

InFrame++: Achieve Simultaneous Screen-Human Viewing and Hidden Screen-Camera Communication

Anran Wang *
Beihang University
wangar@act.buaa.edu.cn

Zhuoran Li *
The Ohio State University
li.5233@osu.edu

Chunyi Peng
The Ohio State University
chunyi@cse.ohio-state.edu

Guobin Shen
Microsoft Research, China
jackysh@microsoft.com

Gan Fang
The Ohio State University
fang.254@osu.edu

Bing Zeng
University of Electronic Science and
Technology of China
eezeng@uestc.edu.cn

ABSTRACT

Recent efforts in visible light communication over screen-camera links have exploited the display for data communication. Such practices, albeit convenient, have led to contention between space allocated for users and content reserved for devices, in addition to their visual anti-aesthetics and distractedness. In this paper, we propose INFRAME++, a system that enables concurrent, *dual-mode*, full-frame communication for both users and devices. INFRAME++ leverages the spatial-temporal flicker-fusion property of human vision system and the fast frame rate of modern display. It multiplexes data onto full-frame video contents through novel complementary frame composition, hierarchical frame structure, and CDMA-like modulation. It thus ensures opportunistic and unobtrusive screen-camera data communication without affecting the primary video-viewing experience for human users. Our prototype and experiments have confirmed its effectiveness of delivering data to devices in its visual communication with imperceptible video artifacts for viewers. INFRAME++ is able to achieve 150-240 kbps at 120FPS over a 24" LCD monitor with one data frame per 12 display frames. It supports up to 360kbps while data:video is 1:6.

Categories and Subject Descriptors

C.2.0 [Computer-Communication Networks]: General—*Data communications*; H.5.1 [Information Interface and Presentation]: Multimedia Information Systems—*Video*; H.5.2 [Information Interface and Presentation]: User Interfaces—*Screen design*

Keywords

Screen-camera communication; Hidden visible communication; Dual-mode visible communication; Full-frame video; InFrame++

*The first two authors are co-primary student authors. The correspondence faculty author is C. Peng.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MobiSys'15, May 18–22, 2015, Florence, Italy.
Copyright © 2015 ACM 978-1-4503-3494-5/15/05 ...\$15.00.
<http://dx.doi.org/10.1145/2742647.2742652>.

1. INTRODUCTION

Recent years have been witnessing the rapid growth of electronic visual displays deployed in the emerging cyber-physical world. Indeed, we are surrounded by such devices in various form factors, ranging from a phone screen, a tablet display, a computer monitor, to a TV, an advertising electronic board and even larger multi-screen displays. These visual displays have become a primary source for information from the user's standpoint. For example, video playback has contributed to 79% of Internet traffic [1]. Under such common usage scenarios, the main role of a screen is to convey information to human eyes. The display thus establishes the *screen-to-eye* communication.

It is often highly desirable to further convey certain side information to a user while she watches the display. A real-life example is to refer the viewer to a movie webpage for additional information. Another popular case is for vendors to present additional information (*e.g.*, product specification and sales) in the TV commercials. With the growing prevalence of cameras, *screen-to-camera* visible communication has rapidly emerged as a convenient, impromptu communication channel [11, 14, 15, 22, 30]. Information is encoded into visual patterns and shown on the screen. Camera-equipped devices subsequently capture the screen and retrieve the data information thereafter.

Unfortunately, such special visual patterns (*e.g.*, Quick Response (QR) codes) are not quite consumable, if at all, by humans who are more comfortable with human-friendly contents such as texts, images and videos. When a display becomes the same source concurrently for both human-friendly primary content and camera-friendly side information, the two contend for the display. Recognizing such contention and the fact that delivering human-consumable content is more primary purpose, current practices have all resorted to some kind of compromise, either spatially or temporally. In the QR code case, the code would take a small area and resides at a corner of the entire screen (Figure 1a), or occupy the whole screen in turn after human-friendly contents (Figure 1b). The former spatial compromise, however, not only limits its information-carrying capability, but also incurs extra work to properly capture codes (*e.g.*, being close enough to QR codes). Albeit small, the appearance of the QR code is still deemed distracting, thus impairing user-viewing experiences. The latter temporal compromise indeed yield a larger display size and makes it less effort demanding in capture and also possible to carry more bits. However, such practice is viable only for interactive scenarios and on interactive devices.

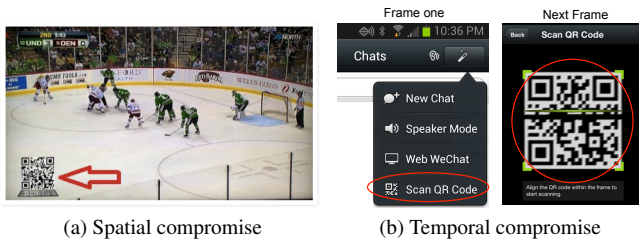


Figure 1: Current practices of compromising the screen-to-eye and screen-to-camera communications. (a) Small QR code located at the corner (in the case of an ice hockey game); (b) Two consecutive frames, one for each channel (while scanning QR code at WeChat [3]).

In this work, we address the key challenge in visual communication systems: Can we completely eliminate the tension between the screen-to-eye and screen-to-camera channels? If succeeded, we can afford the users to enjoy normal full-frame viewing experience (as though the side channel would not exist), while still being able to simultaneously convey side information over the visual screen-to-camera channel. The solution calls for a novel paradigm of *dual-mode, full-frame communication*, which enables concurrent delivery of primary video content to users and additional information to devices over screen-to-camera visual links without impairing user-viewing experiences.

In this paper, we present the design and implementation of INFRAME++, which offers a promising and definite solution to the above challenge. Figure 2 illustrates the concept of INFRAME++. In INFRAME++, composite contents are produced (in frames) for the display by multiplexing the video content frames (intended for human viewers) and the data (intended for devices, also in frames). These composite frames can be rendered to human eyes without affecting the viewing experience. The user thus watches the video as usual without sensing the embedded data frames. In the meantime, the data carried by the composite frames can be captured and decoded by the camera to retrieve the embedded side information. To enable the above functions, INFRAME++ leverages the capability discrepancy and distinctive features of the human vision system and devices (display and camera), *e.g.*, screens can display content faster than human eyes perceive; cameras have shutter but human eyes do not, etc.. Video and data carried by the composite frames thus operate at different time scales. Video contents are perceived at the slow pace due to physical limits of human eyes, whereas data frames are displayed at the fast speed, which can be captured and decoded only by the camera. In essence, INFRAME++ establishes two concurrent visual communication channels – the *primary* screen-to-eye channel for humans and the *secondary* screen-to-camera channel for devices. Both channels originate from the same screen, coexisting and counteracting against each other.

Specifically, we have two concrete goals for INFRAME++. First, the information-carrying data stream over the screen-to-camera channel should not affect the user perception of the content delivered over the primary channel. Second, we seek to achieve high data rate over the secondary screen-to-camera channel, which is vital to enabling new application scenarios for visual data communication in the future. To this end, we have to effectively manage the interference from the primary screen-to-eye channel. Note that the visual content of the primary channel will inevitably interfere the secondary channel. Even worse than the results disclosed in early work, the screen-to-camera channel may suffer from severe dis-

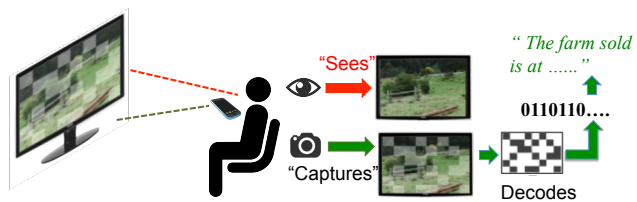


Figure 2: Concept of INFRAME++: screen-to-eye for videos and screen-to-camera for data communication over the full-frame same visible channel simultaneously.

tortions such as frame rate mismatch, rolling shutter effect, poor capture quality, *etc.*.

For the first goal, we propose the novel concept of *spatial-temporal complementary frames* (STCFs). STCFs fully exploit the spatial and temporal low-pass filtering properties of human vision system. Each of the complementary frames in STCFs contains a pair of data frames, which have complementary contents and are displayed back-to-back [31]. Every data frame is further constructed from spatially complementary visual patterns that possess alternating complementary cells (*i.e.*, group of pixels). When data frames are multiplexed to the original screen content and displayed at fast frame rate, we can effectively suppress the visibility of data frames and preserve normal viewing experiences. On the other hand, a camera takes (temporal) snapshots of the multiplexed stream with high spatial resolution. The snapshots exhibit obvious artifacts introduced by complementary visual patterns, which can be processed to extract the information bits. We further design smoothing transitional frames so that STCFs can be morphed into data frames.

For the second goal, we devise a hierarchical data frame structure (consisting of Cells, Blocks and Segments) and a CDMA-like modulation scheme. A Cell is the minimum logical unit making up from neighboring pixels with the same value. A Block, consisting of multiple Cells, is the basic coding unit. Information bits are modulated in Blocks. A Segment is a group of Blocks, over which certain error protection is exerted. These hierarchical units allow for robust data extraction from the captured artifacts. To handle other screen-to-camera channel distortions, we further design special preambles that consist of regularly distributed special locators and alignment blocks, as well as reference symbols in special visual patterns that help to learn dynamic channels.

We have implemented the INFRAME++ prototype and conducted thorough evaluation including user studies. Our experiments confirm that INFRAME++ enables dual-mode full-frame visible communication to both humans and devices simultaneously. Without noticeable artifacts or flickers, INFRAME++ is able to yield up to 360 kbps (150-240kbps in most test cases) of data rates for the screen-to-camera communication, up to 30–60x improvement, compared with our prior work [31].

In summary, we have made three main contributions in this work:

- We propose a novel dual-mode full-frame visible communication paradigm, in which the same display can stream high-rate data to a user device while retaining normal viewing experiences.
- We introduce the spatial-temporal complementary frame concept and verify its the feasibility through user studies. We further address several practical challenges, and prototype and assess a complete working system.
- We demonstrate its promise for high-rate data delivery by achieving 150-240 kbps rate (up to 360 kbps), an order of magnitude higher than early proposals.

We believe that, as more display devices are deployed in the emerging cyber-physical world, the new visual communication paradigm of INFRAME++ will become more appealing for its superior user experience and its high rate for the embedded data.

The rest of the paper is organized as follows. §2 introduces the background on human vision system and modern cameras. §3 gives an overview of INFRAME++. §4 and §5 describe our detailed design in screen-to-eye and screen-to-camera channels. §6 presents the implementation and evaluation of INFRAME++. §7 discusses the potential applications and open issues. §8 compares with related work, followed by the conclusion in §9.

2. BACKGROUND

Vision is the primary information acquisition means for human beings. Human vision system (HVS) is highly advanced and complicated. In this section, we present some related basic properties of HVS, and for comparison purpose, also capabilities of displays and cameras.

2.1 Human Vision System (HVS)

Human eyes are the core instrument in human vision system, which reacts to light and thus obtains the perception of object shapes, colors, depth and motion, *etc.*. The structure of an eye is similar to that of a camera [4], in the sense that an eye also has a lens (crystalline lens) and a sensor plate (retina). The sensor units (receptor cells) on the retina are greater at the center and lowest at the edges, thereby creating central (also called foveal) and peripheral vision. Central vision accounts for the high visual acuity capability, essential to capturing major visual details. The larger peripheral area is dominated by highly-sensitive light sensors (rod cells) and thus it is good at detecting motion, and becomes more effective in the dark.

While offering rich information from the physical world, our naked eyes have inherent physical limits in resolution, both spatial and temporal. In particular, our visual perception possesses three optical features, namely *center-surround response*, *low-pass flicker fusion*, and *phantom array effect*, that relate to the spatial resolution, temporal resolution and motion sensitivity, respectively.

Center-surround response. Human eyes cannot discriminate overly fine details, due to the physical limit of the minimum distance between adjacent sensing cells on the central retina. Typically, the excellent spatial resolution of human eyes is 0.07° , corresponding at 1.2mm at 1 meter watching distance [28]. The spatial resolution is also affected by the spatial contrast. The receptive field of human eyes has different responses at center and surrounding. Small (or large) receptive fields are stimulated by high (or low) spatial frequencies. Typically, our spatial perception follows a bandpass (close to low-pass) characteristics: the largest frequency is about 2-4 cycles per visual angle (equivalent to one cycle contrast within 5mm at 1m watching distance) [28]. In short, human eyes can not distinguish too tiny things, and finer details fuse and appear as some kind of average. One example is the eyesight check: when the fine details exceeds that of the eye, one will only see the rough shape.

Low-pass flicker fusion. Time-variant fluctuations of light intensity are not perceptible to human eyes when it is beyond a certain frequency, termed *critical flicker frequency* (CFF for short hereafter) [18, 27]. Instead, human eyes only perceive the average luminance. The temporal behavior of human vision system can be approximated as a *linear low-pass filter* at a high frequency exceeding the CFF. A common example is the viewing of a rotating car wheel. When it rotates fast enough, semi-transparent perception will result in. Note but the CFF is affected by many factors includ-

ing color contrasts, motion, luminance waveforms, and to name a few. In typical scenarios, the CFF of human eyes is believed to be about 40-50Hz according to vast vision research results [6, 10, 18]. For instance, flickers presented by a 60Hz CRT monitor are not perceptible.

Phantom array effect. This describes another special property of human eyes – sensitive to motion. While fast-moving objects zoom across view (either by object motion, or by eye motion such as rolling eyes), flicker can be noticed even when the display frequency is much higher [13]. For example, we can observe the flashing of an LED flashing at a frequency far exceeding typical CFF if the LED is moving in a dark environment, but we only see a constantly lit LED if it is still. Unlike flicker fusion, the origin of phantom array effect is not fully understood. Recent studies uncover that lower flicker amplitude, larger duty cycle and larger beam size make it less visible [25, 29].

2.2 Modern Display and Camera

With rapid technology advancement in recent years, displays and cameras are thriving rapidly. Most of off-the-shelf LCD displays, especially those 3D capable ones, supports 120Hz or higher refresh rate. For example, Eizo FG2421 24" LCD monitor, which is used in our work, supports 120 frames per second (FPS) and even 240FPS (in turbo mode). Other monitors like LG 47LH55 even physically supports 240 FPS. Moreover, with emergence of organic light-emitting diode (OLED) based displays, a much lower response time (less than 0.01 ms) is supported compared with LCD, indicating its ability to support a rather high refresh rate.

Cameras are also evolving dramatically. Special purpose high resolution cameras can yield giga-pixel images and reach over 2000 FPS video capture frame rate. Even commodity smartphone cameras can support high resolution images at a fast capture rate, which exceeds what the retina is looking for. For example, Samsung Galaxy S5 supports 16-mega pixel resolution and 120FPS capture rate, while iPhone 6 accommodates 8-mega pixel resolution and up to 240FPS capture rate.

2.3 A Comparison

One particular difference between an eye and a camera is that the eye does not contain a shutter. As a result, there is no exposure process while we see things, nor we see things in frames (or snapshots). It also accounts for the flicker fusion property. In contrast, camera takes discrete snapshots, every detail on the snapshot can be examined. Moreover, while human vision system is still far advanced than a camera, per the specific metric of spatial and temporal resolution, it is already well exceeded by the camera.

The distinctive properties of HVS and cameras, and their capability gap provide potential design room we can explore. Considering the fact that HVS has largely remained constant over long time whereas display and camera technologies continue advancing at fast paces, the capability gap, hence the design room, will grow larger in future.

3. INFRAME++ OVERVIEW

Our goal is to enable full-frame, dual-mode visible communication that allows users to enjoy normal full-frame viewing experiences while, at the same time, allowing side information being streamed to camera-equipped devices. Compared with alternative ways to deliver side information (*e.g.* using WiFi for receiving the audio), it enjoys intrinsic synchronization between primary content and side information, and the convenience of impromptus communication (*i.e.* avoidance of device pairing), among others.

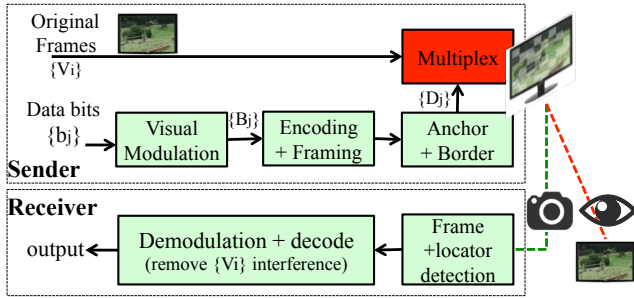


Figure 3: INFRAME++ architecture and operations.

Requirements and Basic Idea. To achieve the goal, we need to establish concurrent screen-to-eye and screen-to-camera communication channels. Given that the fundamental purpose of a screen is to show content to users, the screen-to-eye channel is more primary. Thus, a *first* requirement is to ensure *unimpaired* viewing experience, that is, full-frame viewing without any perceivable quality degradation. To serve as an effective side channel, a *second* requirement is to achieve high data rate at the existence of the primary video content.

The two visible communication channels share the same source – the screen, whereas to achieve the goal, we need to establish two distinctive channels, one for human and one for device. Thus the only possibility is to exploit the different capabilities of human vision system and the camera. As mentioned in previous section, human vision system has the low-pass filtering property in both spatial and temporal domain but a camera can take discrete snapshots of the screen content, we may come up with the design that embeds high frequency artifacts. If these artifacts can be low-pass filtered out but remain catchable by the camera, we can convey information using the artifacts.

Challenges. Despite the ever increasing capability gap between human vision system and the display/camera, it is non-trivial to realize the idea. There are two main challenges arisen from inherently conflicting screen-to-eye and screen-to-camera channels.

- *Retaining normal viewing with side channel.* The requirement not to affect the primary screen-to-eye channel imposes a rigid constraint on the secondary screen-to-camera channel. Although human eyes have a low temporal resolution, they are very sensitive to flickers. We thus need to find an appropriate frame multiplexing scheme to combine a data stream and arbitrary screen content, and ensure multiplexing will not introduce any perceivable color distortion, artifacts and flickers. It is especially challenging when the content in the screen-to-camera channel is dynamic.

- *Enabling data communication with primary content.* To serve the primary video watching goal, the perceived luminance level is mainly dominated by the original frames whereas the injected data patterns only contribute to small “noise”. Since the original luminance level is distinct at different pixels in one frame, its dynamic range usually overwhelm the noise made for data delivery. This is a critical issue to design a data frame structure so that they can be retrieved from the multiplexed frames without loss of its carrying data. In the meanwhile, we have to address the constraints in the traditional screen-to-camera communication, such as blurring and rolling shutter effects in [11, 14, 30].

INFRAME++ design. As will be elaborated in subsequent sections, we overcome the first challenge through the key concept of *spatial-temporal complementary frames* and the superior frame rate of displays (§4). Information bits are carried with spatial complementary visual patterns that are assembled into temporal comple-

mentary frames. The resulting complementary frames are multiplexed with the original screen contents. When displayed at high frame rate (e.g. 120FPS), complementary frames perceptually cancel each out and original screen contents retain. It thus ensures normal viewing experiences. On the contrary, a camera takes (temporal) snapshots and with high spatial resolution. The captured multiplexed frame will have obvious artifacts, which can be processed to extract information bits.

We address the second challenge through the design of a hierarchical data frame structure made up from Cells, Blocks and Segments, and a CDMA-like modulation scheme. Information bits are modulated in Blocks that exhibit spatial complementary visual patterns. Error protection is applied among Blocks in a Segment. These designs allow robust extraction of data from captured artifacts. To handle other screen-to-camera channel distortions, we also design special preambles that consists of regularly distributed special locators, and further design synchronization symbols in special visual patterns to help realign Cells and Blocks.

Architecture and Operations. Figure 3 illustrates the architecture of INFRAME++. It consists of encoding logic at the sender and decoding logic at the receiver.

Given the original frames $\{V_i\}$ and the input data bit stream $\{b_j\}$, INFRAME++ works as follows. Information bits $\{b_j\}$ are first embedded into certain visual patterns $\{B_j\}$ by the visual modulation module. In the subsequent encoding and frame module, error protection is exerted and similarly modulated. The generated visual patterns are grouped into Segments and further formed them into full-size data frames. Preambles and specially designed borders are added subsequently. Finally, the data frames are multiplexed to the origin frames in some way and display them on the screen.

At the receiver side, the process is essentially reversed. It first detects the preamble and restores locator positions, via the locator detection module. It proceeds to detect the existence of visual patterns, and demodulate them if exist, through a correlation-based matching process. The extracted bits then go through the error correction process and give the original information bits. With great efforts to hide data communication, INFRAME++ also ensures unobstructive human viewing experience as anticipated.

4. RESPECT USER VIEWING OVER PRIMARY SCREEN-TO-EYE CHANNEL

This central problem is how to embed information-carrying visual patterns into the original frames so that such changes remain invisible to naked human eyes. Our solution is to use the spatial-temporal complementary frames. In this section, we elaborate its concept and design, and also present the handling of sharp switch between dynamic data frames.

4.1 Spatial-temporal Complementary Frames

Spatial-temporal complementary frames (STCF) are a new development from the (temporal) complementary frames we originally proposed in [31]. For presentation integrity, we also include the basic definition of complementary frames here.

Let us first define *complementary pixels*. Two pixels p and p^* complement each other with respect to the luminance level v if their pixel values sum up to $2v$, i.e., $v_p + v_{p^*} = 2v$. Note that, in actual data frame, the average luminance level is set to zero ($v = 0$) and we have $v_{p^*} = -v_p$, whereas in the illustrations we have set the luminance value to mid-gray ($v = 128$) as it is not possible to show a pixel with negative value.

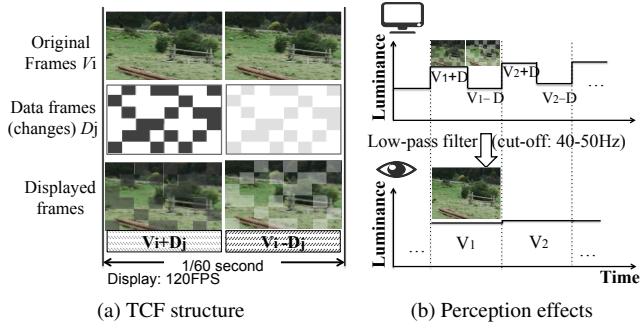


Figure 4: Illustration of temporal complementary frames (TCFs) (a) and their visual effects perceived by eyes (b) through a low-pass filtering flicker fusion.

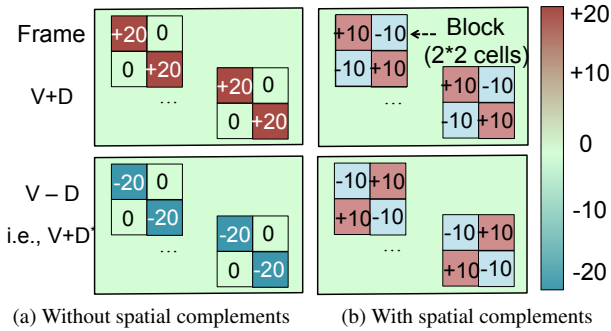


Figure 5: Illustration of spatial complementary frames (SCFs): (a) no spatial complements (only TCF is applied) and (b) spatial complements is used with TCF (that is, STCF).

Temporal complementary frames (TCF). A pair of data frames D and D^* are called temporal complementary frames (complementary frames for short) with respect to the luminance level v if all their pixels are complementary w.r.t v . It is easy to see that, after the temporal low-pass flicker fusion of human vision system, two complementary frames will yield average frames with luminance level v . (Again, in practice, we will set $v = 0$ when multiplexing with video content.) Note that, complementary frames always appear in pairs and take effect with temporal fusion. Given one data frame, it is easy to obtain its complementary frame. Figure 4 illustrates the basic idea of a complementary pattern design over consecutive frames.

Spatial complementary frame (SCF). Based on the complementary pixel concept, a frame consisting of spatially alternating complementary pixels (*i.e.* neighboring pixels are always complementary to each other) are called a spatial complementary frame. Figure 5b illustrates one spatially complementary pattern, where two neighboring pixels meet $v_p + v_p^* = 0$. Due to the spatial low-pass filtering property of human eyes, a spatial complementary frame will be perceived as a grey frame with average luminance. We realize it is actually the exact principle of dithering used in printing that can produce mid-tones with only black ink. When adding a zero-mean spatial complementary frame to a video frame, we will hardly see the difference.

Spatial-temporal complementary frames (STCF). Combining the two concepts above, we define spatial-temporal complementary frames as a pair of complementary frames each of which

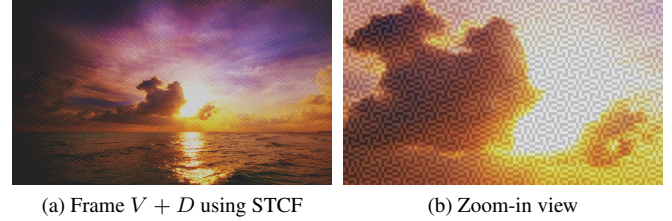


Figure 6: Examples of STCF frame (a) and its zoom-in view (b) using a normal sunset video frame.

is a spatial complementary frame itself. Clearly, STCFs can leverage both the spatial and temporal low-pass filtering properties of human vision system. Compared with TCF, STCF would also be able to mitigate the phantom array effect. Hence, STCFs are more effective in suppressing the visibility of data frames when multiplexed to video content, as confirmed in the following feasibility study.

4.2 STCF Frame Structure Design

Recall the very purpose of design STCF is to convey data to a camera, therefore, we need to make sure enough artifacts (under the assertion of unnoticeable to humans) can be captured by the camera. Considering the possible blurring and geometric distortions (either the display or the camera) of the screen-to-camera channel, in INFRAME++, we make two changes.

First, we use a *Cell* as the basic operation unit (instead of the raw physical pixel of the display or camera). A Cell consists of $p * p$ physical pixels. All the element pixels of a Cell have (or assumed to have) the same luminance value when forming (at the sender) or handling (at the receiver) spatial-temporal complementary frames. Clearly, the cell size defines the minimum spatial resolution in use.

Second, we introduce a *Block* as the basic information-carrying unit. A Block consists $c * c$ (c is an even number) neighboring complementary Cells, and exhibits certain visual pattern. In a simplest case, we may use existence of element cells of a Block area to convey one bit, as done in [31]. To pursue higher throughput, in INFRAME++, we allow flexible composition of Blocks from Cells while ensure Cell-level complementary within a Block (see §5). Thus, Blocks will exhibit multiple visual patterns, and one Block can carry multiple bits.

Figure 5a and 5b illustrate two consecutive frames of the original complementary frame scheme in [31] and the proposed STCF. For frame $V + D$, in the original scheme, some Blocks may contain Cells organized in a chessboard pattern, with values $\delta = 20$ and 0; and other Blocks may contain all zero Cells. In contrast, in STCF, Blocks may contain Cells organized in different pattern, but with values $\delta = \pm 10$, and Cells are all complementary to their neighbors in a Block, even without a temporarily complementary frame. For the temporally complementary frame $V - D$, all the adjustment values are reversed.

Figure 6 illustrates the visual effect of multiplexed frames when SCTFs are adopted. Due to spatially complementary patterns, SCTFs are less noticeable even when being presented in the static form of an image, compared with our prior work [31]. One may argue that the original scheme uses patterns of $\delta/0$ Cells, and it is actually a mean-shifted version of a spatial complementary pattern. We point out that such a mean-shift (of STCF) has profound impact. Without it, the average luminance of a chessboard-patterned Block will have a non-zero change, whereas a zero Block have a zero average. When multiplexed to video content, they will cause unequal

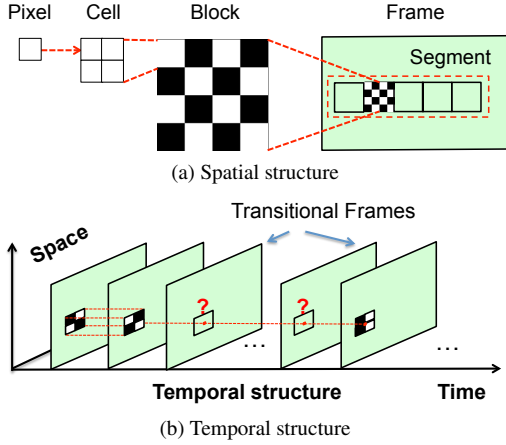


Figure 7: Structure of STCF in space and over time.

luminance changes that can only rely on the complementary frame to mitigate. Worse even, the actual impact on human perception will be dependent on the actual video content. For example, adding the same delta to a bright or dark area will lead to different visual changes. On the contrary, the average of a Block in STCF is always zero. Multiplexing STCF with video content will have minimum visual impact. An alternative way to read this benefit is that we can use much larger δ in STCF if to achieve the same level of invisibility of the original scheme. This is also helpful as it allows more effective capture and detection of artifacts at the receiver.

Note that the introduction of Cells and Blocks makes it possible that the resulting data frame is not spatially complementary in the strict sense as neighboring pixels may be of identical values. Nevertheless, the resulting STCFs are still effective. They are significantly better than the scheme of [31]. Such improvements and the effectiveness of STCF have been validated through an extensive study in § 6.

Figure 7 shows the overall frame structure of formal STCF. Figure 7a decomposes the hierarchical spatial structure. A frame is made from intermediate logic units of Segments (for error protection purpose, see §5.3), Blocks, Cells, and the actual physical pixels. Figure 7b shows the temporal structure. Each data frame is rendered as a pair complementary frames. However, it encounters with a sharp change due to the transition between different data frames. To alleviate that, in addition to the complementary frames (the first two frames), we propose the transitional frames to ensure smooth switch to a different data frame, as described below.

4.3 Smoothing Transitional Frames

STCF can transmit a data frame invisibly. Different data frames will end up with different STCFs. We thus face with a critical challenge when transmitting dynamic data frames. This is concerning how to switch between consecutive STCF pairs. If it sharply switches from one to another (e.g., from $V_1 - D_1$ to $V_1 + D_2$), strong flickers are noticed due to the low-pass filter property of human eyes. The abrupt switching is equivalent to a quick motion and imposes severe vision interference.

To mitigate the effect, we propose *smoothing transitional frames* to ensure gradual switch. We introduce a transition function $\Omega(t)$ to gradually change the amplitude of one data frame in a transition cycle τ . When their luminance amplitude needs to switch from δ_j to δ_{j+1} between two successive data frames D_j and D_{j+1} , the luminance of each Cell (i.e. all physical pixels in the Cell) in the t -th transition frame ($t = 0, 1, \dots, \tau - 1$) turns into

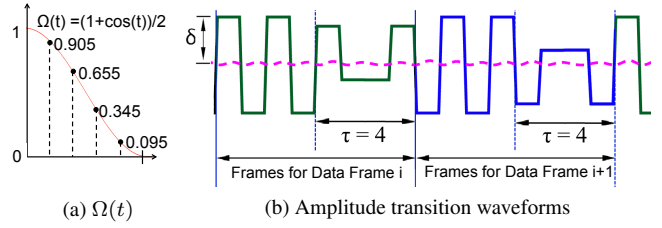


Figure 8: An example of transition waveform when data frame amplitude varies from δ to $-\delta$ (green and blue solid curve) and its perception effect (purple dotted curve) after applying an electronic low-pass filter. Here, $\tau = 4$ and $\Omega(t)$ is half of the square-root raised Cosine waveform. For simplicity, the primary video frame V is not displayed, but the delta part for data frames.

$$v + \begin{cases} (-1)^t [(\delta_j - \delta_{j+1}) \cdot \Omega(t) + \delta_{j+1}], & \delta_j > \delta_{j+1} \\ (-1)^t [(\delta_{j+1} - \delta_j) \cdot \Omega(t) + \delta_j], & \delta_j < \delta_{j+1} \\ (-1)^t \delta_j, & \delta_j = \delta_{j+1} \end{cases} \quad (1)$$

Note that, if a Cell does not need to change, the amplitude will remain constant, just like the original complementary frames. $\Omega(t)$ is a waveform that decreases from 1 to 0. In our experiments, we use the half the square-root raised Cosine waveform for $\Omega(t)$ after comparing with linear and stair function forms. Parameters τ is also critical to data communication. Through our user study, we find that a small τ (≤ 4) is needed when STCF is adopted. This greatly reduces the count of transition frames when the temporary scheme is used where a minimum value is 10-14 in [31].

Figure 8 gives an illustrative example of a switch from δ to $-\delta$. That is, when $\tau = 4$, $\Omega(t) = [0.905, 0.655, 0.345, 0.095]$. By applying the above equation, the amplitude coefficients for each frame (including complementary frame pairs) are $0.81\delta, -0.31\delta, -0.31\delta, 0.81\delta$, in turn. Notice that there have adopted multiple replica complementary frames out of one single data frame. Such repeated transmissions are to cope with possible mismatches between the display refresh rate (e.g. 120FPS) and camera capture rate (e.g. 30FPS). For example, a public display may send data over a variety of cameras.

In essence, we generate multiple intermediate data frames, which are multiplexed with video frames, to ensure imperceptible transition between two different data frames. Spatially, each data frame adopts special patterns to alleviate their visual effects. Temporally, we take two forms of smoothing. One is over two consecutive frames that complement to each other, and the other is over multiple transition frames which gradually switches between different data frames. Note that, the above temporal smoothing also helps to mitigate the rolling shutter effect [24] as the corrupted data can be recovered from other frames. We will address this problem in next section.

5. BOOST DATA COMMUNICATION OVER SCREEN-TO-CAMERA CHANNEL

With the presence of the primary screen-to-eye content, what available at the receiver is some artifacts – a mixture of the original video content plus the added data frame that have undergone various screen-to-camera channel distortions such as blurring, geometrical distortion, imprecise positioning, frame rate mismatch, rolling shutter effects, etc.. Despite the STCF design, the require-

ment not to affect the normal viewing experience dictates the amplitude of the data frame to be relatively small when compared to the dynamic range of video content. This is in sharp comparison with other screen-to-camera work where the “background” of their codes are clean and large amplitude (e.g. black or white) is allowed [11, 14, 22, 30]. It is also different from those work that hide codes in static images [19, 34, 36] where the background does not change, and those video/image watermarking and steganography work [5, 9, 12, 17, 20, 26, 35] where the altered video/images assumed available at the receiver without any errors (no screen-to-camera channels). To achieve high throughput, it is imperative to make good use of these artifacts. In this section, we present our designs and special measures to achieve the goal.

5.1 CDMA-like Modulation

As aforementioned, Blocks can be formed from complementary Cells in different ways and exhibit as different visual patterns. Thus, a Block may carry multiple bits. Assume K bits are mapped into certain Block $D_k[c, c]$, $k \in \{1, 2, \dots, 2^K\}$, and 2^K is the number of unique Block type. Each Block has a size of $c * c$ Cells (c is an even number). Modulation is concerning how to select (or form) these Blocks. After multiplexing, the screen area corresponding to the Block is displayed in the form of $V[c, c] \pm D_k[c, c]$. On the receiver side, it is captured as $V'[c, c] \pm D'_k[c, c]$. Demodulation is to infer which Block (i.e. visual pattern) is used at the specific position of the captured frame.

With the interference of video content and possible channel distortions in mind, we devise a Code Division Multiple Access (CDMA)-like modulation scheme to facilitate accurate and robust demodulation at the receiver. For simplicity, we assume $D_k[c, c]$ is an $c * c$ matrix which is arranged by a code vector ϕ_k (each element is 1 or -1), multiplied by δ . These codes satisfies two properties: the orthogonality between any two codes and the zero sum of each code. That is, $\langle \phi_k, \phi_j \rangle = 0, k \neq j$ and $\sum(\phi_k) = 0$, where \langle, \rangle is an operator of inner product.

By exploiting their orthogonality and the zero sum, we infer the block type with the maximum inner product with the received block. That is,

$$\hat{k} = \arg \max_k \langle D_k[c, c], V'[c, c] + D'_k[c, c] \rangle. \quad (2)$$

This is because

$$\begin{aligned} & \langle D_j[c, c], V'[c, c] + D'_k[c, c] \rangle \\ &= \langle D_j[c, c], V'[c, c] \rangle + \langle D_j[c, c], D'_k[c, c] \rangle \\ &= \sum_r \sum_l (\delta_j(r, l) \cdot v'(r, l)) + \sum_r \sum_l (\delta_j(r, l) \cdot \delta'_k(r, l)) \\ &\approx v' \cdot \delta \cdot \sum(\phi_j) + \delta^2 \cdot \langle \phi_j, \phi_k \rangle \\ &= \begin{cases} 0, & \text{if } k \neq j \\ c^2 \delta^2, & \text{if } k = j \end{cases} \end{aligned} \quad (3)$$

The first part assumes that the luminance level of original video is almost invariant in a small block¹ and thus our demodulation can work regardless of unknown V . The second part reaches its maximum only when $j = k$, otherwise it approximates to zero.

Note our design is different from the typical use of CDMA, where each bit is spread with one specific code (vector). Instead, we select a set of orthogonal codes (the selection criteria given as

¹This assumption holds for most cases as Block size is small (e.g. 12×12 in our implementation) especially on HD videos. If the video indeed contains rich texture, error detection may result. This is confirmed in our experiments and we handle such cases with error protection.




Matrix size	Subset
2*2	
4*4	
8*8	

Table 1: Examples of the matrices used in INFRAME++. Black indicates 1 while white indicates -1.

below) and map them into a bit sequence. For convenience, we assume the Cell size $c = 2^n$. We generate a set of orthogonal vectors of size 2^{n+1} using Walsh functions [2] and arrange each into a matrix. We then choose a subset as desired CDMA code that satisfies the following constraints:

1. The length of contiguous 1 or -1 vertically or horizontally is no more than a given number η .
2. Each matrix is unique subject to circular shifts. That is, a code cannot be identical to a circular shift of any other matrices in the subset.
3. Each matrix is unique subject to reverse operation (i.e. interchange 1 and -1). That is, a code cannot be the exact element-wise opposite of any other matrices in the subset.
4. The absolute Hamming distance of any two different matrices in the subset is 2^{n-1} .

We define the *absolute Hamming distance* of matrix A and B as follows.

DEFINITION 1. Let $d(r, l) = h(\text{cshift}(A, r, l), B)$, where h is the Hamming distance function, and cshift is the circular shift matrix of A by r rows and l columns. The *absolute Hamming distance* is the minimum value, $\min_{r,l} \{N/2 - |N/2 - d(r, l)|\}$, where N is the matrix size.

Each of above constraints has certain implications. The first constraint is to ensure invisibility. A code/matrix represents a visual pattern. We discover that we will not perceive flickers if the continuous cell is small enough (see §6.1). So η should be set to an appropriate number. The second constraint is to tolerate possible imprecise Block locating (e.g. Block position shift) at the receiver, which may cause a detected Block to contain partial contents from two neighboring actual Blocks. The third constraint is due to the complementary frame design, and it is not possible to know which frame (e.g. $V + D$ or $V - D$) is captured. We thus need to avoid opposite patterns within a frame. The last constraint ensures that the chosen matrices are *optimal* in a sense that they are most different from each other, since the value 2^{2k-1} is the maximum absolute Hamming distance of any two $2^k * 2^k$ binary matrices. Hence, this modulation scheme is resilient to errors caused by blurs or distortions.

Modulation operation. In INFRAME++, we modulate each block by the above scheme. Generally, we map K bits to a block containing $2^n * 2^n$ cells. We set $\eta = 2$ and write a program to search for the subset. Table 1 visualizes the qualified matrices when n and K both are chosen from 1, 2 and 3, where black and white color corresponding to element ‘1’ and ‘-1’, respectively. The matrices (i.e. codes) governs the modulation and its element indicates the modifications we will apply to a Cell. If the matrix element is 1,

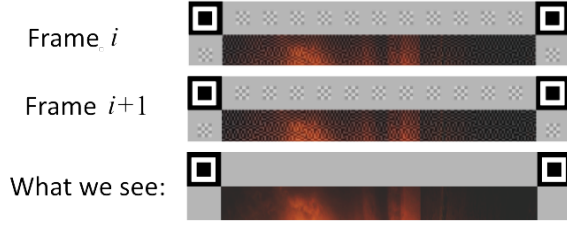


Figure 9: Illustration of frame locators and alignment Blocks along border.

then the luminance level of the Cell (all the $p * p$ physical pixels) increases by δ . Otherwise, it decreases by δ . The spatial coding efficiency – *i.e.*, how many bits are carried per Cell or pixel – can be computed as K/c^2 (per cell) or $K/(cp)^2$ (per pixel). To ensure high throughput, it should be tuned high while keeping the error rate low.

Demodulation in INFRAME++. Basically, demodulation works by comparing, and selecting the maximum, the inner-product of the received Block and all the possible codes, as given in Equ(2). However, the absolute value of the cross-correlation has to be used to cater for the (temporal) complementary frames. In addition, we take the extra measures to address some practical issues. First, the captured Block size may be different from code matrices, we need to match their sizes via down-sampling (mostly) or up-sampling. Assume original Block size is $cp * cp$ (in physical pixels). For an $M * M$ captured Block, we need to resample it and restore a $cp * cp$ Block for correlation matching. In practice, we may use the nearest mapping strategy for computation efficiency. The (i, j) -th element will take value of the $(\lceil \frac{i * M}{cp} \rceil, \lceil \frac{j * M}{cp} \rceil)$ -th element of the captured Block. Second, we perform a local search in a window of $[-2, 2]$ pixels in both X and Y dimension around the predicted Block position to counteract possible block shifting caused by inaccurate Block locating (see next). We thus obtain multiple inner products and will pick the maximum one.

5.2 Robust Block Localization

Despite the CDMA-like modulation and the search process adopted in correlation-based demodulation process, to achieve good performance, a prerequisite condition is to accurately locate the Blocks, as the search process only does local refinement. The Block locating method also needs to be robust to possible channel distortions.

Locator and alignment patterns. In INFRAME++, we follow the practice of a standard QR code [16] design but adapt the design to serve our needs. In particular, we adopt the standard *locator* of a QR code for its proven robustness and ease of detection. A locator is a regular square with each border is arranged with 1:1:3:1:1 for black, white, black, white and black in turn, as shown at corners in Figure 9. We use four locators, one at each frame corner, whereas the QR code needs only three [16]. While we can make locators invisible in principle, we deliberately make them visible in INFRAME++, for several considerations: 1) robust detection of locators is crucial; 2) we need to make user aware of the existence of the side screen-to-camera channel; 3) we can achieve fast locating by avoiding correlation-based detection; and 4) putting them at corners is not distractive anyway.

We also adopt *alignment patterns* along the four borders of the screen (whereas the QR code applies on the top and left borders only). Unlike locators, the border alignment patterns are made invisible. We use the regular chessboard pattern that contains $6 * 6$

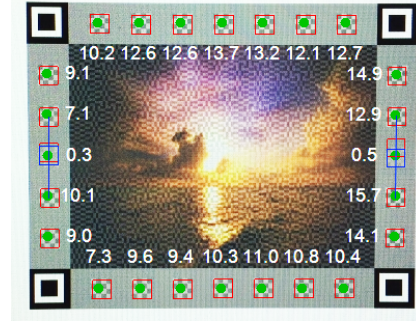


Figure 10: The locating procedure of alignment Blocks. Green points are estimated positions of alignment Blocks, and red rectangles are final determined positions. White numbers are variances of correlation values. The third Block in the left border and corresponding one in the right border are not clear due to rolling shutter effect, thus leading to a low variance (0.3 and 0.5). Their positions are re-calculated by interpolation. Note that locators and Blocks are enlarged for clearer visual.

alternating Cells ($\pm\delta$), using the double width and height of a regular cell. For sake of clear presentation, we term them as *alignment Blocks*. Clearly, alignment Blocks are also spatial complementary patterns. In our design, alignment Blocks are evenly distributed along the borders and take half of space, as shown in Figure 9.

Alignment Blocks are designated to handle potential geometrical distortions, such as radial distortion of display, pincushion distortion of camera, and perspective distortion from imperfect capture, *etc.* Since they serve as references to real data Blocks, special care is taken to ensure accurate identification of alignment Blocks. Note that, like normal data Blocks, alignment Blocks are detected using correlation matching and a local search process is also exerted. We thus design the following logic to infer their positions. First, we calculate their theoretical positions from the positions of frame locators. We then try to detect the alignment Block and record the intermediate correlation values obtained in the search process. If the variance of those correlation values exceeds a threshold, we treat the one with maximum correlation as the actual alignment Block; otherwise, we treat it as a miss detection. Finally, the positions of missing alignment Blocks are re-calculated through interpolation using nearest successful neighbors. Figure 10 shows an example. The rationale is that, if an alignment Block is indeed corrupted, the search process will result in similar correlation values, hence no clear winner and small variances.

Block locating. Ideally we can readily locate the data Blocks using the positions of alignment Blocks. However, channel distortions can be non-linear and non-uniform, *e.g.*, pincushion distortion from a camera. Simple linear locating method can thus lead to erroneous locating results. The locating schemes introduced in [30] are also not applicable since it is difficult to differentiate locators from the original video content. In INFRAME++, we locate a data Block using the following algorithm (for Y coordinate), taking as input the coordinates of frame locators and alignment Blocks on the borders. The key idea is to consider possibly different distortion ratio of the left border (*left*) and the right border (*right*). The algorithm for determine the X coordinate is similar.

$$left = \frac{border_{left}[j].Y - corner_{left-top}.Y}{corner_{left-bottom}.Y - corner_{left-top}.Y}$$

$$right = \frac{border_{left}[j].Y - corner_{right-top}.Y}{corner_{right-bottom}.Y - corner_{right-top}.Y}$$

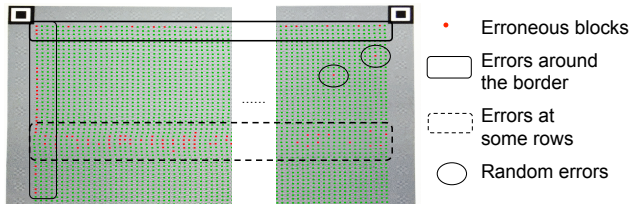


Figure 11: Illustration of the error distribution without error protection mechanism where a CDMA-like modulation is applied only. The red points represent wrong blocks and the green marks the correct blocks after demodulation. Note that the middle parts of both are removed to save space.

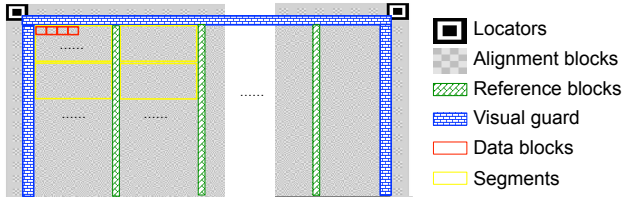


Figure 12: Illustration of one INFRAME++ data frame layout.

$$r_X = \frac{border_{top}[i] \cdot X - corner_{left-top} \cdot X}{corner_{right-top} \cdot X - corner_{left-top} \cdot X}$$

$$r_Y = left * (1 - r_X) + right * r_X$$

$$Y = border_{top}[i] \cdot Y * (1 - r_Y) + border_{bottom}[i] * r_Y$$

5.3 Error Resilience

We analyze the error distribution (see Figure 11 for an illustrative example) and identify three common types of errors: 1) errors around borders; 2) errors at specific rows; and 3) random errors. They match well with observations in other related work. The first is mainly caused by more severe blurring and distortion at borders [11]; the second results from the rolling shutter effect [14]; and the last can arise from the interference from primary video content and many other factors [15, 30]. We propose several techniques to make INFRAME++ more resilient to errors. They induce extra overhead or redundancy, which is a trade-off for reliability.

Visual guard. Instead of taking big efforts to recover data at highly erroneous areas, we choose not to send data there. We thus devise visual guard by leaving blank area around borders, as shown in Figure 12. At the guard areas, the original video content is untouched. This provides a natural isolation (guard interval) between data Blocks, border alignment Blocks and even background (out of the screen). This measure wastes some resources (guard areas do not carry bits) but the overhead is comparably small for the full-screen data delivery.

Channel reference Blocks. Due to inherent rolling shutter effects, the screen-to-camera channel are selectively lossy. It is critical for us to self-learn and respond to the varying channel quality. Borrowing the concept of reference symbol in wireless communications, we place channel reference Blocks (with known patterns) at specific known positions. We insert them evenly into data Blocks (for example, every x rows and y columns, as shown in Figure 12). This allows us to estimate the error rate at specific rows (or areas) from these reference Blocks. Once the error rate ($\frac{|error(Ref)|}{|Ref|}$) is larger than some threshold (say, 0.3), we discard those rows (or areas) and treat them as an erasure. As data frames are repeatedly sent with multiple complementary frames, we can fill up erasures from subsequent frames, or through error protection coding. Note

that techniques [14] to combat rolling shutter effects complements our design.

Error detection and coding. We apply channel coding to enhance its resilience to random errors. To this end, we introduce one additional structural unit – *Segment*. A Segment consists of a group of neighboring Blocks (see Figure 7(a)) and is the minimum unit for error coding. Within a Segment, we apply simple but common error detection and/or correction code such as parity-check, forward error coding (FEC) [33], and RS code [32]. Further framing optimizations [30] are permitted, *e.g.*, to handle erasure. They complement our design.

5.4 Lessons and Discussion

We have tried several techniques proposed in conventional screen-to-camera communication to boost throughput of INFRAME++. Some failed or offered limited help. We summarize several lessons we have learned and discuss potential improvement.

First, individual color channels are hardly useful for INFRAME++. It was expected to double or even triple data throughput if each R,G,B color channel could carry data independently, as demonstrated in [11]. We found three-fold reasons: 1) energy leakage and cross-corruption among channels when being captured by a camera; 2) impact of the primary video; and 3) only very small delta amplitude can be used. The latter two become more an issue due to the former phenomenon.

Second, we noticed that the data-carrying ability depends on the video content. Certain dark and slow-moving portions are able to carry more bits. This shows room for potential improvements through video content adaptation. However, due to heavy computation overhead to analyze the video, this technique is not appealing to real-time rendering. But it is possible to do it offline, especially when the data is pre-defined and paired with the video. In this work, we focus on a generic scheme. We leave the adaptive design for future work.

Third, we admitted that the above scheme for resilience, including visual guard, channel reference blocks, and error coding, indeed lowers the achieved throughput (as shown in §6). The parameters used serve as knotting points between the throughput and the error rate. They should be adapt to dynamic environments (camera capability, visible channel quality, distance) and different communication requirements (tolerance to errors) as well. However, in this work, we used a set of pre-configured parameters and they may not achieve the best performance in all the test environment. We leave the adaptive design as future work.

Fourth, INFRAME++ has two trigger manners during video watching. The first uses an automatic trigger where the video processing will detect whether INFRAME++ is adopted in the video through detecting of four locators. The second is a manual one where the user clicks a button or opens an INFRAME++ receiver app after knowing a INFRAME++-enabled video in advance. In this work, we use the latter and all the videos are INFRAME++ enabled. The automatic discovery yields better usability (free of human efforts) but requires more resource for continuous processing. Energy efficiency and detection error handling are other questions that we need to address in more practical scenarios.

6. IMPLEMENTATION & EVALUATION

We implement the prototype of INFRAME++ with about 5000 lines of code. It consists of a sender and a receiver. The sender takes an original video stream and a data bitstream as its input, generates the multiplexed stream, and then plays back the video stream at precisely controlled frame rate. The receiver takes the captured



Table 2: The screenshots of major test video clips.

frames as its input, detects the existence of visual patterns, and decodes the recovered, possible corrupted frames into the output data. For real-time rendering, the sender is mainly implemented in GPU. Specifically, we use C# to write most logical part of the program, but also use the *DirectCompute* [21] techniques in DirectX 11 to realize the multiplexing and rendering in parallel in GPU. As a result, it supports real-time playback under 120FPS frame rate in all of our experimental settings. Currently, the receiver works in an offline mode. We use the smartphone camera to capture the video frames and run processing afterwards.

We evaluate INFRAME++ in two aspects. First, we conduct a user study to validate whether our design ensures normal screen-to-human viewing without quality degradation or interference. Second, we assess its data communication performance (*i.e.*, throughput, bit error rate and processing speed).

Experimental Settings. We use an Eizo FG2421 24' LCD monitor, which supports 1920×1080 spatial resolution and 120FPS frame rate. In our experiments, we set the brightness as 100%. On the receiver side, we use three phone models (Lumia 1020, Samsung Galaxy S5 and Note 3) to capture the display. The video-capturing resolution and frame rate are set to be 1920×1080 and 30FPS, respectively. We use a pseudo-random data generator with a pre-set seed to generate the input data stream. All experiments are conducted in typical indoor office settings, at the default watching distance 60cm (a desktop width). We have conducted an extensive study to assess the impacts of various design choices and environments conditions, including luminance contrast range δ , cell size c , count of transitional frames τ , CDMA block scheme, and data frame rate, as well as scenario settings such as video sources and watching distance.

6.1 Subjective Perception Assessment

We conduct a user study to evaluate the subjective perception quality and identify a good set of design parameters for INFRAME++. We invited 15 volunteering participants (college students and company employees) in USA and China, 6 female and 9 male, aged between 21 to 36, with half of them wearing glasses. Among the participants, there were a designer and a video expert, who are more sensitive to video quality. We showed original and revised videos side by side, and asked them to rate the flicker (video quality change) with scores 0 to 4, where 0 indicates “no difference at all,” and 1 to 4 signifies “almost unnoticeable,” “merely noticeable,” “evident flicker,” and “strong flicker or artifact,” respectively. In our user study, 0 and 1 denote satisfactory scores. In some tests where the observation is highly consistent among testers and anticipated (*e.g.*, testing with a smaller $\delta = 10$ when invisibility at a larger $\delta > 20$ is validated), the participant scale varies from 6 to 15.

Video sources. We first test with a variety of video sources to examine whether our scheme can be widely applicable. We select 12 video clips with different characteristics on brightness, contrast, texture and motion. Table 2 shows the screenshots of five representative video clips. We employ the pure colored video with dif-

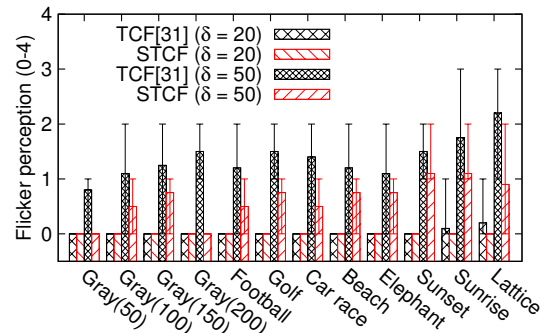


Figure 13: Flicker perception (avg, min, max) in tests with various background videos/images.

ferent brightness where their RGBs vary from (50, 50, 50) to (200, 200, 200) (from dark to light). The pure colored video is adopted for its ease to detect any visual artifact. Other clips are manually classified into several categories: (1) contrast: strong (sunset, sunrise, beach), mild (golf, car race, elephant) and weak/no (football, lattice); (2) motion: fast (football, golf, car race), mild (beach, sunset), slow/static(elephant, lattice); and (3) regular texture (lattice).

Figure 13 show the perceived flicker scores using STCF, compared with TCF [31]. We use a Cell of $3 * 3$ pixels and a Block of $4 * 4$ Cells, that is, $12 * 12$ pixels. We test with two luminance adjustment ranges ($\delta = 20, 50$). That is, the amplitude in STCF is $\pm\delta/2$ whereas the one in TCF is δ or 0. More options are evaluated in the following tests. We make three observations. First, it is viable to alter different video frames without noticeable flickers. When STCF is adopted, most scores are 0 even when δ is as large as 50. Second, the visual perception slightly changes under different brightness, motion and contrast characteristics. Generally speaking, it is easier to notice flickers in a brighter, faster-moving video with stronger contrast. Third, the adoption of spatial complementary patterns greatly alleviates flickers than the pure temporal one. This allows INFRAME++ a larger choice room to chase for good data communication performance. By default, we choose gray and sunset as the source videos.

Luminance adjustment range δ . Figure 14a present the averaged flicker scores as δ varies from 10 to 70. For clarity, we provide the max-min score in the STCF (gray) case. As expected, the smaller the adjustment, the fewer flickers. Even when δ goes up to 50, the resulting visual quality change is still mostly imperceivable for both schemes. However, STCF is much more robust, with fewer flickers under the same condition. Considering invisibility is a subtle and subjective perception (the score is split when $\delta \geq 40$), and the changes should be significant enough to be captured. We believe that $\delta \in (20, 40)$ is a safe choice.

Cell size and distance. Figure 14b shows the perceived flicker with regards to various cell sizes when $\delta = 20$, at a watching dis-

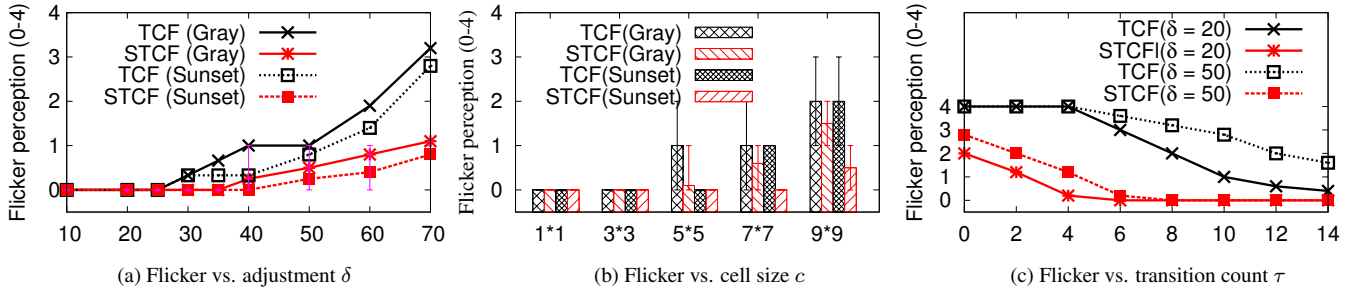


Figure 14: Perception degradation (flickers) versus design options of adjustment range δ , cell size c and transitional frame count τ .

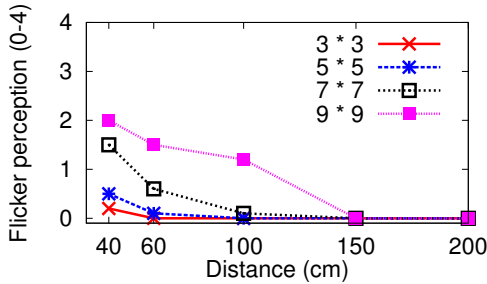


Figure 15: Flicker vs. distance when STCF is used for a gray background video.

Factor	Invisibility	Data	Choice
contrast range $\delta \uparrow$	$\downarrow (< 60)$	\uparrow	20 – 40
cell size $c \uparrow$	$\downarrow (< 7 * 7)$	\uparrow	3 * 3
transition $\tau \uparrow$	$\uparrow (< 4)$	\downarrow	4
distance \uparrow	\uparrow	\downarrow	N/A

Table 3: Summary of factor impact and design choice for INFRAME++.

tance of 60cm. Clearly, the smaller size, the fewer flickers. STCF allows up to $7 * 7$ without noticeable flickers, at a 60cm distance. In this case, the user is more sensitive to the pure color (gray) because the contrast is more obvious. We further test with different watching distance. It is prone to more flickers at a closer distance. When the distance is larger than 150cm, no flickers are noticed. All these match with our common understanding of spatial resolution. However, we want to point out, while it become harder for us to see it, so does the camera. This will be observed in the following evaluation of data communication.

Transitional frame count τ . Figure 14c studies the impact of temporal smoothing cycle τ . Longer cycles tend to reduce the perceived flickers. It is clearly seen that STCF has made significant improvement over TCF. INFRAME++ needs at most 4 transition frames (2 are OK) to handle dynamic data while INFRAME [31] needs at least 10 frames even when $\delta = 20$.

Summary. Our assessment validates that INFRAME++ is able to achieve our first goal of no impaired viewing experience. Table 3 summarizes the parameter impact. All match with our expectation. Basically, a smaller δ , a smaller cell c , and a larger τ tend to hide changes. Considering the requirements for data communication and invisibility robustness, we select design parameters given in Table 3. Here, we set the cell size $3 * 3$ because the good choice for invisibility should be no larger than $7 * 7$ and CDMA-like mod-

ulation may need to concatenate η same cells. When we set $\eta = 2$, it is safe to use a $3 * 3$ cell, though a larger cell is still allowed in some cases.

6.2 Data Communication

We run experiments to evaluate the performance of each design module and its factor impact. We divide INFRAME++ into three module sets: (1) *Basic*, which includes only visual modulation (§5.1) and block localization (§5.2). They are needed to satisfy the minimum requirements for data communication. With them, we are able to retrieve data bits directly from the captured patterns, but it may be prone to a high error rate, without any enhancement from the error protection measures (§5.3); (2) *Basic+Guard+Reference*, also marked as “w/ GR”, covers two more modules to handle errors caused by distortions over screen-to-camera channels. Both are used to manually erase the errors which occurs with known patterns (e.g., at the border or at several rows). As a result, it likely reduce the error rate but also reduce the available throughput. (3) *Basic+Guard+Reference+Code*, short for “w/ GRC”, is a full-version INFRAME++, including all other modules such as channel coding. We deliberately treat channel coding as an independent module because we just apply the well-established technique. Its function is well known to make a tradeoff between redundancy and reliability. We first use *Basic* INFRAME++ to test the performance of demodulation and block location. We then compare with w/ GR to assess the performance impact of virtual guard and channel reference block. Finally, we run comprehensive experiments. We run experiments using the following default setting unless explicitly specified. We use the pure gray color and sunset as our default video sources. In each test, we offline generate 120 multiplexed frames from 30FPS source video clips and 10FPS data frames (randomly generated), and play for 1 second on a 120FPS monitor. That is, each data frame has 12 replica (including its complements), where 4 out of them are transitional frames ($\tau = 4$). Other parameters learnt from the user study is $\delta = 40$ and $c = 3$. The smartphone captures the screen at a distance of 60cm, using ISO 200 at the rear camera. We run experiments 9 runs and show the median results.

CDMA code design. The visual modulation module has two key parameters: the code size (matrix size) and the size of the code block set, which determines the number of bits carried in one block. We choose the first 2, 4, 8 (if applicable) codes using a matrix of size $2 * 2$, $4 * 4$, or $8 * 8$ in Table 1. Specifically, given a $3 * 3$ Cell, each Block has $6 * 6$, $12 * 12$, $24 * 24$ pixels, respectively. Figure 16 shows their BER and raw throughput. Note that raw throughput (throughput hereafter) is not the achievable one in practice, due to the overhead caused by errors. It counts all correctly received bits and it can be simply calculated as the product of the source rate and $(1 - \text{BER})$ if no erasure happens. But it can be regarded as

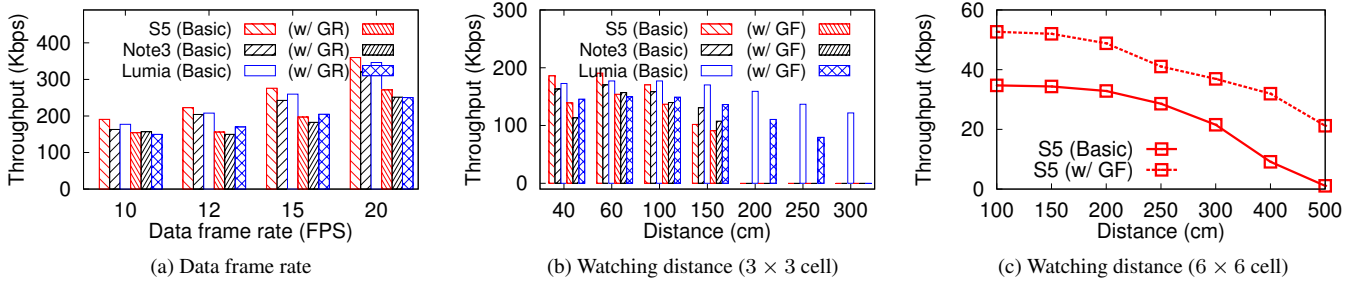


Figure 18: Throughput w.r.t. different design parameters or settings: (a) data frame rate, (b) distance (a small 3×3 cell), (c) distance (using a larger size cell).

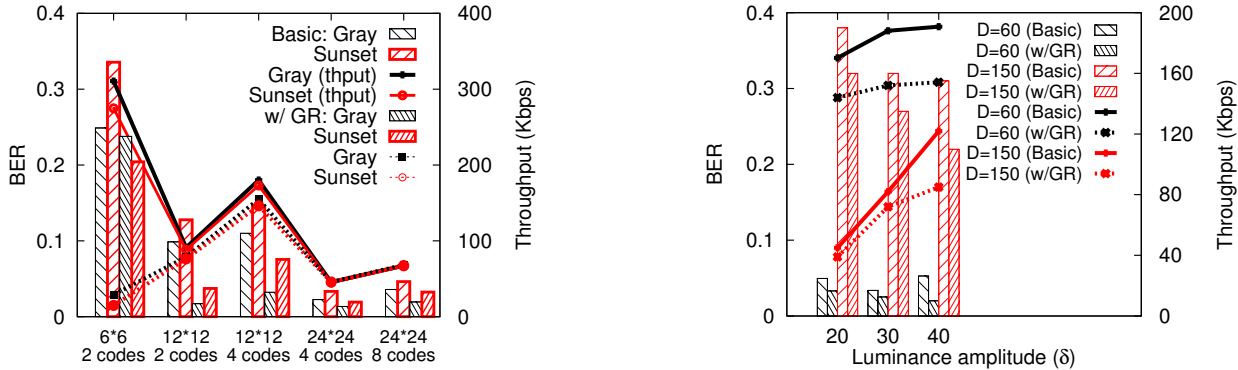


Figure 16: Raw BER (bars) and throughput (lines) under various modulation designs in *Basic* mode and *w/GR* mode where the modules of visual guard and channel reference are employed.

Figure 17: BER and throughput vs. δ .

an indicator of the potential (upper bound) of achievable throughput, and is often used to evaluate screen-to-camera communication. We present the results using Lumia 1020 and the ones using other phones (S5 and Note3) are similar.

We have three observations. First, the larger block size can help to reduce BER when no measures of visual guard and channel reference are employed. The 6×6 block suffers from the highest error rate, as large as 30%. That is because only a 2×2 CDMA code is used and its correlation fails to reliably distinguish itself from the noise of primary video and inevitable distortions. As the block size grows, the orthogonal property is fully utilized. Its raw BER thus reduces to 10% (12×12 block) and 3–4% (24×24 block). Given the same block size, the one with fewer codes performs slightly better. This contributes to fewer block competitors. Second, with the error handling of visual guard and channel reference, the BER greatly reduces, especially when its raw error rate is slightly large. Especially, for a 12×12 Block, it reduces from 10–15% to 1–7%. It implies that many error happen at the border or are caused by rolling shutter effects. In fact, we also take a photo, not a video to capture the screen. We observe that the error rate is much lower, which also indicates that the rolling shutter effect is a major contributor to errors. Third, the impact on throughput is multi-facets. For 12×12 and 24×24 blocks, the throughput almost linearly increases w.r.t. the size of block set (*i.e.* the code number). It is easy to understand. CDMA codes are orthogonal and it can tolerate the existence with other codes. Though the 24×24 case has a lower error rate than the 12×12 one, its spatial efficiency (determining

throughput) decreases due to more spaces to carry a unit bit. For instance, the 24×24 occupies $4 \times$ area but only doubles the throughput. As a consequence, the 12×12 outperforms 24×24 and other larger block design. The exception exhibits between 12×12 and 6×6 blocks. Ideally, the 6×6 has higher raw throughput but is hard to utilize due to a large error rate. So in the following experiments, we mainly use 12×12 Block with 4 codes (*i.e.*, 2 bit/block) as our default scheme. When we use the only one modulation scheme, the error rate and its throughput can be inferred. So we only plot the throughput due to space limit.

Luminance adjustment range. Figure 17 shows the results when δ varies from 20 to 40. We present only the sunset result because it is more challenging than the gray one. Basically, as δ increases, the throughput increases. At a near distance (60cm), no significant improvement is observed (about 20kbps gap). At a farther distance (150cm), a larger δ is of more help. Here we present the result for S5 because its performance degrades more at 150cm (see Figure 18b). This matches with our understanding. It also discloses that INFRAME++ offers a quite flexible room to fulfill its function. The gap between the basic and the w/GF implies a high erasure rate. We checked the captured videos and did observe strong rolling shutter effects. While INFRAME++ can detect the error through the reference block, it still requires extra mechanism to handle and recover from it. This can refer to recent work done by [14].

Data frame rate. Given a 120FPS fresh rate and a data frame rate μ , each data frame will have $120/\mu$ replicas, out of which $\tau = 4$ frames are transitional frames. Figure 18a shows the throughput when the data frame rate increases from 10 to 20, that is, from

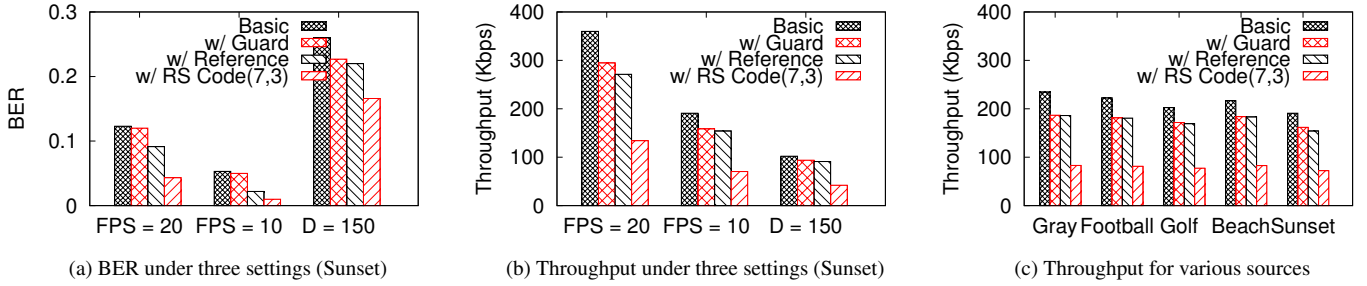


Figure 19: BER and throughput after each module in INFRAME++.

12 replicas to 6 replicas (minimum). Here, we only present the result for the sunset video, and the gray one has about 10–20Kbps improvement. We observe that the data rate steadily increases w.r.t μ . It first implies that the camera is not a bottleneck, and it is able to capture data frames even it moves faster. Theoretically, it can support 360Kbps (*Basic*) and it still able to support as high as 270Kbps after the erasure made by visual guard and reference blocks. This demonstrates a significant improvement as high as 30–45x improvement than our prior work [31]. This is thanks to the adoption of STCF which allows a smaller number of transitional frame and a larger δ . The CDMA design which carries more bits in one block, also contributes to the throughput improvement.

Capture distance and phone models. Figure 18b shows the throughput at different capturing distance. Basically, the nearer, the better. But we also notice there is an exception. For Note 3, when the capture distance reduces from 60cm to 40cm, more distortion occurs and thus hurts INFRAME++. We also notice that INFRAME++ fails when the distance is larger than one threshold (here, 2m for S5 and Note3, 3m for Lumia). We checked the captured videos and indeed observed that there were no patterns in the one captured by S5 and Note 3 and blurred patterns in Lumia. This matches with our expectation. Constrained by the requirement of unimpaired viewing even when being watched in vicinity, we configure the parameters, such as a 3×3 cell size; However, they can not work well at a far distance. In fact, a larger distance can allow a more flexible parameter choice (e.g. a larger δ and c), which can be leveraged to boost INFRAME++ without interfering video viewing. We further test with a larger cell size (6×6 , not 3×3). Figure 18c shows the throughput for S5 with the watching distance up to 5 meters. The results are similar for Lumia (better than S5) and Note 3 (worse than S5). The current setting can ensure unobstructive viewing experience between 1m and 5m. Clearly, the working distance is extended as the cell size increases. Note the overall throughput is decreased as a result of a smaller block density (declines by 4 fold). This is constrained by the physical size of the display. Under the current display and camera settings, when the distance goes beyond 3 meters, the error rate is pretty high, as large as 20-40%. This calls for a more robust scheme for communication.

Design modules. We now examine the role of each design modules. On top of the *basic* INFRAME++, we in turn enable the visual guard, channel reference blocks and channel coding. We use RS code (7,3) in this case. Figure 19a and Figure 19b show the BER as well as its throughput under three settings (1) default, (2) data frame rate = 20 and the other same as default, and (3) distance is 150 cm and the other is the same as default. We only present the results for S5 when the sunset video is used. Figure 19b shows the throughput results for different video sources using $\delta = 40$,

	PC1	PC2	Mac Pro
Pure gray	136FPS	242 FPS	250FPS
720p video	131FPS	235 FPS	185FPS

Table 4: The processing time for video/image encoding and rendering at three senders.

Threads	PC1	PC2	Requirement (10FPS)
single	981ms	908.7ms	100ms
multiple	269ms	202.7 ms	

Table 5: The receiver’s processing time for video decoding.

$D = 60cm$ and 10 FPS data frame. Clearly, the results are similar across video sources and the sunset video is slightly worse. We can see that all the error protection mechanisms are effective in reducing the error rate. However their cost is different. The channel coding depends on redundancy and thus decreases the throughput most. In most cases, Guard and reference block can efficiently reduce error rate while maintaining a graceful degrade on throughput. Under the default setting, all video sources can achieve about 200Kbps without any extra add-on feature (*Basic*) and eventually can achieve about 70-85kbps after all the error handling mechanisms are applied.

Encoding and Decoding Speed. We finally examine the processing speed which is vital to real-time rendering and communication. We first measure the time to encode data frames, multiplex and render them in the screen. We use the default scheme: 12×12 block with 4 codecs, on the 1920×1080 resolution screen. We test with three machines: (1) PC1 with an Intel Core-i5 dual-core CPU, 8GB main memory and an AMD Radeon HD8490 video card; (2) PC2 with an Intel Xeon e5-1620 four-core CPU, 8GB main memory and an AMD FirePro V3900 video card; (3) Macbook Pro using an Intel Core-i5 dual-core CPU, 8GB main memory and Intel Iris integrated video card. Table 4 shows their processing speeds after they are stabilized. It is clear that the sender is able to support real-time operations since the processing speed is larger than 120FPS. This is attributed to the *DirectCompute* technique in GPU. However, the current processing time at the receiver is a little bit slower than the real-time requirement (Table 5). For 10FPS data frame rate (one new data frame is embedded every 100ms), it takes 200ms or 260ms to decode it, using multiple (4x) threads. The current receiver has not been optimized by GPU and thus it supports offline processing with a several seconds delay. We notice that similar GPU optimization is applicable. We are working on improving the decoding speed.

7. POTENTIAL APPLICATIONS

INFRAME++ is able to provide accompanying data communication without requiring any infrastructure (*e.g.*, Wireless, Mobile Networks). It offers a user-friendly approach to fetch information from an electronic display (*e.g.*, TV, ad board). We present two potential applications empowered by INFRAME++.

Enhanced Video Ad (Ad++). It offers an enhanced video advertising by providing metadata to the potential customers who are interested. The target scenario is to employ INFRAME++ on advertising electronic boards at the shopping malls or out of the buildings. It works as follows. At the screen corner, there is a small visible icon which indicates whether INFRAME++ is enabled or not. If so, the target user either use Google Glass or his phone camera to capture what he may be interested in, but not directly covered in the visible ad. Compare with the existing ad, it has several advantages. First of all, it still ensures the original, attractive video ad to most users. Data delivery is almost imperceptible to users and can be done on demand. For those who are really interested in the goods and want to know more, this offers a convenient channel to fetch extra information. Second, without being constrained by the screen size, it is able to carry on much richer information. The existing ad usually highlights several keywords (*e.g.*, price, phone number), while ad++ allows to carry several KB or more. Third, data dissemination adopts a digital form (bytes and texts) and thus it is ready for automatic processing, mining and archiving which are machine-friendly and user-friendly (no or litter human efforts). For instance, the user does not need to recognize and remember the seller's telephone number and manually type it into the phone. Instead, once the number is retrieved from the captured video, it can be automatically saved or prompted to the user for a call. Fourth, this scheme is infrastructure-free. This combats alternative solutions such as WiFi, Bluetooth, NFC which requires that the source (display) is equipped with extra hardware. Certainly, reliability of INFRAME++ should be greatly enhanced before its commercial adoption.

Watching Authentication. The co-channel delivery enforces a close association between human viewing activity and embedded data. Retrieving specific data from this video likely implies that the users indeed is watching this video. This intrinsic watching authentication provides one natural form to attract or allure real consumer attention, as well as a convenient and secure way to deliver some confidential or target information. For example, coupons or deals can be delivered through screen-to-camera links in INFRAME++. Many manufacturers hope to disseminate coupons to their customers only as a reward of watching their ads. If the coupon is embedded in the video, it will force the customers to watch the real videos. Otherwise, someone may play tricks to return to the ad page after the ad video ends, or through a simple click or a link after the video.

8. RELATED WORK

In this section, we compare INFRAME++ with the literature along three dimensions.

Unobtrusive screen-to-camera communication. Several recent studies seek to establish non-obtrusive screen-to-camera communications over static images, including VRCode [34], IVC [7], visual MIMO [36], and HiLight [19]. Specifically, VRCode delivers only static (tag) content using a hue-based barcode design [34]. [7, 36] hide data upon two consecutive frames (an original image and the one with embedded messages) where data is embedded in pyramid decomposition [36] and brightness change [7]. HiLight

conveys data bits through the pixel transparency change within a time window [19]. Different from their work, we work on the dynamic video content with time-varying visual background, thus facing new challenges. In our solution, we leverage capability discrepancy between users and devices, as well as human vision characteristics when designing INFRAME++. Through the novel use of STCF, CDMA-like modulation and error protection, INFRAME++ has not only addressed more practical system issues, but also achieved higher data rate (360kbps with an 30-60x growth of our prior work [31]).

Conventional screen-to-camera communications. They focus on data communications while exclusively occupying the (entire or part of) the screen [11, 14, 15, 22, 30]. They thus differ from our primary goal of establishing dual-mode, concurrent visual channels for data communication and video viewing. Nevertheless, their proposed techniques, which address challenges inherent in screen-to-camera channels such as rolling shutter effect [14], dynamic capture quality [15], error handling [30], can be applicable to INFRAME++.

Watermarking and Steganography. Watermarking and steganography also embed data into signals (*e.g.*, images or videos) in a covert manner [8, 23, 37]. The main approach to data information hiding is to make the altered signals as close to the original ones as possible. For example, a common technique is to manipulate only the LSB (least significant bit) of pixels either in a deterministic fashion [5, 17, 20, 35] or in a randomized manner [9, 12, 26]. Watermarking is designed primarily for authenticity verification or integrity checking, whereas stealthiness is of less interest [37]. INFRAME++ differs from these two topics in both its goals and the target scenario. We seek to enable imperceptible data communications, rather than data hiding or authentication. Therefore, INFRAME++ must address real-world degradations over screen-camera channels while they do not.

9. CONCLUSION AND FUTURE WORK

In this paper, we describe INFRAME++, a system that enables simultaneous full-frame video-viewing experiences for users and screen-camera communications for devices, through multiplexing original video frames and data frames and rendering them on the same display. The proposed design exploits the flicker-fusion property of the human vision system and the superior capability of modern displays and cameras. Our prototype and evaluation have confirmed the viability of INFRAME++.

We believe that INFRAME++ explores a new paradigm for visual data communication. It retains the normal viewing quality for users while enabling high-rate visual communication to devices. From the comparative standpoint, it circumvents the distractive and hard-to-use downsides in the current barcode design. As the capability discrepancy between users and devices continues to expand, we envision the steady rise of opportunistic communications over the device-to-human visual channels. It offers a new venue for visual data communication to devices in the emerging cyber-physical world.

ACKNOWLEDGMENTS

We greatly appreciate our shepherd, Dr. Prabal Dutta, and the anonymous reviewers for their insightful comments and constructive feedback. We also thank Jiaqi Xu, Ouyang Zhang and all other participants in the user study. This work is supported in part by the National Science Foundation under Grants No. CNS-1421933 and CNS-1421440.

10. REFERENCES

- [1] Cisco Visual Networking Index: Forecast and Methodology, 2013–2018. <http://tinyurl.com/mev32z8>.
- [2] Walsh function. http://en.wikipedia.org/wiki/Walsh_function.
- [3] Wechat.
- [4] D. A. Atchison, G. Smith, and G. Smith. *Optics of the human eye*. Butterworth-Heinemann Oxford, UK:, 2000.
- [5] R. Balaji and G. Naveen. Secure data transmission using video steganography. In *IEEE International Conference on Electro/Information Technology (EIT)*, 2011.
- [6] G. Brindley, J. Du Croz, and W. Rushton. The flicker fusion frequency of the blue-sensitive mechanism of colour vision. *The Journal of physiology*, 183(2):497–500, 1966.
- [7] R. Carvalho, C.-H. Chu, and L.-J. Chen. IVC: Imperceptible video communication. 2014. Demo.
- [8] A. Cheddad, J. Condell, K. Curran, and P. McKeivitt. Digital image steganography: Survey and analysis of current methods. *Signal Processing*, 90(3):727–752, 2010.
- [9] J. Fridrich and M. Goljan. Digital image steganography using stochastic modulation. In *Electronic Imaging*, pages 191–202, 2003.
- [10] D. G. Green. Sinusoidal flicker characteristics of the color-sensitive mechanisms of the eye. *Vision research*, 9(5):591–601, 1969.
- [11] T. Hao, R. Zhou, and G. Xing. Cobra: Color barcode streaming for smartphone systems. In *MobiSys*, 2012.
- [12] J. He, J. Huang, and G. Qiu. A new approach to estimating hidden message length in stochastic modulation steganography. In *Digital Watermarking*, pages 1–14. Springer, 2005.
- [13] W. A. Hershberger and J. S. Jordan. The phantom array: a perisaccadic illusion of visual direction. *The Psychological Record*, 48(1):2, 2012.
- [14] W. Hu, H. Gu, and Q. Pu. Lightsync: Unsynchronized visual communication over screen-camera links. In *MobiCom*, 2013.
- [15] W. Hu, J. Mao, Z. Huang, Y. Xue, J. She, K. Bian, and G. Shen. Strata: Layered Coding for Scalable Visual Communication. In *MobiCom*, 2014.
- [16] I18004:2000. Automatic identification and data capture techniques - Bar code symbology - QR Code.
- [17] R. Kavitha and A. Murugan. Lossless steganography on avi file using swapping algorithm. In *Conference on Computational Intelligence and Multimedia Applications*, volume 4, 2007.
- [18] D. Kelly. Flicker. In *Visual psychophysics*, pages 273–302, 1972.
- [19] T. Li, C. An, A. Campbell, and X. Zhoun. Hilight: Hiding bits in pixel translucency changes. In *ACM Workshop on Visible Light Communication Systems (VLCS)*, 2014.
- [20] X. Liao, Q.-y. Wen, and J. Zhang. A steganographic method for digital images with four-pixel differencing and modified lsb substitution. *Journal of Visual Communication and Image Representation*, 22(1):1–8, 2011.
- [21] NVIDIA. Directcompute. <https://developer.nvidia.com/directcompute>.
- [22] S. D. Perli, N. Ahmed, and D. Katabi. Pixnet: interference-free wireless links using lcd-camera pairs. In *MobiCom*, 2010.
- [23] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn. Information hiding – a survey. *Proceedings of the IEEE*, 87(7):1062–1078, July 1999.
- [24] N. Rajagopal, P. Lazik, and A. Rowe. Visual light landmarks for mobile devices. In *ISPN*, 2014.
- [25] J. Roberts and A. Wilkins. Flicker can be perceived during saccades at frequencies in excess of 1 khz. *Lighting Research and Technology*, 45(1):124–132, 2013.
- [26] T. Sharp. An implementation of key-based digital signal steganography. In *Information hiding*, pages 13–26. Springer, 2001.
- [27] E. Simonson and J. Brožek. Flicker fusion frequency: background and applications. *Physiological reviews*, 1952.
- [28] R. D. Valois and K. D. Valois. *Spatial Vision*. Oxford University Press, 1988.
- [29] I. Vogels and I. Hernando. Effect of eye movements on perception of temporally modulated light. <http://2012.experiencinglight.nl/doc/28.pdf>.
- [30] A. Wang, S. Ma, C. Hu, J. Huai, C. Peng, and G. Shen. Enhancing Reliability to Boost the Throughput over Screen-camera Links. In *MobiCom*, 2014.
- [31] A. Wang, C. Peng, O. Zhang, G. Shen, and B. Zeng. InFrame: Multiflexing Full-Frame Visible Communication Channel for Humans and Devices. In *HotNets-XIII*, 2014.
- [32] S. B. Wicker. *Reed-Solomon Codes and Their Applications*. 1994.
- [33] wikipedia. Forward error correction.
- [34] G. Woo, A. Lippman, and R. Raskar. Vrcodes: Unobtrusive and active visual codes for interaction by exploiting rolling shutter. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2012.
- [35] H.-C. Wu, N.-I. Wu, C.-S. Tsai, and M.-S. Hwang. Image steganographic scheme based on pixel-value differencing and lsb replacement methods. *IEE Proceedings-Vision, Image and Signal Processing*, 152(5):611–615, 2005.
- [36] W. Yuan, K. Dana, A. Ashok, M. Gruteser, and N. Mandayam. Dynamic and invisible messaging for visual mimo. In *IEEE Workshop on Applications of Computer Vision (WACV)*, 2012.
- [37] Y. Zhang. Digital watermarking technology: A review. In *Future Computer and Communication (FCC)*, 2009.