5-2021

DeepLight: Robust and unobtrusive real-time screen-camera communication for real-world displays

Vu Huy TRAN

Gihan JAYATILAKA

Ashwin ASHOK

Archan MISRA

DeepLight: Robust & Unobtrusive Real-time Screen-Camera Communication for Real-World Displays

Vu Tran
Singapore Management University

Gihan Jayatilaka
University of Peradeniya

Ashwin Ashok
Georgia State University

Archan Misra
Singapore Management University

ABSTRACT

The paper introduces a novel, holistic approach for robust Screen-Camera Communication (SCC), where video content on a screen is visually encoded in a human-imperceptible fashion and decoded by a camera capturing images of such screen content. We first show that state-of-the-art SCC techniques have two key limitations for in-the-wild deployment: (a) the decoding accuracy drops rapidly under even modest screen extraction errors from the captured images, and (b) they generate perceptible flickers on common refresh rate screens even with minimal modulation of pixel intensity. To overcome these challenges, we introduce *DeepLight*, a system that incorporates machine learning (ML) models in the decoding pipeline to achieve humanly-imperceptible, moderately high SCC rates under diverse real-world conditions. *DeepLight's* key innovation is the design of a Deep Neural Network (DNN) based decoder that collectively decodes all the bits spatially encoded in a display frame, without attempting to precisely isolate the pixels associated with each encoded bit. In addition, *DeepLight* supports imperceptible encoding by selectively modulating the intensity of only the Blue channel, and provides reasonably accurate screen extraction (IoU values $\geq$ 83%) by using state-of-the-art object detection DNN pipelines. We show that a fully functional *DeepLight* system is able to robustly achieve high decoding accuracy (frame error rate < 0.2) and moderately-high data goodput ($\geq$0.95 Kbps) using a human-held smartphone camera, even over larger screen-camera distances ($\approx$ 2m).

CCS CONCEPTS

• **Human-centered computing** → **Ubiquitous and mobile computing**; • **Computer systems organization** → **Embedded and cyber-physical systems**.

KEYWORDS

Screen-camera communication, Visible light communication, Imperceptible, Deep neural networks, Perception, Flicker-free

1 INTRODUCTION

Screen-Camera communication (SCC) is an emerging variant of visible light communication (VLC), where the light source is a large, individually-controllable, multi-pixel LED screen and the receiver decodes the visually-encoded information captured by a camera sensor. SCC has evinced strong interest [1–5], especially as an enabler of ubiquitous computing applications for subliminal machine-machine communication. SCC's unique challenge is to sustain high throughput while simultaneously ensuring that the visual encoding remains *imperceptible*, when applied to *unregulated* and dynamic video content displayed on regular displays. Unfortunately, while recently-proposed SCC techniques can achieve very high communication rates under controlled conditions at short screen-camera distances, we shall show that these techniques are **not robust enough** to support diverse real-world artefacts, such as (a) longer (1.5-2m) screen-camera distances and non-frontal viewing angles, (b) lack of a-priori knowledge of precise screen coordinates, and (c) continuous perturbation of camera views due to human movement.

In this paper, we describe *DeepLight*, a novel SCC system that *employs machine learning (ML) based inferencing*
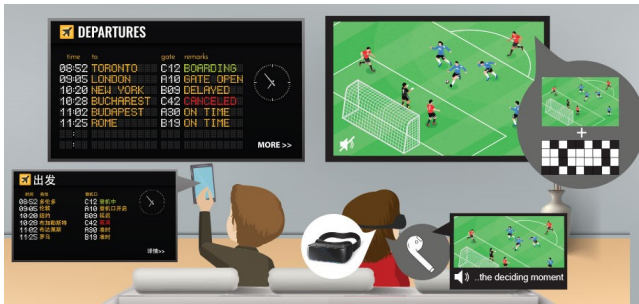
Figure 1: An SCC based audio & text communication use-case scenario in an airline lounge, where SCC is used to encode and transmit a stream of supplementary information accompanying the display screens' content.

techniques in the decoding pipeline to achieve a degree of robustness, to the artefacts mentioned above, that has hitherto been unattainable. To our knowledge, *DeepLight* is the first system to demonstrate *low-flicker* SCC for real-world scenarios involving *commodity* (screen refresh rates of 60Hz) public display screens and handheld smartphone cameras, where: (a) an individual user points the camera sensor towards the screen from different viewing angles and distances, (b) the screen content can include both still images and dynamic video streams, and (c) the screens are located in environments with different ambient lighting levels and backgrounds (e.g., fast moving crowds vs. walls) and have attributes that are *a priori* unknown to the mobile camera.

Figure 1 illustrates a couple of exemplar *DeepLight*-based ubiquitous computing applications (including one that we have built and deployed). Here, SCC is used to visually and subliminally encode the following supplementary information: (a) User U1 watches the sports feed with a see-through head-mounted, camera-equipped display: the wearable *DeepLight* software is able to decode the SCC-encoded *audio* commentary, in real time, and then output this via Bluetooth-enabled earphones; (b) user U2 points a camera-equipped smartphone at the flight information display: the mobile-embedded *DeepLight* decodes the *textual* commentary(encoded in multiple languages via SCC) and overlays the information in U2's native language.

Key Technical Challenges & Innovations of *DeepLight*[1]: Our work provides the following breakthroughs to address several key challenges:

- *A CNN-based Collective-Bit-Decoder to Make Decoding Robust to Imprecise Screen Detection:* Current SCC decoding techniques operate by effectively *explicitly* dividing the estimated screen, in a grid-like fashion, into a set of constituent cells and then independently inferring the encoded bit of each such cell. Consequently, these SCC techniques suffer a dramatic jump in bit-error-rate (BER) even under

[1]Source code & demo video available at: https://deeplightscc.github.io/

relatively modest screen extraction errors–e.g., the BER of HiLight increases by 12% even if the detected screen is offset by only 10% of a cell. To support robust decoding of transmitted bits without needing a perfectly-extracted screen, *DeepLight* uses a novel *Collective Bit Decoder*, a convolutional neural network (CNN) model that *collectively* infers the entire set of bits in a display frame (instead of trying to explicitly isolate the pixels for each individual bit). *DeepLight's* use of CNN-based models for decoding represents a significant departure from prior signal processing-based approaches, and is particularly effective in extracting the bit information (encoded as intensity changes in the blue channel, as explained shortly) from a multi-scale analysis of the image content. Through extensive experiments, we show that this CNN is able to achieve frame error rates (FER) $\leq 1.7\%$ (an improvement of almost 15% over prior techniques), even when the IoU (Intersection over Union) of screen detection is modest ($IoU_{screen} = 96\%, IoU_{cell} = 68\%$).

- *Ensure Accurate Real-world Screen Detection & SCC Operation without Needing Explicit Screen Markers:* Prior work typically utilizes either static cameras (with pre-calibrated screen boundaries) or special markers to delineate the screen edges. Such precise extraction of screen boundaries is not practical, even with state-of-the-art DNN models, for real-world displays. For example, Figure 2 shows a representative set of real-world images containing display screens in public spaces, together with (1) the detected screen border estimated by a common edge analysis algorithm (Canny & Hough transform), and (2) bounding boxes for 'display' objects identified by the popular SSD [6] object detector. We experimentally observe that the edge analysis algorithm fails to extract screens when the content or the background includes many edges. Consequently, the average IOU is considerably low (38%) over a set of 266 representative images of TV screens in public spaces. Similarly, SSD fails to detect screens where the content and the background contain large objects, and thus can correctly detect only 73% of screens in the same test set. Though SSD is not designed to accurately estimate the border of objects (only bounding box), it still achieves, on average, an IOU of 45% (higher than the edge analysis algorithm). This suggests that a properly designed deep-neural network may support sufficiently accurate screen extractions. To improve screen extraction accuracy, we develop a DNN-based pipeline that can extract the coordinates of the display screen, without assuming any external markers or a-priori knowledge, with sufficiently high accuracy (average IoU=93% in indoor environments) to permit robust decoding by our Collective Bit Decoder. Moreover, this pipeline is lightweight enough to achieve screen extraction latency of < 40 msecs/frame on an iPhone 11Pro device.

Figure 2: Screen extraction performance of *(1st row)*: Edge analysis algorithm, and *(2nd row)*: SSD-MobileNet, with photos of representative displays. The ground-truth borders are drawn in "dotted red" lines while the detected borders/bounding boxes are drawn in "solid green" lines.

Figure 3: SCC Fundamentals. Bits are spatially encoded on the display frame for decoding by a camera, while staying imperceptible to human eye.

- *Overcome High Flicker Perception at Common Frame Rates:* Most of the proposed techniques (e.g., [3–5, 7]) involve modulation of the pixel intensity *along all 3 (RGB) channels*. While such RGB modulation is often imperceptible at very high display frame rates (e.g., in excess of 120 fps) used by specialized displays, even minimal intensity changes (±1 out of 255) become noticeable at the typically lower refresh rates (30-60 fps) commonly used in public displays. To tackle this problem, *DeepLight* encodes the bits via intensity changes (Δ units) only in the *Blue*('B') channel of display content. Through experimental studies, with 17 users and viewing distances of 1-2 meters, we show that this modulation scheme causes significantly less flickers (mean opinion scores (MOS) is $\approx 20\%$ higher) even when the intensity is modulated by a larger amount ($\Delta = 3$).

Besides demonstrating these core component technologies, we develop an end-to-end functional prototype of *DeepLight* (deployed as a self-contained iOS App that implements the subtitle overlay application described before), which additionally uses an appropriate error-correcting source coding scheme to support high data goodput across diverse real-world settings. Via extensive empirical experimentation, we show that *DeepLight* can significantly outperform state-of-the-art competitive baselines, achieving both high data goodput (1-1.2 Kbps) and low flicker perception (MOS values of 4 and higher) for larger screen-camera distances (up to 2 meters) and diverse viewing angles ($0 - \pm30°$).

2 BACKGROUND AND RELATED WORK

The conceptual idea of SCC is illustrated in Figure 3. At the display screen or *transmitter* end, the display content image (or frames of video) pixels are logically divided into a rectangular grid, where *each cell in the grid represents at least one encoded bit*. The data bits are modulated via subtle changes in corresponding pixels by an amount of Δ. A practical SCC system must ensure that the camera (i.e., the *receiver* device) is able to decode this original embedded information, even under distortions such as non-linear camera optics, effects of ambient lighting and motion artifacts (caused by the mobility
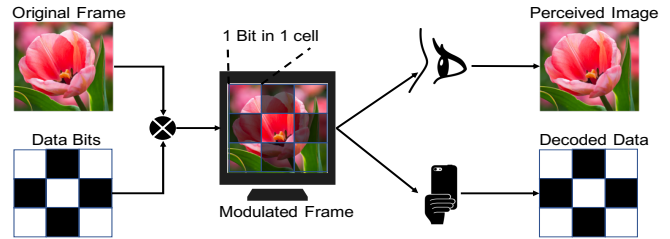
of the camera-embedded device). While decoding accuracy can be enhanced by using a larger Δ, this results in the human eyes perceiving a flickering chessboard-like pattern. Thus, the central challenge in SCC is to achieve higher throughput while avoiding perceptible flickers. Prior work has focused primarily on designing reliable and efficient (a) encoding mechanisms to *improve throughput* [3, 5, 8], and/or (b) embedding mechanisms to *reduce flicker perception* [1, 4, 7]. Broadly, the encoder schemes follow two approaches: (a) Manchester encoding, or (b) Frequency encoding.

Manchester encoding based techniques. The Manchester encoding technique modulates a single bit over two consecutive frames by adjusting the pixel intensity with an amount of $\pm\Delta$. This approach leverages the fact that human eyes average out content changes at rates beyond ≈ 50Hz [9–12]; accordingly, if the screen refresh rate R_S is greater than 100 Hz, the flickers are perceptually canceled out. Therefore, Manchester encoding is visually imperceptible only at high display rates (unless Δ is very small) and is ineffective for commonly-used displays with $R_S = 50/60$Hz. VRCode [1] applies Manchester coding on color channel (in CIE color space), and supports only invariant data (e.g, QR Code). In contrast, Inframe++ [3] supports time-variant data and visual content by encoding data on brightness channel using a novel spatial pattern. However, the variable data transmission causes flickers as it creates additional harmonics. TextureCode [7] modulates only the parts of the frame with higher texture to reduce flickers. Chromacode [5] uses LAB [13] color space, and applies adaptive texture encoding for full frame modulation. All these approaches require precise screen extraction (which is unrealistic in practice), as each bit is decoded individually, based on a predefined, grid-based partitioning of this captured screen.

Frequency Modulation based techniques. HiLight [4] encodes a bit "1" and bit "0" as variations at 30Hz and 20Hz over 16 frames (thereby reducing the overall throughput) respectively. HiLight adjusts the screen's brightness (commonly referred to as α channel in graphics) using low α to suppress flicker. Though HiLight does not use special patterns to estimate screen border, it requires turning the screen OFF/ON for precise screen extraction.

Barcode Communication. Extensive prior work [14, 15] has explored the process of smartphone-based 2D barcode or QR code scanning, which may be viewed as a rudimentary form of SCC. Several approaches ([16–18]) have investigated the transmission of *time-varying QR-Code* data as a form of high bit-rate SCC, with MegaLight [19] using a Random Forest-based ML model to support robust decoding. However, all of these techniques consider the QR code as the *primary* display content and do not attempt to make the transmission imperceptible to human viewers.

Deep Learning based Watermarking & Steganography. The idea of 'hiding' information into image content is borrowed from the well-known concept of watermarking or steganography [20, 21]. Most classical steganographic techniques, which either modify the least significant bits (LSBs) in pixels [22] or use DCT [23] , have very low data rates (bits/frame). While deep learning has recently been explored as a technique for enhancing capacity [24], the techniques fundamentally assume a lossless transfer of the encoded image to the receiver. The recent work by Wengrowski and Dana [25] is the first to utilize a DNN-based framework for screen-camera steganography. However, this work focuses on photographic steganography (where the encoded image must be extracted precisely) and requires the learning of a *per-display* transfer model (that precludes seamless in-the-wild adoption).

3 LIMITATIONS IN EXISTING METHODS

Before presenting the design of *DeepLight*, we shall experimentally quantify two key limitations of prior SCC methods, focusing on, (a) human *perception of flickers* emanating from the screens due to embedded modulation, and (b) vulnerability of the system performance to *imprecise screen extraction*.

3.1 High Flicker Perceptibility

It is essential for SCC is to preserve the visual quality of modulated video frames. Prior works rely on high frequency (120FPS or higher) displays to suppress flickers caused by modulation schemes [3, 5, 7, 8], whereas commercial displays in public spaces usually have 30-60 FPS refresh rate. It turns out that even minimal modulation ($\Delta = \pm 1$) causes significant flickers at 60FPS, making current modulation approaches, which change all 3 channels (via brightness channel [3, 5, 7, 8] or alpha-blend [4]), inapplicable to common displays. Fortunately, humans are known to be relatively less sensitive to Blue light (compared to Red & Green) [11, 26], due to the presence of fewer Blue-sensitive photo-receptors in our eyes [27]. In this paper, we show that, by modulating the data on Blue channel only, our system achieves the highest (and considerably higher) mean opinion score (MOS) compared to prior works, while achieving a higher goodput at practical viewing distances ($\geq 1m$).

To validate this effect in SCC, we conduct a pilot study on flicker perception with modulations on different channels. Figure 4 (left) plots the Mean Opinion Score (MOS) and its variation, across 12 users, who rated the perceived quality of a total of 42 video clips (each clip is 10 secs long), displayed on a 60FPS 25" monitor, and viewed from $d = 1$ meter. The videos were encoded using the widely-used Manchester coding technique [3, 5, 7], with the barest possible pixel modulation amplitude ($\Delta = \pm 1$, out of a total range of 255). Each user was exposed to different treatments: (a) where the modulation is performed on brightness channel (under HSV or LAB color spaces) or a single color (R, G or B) channel, and (b) where the modulation magnitude is higher ($\Delta = \pm 2$). We see that, even with the least possible modulation amplitude ($\Delta = \pm 1$) on brightness channel or Green channel, the MOS score is significantly lower (≤ 3.3). Though Red channel achieves fairly high MOS (avg = 4.36) with low modulation amplitude ($\Delta = \pm 1$), it performs much worse (avg = 3.58) with higher Δ. However, the MOS is consistently high if only the Blue channel is modulated (avg = 4.57 and 4.36 with $\Delta = \pm 1$ and ± 2 respectively), validating the physiological fact. This suggests that performing Blue-channel-based modulation *may* considerably enhance the imperceptibility of SCC for larger displays, at common display rates.

3.2 Vulnerability to Imprecise Screen Extraction

Prior works assume perfect screen extraction achieved using either explicit visible locators (e.g. InFrame++ [3], QR-Code [14]) or adding an explicit set of black and white strips next to the screen border (e.g., ImplicitCode [8], Chromacode [5]). Neither of these is compatible with our goal of in-the-wild SCC using unmodified displays. Such precise screen extraction is necessary because each bit is decoded *independently* using an explicit partitioning of the rectangular screen into a $M \times N$ grid. Accordingly, a lack of precise extraction causes a specific cell to either include superfluous pixels (from neighboring cells) or excluding constituent pixels (which are incorrectly mapped to a neighboring cell). Unfortunately, even practically "precise" screen extractions may easily include a deviation. Even if this deviation is small compared with screen size, it may be significant compared with cell size. For example, the top left image in Figure 2 clearly shows an offset between the estimated lower edge (using Canny & Hough transform) and the ground-truth (even if the screen extraction is considered "precise"), which affects the cells near the lower edge severely. However, as illustrated in Figure 2, accurate estimation of an active screen's borders is not trivial under real-world artifacts. The fact that a display shows dynamic contents, with a plethora of textures and colors, makes "precise" estimation of the screen border impractical. Indeed, past SCC prototypes have used custom screen
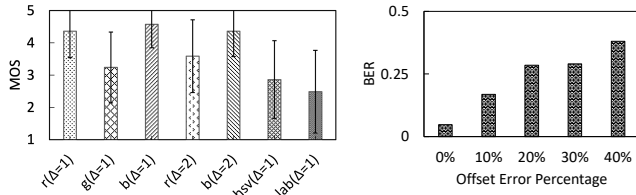
Figure 4: (Left) Flicker score of intensity-modulated video clips (distance= 1 meter, $\Delta = \{1, 2\}$). (Right) *HiLight* BER under varying screen extraction error.

locators [3, 5, 8] or utilized out-of-band techniques to tackle this challenge. As an example of these issues, HiLight [4] documented the significant loss of accuracy for viewing distances larger than 1 meter, due to the higher screen extraction error. To tackle this, HiLight's edge detection mechanism (published code in [28]) explicitly requires the user to turn the screen OFF/ON at the beginning of each SCC session, an approach that is untenable for public venues.

To quantify this issue, Figure 4 (right) plots our empirically observed bit-error-rate (BER) values with HiLight (viewing distance $d = 0.7m$ meter), for varying degree of synthetically generated extraction errors (expressed as a percentage of the size of a single cell of a 10×10 grid). We study HiLight as it is the only prior work whose code we can obtain publicly, and it applies the same approach of explicit grid splitting used in other works. We see that the decoding accuracy drops significantly, even when the extraction error is as low as 10% (a paltry error of 19×11 pixels in our HD display). At 20% offset error, the BER is $\approx 29\%$, an increase of $\approx 25\%$ in BER compared with the perfect screen extraction. We shall demonstrate (Section 4.2) how the use of ML pipelines in *DeepLight* allows us to overcome these limitations.

3.3 DeepLight Design Goals

To achieve the operational vision of a robust and practical SCC system, *DeepLight* should ideally achieve the following objectives:

• *Support Relatively-High SCC Data Rates: DeepLight* use cases extend well beyond the transmission of relatively static content (e.g., QR-codes), and thus require at least modestly-high data goodput. In particular, with advanced encoding [29] human speech transmission (for user U1 in Figure 1 earlier) can be achieved with bit rates of 1-2 Kbps; whereas for transmission of subtitles (for user U2 Figure 1 earlier), the maximum acceptable rate is 200 WPM (word-per-minute), effectively translating into a target rate of ~ 0.2 Kbps (assuming the support of 8 languages). To support such SCC applications, *DeepLight* **should support data rates of at least** 0.5 **Kbps, and ideally** 2+ **Kbps.**

• *Support Variations in Viewing Distances, Angles & Backgrounds:* In public spaces, viewers may use their mobile
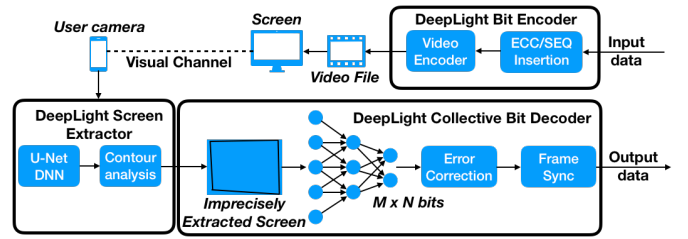

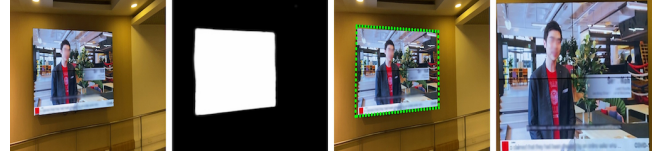**Figure 5: *DeepLight*: Overall System Architecture**

Figure 6: Four stages of screen extractor. L to R: Input, Unet output, Post-segmentation Screen (dotted green bounding box), Transformed Screen

devices to decode the SCC display from a variety of viewing distances and viewing angles. For robust and practical SCC, *DeepLight* **should ideally support reasonably-high decoding goodput (across a variety of spaces such as office cabins, shopping malls and airports) for viewing distances of up to** $1.5 − 2$ **meters and viewing angles of up to** $60°$.

• *Work across a Range of Static & Dynamic Screen Content: DeepLight* **should ensure that its visual encoding remains imperceptible across real-world display content**, ranging from dynamic (e.g., the live feed of a sporting event, where the scene changes once every 10-20 ms) to relatively static (e.g., the airport information feed is likely to refresh only once every 5-10 seconds).

• *Work with Imprecise Screen Capture:* Based on the limitations presented above, *DeepLight* **must be capable of delineating the boundary/contour of diversely-shaped display screens (within a captured image) with reasonable accuracy, and subsequently decoding the embedded content under imperfect screen extraction**.

4 *DEEPLIGHT* SYSTEM & COMPONENTS

To tackle the limitations enumerated above, we propose a new SCC system, called *DeepLight*, that is based on three key ideas: (a) on the encoding side, the information bits are encoded via modest intensity changes of only the display content's Blue channel, (b) the use of a state-of-the-art DNN on the decoder to extract a display screen *fairly accurately* from a captured image, under a variety of ambient context and (c) the design of a new DNN model that provides holistic/joint decoding of multiple bits in a display image, even when the screen extraction is imperfect.

Figure 5 illustrates the high-level architecture of *DeepLight*. On the encoder side, data bits are packed and embedded into
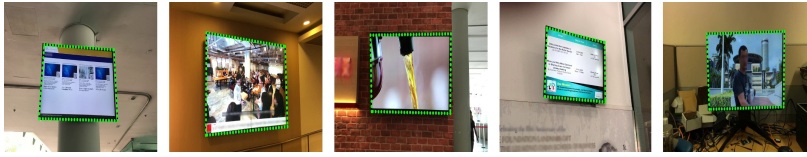
Figure 7: Applying Screen Extractor to different screens & backgrounds, spanning office and public spaces. Dotted green bounding box is the estimated screen.

Figure 8: *DeepLight* **CNN Decoder & Structure of data frame.**

video frames before being shown on a screen. A camera sensor (e.g., belonging to a user's smartphone) then captures images at f_C FPS, with the display screen assumed to be visible (but at an arbitrary angle and position) within each captured image. On the decoder side, an initial enhanced DNN-based *Screen Extractor* component operates on each image to isolate and extract the pixels estimated to be part of the *DeepLight*-encoded screen. A series of L such consecutively "extracted screens" is then forwarded to the *Bit Decoder* (this represents the most innovative aspect of *DeepLight*), which uses another DNN to *collectively* reconstruct the transmitted bits.

4.1 Bit Encoder

To embed data into video frames, *DeepLight* bit encoder encodes the information data bits by mapping a set of bits to a $M \times N$ rectangular grid, where each cell corresponds to a block of $B_r \times B_c$ pixels of the image (video frame) to be encoded. In each cell, the bit encoder applies a Manchester coding strategy by performing opposing intensity modulation of the *Blue* channel of each pixel in the cell by a value of $\pm\Delta$ across two consecutive (F+, F-) frames. By extensive trial and error with different Δ choices (detailed in SectionSection 7), we find Δ values of 2 or 3 to be reasonable, representing a compromise between our desire for lower decoding error (high Δ) and visual imperceptibility (low Δ). These encoded image frames are then displayed on the screen at its original operational frame rate F_d.

4.2 Screen Extractor

At the decoder side, the first step is to localize the *DeepLight* screen in a camera frame. We first evaluate how well the classical edge analysis approaches support screen extraction. Though many techniques have been proposed to detect rectangular objects (similar to a screen) such as license plates [30] & QR-Codes [14, 15], "precise" screen extraction is tricky for two reasons: (1) there may be parts of the content next to the screen border that have similar color with the screen border; (2) the screen contents and the background usually contain lots of textures. We experimentally observe that classical edge analysis approaches work well with uniformly colored screens, but perform quite poorly (examples shown in Figure 2) on our corpus, where screens have high-texture content or occupy a small portion of the overall image.
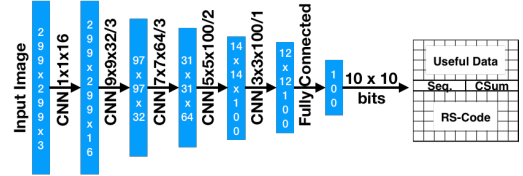
To locate and extract the screen in a camera frame, *DeepLight* borrows from state-of-the-art DNN-based object segmentation and detection techniques to isolate screens under a wide variety of settings. The Screen Extractor consists of three cascaded stages (illustrated in Figure 6), with additional smoothing and thresholding operations to improve the noise tolerance:

(Step 1) A DNN-based pixelwise segmentation pipeline, inspired by Unet architecture [31], that first assigns a probability for every pixel belonging to a screen. This 2D probability distribution is dilated (smoothened) by a uniform kernel and thresholded to assign binary labels ('screen' vs. 'no-screen') to each pixel.

(Step 2) The segmented image is then passed through a contour-based screen localizer [32] that ensures that the identified set of 'screen' pixels adhere to certain natural shape constraints (e.g., the screen must be a quadrilateral).

(Step 3) Finally, the set of pixels representing the screen is run through a linear-interpolation based perspective transform, which effectively warps the oblique view of the screen (when the screen is captured at non-perpendicular viewing angles) to a normalized 'full-frontal view'.

This DNN is trained on a dataset (illustrative examples, which include the bounding boxes estimated by our technique, are provided in Figure 7) consisting of multiple screen types, backgrounds (indoor and outdoor) and camera setups (handheld and tripod). We used multiple techniques to curate a comprehensive training set of representative screen images: (a) using Web search to retrieve over 387 relevant images of public displays from locations such as malls, train stations; (b) personally visiting over 20 real-world locations across different geographies to collect 35 images of digital screens. The images are captured under different lighting conditions and crowd levels and also encompass a variety of screen sizes. Further, to improve the robustness, standard *data augmentation* techniques (involving rotational and translational transformations to the training data) are used to synthetically enhance the training data corpus.

Note that the Screen Extractor operates on each individual image (independent of whether the frame includes actual Manchester-encoded or transition frames), and is thus independent of screen refresh or camera frame rates. However, in

our eventual mobile-based implementation, we shall see that we will prefer to run the Extractor *intermittently*, on selected frames, to better balance the processing throughput and the screen extraction accuracy.

4.3 DNN-Based Collective Bit Decoder

We next present a Convolutional Neural-Network (CNN) model that can extract the subtle temporal variation of Blue channel intensity across each individual cell, where each cell consists of a subset (e.g., 100×100) pixels. Under practical conditions, one can expect a screen extraction error of N pixels (N >= 0) in horizontal and/or vertical directions (e.g., a translation of 30 pixels in both directions). Clearly, the grid-based decoding techniques suffer from cross channel interference if $N > 0$. When $N \geq 0.3 \times cell\_size$ (e.g., 30 pixels) in both directions, lower than 50% of a transmitter cell is captured in the corresponding receiver cell (SNR < 0dB). To cope with these screen extraction errors, we propose to use a DNN model to infer input bits collectively based on the entire extracted screen. Figure 8 shows our proposed DNN model structure which includes a 5-layer CNN and a fully connected layer. Our choice of a CNN-based approach is motivated by the remarkable capability of CNNs in vision-based pattern recognition tasks [33–36]. In particular, CNN structure is well suited for this type of problem. A CNN practically comprises a large number of filters on the input images with different functionalities (e.g., one favours the bright areas, the other favours the darker areas). It extracts the essential "features", which are highly correlated to the "bits" values, through several filter layers. Essentially, each feature is computed from a receptive field which may easily cover many cells.

The input to our CNN model is an image with $L = 3$ channels, with each channel corresponding to the pixel value of the Blue channel of L consecutive frames. These L frames effectively capture both the rising (F+) and falling (F-) sequence of Manchester coded images. (While $L = 3$ *is used as an exemplar, based on the Nyquist assumption that the camera sampling rate is twice the display frame rate, we shall generalize this approach* in Section 8). The first layer is a 1×1 convolutional layer, that is used to extract different *temporal* relationships, for each pixel, across the L consecutive frames. Several additional convolutional layers then extract higher-level and multi-scale features in the input data, corresponding to a hybrid mix of spatial (across the neighboring pixels of a single image) and temporal (across the corresponding pixels of multiple frames) features. We also apply Batch Normalization [37] to improve the training of the convolutional layers. A finally fully-connected layer, with a Sigmoidal activation function, is used to regress the value of each bit, such that the final output of the CNN is a vector of $M \times N$ bits. We emphasize this key property essential for robustness: **each**
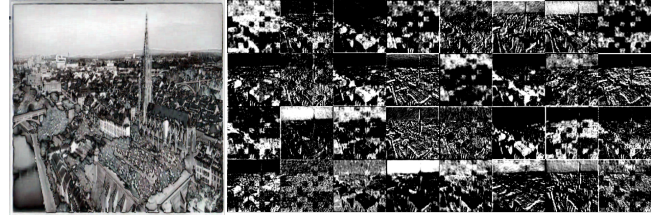

Figure 9: Left: input frame comprises the blue channel of the last 3 frames (with screen extraction error=($\approx 20\%$ of a cell). Right: Intermediate feature map at the 2nd convolutional layer, showing the outlines of the data grid.

of the final $M \times N$ bits of output is derived from the holistic processing of L *entire* frames, rather than via isolated processing of a predesignated subset of pixels. As an illustration of this, Figure 9 shows the output of the second convolutional layer in the decoder model, computed over an imperfectly-extracted representative screen image. We can see that the grid-like separation is already observable at this second layer, indicating that the DNN is able to perform decoding without explicit *a-priori* knowledge of individual cell boundaries.

4.3.1 Training the CNN. To train *DeepLight's* CNN-based decoder, we use a binary cross-entropy (BCE) loss function [38], which attempts to maximize the number of correctly matched bits—i.e., it tries to minimize the bit error rate (BER). The training dataset creation first involves the generation of display content (still images, video snapshots, movie content) video frames embedded with random bits using Manchester coding. To easily generate ground-truth labels that distinguish individual positive ($F+$) and negative ($F-$) Manchester-encoded frames pairs and that allows precise extraction of screen coordinates via color-based segmentation [32]), we insert a dummy landmark frame (all pixels colored RED, $(R, G, B) = (255, 0, 0)$) after every 10 Manchester frame pairs $((F+, F-))$. The frames are encoded into an uncompressed video format and displayed on a G-sync desktop monitor at 60 FPS (we choose this monitor and frame rate to avoid the frame tearing effect [39]). The displayed encoded content is recorded using a mobile phone camera (iPhone X) at 120 FPS and at 720p HD resolution (1280×720 image pixels). Each pair of Manchester frames maps to 4 distinct frames captured by the camera: a fully positive frame (F_+), a transitional frame (from positive to negative–called F_+^t), a fully negative frame (F_-), and a transitional frame from the negative frame to the next pair (F_-^t). We consider the Blue channel images of the 3 frames that include the Manchester pair ($\{F_+, F_+^t, F_-\}$) to represent 1 training image for the representative display content. To make the *DeepLight* CNN model robust to different screen errors, we train it with artificial randomized errors in the screen coordinates (relative to the ground truth screen coordinates), with errors within the range $\pm 0.5 \times cellwidth$

on both the *X* and *Y* axes. Each extracted region of screen pixels with artificial errors is resized to produce a 299×299 pixel training image. Our final annotated dataset consists of 22,500 samples observed by a fixed/stationary camera and 25,200 samples captured by a hand-held camera. Overall, we observe a validation accuracy of $\approx 92.0\%$ (fixed) and $\approx 90.0\%$ (handheld) after 100 training epochs.

4.3.2 Handling Larger Grid Sizes. As mentioned previously, the *DeepLight* decoder effectively uses a CNN-based regressor to simultaneously estimate the entire set of $M \times N$ bits encoded in a single Manchester image pair. A larger value of $M \times N$ (classifier outputs) increases the theoretical SCC throughput (no. of bits/image), but requires a more complex DNN model with higher training and runtime computational overhead. The canonical *DeepLight* CNN decoder is built and trained for a grid size of (M =10 x N =10). Larger grid sizes can then be handled by simply duplicating the canonical model–e.g., if a *DeepLight* encoder uses a 20×20 grid, the decoder applies the 10×10 CNN model four times, each on a quadrant of the input frame. However, a larger grid size means fewer pixels/cell. Consequently, the increased throughput (bits/frame) from larger grid sizes might be counteracted by higher decoding error– we shall empirically study the consequent tradeoff between grid sizes and overall *goodput* in Section 6.2.5.

5 PROTOTYPE IMPLEMENTATION

5.1 Bit Encoder

To facilitate the easy experimentation with *DeepLight*, we implement an offline encoding process (in Python), where the intensity-modulated video files are generated in advance and then played back on the display screen. *As will be explained shortly, the encoding process involves an initial preprocessing step where the data bits are grouped into blocks and then augmented by an error correcting code.* We use OpenCV to generate intensity modulated JPEG images (with data embedded) from original video frames and store the resulting video file in .AVI format. Unless otherwise mentioned, the source videos have an original frame rate of 24 to 30 FPS and an HD resolution (1920×1080). We take the first 100 frames (approx. 3 secs) from each video to create a test set. After modulation, each encoded video clip has a frame rate of 60 FPS (pair of Manchester encoded frames).

5.2 Bit Decoder

DeepLight was implemented in two separate phases to support different facets of experimental studies:

Phase 1: We conducted an offline study on the robustness of *DeepLight* to support the rapid evaluation of a large amount of video data. As we use a diverse set of test videos (which were not provided during the model training phase), the resulted test data contains $\approx$2TB of high quality video. We first implemented *DeepLight* in a desktop environment where the decoding operation was offloaded from the smartphone. In this study, we use an iPhone X (running iOS 13.3) and a 64GB RAM desktop with a Core-i7-7700 CPU running at 3.6 GHz and a Nvidia GTX-1080Ti GPU card. The iPhone X operates its camera at 120 FPS, and captures full-HD quality videos (resolution=1920×1080). Captured videos are stored as ".MOV" files, and transmitted to the desktop computer for processing.

Phase 2: We implemented *DeepLight* on a mobile phone to study how *DeepLight* supports real-time processing when running *entirely* on a mobile device. We use an iPhone 11 Pro (running iOS v13.6), with Apple A13 chipset and 4GB of RAM, for the mobile implementation of *DeepLight*. We port our system to Objective-C and use CoreML [43] for the neural network models, using the same 32-bit floating point precision for both desktop and mobile versions. We shall analyze the processing time of each component in *DeepLight*, and the resulting design choices (e.g., the periodic execution of the ScreenNet on a smaller set of image frames) to make such a real-time mobile implementation possible.

5.3 Implementing Error Correction

To build a fully-functional *DeepLight* decoder that supports reliable communication in spite of inevitable decoding errors, we integrate a Reed-Solomon (RS) [40] error-correcting source code. Figure 8 shows the overall frame structure using a representative 10×10 bit encoded frame. As RS code with N redundant symbols can correct a maximum of N/2 erroneous symbols, we augment it with a checksum code and a sequence number to detect frames that are falsely decoded by the RS decoder. These augmented fields (RS code, checksum and sequence number) can be adjusted without needing to retrain the core neural network.

Checksum: The 5-bit BSD [41] checksum is computed over the data bits and sequence number, and is used to identify any frame that the RS error correction fails to correct.

Sequence number: The sequence number is incremented on each data frame, and is used by the decoder to identify and eliminate duplicate (e.g., transition frames) and incorrectly-decoded frames. Given the frame rate (F_d) and the camera rate F_c, two consecutively numbered data frames should have a separation of at least $Sep = \lfloor \frac{2*F_c}{F_d} \rfloor$ image frames. The decoder logic eliminates potential false positives by verifying that two frames differing by x in their sequence numbers are at least $x * Sep$ frames apart. We empirically found that this combination of mechanisms provides highly robust decoding (all incorrectly decoded frames are identified and filtered out) as long as the percentage of redundant RS-code bits is $\geq 40\%$.
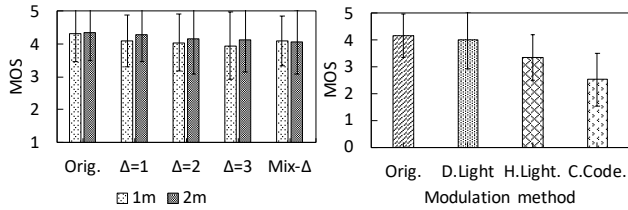
Figure 10: User study MOS for: different Δ choices (left) and different encoding techniques (right).

6 MICROBENCHMARK EVALUATION

Our evaluation (performed with IRB approval from respective institutions) utilizes the following key performance metrics:

(1) Flicker perception, defined using the subjective Mean-Opinion-Score (**MOS**) measure and quantitative Peak-Signal-to-Noise-Ratio (**PSNR**) metric, computed as $PSNR = 10 \times \log_{10}(R^2/MSE)$, where R is the maximum feasible pixel value (for n bit resolution of pixels $R = 2^n - 1$), and $MSE = \sum_{1,1}^{M,N}(I(m,n) - \hat{I}(m,n))^2/(M \times N)$, where $I(m,n)$ and $\hat{I}(m,n)$ are the original and modified pixel intensities respectively.

(2) Frame Error Rate (FER), defined as the fraction of transmitted encoded data frames that are correctly decoded by the receiver. Given the use of RS decoding, a frame is either correct (if the RS code can correct the errors) or completely incorrect (even if some bits in the frame may be correct). We report FER rather than BER (bit-error-rate) as FER counts only recoverable data, while BER includes both recoverable and unrecoverable data.

(3) Goodput (GP), defined as the total number of *data bits*, per second, correctly decoded by the *DeepLight* decoder. GP is computed as $GP = D \times F_r \times (1 - FER)$, where D is the useful data in each frame ($D = [(10 \times 10) \times (1 - RS\_ecc\_rate)] - 10$ in our system), $F_r = 0.5F_d$ is the data frame rate (0.5 scaling due to manchester coding), F_d is the display frame rate, and FER is the frame error rate.

6.1 Flicker perception

To evaluate the imperceptibility of *DeepLight* encoding, we conducted user-studies where individual participants were each shown a total of 84 ten-second long video clips on a 25" display screen. The videos were randomized and included original videos with no encoding, encoded videos with different combinations of Δ (1, 2, 3 and a mix of 2 and 3), and videos encoded using Chromacode [5] and HiLight [4] techniques (we implemented the encoder based on the description in the publication [4, 5]). Users had no background knowledge of encoding and were asked to only view and rate each video clip. The experiments were conducted at distances 1m and 2m, at a reasonably comfortable (to user) frontal viewing angle, in an indoor office room environment with white ceiling lighting. The ratings use a 5-point MOS scale:{(1): Very

unpleasant; (2): It's bad; (3): It could be better; (4): It's good; (5): Cannot differentiate from the original video}. The experiment involved 17 participants (10 males, 7 females), with age ranging [18,32] (median=24 yrs). One of the users has *astigmatism*, 3 have farsightedness, 6 have no eye conditions, and the rest have shortsightedness.

6.1.1 DeepLight Flicker perception Results. Figure 10 (left) plots the average MOS values (and the std. deviation) for the original (unmodified) videos, as well as *DeepLight*-encoded videos with *B*-channel intensity modulation values $\Delta = \{1, 2, 3\}$. We also experimented with a *Mix-Δ* strategy, where we use $\Delta = 2$ if the avg. *B*-channel intensity is higher than a threshold (30), and $\Delta = 3$ otherwise. We see that (a) the MOS scores stay almost unchanged (compared to the original) as long as $\Delta \leq 2$, and (b) as expected, the flicker perception diminishes (MOS increases) when the viewing distance increases. The Mix-Δ strategy has the same perception score as a fixed value of $\Delta = 2$. However, Mix-Δ was seen experimentally to provide higher decoding accuracy, and is used as the *default* encoder, unless otherwise specified.

6.1.2 Comparative Evaluation of Flicker Perception. From the MOS reported in Figure 10 (right), we can see that *DeepLight's* score is significantly higher (avg=4), compared to both HiLight (avg=3.36) and Chromacode (avg=2.53), thereby validating *DeepLight's* selective use of *B*-channel intensity modulation. The result is also validated by the higher average PSNR value (PSNR=42) for *DeepLight*, compared to PSNR=41 for HiLight and PSNR=36 for ChromaCode. These results demonstrate that *DeepLight* provides better viewer experience at typical display rates (60 FPS), compared to these prior state-of-the-art techniques.

6.1.3 DeepLight Flicker Perception with Compressed Videos. Highly compressed videos are relevant when a screen shows online videos. Consequently, the videos may be compressed at different ratios depending on the network condition. Note that *DeepLight* encodes the data after the video has been compressed, so the modulation amplitude (Δ) is not affected by compression. In this experiment, we evaluate the perceived quality at 3 bitrate ratios including: $\{75\%, 50\%, 25\%\} \times original\_bitrate$. To avoid the bias caused by the lower quality of compressed videos, we measure MOS of both un-encoded and encoded videos for each compression ratios. Figure 11 shows that the MOS drop is fairly similar with un-compressed and moderately compressed videos (bitrate ratio $\geq$ 50%). However, when the videos are highly compressed (bitrate ratio = 25%), the MOS drop is significantly lower and the standard deviation is considerably higher. This suggests that it is difficult to notice the flickers caused by *DeepLight* modulation among temporal and spatial artifacts caused by compression.

Δ	Distance [m]				
	1.00	1.25	1.50	1.75	2.00
1	20.0/**0.96**	34.0/**0.79**	46.0/**0.65**	62.0/**0.45**	73.0/**0.32**
2	0.3/**1.2**	1.8/**1.18**	5.9/**1.13**	12.3/**1.05**	14.6/**1.02**
3	0.2/**1.2**	0.2/**1.2**	0.6/**1.19**	5.5/**1.14**	7.1/**1.12**
Mix	0.4/**1.2**	0.9/**1.19**	2.1/**1.18**	7.6/**1.11**	11.7/**1.06**

Table 1: *DeepLight* **FER(%)/GP(Kbps)** vs. (d, Δ)

Rate	Distance [m]				
	1.00	1.25	1.50	1.75	2.00
0.6	0.2/**0.90**	0.3/**0.90**	1.0/**0.89**	4.9/**0.86**	7.2/**0.84**
0.5	0.4/**1.2**	0.9/**1.19**	2.1/**1.18**	7.6/**1.11**	11.7/**1.06**
0.4	0.7/**1.49**	2.1/**1.47**	6.5/**1.4**	12.6/**1.31**	18.4/**1.22**
0.3	4.0/**1.73**	4.5/**1.72**	12.9/**1.57**	22.2/**1.4**	28.7/**1.29**

Table 2: *DeepLight* **FER(%)/GP(Kbps)** vs. $(d, \textbf{rate})$

Viewing Angle	Distance [m]		
	1.00	1.50	2.00
0°	0.4/**1.2**	2.1/**1.18**	11.7/**1.06**
15°	0.3/**1.2**	1.2/**1.19**	9.3/**1.09**
30°	0.2/**1.2**	1.9/**1.18**	6.3/**1.12**
45°	5.6/**1.13**	14.8/**1.02**	30.6/**0.83**
60°	76.1/**0.29**	94.5/**0.07**	100.0/**0.0**

Table 3: *DeepLight* **FER(%)/GP(Kbps)** vs. (θ, d)

Error Type	Screen detection error [cell size]			
	10%	20%	30%	40%
SHF	1.6/**1.18**	2.8/**1.17**	9.6/**1.09**	65.6/**0.41**
EXP	1.2/**1.19**	1.6/**1.18**	2.4/**1.17**	5.9/**1.13**
SHR	1.8/**1.18**	2.4/**1.17**	3.3/**1.16**	6.9/**1.12**
ROT	1.6/**1.18**	1.9/**1.18**	2.8/**1.17**	5.7/**1.13**

Table 4: *DeepLight* **FER(%)/GP(Kbps)** vs. Screen Extraction Error

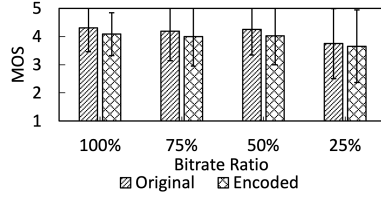| Lighting	FER/GP
eFL+BG	2.8/**1.17**
eFL+LED	1.4/**1.18**
iFL+LED	6.5/**1.12**
iFL	9.7/**1.08**

Table 5: *DeepLight* **FER(%)/GP(Kbps)** vs. lighting types.

Figure 11: Effects of compression on perceived quality.

6.2 FER and Goodput under different conditions.

We next evaluate the decoding performance of *DeepLight* under a variety of instrumented settings. All the experiments were conducted in a meeting room which has a mix of fluorescent lamps and LEDs. We evaluate *DeepLight* using 50 different short video clips (containing a mix of bright/dark content, slow/fast motion scenes, and low/high texture), downloaded from a video sharing website [42] which provides free-of-use (commercial/non-commercial) video clips. Each result is obtained via 5 separate runs, each using 10 randomly selected clips. For these studies, the receiver camera is stationary (tripod-mounted), and the screen coordinates are precisely extracted via edge & contour analysis in OpenCV using a special *Red* frame displayed at the beginning of each experiment.

We evaluate *DeepLight* with the default grid size 10×10. Each 10×10 data frame contains 40–70 bits of random data, 5 bits of sequence number, 5 bits of checksum, and 50–20 bits of RS error correction code with a symbol size of 5 bits. By default, we utilize 50 *redundant* bits (50% of a data frame), implying that the RS-Code can correct any data frame whose BER is ≤5% (5-bit symbols). We also evaluate *DeepLight* with larger grid sizes: $\{20 \times 10, 20 \times 20, 30 \times 30\}$, with the percentage of redundancy bits being set to 60%, 70% and

80%, respectively, thereby ensuring that it can correct frames with BER ≤5%.

6.2.1 DeepLight *vs. d (Viewing distance).* In this experiment we place the camera at $0°$ (perpendicular to the screen), and measure the FER for $d = \{1m, 1.25m, 1.5m, 1.75m, 2.00m\}$ under *perfect screen extraction*. Table 1 lists the *FER* and *GP* values for a 10 X 10 data frame, with the RS-Code=50%. We see that the Mix-Δ scheme simultaneously provides low values of FER (<0.15) and high goodput (>1 Kbps), even at larger viewing distances ($d = 2$m). We also study the sensitivity of the decoding performance as the code rate (i.e., percentage of RS-coded redundancy bits), is varied over the set {60%, 50%, 40%, 30%}. Table 2 lists the corresponding *FER* and *GP* values, as a function of d. We see that, as expected, a larger fraction of redundant bit results in a lower FER (with the FER remaining below 0.1 when the redundancy percentage is 50% or higher). Conversely, a smaller fraction of redundancy bits (a higher code) rate allows *DeepLight* to achieve goodput as high as 1.72 Kbps at shorter distance (≤ 1.25meter) but results in a rapid goodput drop when the distances are larger ($GP = 1.3$Kbps when $d = 2$m). *We infer that a larger redundancy percentage allows the goodput performance to remain stable across a wider operating range.*

6.2.2 DeepLight *vs. θ (Viewing Angle).* We next study the performance of *DeepLight* under 9 different screen viewing angles θ : $\{\pm 60°, \pm 45°, \pm 30°, \pm 15°, 0°\}$, for 3 different viewing distances $d = \{1m, 1.5m, 2m\}$ and code rate (redundancy percentage) = 50%. Table 3 lists the FER and goodput values as a function of θ. We see that the FER stays low (≤ 10%) and goodput remains high (≥ 1Kbps) over at least a 60 deg viewing range (±30 deg), but degrades when the viewing angle gets more acute. To a large degree, this degradation may also be attributed to our use of a TN LCD panel, which

intrinsically has a narrower viewing angle than higher-end LCD screens (e.g, ones with IPS panel).

6.2.3 DeepLight *with Inaccurate Screen Extraction.*

We now study whether *DeepLight* can *support robust SCC even when the screen extraction is not completely accurate*. To perform this in a controlled manner, we conduct several distinct perturbations of the ground-truth coordinates of the screen (in the camera's pixel coordinates), including: (1) SHIFT: introduce translational error in the screen border along both X& Y axes, (2) EXPAND: Move the screen borders outward (the estimated screen now includes background pixels), (3) SHRINK: Move the screen borders inward (the estimated screen now excludes some of the screen pixels), and (4) ROTATE: Rotate the screen borders in clockwise/counter-clockwise direction. *The error magnitude is expressed in percentage of a cell size.* For example, at d = 1.5m (viewing a 25" screen), a cell in a 10×10 grid has a size (in the camera image) of $\approx 55 \times 30$ pixels. Accordingly, a SHIFT of 30% means introducing a translation error of 17 and 9 pixels in the vertical and horizontal borders, respectively. Table 4 tabulates the resulting FER and GP. We see that *DeepLight* is indeed *robust to inaccurate screen extraction*, tolerating as much as 30% noise in the screen extraction process without a dramatic increase in FER. SHIFT=40% has a sharply higher FER, as the grids/cells on the screen boundary now lose over 64% of their constituent pixels.

6.2.4 DeepLight *vs. Different Ambient Lighting.*

We next study the robustness of *DeepLight* to different ambient lighting environments, both with and without outdoor ambient lighting (represented as BG). The *DeepLight* Bit Decoder CNN was trained with the videos recorded under a combination of LED lights and fluorescent lamps (eFL) that use electronic ballast–this (eFL+LED) combination represents the *default* ambient lighting. The eFL lamps have considerably lower flickers compared to older fluorescent lamps using inductive ballasts (iFL). We evaluate *DeepLight* under a variety of additional lighting conditions, including (a) only iFL lights, (b) iFL + LED, and (c) with outdoor ambient lighting in the background (eFL+BG). Table 5 plots the FER and GP values (for viewing distance d = 1.5m) across these lighting alternatives. We see that *DeepLight's* goodput remains steady across different ambient lighting choices. While the FER is higher when only iFL is used, we should note that iFL is an outdated technology that is being progressively phased out.

6.2.5 DeepLight *vs. Grid size.*

As discussed earlier in Section 4, *DeepLight* supports larger grid size by repeating applying the 10×10 decoding pipeline on the input video frames. Of course, a larger grid size implies a smaller number of pixels/cell, and potentially higher decoding FER. Figure 12 compares the FER and GP of different grid sizes at different
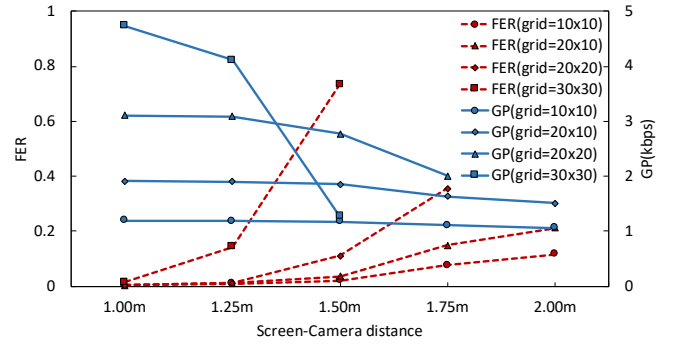

Figure 12: *DeepLight* FER and GP vs. d at different grid sizes.
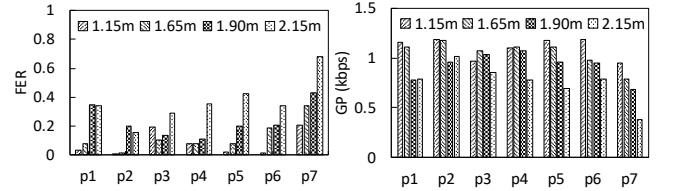

Figure 13: *DeepLight* FER (left) and GP (right) across different distances under real world artefacts.

distances. In comparison to the basic grid size of 10×10, the 20×10 grid size achieves comparable FER up to D = 1.5m, but degrades more quickly at distances larger than 1.5m. In general, as D increases, larger grid sizes cause the pixels/cell to be too low, leading to a sharp rise in FER. However, larger grids provide significantly higher goodput at short distances– e.g.,at D = 1.0m, *DeepLight* achieves a goodput of 4.7kbps using a 30×30 grid.

Key Takeaways: Overall, through these micro studies, we show that our DNN-based holistic decoding mechanism is robust, and not over-fitted to any specific content, screen border and aspect ratio. The test set is diverse, with 50 different clips of different contents, varying in different rates of dynamic change in screen content, and encompassing vistas ranging from nature to urban environments. *DeepLight's* steady performance across different grid sizes also implicitly demonstrates that the *DeepLight* is robust to different aspect-ratios. For example, for a 20x10 grid, the input frame is corresponding to an aspect-ratio of $\approx 10 : 12$, whereas the aspect ratio changes to 16 : 9 in the case of a 10x10 grid. *DeepLight* achieves similarly high accuracy across both grids at a distance d =1m; of course, the larger grids implicitly require shorter viewing distance because of the smaller cell size.

7 EVALUATION OF *DEEPLIGHT* SYSTEM

7.1 *DeepLight* Performance: User-held Smartphone

We next study the real-world performance of the actual *Deep-Light* implementation. The goal of these experiments is to

study the overall accuracy of the *Screen Detector* and *Bit Decoder* components, under typical artifacts of human smartphone usage. Accordingly, for these studies, the ML components are executed on the desktop computer. The studies involved 7 users using a smartphone (iPhone X) to record a display screen showing *DeepLight* video clips. Each user sat on a chair, placed at multiple distances={1.15m, 1.65m, 1.90m, 2.15m} from the screen, and recorded 10 randomly chosen (out of the 50 benchmarks) videos. To ensure motion-related artifacts, each user was instructed to *not* rest their elbow on the chair arms.

7.1.1 DeepLight Screen extraction accuracy. We analyzed the performance of the *DeepLight* Screen Extractor, using two widely-used metrics: (1) Intersection-over-union (IoU), defined as $IoU = (A_T \cap A_P)/(A_T \cup A_P)$, and (2) Intersection-over-Correct (IoC), defined as $IoC = (A_T \cap A_P)/(A_T)$, where A_T represents the true screen coordinates and A_P represents the screen coordinates estimated by the Screen Extractor. While a high IoU is needed to reduce the amount of spurious background content in the extracted screen, a high IoC is necessary to ensure that the *DeepLight* decoder has access to the entire screen area. The optimal (IoU, IoC) combination is achieved by tuning the kernel size (see Table (6)) of the dilation step in the screen extractor pipeline. By default, we use a kernel size of 2×2 (highlighted). We see that *DeepLight* screen extractor is able to include 97% of the screen under both indoor and outdoor conditions, and largely exclude the spurious background content (especially in indoor settings).

	Kernel size		
	1×1	2×2	3×3
Indoor	0.93/**0.95**	0.89/**0.97**	0.83/**0.99**
Outdoor	0.83/**0.97**	0.82/**0.97**	0.80/**0.94**

Table 6: Screen Extractor IoU/IoC

7.1.2 DeepLight FER and GP under real-world artefacts. We evaluate the robustness of the performance of *DeepLight* under real world human mobile device handling scenarios. In Figure 13 we plot *DeepLight's* FER and GP across the 7 users. In general, *DeepLight* achieves relatively low FER ≤ 0.2 and high GP ($\geq 0.95kbps$) even for a relatively long viewing distance of 1.9 meters (for at least 5 of the 7 users). At $d = 1.15$meter, the goodput is usually ≥ 1.1Kbps. The poorer performance (much higher FER) of user P7 was due to the significantly higher amount of observed hand tremor, which causes significant image blur. As a point of contrast, note that, (a) even for a very short viewing distance ($d = 0.7$meter), the default HiLight implementation (Section 3.2) exhibits a much higher BER of $\approx 29\%$ under a comparable screen detection accuracy (IoU= 89%), while (b) As reported in [5],

Chromacode achieves only 0.24 Kbps at 1.8m (0.47 Kbps with a 120FPS display), that too using a fixed camera (no movement artifacts). Overall, the results demonstrate that a smartphone-based implementation of *DeepLight* **offers significantly better (than comparative techniques) and robust performance under real-world usage artifacts**.

7.2 Smartphone Implementation for Real-time Application

We now further consider the use of *DeepLight* in a real-world application, where *DeepLight* is implemented entirely on the smartphone (no edge offloading). To support *real-time* decoding of camera-captured SCC content, *DeepLight* must balance both the *accuracy* and *latency* metrics.

To understand the latency characteristics, we first profile the per-frame processing time of each *DeepLight* component in the system. There are three components that consume noticeable processing time: (1) Screen Detector (a UNet-based DNN), (2) Perspective warping (using OpenCV) that transforms the predicted screen area into a square shape, and (3) the Bit Decoder (a CNN). The latency of the RS Decoder is negligibly low ($\leq 100\mu s$ even on a mobile phone) and can be excluded from the latency profiling.

7.2.1 Processing time profile: We measure the processing time of *DeepLight* in two different environments: (1) a desktop computer, and (2) a mobile device (previously described in Section 5.2). The desktop environment comprises a core-i7 7700 CPU, an AORUS GeForce GTX 1080 Ti GPU card and 64GB of DDR4 DRAM. We port our system to C++ with Tensorflow C++ API to fairly compare with the mobile implementation.

Figure 14 shows the processing time of the three components in a desktop and a mobile device. The Screen Detector (ScrDet) consumes the highest processing time to the system with a value of 17.3ms on the desktop computer and 38.5ms on the iPhone. Similarly, the perspective warping and decoding take 0.75ms and 3.1ms on the desktop; and 4.1ms and 13.9ms on the iPhone accordingly. Given this total processing time of ≈ 56.5ms on the iPhone, *DeepLight* can support a decoding rate of only ≈ 17.7FPS if all the components are executed on each image frame. However, we shall shortly show that it is not necessary to run the Screen Detector continuously.

7.2.2 Supported FPS:. To improve the supported FPS, our idea is to run the screen detector (the most time consuming component) only intermittently. This is based on the hypothesis that, for the common use case where a user stands or sits to watch the content, the screen position is unlikely to change significantly within a short time period (e.g. 100ms). To better
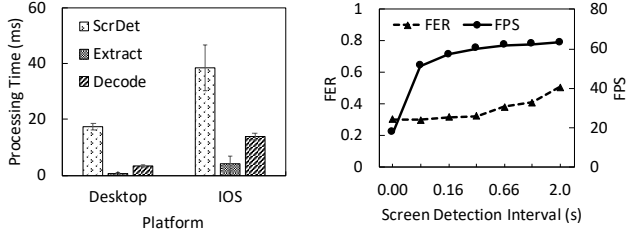
Figure 14: *LEFT*: **Processing time of different *DeepLight* Components (Desktop vs. iPhone 11Pro).** *RIGHT*: **Accuracy of *DeepLight* vs. Period between consecutive executions of screen detection, obtained when the smartphone is exhibiting motion perturbation while being held by a *standing* user.**
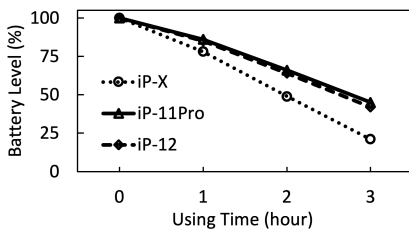

Figure 15: Battery level drop while running *DeepLight* continuously.

understand this drift across frames, we study how much intermittent screen extraction affects FER with a *standing* user to impose more hand motion artefacts. Figure 14(right) shows that when the screen detector interval increases to 83ms, the achievable decoding rate almost triples to 51.4 FPS, with only a negligible increase in the FER. (To support a decoding rate of 60 FPs, the Screen Detector can execute once every 330ms, resulting in a relatively small 2.4% increase in FER.) Beyond this point, the processing rate is bottle-necked by the latency of the Bit Decoder (which must, of course, be executed on each frame). Accordingly, **for our final real-time *DeepLight* implementation, we execute the Screen Detector only once every 32 frames (≈265msecs).**

7.2.3 Energy consumption. To examine how long a mobile device can support *DeepLight*, we let the smartphone run *DeepLight* continuously for 3 hours and observe the battery level. We use three different smartphones for this experiment, including an iPhone X, an iPhone 11 Pro, and an iPhone 12. Figure 15 shows battery level of the three smartphones after every one hour. We can see that even the "quite old" smartphone (iPhone X) can support more than 3 hours of continuous processing, while the more modern ones consume lower than 60% of battery capacity. We believe that an active operational lifetime of 4-5 hours (on a mobile device) is quite acceptable for our exemplar applications, such as smart, multi-lingual subtitle/notification or subliminal audio (Figure 1), which the user is unlikely to use continuously for more than 1-2 hours.

7.3 Closed-Captioning Application

We implemented a live-captioning iOS-based application (on the iPhone 11 Pro we used in our studies). We first encode part of the text-book "Alice in Wonderland" into a video clip using blue-channel modulation as described in Section 4. On the mobile device side, we integrate *DeepLight* into a self-developed camera recorder application which captures camera frames at 120FPS. To make sense of the response time of *DeepLight*, our application works in a triggered manner. Whenever a user presses the "START" button, it executes the entire *DeepLight* pipe-line to process the latest 32 frames (i.e., roughly 0.25 secs of video) stored in a circular-buffer. As discussed in Section 7.2.2, *DeepLight* only performs screen detection for the first frame (in 32 frames). Figure 16 shows a representative photo of our live-captioning application running in front of a screen. The decoded text can be seen with the content of *"Which was very likely"*. The estimated 4 corners of the screen are also shown as 4 "green" dots on the mobile screen. Note that *DeepLight* is able to decode the SCC data correctly, in spite of the modest errors in screen coordinate estimation.

7.4 Comparison with prior works

To compare *DeepLight* with prior works, we first formalize 3 distinct technical terms: (a) *Raw throughput* is defined as the total number of transmitted bits multiplied by the bit error rate; (b) *Throughput* is defined as the total number of bits correctly received after RS-based error correction; (c) *Goodput* is defined as the number of useful bits correctly received, excluding the overhead of error correction. *Note that the raw throughput is not useful if the error rate is high.* For example, for a data transmission rate=2Mbps (1920*1080 resolution), the raw throughput would be 1Mbps if the receiver made a random guess; however, the actual number of decoded *symbols* would be vanishingly low. As the source code of the state-of-the-art (Chromacode [5]) is not available, we compare *DeepLight* performance with the reported performance of previous works. However, we implement the encoder of Chromacode [5] and HiLight [4] to measure the perceived visual quality. To fairly compare different approaches, we normalized the throughput and goodput to a common screen frame rate of 60FPS, with a viewing distance of 1m.

Figure 17 provides a comparison of *DeepLight* against other state-of-the-art approaches. Note that, due to our CNN-based decoding techniques, *DeepLight* does not require either special display patterns or explicit ON/OFF activation of the screen to identify the screen coordinates. *DeepLight's* use of Blue-channel modulation also offers significantly higher MOS (greater imperceptibility) compared to the alternatives. Due to the large number of videos assigned to each participant, we only evaluate the visual quality of *DeepLight*, the most recent SCC work (Chromacode [5]) and HiLight [4]

Vu Tran, Gihan Jayatilaka, Ashwin Ashok, and Archan Misra

Figure 16: A smartphone, running *DeepLight*-enabled captioning application, displays the text (in 'Alice in wonderland' textbook) decoded from a monitor.

Work	Require SCR locator	Visual Quality	Throughput (kbps)	Goodput (kbps)	Processing time on mobile phone (ms)
DeepLight	No	Very high	26.6	4.7	16.6 (iPhone 11 Pro)
ChromaCode (2018) [5]	Yes (Black & White lines)	Low	N.A. (220 -- Raw throughput)	0.5 − 1.5	500 (Pixel2)
TextureCode (2016) [8]	N.R (Offline extracted)	N.A.	11.25	N.A.	N.A. (Offline)
Inframe++ (2015) [3]	Yes (QRCode locator)	N.A.	9	N.A.	200 (Core-i5 CPU + FirePro V3900 GPU)
Hilight (2015) [4]	Yes (OFF/ON screen)	High	4.6	N.A.	5 (iPhone 5)

Figure 17: *DeepLight* compared with prior works at 60FPS, 1m distance between screen and camera (N.A.= Not available).

(which shown to achieve low flicker at 60FPS). In terms of throughput, TextureCode [7] replicated Inframe++ [3] and HiLight [4] and reported the corresponding throughput values (TextureCode definition of 'correctly received bits' is equivalent to our Throughput definition). We borrow the throughput values reported in TextureCode [7], but normalized them to 60FPS instead of 120FPS. The real goodput values were not reported in those studies. Chromacode [5] reported raw throughput and goodput, but the throughput was not reported. We estimate the raw throughput and goodput of Chromacode from the reported values and normalize them to 60FPS. In general, *DeepLight* outperforms prior works in terms of both throughput and goodput. At 1m, using a grid size of 30×30, *DeepLight* achieve a throughput and goodput of 26.6Kbps and 4.7Kbps respectively. In terms of processing time, *DeepLight* achieves an average full-pipeline processing time of $\approx 16.6ms$; in contrast, HiLight [4] achieves a lower processing time of $\approx 5ms$. Though *DeepLight* does not achieve the lowest processing time, it supports $\approx 60FPS$ on a mobile platform (iPhone 11 Pro).

8 DISCUSSION

In this section, we discuss some of the limitations of our current work and possibilities for future improvement.

Grid Sizes and Smarter Spatial Codec (coding & decoding): While *DeepLight* currently assumes explicit knowledge of the grid size, we believe that this restriction can be overcome by having a preamble frame (with a predefined grid size) bearing metadata (e.g., grid size) of payload frames. In addition, it might be possible to design a smart encoder-decoder co-design that learns to optimize both shape and intensity modulation (e.g., using Generative Adversarial Network) for both low flicker and high goodput. In long term, it is intriguing to explore the possibility of a single model that performs both screen extracting and decoding.

Handling Moving Users: Current *DeepLight* implementation does not work well if the user is walking while holding the smartphone due to (a) the significant change in screen position across even consecutive frames caused by user's steps, and (b) the blurry nature of such in-motion images. While

the first problem may be addressed by introducing a smaller ScreenNet model suitable for continuous per-frame execution, the second can be addressed by well-known camera stabilization techniques [24, 43, 44].

Supporting Different Display Rates: We additionally experimented with a 30FPS video on our default 25" screen ($F_D = 60Hz$); *DeepLight* was seen to achieve high decoding accuracy ($FER = 0.03$, GP=0.58Kbps, $d = 1m$). Formally speaking, a Manchester pair of 30FPS video is translated into 8 camera frames ($F_C = 120FPS$), in which the transition from F+ to F− (or vice versa) still occurs across a $L = 3$ ($F+, T, F−$) frame triple, but albeit with this frame triple occurring less frequently. After finding the temporal separation between such frame triples, *DeepLight* can compute the actual frame rate.

9 CONCLUSION

In this paper, we presented *DeepLight*, an end-to-end SCC system whose receiver uses two novel DNN models to both (a) perform reasonably accurate extraction of screen content from a captured image without any external knowledge, and (b) decode the encoded bits collectively, without explicit partitioning of the screen content into constituent cells. These two innovations help us develop a practical SCC system that can work in the real-world—without special frame markers, under different lighting conditions and with varying viewing distances and angles. *DeepLight* consistently supports data goodput of 1Kbps at viewing distance of up to $\approx 2m$, with a viewing angle of more than $\pm 30°$. These innovations allow *DeepLight* to support several new classes of in-the-wild SCC applications via *personal mobile/wearable devices*. These advances are also testimony to the power of carefully introducing ML pipelines in advanced communication systems.

10 ACKNOWLEDGMENTS

REFERENCES

[1] Grace Woo, Andy Lippman, and Ramesh Raskar. Vrcodes: Unobtrusive and active visual codes for interaction by exploiting rolling shutter. In *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 59–64. IEEE, 2012.

[2] Anran Wang, Chunyi Peng, Ouyang Zhang, Guobin Shen, and Bing Zeng. Inframe: Multiflexing full-frame visible communication channel for humans and devices. In *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*, pages 1–7, 2014.

[3] Anran Wang, Zhuoran Li, Chunyi Peng, Guobin Shen, Gan Fang, and Bing Zeng. Inframe++ achieve simultaneous screen-human viewing and hidden screen-camera communication. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 181–195, 2015.

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

[5] Kai Zhang, Yi Zhao, Chenshu Wu, Chaofan Yang, Kehong Huang, Chunyi Peng, Yunhao Liu, and Zheng Yang. Chromacode: A fully imperceptible screen-camera communication system. *IEEE Transactions on Mobile Computing*, 2019.

[6] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[7] Viet Nguyen, Yaqin Tang, Ashwin Ashok, Marco Gruteser, Kristin Dana, Wenjun Hu, Eric Wengrowski, and Narayan Mandayam. High-rate flicker-free screen-camera communication with spatially adaptive embedding. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.

[8] Shuyu Shi, Lin Chen, Wenjun Hu, and Marco Gruteser. Reading between lines: high-rate, non-intrusive visual codes within regular videos via implicitcode. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 157–168, 2015.

[9] DH Kelly. Flicker. In *Visual psychophysics*, pages 273–302. Springer, 1972.

[10] Ernst Simonson and Josef Brozek. Flicker fusion frequency: background and applications. *Physiological reviews*, 32(3):349–378, 1952.

[11] GS Brindley, JJ Du Croz, and WAH Rushton. The flicker fusion frequency of the blue-sensitive mechanism of colour vision. *The Journal of physiology*, 183(2):497–500, 1966.

[12] Arnold Wilkins, Jennifer Veitch, and Brad Lehman. Led lighting flicker and potential health concerns: Ieee standard par1789 update. In *2010 IEEE Energy Conversion Congress and Exposition*, pages 171–178. IEEE, 2010.

[13] Colorimetry: Cielab color space. https://spie.org/publications/fg04_p71_cielab. Accessed: 2020-02-27.

[14] Eisaku Ohbuchi, Hiroshi Hanaizumi, and Lim Ah Hock. Barcode readers using the camera device in mobile phones. In *2004 International Conference on Cyberworlds*, pages 260–265. IEEE, 2004.

[15] Yue Liu, Ju Yang, and Mingjun Liu. Recognition of qr code with mobile phones. In *2008 Chinese control and decision conference*, pages 203–206. IEEE, 2008.

[16] Samuel David Perli, Nabeel Ahmed, and Dina Katabi. Pixnet: Interference-free wireless links using lcd-camera pairs. In *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*, MobiCom '10, page 137–148, New York, NY, USA, 2010. Association for Computing Machinery.

[17] Wenjun Hu, Hao Gu, and Qifan Pu. Lightsync: Unsynchronized visual communication over screen-camera links. In *Proceedings of the 19th annual international conference on Mobile computing & networking*, pages 15–26, 2013.

[18] Wan Du, Jansen Christian Liando, and Mo Li. Softlight: Adaptive visible light communication over screen-camera links. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.

[19] Tong Zhan, Wenzhong Li, Xu Chen, and Sanglu Lu. Megalight: Learning-based color adaptation for barcode stream recognition over screen-camera links. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(2):66, 2019.

[20] Niels Provos and Peter Honeyman. Hide and seek: An introduction to steganography. *IEEE security & privacy*, 1(3):32–44, 2003.

[21] Ingemar Cox, Matthew Miller, Jeffrey Bloom, Jessica Fridrich, and Ton Kalker. *Digital watermarking and steganography*. Morgan kaufmann, 2007.

[22] Tayana Morkel, Jan HP Eloff, and Martin S Olivier. An overview of image steganography. In *ISSA*, volume 1, page 2, 2005.

[23] Mauro Barni, Franco Bartolini, Vito Cappellini, and Alessandro Piva. A dct-domain system for robust image watermarking. *Signal processing*, 66(3):357–372, 1998.

[24] Shumeet Baluja. Hiding images in plain sight: Deep steganography. In *Advances in Neural Information Processing Systems*, pages 2069–2079, 2017.

[25] Eric Wengrowski and Kristin Dana. Light field messaging with deep photographic steganography. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1515–1524, 2019.

[26] Michael A Webster. Human colour perception and its adaptation. *Network: Computation in Neural Systems*, 7(4):587–634, 1996.

[27] Austin Roorda and David R Williams. The arrangement of the three cone classes in the living human eye. *Nature*, 397(6719):520–522, 1999.

[28] Hilight: Real-time screen-camera communication behind the scene. https://github.com/Tianxing-Dartmouth/HiLight. Accessed: 2020-02-27.

[29] Manfred Schroeder and BS Atal. Code-excited linear prediction (celp): High-quality speech at very low bit rates. In *ICASSP'85. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 10, pages 937–940. IEEE, 1985.

[30] Shyang-Lih Chang, Li-Shien Chen, Yun-Chung Chung, and Sei-Wan Chen. Automatic license plate recognition. *IEEE transactions on intelligent transportation systems*, 5(1):42–53, 2004.

[31] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *ArXiv*, abs/1505.04597, 2015.

[32] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010.

[33] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[34] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[35] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[37] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[38] tf.keras.losses.binarycrossentropy. https://www.tensorflow.org/api_docs/python/tf/keras/losses/BinaryCrossentropy. Accessed: 2020-02-27.

[39] Screen tearing effect. https://forums.developer.nvidia.com/t/screen-tearing-effect/30477. Accessed: 2020-02-27.

[40] Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

[41] Official bsd checksum source code. https://svnweb.freebsd.org/base/stable/9/usr.bin/cksum/sum1.c?revision=225736&view=markup. Accessed: 2020-02-27.

[42] The best free stock videos shared by the pexels community. https://www.pexels.com/videos/. Accessed: 2020-02-27.

[43] Brent Cardani. Optical image stabilization for digital cameras. *IEEE Control Systems Magazine*, 26(2):21–22, 2006.

[44] Ahmad Rahmati, Clayton Shepard, and Lin Zhong. Noshake: Content stabilization for shaking screens of mobile devices. In *2009 IEEE International Conference on Pervasive Computing and Communications*, pages 1–6. IEEE, 2009.