Smooth Transmission over Unsynchronized VLC Links

Qifan Pu
University of Science and Technology of China
puqf@mail.ustc.edu.cn

Wenjun Hu
Microsoft Research Asia
wenjun@microsoft.com

ABSTRACT

Screens and cameras on off-the-shelf personal devices have become widely available in recent years, which has prompted interest in exploiting the communication channels between them in the visible light band [3,4]. However, the basic issue of frame synchronization in such systems has not been adequately addressed. As a result, the receiver is not guaranteed to detect and capture decodable frames under many settings, and the link would become unusable.

In this work, we address this synchronization issue with a rateless code design across frames. Without modifying the hardware, the firmware or the driver, our application-level software-based approach, Smooth Codes, enables screen-camera pairs with any frame rate condition to transmit and receive data smoothly to achieve high throughput.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless Communication

Keywords

Mobile Computing, Optical Links, Unsynchronized Communication, Visible Light Communication

1. INTRODUCTION

Given the wide availability of off-the-shelf cameras, LCD screens, and smart phones, researchers have been interested in building optical communication systems between them [3, 4]. The screen can display data encoded in rapidly changing image frames, and a camera can capture this stream of data. Users with such personal devices can set up impromptu and secure communication channels for remote control, file sharing, or even ad-hoc gaming. Such Visible Light Communication (VLC) systems are also well suited to multicast, because multiple cameras can easily be pointed at the same screen. This is useful in scenarios like conference material distribution and public advertisements.

Compared with other wireless technologies such as WiFi and Bluetooth, VLC links enjoy easy access, high directionality, and better security, while offering significant capacity. For example, the newly-released Nokia Lumia 920 sports a 4.3-inch, 800×480 display and a camera capable of recording video at 1080p. Theoretically, this phone could transmit data at 740Mbit/s and receive at over 3Gbit/s.

CoNEXT Student'12, December 10, 2012, Nice, France.

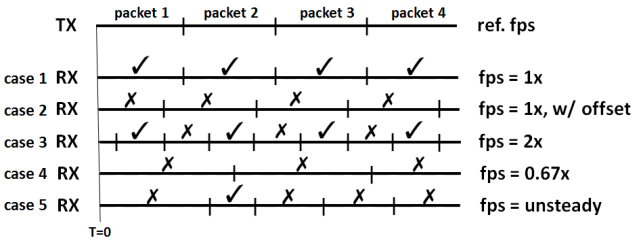

Figure 1: Mismatched FPS between TX-RX pairs. "|" marks the boundary between two consecutive frames. ✗: "mixed" frames, ✓: "clean" frames.

However, real systems are hindered by imperfect frame synchronization, image distortion, and computational complexity, which result in much lower observed data rates of, say, 172 bps [4]. While previous research often focused on reducing image distortion, we try to solve the basic synchronization problem in such a system.

Lack of synchronization between a transmitter and receiver happens mainly due to mismatched frame rates and a phase offset between the screen and the camera. Figure 1 shows several examples. Typically, only frames staying within the period of one transmission are *clean* (marked "✓"), and thus decodable. If a captured spans across different TX frames, it becomes a *mixed* frame (marked "✗") and is normally dropped. Only after receiving all the transmitted frames "cleanly" can a receiver successfully decode the original message, as in cases 1 and 3.

With a traditional coding scheme, a receiver should capture frames at a rate at least twice that of the transmitter's display frame rate to get all frames (case 3). Unfortunately, the opposite often happens in practice. Modern LCD/AMOLED screens usually have a 60 fps frame rate or at least 30 fps, while cameras have lower and more diverse frame rates in the range of 24 to 30 fps. Moreover, many smartphone cameras exhibit unstable frame rates because of the hardware design and firmware settings. Our tests on a series of Android phones show that the camera will automatically adjust its frame rate according to the lighting conditions. It is neither desirable nor possible to control the frame rate at the application level. The system performance can be significantly bottlenecked by the low, fluctuating frame rates at the camera end.

To achieve synchronization, previous systems like PixNet [3] and COBRA [4] simply reduce the screen frame rate to ensure that the receiver can pick out a good frame every other frame. Langlotz et al [2] embed 2D barcodes into time-multiplexed 3D barcodes by repeating frames on multiple, orthogonal color channels, essentially limiting the transmitter's frame rate. These approaches are inefficient because the transmitter side capacity is under-utilized and the receiver drops a large number of "mixed" frames.

This paper introduces rateless coding into a VLC system and describes both an algorithm and a frame format design to solve the synchronization problem. The idea is to encode an original message into a series transmitted frames, such that mixed frames

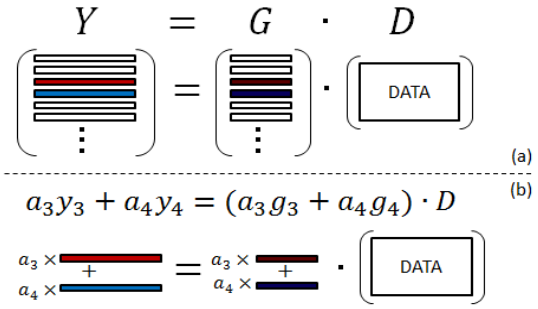

Figure 2: The last step in the whole encoding chain. (a) An encoded frame Y is generated from the original message D with an encoding matrix G. (b) Mixed frames are still useful.

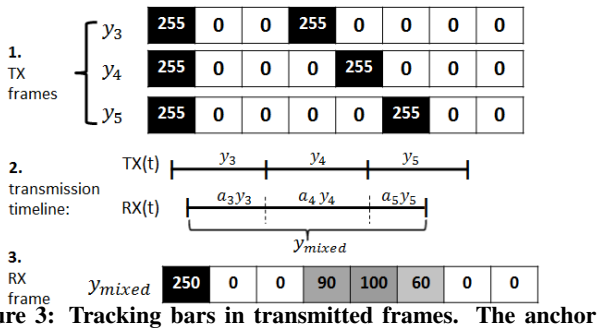

Figure 3: Tracking bars in transmitted frames. The anchor block and a frame-specific tag block are set to 255. The receiver estimates the weights of original frames based on the received values of those blocks. The weights in this received frame are $a_3 = \frac{90}{250}, a_4 = \frac{100}{250}, a_5 = \frac{60}{250}$.

can still be used to recover the original message, provided that the frame indices are properly tracked.

2. SMOOTH CODES

Smooth Codes adds two steps to the conventional encoding process to generate image frames. First, after the original messages are encoded into a data matrix, we generate frames by calculating projections of the data matrix, similar to the rateless coding approach in Strider [1]. Second, each encoded frame is prefixed with a tracking bar to aid decoding. The receiver calculates the weights of the original frames based on the tracking bars and the encoding matrix, and then decodes the original frames.

2.1 Coding Algorithm

The coding process can be represented by a matrix multiplication (Figure 2(a)). The input data message is assembled into a matrix D and multiplied by a pseudo-random encoding matrix G to generate an output matrix Y. Each row of Y is a single TX frame and is transmitted in one time slot:

$$Y = G \cdot D \quad or \quad \vec{y}_i = \vec{g}_i \cdot D$$

G is randomly generated and normalized, so each frame is a distinct projection of the original message. Note that G can have infinitely many rows.

The i^{th} frame in Y is related to the i^{th} row of G alone but not to any other row of G. The equation will hold true no matter how we reorder the rows of Y and G consistently. This means that the receiver only needs enough frames to decode the original data, irrespective of which frames are received and in what order. The number of frames needed is related with the channel distortion and noise level. The encoding matrix G can be generated offline and is known to the receiver. The actual decoding is straightforward, and mainly requires inverting the encoding matrix.

If the receiver gets a combination of TX frames, $\vec{y}_{mixed}$, we have:

$$\vec{y}_{mixed} = \sum_{i=p}^{q} a_i\vec{y}_i = \sum_{i=p}^{q} a_i\vec{g}_i D = (\sum_{i=p}^{q} a_i\vec{g}_i) \cdot D = \vec{g}_{mixed} \cdot D$$

where $\vec{y}_i$ is the i_{th} frame in the combination and a_i is its weight, and $\vec{g}_{mixed}$ serves as the corresponding random encoding vector for $\vec{y}_{mixed}$. Most importantly, $\vec{y}_{mixed}$ is still a distinct projection of D just as any other "clean" RX frame. If the corresponding $\vec{g}_{mixed}$ can be found, we can use the mixed packets instead of dropping them. We now discuss how to get $\vec{g}_{mixed}$.

2.2 Frame Tracking

Although the above encoding process allows us to recover messages from any subset of the transmitted frames, we still need to know the indices of the frames to identify the appropriate rows of the original G for decoding. Furthermore, in the case of "mixed" frames, we also need to track the weight, a_i, of each transmitted frame, then we can calculate $\vec{g}_{mixed}$ from a_i and g_i. In our solution, we obtain these indices and weights by adding tracking bars in each transmitted frame. These are analogous to known training sequences and pilots in 802.11 frames.

Figure 3 shows the structure of a tracking bar. The first block of the tracking bar is always tagged by the same color across frames to serve as an anchor. Then each frame will tag one block from the remaining blocks with the same anchor color. The tagged block index $s_i = mod(y_i, MAX\_BLOCKS)$, where y_i is the frame index, and MAX_BLOCKS is such that a receiver with the lowest capture rate can still distinguish the frames. The receiver can infer the composition of a received frame by estimating the weights of the original frames using the received values of the tracking bar blocks.

3. PRELIMINARY RESULTS

We have implemented a prototype as a proof of concept. Our system consists of a laptop screen as the transmitter, a phone camera as a receiver, and an offline decoder. The Lenovo X201i laptop takes the original data, encodes them with a randomly generated encoding matrix G, assembles the encoded data into frames, and displays them at an arbitrary frame rate. The camera from an HTC EVO 3D takes pictures of the screen with an arbitrary shutter speed, and feeds all the received images into the offline decoder. We vary the frame rates at both ends to assess the decoding quality under a wide range of channel conditions. Initial results show that Smooth Codes achieves a much higher throughput than previous approaches.

In future work, we will aim to build a real-time phone-to-phone VLC system based on Smooth Codes, including an immediate feedback mechanism. We will also build applications leveraging such VLC links.

4. REFERENCES

[1] A. Gudipati and S. Katti. Strider: automatic rate adaptation and collision handling. In *Proceedings of the ACM SIGCOMM 2011 conference*, 2011.

[2] T. Langlotz and O. Bimber. Unsynchronized 4D Barcodes Coding and Decoding Time-Multiplexed 2D Colorcodes. 2007.

[3] S. D. Perli, N. Ahmed, and D. Katabi. Pixnet: Lcd-camera pairs as communication links. In *Proceedings of the ACM SIGCOMM 2010 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. ACM, 2010.

[4] G. X. Tian Hao, Ruogu Zhou. Cobra: Color barcode streaming for smartphone systems. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys)*. ACM, 2012.