FareQR : Fast and Reliable Screen-Camera Transfer System for Mobile Devices using QR Code

Tongyu Wang†, Hao Han†, Zijie Wang†
†College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China
Email: tonywang@nuaa.edu.cn, hhan@nuaa.edu.cn, 1102831043@qq.com

2020 IEEE/ACM Symposium on Edge Computing (SEC) | 978-1-7281-5943-0/20/$31.00 ©2020 IEEE | DOI: 10.1109/SEC50012.2020.00053

***Abstract*—With the availability of mobile devices display and camera, Screen-Camera links for Visible Lite Communications (VLC) has attracted much more attention due to its convenience, infrastructure-free and security. In conventional Screen-camera link, the senders encode data bits into a barcode stream and receivers capture the barcode scream and decode the barcodes. But conventional Screen-camera communication systems all suffer from both CMOS rolling shutter and inter frame mixing problem when display rate is close to camera capture rate and this leads to a high block transfer error rate. In this paper, we propose a system called FareQR by adding an outline border to the barcode stream to help the receiver detect mixed frames and de-obfuscation each mixed frames into perfect QRCodes. We formulate the mixed frame de-obfuscation problem as a hard-decision-decoding task, and propose a Viterbi algorithm to resolve each block in the mixed areas. We test the FareQR and result demonstrate that our work can recover the mixed area and reduce the block transfer error rate.**

***Index Terms*—Visible light communication, color barcode, screen-camera communication, dynamic programming**

I. Introduction

The widespread digital displays and cameras bring screen-camera communication (SCC) between mobile devices a convenient medium to transfer information. With the advance in both camera and display technology, Visible Light Communication (VLC) over SCC has become a promising technique for short-range communication on smartphones. Compared with existing short-range communication such as WiFi, Bluetooth and Near Field Communication (NFC), SCC eliminates the complicated configuration of establishing communication links and authorization procedures and VLC is also functional in some extreme cases.

Imagine that medics at the point of injury (POI) rely on their mobile devices to exchange casualty care information in the battlefield. However, these capabilities have not yet been realized in the pre-hospital environment when communication networks are unavailable due to enemy's denial of service attacks. Thus, there is a critical need for an fast and reliable solution for the medical information exchange of pre-hospital care without RF communication.

The screen-camera communication technology addresses also enables many other useful applications (e.g., ease of file exchange) between mobile devices, thus receiving lots of research interests. There are tow parts of Screen-camera communications based on dynamic barcodes: Visual barcodes which are observable by human eyes ([2]–[10]); Imperceptible barcodes which almost invisible to the naked eyes due to the flicker-fusion property of Human Vision System([11]–[14]).In our work, we mainly focus on the visual barcodes.

The communication quality of screen-camera links is highly influenced by the capturing process, such as rolling shutter effect. Mobile devices these days almost all equipped with a CMOS sensor and a LCD or OLED screen. The image capture procedure is realized through line-by-line exposure of the CMOS sensor. At the beginning of the exposure, the image sensor scans the scene line by line until all pixels are exposed. Each LCD or OLED screen is divided into rows, rows are divided into pixels, pixels are displayed one by one. Among them, the LCD driver drives the LCD molecule of a pixel to rotate according to the display data given by the LCD controller, so that the pixel displays the corresponding color value. For an OLED, the driver directly controls the brightness of the corresponding pixel and then controls the corresponding color value. If the current frame is switching to a new frame, during the exposure time of the camera then the captured image is a mixed frame, the receiver will capture this mixed frame and this mixed frame is hard to decode. The conventional approach to solve this problem is to double the capture rate compared to the display rate so that there will be more than one clear frame in the two continuous captured frames,such as ([2], [4]).

In this work, we proposed FareQR to solve the rolling shutter issue. Different from the previous work, we add an outline for decoder to determine the position of the screen refresh as well as de-obfuscate the mixed area in a mixed frame. Although we can resolve the mixed area, but some QRCodes still miss some fragment and remain imperfect, we arrange the imperfect QRCodes into proper position and decode the QRCodes when gathering a intact QRCode. Specifically, FareQR makes the following contributions.

- We make an observation that there is a different response time on LCD screen between the pixel value changes from 0 to 255 and 255 to 0 and the response time on the negative liquid crystals TFT screen from 0 to 255 is faster than that from 255 to 0 .
- We propose an outline border framework with encoded RGB color to infer the screen refresh position and split a mixed frame into corresponding frames.
- We design a dynamic programming (DP) algorithm, seeking to recover QR blocks under transition from one color to another color in mixed frames.
- We devise a decoding algorithm that can merge differ-

978-1-7281-5943-0/20/$31.00 ©2020 IEEE
DOI 10.1109/SEC50012.2020.00053

ent parts into an intact barcode out of multiple mixed frames. Such an algorithm can significantly improve the transmission rate of our screen-camera transfer system.

II. RELATED WORK

A. Screen-Camera Communication Systems

Screen-camera communication attracts lots of research interests in recent decades.The basic idea of screen-camera communication is the sender display static or dynamic barcode on the screen which is captured by the receiver's camera to recover the original information.The 2D barcode such as QRCode [1], Data Matrix [16] have beem wildely adoped in different areas.In order to expand the capacity to convey more information, PixNet [17] proposed the original idea of using black-white barcode stream to achieve continuous communication over the screen-camera links. And colored barcode was proposed in ([15], [16]) to gain more capacity in spatial.Then Rainbar [6] makes improvement on block locating and frame synchronization to gain a better stability in transmission.RDCode [10] develops a barcode layout to handle the problems of locality and partial availability.SoftLight [9] devised a rateless coding scheme that tolerates the false positive of per-bit soft hints.Some of the works adopt the design of implicit barcode.HiLight [14] encodes data into pixel translucency change atop any screen content. Inframe [11] and InFrame++ [18] employed the spatial-temporal flicker-fusion property and CDMA-like modulation to embed data bits in video streaming without affecting users' video-viewing experience. ImplicitCode [13] proposed a comprehensive technique to achieve visually non-intrusive communication. TextureCode [19] further combined the techniques of HiLight and InFrame to improve the performance of communication basing on hidden barcode . ChromaCode [12] proposed an outcome-based adaptive embedding scheme, which adapts to both pixel lightness and regional texture.

B. Rolling Shutter Mechanism

Some related work proposed new solution to address rolling shutter mechanism. ERSCC [3]proposed a Time-domain Self-restoration coding to make sure that the blended color is in the color subspace, but it doesn't mention how to identify the two continuous captured is a same frame or not, and directly demodulate the continuous captured frame may confuses the raw data. MegaLight [5] adopts machine learning techniques to train a model based on the Mixture of Experts(MoE) to recognize block color, but it cannot tackle the barcode rotation very well because they draw a horizontal scan line across the rolling shutter indicator, when the barcode rotate, they will fail to locate the rolling shutter indicator. These above two works need to make sure the receiver and the sender start at the same time, in order to transmit necessary information. RainBar [7] and RainBar++ [6] uses the outline border to locate the mixed area when capturing the barcode displayed on a OLED screen and find four boundary points, but they don't apply their system on LCD screen, and LCD has a very high response time compared to the OLED screen, so the boundary points on LCD will be three or four and they don't mention how to deal with the frame which has only three boundary points. LightSync [8] put forward the method of gray scale tracking to restore mixed blocks but it has to make sure that the scan line match the row of block, and it can't deal with the sloping scan line.

III. BACKGROUND AND SYSTEM OVERVIEW

A. Rolling Shutter Mechanism And Frame Mixture

Image sensor in our mobile devices commonly use CMOS, which utilize rolling shutter mode to readout each pixel and any different pixels in different rows will be exposed with different beginning times. Similarly, the LCD and OLED refresh row-by-row with the refresh rate in range 60HZ-120HZ, different rows also refresh at different beginning times. When the CMOS scans the row which is updating the scene to the next frame, the captured image is resulted in a mixed frame. We define **Cut Line:** A slanted, vertical, or horizontal line with points on the line consisting of the intersection of multiple CMOS scan lines and screen refresh lines at the same time interval.

B. System Overview

FareQR framework is illustrated in Fig.1. At the sender side, there are two main sub modules, BarcodeGenerator and BorderGenerator. The BarcodeGenerator will generate an unique ID to represent the absolute location of this QRCode and embed the data bits into this QRCode. The BorderGenerator is added to generate RED, GREEND, BLUE outline border respectively outside three consecutive QRCodes after the QRCode sequence is generated. Finally, the sender loop play the whole QRCode sequence.

At the sender side, it consists of three main sub modules, FrameAlignment module, DataDecoder module and DataStorage module. First, the receiver will collect n frame as a group and send this group the FrameAlignment module.

The FrameAlignment module is used to realign unmixed area in to correct relative position and de-obfuscate mixed area in to two continuous imperfect QRCode. The DataDecoder module is used to decode unmixed frames, if any QRCodes in the group is successfully decoded , the dataDecoder will extract its data bits and ID and store the data bits , meanwhile, realign the other imperfect QRCode in order according to the ID. The DataStorage module is used to store the imperfect QRCode and combine the previous imperfect QRCode and new imperfect QRCode into one prefect QRCode and decode the prefect QRCode to obtain the data bits.

IV. SYSTEM DESIGN

In this section, we introduce the design detail of FareQR . Including the QRCode layout, FrameAlignment module, DataDecoder module , DataStorage module.

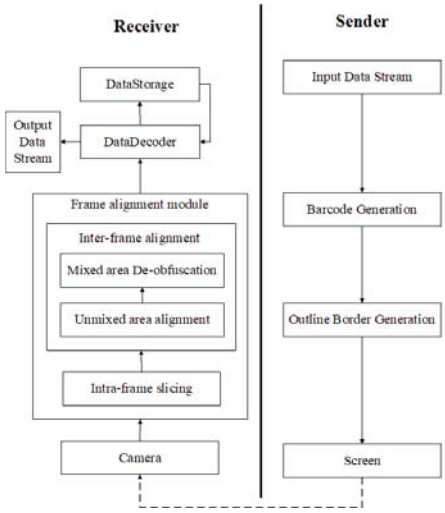

Fig. 1. System Architecture.

A. Layout design

One frame contains one data area and one outline border.

Data area: In the middle of the frame is a three dimension QRCode which is utilized to store the data bytes, Fig. 2. The three dimension QRCode has three channels red green and blue, and the smallest unit of data storage is block, which represents three bits and the corresponding colors in HEX are shown in the Table I:

Outline border: The outline borders only have three colors red, green and blue. The reason to choose these colors is to make sure there is one channel goes from 0 to 255, one channel goes from 0 to 255 and one channel remain 0 when the screen refresh into a new frame. Assume the whole QRCodes scream contains N QRCodes, to make sure the outline border color is different between the first QRCodes and the last QRCodes, we extend the number of total QRCodes from N to the least common multiple of N and 3 which is N', and loop the N QRCodes into N' QRCodes.

Fig. 2. Color QRCode layout.

B. Frame alignment module design

In this section, we propose the design of FrameAlignment module at the receiver side which is applied to detect unmixed and mixed area and then arrange the unmixed area into proper

TABLE I
COLOR AND CORRESPONDING DATA

Block value	Color
000	0xFFFFFF
111	0x000000
100	0x00FFFF
010	0xFF00FF
001	0xFFFF00
110	0x0000FF
011	0xFF0000
101	0x00FF00

position,the input of the this module is n captured frames and the output are m continuous imperfect/perfect QRCodes. The design principle of this module is that every pixel in the same Cut Line has the same value, more specific, assume the outline border is changing from red to green, so the red channel descends from 255 to 0 and green channel ascends from 0 to 255, so the refreshed value from 0 to 255 is p and refresh value from 255 to 0 is q, and for any channel within this refresh line whose ascending value is supposed to be q and in the same way, the declining value is supposed to be q. The receiver will not lose any QRCode frame but may capture two different fragments from two consecutive QRCodes or one mixed area between two QRCodes when the display rate is little bit lower than the capture rate. This module consists of two parts: **Intra-frame slicing** and **Inter-frame alignment**.

a) Intra-frame slicing: This step is designed to find the PreArea, MixArea and Postarea in a mixed frame, and split MixArea into several subgroup as well as calculate the distance between Mixarea and blocks within both PreArea and Postarea. After localizing the QRCode, we are able to apply perspective transformation based on the three positioning points and right bottom alignment pattern on the right bottom corner to sample the outline border in the order of top right corner, top left corner, bottom left corner, and bottom right corner. But the sampled points are disorder owing to effect of the ambient light and capture distortion. We apply Savitzky-Golay for smoothing filter to obtain the filtered data list named FDL from RGB channels respectively and store them into three lists RFDL , GFDL, BFDL and calculate the standard deviation of RFDL, GFDL, BFDL as std(RFDL), std(GFDL), std(BFDL). Based on the three lists, we are able to distinguish this frame between a mixed frame and a unmixed frame by setting a threshold MIXTHRESHOLD. If any of std(RFDL), std(GFDL), std(BFDL) is larger than the MIXTHRESHOLD, the corresponding channel is a changing channel. If there exists no less than one changing channel then this frame a mixed frame and the others are unmixed frame. As for a mixed frame, we need to find the none changing channel, whose FDL value keeps very low and minimum variation. Though the none changing channel we can deduce the ascending channel and descending channel (eg, if the none changing channel is G, and according to the RGB sequence, the G channel only keeps low when the outline border's color changes B into R, so B channel changes from 255 to 0 and R channel

changes from 0 to 255, similarly if the none changing channel is B,then R channel changes from 255 to 0,and G channel changes from 0 to 255).We set a threshold THRESHOLD to split one changing channel into two sub parts.And we name the channel whose value changing from 0 to 255 an ascending channel and the channel whose value changing from 255 to 0 an descending channel. For the declining channel, the sub part value greater than the THRESHOLD represents the previous QRCode border, and the sub part values less than the THRESHOLD is the post QRCode border. On the contrary, for the ascending channel, the sub part value greater than the threshold belongs to the post QRCode border and the smaller one belongs to the previous QRCode border. Now we have four sub parts,then we map these four sub parts based on the index value in the list to the 2D QRCode border in one frame and use Bresenham algorithm to draw a line with a width of one block to connect the first point and the last point in one sub part. Now we have drawn four polygons in one frame, and we named these polygons as following.

- preLarger: The blocks surrounded by the declining channel whose sub part value greater than the THRESHOLD and a Bresenham line.
- preSmaller: The blocks surrounded by the declining channel whose sub part value smaller than the THRESHOLD and a Bresenham line.
- postLarger: The blocks surrounded by the ascending channel whose sub part value greater than the THRESHOLD and a Bresenham line.
- postSmaller: The blocks surrounded by the ascending channel whose sub part value greater than the THRESHOLD and a Bresenham line.

The PreArea is the intersection of PreLarger and PostSmaller, Postarea is the intersection of PostLarger and PreSmaller; MixArea is a complement of the intersection of PostArea and PreArea in all blocks. And there are n frames in one group so this procedure will repeat n times in one decoding session. For blocks in Mixarea, the channel value is not steady low or steady high but a trend of rising or falling, so we need to know how much does the channel value have changed. Thanks to the outline border, it represents the different p,q in different Cut Line in one frame. We group up the similar p or q in to one Cut Line and set the corresponding blocks the same p and q. In the PreArea and PostArea, we do not need to calculate p and q, instead we calculate the distance between the block and Mixarea using equation (1) where $P(x,y)'$is the block coordinate in PreArea and PostArea, $Q_1(x,y)'$and $Q_2(x,y)'$ are the intersection of the boundary and outline border of the mixed zone and the unmixed zone, det(m) is to calculate the determinant of matrix.

$$d = \left| \frac{det(\begin{bmatrix} Q_2 - Q_1 & P - Q_1 \end{bmatrix})}{\parallel Q_2 - Q_1 \parallel_2} \right| \tag{1}$$

b) Inter-frame alignment: The requirement is that the captured n frames can be alignment in to m imperfect/perfect QRcodes. From the previous step, We have a list of n frames, where each frame is divided into three non-intersect area. Based on the four channel status, we define FD: The relative changes between two frames. For example, if the border changes from red to green or the border color is a gradient from red to green becomes a gradient from green to blue in two consecutive frames, it means that the FD is 1. And if the border color is a gradient from red to green becomes green,then the FD is 0.5 and vice versa. With FD and n frames, we can align the n frames into m QRcodes. This step also contains two parts **Unmixed area alignment** and **Mixed area De-obfuscation**.

Unmixed area alignment: In this part, we only take PreArea, PostArea and ClearArea into account. Assume the index of one frame is i and the FD between previous frame and new frame is K which is 0,1 or 1.5 and its Preareas and ClearAreas are placed in the corresponding position of i^{th} QRCode and the PostArea is placed in the corresponding $(i+K)^{th}$ QRCode. If FD divided by 1 has a remainder of 0.5, the new frame will modify the ith or $(i+K)^{th}$ QRCode. Now we get a list of m QRCode fragments but these m QRCode fragments may not decodable, the following part will specify how to repair these imperfect QRCodes.

Mixed area De-obfuscation: Each mixed area consist of two continuous frame and our goal is to extract one mixed area into two independent unmixed area. To be mentioned that, in n consecutive frames, the mixed area may also be continuous, thus we cannot directly De-obfuscate one mixed area into two clear area frame by frame because the de-obfuscated frames affect each other, for example, the area that belongs to the next frame after the de-obfuscation of the first frame is the area of the previous frame that is de-obfuscated in the second frame. Instead, the n frames should be taken in to account together to extract m unmixed area. Recall that each MixedArea is divided into several groups and each of them has its own p and q. For each color channel in one block we have the equation (2), assuming the sampled channel value is S.

$$\begin{cases} d_1 = & |0 - S| \\ d_2 = & |p - S| \\ d_3 = & |q - S| \\ d_4 = & |255 - S| \end{cases} \tag{2}$$

Each d_1 to d_4 is formulated as the distance between the ideal four value, also the weight between two unmixed channel shown in the figure 5, and any n mixed channel are able to extract n+1 unmixed channel. We formulate this problem as a hard-decision-decoding problem using Viterbi algorithm. The trellis diagram definition detail is described in Table II We proposed a De-obfuscation algorithm based on Viterbi algorithm to solve this problem , see equation (3). Assume there are n stages and $k+1$ vertexes, this algorithm will find the shortest path from the first stage to the last stage and the vertexes in the path are the unmixed channel value in the unmixed block. What's more, each block has three channels, so we repeat this algorithm three times to resolve all the data

TABLE II
TRELLIS DIAGRAM DEFINITION

V	All unmixed channel value candidates in the n mixed channel. Each vertex has an id.
E	All weights calculate in formula above
$G =< V, E >$	Trellis diagram construct with V and E,shown in Fig. 5
$Stage_i$	One unmixed channel, and there are only two vertices namely 0 and 255. And there are 4 path between two continuous stages. $Stage_{end}$ is the last stage and $Stage_{start}$ is the first stage.
Cost(i,j)	The minimum distance between $vertix_j$ in $Stage_i$ and $Stage_{end}$.
$C_{(i,j)}$	The distance between $vertix_i$ and $vertex_j$.

block in the mixed areas.

$$\begin{cases} cost(n,k) = cost(n,k+1) = 0 \\ cost(i,j) = \min\limits_{\substack{j \in stage_i \\ p \in stage_{i+1} \\ <j,p> \in E}} \{C_{j,p} + cost(i+1,p)\} \end{cases} \quad (3)$$

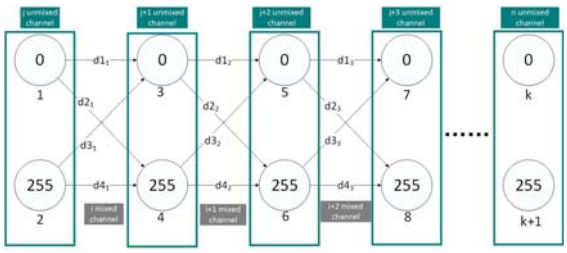

Fig. 3. Trellis diagram construct with V and E.

Here is an example of how this Viterbi algorithm and outline border work. Assume we have already fond the mixed zone in these four QRCodes as well as obtained the sub groups of a mixed area. Fig.4 shows one sub group from the four mixed areas, and each sub group has it own p and q fetched from the outline border which are represented the blue blanks. The blocks S_1 to S_4 are the four blocks we are about to de-obfuscate and these blocks have the same coordinate. in 1^{th} mixed frame the border color is changing from blue to red,in 2^{th} mixed frame the border color is changing from red to green,in 3^{th} mixed frame the border color is changing from green to blue,in 4^{th} mixed frame the border color is changing from blue to red. We only show how to de-obfuscate the red channel in S_1 to S_4, the de-obfuscation of green and blue channel are the same. The trellis diagram in 6 shows all the available path of the five red channel value after de-obfuscate the four mixed red channel value. After using the Viterbi algorithm, we can find the minimum path from 1^{th} unmixed channel to 5^{th} unmixed channel, and the vertexes in the path are 0,255,0,0,255 which are the same as the original red channel values.

C. DataDecoder module design

The data decoder module is used to decode QRCode data, extract QRCode ID and realign the imperfect QRCode into its real position. From the Frame alignment module, we have obtained the m QRCodes. Then we traverse the m QRCode and check if any of them can pass the RS check, if there exists more than one QRCode pass the check, we will extract the raw byte data and QRCode ID to export the byte data into the corresponding position based on the QRCode ID and then rearrange the other failed decoded QRCode based on this ID. For example, when m = 5, and the 4_{th} QRCode is correctly decoded, and its ID is 10, then the total 5 five QRCode ID is rearranged as: 7,8,9,10,11. If none of the m QRCodes can be successfully decoded, we have no choice but drop them.

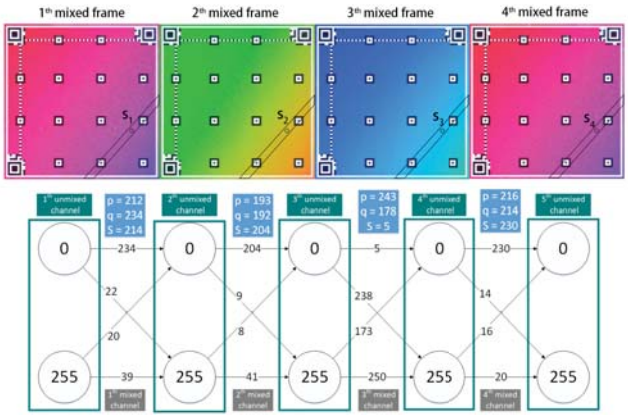

Fig. 4. An example of Viterbi algorithm.

D. DataStorage module design

After the DataDecorder module, we need to store the m QRCodes for further decode. As soon as we achieve this module, we have already decoded some of the QRCodes and obtain the amount of the total QRCodes. Then FareQR will initialize a memory space to store all the QRCodes. For any QRcode which passed the RS check, the ID of that memory space is mark "decode Complete" and that space will not store any QRcode fragments of that QRCodes. For the QRcodes which don't passed the RS check, DataStorage module will first remove the area which is extracted from Mixedareas and decode this QRcode until receive this QRcode again. Once The a new QRcode comes to this module, this QRCode may have the different value compare to the previous QRcode. Recall the previous section, we have calculated the distance between each unmixed block and the mixed area, based on this distance, we proposed algorithm 1 to decide which block to store. With the receive session proceeding, this module will keep storing the new QRcode and updating the exiting QRcode.

V. EXPERIMENTAL RESULTS

We conduct experiments to evaluate the performance the Frame alignment module and the FareQR . The experiment is conducted under normal indoor lighting condition(100LUX). The distance between receiver and sender is set to 10cm. One frame contain one color QRCode which has 77*77 blocks in

Algorithm 1 Decide which one to store

Input: $QRcodelist$ //contains m QRcode

Output: $storeBlock$ //to store the previous block or new block

1: $TotalDistance = preDistance + newDistance$
2: $Precentage = \frac{preDistance}{TotalDistance}$
3: $a = rand()$ //generate a random number from 0 to 1
4: **if** $a < Precentage$ **then**
5: return $newBlock$ //store new block
6: **else**
7: return $previousBlock$ // store previous block
8: **end if**

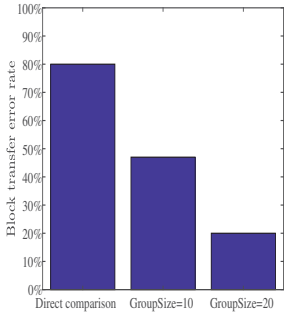

Fig. 5. Block transfer error rate in different group size

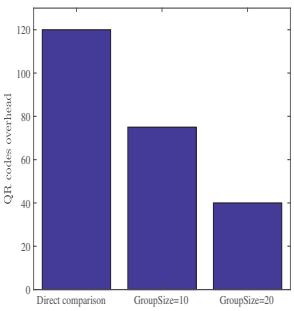

Fig. 6. Number of QR codes involved in decoding

the data area, exclude the outline border. The Sender is Huawei P10 with TFT. Receiver is OnePlus 7 with a video capture rate of 30 fps. The camera capture resolution is set to 1920*1080 and the sender display the QRCode sequence with the frame rate of 27fps and the brightness of the display is set to 100%. We transmit 60 QRCodes in this experiment. We define *block transfer error rate:* the percentage of error block in one QRcode. *QRcode overhead:*The number of QR codes involved in decoding exceeds the number of QRCodes sent. From the result in Fig. 5, after applying the Frame alignment module,the block transfer error rate decrease significantly because areas that do not belong to the same QRCode in the same frame will be correctly arranged in the corresponding position and the mixed area can be split into two continuous unmixed areas, and the larger the group size, the lower the block transfer error rate. From the result in Fig. 6, if we do nothing then some of the QRCodes are always obfuscate, so more loop is needed to decode the QRCode.With FareQR , the QRcode overhead dropped a lot because the mixed frames can be correctly de-obfuscated and arranged. This paper proposed a new method to resolve the mixed frame and realign the unmixed QRcode into correct position. We implement the framework on color QRCode and test the the Frame alignment module by using MATLAB and result demonstrate that our work can recover the mixed area and improve the block error rate. And we will implement the rest of this work on Android and test its performance in the future.

REFERENCES

[1] ADC Denso. 2011. QR code essentials. Denso Wave 900 (2011).

[2] Tian Hao, Ruogu Zhou, and Guoliang Xing. 2012. COBRA: color QRcode streaming for smartphone systems. In Proceedings of the 10th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys'12). ACM, 85–98.

[3] Ouyang Zhang, Zhenzhi Qian, Yifan Mao,etc. 2019. ERSCC: Enable Efficient and Reliable Screen-Camera Communication. In Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc '19).

[4] X. Chen, W. Li, T. Zhan etc, "MMCode: Enhancing Color Channels for Screen-Camera Communication with Semi-Supervised Clustering," 2018 27th International Conference on Computer Communication and Networks (ICCCN)

[5] Tong Zhan, Wenzhong Li, Xu Chen,etc. 2019. MegaLight: Learning-based Color Adaptation for Barcode Stream Recognition over Screen-Camera Links. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 3, 2, Article 66 (June 2019), 23 pages.

[6] Q. Wang, M. Zhou, K. Ren,etc, "Rain Bar: Robust Application-Driven Visual Communication Using Color Barcodes," 2015 IEEE 35th International Conference on Distributed Computing Systems, Columbus, OH, 2015, pp. 537-546.

[7] M. Zhou, Q. Wang, T. Lei, etc, "Enabling Online Robust Barcode-Based Visible Light Communication With Realtime Feedback," in IEEE Transactions on Wireless Communications, vol.

[8] Wenjun Hu, Hao Gu, and Qifan Pu. 2013. LightSync: unsynchronized visual communication over screen-camera links. In Proceedings of the 19th annual international conference on Mobile computing and networking (MobiCom'13). ACM, 15–26.

[9] Wan Du, Jansen Christian Liando, and Mo Li. 2016. Softlight: Adaptive visible light communication over screen-camera links. In Proceedings of the 35th Annual IEEE International Conference on Computer Communications (INFOCOM'16).

[10] AnranWang, Shuai Ma, Chunming Hu, etc. 2014. Enhancing reliability to boost the throughput over screen-camera links. In Proceedings of the 20th annual international conference on Mobile computing and networking (MobiCom'14). ACM, 41–52.

[11] Anran Wang, Chunyi Peng, Ouyang Zhang, etc. 2014. Inframe: Multiflexing full-frame visible communication channel for humans and devices. In Proceedings of the 13th ACM Workshop on Hot Topics in Networks (HotNets'14). ACM, 23.

[12] Kai Zhang, Chenshu Wu, Chaofan Yang, Yi Zhao, etc. 2018. ChromaCode: A Fully Imperceptible Screen-Camera Communication System. In Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom'18). ACM, 575–590.

[13] Shuyu Shi, Lin Chen, Wenjun Hu,etc. 2015. Reading between lines: high-rate, non-intrusive visual codes within regular videos via ImplicitCode. In Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp'15). ACM, 157–168.

[14] Tianxing Li, Chuankai An, Xinran Xiao,etc. 2015. Real-Time Screen-Camera Communication Behind Any Scene. In Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '15).

[15] Devi Parikh and Gavin Jancke. 2008. Localization and segmentation of a 2D high capacity color barcode. In IEEEWorkshop on Applications of Computer Vision (WACV'08). IEEE, 1–6.

[16] ISO/IEC 16022:2000 2000. Information technology – International symbology specification – Data Matrix. Standard. International Organization for Standardization, Geneva, CH.

[17] Samuel David Perli, Nabeel Ahmed, and Dina Katabi. 2010. Pixnet: Lcd-camera pairs as communication links. In ACM SIGCOMM Computer Communication Review, Vol. 40. ACM, 451–452.

[18] Anran Wang, Zhuoran Li, Chunyi Peng, etc. 2015. Inframe++: Achieve simultaneous screen-human viewing and hidden screen-camera communication. In Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys'15). ACM, 181–195.

[19] Viet Nguyen, Yaqin Tang, Ashwin Ashok,etc. 2016. High–Rate Flicker–Free Screen–Camera Communication with Spatially Adaptive Embedding. In Proceedings of the 35th Annual IEEE International Conference on Computer Communications (INFOCOM'16).