ARTcode: Preserve Art and Code In Any Image

Zhe Yang[*], Yuting Bao[*], Chuhao Luo[*], Xingya Zhao[*], Siyu Zhu[*], Chunyi Peng[†], Yunxin Liu[§], Xinbing Wang[*]

[*]Shanghai Jiao Tong University, China, [†]The Ohio State University, USA, [§]Microsoft Research, China

Email: {youryz, smiling-forever, leeklch, claret.zhao, zhusy54}@sjtu.edu.cn, chunyi@cse.ohio-state.edu, yunxin.liu@microsoft.com, xwang8@sjtu.edu.cn

ABSTRACT

The ubiquitous QR codes and some similar barcodes are becoming a convenient and popular approach to impromptu communication between mobile devices and their surrounding cyber-physical world. However, such codes suffer from two common drawbacks: poor viewing experience and inability to be identified through itself. In this work, we propose ARTCODE– **A**daptive **R**obust do**T** matrix bar**code**, which aims to preserve ART and CODE features in one visual pattern. It works on any surface (paper or electronic displays) and is able to convert any image or any form of human-readable contents (*e.g.*, a picture, a logo, a slogan) into an ARTCODE. It *looks like* an image which retains human-readable and aesthetically pleasant contents, and in the meanwhile, it *acts as* a QR code which conveys data bits over the visual channel. The core enablers in ARTCODE are (1) the design of the colored dot matrix for data embedding with little distortion from the original image and (2) a comprehensive error correction scheme which enhances decoding robustness against noises and interferences from the original image in ARTCODE. We implement ARTCODE with the receiver on Android phones and the sender from a PC or a phone (it can be printed in paper). We conduct extensive user survey and experiments for evaluation. It validates the effectiveness and wide applicability of ARTCODE: It works well with all of 197 images randomly downloaded, covering representative categories of the gray-scale images, logos, colored ones with low/medium/strong contrasts. The image quality is quite acceptable in a subjective user-perception survey with 50 participants and data communication accuracy achieves as high as 99% in almost all the cases ($>$ 96% raw accuracy in ARTCODE without error detection and other schemes).

Author Keywords

Screen-camera communication; visualization barcode; data hiding; ARTCODE.

ACM Classification Keywords

H.5.2. Information Interfaces and Presentation (e.g. HCI): User Interfaces

UbiComp '16, September 12-16, 2016, Heidelberg, Germany

 ISBN 978-1-4503-4461-6/16/09…$15.00

DOI: http://dx.doi.org/10.1145/2971648.2971733

(a) QR code (original) (b) QR code + a thumbnail (c) ARTCODE

Figure 1: Three visualization code examples encode the same information: a URL to the Wikipedia page of Lenna: *https://en.wikipedia.org/wiki/Lenna*.

INTRODUCTION

Scanning is a novel, fast, and convenient approach for *impromptu communication* between mobile phones and their surrounding cyber-physical world. Information is encoded in special visual patterns (*e.g.*, Quick Response (QR) codes [4]) and displayed on the sender surface in various form factors, ranging from the exterior of a physical object, a poster/printout to an electronic display such as a phone screen, a computer monitor, a TV, and even a multiple-screen display wall. Without need of any device pairing or communication setup beforehand, mobile phones simply scan these patterns and directly extract data bits therein. With inherent simplicity, scanning has been widely adopted to deliver information (especially short messages like URLs and object identities) in a variety of domains across project tracking, item identification, mobile advertising, publicity, payment, eCommerce, social networking, and in-proximity data sharing, to name a few. Recent statistics reports that the download number of Android scanner apps has already exceeded 100 million by November 2014; "`Scan.me`" [5], one popular scanner app, has carried out 87 million QR code scans [24].

Scanning is easy but *knowing me* is hard. While QR code is machine-friendly to facilitate fast data transmission, it is not quite consumable by humans to distinguish or identify the code they scan through itself. As shown in Fig. 1a, QR codes, as well as some other similar ones, employ random permutations of black and white modules (square dots). It is hard, if not impossible, for humans to read the code and then relate it with the context or the information that it conveys. Additionally, it is aesthetically unpleasant while it is placed with human-readable content (*e.g.*, text, logo, picture, video) [29].

It is often highly desirable for the user to *know* what is being *scanned*. We often demand for human-readable (human-

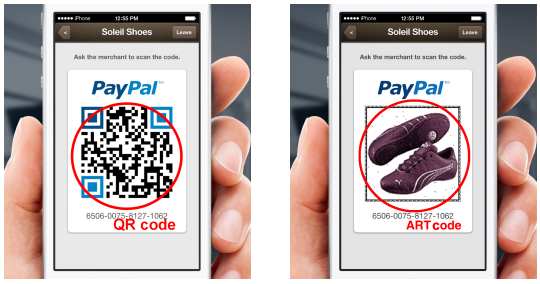


Figure 2: A motivating example (Paypal link for Puma Soleil shoes) uses two-in-one visual patterns (here, ARTCODE) (right) which combine data bits into human-identifiable contents, compared with the QR code-based solution (left).

friendly) information along with this device-oriented data communication channel. A common real-life example is payment, as illustrated in Fig. 2. The current practice (left) is to separate the context targeted to humans (here, image of the shoes which is not shown on this payment page) from the QR code that embeds the URL to its extended information like the website of a company or the link to pay an order. Such practice suffers two problems. First, both contend for the display space. A sacrifice has to be made on human-readable content as shown on the left. Human has to count on previous context to know what we are paying (doing) and makes a temporal compromise. Such activity is viable only in interactive scenarios when the screen can change, not possible for the printout or the static screen. Second, it impairs user experience. It is distractive and unpleasant to human eyes even when it takes a small area. Moreover, it loses the opportunity to deliver or highlight the logo, brand, slogan, goods and other essential information that the merchants care most. Clearly, a two-in-one visual code can easily resolve these issues. If such code exists, it would *look like* an image or any form that carries human-readable content (see the right plot of Figure 2); It would retain normal user-viewing experience to the best extent possible. On the other hand, this code would *act as* a QR code and convey side information over the visual display-to-camera channel.

All these motivate us to design **ARTCODE**, **A**daptive **R**obust do**T** matrix bar**code**, which carries the human-friendly information (mostly, artistic images and layouts) and machine-friendly information (coded data bits) together in the same source over the visual channel. It preserves pleasant user-viewing experience (as if the device-oriented code would not exist), while be able to simultaneously empower convenient camera-assisted data communication.

To this end, we devise a visualization approach that can turn *any image* into a code-embedded image. In particular, ARTCODE is desired to resolve the tension between data embedding and image preserving. Intuitively, less distortion of original image leads to smaller or fewer embedded modules, bringing in difficulties for scanning process. Machine-readable codes may appear conspicuous to human eyes. Therefore, the major issue is to achieve human-readable code design with implementing the functionality of code-camera data transmission. To address this issue, we devise techniques to tackle technical problems in encoding and decoding:

On the encoding side, we introduce the conception of colored dot matrix, that forms original image by blending neighboring dots. We intend to embed data by altering partial dots. The dot matrix is obtained with an error diffusion dithering, based on a color set adaptively generated by K-means clustering. Therefore, this dot matrix is able to preserve images with facilitating data embedment. We further adopt a data embedment scheme within a matrix proposed in [10], and integrate it into code design. The scheme embeds multiple bits into a block with only one bit altered, enabling large capacity transmission with little damage on image. It was designed for the purpose of data hiding within an image. We adopt its notion and employ the approach within a wireless communication issue. Finally, we design a scheme that selects encoding colors based on both the proportion of a color in whole image and two colors' distances in RGB space. The first criterion preserves code's image quality and the second criterion guarantees that code can be successfully detected and decoded in receiver side. Besides, in order to enable reliable data transmission, we design a reliable ARTCODE detection scheme and a comprehensive error correction scheme.

On the decoding side, we perform code detection and then decoding and gradually recover the codes within the background image noise. We perform preprocessing and module localization in context of ARTCODE and perform comprehensive error correction.

Fig. 3 shows the architecture of ARTCODE. It consists of the sender for code generation and the receiver for detection and decoding. The details will be elaborated in the following sections. We implement ARTCODE's sender on the PC and Android phones and its receiver on Android phones.

We evaluate ARTCODE through extensive experiments and a subjective user study with 50 participants. We show that ARTCODE is widely applicable to any image which preserves nice image quality and retains high data communication accuracy. We also assess ARTCODE under various settings and find it works well under various conditions with appropriate parameter settings. Basically, the image quality is below 4 (mostly under 3) out of 10 in terms of image distortion and the accuracy is as high as 99% when ARTCODE is fully used. It achieves 96% raw accuracy when data are encoded into colors.

We make three main contributions in the paper:

1. We design ARTCODE which enables two-in-one visual communication for both humans and devices;
2. We propose a variety of techniques including adaptive colors, dithering, shuffling, data-hiding to achieve both goals;
3. We implement and evaluate ARTCODE in extensive user surveys and experiments and validate its effectiveness, robustness and nice properties in visual-effect preservation and information-carrying.

PRESERVE ART IN ARTCODE ENCODING

The core problem is how to modify the original image so that such changes are significant enough for the camera to recover embedded data still without obvious distortion in visual effects. ARTCODE largely preserves the original human-readable im-

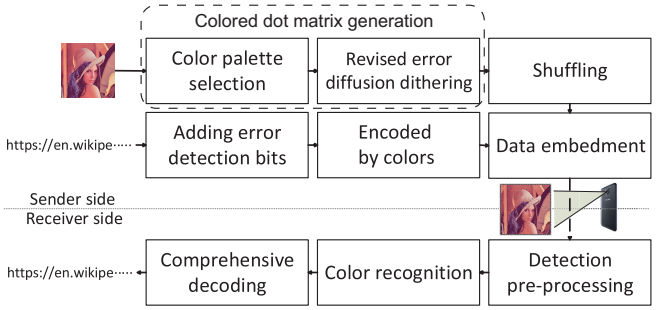


Figure 3: ARTCODE architecture.

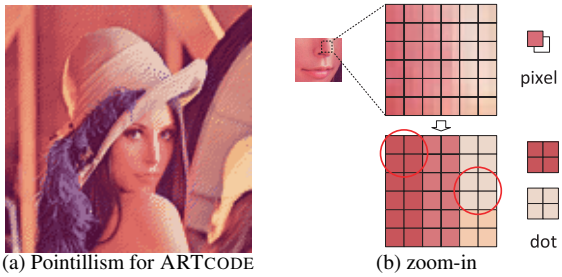
age contents. To address this tension between both communications for humans and devices, we tackle the following three technical questions in encoding:

(1) How to make changes to an image to largely keep its form and enable data embedment? This relates how to transform an image into a code structure that maintains the form of original image and allows partial changes at proper positions to encode data. *(2) How to reduce visual distortion in data embedment?* Embedded information targeting at cameras is regarded as noises to human eyes. In order to well preserve the original image, the introduced *noises* should be made less visually conspicuous. *(3) How to manipulate colors to encode data?* In colored ARTCODE, which colors should be selected to encode data? This has two requirements: a) selected colors should be adaptive to images and introduce minor distortion to images; b) selected colors should be distinguished by cameras so as to encode data. In the following three subsections we propose our encoding scheme addressing these technical questions.

Colored Dot Matrix and Shuffling

The first issue is to transform any image into a structure suitable for data embedment while still retaining the basic image effect, and to determine where to embed data in this structure. We are motivated from one drawing/painting art technique called pointillism and transform any image into a combination of small dots, some of which can be altered to encode data. Pointillism is a technique of painting in which small, distinct dots of color are applied in patterns to form an image. The technique relies on the ability of the eye and mind of the viewer to blend the color spots into a fuller range of tones [7]. The next step is to decide which dots can be altered. In order to disperse visual distortion to the whole image, we perform a shuffling algorithm where we refer to a shuffling table to find embeddable modules.

Colored Dot Matrix Generation. ARTCODE uses a pointillism-like blending technique to generate a structure, maintaining original images' general appearance. By extracting major colors of the original image to form a color palette and performing a dithering algorithm based on this palette, ARTCODE converts an image into a colored dot matrix, which displays the original image by blending colors of neighboring dots (we call modules in ARTCODE). Dot matrix is designed to be squares, facilitating the implementation of barcode communication. Data can be easily embedded into this dot matrix through changing the colors or positions of partial dots.



(a) Pointillism for ARTCODE (b) zoom-in

Figure 4: (a) ARTCODE uses a pointillism-like technique to convert any image into ARTCODE structure with many small dots; (b) a zoom-in of original image (above) and a zoom-in of ARTCODE structure (below).

Colored dot matrix generation consists of the following two phases.

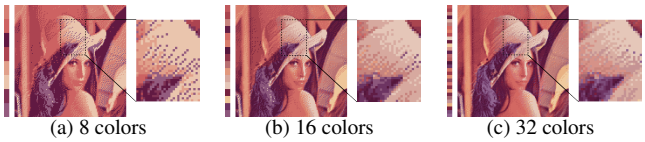
Phase 1. Color palette selection by clustering: Colored dot matrix is an image form that maintains the original images' major information. Embedding data into a colored dot matrix composed of a few major colors of images is less conspicuous than embedding data into the original images. The reason behind is that encoding modules[1], which also take these major colors, look more "similar" to their backgrounds and appear less abrupt, while the original images' major information can be preserved by non-encoding modules.

In order to best display the original image with a dot matrix of fewer colors, we select the colors that represent a majority of the original image to form a color palette. Specifically, to generate such a color palette, we use K-means clustering that partitions all colors in the original image into n clusters so that the original image can be preserved with these n colors. With pixels' RGB values in a colored image being the input, K-means clustering firstly assigns n random 3-dimensional (RGB space) vectors as initial centers, from which the algorithm finds a nearest center for each color and assigns each color a cluster accordingly. Then, K-means obtains the new centers by calculating the average RGB values of colors belonging to the same cluster. The process undergoes several iterations until convergence is reached. We take the image of Lenna as an example, illustrated in Fig. 5, where extracted colors are shown in palettes and n is chosen among 8, 16 and 32.

Phase 2. Revised error diffusion dithering: We use a revised error diffusion dithering algorithm to generate a colored dot matrix with adaptive color palette obtained previously.

Dithering is a technique that converts complicated colored or gray-scale images to images with a palette of much fewer colors, retaining original images' appearance. Error diffusion dithering quantifies RGB values of each module to colors in palette and delivers quantification errors to neighboring

[1]There is a slight difference between what we call *embeddable modules* and *encoding modules*. *Embeddable modules* refer to modules that can be manipulated to embed data, while *encoding modules* refer to modules that are already manipulated with data embedded. However, in Shuffling part, this difference can be ignored, for finding encoding modules is equal to finding embeddable modules.



(a) 8 colors (b) 16 colors (c) 32 colors

Figure 5: Colored dot matrix forms of Lenna generated by K-means color palette selection and revised error diffusion. Color cluster numbers in each palette are 8, 16 and 32, respectively.

Input: Colored image I, Color palette C
Output: Dithered image *Modules*
Divide I into groups, each group contains 4×4 modules
Initiate *value* to be a 3 dimensional matrix of all 0s
for *each group (i, j) of this image* **do**
  $ave \leftarrow$ average of pixels' RGB values in a group
  $value_{i,j} \leftarrow value_{i,j} + ave$
  $m \leftarrow \arg\min_m Dist_{Euc}(value_{i,j}, C_m)$
  $Modules_{i,j} \leftarrow C_m$
  $error \leftarrow C_m - value_{i,j}$
  $value_{i+1,j} \leftarrow value_{i+1,j} + error \times \frac{3}{8}$
  $value_{i,j+1} \leftarrow value_{i,j+1} + error \times \frac{3}{8}$
  $value_{i+1,j+1} \leftarrow value_{i+1,j+1} + error \times \frac{1}{4}$
  update i, j
end

Algorithm 1: Revised error diffusion dithering

modules. Note that traditional error diffusion dithering is performed on colors with equidifferent RGB values, which does not hold in our case since the colors obtained with K-means rely only on the original image and may not be equidifferent. We therefore develop a revised error diffusion algorithm, the detailed algorithm of which is given in Alg. 1, where *value* is a 3-dimensional matrix representing the error to be diffused to neighboring modules. Note that C, *ave*, *value* and *Module* all represent RGB 3 channels. Parameters $\frac{3}{8}$ and $\frac{1}{4}$ are common diffusion proportions. Fig. 5 shows dithered dot matrix generated by 8, 16 and 32 clustered colors.

Shuffling. Once we obtain the colored dot matrix awaiting embedment, one crucial problem emerges: which modules in this dot matrix are manipulated as encoding modules and encode data. One direct approach is to encode data on all modules of this dot matrix. However, with this approach image could hardly be preserved. We perform a shuffling algorithm which spreads encoding modules over the dot matrix according to a shuffling table, so that visual distortion caused by encoding is dispersed. Particularly, ARTCODE finds embeddable modules by referring to a shuffling table, which is an L-length vector, the values of which are randomly generated. The same shuffling table is informed to both sender and receiver a priori to find encoding modules and embed data onto or extract data from these shuffled modules. The modules localized by shuffling become encoding modules while the rest are non-encoding ones. Embedding process will only deal with encoding modules, and non-encoding ones remain unchanged throughout the embedment process.

An important advantage of employing shuffling table as embedding key lies in that, different users scan an ARTCODE and obtain different messages. ARTCODE embeds information to positions determined by shuffling table and the receiver side uses shuffling table to localize embedding positions, enabling multiple messages transmission. ARTCODE releaser can embed several messages into one code and ARTCODE scanners can only obtain messages whose shuffling table they possess. This may bring promising application to electronic payment. QR codes or some similar barcodes are currently utilized for communication between sellers and buyers on their mobile payment applications. Considering the variety of electronic payment service providers, merchants have to display several companies' QR codes. However, with ARTCODE, related companies can form a unity and publish only one code while customers have access to all payment links of these companies.

Embedding Data with Minor Modifications

Once we obtain a structure suitable for embedment, how to embed data with tiny distortion is to be determined. In a human-recognizable and machine-readable dual-channel visualization code, the embedment of machine-readable information in the dot matrix is regarded as noises for human perception. Therefore, a demand for preserving original image is to embed data of large capacity with as few changes to colored dot matrix as possible. In order to trade off the capacity and visual quality demands of ARTCODE, we integrate the method proposed in [10] and adapt it to the present work.

The main idea in [10] is to divide the ensemble of encoding modules into unit blocks and use position information of modules in a block to encode data. Each module position in a block is given a weight; to embed data of more than 1 bit requires changing only 1 module in position of corresponding weight. The authors proposed that in a binary matrix, an l-length (l can be any positive integer) block embeds $k = \lfloor \log_2(l+1) \rfloor$ bits by changing at most one bit, for their are in total l possible position changes, plus 1 original unchanged block.

We divide the set of L embeddable modules obtained in shuffling into subsets of modules, and call these subsets blocks. Each block contains l modules. We map the color of each module to 0 or 1 and obtain an l-length binary vector for each block[2], and denote A to be one of these vectors. We define an l-length decimal serial number vector W, each value of which represents the weight of each position in A. In this vector W, values are integers from 1 to $2^k - 1$. The remaining $l - (2^k - 1)$ values can be any integer from 1 to $2^k - 1$. Binary representation of W is denoted by W'. With the following three steps, vector A turns into A', which is the encoded vector, with only one bit changed. We denote the information to be embedded in block A as s_1, s_2, ..., s_k.

Step 1: Calculate hiding functions as in Equ. 1, where $A(x)$ denotes the value in position x of vector A, $W'_j(x)$ denotes the

[2]Mapping from color to bit is explained in next subsection. Note that although we deal with colored images, we adopt an approach that embeds data into binary vectors, assuming that each encoding module encodes two states (0 or 1) by changing the module's colors. Fewer encoded states guarantee that these colors can be robustly recognized.

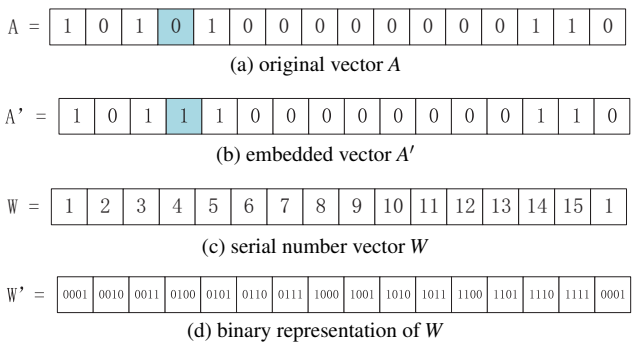


Figure 6: An illustration of our adopted data embedding technique.

j-th bit in binary number $W'(x)$, j ranges from 1 to k.

$$H_j = \sum_{x=1}^{l} A(x) \bullet W'_j(x),\ j = 1,2,...,k. \tag{1}$$

Step 2: Calculate original hiding number h_j.

$$h_j = H_j \bmod 2. \tag{2}$$

Step 3: Find binary modification position R and modification position MP. $\oplus$ stands for xor operator. Changing the bit in position MP of vector A embeds k bits of data in A.

$$\begin{aligned} R_j &= h_j \oplus s_j, \\ MP &= \sum_{j=1}^{l} 2^{j-1} \times R_j. \end{aligned} \tag{3}$$

We take a vector A sized 16 as an example, shown in Fig. 6a, and embed 4 bits 0100 into A. Serial number vector W and its binary representation are presented in Fig. 6c and Fig. 6d. According to Equ. 1, $H_1 = 4$, $H_2 = 3$, $H_3 = 3$, $H_4 = 2$. With Equ. 2 and Equ. 3, we obtain $R_1 = 0$, $R_2 = 0$, $R_3 = 1$, $R_4 = 0$, $MP = 4$. Therefore, we change $A(4)$ from 0 to 1, as in Fig. 6b. Then we map the bit change to a change in color and obtain the embedded dot matrix.

To extract data from an embedded dot matrix, the scheme is similar to data embedding part. We divide the ensemble of encoding modules into l-length vectors. Serial number vector W remains the same as in encoding part. Data extraction process follows two steps: (1) Calculate hiding functions with Equ. 1 and obtain H_j, $j = 1,2,...,k$. (2) Calculate h_j with Equ. 2. The bits embedded in this block are h_j.

Encoding Color to Bit

ARTCODE is composed of colored dots, and in the ideal case, all n colors used for colored dot matrix generation can be employed as encoding colors. However, decoding accuracy and visual effect preservation are the two constraints in encoding color selection from n colors. To guarantee that embedded data can be correctly extracted, encoding colors should be "sufficiently distant" from each other to be recognized by scanning devices. The requirements on visual effect preservation suggest that selected encoding colors take large proportion in embedding positions, so that colored dot matrix after embedment looks more "similar" to the one before embedment.

Figure 7: ARTCODE structure.

Considering the above two criteria, we design a scheme for encoding color selection, consisting of following steps:

Step 1: In color palette, calculate all n colors' proportion they occupy in L encoding modules.

Step 2: Find all feasible color sets. The Euclidean distance in RGB space of any two colors in a feasible color set is larger than a threshold r.

Step 3: Calculate the sum of colors' proportion in each feasible color set. Compare them and select the color set with the largest proportion.

Step 4: If, by any chance, the algorithm fails to find a feasible color set, the algorithm decreases the threshold until feasible color sets are found. Color set proportion is calculated again and the set with largest proportion is selected. Then, we adjust colors in this set to fit the demand of original threshold r.

ARTCODE maps each color in feasible color set to 0 or 1. There is no specific constraint in determining whether a color should take 0 or 1. It is sufficient to make the number of encoding modules of 0 and that of 1 be approximately equal.

RELIABLE ARTCODE DETECTION AND DECODING

In this section we present our detection and decoding scheme. Detection part consists of pre-processing and localization. Decoding scheme recognizes colors in localized positions, maps it to a vector according to shuffling table and extracts data.

ARTCODE Detection

Pre-processing Pre-processing includes two steps. Firstly we implement a local thresholding binarization algorithm. The algorithm divides captured image to sub-blocks and determines a threshold of binarization for each block. Then, we perform an image erosion-dilation approach to eliminate moiré pattern. Moiré pattern is noisy straight lines in image outputs of binarization, caused by camera capturing screens.

Module Localization As huge finder patterns greatly affect the visual effects of the barcode as in [12], we determine not to insert any locators in the code area and place an alignment pattern on the four borders of the barcode for module localization. To enhance the color recognition accuracy we place all encoding colors on the second deck of alignment pattern as reference colors. In this way we obtain a double deck alignment

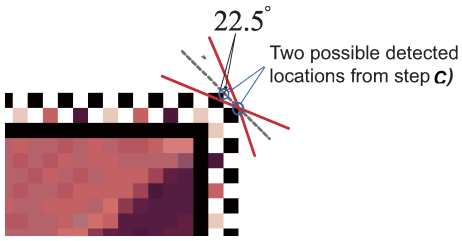


Figure 8: We count black pixels from two directions (red lines) starting from detected module to determine the relative positions of detected black module and white corner.

(shown in Fig. 7). On each of the four borders, double deck alignment consists of two lines of alignment pattern. The outer line of double deck alignment is an alternating permutation of black and white modules and the inner line is an alternating permutation of all encoding colors.

The alignment pattern localization step takes pre-processed images as input and locates firstly four corners of the barcode. Corner detection process consists of three steps: ***a)*** Detection process uses a small rectangle located at the center of the frame, enlarges the rectangle until it covers the whole barcode. ***b)*** On four corners of the rectangle, we use straight lines which are 45° to the corner's adjacent borders and shift them towards center of the rectangle, until these lines reach black pixels (corners of the barcode). ***c)*** We move $\frac{\sqrt{2}}{2}$ of estimated code size to reach the center of corner modules. The bottom-left and top-right corners are white, so we have to scan surrounding black modules to determine in which direction we find the white corner center, shown in Fig. 8.

Once we obtain the four corners of the alignment pattern, we first estimate the module size. Then we start from one corner and find all the black module centers in the alignment pattern. If, unluckily, a module fails to be found, we will estimate its location with its adjacent modules.

Based on the alignment pattern, we can now locate the modules inside the code. Considering the existence of camera distortions, including pincushion distortion, barrel distortion, linear distortion, *etc.*, we utilize a localization algorithm inspired by Anran Wang *et al.* in [29]. This algorithm calculates the distortion ratio of the four borders and combines them linearly in order to restrain the effects of distortions. The algorithm utilized to calculate the coordinate y (x is analogous) can be expressed as:

$$
\begin{aligned}
left &= \frac{align_{left}[j].y - corner_{left-top}.y}{corner_{left-bot}.y - corner_{left-top}.y} \\
right &= \frac{align_{left}[j].y - corner_{right-top}.y}{corner_{right-bot}.y - corner_{right-top}.y} \\
dis_x &= \frac{align_{top}[i].x - corner_{left-top}.x}{corner_{right-top}.x - corner_{left-top}.x} \\
dis_y &= left \times (1 - dis_x) + right \times dis_x \\
y &= align_{top}[i].y \times (1 - dis_y) + align_{bot}[i] \times dis_y
\end{aligned}
\tag{4}
$$

In Equ. 4, $left$ and $right$ denote respectively the distortions of the left and right alignment patterns.

ARTCODE Decoding

Color recognition Once we localize the modules inside the barcode and the second layer of double deck alignment pattern, we calculate the average RGB values for each color in the color palette located on double deck alignment pattern. Color recognition consists assigns nearest RGB values in encoding color set obtained by double deck alignment pattern. Then the raw data is sent to the decoder to obtain the encoded information.

Comprehensive Error Correction One important problem caused by introduction of colors is that color recognition rate decreases compared with black and white modules, although we only select a small set of colors to encode data. In order to guarantee the transmission reliability, we propose an error correction scheme for ARTCODE, dividing data into Segments. Transmission reliability is a crucial issue in ARTCODE or traditional barcode design [30].

When designing the data correction scheme, we have to consider two aspects. ***a)*** As we utilize adaptively selected encoding colors instead of two colors in traditional barcodes, the recognition success rate of each module decreases. ***b)*** As we employ a shuffling algorithm to locate embeddable modules, these embeddable modules spread randomly in ARTCODE. In this way errors occur randomly in the decoded binary matrix.

We design an error correction scheme for ARTCODE called Comprehensive Decoding, considering the fact that one single frame captured by the camera may fail to extract complete data. We apply both Reed Solomon (RS) error correction scheme and Cyclic Redundancy Code (CRC) error detection scheme. Our proposed approach is to append error detection and correction redundancy to adjacent several bits. The proportion of redundancy is adaptive to overall detection accuracy. We call these adjacent data bits with error detection and correction bits a Segment. Each Segment has specific coordinates in the dot matrix. A Segment is firstly corrected by RS error correction algorithm. Afterwards if CRC error detection algorithm detects an error, processor will demand the camera to send in another frame for detection and decoding.

PUTTING THEM TOGETHER

We now put all the components together. Fig. 3 shows the ARTCODE architecture. The whole system consists of two major parts on the sender and the receiver sides.

At the sender side, ***a)*** from an image we employ K-means clustering to select a number of colors that can best represent the original image to form a color palette, and perform a revised dithering to generate colored dot matrix form of the image. ***b)*** We apply a shuffling algorithm to generate a shuffling table, with which the algorithm finds embeddable modules in the colored dot matrix. ***c)*** We rank all colors in the color palette according to their occupied proportion in colored dot matrix and select colors with large proportion and sufficient RGB space distance as encoding colors. ***d)*** We assign 0 and 1 to each encoding color, and adopt a data hiding scheme which

embeds multiple bits into a fixed-sized block with only one bit changed. ***e)*** We integrate RS error correction algorithm and CRC error detection algorithm to guarantee a reliable data transmission.

In the process of scanning, the camera delivers several frames to the processor. ***a)*** The first step is pre-processing and localization. Binarization and erosion are performed as pre-processing step in order to eliminate the moiré patterns. Then we localize the double deck alignment and calculate the locations of each module inside the code by integrating a distortion-aware localization algorithm proposed in [29]. Afterwards, we extract the colors in the reference palette located at the second layer of double deck alignment pattern. ***b)*** Detection algorithm refer to the shuffling table to localize embeddable modules and compare their RGB values with the colors given on the reference palette, and assigns 1 or 0 to each module. Note that to keep the integrity and correctness of data we design a comprehensive decoding scheme.

We implement ARTCODE with about 1300 and 2500 lines of code for the sender and receiver parts. Sender takes an original image and a short data bitstream as input, generates the ARTCODE-ed image and displays it on the screen. Receiver takes captured frames as input, detects the existence of alignment patterns, and retrieves data in one or multiple possible corrupted frames. We implement the sender for the PC and Android phones and the receiver only for Android phones.

EVALUATION

We evaluate ARTCODE from two aspects: image visual quality and information-carrying capability. We first present an overall performance for all the images under the default (identical) setting. We then pick a subset of images for the image visual quality assessment through a subjective user survey and the evaluation of data throughput under various parameter and environment settings.

Experimental setting We use an LG D2792P screen (with a Dell PC) or a Samsung Galaxy S5 smartphone as the sender. The screens are 24 inches and 5.1 inches, respectively; The resolutions both are 1920×1080 pixels. We test with two android phone models as the receiver: Galaxy S5 and Nexus 4 E960. Their maximal resolutions of the primary (back) cameras are 16 MP and 8 MP, which are 5312×2988 pixels and 3264×2448 pixels. The results are similar for both receiver phones and thus merged in the following experiments.

Overall Performance

We download 197 images from Google Image to test with the overall performance under any image. These include images randomly selected from research results under keywords *Logo*, *Gray-scale*, *Portrait*, *Landscape*, *Animal*, and also classical images like Lenna. There are 25 gray-scale images, 22 logos and 150 colored images. Considering contrast is a key factor in color selection, we further divide the colored images into 48 low contrast, 64 intermediate contrast and 38 high contrast ones. Such classification are based on the variance of their gray-scale histograms. Larger value means stronger contrast. The default setting is to embed 50 bytes (m=50) using 16 clustered colors ($k = 16$) with embedding block size $l = 15$.

We first examine ARTCODE's visual quality through a user study with 50 participants (college students). 47 participants frequently use QR codes in their daily life. To assess ARTCODE's image preserving quality, each participant gives a subjective distortion score δ given the original and coded images. The score scales from 0 to 10, where 0 means that the distortion is not perceptible, 5 means that the embedment is noticeable but visually acceptable and 10 means that the imprints of data embedment heavily corrupts the original image quality. For each participant, we randomly select 40 images out of 197 images (including six mandatory ones in Fig. 9) and test with 14 codes under various settings (only the default setting for the overall evaluation).

Six samples are given in Fig. 9. They are representative images from 5 categories: a gray-scale one(Gray), two logos (Logo1, Logo2), a low contrast image (Elephant), a medium-contrast image (Butterfly), and a high contrast image (Lenna). Clearly, ARTCODE preserves image quality. Fig. 10a shows the results for the whole image set. All are below 4, which is acceptable. In all five categories, grayscale and logo images have relatively smaller distortion. This is likely because their images are relatively simpler and quality is less affected under changes.

Fig. 10b further shows its information-carrying accuracy. We use module accuracy as the accuracy of all bit vector encoded by colors, regardless of data embedding scheme and error correction scheme. We use the PC screen as the sender and each code is $8.4 \times 8.4 cm^2$. The camera-screen distance is 18 *cm*. For each image, we have 5 runs and calculate the average accuracy. The statistics is also shown in Table 10c. The median of accuracy is larger than 96% in all the cases and the worst case in 197 images is 92.2%.

In a nutshell, ARTCODE is effective to a variety of images. We believe that it can be applicable to any image to achieve both goals of image preservation and information carrying.

Assessment on Image Visual Quality

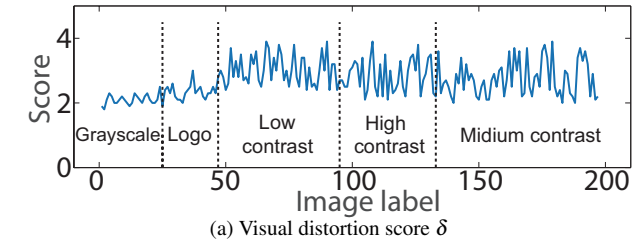
We now examine ARTCODE's effects on visual quality with regards to various encoding parameters: number of clustered colors, bytes embedded and block size. If not specifically stated, the default values are used.

Number of clustered colors k We vary the number of clustered color in a range of 4, 8, 16 and 32. Fig. 12a shows the average distortion scores over all image sources and six representative samples. The distortion decrease as k increases. It is obvious that the more colors selected via K-means clustering, the smaller visual damage ARTCODE imposes on original image. The example of Lenna can be found in Fig. 11. Clearly, the color number larger than 16 is a good choice ($\delta < 4$) with acceptable visual quality.

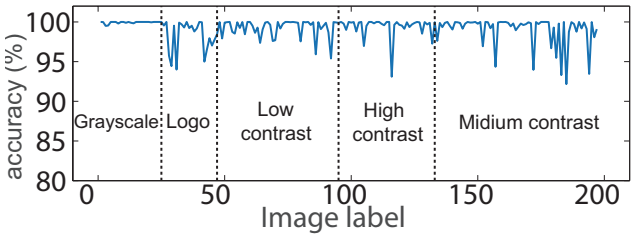
Number of embedded bytes m We vary it from 10B, 30B, 50B, 100B to 300B and obtain the distortion scores in Fig. 12b . The quality decreases as m grows. When m is large than 100B, the damage on images are not tolerable. It indicates that ARTCODE is only suitable to carry small-volume data (like a URL message).

(a) Gray ($\delta = 2.1$) (b) Logo1 ($\delta = 2.3$) (c) Logo2 ($\delta = 2.2$) (d) Elephant ($\delta = 2.8$) (e) Butterfly ($\delta = 2.4$) (f) Lenna ($\delta = 2.6$)

Figure 9: Six ARTCODE samples with their original images under the default setting.



(a) Visual distortion score δ



(b) Decoding accurate rate

	Gray	Logo	Low	High	Medium
δ	2.15	2.5	3.15	3	2.95
(min-max)	(1.8–2.5)	(2–3)	(2.4–3.9)	(2.1–3.9)	(2–3.9)
accuracy (%)	99.8	97	97.7	96.6	96.1
(min-max)	(99.5-100)	(94 – 100)	(95.4-100)	(93.1–100)	(92.2–100)

(c) Statistics

Figure 10: Overall performance of ARTCODE.

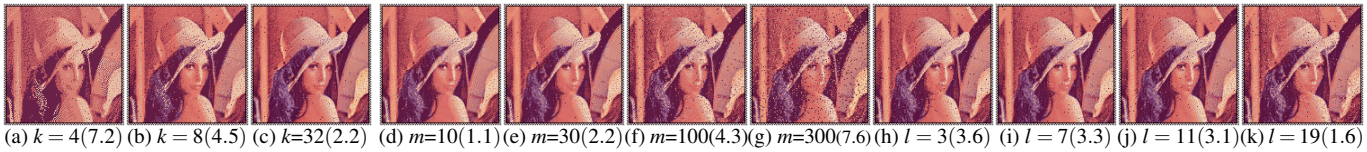
Size of embedding block l We test with five different values: 3, 7, 11, 15 and 19 and show the distortion scores in Fig. 12c. As we mentioned above, l denotes the size of embedding block which can embed $\lfloor \log_2(l+1) \rfloor$ *bits* by changing 1 bit. Assuming that we embed data sized m *bits*, the error correction bits we need $\propto m$. The number of modules which should be altered $\propto \frac{m}{\lfloor \log_2(l+1) \rfloor}$. In this way, visual distortion basically decreases as l increases, but it changes only when l equals $2^i - 1$, i is a positive integer and larger than 1, for we round down $\log_2(l+1)$. However, if the embedding block size is too large, it will be difficult to decode all the modules in the block. In the following data communication experiments, we find that both high decoding accuracy and low visual distortion can be achieved when l is between 7 and 15.

From all the plots in Fig. 12, we observe that the grayscale images and logos have better visual perception. This is consistent with our overall evaluation. Such user study again validates ARTCODE can preserve image under appropriate parameters. Taking both visual effects and data capacity into account, we set the number of clustered colors $\in [16, 32]$, the block size $\in [7, 15]$ and the number of bytes no larger than 100B.

Information-Carrying Capability

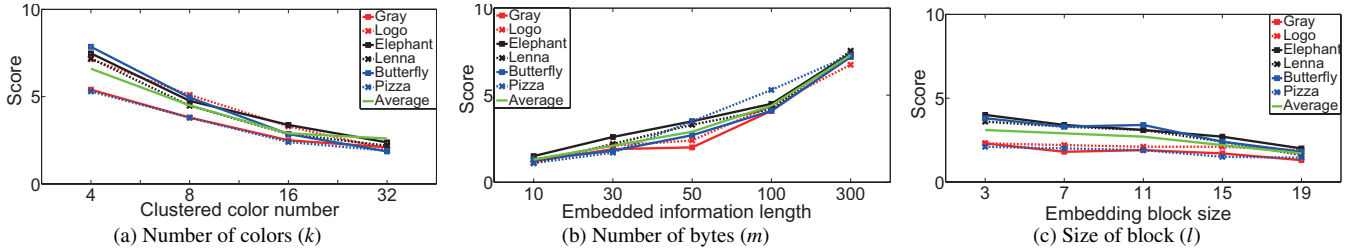
In this part we analyze the accuracy performances of ARTCODE under different scanning conditions: ambient light illuminance, screen brightness, camera resolution and scanning distance. If not specifically stated, the experiments are done in indoor conditions, with ambient light intensity equaling to 75 *lux*, screen brightness equaling to 80 *lux*. For the distance-code size test, we utilize both an LED screen and a Nexus 4 smartphone screen to display the code, while for other tests we use the smartphone screen. The distance between code and camera is set to be 18*cm* and the code is displayed with a size of 8.4×8.4 cm^2. The default receiver is Samsung Galaxy S5 with its camera resolution to be 3264×1836 *pixels*. We use a digital Lux meter (SIGMA AS803) to measure light intensity. We use the same five images as mentioned in previous section and generate five ARTCODEs of each of them because the clustered colors are different every time we use K-means. All data in this part is the average value of these ARTCODEs. We compare module accuracy, single frame accuracy and multiple frame accuracy. Single frame accuracy represents successfully decoded data rate in a single frame and multi frame accuracy represents data rate with Comprehensive decoding. For each ARTCODE we take 20 adjacent frames and the module accuracy and single frame accuracy are taken from an averaging of all these frames under each condition and the multiple frame accuracy is measured by using up to 5 frames to decode.

Illuminance and Display Brightness As light being a transmission media in code detection process, the light intensity is an essential factor when evaluating barcode performances. We conclude ambient light intensity and sender screen brightness as the main indices in measuring light intensity. We choose five different ambient light conditions, where the intensities are respectively 10 *lux*, 30 *lux*, 70 *lux*, 150 *lux*, and 300 *lux* with screen brightness to be 80 *lux*. The screen



(a) $k = 4(7.2)$ (b) $k = 8(4.5)$ (c) k=32(2.2) (d) m=10(1.1) (e) m=30(2.2) (f) m=100(4.3) (g) m=300(7.6) (h) $l = 3(3.6)$ (i) $l = 7(3.3)$ (j) $l = 11(3.1)$ (k) $l = 19(1.6)$

Figure 11: The ARTCODE samples under various settings (default: $k = 16, m = 50, l = 15$ in Fig. 9f) with distortion scores.



(a) Number of colors (k) (b) Number of bytes (m) (c) Size of block (l)

Figure 12: Visual quality (distortion scores) under different setings.

brightness intensities are respectively 10 *lux*, 42 *lux*, 80 *lux*, 127 *lux* and 168 *lux* with ambient light intensity to be 75 *lux*. The accuracy analysis is shown in Fig. 13a and Fig. 13b. It is clear from the figures that the accuracy increases tremendously both under screen brightness increase and ambient light change. Module accuracy, single frame accuracy and multi-frame accuracy increase from around 75.2% to 99%, 20.1% to 95% and 30% to 99% respectively. After reaching a threshold of minimum light condition for decoding, the increase tendency goes smooth and steady, and the three accuracies go steady at around 99%, 95% ,99%.

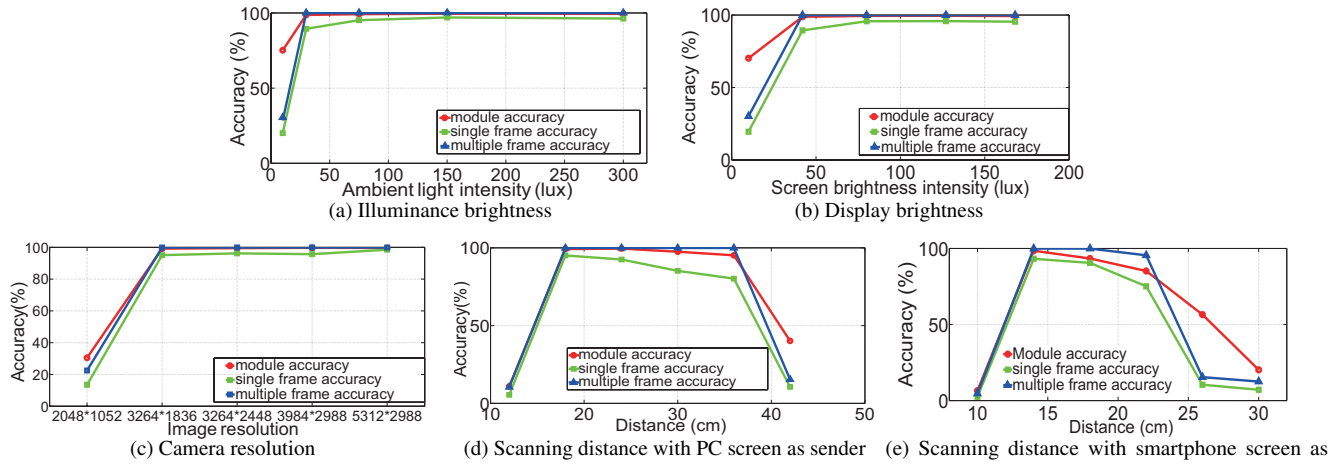
Camera Resolution We analyze the influence of camera resolution and show the results in Fig. 13c. Image resolution is an important factor, for it is closely related to the image quality and therefore affects the decode rate. We choose five different image resolutions (2048×1152 *pixels*, 3264×1836 *pixels*, 3264×2448 *pixels*, 3984×2988 *pixels*, 5312×2988 *pixels*) and the experiment is carried out with same lighting conditions (75 *lux* of ambient light and 80 *lux* of screen light). The result shows that accuracy increases with high resolution due to the increasing pixels number in each module. Module accuracy, single frame accuracy and multi-frame accuracy go steady at around 99%, 95% ,99%. The image resolution under 3264×1836 sharply brings down the three accuracies to 30%,13.5% and 22.5%. As more smartphones are equipped with high quality camera, the off-the-shell smartphones are sufficient to meet ARTCODE's requirements.

Scanning Distance In Fig. 13d and Fig. 13e we show the accuracy of ARTCODE with scanning distance varying. We carry out this experiment with two code sizes — 5.2 *cm* and 8.4 *cm*. The 5.2 *cm* sized ARTCODE is displayed by Galaxy S5 screen and the 8.4 *cm* sized ARTCODE is displayed on LG D2792P screen. The accuracy result is symmetric and unimodal. At 8.4 *cm* ARTCODE size condition, accuracies start at 10.9%, 5.5% and 10.5%, go steady at around 99%, 93% and 99% between 18 *cm* and 36 *cm* and drop to 40.2%, 10.6% and 15.4% at 42 *cm*. At 5.2 *cm* ARTCODE size condition, accuracies start at 6.5%, 2.5% and 4.6%, go steady at around 90%, 85% and 99% between 14 *cm* and 22 *cm* and drop to 56.5%, 10.4% and 15.4% at 26 *cm*. Results show that for the 5.2 *cm* sized ARTCODE and the 8.4 *cm* sized ARTCODE, acceptable distances are respectively from 14 *cm* to 22 *cm*, and from 18 *cm* to 36 *cm*. As recognition and localization algorithms need to avoid close scanning and noise introduced by long distance scanning is harmful to decoding, an appropriate scanning distance is essential in decoding process and that's why there is usually a scanning zone centered in the screen in barcode scanning applications.

RELATED WORK

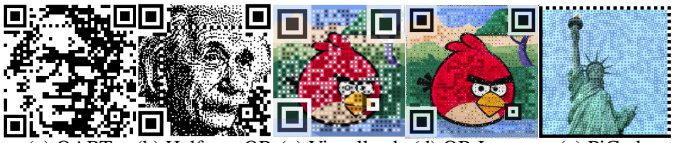
We compare ARTCODE with the literature in three aspects.

Artistic barcode design Several recent studies seek to preserve image or artistic design in QR codes [1–3, 12–15, 19, 20, 22, 37]. Several coded samples are given in Fig. 14. To make QR code readable or attractive to human eye, the most intuitive way is to incorporate a logo, a thumbnail or other information in the QR code [2, 22] (Example: Fig. 1b). However, this approach often suffers with a small area for readable content. Afterwards, researchers have applied different image processing techniques to ameliorate the code design, such as applying obfuscating masks in QART [1], halftone-based algorithms (halftoning is to replace some modules with preserving particular spectral distribution properties) in Halftone QR [12] and image barcode (IBC) [13], image blending [37], information-theoretic approach [14], optimization under given image distortion [15], pre-defined tilting patterns (two patterns for 0 or 1 bit) in PiCode [19], patterns of luminance modulation [20]. ARTCODE is inspired by these prior work but differ from them in several aspects. First, they focus on image rendering and overlooks the code-to-camera data communication. They do not take information-carrying into account, at least, not explicitly evaluate the impact of their designs; In contrast, ARTCODE is designed to serve both goals of image preservation and information carrying. Second, by leveraging novel techniques on pointillism, shuffling and adaptive color selection, ARTCODE even preserves better image quality than these existing work (see Figures 9 and 14). Moreover, ARTCODE makes great technical efforts in exploring the color choice whereas they either fail to leverage it [1,12,13,19,20] or does not exploit the full data-carrying potential [15]. Last but not least, ARTCODE incorporates techniques (pre-processing,



(a) Illuminance brightness (b) Display brightness (c) Camera resolution (d) Scanning distance with PC screen as sender (e) Scanning distance with smartphone screen as sender

Figure 13: These images illustrate the performances of ARTCODE under different conditions.



(a) QART (b) Halftone QR (c) Visuallead (d) QR Image (e) PiCode

Figure 14: Other artistic code samples in [1, 3, 12, 15, 19].

module localization, color recognition *etc.*), for reliable code-to-camera communication.

Watermarking and steganography They have been actively studied to embed (often hide) information (*e.g.*, copyright) in an image (*e.g.*, [9,11,23,26–28,32–34,36]). They seek to make the altered image as close to the original one as possible. [11, 26, 36] gave nice surveys on versatile techniques such as embedding data in the least significant bit [9, 33], manipulating the histogram [23], pattern-reserved injection [34], data concealing using a limited number of bit changes (2 bits) [28], and embedding in multi-level [32]. Our work differs from them. ARTCODE seeks image-preserved data communication and must address degradations over the optical channel while they do not. Indeed, a commercial application developed by Digimarc integrates watermarking technologies to encode data into an image, which can be further scanned by smartphone cameras [6]. Its capacity is small (6 *bytes*), thus the code encodes only a short ID and refers to the company's database via cellular network or Wifi to obtain full length of data. Our work differs in that ARTCODE fulfills complete functionality of barcodes and do not need extra information to obtain the data transmitted.

Unobtrusive and obtrusive screen-to-camera communication There are several non-obtrusive screen-to-camera communications which hide data bits over the screen-to-camera links, including InFrame++ [29], HiLight [21], Visual MIMO [35] and IVC [8]. In particular, InFrame++ hides data into video contents through complementary frames [29]; HiLight conveys data bits through the pixel transparency change within a time window [21]; VRCode [31] and IVC [8] hide data upon two consecutive frames with one original one and one altered one (using a pyramid decomposition in [31] and brightness change [8]). Our difference is that they hide data into a coded streaming with varying frames whereas ARTCODE targets at image-preserved data communication which can work with any static image on any display, including poster/printout *etc.*. Obtrusive (normal) screen-to-camera communications have been extensively studied in recent years, such as Pixnet [25], COBRA [16], LightSync [17], Strata [18], RDcode [30]. The proposed techniques mainly address challenges in screen-to-camera optical channels such as rolling shutter effect [17], color effects [16], dynamic capture quality [18], and error handling [30]. These techniques complement our design in ARTCODE.

CONCLUSION

In this paper we put forward ARTCODE, a new two-in-one code design, that embeds data entirely in an image and preserves artistic image with provisioning barcode communication functionality. We leverage an error diffusion dithering algorithm to generate the dot matrix, integrated with an adaptive color palette. In order to disperse visual distortion, we perform shuffling onto the dot matrix and thus spreads encoding modules and encode bits into colors conforming to the image. To tackle interference from the background image, we design a reliable barcode decoding scheme specifically for ARTCODE. Envisioning the increasing need of interactions between devices and their surroundings, we believe that such intuitive codes will have larger potential in practice. It delivers contents to both humans and their devices and offers more pleasant and informative user experience.

ACKNOWLEDGEMENT

We would like to thank the reviewers and the shepherd for their suggestions. This work was supported by NSF China (No.61532012, 61325012, 61271219, 61521062, 61428205) and Microsoft Research Asia Collaborative Research Program 2016 (No.FY16-RES-THEME-025).

REFERENCES

1. 12 Apr. 2012. QArt Codes. (12 Apr. 2012). http://research.swtch.com/qart/.
2. 2010. Form Meets Function: Extreme Makeover QR Code Edition. (2010). http://blog.360i.com/mobile-marketing/creative-qr-codes.
3. 2012. Visualead. (2012). http://www.visualead.com/.
4. 2015. QR Code Wikipedia. (2015). http://en.wikipedia.org/wiki/QR_code.
5. 2015. Scan.me. (2015). https://scan.me/.
6. 2016. Digimarc. (2016). https://www.digimarc.com.
7. 2016. Pointillism. (2016). https://en.wikipedia.org/wiki/Pointillism.
8. R. Carvalho, C. H. Chu, and L. J. Chen. 2014. IVC: Imperceptible Video Communication. In *Proc. HotMobile*. Demo.
9. C. K. Chan and L. M. Cheng. 2004. Hiding Data in Images by Simple LSB Substitution. *Pattern recognition* 37, 3 (2004), 469–474.
10. C. C. Chang, C. S. Tseng, and C. C. Lin. 2005. Hiding Data in Binary Images. In *Proc. Information Security Practice and Experience*. 338–349.
11. A. Cheddad, J. Condell, K. Curran, and P. McKevitt. 2010. Digital Image Steganography: Survey and Analysis of Current Methods. *Signal Processing* 90, 3 (2010), 727–752.
12. H. K. Chu, C. S. Chang, R. R. Lee, and N. J. Mitra. 2013. Halftone QR Codes. *ACM Trans. Graphics* 32, 6 (2013), 217.
13. N. Damera-Venkata, J. Yen, V. Monga, and B. L. Evans. 2005. Hardcopy Image Barcodes Via Block-Error Diffusion. *IEEE Trans. Image Processing* 14, 12 (2005), 1977–1989.
14. J. Duda, N. J. Gadgil, K. Tahboub, and E. J. Delp. 2014. Generalizations of the Kuznetsov-Tsybakov Problem for Generating Image-like 2D Barcodes. In *IEEE International Conference on Image Processing*. 4221–4225.
15. G. J. Garateguy, G. R. Arce, and D. L. Lau. 2014. QR Images: Optimized Image Embedding in QR Codes. *IEEE Trans. Image Processing* 23, 7 (2014), 2842–2853.
16. T. Hao, R. Zhou, and G. Xing. 2012. Cobra: Color Barcode Streaming for Smartphone Systems. In *Proc. ACM MobiSys*. 85–98.
17. W. Hu, H. Gu, and Q. Pu. 2013. LightSync: Unsynchronized Visual Communication over Screen-Camera Links. In *MobiCom*. 15–26.
18. W. Hu, J. Mao, Z. Huang, Y. Xue, J. She, K. Bian, and G. Shen. 2014. Strata: Layered Coding for Scalable Visual Communication. In *Proc. ACM MobiCom*. 79–90.
19. W. Huang and W. H. Mow. 2013. PiCode: 2D Barcode with Embedded Picture and ViCode: 3D Barcode with Embedded Video. In *Proc. ACM Mobicom*. 139–142.
20. Y. L. Lee and W. H. Tsai. 2014. A New Data Transfer Method via Signal-Rich-Art Code Images Captured by Mobile Devices. *IEEE Trans. Circuits and Systems for Video Technology* 25, 4 (2014), 688–700.
21. T. Li, C. An, X. Xiao, A. T. Campbell, and X. Zhou. 2015. Real-time Screen-camera Communication Behind Any Scene. In *Mobisys*. 197–211.
22. Y. Lin, S. Luo, and B. Chen. 2013. Artistic QR Code Embellishment. *Computer Graphics Forum* 32, 7 (2013), 137–146.
23. Z. Ni, Y. Q. Shi, N. Ansari, and W. Su. 2006. Reversible Data Hiding. *IEEE Trans. Circuits and Systems for Video Technology* 16, 3 (2006), 354–362.
24. K. Ouimet, B. Turley, A. Vance, T. Kohno, and F. Roesner. 2015. Analyzing the Use of Quick Response Codes in the Wild. In *Proc. ACM MobiSys*. 359–374.
25. S. D. Perli, N. Ahmed, and D. Katabi. 2010. Pixnet: Interference-free Wireless Links Using LCD-camera Pairs. In *MobiCom*. 137–148.
26. F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn. 1999. Information Hiding – A Survey. *Proc. IEEE* 87, 7 (July 1999), 1062 –1078.
27. H. W. Tseng, F. R. Wu, and C. P. Hsieh. 2007. Data Hiding for Binary Images Using Weight Mechanism. In *IEEE Intelligent Information Hiding and Multimedia Signal Processing (IIHMSP)*. 307–310.
28. Y. C. Tseng, Y. Y. Chen, and H. K. Pan. 2002. A Secure Data Hiding Scheme for Binary Images. *IEEE Trans. Communications* 50, 8 (2002), 1227–1231.
29. A. Wang, Z. Li, C. Peng, G. Shen, G. Fang, and B. Zeng. 2015. InFrame++: Achieve Simultaneous Screen-Human Viewing and Hidden Screen-Camera Communication. In *MobiSys*. 181–195.
30. A. Wang, S. Ma, C. Hu, J. Huai, C. Peng, and G. Shen. 2014. Enhancing Reliability to Boost the Throughput over Screen-camera Links. In *Proc. ACM MobiCom*. 41–52.
31. G. Woo, A. Lippman, and R. Raskar. 2012. Vrcodes: Unobtrusive and Active Visual Codes for Interaction by Exploiting Rolling Shutter. In *IEEE International Symp. Mixed and Augmented Reality (ISMAR)*. 59–64.
32. M. Wu and B. Liu. 2003. Data Hiding in Image and Video .I. Fundamental Issues and Solutions. *IEEE Trans. Image Processing* 12, 6 (2003), 685–695.
33. C. H. Yang, C. Y. Weng, S. J. Wang, and H. M. Sun. 2008. Adaptive Data Hiding in Edge Areas of Images with Spatial LSB Domain Systems. *IEEE Trans. Information Forensics and Security* 3, 3 (2008), 488–497.

34. H. Yang and A. C. Kot. 2007. Pattern-based Data Hiding for Binary Image Authentication by Connectivity-preserving. *IEEE Trans. Multimedia* 9, 3 (2007), 475–486.

35. W. Yuan, K. Dana, A. Ashok, M. Gruteser, and N. Mandayam. 2012. Dynamic and Invisible Messaging for Visual MIMO. In *IEEE Workshop on WACV*. 345–352.

36. Y. Zhang. 2009. Digital Watermarking Technology: A Review. In *Future Computer and Communication (FCC)*. 250–252.

37. Y. Zhang, S. Deng, Z. Liu, and Y. Wang. 2015. Aesthetic QR Codes Based on Two-Stage Image Blending. *MultiMedia Modeling. Springer International Publishing* (2015), 183–194.