UnseenCode: Invisible On-screen Barcode with Image-based Extraction

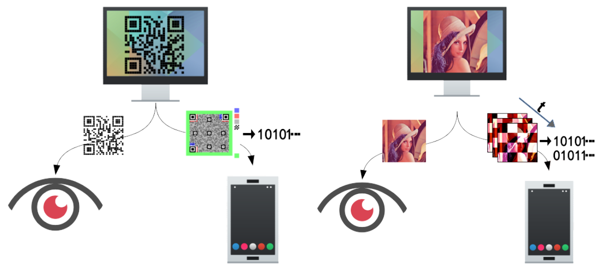
Hao Cui, Huanyu Bian, Weiming Zhang, Nenghai Yu
CAS Key Laboratory of Electromagnetic Space Information
University of Science and Technology of China
Hefei, China
{cvhc,hybian}@mail.ustc.edu.cn, {zhangwm,ynh}@ustc.edu.cn

***Abstract*—Screen-camera communication techniques achieve one-way communication with widely-used screen and camera devices. Popular barcode methods use visible spatial patterns to represent data, which has been considered obtrusive to human observers. Recent works borrow ideas from visual light communication (VLC), and use inter-frame pixel change to modulate data. To recover pixel change, the receiver has to record and process video frames. Such video-based extraction has high hardware requirements and lacks reliability. Inspired by VLC-based methods, we propose UnseenCode, an invisible on-screen barcode scheme. It leverages inter-frame embedment from VLC-based methods to embed invisible barcodes into arbitrary on-screen contents. Unlike VLC-based methods, UnseenCode does not require video-based extraction. We propose an image-based extraction method based on cross-component correlation of color images. Any off-the-shelf smartphones with camera capability can be used to read UnseenCode by capturing on-screen contents. We propose the full implementation of UnseenCode for evaluation. Experimental results show that UnseenCode decoding algorithm is reliable and applicable under various screen and camera settings. UnseenCode provides up to 2.5 kbit capacity with less than 5% error rate.**

I. Introduction

Screen-camera communication is a popular technology in the field of human-computer interaction (HCI). It utilizes commonly available screen and camera devices to achieve one-way device-to-device communication assisted by human. In a typical application scenario, the user uses a smartphone to read the message embedded on the screen. Here, screen devices (such as computer monitors) serve as transmitters, and camera devices (such as smartphone cameras) serve as receivers. Barcode methods like QRCode [1] have been long used for screen-camera communication. Barcodes use visible spatial patterns to represent data (Fig. 1a). Traditional barcodes are considered obtrusive to human observers, because they occupy part of the screen to show human-unreadable contents. Some works of beautified barcodes, such as PiCode [2] and [3], embed barcodes into images without losing human readability of images. However, beautified barcodes are still visible and degrade the visual experience of images. It is desired to design *invisible barcode* to minimize obtrusiveness to human observers.

Recently, several works gain inspiration from visible light communication (VLC). They embed data into temporal dimensions of on-screen contents as high-frequency or low-amplitude inter-frame pixel change (Fig. 1b), which appears as



(a) Barcode methods (b) VLC-based methods

Fig. 1: Illustration of two main types of screen-camera communication techniques

flicker to human eyes. Due to the limited sensitivity of human vision to flicker, the pixel change is completely invisible. Specifically, HiLight [4], InFrame++ [5] and TextureCode [6] modulate data with high-frequency luminance change. VRCodes [7], Uber-in-Light [8] and [9] embed data into chromatic components, because human vision is less sensitive to chromatic flicker.

Although VLC-based methods are sometimes informally named as invisible barcodes, they are fundamentally different from barcode methods in the sense of data embedment and extraction. Barcode methods embed data into spatial dimensions only, while VLC-based methods also make use of temporal dimensions. To borrow the terminology of video coding, we call these two schemes as *intra-frame embedment* and *inter-frame embedment*, respectively. Correspondingly, they require different extraction methods. A barcode can be captured by taking a single image (*image-based extraction*). In VLC based methods, inter-frame pixel change cannot be recovered from just one image, and the receiver has to record successive frames to extract inter-frame pixel change (*video-based extraction*).

Inter-frame embedment not only achieves invisibility but also expands the data capacity of screen-camera communication in the temporal dimension. Seemingly, VLC-based methods are strong competitors of barcodes. However, on the receiver side, we find that corresponding video-based extraction lacks applicability and reliability. Specifically, we point out three significant drawbacks of VLC-based methods:

- *(a) High hardware requirements.* To capture inter-frame pixel change, the receiver must be capable of recording video at Nyquist rate of display framerate, such as 240 FPS (frame per second) for HiLight and TextureCode. We investigate recent smartphone models and conclude that high-speed video recording is not supported by most of off-the-shelf smartphones (Tab. I) due to the limited performance of video encoding and file writing.

- *(b) Low decoding reliability.* Videos taken by handheld smartphones always suffer from shake, frame drop and other distortions, which are difficult to correct and can greatly impact on extraction accuracy. In contrast, images captured in well-lit conditions are generally not affected by shake because of the very short exposure time. A barcode scanner can even take advantage of the temporal redundancy of barcode images to improve extraction reliability. Barcode scanners apply a trial-and-error process to find decodable patterns among many captured images.
- *(c) High time complexity.* Video processing is generally more computationally expensive than image processing, which can be very challenging on mobile devices. In contrast, image-based extraction of barcodes has been proved to be very efficient on smartphones.

TABLE I: Camera hardware capability of recent entry- to mid-level smartphones (2013-2017)

Model	**Minimum Exposure Time**	**Camera Resolution**	**Maximum Video Recording FPS***
Huawei Mate 9	1/4000s	20 Mp	60 FPS
Nokia 7	1/500s	16 Mp	30 FPS
Xiaomi Mi 3	1/1000s	13 Mp	30 FPS
vivo X3L	1/1000s	13 Mp	30 FPS

* Burst shooting mode is not considered.

These limitations motivate us to rethink the design of truly invisible barcode. We expect that this kind of barcode is completely invisible to human, as in VLC-based methods. It should still support reliable image-based extraction and be applicable with most off-the-shelf screens and smartphones, as in traditional barcode methods. As for capacity, it should be comparable to existing barcode methods. In this paper, we propose UnseenCode, a novel invisible on-screen barcode scheme used in screen-camera communication. On the embedder side, UnseenCode uses inter-frame embedment method from VLC-based methods to hide barcode image in arbitrary on-screen contents. On the extractor side, we analyze the process of camera exposure and propose a reliable and effective barcode extraction method based on cross-component correlation of color images. We show that UnseenCode works with different hardware settings. Compared to VLC-based method, UnseenCode provides better reliability and applicability. In Tab. II, we compare UnseenCode with existing methods.

This paper is organized as follows. Section II introduces basic knowledge about flicker perception of human vision, which is the theoretical basis to achieve invisibility with inter-frame embedment. In Section III, we outline the core techniques used in UnseenCode. In Section IV, we present the design details of UnseenCode system. In Section V, we show various evaluation results to prove the performance of UnseenCode. Finally, we introduce some related works and conclude our work in the last two sections.

II. Background

VLC-based screen-camera communication leverages the limitation of the temporal sensitivity of human vision to achieve invisible embedment. In this section, we briefly introduce the characteristics of human vision. Previous works have given discussions about this topic. VRCodes [7] has a deep discussion of human visual system and camera imaging. HiLight [4] discusses the principles of inter-frame embedment in VLC-based screen-camera communication. We summarize the important parts which will be used to design UnseenCode.

A. Flicker Perception of Human Vision

Human eyes perceive temporal change of light intensity, i.e. flicker, in a low-pass manner. When light intensity fluctuates quickly enough, human eyes only perceive the average intensity instead of flicker. This phenomenon is called *flicker fusion effect*. The lowest frequency that causes flicker fusion is called *critical flicker frequency* (CCF). CCF is not a constant value and it varies with many factors. We point out some important characteristics of flicker fusion as follows:

- CCF depends on the amplitude of flicker. With weaker flicker, CCF is also lower.
- Human vision is more sensitive to luminance change than chromatic (color) change. The chromatic CCF is about 25Hz, only half of the luminance CCF [10].
- CCF is generally considered to be less than 60Hz under most circumstances [11]. Screen devices are designed to refresh at more than 60Hz frequency to avoid visible flicker [12].

Although human cannot perceive high-frequency flicker, it is possible to record such flicker with camera devices. The difference between human vision and camera exposure enables VLC-based screen-camera communication methods to achieve unobtrusive transmission with inter-frame embedment.

B. Color Fusion Rule

Flicker fusion can be quantitatively described in CIE XYZ color space by the *color fusion rule*. CIE XYZ color space represents every perceived color with three tristimulus values (X, Y, Z). Specifically, Y component measures the luminance of a color. The color fusion rule is a direct conclusion from the Grassmann's law of color matching [13]. It states that if two colors $\mathbf{L}_1 = (X_1, Y_1, Z_1)$ and $\mathbf{L}_2 = (X_2, Y_2, Z_2)$ alternate at a constant frequency (higher than CCF) on the screen, human vision perceives the fused color as:

$$\mathbf{L} = \frac{\mathbf{L}_1 + \mathbf{L}_2}{2} = (\frac{X_1 + X_2}{2}, \frac{Y_1 + Y_2}{2}, \frac{Z_1 + Z_2}{2}) \quad (1)$$

We call color $\mathbf{L}_1$ and $\mathbf{L}_2$ as a *fusion pair* of $\mathbf{L}$, which can be seen as a decomposition of $\mathbf{L}$ in CIE XYZ color space (Fig. 2). Specially, if we keep luminance component Y invariant, i.e.

TABLE II: Comparison of UnseenCode and existing screen-camera communication methods

	Method		Features		
	Embedment	Extraction	High Capacity	Unobtrusive	Reliability
Barcode methods (QRCode, PiCode [2]...)	intra-frame	<u>image-based</u>			√
VLC-based methods (VRCodes [7], HiLight [4], TextureCode [6]...)	<u>inter-frame</u>	video-based	√	√	
Proposed method (UnseenCode)	<u>inter-frame</u>	<u>image-based</u>		√	√

$Y_1 = Y_2$, then $\mathbf{L}_1$ and $\mathbf{L}_2$ form a *luminance-invariant fusion pair* of L, which can be seen as a decomposition of $\mathbf{L}$ in the XZ plane.

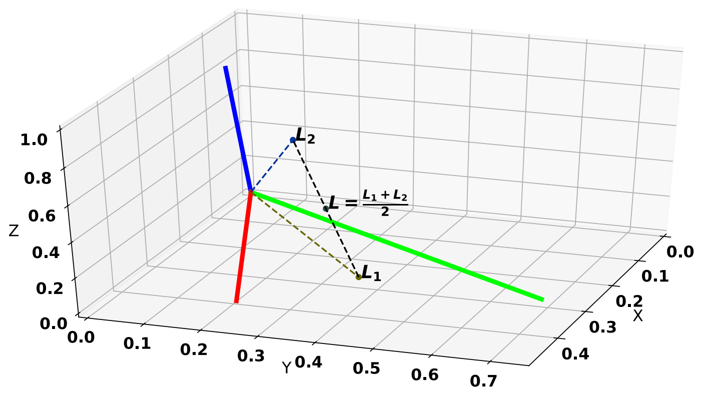


Fig. 2: Example of color fusion in CIE XYZ color space. $\mathbf{L}_1$ and $\mathbf{L}_2$ form a fusion pair of $\mathbf{L}$. The red, blue, and green lines correspond to sRGB primary colors.

III. Proposed Method

In this section, we propose the core techniques used in UnseenCode. First, we review inter-frame embedment method from VLC-based screen-camera communication, based on which we propose to embed invisible barcodes into on-screen contents by constructing luminance-invariant fusion pairs. Then we analyze the process of camera exposure and discover the possibility to use image-based extraction with UnseenCode. We propose an extraction method to recover the barcode from single captured image based on cross-component correlation of color images.

A. Inter-frame Embedment of Barcodes

VLC-based screen-camera communication leverages color fusion rule to construct fusion pairs and uses the inter-frame difference between a fusion pair to encode data. Here is a simple illustration. Let $I_o(i,j)$ be original on-screen content (a two-dimensional image), where (i,j) denotes pixel coordinate, and I denotes specific component of an image (like Y component in CIE XYZ color space). We can construct a sequence of frames as:

$$\begin{aligned} I_{2k-1}(i,j) &= I_o(i,j) - \frac{1}{2}\Delta I_k(i,j) \\ I_{2k}(i,j) &= I_o(i,j) + \frac{1}{2}\Delta I_k(i,j) \end{aligned} \tag{2}$$

I_{2k-1} and I_{2k} form a fusion pair of I_o. The fusion pairs are then displayed on the screen at high framerate. Due to flicker fusion, human observers perceive fused image I_o. Inter-frame difference is given by:

$$\Delta I_k(i,j) = I_{2k}(i,j) - I_{2k-1}(i,j) \tag{3}$$

We can use $\Delta I_k(i,j)$ to encode data. For example, TextureCode uses temporal Manchester coding and embed coded data in luminance component (so I is Y component of CIE XYZ color space). It takes $\Delta I_k(i,j) = \alpha s(k)$, where α is a constant scale factor to determine inter-frame luminance difference, and $s(k) = +1$ or -1 represents bit 0 or 1, respectively. HiLight uses frequency modulation in Y component. HiLight and TextureCode require at least 120Hz display framerate to maintain invisibility. Uber-in-Light and VRCodes use frequency modulation in R and B chromatic components of RGB color space and limit the amplitude of inter-frame difference, which can maintain invisibility with common 60Hz framerate.

These VLC-based methods effectively make use of temporal dimension k of $\Delta I_k(i,j)$ to transmit time-series data. Remember that each $\Delta I_k(i,j)$ is a two-dimensional image. We can use its distribution of pixel values over spatial dimensions i,j to represent data as well. By this way, UnseenCode embeds invisible barcode into on-screen contents. We illustrate our embedment method as:

$$\begin{aligned} I_{2k-1}(i,j) &= I_1(i,j) = I_o(i,j) - \frac{1}{2}\Delta I(i,j) \\ I_{2k}(i,j) &= I_2(i,j) = I_o(i,j) + \frac{1}{2}\Delta I(i,j) \\ \Delta I(i,j) &= \alpha(i,j) S_{\mathbf{m}}(i,j) \end{aligned} \tag{4}$$

where $S_{\mathbf{m}}(i,j) \in \{1,-1\}$ is a barcode image which encodes message $\mathbf{m}$, and $\alpha(i,j)$ is a scale factor to determine the amplitude of inter-frame difference of each pixel. The implementation details of barcode design and scale factor will be given in Section IV.

UnseenCode leverages chromatic flicker to improve visual experience as well. Specifically, UnseenCode constructs luminance-invariant fusion pair by embedding barcodes in X component of CIE XYZ color space. As in Uber-in-Light and VRCodes, UnseenCode is able to maintain invisibility at 60Hz to 75Hz display framerate, which is commonly supported by off-the-shelf screen devices.

B. Image-based Extraction

Now let's focus on the extractor side. Seemingly, we need to capture adjacent frames I_{2k-1} and I_{2k} to recover inter-frame difference ΔI. Because there is no timing synchronization mechanism between camera exposure and display, the camera has to sample on-screen content at twice the display framerate. This is essentially a video-based extraction process. However, video-based extraction imposes high hardware requirements for camera devices and lacks reliability. We analyze the exposure process of camera and find that the inter-frame difference is implicit in the captured image if the exposure time meets certain condition. And we propose a background removal method to recover barcode image based on cross-component correlation of color images.

1) Analysis of Camera Exposure: In contrast to human vision, a digital camera captures images via sampling. In a camera, each pixel sensor is exposed for a short time (*exposure time*) and records the light intensity integrated over time. Typical exposure time settings of smartphone cameras are 1/30s, 1/60s, 1/125s, etc. in image capture mode and 1/30s, 1/60s in video recording mode.

An intuitive idea is that we can capture each of the individual frames of a fusion pair with short exposure time to extract barcode image. However, it requires that exposure time is shorter than display frame interval and cannot span two frames. With typical 60FPS to 144FPS framerate of screen devices, it is still difficult to capture individual frames.

To discover the possibility of image-based extraction, we do a simple analysis of the exposure process. Consider the case of Equation (4), a fusion pair I_1 and I_2 are displayed alternately on the screen. Then the fused image is $I_o = \frac{I_1+I_2}{2}$, and the inter-frame difference (which carries barcode image) is $\Delta I = I_2 - I_1$. Suppose that the exposure time of I_1 and I_2 is given by $t_1 = \frac{1}{2}t_0 - \Delta t$ and $t_2 = \frac{1}{2}t_0 + \Delta t$, and ignore camera processing and distortions. The captured image I_{cap} is:

$$\begin{aligned} I_{\mathrm{cap}} &= t_1 I_1 + t_2 I_2 \\ &= t_0 I_o + \Delta t \Delta I \end{aligned} \tag{5}$$

This is an ideal model. In practice, the camera normalized the intensity histogram of the captured image. To simplify the discussion, we apply Gaussian normalization to I and assume that the term $\Delta t \Delta I$ has little impact on histogram compared to $t_0 I_o$. Then it is safe to leave out the factor t_0 in the normalized I'_{cap}. We simplify Equation (5) as:

$$I'_{\mathrm{cap}} = \frac{(I_{cap} - \mu_I)}{\sigma_I} = I'_o + \lambda \Delta t \Delta I \tag{6}$$

where λ is the scale factor due to normalization. We can find that inter-frame difference ΔI is implicit in the captured image if $\Delta t \neq 0$. Obviously, $\Delta t = 0$ happens when exposure time spans exactly even number of frames ($t_1 = t_2$). Such case rarely happens and can be prevented by custom exposure setting. Now the problem is how to remove the term I'_o from Equation (6), which we call as *background* image.

2) Background Removal based on Cross-component Correlation: Until now, our discussion only refers to single component of the image. In UnseenCode, X component of a color image is used for embedment, but another chromatic component Z is not used. We discover the possibility of background removal from *cross-component correlation* of color images. For a color image, there is generally a high correlation between different chromatic components. So we can assume the normalized pixel values between two components are almost the same, i.e. $X'_o \approx Z'_o$. Cross-component correlation has been effectively used for image restoration [14] [15].

For our case, Z component remains unchanged between frames, so we can expect Z component $Z_o = Z_1 = Z_2$ is highly correlated with original X component X_o. We can use $Z_{cap} \approx Z_o$ to estimate X_o. As before we apply histogram normalization to each component and get X'_{cap} and Z'_{cap}. Finally, we extract the inter-frame difference as:

$$\Delta X' = \lambda \Delta t \Delta X = X'_{\mathrm{cap}} - Z'_{\mathrm{cap}} \tag{7}$$

Note that above-mentioned derivations do not take noise, distortions and camera processing into account. The real pixel values are affected by these factors, and we cannot perfectly recover ΔX. However, as long as the pattern of ΔX in the captured image is strong enough, we can still decode the barcode by template matching method. We will discuss the implementation details in the next section.

IV. System Design

In the previous section, we have outlined the barcode embedment and extraction techniques used in UnseenCode. In this section, we will propose the design details of UnseenCode embedment and extraction algorithm.

A. Embedment Algorithm

The embedment process of UnseenCode is summarized into 4 steps, as shown in Fig. 3. The input is on-screen content represented as a color image, and the output is a luminance-invariant fusion pair of the input carrying the barcode image. As for dynamic content (video), each frame is processed independently to generate output frames.

Spatial Partition: To take advantage of high spatial resolution of smartphone cameras, the input image is divided into blocks, and each block encodes one bit of data. UnseenCode uses $45°$ tilted square barcode templates (explained later) to cover the screen (Fig. 4 and 5). The number of blocks determines data capacity. If we define N as the number of complete blocks in the first row of output frames, then the capacity is $2N(N-1)+1$ bits, because incomplete blocks on the image boundary are not used.

Color Separation: Then the input image is separated into X, Y and Z components. Typically, on-screen contents use sRGB color space to represent pixel values. We need to transform pixel values to CIE XYZ color space. The transformation is given by:

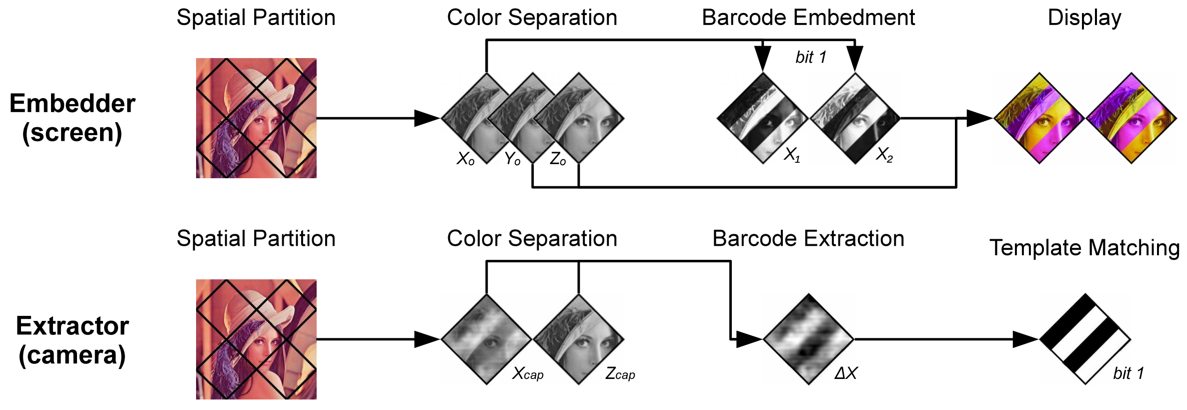


Fig. 3: Illustration of UnseenCode system design

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = T \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{8}$$

After this step, we get X, Y, Z components of each block as three two-dimensional arrays.

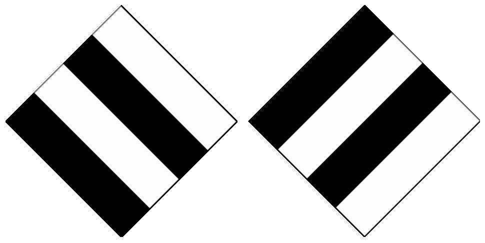


Fig. 4: Barcode templates to represent one bit

Barcode Embedment: Then UnseenCode embeds either barcode template $S_0(i,j)$ or $S_1(i,j)$ into each block to encode bit 0 or 1. We illustrate S_0 and S_1 in Fig. 4, where black and white pixels represent $S(i,j) = -1$ and $+1$, respectively. We choose $+45°$ and $-45°$ line stripes for better detectability on the extractor side, because the pixel array on the screen devices usually introduces horizontal or vertical line stripes into the captured images. Given the encoded bit in the current block, UnseenCode embeds corresponding barcode template into $X_o(i,j)$:

$$\begin{aligned} X_1(i,j) &= X_o(i,j) - \frac{1}{2}\Delta X(i,j) \\ X_2(i,j) &= X_o(i,j) + \frac{1}{2}\Delta X(i,j) \\ \Delta X(i,j) &= \begin{cases} \alpha(i,j) S_0(i,j), & \text{to encode bit 0} \\ \alpha(i,j) S_1(i,j), & \text{to encode bit 1} \end{cases} \end{aligned} \tag{9}$$

As for scale factor $\alpha(i,j)$, we have two constraints on it. First, to avoid noticeable flicker, we want to set a threshold $\alpha_{\max}$ to limit the inter-frame change. Second, we must ensure that generated frames can be converted back to valid RGB values, so that they can be displayed on the screen. With sRGB conversion formula given in Equation (8), we can get:

$$\begin{aligned} \begin{bmatrix} R_k & G_k & B_k \end{bmatrix}^\intercal &= T^{-1} \times \begin{bmatrix} X_k & Y_k & Z_k \end{bmatrix}^\intercal \\ &= \begin{bmatrix} R_o \pm 3.2409\,\alpha(i,j) \\ G_o \mp 0.9689\,\alpha(i,j) \\ B_o \pm 0.0557\,\alpha(i,j) \end{bmatrix}^\intercal \end{aligned} \tag{10}$$

So $\alpha(i,j)$ is the maximum value such that:

$$\begin{cases} \alpha(i,j) \le \alpha_{\max} \\ 0 \le R_k, G_k, B_k \le 1, \quad k = 1, 2 \end{cases} \tag{11}$$

In this step, we generate the luminance-invariant fusion pair (X_1, Y_o, Z_o) and (X_2, Y_o, Z_o) of the input image block and embed barcode image into X component.

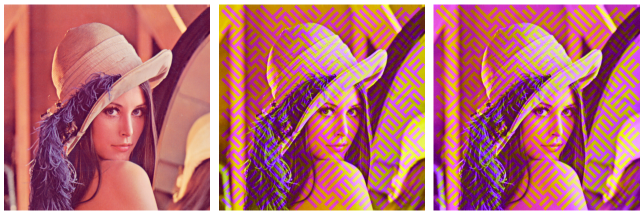
Display: After every block is processed, we get the luminance-invariant fusion pair of the input image with barcode embedded. Fig. 5 is an example of the output fusion pair. Output frames are inversely transformed back to RGB color space and then displayed alternately on the screen at desired framerate.

As for video, each frame is decomposed and displayed sequentially. Frame duplicating is usually needed to make up difference between video framerate and monitor refresh rate. For example, a typical a 24FPS video is re-encoded to 48FPS by UnseenCode, and every fusion pair is duplicated three times to synchronize with 144Hz refresh rate.

B. *Extraction Algorithm*

On the UnseenCode extractor side, the extraction process has 4 steps, as shown in Fig. 3. The input is the captured image of on-screen content. As in embedment algorithm, **spatial partition** and **color separation** are applied to the input image, and X_{cap} and Z_{cap} are extracted.

Barcode Extraction: As is stated in Section III, UnseenCode applies histogram normalization (Equation (6)) to X_{cap} and Z_{cap}, and gets X'_{cap} and Z'_{cap} respectively. The barcode image of each block is extracted by subtracting correlated Z component from X component:



(a) Original image (b) First frame of the fusion pair (c) Second frame of the fusion pair

Fig. 5: Example of generated fusion pair. Here $N = 12$ and 265 bits are embedded into the image of Lenna.

$$\Delta X'(i,j) = X'_{\text{cap}}(i,j) - Z'_{\text{cap}}(i,j) \tag{12}$$

In general, pixel values of $\Delta X'(i,j)$ is not equal to original $\Delta X(i,j)$. The capture process scales the intensity of pixel values and introduces various distortions.

Template Matching: To determine whether S_0 or S_1 is embedded in ΔX, a simple template matching is applied to $\Delta X'$. UnseenCode checks the correlation between extracted inter-frame difference and two barcode templates. The template with higher correlation is treated as embedded one:

$$\begin{aligned} \text{Corr}(I_1, I_2) &= \left| \sum_{i,j} I_1(i,j) I_2(i,j) \right| \\ m &= \begin{cases} 0, & \text{Corr}(\Delta X', S_0) < \text{Corr}(\Delta X', S_1) \\ 1, & \text{otherwise} \end{cases} \end{aligned} \tag{13}$$

After each block is processed, we get the embedded bits. Due to the noise introduced by camera capture, there are always error bits. We will evaluate the error rate in Section V.

(Optional) Multiple-image Extraction: This is an extra step to improve decoding accuracy. As in QRCode, UnseenCode can leverage temporal redundancy of barcode embedment as well. We use a simple per-bit voting mechanism. For multiple input images, the extractor decodes every image, compares every decoded bit, and take the most common decoding result for each bit as the final result.

V. Evaluation

In this section, we evaluate the performance of UnseenCode prototype. Our evaluation focus on visual experience, applicability with different hardware settings and decoding reliability. First, we briefly introduce the implementation of UnseenCode prototype. Next, we introduce our default experimental settings and evaluation metrics. Then we show detailed experimental results to prove the performance of UnseenCode and give discussions on the experimental results.

A. Prototype Implementation

The prototype of UnseenCode system consists of an embedder, a player and an extractor written in Python and C++ with OpenCV and SDL libraries. The embedder takes a message and images (either a static image or video frames) as input and outputs a sequence of frames with barcode embedded. The player then displays the embedded contents at specified framerate. The extractor parses the captured image, extracts the barcode image and decodes the embedded data. Currently, the decoder runs on computer and parses images captured by smartphone offline. The built-in camera application on smartphones is used to capture images.

B. Experimental Setup & Metrics

The default experiment settings are as follows. We display UnseenCode with AOC G2770PF 27' screen, which supports a wide range of refresh rate from 60Hz to 144Hz. The camera used to capture UnseenCode is Nokia 7 smartphone, which is an entry-level smartphone and supports up to 1/1000s exposure time. All experiments were conducted in a well-lit room.

大标题 タイトル Lorem ipsum

子曰："学而时习之，不亦说乎？有朋自远方来，不亦乐乎？人不知而不愠，不亦君子乎？"
子曰："巧言令色，鲜矣仁！"
子曰："为政以德，譬如北辰，居其所，而众星共之。"
孟武伯问孝。子曰："父母，唯其疾之忧。"

ウィキソとは、自由（フリー）なソーステキストや、その翻訳文をウィキで作成することを目的としています。詳細はウィキソースとは何かをご覧ください。

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

(a) Document (b) Lenna (c) Movie (dynamic)

Fig. 6: Test inputs in the performance evaluation.

We test three types of on-screen contents in all the experiments, including two static contents and one dynamic video content:

- Document, which contains only grayscale background and texts. (Fig. 6a)
- Image of Lenna, which serves as an example of texture color images. (Fig. 6b)
- Video with movements. We take the video clip from the movie *Big Buck Bunny*. (Fig. 6c)

For the objective performance evaluation, we take 7 images for each test case to perform multiple-image extraction to improve decoding accuracy. We keep the proportion of screen content at 30% area of captured image to simulate barcode scanning scenario. Capacity is set to 481 bits ($N = 16$). Decoded bits are compared with original encoded bits to calculate bit error rate (BER) of each test case. BER is the main metric of performance evaluation.

C. Subjective Evaluation

Invisibility is the fundamental design goal of UnseenCode, so we first evaluate the perceptual quality of embedded contents. For better decodability, we want to maximize the inter-frame change threshold $\alpha_{\max}$. But larger $\alpha_{\max}$ also increase the possibility of visible flicker. We want to find the maximum $\alpha_{\max}$ values at each display framerate without causing visible flicker. We conduct this part of tests by subjective evaluation. We also discuss other visible distortions caused by UnseenCode.

We ask 5 testers to give feedback on perceptual quality of UnseenCode embedded contents under different settings of display framerate. The original content and the embedded content is displayed side by side. Testers should report whether they can observe flicker, and if not, whether there are other differences between original and embedded contents.

The maximum flicker-free $\alpha_{\max}$ values at each framerate are shown in Tab. III. Below this maximum $\alpha_{\max}$, testers cannot notice flicker regardless of contents. As is expected, for lower framerate, $\alpha_{\max}$ has to be set smaller to prevent visible flicker. For further objective performance test, we will set $\alpha_{\max}$ to these maximum flicker-free level.

TABLE III: Maximum $\alpha_{\max}$ without causing flicker

Frame Rate	60FPS	75FPS	100FPS	120FPS	144FPS
$\alpha_{\max}$	0.10	0.15	0.40	no limit	no limit

In the absence of flicker effect, there are still other visible differences between original and embedded contents. We will discuss them below.

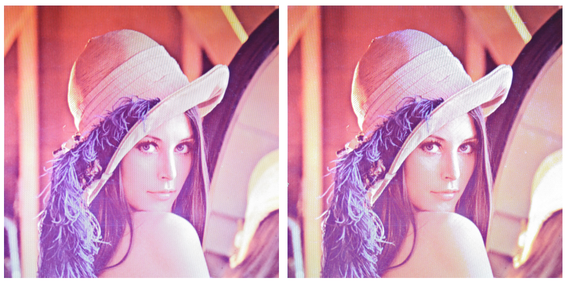

(a) Original image (b) Embedded image

Fig. 7: Illustration of perceptual quality by capture screen content (120FPS) with relatively long exposure time (1/15s). Two images are captured under the same camera settings. One can notice color distortion on the hat of Lenna.

The first is *color distortion*. This is due to the inaccurate color space transformation used in UnseenCode. The real RGB to CIE XYZ transformation is device-dependent and usually non-linear due to possible gamma correction to pixel values. In the current implementation of UnseenCode, we do not take gamma correction into consideration, and the generated fusion pairs do not correctly reproduce desired colors. To illustrate the distortion, we capture both original and embedded contents with relatively long exposure time to simulate the perception of human, as is shown in Fig. 7. The distortion is not very severe, and testers even cannot tell which one is distorted.

The second one, which we call it *uncertain visual disturbance*, is still not fully understood by us. When displaying embedded contents at very high framerate (144 FPS), testers sometimes still report visual disturbance, even if they cannot distinguish between original and embedded contents. We suspect that this is caused by *saccades* of human eyes. Saccades are random quick movements of eyes to maintain high sensitivity of human vision. It is reported human can perceive 500Hz flicker during saccades under specific circumstances [16]. It is possible that above-mentioned uncertain visual disturbance is noticeable flicker during saccades. We believe that all the VLC-based screen-camera communication methods suffer this problem. Detailed analysis is beyond the scope of this article.

D. Objective Evaluation

We evaluate decoding performance of UnseenCode. Our main concerns are:

- *Applicability*: To test whether UnseenCode works with a wide range of screen and camera setups.
- *Capacity*: To test how many bits UnseenCode can embed with an appropriate error rate.
- *Reliability*: To test how several practical factors impact on UnseenCode decoding accuracy.

TABLE IV: UnseenCode bit error rate evaluation under different settings of on-screen contents, display framerate and camera exposure time

Display Framerate	**Camera Exposure Time**	**Bit Error Rate**		
		Document	Lenna	Video
75 FPS ($\alpha = 0.15$)	1/30s	4.56%	17.58%	13.56%
	1/60s	0.10%	2.67%	3.75%
	1/125s	0.04%	0.92%	1.50%
	1/250s	0.00%	0.54%	1.60%
100 FPS ($\alpha = 0.40$)	1/30s	0.10%	3.88%	6.54%
	1/60s	0.00%	1.90%	4.44%
	1/125s	0.00%	0.42%	1.00%
	1/250s	0.00%	0.38%	0.98%
144 FPS ($\alpha = \infty$)	1/30s	0.88%	6.85%	10.17%
	1/60s	0.06%	4.48%	7.69%
	1/125s	0.00%	2.27%	1.00%
	1/250s	0.02%	2.77%	1.58%

1) Evaluation of Applicability: In this evaluation, we fix the relative positions of smartphone camera and screen with tripod. We change the framerate of on-screen contents and camera exposure time to see how screen and camera setups impact on decoding accuracy. On the camera side, we test commonly-used standard exposure time 1/30s, 1/60s, 1/125s and 1/250s. These exposure time settings are supported by most smartphone cameras. To avoid the case that exposure time is an even multiple of frame interval, we only test 75 FPS, 100 FPS and 144 FPS framerate on the screen side. $\alpha_{\max}$ values in Tab. III are used to limit inter-frame pixel change.

The result is given in Tab. IV. In most settings, BER is less than 5%, and we can expect good decoding performance with appropriate error correcting code. High BER can happen in the following conditions:

- The screen content is complex and/or dynamic. For Lenna image, the BER is higher than smooth and monochromatic document scene. And for dynamic video scene, the BER is even higher.
- The exposure time is too long relative to frame interval, like 144 FPS & 1/30s setting. In this case, the difference in exposure time between each frame of a fusion pair is not significant, and the barcode image is submerged in noise.

- The $\alpha_{\max}$ value is small, like 75 FPS screen setting. In this case, the amplitude of inter-frame difference is too weak to be captured.

On the camera side, shorter exposure time is preferred. Android system provides APIs for exposure control, and most of recent smartphones support up to 1/1000s exposure time. Older smartphone may not support custom exposure control. In this case, auto exposure should work as well. On the screen side, 100 FPS setting provides better decodability. For low-end screens, 75 FPS setting still maintains a reasonable BER.

2) Evaluation of Capacity: In this evaluation, we use 100 FPS & 1/60s screen-camera setting. To check the appropriate capacity range of UnseenCode, we change block partitioning parameter N and see how it impacts on BER. The result is given in Tab. V. The capacity can reach at least 1.5 kbit with BER less than 5%. For document content, the capacity can reach at least 2.5 kbit, which is roughly equivalent to QRCode version 8 to 9.

TABLE V: UnseenCode bit error rate evaluation under different settings of on-screen contents and capacity

N	Capacity /bits	Bit Error Rate		
		Document	Lenna	Video
12	265	0.00 %	0.38 %	3.62 %
16	481	0.01 %	1.94 %	4.73 %
20	761	0.02 %	3.81 %	3.15 %
24	1105	0.21 %	3.86 %	2.96 %
28	1513	0.53 %	4.38 %	5.79 %
32	1985	1.38 %	5.88 %	7.98 %
36	2521	3.52 %	8.53 %	10.02 %

3) Evaluation of Reliability: Lastly, we evaluate several practical factors that may or may not impact on decoding performance. Because we focus on practical usage and hope to draw qualitative conclusions about reliability, all the experimental images are captured with handheld smartphones without tripod.

a) Distance: The impact of the distance from camera to screen is examined. As before, we use the area ratio of on-screen contents to measure the distance indirectly. The result is shown in Fig. 8a. As is expected, the BER increases as the distance increases, because the number of valid pixels decreases. As for our test phone (Nokia 7), the BER remains low until the area ratio drops to about 20 to 25%. A possible solution is to use cameras with higher spatial resolution.

b) Angle: The impact of (horizontal) camera view angle is examined, as is shown in Fig. 8b. The BER only fluctuates by about 1%, so we conclude that view angle does not impact on decoding performance significantly.

c) Shake: As for video-based extraction in VLC-based screen-camera communication, video shake impacts on decoding performance significantly. As we cannot measure the degree of shaking accurately, we only give a qualitative discussion. Since all the images in our reliability evaluation are captured by handheld smartphones, we can compare the results to previous evaluations. We find that there is no significant performance degradation. In UnseenCode, image-based extraction requires relatively short exposure time. Hand shake can be ignored in such a short time, as long as the captured image is clear and not blurred.

d) Phone Model: Lastly, we compare the performance on other smartphones as well. Camera capabilities have been given in Fig. I. The result is shown in Fig. 8c. There are apparent performance differences between different models. Xiaomi Mi 3, which was released in 2013, performs worse. Nokia 7, which was released in 2017, performs best. We can expect that new smartphone cameras generally perform better than old ones.

VI. Related Works

A. Barcodes

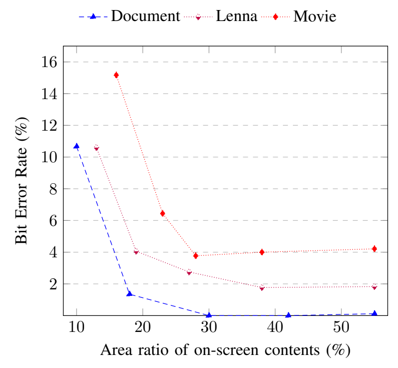
Barcodes use machine-readable image patterns to encode data. UPC and QRCode [1] are most popular barcode techniques. Barcodes can be printed on paper or displayed on screen (screen-camera communication) and read out by barcode scanners. Previously, many works focus on high-capacity barcode design. However, the capacity of QRCode is thought to be enough in most practical scenario. Some recent works focus on visual experience of barcodes [2] [3]. In these beautified barcode design, barcodes are embedded into images without losing the human readability of original contents. UnseenCode focuses on visual experience as well. However, different from beautified barcodes, UnseenCode focuses on the scenario of screen-camera communication, i.e. barcodes on the screen. So we can borrow ideas from VLC-based screen-camera communication to construct invisible barcodes on the screen.

B. VLC-based Screen-camera Communication

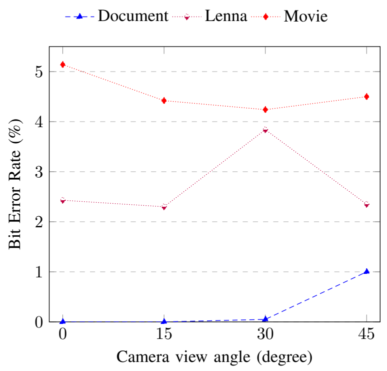
Visible light communication (VLC) is a very mature field of research. Screen-camera communication leverages visible light channel as well. Naturally, techniques of VLC can be used in this emerging field. Most works of VLC-based screen-camera communication emphasize obtrusiveness and capacity. VRCodes [7] and HiLight [4] are early works of this field, which use inter-frame chromatic and luminance change, respectively, to modulate data. We refer to such data embedment scheme as inter-frame embedment. TextureCode [6] optimizes the selection of embedding area to effectively reduce obtrusiveness. Uber-in-Light [8] leverages chromatic flicker to modulate data and proposes an enhanced MUSIC-based demodulation method to enhance transmission accuracy. VLC-based screen-camera communication requires different data extraction scheme (video-based extraction) to barcodes (image-based extraction). In this paper, UnseenCode uses inter-frame embedment to construct invisible barcodes. However, UnseenCode does not require video-based extraction. By using image-based extraction as in barcodes, UnseenCode achieves both high applicability and reliability with off-the-shelf screen and camera devices.

C. Image Watermarking

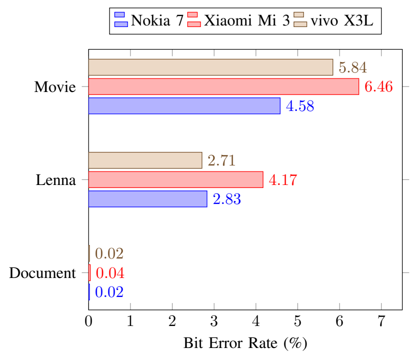
Image watermarking aims to covertly embed copyright data into images or video. The embedment scheme should be



(a) Area ratio of on-screen contents (Camera distance)



(b) Camera view angle



(c) Phone model

Fig. 8: Impact of other practical factors on decoding performance

robust enough, so the embedded data can survive from various image processing, such as resizing, compression and even camera re-capture. Anti-capture watermarking is a relatively popular field in watermarking. Most of existing works embed data in specific transform domains in which some coefficients are kept invariant after re-capture. For example, [17] embeds watermarking data into log-polar transform domain. Digimarc [18] is a commercial application of anti-capture watermarking. In this paper, UnseenCode can survive from camera re-capture. It may serve as a method of anti-capture watermarking.

VII. Conclusion

This paper proposes UnseenCode, a novel invisible on-screen barcode scheme, which supports image-based extraction like traditional barcodes with off-the-shelf smartphones. Our design leverages inter-frame embedment method, which originates from VLC-based screen-camera communication, to embed barcodes invisibly into chromatic component of the on-screen contents. We examine the exposure process of camera and propose to use cross-component correlation to recover UnseenCode barcode from single captured image (image-based extraction). We present UnseenCode implementation and evaluate both visual experience and decoding performance with off-the-shelf screen and camera devices. These evaluations confirm the applicability and reliability of UnseenCode prototype.

Acknowledgments

This work was supported in part by the Natural Science Foundation of China under Grant U1636201 and 61572452.

References

[1] International Organization for Standardization, *Information technology – Automatic identification and data capture techniques – Bar code symbology – QR Code*, ISO Std., 2000.
[2] C. Chen, W. Huang, B. Zhou, C. Liu, and W. H. Mow, “Picode: A new picture-embedding 2d barcode,” *IEEE Trans. Image Process.*, vol. 25, no. 8, pp. 3444–3458, 2016.
[3] Y.-H. Lin, Y.-P. Chang, and J.-L. Wu, “Appearance-based qr code beautifier,” *IEEE Trans. Multimedia*, vol. 15, no. 8, pp. 2198–2207, 2013.
[4] T. Li, C. An, X. Xiao, A. T. Campbell, and X. Zhou, “Real-time screen-camera communication behind any scene,” in *Proc. of the 13th Annual Int. Conf. on Mobile Systems, Applications, and Services (MobiSys)*. ACM, 2015, pp. 197–211.
[5] A. Wang, Z. Li, C. Peng, G. Shen, G. Fang, and B. Zeng, “Inframe++: Achieve simultaneous screen-human viewing and hidden screen-camera communication,” in *Proc. of 13th the Annual Int. Conf. on Mobile Systems, Applications, and Services (MobiSys)*. ACM, 2015, pp. 181–195.
[6] V. Nguyen, Y. Tang, A. Ashok, M. Gruteser, K. Dana, W. Hu, E. Wengrowski, and N. Mandayam, “High-rate flicker-free screen-camera communication with spatially adaptive embedding,” in *Proc. IEEE Int. Conf. on Computer Communications (INFOCOM)*. IEEE, 2016.
[7] G. Woo, A. Lippman, and R. Raskar, “Vrcodes: Unobtrusive and active visual codes for interaction by exploiting rolling shutter,” in *Proc. IEEE Int. Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2012, pp. 59–64.
[8] M. Izz, Z. Li, H. Liu, Y. Chen, and F. Li, “Uber-in-light: Unobtrusive visible light communication leveraging complementary color channel,” in *Proc. IEEE Int. Conf. on Computer Communications (INFOCOM)*. IEEE, 2016.
[9] E. Wengrowski, K. J. Dana, M. Gruteser, and N. Mandayam, “Reading between the pixels: Photographic steganography for camera display messaging,” in *Proc. IEEE Int. Conf. on Computational Photography (ICCP)*. IEEE, 2017.
[10] Y. Jiang, K. Zhou, and S. He, “Human visual cortex responds to invisible chromatic flicker,” *Nature Neuroscience*, vol. 10, no. 5, p. 657, 2007.
[11] A. Eisen-Enosh, N. Farah, Z. Burgansky-Eliash, U. Polat, and Y. Mandel, “Evaluation of critical flicker-fusion frequency measurement methods for the investigation of visual temporal resolution,” *Scientific Reports*, vol. 7, no. 1, p. 15621, 2017.
[12] M. Menozzi, F. Lang, U. Naepflin, C. Zeller, and H. Krueger, “CRT versus LCD: Effects of refresh rate, display technology and background luminance in visual performance,” *Displays*, vol. 22, no. 3, pp. 79–85, 2001.
[13] E. Reinhard, E. A. Khan, A. O. Akyuz, and G. Johnson, *Color imaging: fundamentals and applications*. AK Peters/CRC Press, 2008.
[14] J. Park, Y.-W. Tai, and I. S. Kweon, “Identigram/watermark removal using cross-channel correlation,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012, pp. 446–453.
[15] N. P. Galatsanos, A. K. Katsaggelos, R. T. Chin, and A. D. Hillery, “Least squares restoration of multichannel images,” *IEEE Signal Process. Mag.*, vol. 39, no. 10, pp. 2222–2236, 1991.
[16] J. Davis, Y.-H. Hsieh, and H.-C. Lee, “Humans perceive flicker artifacts at 500 hz,” *Scientific Reports*, vol. 5, p. 7861, 2015.
[17] L. A. Delgado-Guillen, J. J. Garcia-Hernandez, and C. Torres-Huitzil, “Digital watermarking of color images utilizing mobile platforms,” in *Proc. IEEE Int. Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2013, pp. 1363–1366.
[18] (2018) Digimarc: The Barcode of Everything. [Online]. Available: https://www.digimarc.com/