaustively evaluate StrLight under different conditions. We use three Commercial-Off-The-Shelf (COTS) smartphone models and a customized mobile device as receivers. We build five string lights of 80 to 100 LEDs and form them into different shapes. The experiment results show that StrLight provides an efficient and robust communication. For example, a string light of 100 LEDs working in $450Hz$, and a camera with a capture rate of $30Hz$ and an image resolution of 320×240, delivers a data rate of $\sim1kbps$.

2 Background & Challenges

Compared with existing VLC systems [3], [17], [18], [19], [22], the unique features of string light communication, i.e., small size and large number of LEDs, unknown linking relationship of LEDs, and LED failure/blockage, make StrLight a difficult system design.

2.1 Data Modulation & Encoding

We consider the most widely-deployed low-profile string lights in which the LEDs can only be turned *on* or *off*. Based on the *on* and *off* control of LEDs, we may use existing modulation schemes such as Frequency Shift Keying (FSK), Pulse Width Modulation (PWM) and Pulse Phase Modulation (PPM) [23]. To avoid light flickers that may cause stress and anxiety [6], LEDs need to flash in at least hundreds of hertz. To recover the signal waveform and thus decode the data, receivers require to sample in at least double the frequency of transmitting signals (Nyquist sampling theorem). To meet the sampling rate requirement, existing VLC systems either rely on photodiodes with high sampling-rate ADCs [14], [15] or rolling shutter effect of CMOS cameras [17], [18], [19]. However, they cannot be applied to the proposed string light communication system.

Photodiodes with high-sampling-rate ADCs. A photodiode detects accumulated light intensities from all LEDs within its Fields of Vision (FoV). To distinguish signals from each LED, a unique base flashing frequency is required for each LED (in analogy to a carrier frequency in radio communication). Unlike radio front-ends, however, a LED is not equipped with a bandpass filter to compress its interference to other frequency bands from the harmonics of its base flashing frequency. Assume LED i works in a base frequency of f_i and a signal bandwidth of w_i. The interfered frequency bands caused by LED i are $(k \cdot f_i - w_i/2,\ k \cdot f_i + w_i/2),\ k \in \mathbb{Z}$. A string light has a large number of LEDs, say 100 ($i = 1, 2, ..., 100$), and thus it is difficult or even impossible to solve f_i and w_i to avoid cross-interference. Please refer to [15] for the consideration of determining f_i and the difficulty of f_i selection when only 5 LEDs are concerned, not to mention hundreds of LEDs in a string light. Moreover, the receiver requirement is boosted since receivers need to sample in frequency of at least $2 \times \max_i f_i$ (theoretically, the sampling rate is infinite due to the un-compressed harmonics), in demanding of higher-speed ADCs and resulting in increased system complexity and more energy consumption. Therefore, photodiodes based solutions cannot be applied to a string light communication system of hundreds of LEDs, because allocating each LED with an identifiable base flashing frequency is unpractical.

Rolling shutter effect of CMOS cameras. Compared to photodiode-based transmission schemes, camera-based schemes do not need to assign LEDs with different and distinguishable base flashing frequencies since receivers can spatially separate signals from LEDs on the captured images. However, the capture rates of cameras are very limited, e.g., 60 Frames Per Second (FPS) on mainstream smartphones. To decode data from LEDs working in high frequencies, rolling shutter effect of CMOS cameras is explored by some systems [17], [18], [19]. The flashes of LEDs result in bright and dim strips on a captured image from a CMOS camera. However, such a decoding method requires that each LED occupies a sufficient number of pixel rows on the captured image and thus it only supports a few big LED bulbs. For example, Luxapose [18] decodes data from 5 LEDs using a CMOS camera with an image resolution of 7712×5360 pixels and recent work,

RollingLight [19], improves data decoding from 4 LEDs with an image resolution of 1920×1080. In our prototype system, a string light has 100 small LEDs and every LED only occupies less than 20×20 pixels in one captured image. As a consequence, rolling shutter effect cannot be used to measure parameters of LED flashes (e.g., frequencies, pulse width or pulse phases).

Consequently, a new modulation and encoding scheme is required to enable data transmission from a large number of small LEDs in a string light to cameras.

2.2 Data Representation

A string light system transmits data simultaneously using a large number of LEDs. To format data from LEDs into a meaningful bit stream, existing works on spatial pixels [3], [4], [24], [25] send LEDs' IDs before transmitting data. However, the proposed string light communication system targets the come-and-serve data broadcasting applications, in which transmitters and receivers are not synchronized and cannot exchange LEDs' IDs beforehand.

Different from existing works [3], [4], [24], [25] in which LEDs are un-related in terms of identification, LEDs in a string light are connected by a wire and thus they can be indexed based on the topology of the string light. Therefore, a string light system can leverage "location"-based identification of LEDs/pixels as in screen-to-camera VLC systems [20], [21]. Without requiring explicitly sending IDs, the system is simplified and thus more practical, especially for a string light system with hundreds of LEDs that need to be identified.

Unlike the row-column addressing architecture in screen-to-camera systems, however, the string light is used to form arbitrary shapes. It is hard for receivers to recognize the sequence of LEDs, as the wire is difficult or even impossible to be detected on captured images. Consequently, a topology discovery scheme is required to identify the linking relationship of LEDs and further form the decoded bits from all LEDs on the wire into a meaningful data stream.

2.3 Fault Tolerance

We format bits from all LEDs at the same time into a data frame. However, if some LEDs are blocked or broken, they are always dim on captured images and cannot be detected by receivers. Because string lights do not have a structured topology, it is difficult for receivers to know the indexes or even the number of undetected LEDs. In the presence of some blocked or broken LEDs, receivers cannot infer the indexes of working LEDs in the frame as well. As a consequence, lacking the index information of LEDs, it is impossible to apply Forward Error Correction (FEC) codes or rateless codes [17], [26] to recover the data carried by the undetected LEDs. Existing VLC systems, especially screen-to-camera systems [20], [21], do not have such an issue, since they have structured data representation and thus they can easily localize the undetected symbols (i.e., the indexes of working pixels in the data frame are known).

2.4 Other Considerations

To develop a practical string light communication system, we make the following practical considerations.

- *A one-way broadcasting system*. Our system aims at broadcasting data from string lights to visitors. Transmitters with string lights do not receive any signals/feedback from receivers with cameras.
- *Ignorant transmitters and receivers*. Receivers have no prior knowledge of the string light, e.g., the number of LEDs in the string light and the topology of the string light. In addition, transmitters and receivers are not synchronized.
- *Unmodified cameras*. People use default cameras on mobile devices to receive data from string lights.

3 Data Modulation & Encoding

The data modulation and encoding aims at enabling cameras with limited capture rates (e.g, dozens of hertz) to decode data from a large number of small LEDs in a string light, and meanwhile avoiding light flickers of LEDs to human users. Observable high frequency light flickers are irritating and may cause stress and anxiety [6]. In StrLight, LEDs in a string light flash with the same frequency, denoted by $1/f_{LED}$. To transmit imperceptible data (i.e., without light flickers), LEDs need to work in at least $450Hz$ (i.e., $f_{LED} \geq 450$, experiment setups in §7.2). Intuitively, it is impossible for a receiver with a camera to decode such high-frequency LED data since the receiver's sampling rate (i.e., a camera's capture rate) does not meet the Nyquist sampling rate ($2\times450Hz$).

Instead of independently decoding data from each LED, which is impossible considering the Nyquist sampling theorem, we construct unique data pattern across all the LEDs in a string light and take into consideration of the statuses (*on*/*off*) of all LEDs on a captured image to decode the transmission data. StrLight transmits data in frames. A long message (i.e., the original data object) is first divided into multiple frames. The bits in one frame are sequentially represented by the LEDs in a string light.

We adopt On-Off Keying (OOK) [23], in which LED status *on*/*off* represents bit 1/0. OOK modulates a frame in single time-slot and thus does not have sampling rate requirement (the differences between CCD cameras and CMOS cameras are discussed in §6.2). A receiver only needs to capture instantaneous LED statuses to perform demodulation and recover the frame. However, OOK has a severe issue of light flickers, because multiple and consecutive *on* or *off* occur if a LED continuously transmits the same bit (i.e., '0' or '1'). The minimum LED working frequency that transmits 0-1 uniformly distributed data without causing light flickers is $450Hz$ in FSK (§7.2). By contrast, to avoid light flickers in OOK, LEDs are required to work in greater than $3kHz$ (i.e., $f_{LED} > 3000$, see §7.2). Since cameras need exposure time to capture objects, LEDs with such a high working frequency result in one captured image containing multiple time-slots of LEDs, i.e., frame mixtures. Even a camera's shutter speed is set to as low as $0.1ms$, the probability of mixed frames is larger than 30% (exposure time/timeslot duration, i.e., $0.1ms/0.333ms$). Please note the difference between camera's capture rate and camera's exposure time: a capture rate of $30Hz$ means that 30 images are captured/output per second, while exposure time of $0.1ms$ means that the camera senses the environment for the duration of $0.1ms$ to generate one image. A camera's capture rate is much less than the reciprocal of the camera's exposure time.

3.1 Alleviating Frame Mixtures

StrLight alleviates frame mixtures by reducing the LED working frequency required for imperceptible transmission. To distinguish with the original OOK scheme, we call our data modulation as

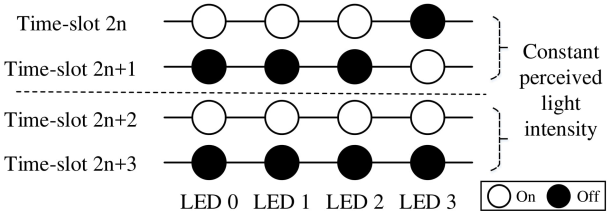
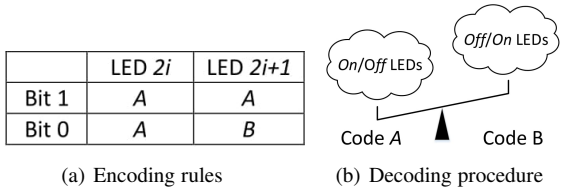


Fig. 2: StrLight applies on-off toggling modulation to reduce the LED working frequency required for imperceptible transmission from greater than $3kHz$ in OOK to $450Hz$, and thus alleviates frame mixtures



(a) Encoding rules (b) Decoding procedure

Fig. 3: StrLight proposes an imbalanced encoding to decipher LED statuses. (a) IE results in more code *A* than code *B* in any time-slot. (b) Receivers decipher the code that the LED status *on*/*off* represents by comparing the number of *on* LEDs and *off* LEDs in the image

On-Off Toggling Modulation (OOTM). OOTM binds two time-slots as one unit. The status *on* or *off* of each LED is toggled in these two time-slots. Figure 2 illustrates OOTM. In this example, time-slot $2n$ and $2n+1$ constitute a unit, whereas time-slot $2n+2$ and $2n+3$ is another unit. If these four LEDs need to transmit a frame of bits $\{1, 1, 1, 0\}$, we turn on the first three LEDs and turn off the last LED; in the next time-slot, we toggle the LED statuses, i.e., turning off the first three LEDs and turning on the last LED. Therefore, the cycle of each LED (two time-slots) has exactly one *on* status and one *off* status, which produces a constant light intensity for each LED. A constant light intensity is essential for avoiding light flickers to human eyes. Compared to OOK that does not have the periodicity of LED statuses, OOTM reduces the LED working frequency from $3kHz$ to $450Hz$ for imperceptible transmission (§7.2). When the LEDs are working in the frequency of $450Hz$, the cameras in our prototype system rarely have frame mixtures.

Because of the toggling operation in OOTM, each frame is transmitted two times in the cycle of two consecutive time-slots. OOTM results in similar LED controls as in Manchester encoding [17], in which 01/10 represents bit 0/1. However, StrLight is different from Manchester encoding based transmission schemes in that a camera's capture rate (e.g., $30Hz$) is much less than the LED's working frequency ($450Hz$), and thus receivers do not meet the Nyquist sampling theorem. With only the information in one time-slot (i.e., one captured image), the receivers do not know whether the LED status *on* and *off* represents bit 0 or 1. Taking the example in Figure 2, LED 0 is *on* in time-slot $2n$, where it is *off* in time-slot $2n+1$. In both time-slots, LED 0 transmits the same bit 1, but the LED statuses in these two time-slots are contrary.

3.2 Imbalanced Encoding & Decoding

To decipher the LED statuses in OOTM, we propose an encoding method, called *Imbalanced Encoding* (IE). IE transmits one bit using two neighbor LEDs, instead of one LED, and the number of bits in each frame is equal to half of the number of LEDs in the string light. Figure 3(a) gives the encoding rules of IE. Specifically, IE encodes bit 1 in $\{A, A\}$ and bit 0 in $\{A, B\}$. Thus, IE generates more code *A* than code *B* in a frame of any time-slot. Correspondingly, in any time-slot, the LEDs encoded in code *A* outnumber the LEDs in code *B* (that's why we call it imbalanced encoding). Please note that we use *A* and *B* to denote the two LED statuses (i.e., *on*/*off*). Whether the code *A* or the code *B* represents the LED status *on* or *off* depends on the number comparison of *on* LEDs and *off* LEDs in the image. In some images, LED *on* represents code *A* and LED *off* represents code *B*, while in other images, LED *on* represents code *B* and LED *off* represents code *A*.

Figure 3(b) illustrates the decoding procedure of IE. Receivers decipher the code that LED status *on*/*off* represents in each time-slot by comparing the number of *on* LEDs and *off* LEDs in the image. The status with more LEDs in the image represents code *A*, and the other status represents code *B*. For example, if a receiver captures the LEDs in the time-slot $2n$ in Figure 2, the status *on* represents code *A*, since the *on* LEDs outnumber the *off* LEDs; if the receiver captures the LEDs in the time-slot $2n+1$, the status *on* represents code *B*, as in this time-slot the *off* LEDs outnumber the *on* LEDs. In both time-slots, the decoded data are the same, i.e., $\{A, A, A, B\}$, corresponding to bits $\{1, 0\}$. Thus, the receiver in StrLight decodes the LED data by only capturing instantaneous statuses of LEDs in one time-slot. In other words, each captured image is self-contained and decodable.

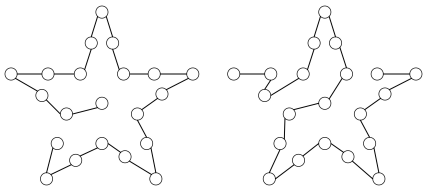
With the proposed OOTM modulation/demodulation and IE encoding/decoding, a receiver only requires to detect the status (*on* or *off*) of every LED in one captured image to decode the image and receive the data. To be clearly detected, each LED only needs to occupy a small region/dot in the images. In our prototype system, the occupancy area of each LED is as small as 3×3 pixels to be successfully detected. Consequently, StrLight can decode a large number of LEDs in a string light using low-cost cameras with small image resolutions (see §7.1 for more details).

The transmitter broadcasts data in the flashes of $450Hz$, corresponding to $450FPS$, where 225 frames are new due to the toggling operation in OOTM. On the other hand, the capture rate of a camera is much less than the LED flashing rate and thus the receiver only receives a small fraction of frames (i.e., capture rate/255). However, because the string light system repeatedly broadcasts the message that is composed of multiple frames, the receiver receives the whole message after several rounds of transmission. To the best of our knowledge, extracting messages in multiple rounds of reception is the only practical way considering that the transmitter and the receiver are not synchronized and that cameras' capture rate is not stable on mobile devices[1].

4 Data Representation

In this section, we assume that LEDs are working correctly and receivers capture all LEDs in the images; problems of broken or blocked LEDs are addressed in §5. After illustrating our topology discovery, we introduce the frame format in StrLight, which

1. Although camera capture rates of nominal $24Hz$, $30Hz$ and $60Hz$ are common, they are not stable. The intervals between two image captures are affected by system payload.



(a) (b)

Fig. 4: The topology of a string light cannot be discovered by merely locating the LEDs. (a) and (b) has same placements of LEDs but different topologies

facilitates the recovery of the bit sequences and strives to transmit more data in each frame.

4.1 Topology Discovery

The topology of a string light cannot be discovered by merely locating the LEDs. Figure 4 depicts two different topologies made by a same string light. To discover the topology, StrLight exploits two observations: (1) According to IE, two neighbor LEDs cannot both be encoded in code *B* in the same frame (i.e., the neighborhood constraint); (2) Neighbor LEDs are located near to each other because they are physically restricted by the connecting wire. We leverage the qualitative short distance (i.e., the relative distance) between neighbor LEDs for the topology discovery, and thus do not need to know the absolute value of the distance or require the distance to be same for identifying neighbors.

Encoding-based topology discovery. The topology discovery in StrLightworks in this way. Initially, the receiver constructs a neighbor set for each LED (target LED) by selecting the nearest M LEDs to each target LED. When a new frame is received, the receiver decodes data of each LED and removes LEDs from the neighbor set of the target LED if the LEDs and the target LED are both encoded in code *B* in the frame. As more frames are received, the receiver finally discovers the topology when the size of the neighbor set of each LED reduces to two (except for the two ends of the string light which may be far from each other but they can be easily identified).

Shifting the light pattern one LED forward per frame. A problem in the above topology discovery algorithm is that even-indexed LEDs are always encoded in code *A*, and thus they never violate the IE encoding constraints. To spread the encoding constraints to all LEDs, StrLight shifts one LED forward per frame. The two end LEDs of the string light are logically connected and thus a string light can be regarded as a closed loop. For example, the first frame is represented by {LED 0, $\underline{LED\ 1}$, LED 2, $\underline{LED\ 3}$}, thereby $LED1$ and $LED3$ have the constraints; the next frame is represented by {LED 1, $\underline{LED\ 2}$, LED 3, $\underline{LED\ 0}$}, and thus LED 0 and LED 2 have the constraints. With such a scheme of frame representation, all LEDs can gather enough observations of violations with incorrect candidate neighbors and discover the two neighbors.

Besides the neighborhood constraint, IE has another encoding constraint, which can be used to check the correctness of the identified topology. The two encoding constraints of IE are summarized below.

- *Constraint 1*: in a frame, LEDs that are encoded in code *B* are not neighbors;
- *Constraint 2*: in a frame, the encoding of 3^{rd} LEDs before and after LEDs that are encoded in code *B* are code *A*.

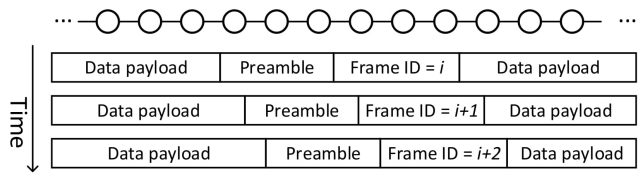


Fig. 5: StrLight formats each frame with a preamble field, a frame ID field and a data payload field. StrLight shifts one LED forward per frame

The encoding rules of IE can easily prove these two constraints, which are used in topology discovery. Because all LEDs discover their neighbors simultaneously and thus the speed of discovering the topology is not significantly affected by the number of LEDs in a string light. Since our topology discovery leverages the encodings of IE, the data distribution of the message affects the speed of topology discovery. From our experiments with 100 LEDs and 0-1 uniformly distributed data, a receiver needs to receive an average of about 40 frames (i.e., $40/30 \approx 1.3$ seconds if the camera's capture rate is $30FPS$). In addition to the high efficiency, the topology discovery has another desirable property: receivers discover the topology of a string light on-the-fly with normal data receptions; therefore, *the topology discovery in StrLight does not result in any overheads to data transmissions, i.e., no goodput reductions*.

4.2 Framing

After the topology discovery, StrLight decodes the bit sequence embedded in the LED statuses in the captured images. Since StrLight shifts one LED forward per frame, the LED representing the first bit in the frame is moving along the topology of the string light. We propose a frame structure to discover the LEDs representing the start of the received frames, which forms the decoded bits from all LEDs in the string light into a meaning frame.

Figure 5 illustrates the framing in StrLight. A frame in StrLight includes three key fields, i.e., a preamble, a frame ID, and a data payload (correspondingly preamble LEDs, frame ID LEDs and data LEDs). The preamble is used to indicate the start of each frame. For example, the preamble LEDs of frame 3 are two LEDs away from those of frame 1 (due to the shifting operation). By detecting the preamble locations from several frames, receivers identify the shifting direction.

Preamble Detection. Initially, we plan to assign a long sequence of code *A* to the preamble (code *B* cannot be adjacent due to encoding constraints) to distinguish the preamble LEDs from other fields of LEDs. However, if the frame ID and/or the data payload contains a long sequence of consecutive bit 1, conflicts occur because IE encodes bit 1 by {*A*, *A*}. To solve this problem, we improve IE to restrict the largest number of consecutive code *A* in the frame ID field and the data payload field. Therefore, we can use a slightly longer sequence of code *A* to distinguish the preamble LEDs.

Enhanced Imbalanced Encoding (EIE): If the number of consecutive bit 1 (denoted by L) in the frame ID field and the data payload field is greater than or equal to 4, EIE changes the encoding of i^{th} bit 1 from {A, A} to {B, A} with $3 \leq i \leq L-1, L \geq 4$.

Because we leverage the two encoding constraints of IE to discover the topology of string lights, EIE is required to obey these encoding constraints.

Proposition 1. *EIE preserves the encoding constraints, i.e., Constraint 1 and Constraint 2, of IE.*

Proof. The construction procedure of EIE proves this proposition. First, we consider the first index of bit 1 that can be changed from $\{A, A\}$ to $\{B, A\}$ without violating the encoding constraints of IE. If the first bit 1 is changed, the codes are $\{A, B, B, A\}$ (bit $\{0, \underline{1}\}$), violating Constraint 1. If the second bit 1 is changed, the codes are $\{A, B, A, A, B, A\}$ (bit $\{0, 1, \underline{1}\}$), violating Constraint 2. If the third bit 1 is changed, the codes are $\{A, B, A, A, A, B, A\}$ (bit $\{0, 1, 1, \underline{1}\}$), without violation. Then, we consider the last index of bit 1 that can be changed using the same line of reasoning. The bits 1 between the first index and the last index can be changed without violations because they are similar to the encoding of consecutive bit 0 in IE. □

Preamble Encodings. EIE restricts the largest number of consecutive code *A* to 7 (the bit patten of $\{0, 1, 1, 1, 0\}$). Therefore, StrLight assigns the preamble with $\{A, A, A, A, A, A, A, A, B\}$ (9 LEDs) or $\{A, A, A, A, A, A, A, A, A, B\}$ (10 LEDs) if the string light has an odd or even number of LEDs respectively. The number of remaining LEDs is even, which is required by EIE/IE. In addition to the ability of distinguishing the preamble field from other fields, EIE is superior to IE in the topology discovery since EIE generates more code *B* than IE. More code *B* means that more violations occur among LEDs that are not neighbors; thus discovering the topology of the string light is faster in EIE (see §7.3).

Frame ID Field. Frame ID bits are located after the preamble bits. Fixing the number of LEDs for the frame ID not only fails to support different lengths of messages (when the frame ID field is too short), but also results in throughput waste (when the frame ID field is too long). StrLight adapts the length of the frame ID field according to both the message size and the number of LEDs in the string light (denoted by N_{LED}). The minimum number of frame ID LEDs is determined by:

$$2^{\frac{n_{id}}{2}} \cdot \frac{Const - n_{id}}{2} = L_{msg} \quad (1)$$

where n_{id} is the number of LEDs for the frame ID; $Const$ is the total number of LEDs for the frame ID and the data payload, and equals to $N_{LED} - 9$ (N_{LED} is odd) or $N_{LED} - 10$ (N_{LED} is even); L_{msg} is the message size in bits.

To enable receivers to delimit the frame ID field and the data payload field, StrLight adds a flag bit to the end of the frame ID field. The flag bit has the same value as the last bit of the frame ID. Receivers obtain the length of the frame ID field by detecting two bits after the preamble that (1) the bit values are always same (2) the bit value changes frequently in received frames because it is the last bit of the frame ID.

Data Payload Field. The remaining LEDs are data LEDs. The message size, i.e., the number of frames to represent the whole message, is unknown to receivers. StrLight indicates the last frame of the message by assigning a unique data pattern in the data payload.

4.3 System Throughput

For a string light of N_{LED} LEDs, N_{id} LEDs are used for the frame ID and 9 or 10 (N_{LED} is odd or even) LEDs are used for the preamble (denoted by N_{prea}). Because two LEDs are used for transmitting one bit, each frame conveys a data capacity of $\frac{N_{LED} - N_{id} - N_{prea}}{2}$ bits. Every second, the string light transmits $f_{LED}/2$ new frames, while a receiver with a capture rate of f_{recv} receives f_{recv} frames. Therefore, the data rate (in *bps*) of StrLight is:

$$\text{Data rate} = \frac{N_{LED} - N_{id} - N_{prea}}{2} \cdot \min(f_{LED}/2, f_{recv}) \quad (2)$$

which is very close to the upper bound where each LED transmits 1 bit in every time-slot, i.e., $N_{LED} \cdot \min(f_{LED}, f_{recv})$. In data modulation/encoding level, OOTM/EIE achieves the same throughput as optimal FSK, with details in §7.2.

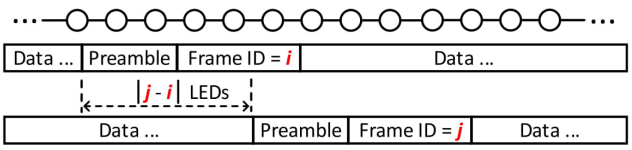


Fig. 6: StrLight leverages the frame structure to identify the indexes of both working LEDs and failed LEDs in the received frames

5 Fault Tolerance

Receivers cannot detect LEDs that are blocked or broken, because they are always dim in the captured images (algorithms to remove background light noises are given in §7.1). For simplicity, we call them missing/failed LEDs. The other LEDs are working LEDs. In the presence of missing/failed LEDs, a string light is divided into several sub-strings of working LEDs. The topology discovery fails to work in this case, because it either falsely connects or disconnects sub-strings of working LEDs. Thus, the indexes of LEDs in the frames decoded by receivers are incorrect.

The fault tolerance in StrLight has two unique challenges: (1) The number of LEDs in the string light is unknown to the receivers. For example, if a receiver detects 98 LEDs in the captured images, it cannot confirm whether the string light is composed of only 98 LEDs or some LEDs are missing; (2) Even if, by some means, the receiver knows the number of failed LEDs, it cannot infer the indexes of working LEDs in the frame.

5.1 Identification of Missing LEDs

StrLight leverages the frame structure to confirm sub-strings of working LEDs and to infer the number of failed LEDs between sub-strings of working LEDs. Each frame includes a preamble field and a frame ID field and StrLight shifts one LED forward per frame. Therefore, as Figure 6 illustrates, there are $|j - i|$ LEDs between the preamble LEDs of the ith frame (i.e., the frame ID = i) and the preamble LEDs of the jth frame.

If the receiver detects $|j - i|$ LEDs between the preambles of the frame i and the frame j, it confirms these $|j - i|$ LEDs as a sub-string of working LEDs. Otherwise, the receiver labels that there are $|j - i|$ LEDs between the preamble LEDs of the frame i and the frame j. The receiver further reduces the range of the failed LEDs among the $|j - i|$ LEDs by two criteria. (1) If $j - i > N_{prea} + N_{id}$, the number of failed LEDs is reduced to $j - i - N_{prea} - N_{id}$ LEDs between the end of the frame i's frame ID field and the start of the frame j's preamble field. A symmetric argument holds if

$i-j > N_{prea}+N_{id}$; (2) If the $|j-i|$ LEDs have overlapped sub-strings of LEDs that have been confirmed by other comparisons of frames, the range and the number of failed LEDs are reduced by these confirmed sub-strings. The receiver infers the indexes of both failed LEDs and working LEDs in the received frames when the preamble LEDs loop through the string light, i.e., in $\frac{N_{LED}}{f_{LED}/2}$ seconds ($< 0.5s$ when $N_{LED} = 100$, $f_{LED} = 450$).

Determining the number of LEDs in a string light. Since the working LEDs and the missing LEDs are identified, the number of LEDs in the string light is determined by simply adding the number of working LEDs and missing LEDs. The total number of LEDs can be further verified by looking into two frames with same preamble LEDs (refer to Figure 6). For example, frame 22 (ID field is 22) and frame 111 have same preamble locations, then the number of LEDs of the string light is 89 ($111 - 22$) or the divisors.

StrLight fails only if it cannot infer the indexes of LEDs because all of the sub-strings of working LEDs have length smaller than the sum of the preamble length and the frame ID length. In other words, none of the received frames has a complete preamble and a complete frame ID. Given the number of failed LEDs in a string light, the probability that StrLight fails is largest if the failed LEDs are scattered along the string light, i.e., the LEDs independently fail. In this case, it has the largest probability that none of the sub-strings of working LEDs has a length greater than the sum of the preamble length and the frame ID length. Proposition 2 gives the probability.

Proposition 2. *Given the number of failed LEDs in a string light, the probability of the worst case that StrLight fails to work is* $1 - Q^{(N_{LED})}_{S_0 S_{N_t}}$*, where the LED independently fails with a probability of* q*,* S_0 *and* S_{N_t} *is the first and the last Markov state respectively,* $Q^{n+m} = Q^n \cdot Q^m$*,* Q *is a square matrix with a dimension of* $N_{prea}+N_{id}+1$*:*

$$Q = \begin{bmatrix} q & 1-q & 0 & ... & 0 & 0 \\ q & 0 & 1-q & ... & 0 & 0 \\ ... & ... & ... & ... & ... & ... \\ q & 0 & 0 & ... & 1-q & 0 \\ q & 0 & 0 & ... & 0 & 1-q \\ 0 & 0 & 0 & ... & 0 & 1 \end{bmatrix} \quad (3)$$

Proof. The problem can be re-stated in this way. For an array of N_{LED} elements, where each element could be *F* (probability of q) or *W* (probability of $1-q$), we ask the probability that the length of consecutive *W* is always less than N_t (i.e., $N_{prea}+N_{id}$). This problem can be solved by a Markov chain with N_t+1 states, i.e., $S_0, S_1, ..., S_{N_t}$, where S_i means that the current length of the *W* string is i. Therefore, the probability of system breakdown is equivalent to the probability that S_{N_t} is never reached after N_{LED} transitions, i.e., $1 - Q^{(N_{LED})}_{S_0 S_{N_t}}$. □

Because S_{N_t} is the absorbing state of the Markov chain, it signifies that a string light having more LEDs is less probable to fail. The result is consistent with the intuitiveness, since a string light of more LEDs has a larger probability that the sub-strings of working LEDs have length greater than the sum of the preamble length and the frame ID length. Please note that human bodies or moving objects blocking the string light and resulting in n failed/missing LEDs has smaller probability of system failures than the probability given in Proposition 2 which is the worst case.

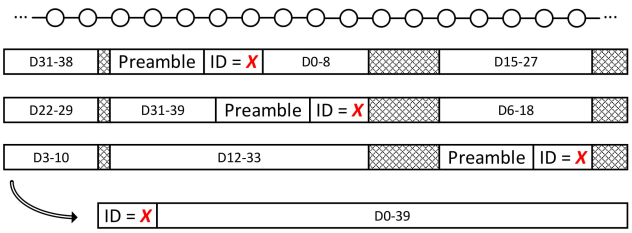


Fig. 7: StrLight restores the data of the failed LEDs by combining multiple received frames of the same ID

5.2 Data Recovery

After discovering the true topology of the string light, receivers apply the following two steps to recover missing bits and correct bit errors.

Step 1: restoring missing bits carried by the failed LEDs by combining multiple frames of the same ID. Because frames in StrLight are represented by all the LEDs in a string light, the error pattern of frames is different from conventional corrupted frames. In StrLight, each frame has missing data due to the failed LEDs and highly reliable data of the working LEDs. To restore the missing data in the frame, StrLight combines multiple received frames of same IDs. Figure 7 illustrates the combining procedure. Because StrLight shifts one LED per frame, multiple received frames of the same ID have different missing bits. For example, the first received frame X lacks bits 9-14, 28-30 and 39; the second frame X lacks bits 0-5, 19-21 and 30; and the third frame X lacks bits 0-2, 11, and 34-39. By combining these three frames, a complete frame X is restored. Instead of combining multiple frames, rateless codes are good candidates to encode message so that every frame contains extra information.

Step 2: recovering the message with forward error correction codes. The missing bits in some frames may not be recovered by combining a limited number of copies. Additionally, while frames of some IDs may be received multiple times, frames of some other IDs may never be received since the transmitter and the receiver are not synchronized. StrLight adds some FEC [17], [26] parity checking bits at the end of the message. Before recovering the message using FEC codes, StrLight assigns code *A* to the missing data of the combined frames and the frames with IDs that are never received. The reasons of assigning code *A* are two-folds: 1) it does not violate the encoding constraints of EIE; 2) statistically speaking, the probability that the missing data is code *A* is about $3/4$, greater than that of code *B*.

6 Implementation

6.1 System Components

Figure 8 shows the components of StrLight. The transmitter includes a string light, a LED driving board, and a controller. We have two types of receivers: a customized mobile device and COTS smartphone models. Table 1 tabulates the cost of each component.

String Light. We customize five string lights as shown in Figure 9. We shape them into a star, a heart, a letter, a polygon and a hat. Each string light has 80 to 100 LEDs. The distance between two adjacent LEDs is about $1.5cm$. Each LED only consumes $\sim 2.3mA$ (the voltage is $3.3V$) when it is powered on. Because of the toggling operation in OOTM, the average current is halved

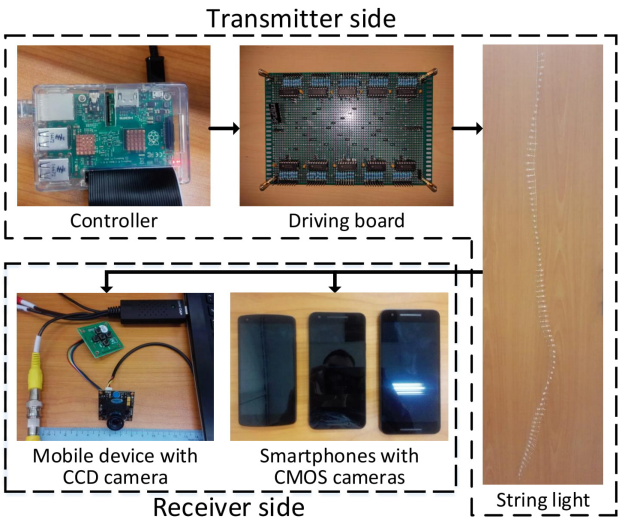


Fig. 8: System components

to $\sim 1.2mA$. Each LED costs $\$0.008$ and a string light of 100 LEDs only costs $\$0.80$.

LED Driving Board. We design LED driving boards to synchronize LEDs in a string light. We use $74HC595$ shift registers (unit price: $\$0.09$) to turn on/off LEDs. $74HC595$ is a serial-in parallel-out register. It has a storage register clock pin (STR pin) to output the stored eight-bit data in parallel. We connect STR pins of shift registers and thus all shift registers synchronously output the stored data. Each driving board has 10 $74HC595$ and can control 80 LEDs. We connect two driving boards to control string lights of more than 80 LEDs. A LED driving board costs $\$4.0$.

LED Controller. We use a Raspberry Pi II as the controller to instruct the LED driving board. We do not use Raspberry Pi II to directly power the LEDs because the sequential execution of operating systems causes time differences of controlling different LEDs. Instructing the LED driving board to synchronize 80 LEDs into statuses of next time-slot is very lightweight. It only takes Raspberry Pi II an average time of $2.7860\mu s$ (1000 traces, std=$3.1585\mu s$, max=$101\mu s$ and min=$2\mu s$).

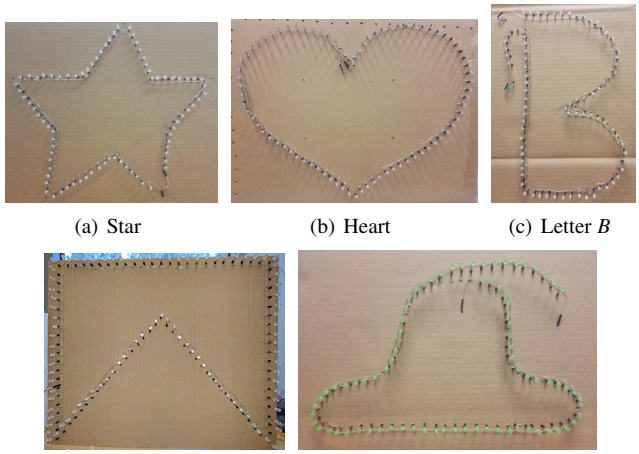
Receivers. We evaluate StrLight using two types of receivers, i.e., a customized mobile device with a Charge-Coupled Device (CCD) camera and COTS smartphone models with CMOS cameras.

(1) A customized mobile device with a CCD camera. We use a Sony Exview HAD CCD II camera with a focal length of $16mm$. We connect the camera to a laptop through an EasyCap video capture adapter. The camera has a global shutter speed of $1/60s - 1/10000s$ and a maximum image resolution of 976×494 pixels. The size of the camera with its controller is only $3.2cm \times 3.2cm$. The price of the camera with its controller is $\$28.16$. In market, there are smartphones equipped with CCD cameras, such as Sharp AQUOS SHOT SH-06A and Sharp AQUOS SHOT 933SH. We are trying to implement StrLight on them.

(2) Smartphone models with CMOS cameras. We use different smartphone models with CMOS cameras, e.g., LG Nexus 5, Huawei Nexus 6p, and LG Nexus 5x. They support a minimum exposure time of $1/8000s$ and output images of large resolutions (3264×2448 for LG Nexus 5, 4032×3024 for Huawei Nexus 6p and LG Nexus 5x). Due to the rolling shutter effect of CMOS cameras, one captured image may contain multiple time-slots of LED statuses.

TABLE 1: System cost

Transmitter Side	String Light (100 LEDs)	$0.80
	LED Driving Board	$4.00
	LED Controller (Raspberry Pi II)	$41.28
Receiver Side	CCD Camera (with Controller)	$28.16
	Video Capture Adapter (Easycap)	$9.44



(a) Star (b) Heart (c) Letter *B*

(d) Polygon (e) Hat

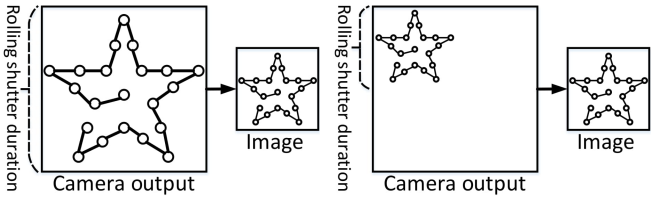
Fig. 9: We customize five string lights and form them into different shapes

Reduction of Rolling Shutter Durations. We propose a technique to reduce the rolling shutter durations of images from CMOS cameras by exploiting the large image resolutions of smartphone cameras. Camera sensors always output the largest images and the rolling shutter durations of these images are fixed. After receiving the images from the camera sensor, the operating system reduces the images to the target size specified by users. Therefore, systems can reduce the rolling shutter durations of the final images by the ratio of the largest image resolution to the final image resolution. By the default setting of cameras, however, smaller images have unchanged rolling shutter durations as shown in Figure 10(a). Instead, StrLight adjusts the camera's focal length to zoom off the objects to the target image resolution as shown in Figure 10(b). From users' perspective, no differences between Figure 10(a) and 10(b) are observed. This technique can be implemented by specifying the LENS_FOCAL_LENGTH field (zoom off the objects) and the SCALER_CROP_REGION field (crop the region) in CaptureRequest of Android camera API 2[2].

6.2 CCD Cameras vs. CMOS Cameras

Both types of cameras are widely used and have advantages and disadvantages. CMOS cameras have the rolling shutter effect, which means different rows of pixels in an image are exposed at different time. In contrast, CCD cameras are equipped with global shutters and thus all pixels in an image are captured at the same time. Therefore, CCD cameras are preferred to CMOS

2. Android camera API 2: https://developer.android.com/reference/android/hardware/camera2/package-summary.html



(a) Default (b) StrLight

Fig. 10: A technique to reduce the rolling shutter durations of images from CMOS cameras

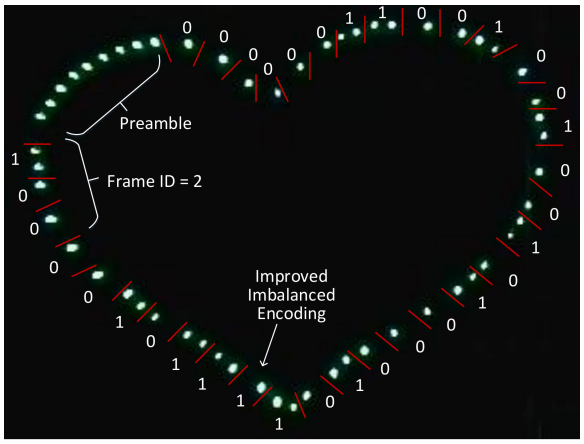


Fig. 11: A raw image captured in StrLight. Each LED only occupies 5×5 pixels in the image

cameras in capturing moving objects. In poor illuminative conditions, CCD cameras work better than CMOS cameras due to CCD cameras' inherent tolerance to bus capacitance variations and amplifier structures [27], [28]. Mainstream smartphones adopt CMOS cameras mainly because of their less energy consumptions than CCD cameras.

6.3 A Show Case

Figure 11 shows a typical image captured in StrLight. The image is taken from our customized mobile device under standard office light conditions ($\sim 300lux$). The string light is located $1.7m$ from our customized mobile device. The LEDs work in $450Hz$, without light flickers to human eyes. The shutter speed of the camera is set to $0.1ms$ to remove ambient light interference, which outputs images of only *on* LEDs at the time-slot (the example image is the raw image from the camera without image processing). The capture rate and the image resolution of the camera are set to $30Hz$ and 320×240. Each LED only occupies about 5×5 pixels in the image. For illustration, we manually label the bit of each LED. We can see that the LED statuses are clearly identified.

7 Evaluation

We use our customized mobile device of a CCD camera and different smartphone models of CMOS cameras to evaluate StrLight. Our customized mobile device fully supports our system, while CMOS-based receivers need mitigation of system requirement (the experiments with smartphones). By default, the experiment results are conducted using our customized mobile device.

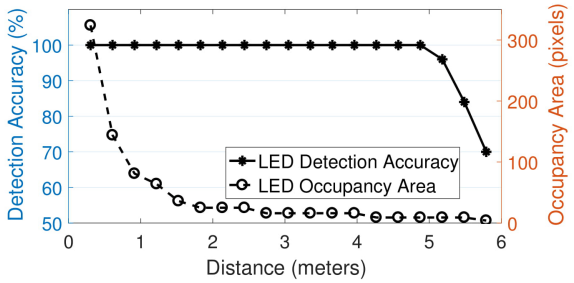


Fig. 12: The performance of detecting LED statuses. We plot the LED detection accuracy, as well as the occupancy area of each LED on the captured images

In the experiments with our customized mobile device, the LEDs work in $450Hz$ without light flickers to human eyes. The shutter speed of the camera is set to $0.1ms$. The capture rate and the image resolution of the camera are set to $30Hz$ and 320×240 pixels.

In the experiments with smartphones, the exposure time and the image resolution of the camera are set to $0.125ms$ $(1/8000s)$ and 1080×720. Since the flashing rate of the LEDs is $450Hz$, corresponding to a frame duration of $2.2ms$ $(1/450)$, the camera can easily capture an image without any frame mixture in the exposure time of $0.125ms$. However, even though the Android operating system allows us to set the exposure time to $0.125ms$, due to the hardware limitations of cameras on current smartphones, the real exposure time is much larger than $0.125ms$ and we experienced frame mixtures in our experiments. Conservatively, when conducting experiments with smartphones, we reduce the LED working frequency by $10\times$ simply to demonstrate our system.

7.1 Detection of LED Statuses

Receivers in StrLight use two techniques to remove ambient light noises and identify the LED statuses: (1) Short exposure time. The light intensities of LEDs are normally much higher than their surroundings. By setting a short exposure time, cameras remove most of ambient light noises and output images with LEDs that are turned on. Figure 11 shows a raw image from the camera; (2) Image processing. We first transform RGB images into gray-scale images. Then, to remove interference from strong light sources, such as sunshine and other lighting LEDs, we calculate the minimum and the maximum value of each pixel point from 20 consecutive images (less than one second for $30FPS$ capture rate). Since LEDs in StrLight are turned on and off repeatedly in a cycle of two time-slots, the pixel values in these LEDs change dramatically on multiple images. We set the pixel values to 0 if the difference of their maximum and minimum values are smaller than 100, i.e., light intensities of these pixel locations do not change obviously (i.e., background). Afterwards, we perform morphological dilation and erosion [29] on the images with a square structure whose width is 2 pixels. Last we calculate the maximum value of each image: the pixels on images are set to 255 if their values are greater than 1/3 of the maximum value; otherwise, they are set to 0.

LED Detection Accuracy. We evaluate the LED detection accuracy versus the distance between the string light and the receiver in normal indoor light conditions ($\sim 300lux$). Figure 12 shows the LED detection accuracy. The experiments are conducted

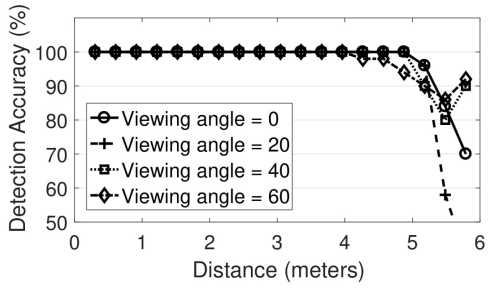


Fig. 13: The performance of detecting LED statuses versus viewing angles (in degrees)

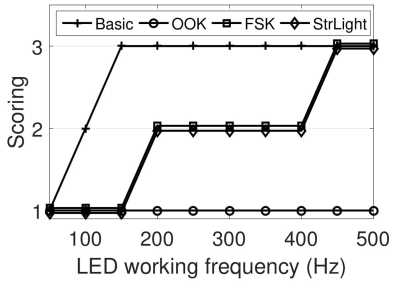


Fig. 14: Comparison of light flickers with different modulation schemes

using a LG Nexus 5x smartphone camera with ISO:100 and f:2.0. As we can see from the figure, the LED detection accuracy is 100% when the transmitter-to-receiver distance is smaller than $4.8m$. With larger distances, different LEDs are overlapped with each other in the captured images and the LED detection fails. We also plot the occupancy area of each LED in the captured images. Since StrLight only requires to detect whether a LED is turned on or off, each LED needs to occupy a small region in images to be successfully detected. For example, when the transmitter-to-receiver distance is $4.5m$, each LED only occupies 3×3 pixels in images, but they can still be successfully detected. Therefore, StrLight can decode data from hundreds of LEDs using a low-cost camera with very small image resolutions, whereas rolling shutter effect based schemes are only able to support a few big LEDs.

Viewing Angles. We evaluate the performance of detecting LED statuses when the receiver is not directly facing the string light. We conduct experiments with viewing angles of $20°$, $40°$, $60°$ and compare their performance with the direct-viewing case (i.e., viewing angle of $0°$). Figure 13 plots the LED detection accuracies versus viewing angles. The irregularity of results may stem from the fact that the LEDs in our prototype string light are not strictly pointing to one direction. Nonetheless, we have the following observation: within a given transmitter-to-receiver distance (e.g., 4 meters) of our prototype system, receivers have no LED detection errors, while beyond the distance, the LED detection error rises dramatically.

StrLight achieves excellent LED detection accuracies because it only needs to detect binary statuses of LEDs in each image, i.e., *on* or *off*. StrLight supports receivers with any capture rates, and the receiver's throughput is relate to its camera's capture rate. In our prototype system with a string light of 100 LEDs, we achieve throughput of $\sim 1260bps$ when we use 6 LEDs for the frame ID, 10 LEDs for the preamble and a camera capture rate of $30Hz$. If a camera with higher capture rate is used, say $240FPS$, the system throughput could be increased to $\sim 9kbps$ according to Equation (2). In the rest of this section, we focus on the cases that transmitter-to-receiver distance is within $4.5m$ (i.e., no LED detection errors). We will use simulations to evaluate StrLight when the large number of simulation traces give more accurate evaluation results. Please note that our simulation results are not biased since our LED detection accuracy is 100% within decent transmitter-to-receiver distances.

7.2 Data Modulation & Encoding

We compare the performance of StrLight (OOTM + IE/EIE) with OOK and FSK. The performance analysis is applicable to other data modulation/encoding schemes, e.g., PWM and PPM. We design an optimal FSK scheme, in which bit 1 and bit 0 is transmitted by [*on*, *off*, *on*, *off*] (LED flashing rate of $f_{LED}/2$) and [*on*, *on*, *off*, *off*] ($f_{LED}/4$) respectively. A basic scheme of repetitively turning LEDs on and off is used to benchmark the performance.

Ability against Light Flickers. We stand $1.5m$ away from a string light of 80 LEDs and rate light flickers with three scores: 1 - obtrusive, 2 - acceptable and 3 - imperceptible. Figure 14 depicts the scorings. It shows that StrLight and FSK have the same ability against light flickers. The results are consistent with the reasoning: the LED flashing rate is $f_{LED}/2$ when FSK transmits bit 1 or StrLight transmits same bits; the LED flashing rate is $f_{LED}/4$ when FSK transmits bit 0 or StrLight transmits different bits. Both StrLight and FSK achieve imperceptible transmission in $450Hz$. For OOK, however, we observe obtrusive light flickers even the LEDs work in $500Hz$. From our experiments, to achieve imperceptible transmission, OOK requires LEDs to work in greater than $3kHz$. Although other factors such as light color and light intensities affect light flickers, the performance comparison among StrLight, FSK and OOK remain the same. Considering the severe frame mixture issue of OOK, we exclude OOK from further performance comparisons with StrLight.

Data Capacity. FSK requires four time-slots and one LED to transmit one bit, whereas StrLight transmits one bit with two time-slots and two LEDs. Therefore, StrLight and FSK have the same data capacity of $f_{LED}/4$ bits per LED per time-slot.

Receiver's Sampling-Rate Requirement. To decode FSK data, if transmitters and receivers are not synchronized, the receivers need to sample in frequency at least two times of the LED's working frequency according to the Nyquist sampling theorem. Even transmitters and receivers are strictly synchronized, the receivers are required to sample in frequency no smaller than the LED's working frequency. In comparison, receivers in StrLight decode data with any sampling rate because every image is decodable by its own. Even receivers are required to receive all the data from LEDs, i.e., without data loss, StrLight is still superior to FSK because StrLight only requires receivers to sample every two time-slots rather than every time-slot as in FSK.

7.3 Topology Discovery

In the experiments of topology discovery, each LED selects nearest four LEDs as its candidate neighbor set. We consider three kinds of data distributions: (1) all bits 0; (2) 0-1 uniform, and (3) all bits 1. We collect 100 simulation traces for each parameter.

Figure 15 depicts the average and the maximum number of frames that are needed to discover the topologies of string lights

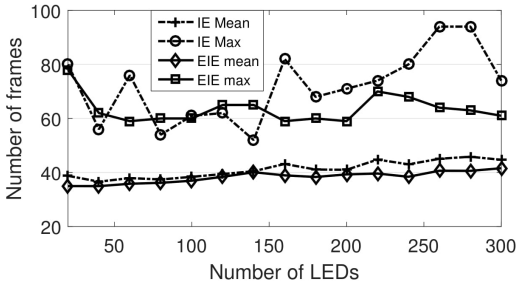
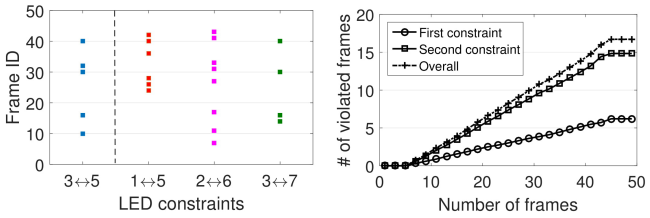


Fig. 15: Number of frames required to discover the topologies of string lights when data follows 0-1 uniform distribution

TABLE 2: Performance comparisons between EIE and IE in discovering the topologies

Data distribution	LED number	IE			EIE		
		Min	Mean	Max	Min	Mean	Max
All 0s	N=100	12	12	12	12	12	12
	N=200	12	12	12	12	12	12
0-1 uniform	N=100	25	39.7	64	22	36.6	64
	N=200	29	43.5	80	23	40.4	69
All 1s	N=100	∞	∞	∞	14	14	14
	N=200	∞	∞	∞	14	14	14



(a) One trace (b) Rate

Fig. 16: Both encoding constraints are useful for discovering the topologies of string lights. (a) One trace illustrates the violated frames (in square). (b) The rates of violating the two encoding constraints

when data are 0-1 uniformly distributed. For string lights of less than 300 LEDs, it shows that both EIE and IE require an average of less than 50 frames and a maximum number of less than 100 frames to discover the topologies. In other words, the receiver takes an average of less than 1.7 (50/30) seconds to discover the topology of string lights when the capture rate of its camera is $30Hz$.

Table 2 compares EIE and IE under different data distributions. Both EIE and IE only need 12 frames to discover the topology when data are all bits 0. When data follows 0-1 uniform distribution, EIE and IE require a few dozens of frames, with EIE slightly better than IE (i.e., less frames required). Compared to IE, EIE can be applied to any data distributions, while IE fails to work when data are all bits 1.

Constraint 2 in EIE/IE is useful for checking the correctness of the identified topology. We block one LED, say LED 4, and explore the violation of EIE constraints among adjacent six LEDs. Constraint 1 is represented by LED $3 \leftrightarrow 5$ and Constraint 2 is represented by LED $1 \leftrightarrow 5$, $2 \leftrightarrow 6$ and $3 \leftrightarrow 7$. We consider a string light of 50 LEDs in which 10 LEDs are used for the preamble, and

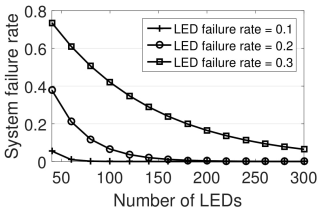


Fig. 17: The probability that StrLight fails to work versus the LED failure rate

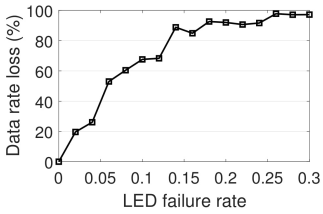


Fig. 18: The data rate loss of the system versus the LED failure rate

data distribution of 0-1 uniform. Figure 16(a) depicts one trace of violated frames (in square) in EIE. We can see that different frames may violate different constraints. Figure 16(b) depicts the rate of violating the constraints. It shows that Constraint 1 has smaller rate than Constraint 2. The overall rate of violating the encoding constraints is 0.42, which means that one violated frame is detected for an average of 2.4 received frames if the identified topology is incorrect.

7.4 Fault Tolerance

Figure 17 depicts the system breakdown rate of StrLight versus the LED failure rate. For example, if a string light has 300 LEDs and each LED fails at probability of 0.3, i.e., an average of 90 LEDs are blocked or broken, StrLight still works in the probability of 93.5%. This figure also verifies that StrLight becomes more robust when the string light is composed of more LEDs (Proposition 2).

Receivers discard frames without a complete preamble and a complete frame ID, resulting in data rate loss. We consider a string light of 100 LEDs in which 10 LEDs are used for the preamble. The experiment results are averaged from 100 simulation traces. Figure 18 shows that, although StrLight is robust against LED failures, the data rate decreases rapidly as the LED failure rate increases. For example, if a system requires the data rate loss to be less than 20%, it should guarantee that the LED failure rate is less than 0.02.

7.5 Data Rate without LED Failures

Equation (2) shows that the data rate is related to the number of LEDs in the string light, the length of the frame ID field, the LED working frequency and the receiver's capture rate. Figure 19 depicts the data rate of StrLight (with $N_{id} = 0$). The data rate increases linearly with more LEDs and higher capture rates.

7.6 Length of Frame ID Field

StrLight adopts the minimum number of frame ID LEDs as shown in Figure 20. The optimal number of frame ID LEDs changes with the number of LEDs in a string light and the message size. This figure also shows that fixing the length of the frame ID field lacks flexibility. For example, if a string light of 100 LEDs needs to transmit a message of $100kb$, the optimal number of frame ID LEDs is 26; if a string light of 300 LEDs require to transmit a message of $1kb$, 8 LEDs for the frame ID LEDs are optimal.

8 Discussion

StrLight is the first practical string light VLC system and thus has limitations/extensions.

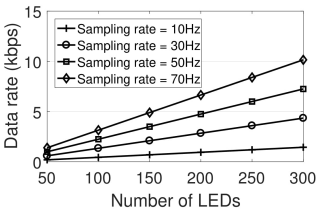


Fig. 19: The data rate without LED failures

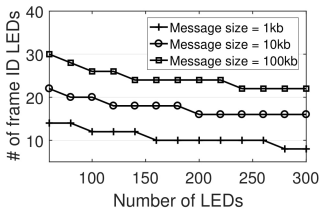


Fig. 20: The optimal number of frame ID LEDs

Benchmark Experiments. We evaluate StrLight using a large number of simulation traces. However, the performance is not biased since StrLight only needs to detect whether the LEDs are turned *on* or *off* to decode data and discover the topology. We build several prototypes of string light transmitters and test with different smartphone models and a customized mobile device as receivers. As Figure 11 shows, StrLight clearly identifies each LED status. Since StrLight only needs to detect the LED status (*on* or *off*), we expect that the performance of StrLight is not severely affected by, e.g., the user hand movements and the distance between transmitters and receivers. In addition, StrLight benefits from the techniques to combat the hand movements from existing LED based localization systems [18], [19].

System Robustness. We consider the failed LEDs that are blocked or broken, which are always dim on the captured images. Other failures or interference may exist from LEDs not from the string light, and un-controlled string LEDs that always stay *on* or glitter. Thanks to IE/EIE, these LEDs are bound to violate the encoding constraints with working LEDs. Therefore, they can be easily recognized. StrLight can also handle the case that some LEDs are temporally blocked by moving objects, since these LEDs are easier to be identified than the persistently blocked LEDs.

Future Works. StrLight may inspire various research points. (1) New modulations and encodings. How to effectively transmit data from hundreds of LEDs to low-cost receivers remains an open issue. In StrLight, we propose a novel scheme of a spatial data modulation and encoding to enable the data transmission. (2) Complex topologies of string lights. In Three-Dimensional (3D) space, string lights may have complex shapes on the images captured by receivers. As the first work, StrLight proposes a topology discovery of string lights that works effectively for the 2D representation of string lights. (3) Error corrections of uncertain bit indexes. To the best of our knowledge, no error correction mechanisms work when the received frames are represented by many sources in space. To combat the issue of erroneous indexes due to failed LEDs, StrLight infers the indexes of both working LEDs and failed LEDs in the received frames by leveraging our unique frame structure. (4) Full supports for CMOS cameras. StrLight fully supports CCD cameras. However, although we propose a modulation method that reduces the LED working frequency required for avoiding light flickers and a technique that decreases the rolling shutter durations of images taken from CMOS cameras, in current experiments, we need to reduce the LED working frequency to mitigate the frame mixtures of CMOS cameras. Further research is required to make StrLight fully support CMOS cameras.

9 Related Works

As the first practical string light system, StrLight differs from existing systems in its unique transmitter-to-receiver design, topology discovery and fault tolerance. Most of important related works have been analyzed in §2.

VLC systems have inspired a variety of applications, including but not limited to message broadcasting [30], localization/positioning [16], [18], [31], [32], [33], human gesture sensing [15], visual association/tagging [19] and collective visualization [34]. Advanced transmission techniques are also proposed, such as VLC-based backscattering [22], [35], reflection-based transmission [17], transmission using polarized light [32], [36]. StrLight extends existing VLC applications by proposing a string light based VLC system to broadcast messages.

10 Conclusions

StrLight is the first practical VLC system that leverages the widely-deployed string lights to disseminate data. We believe that StrLight can improve user experience, and thus useful for both personal (e.g., for amusement) and commercial usage (e.g., increasing revenue). To tackle the unique challenges of string light communication, we propose a new scheme of transmitter-to-receiver design, a topology discovery of string lights, and a fault tolerance against blocked or broken LEDs. The experiment results show that StrLight is an efficient and robust communication system. StrLight could inspire new research points, such as how to transmit data using a large number of LEDs, how to identify the topologies among these LEDs, and how to make system robust in the presence of the large number of LEDs that can be broken and blocked.

Acknowledgments

This research was supported in part by the China NSFC Grant 61472259, U1736207, Guangdong Natural Science Foundation 2017A030312008, Shenzhen Science and Technology Foundation (No. JCYJ20170302140946299JCYJ20170412110753954), Fok Ying-Tong Education Foundation for Young Teachers in the Higher Education Institutions of China (Grant No.161064)Guangdong Talent Project 2015TX01X111 and GDUPS (2015).

References

[1] Jimmy Schaeffler. *Digital Signage: Software, Networks, Advertising, and Displays: A Primer for Understanding the Business*. Focal Press, 2008.

[2] Research and Markets. Global digital signage market - forecasts and trends (2015-2020). Technical report, Cisco, HP, LG, 3M, Samsung & Panasonic, 2014.

[3] Angie Chandler, Joe Finney, Carl Lewis, and Alan Dix. Toward emergent technology for blended public displays. In *ACM UbiComp*, pages 101–104, 2009.

[4] Anja Thieme, Helene Steiner, David Sweeney, and Richard Banks. Body covers as digital display: a new material for expressions of body & self. In *ACM UbiComp*, pages 927–932, 2016.

[5] David Sweeney, Nicholas Chen, Steve Hodges, and Tobias Grosse-Puppendahl. Displays as a material: A route to making displays more pervasive. *IEEE Pervasive Computing*, 15(3):77–82, 2016.

[6] T Tashiro, S Kawanobe, T Kimura-Minoda, S Kohko, T Ishikawa, and M Ayama. Discomfort glare for white led light sources with different spatial arrangements. *Lighting Research & Technology*, 47(3):316–337, 2015.

[7] Wahhab Albazrqaoe, Jun Huang, and Guoliang Xing. Practical bluetooth traffic sniffing: Systems and privacy implications. In *ACM MobiCom*, pages 333–345, 2016.

[8] Yuanchun Li, Fanglin Chen, Toby Jia-Jun Li, Yao Guo, Gang Huang, Matthew Fredrikson, Yuvraj Agarwal, and Jason I. Hong. Privacy streams: Enabling transparency in personal data processing for mobile apps. In *ACM IMWUT*, pages 1–26, 2017.

[9] Ardalan Amiri Sani. Schrodin text: Strong protection of sensitive textual content of mobile applications. In *ACM MobiSys*, pages 197–210, 2017.

[10] Abhinav Mehrotra, Veljko Pejovic, Jo Vermeulen, Robert Hendley, and Micro Musolesi. My phone and me: Understanding people's receptivity to mobile notifications. In *ACM CHI*, pages 1021–1032, 2016.

[11] Krittika D Silva, Anastasios Noulas, Micro Musolesi, Cecilia Mascolo, and Max Sklar. If i build it, will they come? predicting new venue visitation patterns through mobility data. In *ACM SIGSPATIAL*, pages 1–4, 2017.

[12] Vamsi Talla, Mehrdad Hessar, Bryce Kellogg, Ali Najafi, Joshua R. Smith, and Shyamnath Gollakota. Lora backscatter: Enabling the vision of ubiquitous connectivity. In *ACM IMWUT*, pages 1–24, 2017.

[13] Yunfei Ma, Nicholas Selby, and Fadel Adib. Drone relays for battery-free networks. In *ACM SIGCOMM*, pages 335–347, 2017.

[14] Liqun Li, Pan Hu, Chunyi Peng, Guobin Shen, and Feng Zhao. Epsilon: A visible light based positioning system. In *USENIX NSDI*, pages 331–343, 2014.

[15] Tianxing Li, Chuankai An, Zhao Tian, Andrew T. Campbell, and Xia Zhou. Human sensing using visible light communication. In *ACM MobiCom*, pages 331–344, 2015.

[16] Chi Zhang and Xinyu Zhang. Litell: indoor localization using unmodified light fixtures. In *ACM MobiCom*, pages 230–242, 2016.

[17] Yanbing Yang, Jiangtian Nie, and Jun Luo. Reflexcode: Coding with superposed reflection light for led-camera communication. In *ACM MobiCom*, pages 193–205, 2017.

[18] Ye-Sheng Kuo, Pat Pannuto, Ko-Jen Hsiao, and Prabal Dutta. Luxapose: indoor positioning with mobile phones and visible light. In *ACM MobiCom*, pages 447–458, 2014.

[19] Hui-Yu Lee, Hao-Min Lin, Yu-Lin Wei, Hsin-I Wu, Hsin-Mu Tsai, and Kate Ching-Ju Lin. Rollinglight: Enabling line-of-sight light-to-camera communications. In *ACM MobiSys*, pages 193–205, 2015.

[20] Wenjun Hu, Jingshu Mao, Zihui Huang, Yiqing Xue, Junfeng She, Kaigui Bian, and Guobin Shen. Strata: Layered coding for scalable visual communication. In *ACM MobiCom*, pages 79–90, 2014.

[21] Shuyu Shi, Lin Chen, Wenjun Hu, and Marco Gruteser. Reading between lines: High-rate, non-intrusive visual codes within regular videos via implicitcode. In *ACM UbiComp*, pages 157–168, 2015.

[22] Xieyang Xu, Yang Shen, Junrui Yang, Chenren Xu, Guobin Shen, Guojun Chen, and Yunzhe Ni. Passivevlc: Enabling practical visible light backscatter communication for battery-free iot applications. In *ACM MobiCom*, pages 180–192, 2017.

[23] Parth H. Pathak, Xiaotao Feng, Pengfei Hu, and Prasant Mohapatra. Visible light communication, networking, and sensing: A survey, potential and challenges. *IEEE Communications Surveys & Tutorials*, 17(4):2047–2077, 2015.

[24] Carl Lewis, Angie Chandler, and Joe Finney. Where's my pixel? multi-view reconstruction of smart led display. In *ACM UbiComp*, pages 83–89, 2009.

[25] Naoya Isoyama, Tsutomu Terada, Junichi Akita, and Masahiko Tsukamoto. A method to control led blinking for position detection of devices on conductive clothes. In *ACM MoMM*, pages 123–130, 2011.

[26] Wan Du, Jansen Christian Liando, Huanle Zhang, and Mo Li. When pipelines meet fountain: Fast data dissemination in wireless sensor networks. In *ACM SenSys*, pages 365–378, 2015.

[27] Alen Lustica. Ccd and cmos image sensors in new hd cameras. In *IEEE ELMAR*, pages 133–136, 2011.

[28] Gerald C. Holst and Terrence S. Lomheim. *CMOS/CCD Sensors and Camera Systems*. SPIE Press, 2011.

[29] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Pearson, 2007.

[30] Yanbing Yang, Jie Hao, and Jun Luo. Ceilingtalk: Lightweight indoor broadcast through led-camera communication. *IEEE Transactions on Mobile Computing*, 16(12):3308–3319, 2017.

[31] Chi Zhang and Xinyu Zhang. Pulsar: Towards ubiquitous visible light localization. In *ACM MobiCom*, pages 208–221, 2017.

[32] Yu-Lin Wei, Chang-Jung Huang, Hsin-Mu Tsai, and Kate Ching-Ju Lin. Celli: Indoor positioning using polarized sweeping light beams. In *ACM MobiSys*, pages 136–147, 2017.

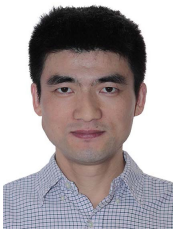
[33] Shilin Zhu and Xinyu Zhang. Enabling high-precision visible light localization in today's buildings. In *ACM MobiSys*, pages 96–108, 2017.

[34] Chungkuk Yoo, Inseok Hwang, Seungwoo Kang, Myung-Chul Kim, Seonghoon Kim, Daeyoung Won, Yu Gu, and Junehwa Song. Cardstunt as a service: Empowering a massively packed crowd for instant collective expressiveness. In *ACM MobiSys*, pages 121–135, 2017.

[35] Sihua Shao, Abdallah Khreishah, and Hany Elgala. Pixelated vlc-backscattering for self-charging indoor iot devices. *IEEE Photonics Technology Letters*, 29(2):177–180, 2017.

[36] Chun-Ling Chan, Hsin-Mu Tsai, and Kate Ching-Ju Lin. Poli: Long-range visible light communications using polarized light intensity modulation. In *ACM MobiSys*, pages 109–120, 2017.

Huanle Zhang received the B.S. degree in Communication Engineering from Hangzhou Dianzi University, China, in 2011, and the M.S. degree in Communication and Information System from University of Electronic Science and Technology of China, in 2014. He was a Project Officer with the School of Computer Science and Engineering, Nanyang Technological University, Singapore, from 2014-2016. He is currently a Ph.D. student with the Department of Computer Science, University of California, Davis. His research interests include mobile systems, cellular communications, virtual reality, artificial intelligence, and Internet of Things. He is a member of the IEEE.



Wan Du is currently an Assistant Professor in Computer Science and Engineering at the University of California, Merced. Before moving to Merced, Dr. Du had worked as a Research Fellow in the School of Computer Science and Engineering, Nanyang Technological University, Singapore, from December 2011 to August 2017. He received the B.E. and M.S. degrees in Electrical Engineering from Beihang University, China, in 2005 and 2008, respectively, and the Ph.D. degree in Electronics from the University of Lyon (cole centrale de Lyon), France, in 2011. His research interests include the Internet of Things, cyber-physical system, distributed networking systems, and mobile systems. He is a member of the IEEE and ACM.

Mo Li received the BS degree in computer science and technology from Tsinghua University, Beijing, China, in 2004, and the PhD degree in computer science and engineering from Hong Kong University of Science and Technology, in 2009. He is a Nanyang assistant professor with the Computer Science Division, School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include distributed systems, wireless sensor networks, pervasive computing and RFID, and wireless and mobile systems. He is a member of the IEEE and ACM.

Kaishun Wu received the BS degree in computer science and technology from Tsinghua University, Beijing, China, in 2004, and the PhD degree in computer science and engineering from Hong Kong University of Science and Technology, in 2009. He is a Nanyang assistant professor with the Computer Science Division, School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include distributed systems, wireless sensor networks, pervasive computing and RFID, and wireless and mobile systems. He is a member of the IEEE and ACM.

Prasant Mohapatra is a Professor in the Department of Computer Science and is serving as the Dean and Vice-Provost of Graduate Studies at University of California, Davis. He was the Editor-in-Chief of the IEEE Transactions on Mobile Computing. He has served on the editorial board of the IEEE Transactions on Computers, IEEE Transactions on Mobile Computing, IEEE Transaction on Parallel and Distributed Systems, ACM WINET, and Ad Hoc Networks. He is a Fellow of the IEEE and a Fellow of AAAS.