RescQR: Enabling Reliable Data Recovery in Screen-Camera Communication System

Hao Han, *Member, IEEE*, Kunming Xie, Tongyu Wang, Xiaojun Zhu, *Senior Member, IEEE*, Yanchao Zhao, *Member, IEEE*, and Fengyuan Xu, *Member, IEEE*

***Abstract*—With an increasing number of mobile devices equipped with screens and cameras, screen-camera communication (SCC) systems enable data exchange between devices conveniently and efficiently. By encoding data with spatial and temporal diversity on a screen, multiple users with a camera can receive data without setting up a wireless network. However, as the transmitter pushes the limits of increasing throughput with a high display rate, the receiver actually suffers from a low goodput caused by composite frames. Those frames cannot be decoded correctly with existing methods. To address this problem, we propose a reliable data recovery scheme named RescQR. In RescQR, a mixture separation scheme coupled with a dedicated frame border is proposed to separate composite frames. A Viterbi-based data recovery scheme is proposed to recover data from blurred regions in composite frames. Additionally, an auto-configuration method with the help of a front camera is proposed to adjust parameters automatically according to the estimated distance between the screen and the camera. Our prototype and experiments demonstrate that RescQR achieves a data goodput of 400+kbps even with standard QR codes, which significantly outperforms previous solutions.**

***Index Terms*—Data recovery, dynamic QR code, screen-camera communication, viterbi algorithm.**

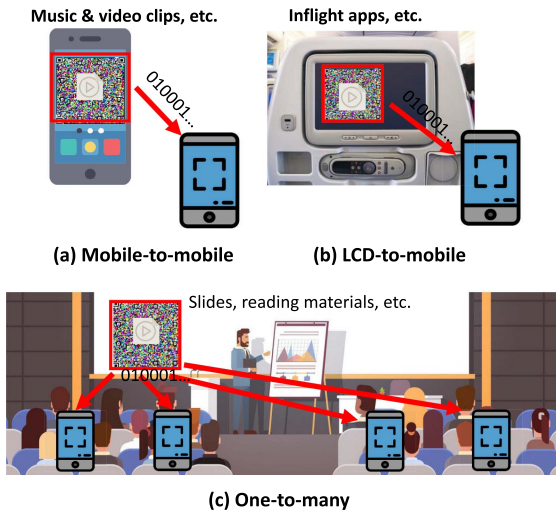


Fig. 1. Example use cases of high-throughput SCC systems.

I. Introduction

SCREEN-CAMERA communication (SCC) is a prominent technology used to transfer data over screen-camera links. Different from traditional visible light communication (VLC) and Optical-Camera communication (OCC) [3], [14], [15], [19], [23], [26], [34], [35], [36], [39], which typically rely on the modulation of the intensity/frequency of light signals (e.g., from LED lamps), SCC can fully utilize the spatial and temporal diversity (i.e., modulating multiple pixels on the screen) to deliver high-rate data communication. Also, SCC does not need any modification to the off-the-shelf screen and camera devices so it can be easily integrated with smartphones. Thus, SCC grows as an increasingly hot topic.

Compared to radio frequency (RF) communications, SCC has several unique benefits. First, SCC establishes a *fast* connection without the explicit setup of a network, which Wi-Fi and Bluetooth typically suffer from. Second, the limited RF spectrum is extremely crowded, while SCC has a large amount of available spectrum, independent regulation, and support of one-to-many communication naturally. Last but not least, SCC has inherent security. Unlike RF signal which passes through walls, SCC is confined into a well-defined coverage zone. Thus, SCC enables many new applications in areas such as mobile communication, digital advertising, smart homes, and intelligent transportation systems.

Fig. 1 shows three typical use cases of high-throughput SCC. In Fig. 1(a), smartphone users could share files such as music and video clips quickly without consuming mobile data when they meet. Unlike Apple's AirDrop, SCC supports data transfer among multiple heterogeneous devices, including ones without Wi-Fi or Bluetooth. Fig. 1(b) shows that airline passengers could download inflight entertainment apps from seatback screens when they get aboard. In Fig. 1(c), the presenter in a conference could share their slides and supporting materials with the live audience during the presentation. Note that Quick Response

Manuscript received 22 August 2022; revised 13 March 2023; accepted 9 May 2023. Date of publication 17 May 2023; date of current version 4 April 2024. This work was supported in part by the National Key R&D Program of China under Grant #2021YFB3100300, in part by the NSFC under Grants #61972200, #61972199, #62172215, and #62272224, in part by the Natural Science Foundation of Jiangsu Province under Grant #BK20200067, and in part by the A3 Foresight Program of NSFC under Grant #62061146002. Recommended for acceptance by G. Xylomenos. *(Corresponding author: Hao Han.)*

Hao Han, Kunming Xie, Tongyu Wang, Xiaojun Zhu, and Yanchao Zhao are with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu 211106, China (e-mail: hhan@nuaa.edu.cn; kunmingxie@163.com; tonywang@nuaa.edu.cn; gxjzhu@gmail.com; yczhao@nuaa.edu.cn).

Fengyuan Xu is with the National Key Lab for Novel Software Technology, Nanjing University, Nanjing, Jiangsu 210023, China (e-mail: fengyuan.xu@nju.edu.cn).

Digital Object Identifier 10.1109/TMC.2023.3277212

(QR) Code has limited data capacity, so it does not work well in the above use cases.

For decades, a couple of high-throughput SCC systems have been proposed but mainly focused on designing new coding schemes, such as PixNet [21], COBRA [10], RDCode [29], RainBar [31], and ERSCC [41]. Also, some efforts are made to improve the reliability of SCC in practice, such as ShiftCode [37] and MegaLight [38]. Most recently, imperceptible SCC communication [22], [27], [40] attracts researchers' attention. Yet little work has studied data recovery without proposing new coding schemes so far.

The maximum throughput supported by a screen-camera link is dependent on the display frame rate. As modern displays can support 120 Hz, 144 Hz, or even higher refresh rates, it gives an opportunity to improve the data rate of an SCC system. However, the increase of camera frame rate on smartphones is slow (e.g., many smartphones can only capture at 30fps or 60fps). When the sender's display frame rate is set higher than the receiver's capture frame rate, the *frame composition* problem occurs that one frame captured by the camera may in fact come from two or more successive video frames played on the screen. This problem is also mentioned in the literature [37], [38], [41]. To alleviate this problem, most approaches require that the capture rate must be higher than the display frame rate but at the cost of sacrificing the maximum throughput. By contrast, we believe that the ability to recover data from composite frames is the key to further improving the data rate of SCC systems.

Currently, several options exist to correct data, e.g., by adding redundancy such as RaptorQ [13] or Reed-Solomon code [33]. Unfortunately, existing error correction mechanisms designed to handle transmission errors cannot address the composite frame problem alone. Suppose a captured image is mixed with frames A and B, and we also have captured another image mixed with frames B and C. If we could recover all parts of frame B from two captured frames, we do not need to waste time retransmitting frame B. The idea seems simple but there are several obstacles as follows:

1) *Unknown mixture patterns:* The mixture pattern in composite frames is non-deterministic in practice. The relationship between the display rate and camera rate affects such a pattern (see Section II-B). The relative angle between the screen and the camera also matters (see Section II-C). Furthermore, the screen-camera link may be occasionally interrupted by hand movement or other effects. It is likely that the content of consecutive frames is totally irrelevant.
2) *No feedback channels:* The screen-camera link is one-way thus lacking the feedback from the camera to report transmission failure. Some studies [22], [29], [41] establish out-of-band (OOB) feedback channels by means of flashlight or audio, but they only work in one-to-one communication. As different receivers may start the transmission and encounter error frames differently, providing high-rate SCC without relying on feedback channels is not straightforward.
3) *Dynamic link conditions:* The quality of screen-camera links varies dynamically in practice, which is affected by ambient luminance, display type, distance, and many more factors. Thus, some pixels in captured frames may be lost, corrupted, or distorted. It is challenging to achieve reliable communication in a self-adaptive way to the environment.

To address the above issues, we propose a novel data recovery scheme dubbed RescQR to further improve existing SCC systems at high frame rates. In RescQR, a sequence of 2D barcodes is repeatedly played on the screen in a *carousel* mode, a receiver can join at any time to decode data from captured frames until all data are received successfully. When capturing composite frames, the receiver uses a mixture separation scheme coupled with a specially-designed frame border to separate composite frames. RescQR also includes a *Viterbi*-based inference method that can recover data from blurred areas in composite frames. In addition, RescQR has an auto-configuration mode to adjust parameters automatically according to the estimated distance between the screen and cameras with the help of a front camera. RescQR can deliver raw throughput of 804kbps and data goodput of 402kpbs when using traditional QR code [6].

It is worth noting that our approach is compatible with any barcode layout. We choose QR code for ease of implementation, although it is not designed for bulky data communication. That is exactly what the word "QR" in the name of our system stands for. With little adaption, RescQR can rescue other barcodes such as some high-capacity codes [10], [29], [31]. If so, RescQR is expected to achieve a higher data rate than we reported in this paper.

To the best of our knowledge, this is the first work focused on improving the data recovery for SCC systems at a high frame rate without designing new coding schemes. The contributions of this work are summarized as follows:

- We present a novel mixture separation algorithm coupled with a newly-designed frame border that addresses the frame composition problem caused by the effect of rolling shutter and unsynchronized screen fresh rate and camera capture rate.
- We present a Viterbi-based inference algorithm to decode data from blurred areas of composite frames with maximized posterior probability.
- We propose an auto-configuration method with the help of a front camera to automatically adjust barcode parameters according to the estimated distance between the screen and the camera.
- We present an extensive empirical study of RescQR under various environmental factors and conditions such as distances, relative angles, ambient light intensity, screen brightness, and different coding parameters. The evaluation results show that RescQR achieves higher goodput than state-of-the-art approaches.

The rest of the paper is organized as follows. Section II discusses the basics of QR code and the underlying causes for composite frames. Section III presents the design of RescQR in detail. Section IV reports the implementation of the system and

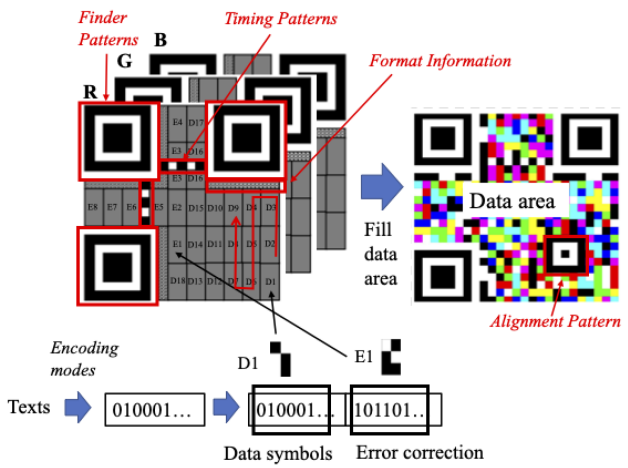


Fig. 2. Function patterns and encoding process for QR code.

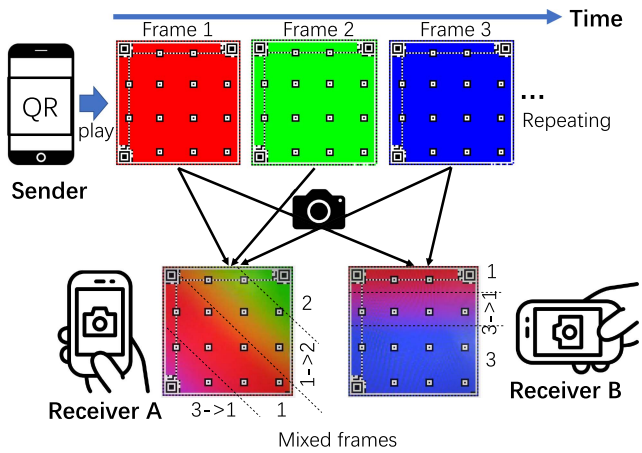


Fig. 3. Effect of refresh rate, rolling shutters, and phone orientations.

the results of extensive experiments. Section V reviews related work, followed by Section VI to conclude the paper.

II. Background

A. Understanding QR Code

Again, the proposed approach does not rely on any specific coding scheme to carry information, but we use QR code in RescQR. Thus, it is necessary to introduce QR code basics before presenting the detail of our system.

The QR code is a 2D matrix code that conveys information by arranging its dark and light squares called "modules" in a grid. Each dark or light module of a QR code symbol represents a 0 or 1, respectively. QR codes can be generated in 40 different versions, from Version 1 (21×21 modules) to Version 40 (177×177 modules). Every QR code has error correction by distributing Reed-Solomon modules. The standard defines four levels that provide approximately 7% to 30% error correction. Besides data and error correction modules, a QR code also contains special modules grouped into various function patterns as shown in Fig. 2, such as *finder patterns*, *alignment patterns*, and *timing patterns* to improve reading performance. Thanks to this design, RescQR does not need to handle many practical issues such as symbol alignment and distortion compensation.

QR code defines several encoding modes, such as numeric and alphanumeric. In RescQR, we directly break up the raw bits of a transmitted file into 8-bit symbols interleaved by error correction codewords, meanwhile adding pads and a terminator if necessary. The transmitted bits are placed starting at the bottom-right of the QR matrix and proceeding upward or downward symbol-by-symbol as shown in Fig. 2. When a function pattern is encountered, occupied modules are skipped until the next unused module is reached. During the placement, data masking is applied to change some modules from dark to light or vice versa according to a particular rule. This step aims to reduce the number of hard-to-read patterns. Finally, if color QR codes are used, each RGB channel is composed of independent black-white QR codes.

B. Effect of Rolling Shutter and Screen Refresh

Mainstream digital displays such as LCD refresh the image line-by-line, measured by *screen refresh rate*. At a time point, the captured image by a camera may stay in the transition that some pixels of the image have already changed to new content while the others have not started changing yet. In addition, image sensors in today's smartphones mostly use CMOS technology [32], which read pixels by *rolling shutter* [17] that each video frame is captured by scanning across the scene line-by-line instead of by taking a snapshot of the entire scene. As a result, the pixels in a captured image may be actually captured at different time instants.

Due to the combined effect of the rolling shutter and screen refresh, it is common to capture a composite frame in practice. If the refresh rate is higher than the rolling shutter's scanning rate, the composite frame may contain more than two images. As shown in Fig. 3, the sender repeatedly plays a video of three frames at 60fps, while receiver A captures the scene at 30fps. We observed that the captured frame might contain regions where pixels were in the middle of the transitions from Frame 3 to Frame 1 and from Frame 1 to Frame 2, as well as regions where pixels have already changed to Frame 1 and Frame 2, respectively. The composite frames, especially those mixed with more than two frames, significantly increase the difficulty of decoding.

C. Effect of Phone Orientation

The relative orientation of the screen and camera also affects composite frames. In Fig. 3, Receiver A held the phone vertically, and Receiver B held the phone horizontally. It shows that the frame captured by Receiver A has inclined mixture bands, while the mixture bands captured by Receiver B are horizontal. The reason is that a screen typically refreshes vertically; By contrast, the rolling shuttle of the smartphone's camera scans horizontally. The actual mixture pattern depends on the relative angle between the screen's and camera's orientations. Thus, some works such as LightSync [12] fail since rows in the same line no longer share the same mixture pattern.

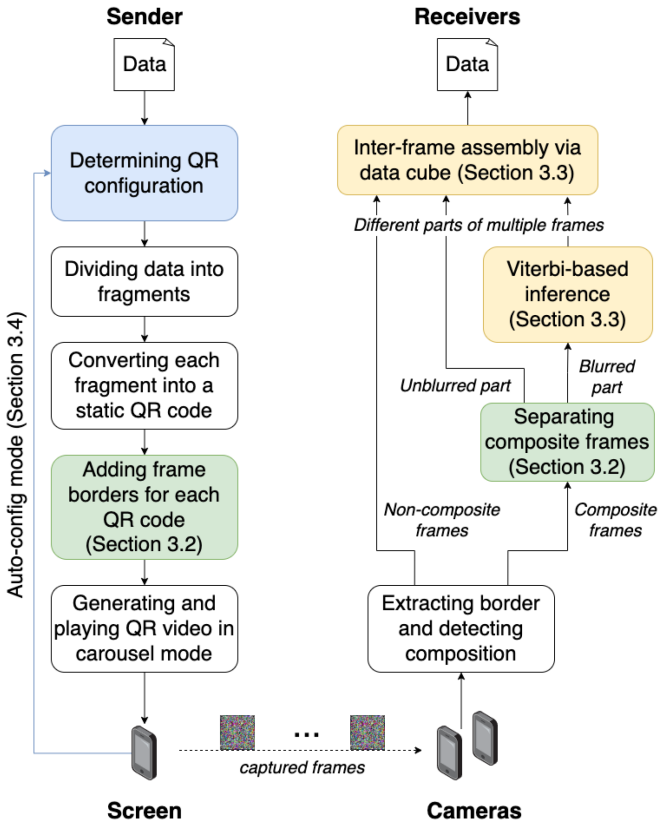


Fig. 4. Workflow of the proposed RescQR system composed of a sender and multiple receivers.

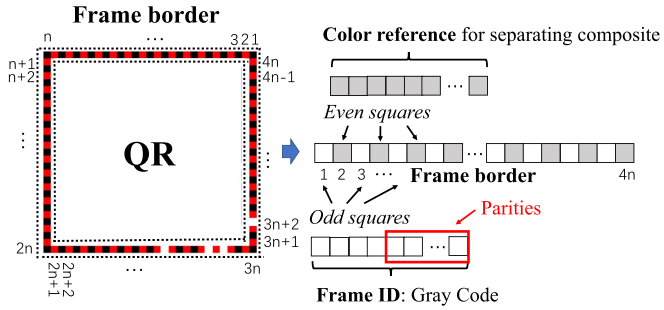


Fig. 5. Frame layout of RescQR.

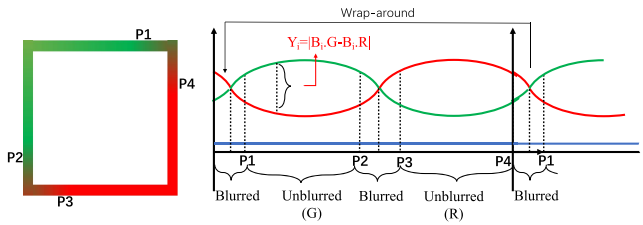


Fig. 6. Illustration of the algorithm to separate a mixed frame into different regions.

D. Effect of Response Time

Screen response time is the time to shift from one color to another. If a pixel is shifting from black to white or vice versa on a screen, the corresponding pixel in a captured frame may show an intermediate value. As a result, a composite frame may contain some blurred bands, and their size depends on the response time. In our experiments, we compared two captured frames with standard LCD and OLED (Organic Light-Emitting Diode) screens. It is found that the composite frame captured on LCD has much larger blurred bands because OLEDs typically have the advantage of a faster response time than standard LCD screens.

III. System Design

A. RescQR Overview

Fig. 4 shows the overall workflow of the proposed RescQR system. Given a file to be transferred, the sender first determines the configuration of QR codes, such as color space, QR version, and error correction level. Next, the raw bits of the file are divided into fragments. Each segment is converted into a static QR code and then surrounded by a specially designed frame border. To transfer, a video consisting of a sequence of generated QR codes is played on the screen repeatedly in a carousel mode.

The receiver can start the camera at any moment and continues capturing video frames until the file is received successfully. To analyze each captured frame, the receiver first extracts the frame border, decodes the embedded information, and detects composite frames. If a captured frame is mixed, which means it comes from two or more successive video frames played on the screen, RescQR attempts to separate every composite. For the blurred parts (see Fig. 6) that usually occur in a composite frame, RescQR adopts a Viterbi-based algorithm [9], [11] to infer correct colors before and after the transition. Finally, partially decoded frames are filled into a data cube by the inter-frame assembly algorithm like solving a Jigsaw puzzle.

B. Separation of Composite Frames

In RescQR, separating composite frames is based on the new design of a frame border. Although the idea of adding a frame border is not new, the encoded data is different from previous approaches in that frame ID and color reference are embedded in the frame border. The layout of each frame displayed on the screen is shown in Fig. 5, where the internal area surrounded by the border is a traditional QR code plus whitespace. The frame border consists of a group of squares of the same size as the inner QR modules. Every square has an index. We define that the first square is located in the top-right corner of the frame border, and the index increases in counterclockwise order. For any frame i, we have

$$\overline{B_i} = (B_i[1], B_i[2], B_i[3], \ldots, B_i[4n]),$$

where $4n$ is the total number of squares in the border. Among them, even squares are used as color references for separating composite frames, and odd squares are used to encode frame ID i with white and black. Thus, the j-th square in $\overline{B_i}$ can be

expressed as follows:

$$B_i[j] \in \begin{cases} (Red|Green|Blue) & \text{if } j \text{ is even} \\ (White|Black) & \text{otherwise,} \end{cases} \quad (1)$$

where $B_i[j] = B_i[k]$ for any even j and k, i.e., a frame has the same reference color.

Next, we present the frame ID correction based on odd squares and the separation of composite frames based on even squares, respectively.

Frame ID Correction Method: In RescQR, the total number of odd squares of $\overline{B_i}$ is equal to $2n$. Each square carries 1-bit information that black is 1 and white is 0. Among $2n$ bits, k bits are used for ID by means of *Gray Coding*, and the remaining $2n-k$ bits are Reed-Solomon (RS) parities [33]. The use of Gray code ensures that two adjacent IDs differ by only one bit. The Gray code and RS code are used together to correct frame ID if some pixels in the border are misinterpreted in practice because we found that the RS code alone was insufficient. It is also worth noting that if QR version 15 is used in the QR area, we have 77×77 modules plus whitespaces surrounded by the frame border. As a result, one side of the frame border contains 80 squares, and the total number of odd squares in the border is equal to 160. Suppose 40 parity bits, we have 120 bits to encode the frame ID, which is large enough to transfer music files and video clips between mobile devices.

On the receiver side, the odd squares of the border are first decoded to derive the frame ID. If some squares are mistakenly recognized, RS bits are the first resort to correct these errors. If failed, we rely on Algorithm 1 seeking to recover them. As shown in the algorithm, the derived ID is first tentatively adjusted by a small deviation (e.g., adding or subtracting 1 or 2) and then applied with RS correction again. If still failed, we use the ID of the preceding and succeeding frames to recover the current ID. For example, the decoded IDs of captured frames are $\{0, 1, 2, 3, 3, 4, 20, 6, 8\}$, we locate the inaccurate ID (i.e., 20 in this example) by finding the longest sub-sequence, and try candidates 4, 5 or 6 to correct 20.

In addition, the color reference is also used to correct IDs. Equation (1) tells that the color is associated with the frame number. If the first frame has a red border, all frames with $i \, mod \, 3 = 0$ should have a red border, all frames with $i \, mod \, 3 = 1$ should have a green border, and a blue border indicates $i \, mod \, 3 = 2$. If all attempts failed, such a frame is discarded. Note that the receiver may join the communication at any time, so the ID of the first captured frame may not start with 0. Our algorithm handles this situation correctly.

Mixture Separation Method: All even squares in a frame border displayed on the screen are colored in RGB space, and they change in a specified sequence (i.e., R, G, B, R,...) frame-by-frame. With this design, we can easily track how frames are mixed. The left panel of Fig. 6 illustrates a composite frame that mixes frame i with a red border and frame $i+1$ with a green border. As shown in the figure, this frame can be divided into three regions, including 1) the *unblurred region* in the bottom-right corner belonging to frame i (i.e., the transition has not started), 2) the *unblurred region* in the top-left corner belonging to frame $i+1$, and 3) the *blurred region* in the middle

Algorithm 1: Frame ID Correction Algorithm.

input : A sequence of received frame borders $S_B = \{\overline{B_1}, \overline{B_2}, ..., \overline{B_k}\}$
output: Corrected IDs $S_{ID} = \{ID_1, ID_2, ..., ID_k\}$

$S_{ID} \leftarrow \{-1\}$;
for $i \leftarrow 1$ **to** k **do**
  $\overline{Odd_i} \leftarrow$ Extract odd squares from $\overline{B_i}$;
  $ID'_i \leftarrow$ DecodeGrayCode($\overline{Odd_i}$);
  $Parities_i \leftarrow$ FindRS($\overline{Odd_i}$);
  $\Delta \leftarrow 0$;
  while *Verify*$(ID'_i \pm \Delta, Parities_i)$ ==*failed* **do**
    $\Delta \leftarrow \Delta + 1$ until reaching a threshold ;
  end
  Update $S_{ID}[i]$ with the successed ID ;
end
Correct S_{ID} with longest sub-sequence ;
Correct S_{ID} with associated color references ;

of the transition from frame i to frame $i+1$. It should be noticed that not all composite frames have these three regions. According to the actual condition, some composite frames may have fewer or more blurred or unblurred regions.

Algorithm 2 presents how to separate these regions based on the color references in the frame border. Recall that frame borders change from Red (255,0,0) to Green (0,255,0) to Blue (0,0,255), and then wrap around to Red (255,0,0) again. Hence, any two consecutive frames should have one RGB channel changing from 0 to 255, one channel from 255 to 0, and the other unchanged. As shown in the right panel of Fig. 6, we plot three curves for RGB channels where the x-axis depicts the index and the y-axis is the RGB value. All curves are smoothed using polynomial curve fitting [1] to remove noise. The curve with the lowest value (i.e., unchanged channel) is omitted. The algorithm calculates the absolute value of the difference between the other two channels, i.e., $\delta = |B_i.G - B_i.R|$. Next, we find pairs of two points between which the difference is greater than a predefined threshold, such as a pair of P_1 and P_2 and a pair of P_3 and P_4. By connecting two end points of these pairs, we can separate the composite frame into different regions. The area surrounded by P_1, P_2, P_3, and P_4 is a blurred region, and others are unblurred regions. Note that this method also works for cutting frames mixed by more than two frames.

The next step is to determine the ID for each separated region. Recall that we have a 1-bit difference in the ID due to the use of Gray Code [2]. If such a difference is located in the unblurred region, we can easily determine the IDs for composite frames. For example, given the decoded ID in the composite frame is i and the difference bit is located between P_1 and P_2, the composite frame consists of frames $i-1$ and i. If the difference bit is located between P_3 and P_4, the composite frame consists of frames i and $i+1$. However, it is complicated if the difference is located in blurred regions. Suppose the decoded ID is i and the difference bit is in a blurred region. We will check how the frame border changes its color to determine whether i belongs to the new frame or the old one. For example, if $i \, mod \, 3 = 1$ and

Algorithm 2: Mixture Separation Algorithm.

input : A received frame border $\overline{B_i}$
output: A set of cutting points S_P

$\overline{Even_i} \leftarrow$ Extract even squares from $\overline{B_i}$;
$\overline{Even_i} \leftarrow$ Smooth($\overline{Even_i}$) ;
$(X, Y) \leftarrow$ Determine the changed RGB channels ;
$S_P \leftarrow \{\}$;
for $k \leftarrow 1$ **to** $2n$ **do**
  $\delta[k] \leftarrow |B_i[k].X - B_i[k].Y|$;
  if $\delta[k] > threshold$ **then**
    Add point k into the set S_P ;
  end
end
$S_P \leftarrow$ Find pairs of cutting points in S_P and remove other points that can be covered by a pair;
Connect endpoints to separate the composite frame ;

the border of the composite frame changes from red to green, we infer that i is the new frame's ID. However, if the decoded ID i is the same but the border changes from green to blue, i should belong to the old frame.

C. Data Recovery Mechanism

Viterbi-Based Inference Scheme: Determining the encoded data for a QR module in an unblurred region is relatively easy. For each RGB channel, if the observed value is closer to 0 than 255, the encoded bit is 1; Otherwise, it is 0. However, we may observe an intermediate value between 0 and 255 for QR modules in a blurred region. No matter how large or small is this intermediate value, it is difficult to determine whether such a value is during the transition from 0 to 0, 0 to 255, 255 to 0, or 255 to 255.

In RescQR, we rely on the color references in the frame border to infer the exact transition. Since reference squares are scanned by the rolling shutter at different time points, some squares are faster to reach the next color than others. As we know the exact colors before and after the transition, the value of each reference square becomes a good measure of the transition speed. Suppose the frame border changes from (255,0,0) to (0,255,0). If we observed (x, y, z) for a reference square in the border, it is inferred that the transition from 255 to 0 has finished by $\frac{255-x}{255}$ and the transition from 0 to 255 has finished by $\frac{y}{255}$ at that location. By connecting pairs of reference squares with similar values, we can draw multiple contour lines as shown in Fig. 7. In this figure, a composite frame has its border changing from red to green and all QR modules changing from white to black. We observed that all QR modules located on a contour line had a similar transition speed which can be estimated by two intersected reference squares. Motivated by this observation, we design a Viterbi-based approach to infer data blocks blurred regions based on the reference squares in the frame border. The detail is described in Algorithm 3 as follows:

Given a QR module, we first leverage Algorithm 2 to decide its region. If such a module is located in a blurred region, we

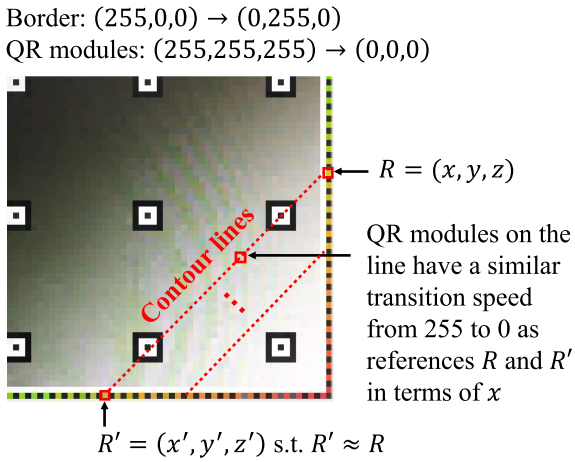


Fig. 7. Illustration of contour lines used to measure transition speed.

Algorithm 3: Viterbi-Based Data Recovery.

Input : Observed values $O = \{O_1 \cdots O_k\}$ of a QR module in a burred region and its contour references $\{(R_1^d, R_1^u) \cdots (R_k^d, R_k^u)\}$
Output: Inferred values $V = \{V_1 \cdots V_{k+1}\}$ of such a QR module on frames 1 to $k+1$

1 **Function** `FindShortest(`i, j`)`**:**
2 **if** $i = j$ **then**
3 $(S_0, S_{255}) \leftarrow (0, 0)$;
4 **else**
5 $(S_0, S_{255}) \leftarrow$ `FindShortest(`$i+1$, j`)`;
6 $e_i^{0\to0} \leftarrow |O_i - 0|$;
7 $e_i^{0\to255} \leftarrow |O_i - R_i^u|$;
8 $e_i^{255\to0} \leftarrow |O_i - R_i^d|$;
9 $e_i^{255\to255} \leftarrow |O_i - 255|$;
10 $S_0 \leftarrow \min\{e_i^{0\to0} + S_0, e_i^{0\to255} + S_{255}\}$;
11 $S_{255} \leftarrow \min\{e_i^{255\to0} + S_0, e_i^{255\to255} + S_{255}\}$;
12 **end**
13 **return** (S_0, S_{255}) ; `// Two shortest paths starting from values 0 and 255`
14 **Function** `Main`**:**
15 **for** *each RGB channel* **do**
16 `FindShortest(`1, k`)`;
17 $V \leftarrow$ Values on the shortest path
18 **end**

derive the slope of the contour line that passes through such a block. Two intersections at the frame border are averaged to indicate the transition speeds from 255 to 0 (down) and 0 to 255 (up), denoted as R^d and R^u, respectively. Next, we adopt a Viterbi-like algorithm [9], [11] to infer the exact value before and after the transition with the maximum a posteriori probability. As shown in Fig. 8, the inference is performed per RGB channel. Each channel is built with an individual trellis, where vertices represent black and white colors and edges represent the color transition between consecutive frames. Suppose the observed value of a QR module in frame i is denoted as O_i and the transition speeds at that location are R_i^d and R_i^u. By following lines 6 to 9 in the algorithm, we estimate the "error" between

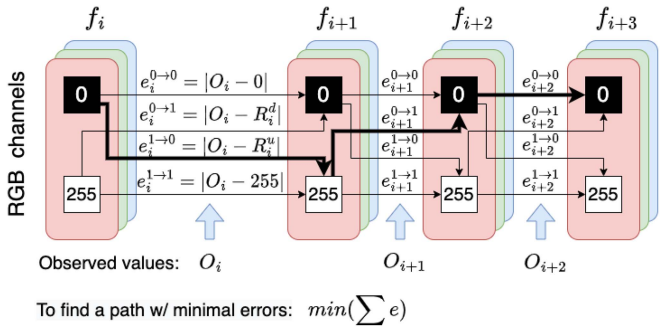


Fig. 8. Illustration of our Viterbi-based inference for data recovery in RGB channels.

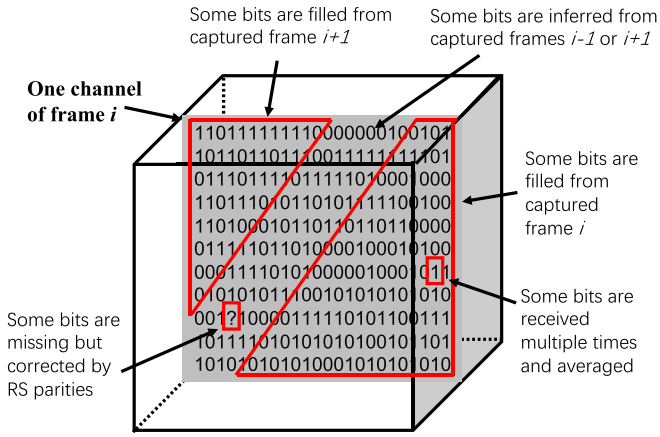


Fig. 9. Illustration of inter-frame assembling via a data cube.

the observed transition and the actual transition from 0 to 0, 0 to 255, 255 to 0, or 255 to 255.

This error is accumulated as the evolution of the state machine in the trellis. Our goal is to find the shortest path with the minimal sum of errors out of all path combinations. We leverage the *Dynamic Programming* technique to accelerate the process. The solid line in the example shows an example path. The restored data is $\{0 \to 255 \to 0 \to 0\}$. To prevent the computation from growing too complex, our algorithm starts to find the Viterbi path [11] when the number of blocks reaches a predefined threshold.

Inter-Frame Assembling: After decoding QR modules in both blurred and unblurred regions and their belonged frames, the receiver assembles bits into a data cube as shown in Fig. 9 to restore the original data transmitted by the sender. Each data panel in the cube represents a single RGB channel of a QR code displayed on the screen, where some bits may be directly filled from non-composite video frames or unblurred regions in composite frames after mixture separation, and some may be inferred from blurred regions using the proposed Viterbi-based method. When enough bits on a data panel have been received, the encoded parity bits are used to verify the correctness of these bits.

Since the transmitter displays all QR codes repeatedly in a carousel mode, if the receiver missed some positions in the cube, he still has a chance to obtain them in the next round. Also, some positions may be filled in multiple rounds. In that case, we update the cube with a mean value. However, it is likely that some positions might be lost several times due to unpredicted reasons, the receiver attempts to guess a value based on the same position in preceding and succeeding frames. To prevent errors from occurring in a fixed place, the sender will shuffle the starting frame of the sequence (note that the Gray-coded ID is not affected). Eventually, the receiver will report receiving all information until the data cube is completed successfully.

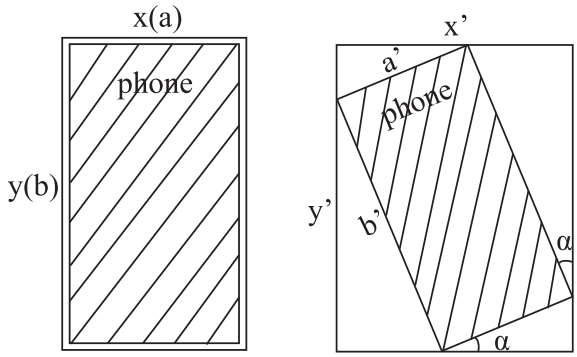
D. Auto-Configuration Mode of RescQR

The parameters of a QR code such as color space, version, and error correction level play an important role in determining the data capacity (i.e., how many frames are needed to transfer the data), and further affect the throughput of the proposed SCC system. For example, if the distance between the sender and receiver is close, we can increase the capacity by using RGB colors and more modules for each QR code. However, the sender typically uses a fixed configuration, because he does not know where the receivers are. In RescQR, we propose a new way of adjusting the parameters of QR codes adaptively based on the estimated distance between the sender and receivers with the help of a front camera. If no receivers are detected before data transmission, the sender uses back the default configuration. If more than one receiver is detected, the longest distance is used. Note that this is an optional mode that is only supported by the sender with a front camera (e.g., smartphones). A rough estimation of the distance is enough for RescQR. Measuring an accurate distance such as [5] within an image is out of the scope of this paper.

With computer vision and deep learning techniques, the sender attempts to recognize the receivers' smartphones within a captured image. If detected, the outline and category of smartphones are derived. Based on the assumption that smartphones have a reasonably similar size, we can estimate the distance to a recognized device by computing the scale of the length/width of that device. But this demands at least a reference distance denoted by D with a known size $a \times b$ of the smartphone. Fortunately, this reference model can be trained offline once and is free to use later. We use YOLO and ImageNet for object detection and recognition. The bounding boxes of the objects are extracted from the image, and the outline of the smartphone is determined by the Gaussian edge detection algorithm and Hough line detection [8]. As shown in Fig. 10(a), if the phone is not tilted, the size of the phone $a \times b$ in the captured image can be measured by the pixel size of the bounding box $x \times y$.

However, a difficult problem is that the recognized smartphone inside the bounding box is tilted, as users may hold their smartphones freely when taking videos. Then, the direct use of the bounding box to estimate the distance will cause estimation errors. To address this problem, we define the tilted angle between the bounding box and the phone in the image as α. The size of the box is $x' \times y'$. Equation (2) is used to calculate a' and b' of the phone, respectively.

$$a' * \sin\alpha + b' * \cos\alpha = y'$$



(a) Not tilted (b) Tilted

Fig. 10. The schematic of estimating the length and width of the phone.

$$a' * \cos\alpha + b' * \sin\alpha = x' \tag{2}$$

then

$$\alpha = \arctan\frac{c-k}{1-c*k} \tag{3}$$

where $c = \frac{x'}{y'}$, $\frac{a}{b} = \frac{a'}{b'} = k$. We can substitute back to the original (2) to replace α to calculate the values of a' and b'.

Given the pixel width a and length b of the phone at the reference distance D, the actual distance D' is estimated by the following formula:

$$f = \frac{2*D*(a+b)}{C} = \frac{2*D'*(a'+b')}{C}$$

$$D' = \frac{D*(a'+b')}{a+b},$$

where f is the focal length, and $C/2$ is half of the perimeter of the captured image.

Next, the relationship between the distance and the optimal configuration is rule-based and set heuristically based on our extensive experiments (see Section 4.6 for detail). In general, long distances mean a monotonous color space, a high error correction level, and a low QR code version for reliable data transmission. For example, in the transmission distance less than 50 cm, RescQR uses RGB QR codes, the error correction level of 12.5%, and the QR version equal to 15. At a distance of 110 cm or further, RescQR set the black and white QR codes with 25% error correction level and a QR version number of 7.

IV. Implementation and Evaluation

A. System Implementation

A prototype of RescQR system was implemented under Android 10.0 (API 29), but compatible with all Android devices with Android 8.1 (API 27) or later. The video capture and image analysis are performed using Android's CameraX library.

On the Transmitter Side, we adopt open-source ZXing [24] to generate static QR codes with the proposed frame border and then invoke the FFmpeg APIs to play the QR video composed of the generated QR codes on the screen. All frames in the video are organized in a ring. When finishing playing the video once, we randomly pick a new starting frame to play. Again, the goal of this step is to prevent the constant errors occurring in periodic frames.

On the receiver side, the camera is set to non-blocking mode, where the latest image is always cached into an image buffer while the application analyzes the previous image. To minimize the processing delay, the app was implemented with a thread pool and shadow buffering technique. The receiver needs to perform several tasks, including extracting frame borders and decoding frames. In our implementation, tasks were executed parallelly on multi-cores, and the multi-thread communication was implemented by Handler/Message mechanism. To further increase the frame rate, we adjusted the light threshold of the camera interface and disabled the camera's autoexposure routine. Note that many Android Phones can only support YUV420 and the YUV-RGB conversion involves a high computation load. In that case, we use OpenCV to convert the YUV image into RGB color space.

As a comparison, we also implemented a *baseline* system, where each raw frame of the captured video is directly decoded by ZXing without any technique proposed in RescQR.

Fig. 11. Experimental setup of Mobile-to-Mobile and PC-to-Mobile using RescQR.

B. Experimental Methodology

Experiment Settings: The implemented prototype was tested with commercial computer monitors and smartphones in two scenarios: *Monitor-to-Mobile* and *Mobile-to-Mobile* as shown in Fig. 11. In the former scenario, we used a Dell E2420H 24-inch LED monitor as the transmitter, which supports a 60 Hz refresh rate along with a 1920 × 1080 screen resolution. The receiver includes Xiaomi Mi 8, OnePlus7, and Huawei P30 pro smartphones. All phones have a 6 inches display with a 60 Hz refresh rate. Except Xiaomi Mi 8 which has 1080 × 2248 pixels, other phones have 1080 × 2340 resolution. In the latter scenario, smartphones were tested as either a transmitter or a receiver alternatively. Unless otherwise stated, the display rate is set to 30fps, and the capture rate is 60fps at 1080p resolution. The distance between the transmitter and the receiver is 45 cm. The version of QR Code is set to 19 (81 × 81 modules). The screen brightness uses the factory value. The angle between the phone and the screen is zero. The error correction level is configured to 25%, and RGB colors are used to encode data.

TABLE I
OVERLL COMPARISON OF RESCQR AND MOST RECENT APPROACHES

Approaches	Settings (fps)	Thoughput (kbps)	Goodput (kbps)	Distance (cm)	Feedback Channel	Perceptible
RescQR	31@60	804.4	402.1	45	No	Yes
RainBar+ [43]	30@60	648.8	325.4	12	Microphone	Yes
ERSCC [41]	15@30	245.2	N/A	12	Light	Yes
DeepLight [27]	60@120	26.6	4.7	100	No	No
ChromeCode [40]	120@120+	777	120	50	No	No
AirCode [22]	120@120+	1086	159	50	Microphone	No

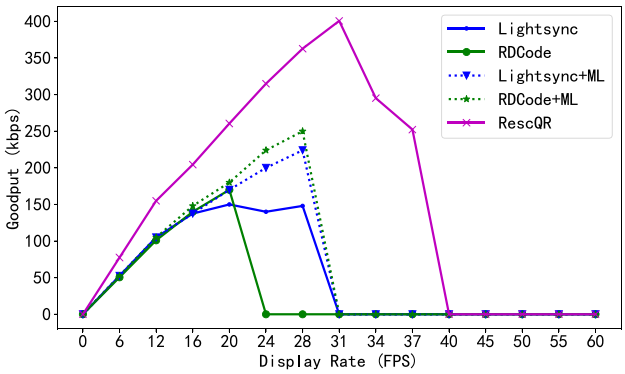


Fig. 12. Goodput achieved by RescQR and state-of-the-art systems at different display rates.

Performance Metrics: All the experiments were conducted in terms of goodput, throughput, and block error rate (BER). *Goodput* is the amount of useful information delivered to the receiver per unit of time. We calculate the goodput by the bit size of data divided by the total delivery time that the receiver successfully receives the data. *Throughput* is the amount of all received bits including protocol overhead bits and duplicated bits per unit of time. The *BER* is the number of incorrect QR modules divided by the total number of modules in a received frame.

Evaluation Goals: With the above experiment settings and performance metrics, we evaluated the performance of RescQR by answering the following questions:

1) Q1: *Can RescQR improve the state-of-the-art SCC systems?*
2) Q2: *What is the performance of RescQR in various environmental factors and conditions?*
3) Q3: *Is the auto-configuration mode effective to improve RescQR in practice?*

C. Performance Comparison

To answer Q1, we compared the performance of RescQR to the state-of-the-art SCC systems. As we have no access to the source code of these systems, we use the results reported in their papers and evaluate RescQR in the same settings for comparison.

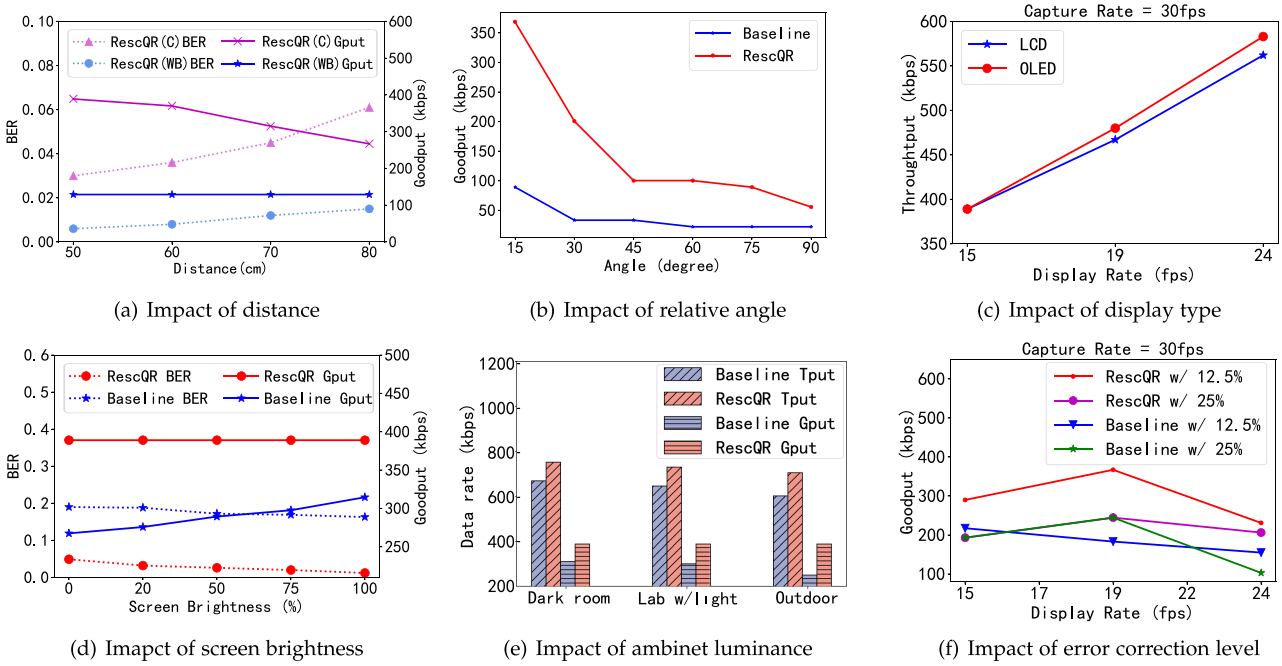
Fig. 12 shows the results of the proposed RescQR compared to selected SCC systems in terms of the goodput at a fixed capture rate of 60 Hz and different display frame rates. The reason we only compared Lightsync [12], RDCode [29], and MegaLight (ML) [38] is that the selected approaches all reported the experiment results with different display rates in their papers. For other approaches, we could only compare them at specific frame rates as shown in Table I. In Fig. 12, the results of RescQR were averaged over 10 experiments at each frame rate. We find that the goodput of all approaches continues increasing when the display rate is low. After the display rate exceeds 20fps, the goodput of Lightsync and RDCode starts to drop due to the effect of composite frames. With the help of Megalight (ML), both RDCode and Lightsync could increase the goodput by 12-28fps but failed to deliver any information when the display-capture ratio is greater than 1/2 (i.e., when the display rate exceeds 30 Hz). Our approach achieves slightly better performance in low frame rates and much better performance in high frame rates due to the ability to recover data from composite frames.

Again, RescQR is not limited to using QR codes only. If advanced coding schemes such as COBRA [10] and Rainbar [31] replace QR codes to carry information, our approach is expected to achieve a higher goodput in theory. That is because QR codes are designed for sparse data communication in a reliable way, but not for bulky data. Simply, if the full rectangular area rather than the square size of the screen is used, the goodput reported here will get further increased. Even in this case, RescQR has a better goodput than compared approaches.

Table I shows the compared results of RescQR with the most recent SCC systems in terms of throughput, goodput, and other features. Note that these approaches may have different goals from ours. For example, DeepLight [27], ChromeCode [40], and AirCode [22] focus on hidden screen-camera communication without interfering experience of the audience watching actual video content. RainBar+ [43] and ERSCC [41] aim to build a high-goodput peer-peer communication system that has real-time feedback channels. Due to the design of dedicated frame boards and Viterbi-based data recovery mechanism, RescQR has an averaged throughput/goodput of 804kbps/402 when the display rate equals 31fps and the distance is 50 cm but other settings are default. The result has nearly 24% improvement over RainBar+, as well as 3x over ChromeCode and 2x over AirCode even though the frame rates are much lower for RescQR. Although AirCode can achieve a higher throughput of around 1 Mbps, its effective rate (i.e., goodput) is lower than RescQR. It is seen that our approach outperforms the state-of-the-art SCC systems even with standard QR codes.

D. Parameter Study

To answer Q2 we conducted experiments to study the impacts of various factors and conditions on RescQR, including distances, angles, display types, ambient luminance, and more.



(a) Impact of distance (b) Impact of relative angle (c) Impact of display type

(d) Imapct of screen brightness (e) Impact of ambinet luminance (f) Impact of error correction level

Fig. 13. Parameter study of RescQR in various enviromental conditions and factors.

For each factor, we only adjusted such a factor with others unchanged.

Impact of Distance: Fig. 13(a) shows the performance of RescQR in the distance from 50 cm to 80 cm. Note that we omit the results for distances less than 50 cm because their performance does not vary much. It is seen that the goodput of RescQR using color QR codes drops by 20%–60% with distance increased, while RescQR using black-and-white QR codes keeps a stable performance. That is because when the camera is close to the screen, the image sensor has a high chance to recognize the color correctly. Recall that each QR module in RescQR contains 3 bits of information in an RGB color space. Thus, the receiver needs to correctly recognize eight different colors for decoding the data correctly. By contrast, if each module is either dark or light, the receiver only needs to distinguish two colors. As the distance between the transmitter and receiver increases, the receiver has fewer pixel samples per QR module as if the module got smaller. Correctly recognizing eight colors may occur more errors than just identifying two colors.

Impact of Angle: Fig. 13(b) shows the goodput of RescQR with relative angles between the phone and the screen adjusted from 0° to 90° in clockwise order when the display rate and capture rate are fixed at 30fps and 60fps, respectively. To demonstrate the performance gain, we compared RescQR to the baseline (i.e., QR codes enhanced by RescQR v.s. standard QR codes). As the angle changes, both the captured QR codes and mixture patterns become irregular. To solve these issues, our approach relies on functional patterns (see Section II for alignment and distortion compensation, while the skewed mixture patterns are handled by the proposed mixture separation mechanism. Due to this design RescQR achieves about 3x-4x improvement over the baseline in most cases.

Impact of Display Type: We compared the Dell monitor (LCD) to different phone screens (OLED) and the results are shown in Fig. 13(c). We find that RescQR has slightly better performance on OLED screens over LCD screens. That is because OLED screens typically have brighter colors and less response time when changing from one color to another color. Thus, the receiver will capture smaller blurred bands and have a high chance to recognize the color in each module correctly. Note that we do not distinguish different phones in the figure because they all have similar performance.

Impact of Screen Brightness: Fig. 13(d) shows the performance of RescQR with the screen brightness adjusted from 0 to 100%. A brighter screen typically leads to more accurate color recognition, as the BER of the baseline decreases with the increasing screen brightness. Compared to the baseline, RescQR improves the BER and goodput, especially in the case of low brightness.

Impact of Light: Ambient luminance plays an important role in scanning QR codes. Fig. 13(e) shows the performance of RescQR under three real-world lighting scenarios, including a dark room, a lab with lamps, and outdoors with sunlight. We measured the ambient luminance for each scenario with a light meter app on the phone: dark room (0 lux), lab with an incandescent lamp (200 lux), and outdoor (1000 lux). From the figure, we find that the impact of ambient luminance on goodput is negligible, although the throughput decreases. That is because the Viterbi-based inference algorithm indeed helps data recovery.

Impact of Error Correction Level: Error correction level is a parameter of QR codes. We studied its impact by comparing RescQR to the baseline with the error correction level ranging from 12.5% to 25%. For each level, we evaluated both RescQR

TABLE II
RELATIVE ERRORS OF RESCQR AT DIFFERENT DISTANCES

Distance	Small error (RE $<$ 20%)	Large error (RE $\geq$ 20%)
30cm	95%	5%
50cm	93%	7%
70cm	92%	8%
90cm	89%	11%
110cm	87%	13%
130cm	85%	15%

and the baseline with different display rates. It is worth noting that the experiments were conducted with the capture rate fixed at 30fps. As shown in Fig. 13(f), when the display-capture ratio is less than 1/2, there is no gain between RescQR and the baseline both with the error correction level of 25%, because the Reed-Solomon codec alone is enough to correct all transmission errors. However, in face of a high display-capture ratio large error correction level is insufficient, because composite frames dominate the captured frames. Our approach can achieve up to 3.5x goodput over the baseline. It is also obvious for the error correction level of 12.5%.

E. Effectiveness of Auto-Configuration Mode

To answer Q3, we further evaluated the auto-configuration mode of RescQR in terms of the accuracy of our distance estimation and the performance improvement gained from the adaptive configuration according to the estimated distance.

Accuracy of Distance Estimation: We conducted extensive experiments with Xiaomi Mi 8 as the sender and other phones as the receiver. The dimensions of Mi 8 are 155 $\times$ 75 mm and the tested distance between the sender and receiver ranges from 30 cm to 130 cm. The relative error (RE) within 20% is deemed as *small error*. If greater than 20%, the error is a *large error*. Table II shows the percentage of total tests with small errors and large errors, respectively. We find that the overall accuracy decreases slightly with increased distances, but even at a distance of 130 cm, we still have around 85% tests with RE less than 20% (i.e., absolute error is less than 26 cm). It is worth noting that RescQR does not require a high-accuracy of distance estimation. Given that most of the estimations are within a small error range, we can obtain the optimal QR parameters stably.

Performance Gain From Auto-Configuration: Fig. 14 shows the goodput of RescQR in auto-configuration and fixed configurations at different distances. These experiments were conducted with default settings. The configurable parameters include color space, error correction level, and QR version. We conclude that: 1) The color codecs can only work within a limited distance since the receiver has more difficulty in identifying RGB colors correctly at a long distance. Our experiments show that beyond 80 cm only black-and-white QR codes could deliver non-zero goodput. 2) For every range of distance, there exist a dominant set of parameters (i.e., a configuration) that have the best performance over others. For example, the configuration of black-and-white QR codes, 25% error correction level, and version 11 could bring the best goodput within the range from 75 cm to 100 cm. 3) The proposed auto-configuration method is effective to improve the overall performance adaptively. If sticking to an inappropriate configuration, we may have a less-utilized communication channel when the sender and receiver are close, or cannot convey any information when the receiver is far away from the sender.

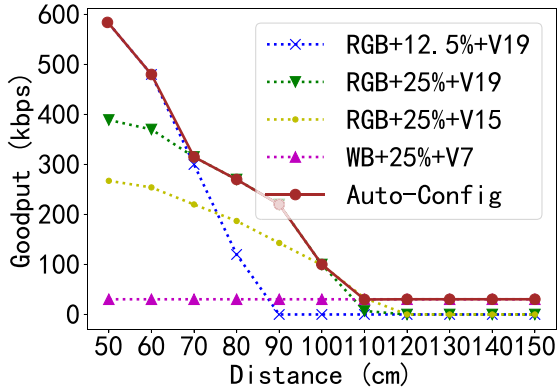


Fig. 14. Comparison of RescQR with auto-configuration and fixed configurations.

V. RELATED WORK

Screen-Camera Communication (SCC) is a special type of broader category of Visible Light Communication (VLC) [19], [26], [39] and Optical-Camera communication (OCC) [25], [42]. The VLC technologies hinge on visible bands of light sources, while the OCC technologies utilize optical image sensors as receivers based on IR or visible bands. Compared to these technologies, SCC has many advantages, such as easy integration with off-the-shelf smartphones and a relatively high data rate. In this section, we mainly focus on SCC technologies, while a comprehensive survey of VLC or OCC can be found in [3], [15], [23].

Existing SCC technologies can be classified into two broad categories: 1) approaches *to improve data rate*, allowing visible content displayed on the screen for data transmission; and 2) approaches *to reduce user perception*, focusing on the data transmission without interfering watching experience of the audience.

Visible Screen-Camera Communication: The goal of this direction is to achieve high throughput reliably. It does not matter whether the user notices the dedicated content displayed on the screen. PixNet [21] designs a pioneering method of encoding data in 2D OFDM symbols and enables data streaming between LCD screens and cameras. COBRA [10] proposes a novel color barcode optimized for low-complexity processing and adaptive code block arrangement to be blur-resilient. However, these approaches suffer from rolling-shutter effects, so they are limited at low display/capture frame rates. LightSync [12] uses interframe coding to support asynchronous communication, but it limits the colors to black and white. Styrofoamt [18] proposes a coding scheme for resolving inter-symbol interference by inserting blank frames into the transmission pattern but at the cost of reducing the throughput. To further improve the throughput of SCC in practice, RDCode [29] contributes a robust dynamic code

design with multi-level error correction schemes. RainBar [31] and its extended version RainBar+ [43] make improvements on block locating and frame synchronization to get better stability and capacity in transmission. SoftLight [7] designs a scheme with automatic adaptation of transmission rates in diverse scenarios and shows the improved overall goodput. By leveraging machine learning technologies, MMCode [4] designs a semi-supervised Gaussian Mixture Model (GMM) algorithm, which improves the accuracy of recognizing colors. Meglight [38] uses a Random Forest-based model to support robust decoding with pre-inserted training frames. In ERSCC [41], a self-restoration coding and an additional feedback channel are proposed to increase coding efficiency.

RescQR differs from existing works in that our approach provides an orthogonal direction to resolve the lack of synchronization between the screen and camera frame rates. When encountering mixed frames, RescQR attempts to recover data according to historical information with best efforts rather than just discarding the frames with error.

Hidden Screen-Camera Communication: In this direction, pioneering works such as InFrame [30] and its extended version InFrame++ [28] leverage the spatial-temporal flicker-fusion property and CDMA-like modulation to deliver data streams without interfering with watching experience of the audience. HiLight [16] leverages the orthogonal transparency (alpha) channel to embed bits into pixel translucency changes without modifying pixel color values. TextureCode [20] improves invisibility by adaptive embedding based on video texture. ChromaCode [40] improves code invisibility by modifying lightness in uniform color space and achieves full imperception. DeepLight [27] enhances reliable data transmission rates of hidden SCC in diverse real-world conditions by selectively modulating the intensity of only the Blue channel and incorporating ML models in the decoding pipeline. To further improve the data rate, AirCode [22] takes the complementary advantages of video and audio channels and adopts visual odometry for accurate screen detection.

The above works aim at invisible communication with a screen-camera link as a side channel. On the contrary, RescQR focuses on reliable data recovery to achieve a higher data rate using existing coding schemes instead of the imperception to human eyes.

VI. Conclusion

In this paper, we present RescQR, an end-to-end SCC system that achieves a reliably high goodput by recovering data from composite video frames without the help of new coding schemes. Due to the combined effects of rolling shutter and display refresh rate, the problem of frame composition is inevitable for SCC systems that pursue high throughput using a high display rate. In RescQR, the receiver uses a novel mixture separation algorithm coupled with a newly-designed frame border to separate composite frames, and a novel Viterbi-based inference algorithm to guess data from blurred areas of composite frames with maximum posterior probability. The sender of RescQR uses a novel auto-configuration method with the help of a front camera to automatically adjust barcode parameters according to the estimated distance between the screen and the camera. We prototype RescQR and evaluated it by extensive experiments. The results show that RescQR has a better goodput than previous works even with standard QR codes. The performance is expected to be further improved by integrating with other high-capacity coding schemes. These innovations allow RescQR to support several new classes of high-data-rate SCC applications via personal mobile devices.

Acknowledgments

We would like to thank the editors and anonymous reviewers for their valuable comments helping us to improve this work.

References

[1] H. Akima, "A new method of interpolation and smooth curve fitting based on local procedures," *J. ACM*, vol. 17, no. 4, pp. 589–602 1970.
[2] J. R. Bitner, G. Ehrlich, and E. M. Reingold, "Efficient generation of the binary reflected gray code and its applications," *Commun ACM*, vol. 19, no. 9, pp. 517–521, 1976.
[3] W. A. Cahyadi, Y. H. Chung, Z. Ghassemlooy, and N. B. Hassan, "Optical camera communications: Principles, modulations, potential and challenges," *Electronics*, vol. 9, no. 9, 2020, Art. no. 1339.
[4] X. Chen, W. Li, T. Zhan, and S. Lu, "MMCode: Enhancing color channels for screen-camera communication with semi-supervised clustering," in *Proc. 27th Int. Conf. Comput. Commun. Netw.*, 2018, pp. 1–9.
[5] R. Couturier, M. Salomon, E. A. Zeid, and C. A. Jaoude, "Using deep learning for object distance prediction in digital holography," in *Proc. Int. Conf. Comput. Control Robot.*, 2021, pp. 231–235.
[6] Denso Wave Incorporated, "Qr code essentials," 2011. [Online]. Available: http://www.qrcode.com/en/
[7] W. Du, J.C. Liando, and M. Li, "SoftLight: Adaptive visible light communication over screen-camera links," in *Proc. IEEE 35th Annu. Int. Conf. Comput. Commun.*, 2016, pp. 1–9.
[8] O. R. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11–15, 1972.
[9] G. D. Forney, "The viterbi algorithm," *Proc. IEEE*, vol. 61, no. 3, pp. 268–278, Mar. 1973.
[10] T. Hao, R. Zhou, and G. Xing, "COBRA: Color barcode streaming for smartphone systems," in *Proc. 10th Int. Conf. Mobile Syst. Appl. Serv.*, 2012, pp. 85–98.
[11] J. F. Hayes, "The viterbi algorithm applied to digital data transmission," *IEEE Commun. Mag.*, vol. 40, no. 5, pp. 26–32, May 2002.
[12] W. Hu, H. Gu, and Q. Pu, "LightSync: Unsynchronized visual communication over screen-camera links," in *Proc. 19th Annu. Int. Conf. Mobile Comput. Netw.*, 2013, pp. 15–26.
[13] Internet Engineering Task Force (IETF), "RaptorQ forward error correction scheme for object delivery," 2020. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6330.html
[14] C.-M. Kim, S.-I. Choi, and S.-J. Koh, "IDMP-VLC: IoT device management protocol in visible light communication networks," in *Proc. 19th Int. Conf. Adv. Commun. Technol.*, 2017, pp. 578–583.
[15] N. Tuan Le, M. A. Hossain, and Y. M. Jang, "A survey of design and implementation for optical camera communication," *Signal Processing: Image Commun.*, vol. 53, pp. 95–109, 2017.
[16] T. Li, C. An, X. Xiao, A. T. Campbell, and X. Zhou, "Real-time screen-camera communication behind any scene," in *Proc. 13th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2015, pp. 197–211.
[17] C.-K. Liang, L.-W. Chang, and H. H. Chen, "Analysis and compensation of rolling shutter effect," *IEEE Trans. Image Process.*, vol. 17, no. 8, pp. 1323–1330, Aug. 2008.
[18] R. LiKamWa, D. Ramirez, and J. Holloway, "Styrofoam: A tightly packed coding scheme for camera-based visible light communication," in *Proc. 1st ACM MobiCom Workshop Visible Light Commun. Syst.*, 2014, pp. 27–32.
[19] S. Naribole, S. Chen, E. Heng, and E. Knightly, "LiRa: A WLAN architecture for visible light communication with a Wi-Fi uplink," in *Proc. 14th Annu. IEEE Int. Conf. Sens. Commun. Netw.*, 2017, pp. 1–9.
[20] V. Nguyen et al., "High-rate flicker-free screen-camera communication with spatially adaptive embedding," in *Proc. IEEE 35th Annu. Int. Conf. Comput. Commun.*, 2016, pp. 1–9.

[21] S.D. Perli, N. Ahmed, and D. Katabi, "PixNet: Interference-free wireless links using LCD-camera pairs," in *Proc. 16th Annu. Int. Conf. Mobile Comput. Netw.*, 2010, pp. 137–148.

[22] K. Qian et al., "AIRCODE: Hidden screen-camera communication on an invisible and inaudible dual channel," in *Proc. 18th USENIX Symp. Netw. Syst. Des. Implementation*, 2021, pp. 457–470.

[23] N. Saha, M. S. Ifthekhar, N. T. Le, and Y. M. Jang, "Survey on optical camera communications: Challenges and opportunities," *Iet Optoelectron.*, vol. 9, no. 5, pp. 172–183, 2015.

[24] Srowen, "Zxing," 2016. [Online]. Available: https://github.com/ZXing

[25] M. C. Tapia, T. Xu, Z. Wu, and M. Z. Zamalloa, "SunBox: Screen-to-camera communication with ambient light," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 6, no. 2, Jul. 2022, Art. no. 46.

[26] Z. Tian, K. Wright, and X. Zhou, "The darklight rises: Visible light communication in the dark: Demo," in *Proc. 22nd Annu. Int. Conf. Mobile Comput. Netw.*, New York, NY, USA, 2016, pp. 495–496.

[27] V. Tran, G. Jayatilaka, A. Ashok, and A. Misra, "DeepLight: Robust & unobtrusive real-time screen-camera communication for real-world displays," in *Proc. 20th Int. Conf. Inf. Process. Sensor Netw.*, 2021, pp. 238–253.

[28] A. Wang, Z. Li, C. Peng, G. Shen, G. Fang, and B. Zeng, "InFrame: Achieve simultaneous screen-human viewing and hidden screen-camera communication," in *Proc. 13th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, New York, NY, USA, 2015, pp. 181–195.

[29] A. Wang, S. Ma, C. Hu, J. Huai, C. Peng, and G. Shen, "Enhancing reliability to boost the throughput over screen-camera links," in *Proc. 20th Annu. Int. Conf. Mobile Comput. Netw.*, 2014, pp. 41–52.

[30] A. Wang, C. Peng, O. Zhang, G. Shen, and B. Zeng, "InFrame: Multiflexing full-frame visible communication channel for humans and devices," in *Proc. 13th ACM Workshop Hot Topics Netw.*, New York, NY, USA 2014, pp. 1–7.

[31] Q. Wang, M. Zhou, K. Ren, T. Lei, J. Li, and Z. Wang, "Rain Bar: Robust application-driven visual communication using color barcodes," in *Proc. IEEE 35th Int. Conf. Distrib. Comput. Syst.*, 2015, pp. 537–546.

[32] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI design: A Systems Perspective*, Boston, MA, USA: Addison-Wesley, 1985.

[33] S. B. Wicker and V. K. Bhargava, *Reed-Solomon Codes and Their Applications*. Hoboken, NJ, USA: Wiley, 1999.

[34] Y. Yang, J. Hao, and J. Luo, "CeilingTalk: Lightweight indoor broadcast through LED-camera communication," *IEEE Trans. Mobile Comput.*, vol. 16, no. 12, pp. 3308–3319, Dec. 2017.

[35] Y. Yang, J. Luo, C. Chen, W.-D. Zhong, and L. Chen, "SynLight: Synthetic light emission for fast transmission in cots device-enabled VLC," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1297–1305.

[36] Y. Yang, J. Nie, and J. Luo, "ReflexCode: Coding with superposed reflection light for LED-camera communication," in *Proc. 23 rd Annu. Int. Conf. Mobile Comput. Netw.*, New York, NY, USA, 2017, pp. 193–205.

[37] T. Zhan, W.X. LiChen, and S. Lu, "Capturing the shifting shapes: Enabling efficient screen-camera communication with a pattern-based dynamic barcode," *Proc. ACM Interactive Mobile Wearable Ubiquitous Technol.*, vol. 2, no. 1, pp. 1–25, 2018.

[38] T. Zhan, W. Li, X. Chen, and S. Lu, "MegaLight: Learning-based color adaptation for barcode stream recognition over screen-camera links," *Proc. ACM Interactive Mobile Wearable Ubiquitous Technol.*, vol. 3, 2, pp. 1–23, 2019.

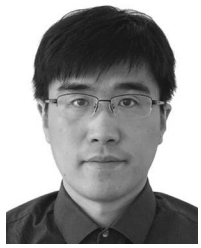
[39] J. Zhang, C. Zhang, X. Zhang, and S. Banerjee, "Towards a visible light network architecture for continuous communication and localization," in *Proc. 3rd Workshop Visible Light Commun. Syst.*, New York, NY, USA, 2016, pp. 49–54.

[40] K. Zhang et al., "ChromaCode: A fully imperceptible screen-camera communication system," *IEEE Trans. Mobile Comput.*, vol. 20, no. 3, pp. 861–876, Mar. 2021.

[41] O. Zhang, Z. Qian, Y. Mao, K. Srinivasan, and N. B. Shroff, "Er-scc: Enable efficient and reliable screen-camera communication," in *Proc. 20th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2019, pp. 281–290.

[42] X. Zhang, J. Liu, Z. Ba, Y. Tao, and X. Cheng, "MobiScan: An enhanced invisible screen-camera communication system for IoT applications," *Trans. Emerg. Telecommun. Technol.*, vol. 33 no. 4, Apr. 2022, Art. no. e4151.

[43] M. Zhou, Q. Wang, T. Lei, Z. Wang, and K. Ren, "Enabling online robust barcode-based visible light communication with realtime feedback," *IEEE Trans. Wireless Commun.*, vol. 17, no. 12, pp. 8063–8076, Dec. 2018.



Hao Han (Member, IEEE) received the BS degree in computer science from Nanjing University, China, in 2005, and the PhD degrees in computer science from the College of William and Mary, Williamsburg, Virginia, in 2014. He is now a professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics. His research interests include span the fields of mobile systems, industrial control systems, and connected autonomous vehicles with a focus on efficiency and security.

Kunming Xie received the BE degree from the Computer College, Hubei University of Technology, in 2020, and the MS degree from the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, in 2022. His research interests include visible light communication and mobile computing.

Tongyu Wang received the BE degree from the School of Software, Zhengzhou University, in 2019, and the MS degree from the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, in 2022. He is currently working with Huawei as a software engineer. His research interests include visible light communication and mobile computing.

Xiaojun Zhu (Senior Member, IEEE) received the BS and PhD degrees in computer science from Nanjing University, in 2008 and 2014, respectively. He is an Associate Professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics. From August 2011 to August 2012, he was a visiting scholar with the College of William and Mary. He is an associate editor for Computer Communications. His research interests include wireless networks, UAV networks, and smartphone systems.

Yanchao Zhao (Member, IEEE) received the BS and PhD degrees in computer science from Nanjing University, Nanjing, China, in 2007 and 2015, respectively. He is currently a professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing. Since 2011, he has been a visiting student with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA. His current research interests include wireless networks, mobile computing, edge computing, and device-free sensing.

Fengyuan Xu (Member, IEEE) received the PhD degree, with the Distinguished Dissertation Award, from the College of William and Mary. He is currently a professor with the Computer Science Department, Nanjing University. His research interests include the intersection of mobile systems, intelligent security, and autonomous software, with a focus on the cyber-physical continuum, trust and privacy autonomy, and AI empowered automation.