Demo: A Robust Barcode System for Data Transmissions over Screen-Camera Links

Anran Wang[1] Shuai Ma[1] Chunming Hu[1] Jinpeng Huai[1] Chunyi Peng[2] Guobin Shen[3]
[1]SKLSDE Lab, Beihang University, China [2]The Ohio State University, USA [3]Microsoft Research, China
{wangar@act., mashuai@, hucm@, huaijp@}buaa.edu.cn chunyi@cse.ohio-state.edu jackysh@microsoft.com

ABSTRACT

Visible light communication (VLC) over screen-camera links emerges as a novel form of near-field communication, and it offers a user-friendly, infrastructure-less and secure communication, which is highly competitive for one-time file transfer [1–4]. However, the limitations of smart devices and the uncertainty of user behaviors seriously impair the transmission reliability and hinder its applicability. Worse still, existing approaches [1,2,4] mostly focus on improving the transmission speed and ignore the transmission reliability. Hence, RDCode is proposed to boost the throughput over screen-camera links, by making use of a novel barcode design and several effective techniques to enhance the transmission reliability [3]. In this demo, we show that our RDCode prototype system addresses many practical challenges.

A short video on our prototype system is accessible from http://mashuai.buaa.edu.cn/demo/RDCode.mp4.

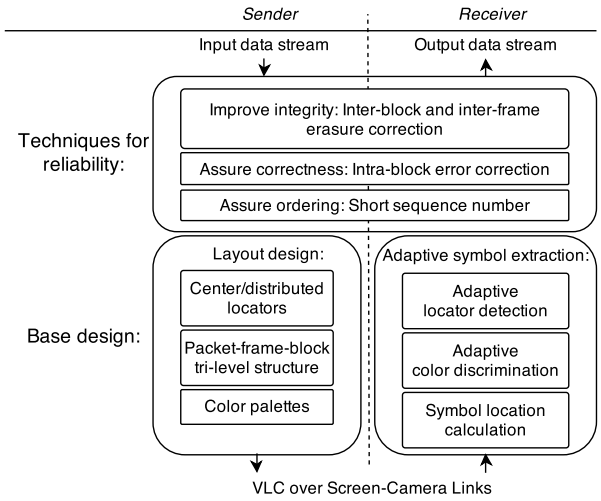

Figure 1: RDCode architecture

1. INTRODUCTION

As the rapid proliferation of camera-equipped smart devices, visible light communication (VLC) over screen-camera links, as a novel form of near-field communication, has provided an attractive solution for its user-friendliness, security and infrastructure-less. Compared with the traditional radio frequency (RF) techniques such as Bluetooth and WiFi, VLC does not depend on the Internet infrastructure and enables direct and secure communications, which simplifies the complicated authentication process for setting up link connections. Hence, VLC is well suited for one-time transfer as it incurs no charge and no overhead in link setup and management, compared with traditional RF techniques [3,4].

To achieve high-speed data transmissions, the *transmission rate* is only one side of the coin. The *transmission reliability* is equally important for many applications such as file transmission. However, previous works [1, 2, 4] mostly spend their efforts on improving the transmission speed, while ignore the transmission reliability. We give an analysis of the typical problems that impair the transmission reliability and summarize these into two categories: smartphone limitations and user behavior uncertainty [3], which introduce two challenging issues: (a) the *locality problem* that for an image with uniform features, the close places in its captured image have similar features, while places far from each other present different features, and (b) the *partial unavailability problem* that leads to either unrecognizable parts of a frame caused by *e.g.,* unsuitable shooting positions or temporally sequential frames loss caused by *e.g.,* trembles [3].

As a first step towards solving all these problems, we have proposed RDCode [3], a robust barcode system for high-speed data transmissions over screen-camera links. The architecture of RDCode is shown in Fig. 1. Finally, we implement the RDCode prototype system as a file transmission App on an Android platform.

2. RDCode DESIGN

In this section, we briefly introduce our RDCode design. RDCode is a VLC system designed for high-speed data transmissions over screen-camera links, and it consists of three layers. (a) The first layer is the barcode layout and its corresponding locating and data extraction methods, which lay a foundation for reliable transmissions; (b) The second layer is the error correc-

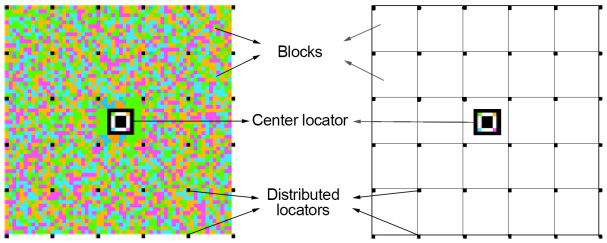

Figure 2: The frame layout of RDCode design

tion and data ordering methods to enhance the reliability; And (c) the last layer is for a specific application, *i.e.,* file transmissions in our implementation. These together enable RDCode to reach a high transmission speed by enhancing the transmission reliability.

2.1 Barcode Layout

RDCode has a novel tri-level barcode layout such that (a) a packet comprises a sequence of frames, (b) each frame consists of a set of independent blocks, and (c) each block is a square of symbols, shown in Fig. 2.

To increase the locating success rate for receivers, a frame of RDCode contains a single specifically designed locator at the center block, referred to as the *center locator*, and each block in a frame also contains another pattern (*i.e.,* a single black symbol) at the left-top corner of the block, referred to as the *distributed locator*.

We incorporate an adaptive locator detection method into RDCode to recognize the barcode sequence. It first searches and identifies the center locator, then makes use of a best-effort approach to identifying all the distributed locators, by taking advantage of the positions of the located neighboring locators of a distributed locator to speed up the detection process. It is easy to know that the four distributed locators around a block can determine the positions of the symbols in the block. The adaptive techniques for symbol extraction improve the locating and decoding accuracy, by using successive frames in our dynamic barcode system.

The barcode layout of RDCode along with the adaptive locating methods lay a foundation for reaching high-speed reliable data transmissions.

2.2 Data Protection Techniques

Based on an analysis of the spatial and temporal distributions of errors, we apply (a) the Reed-Solomon (RS) code for block level error corrections, and (b) two kinds of parity-check codes for frames and packets, respectively. Our experiments show that the error correcting method behaves much better than using the Reed-Solomon code alone. In order to keep the block and frame ordering, we further put a short header denoting the relative sequence number in each block, and the full sequence number can be calculated by looking up a certain kind of meta data in the center block.

2.3 File Transmission Techniques

We finally build a layer on the top of RDCode for file transmissions. A file is treated as binary data, and read into a bit array that keeps repeating in the source data stream. During the transmission process, we need to keep the data intact and complete within a short time. Hence, we propose an efficient optimization to encode the packets into a combination of these packets in certain order. This significantly reduces the extra transmissions when the file is not completely received in the first transfer.

Remarks. RDCode significantly improves the transmission speed by enhancing the transmission reliability. (1) The barcode layout and adaptive symbol extraction methods are in nature to address locality problems and partial unavailability problems, since different blocks are autonomous. They are located, recognized and decoded independently and asynchronously. As a result, the data reliability is improved. Moreover, our adaptive techniques improve the locating and decoding accuracy, and hence increase the data capacity per frame as well as transmission speed.
(2) The novelty of the data protection techniques lies in that instead of simply applying a single Reed-Solomon code like most barcode systems, we analyze the characteristics of data errors, utilize our tri-level barcode layout and design a tri-level error correction method to decode the decodable parts of data as much as possible to increase the data integrity.

3. RDCODE DEMONSTRATIONS

In this section, we describe two demonstrations to show that RDCode achieves a high transmission performance, and that our work has a potential to be adopted by commercial products. The file transmission using our RDCode prototype system is illustrated in Fig. 3.
(1) In the first demonstration, we show how to transmit a file from a laptop to a smartphone. We use our laptop as the sender, and an Android smartphone Samsung Galaxy S3 as the receiver. The file can be in any format such as JPEG image files and plain text files. We allow a user to choose a file to transmit.

There exist a number of files in various types in a specific directory of our laptop. As long as the source file is chosen, the online encoder of RDCode immediately starts generating frames and simultaneously play the barcode stream. At the other side, the receiver, held by hands, captures the screen of the sender, and simultaneously decodes the image frames. During the demonstration, we (a) change the distance between the sender and receiver, (b) adjust the shooting angle, and (c) even block some area of the sender display. Users can see that the transmission process eventually terminates and the received file is intact and complete.

Figure 3: File transmissions using the RDCode prototype system

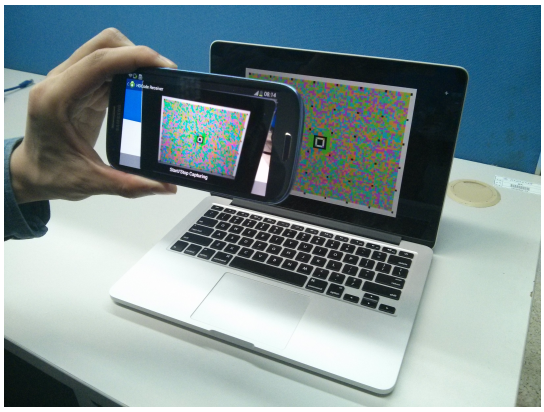

Figure 4: RDCode setup

After the transmission is done, we then plug the receiver into the laptop and show some statistics such as transmission rates and successfully decoded block ratios during the transmission. These statistics are shown with another application running on our laptop.

(2) In the second demonstration, we show how to transmit a file from a tablet to a smartphone. Here we use a tablet Asus Nexus 7 as the sender, and an Android phone Samsung Galaxy S3 as the receiver. An image file is chosen to be transmitted from the sender, and an RDCode frame with a customized shape is generated in the tablet, by disabling some blocks within a frame. The receiver is able to successfully capture and decode RDCode frames, and finally the entire received image is presented on its screen.

4. DEMO SETUP

We need to port our RDCode App to an Android smartphone and an Android tablet with the Android 4.2 operating system. The RDCode App contains both an online encoder and an online decoder, which are developed using Scala with the Android SDK and Scala compiler. For the convenience of demonstration, we migrate the sender to a laptop with Mac OS X 10.9.

Our prototype system is easy to set up. As shown by Fig. 4, we initiate an application to encode a file into RDCode frames, and trigger the RDCode App on our Samsung Galaxy S3 smartphone to receive the file. The tablet is put on its holder to keep still, and is placed beside the laptop when not needed.

5. DEMO REQUIREMENTS

Our demonstration requires a laptop, a tablet and a smartphone. To put on our devices, a desk larger than 80cm*50cm is needed, and we also need an AC power supply with more than 4 plugs. There are two extra requirements. First, a large LCD display connected to our laptop would make people easier to see our prototype system. Second, it would be nice if we could pin a poster around our demonstration desk.

Acknowledgments. This work is supported in part by 973 program (No. 2014CB340304), NSFC (No. 61322207) and SKLSDE grant (No. SKLSDE-2014ZX-20).

6. REFERENCES

[1] T. Hao, R. Zhou, and G. Xing. Cobra: color barcode streaming for smartphone systems. In *MobiSys*, 2012.
[2] T. Hao, R. Zhou, and G. Xing. Demo: a barcode streaming system for smartphones. In *MobiSys*, 2012.
[3] 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.
[4] Q. P. Wenjun Hu, Hao Gu. Lightsync: Unsynchronized visual communication over screen-camera links. In *MobiCom*, 2013.