Highlights

Back-compatible Color QR Codes for colorimetric applications

Ismael Benito-Altamirano, David Martínez-Carpena, Olga Casals, Cristian Fàbrega, Andreas Waag, Joan Daniel Prades

- Color correction often relies on color charts that can accomodate limited colors (10s)
- QR Codes can be used to accomodate a large number of color samples (1000s)
- QR Code have compact form factor (a few inches) while chart have a wide factor form
- Our grayscale affinity to black and white during color mapping is fully back-compatible
- Grayscale method leads to much higher readability in adversarial opticial conditions

Back-compatible Color QR Codes for colorimetric applications

Ismael Benito-Altamirano[a,b,*], David Martínez-Carpena[b,d], Olga Casals[a], Cristian Fàbrega[a], Andreas Waag[c], Joan Daniel Prades[a,b]

[a] *MIND/IN2UB, Department of Electronic and Biomedical Engineering, Universitat de Barcelona, Carrer de Martí i Franquès, 1, Barcelona, 08028, Barcelona, Spain*
[b] *ColorSensing SL, Carrer Morales, 21, 1L, Barcelona, 08029, Barcelona, Spain*
[c] *Institute for Semiconductor Technology, Braunschweig University of Technology, Universitätspl. 2, Braunschweig, 38106, Lower Saxony, Germany*
[d] *Department of Mathematics and Computer Science, Universitat de Barcelona, Gran Via de les Corts Catalanes, 585, Barcelona, 08007, Barcelona, Spain*

Abstract

Color correction techniques in digital photography often rely on the use of color correction charts, which require including this relatively large object in the field of view. We propose here to use QR Codes to pack these color charts in a compact form factor, in a fully compatible manner with conventional black and white QR Codes; this is, without losing any of their easy location, sampling and digital data storage features. First, we present an algorithm to build these new colored QR Codes that preserves the original QR Code functionality - much more than other coloring proposals based on the random substitution of black and white pixels by colors - that relies on the ability of the native CRC code to correct and counteract these alterations. Second, we demonstrate that, as a result, these QR Codes can allocate far many more colors than the conventional color correction charts, enabling much more accurate color correction schemes in a more convenient and usable format.

Keywords: barcodes, QR Codes, color correction, color calibration, colorchecker, colorimetry

1. Introduction

The rise of the smartphone technology developed in parallel to the popularization of digital cameras enabled an easier access to photography devices to the people. Nowadays, modern smartphones have onboard digital cameras that can feature good color reproduction for imaging uses. However, when actual colorimetry is needed, the smartphone camera sensor does not suffice, needing auxiliary ad hoc tools to evaluate color and guarantee its reproduction [1]. The traditional approach to solve these problems are machine-readable patterns which embed color reference, e.g. color charts, introduced by C.S. McCamy et. al. in 1976 [2] (see Figure 1.a).

Despite the introduction of color charts was done almost half a century ago, these solutions are widely implemented and used nowadays [3]. However, these real-world applications often lack of an automated, scalable and optimal solution from the pattern recognition and signal acquisition point-of-view, up to the point some research fields prefer the use of fixed capture setup instead of using the advances of modern pattern recognition [4].

These real-world applications comprise a wide range of industries, such as: as health care, food manufacturing and environmental sensing. Regarding health care, dermatology is one of the main fields where color measurement is a strategic problem, from measuring skintones in order to avoid dataset bias [5] to medical image analysis to identify skin lesions [6]. In food manufacturing, color is used as an indicator for quality control and freshness problems [7]. As for environmental sensing [8], color-changing chemical indicators (i.e. colorimetric) are widely used as a sensor for humidity [9] or many other gases [10].

The constraint to fully automate the capture process of these color charts affects the quality of the color correction, as it is safe to say that, in most of these post-capture color correction techniques, increasing the number and quality of the color references offers a systematic path towards better color calibration results

*Correspondence: ismael.benito@ub.edu; Tel.: +34-93-403-48-04

[11, 12].

This strategy, however, comes along with more image area dedicated to accommodate these additional color references and therefore, a compromise must be found. In this direction, Pantone presented in 2020 an improved color chart called Pantone Color Match Card (see Figure 1.b), based on the ArUco codes introduced by S. Garrido-Jurado et al. in 2015 [13] to facilitate the location of a relatively large number of colors. Still, the size of these color charts is too big for certain real applications with size constraints.

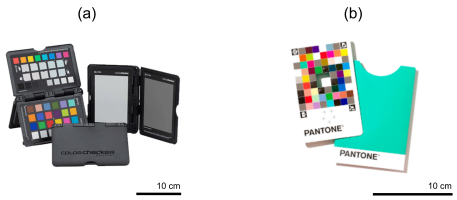

Figure 1: Previous state-of-the-art color correction charts from Pantone and X-Rite. (a) The X-Rite ColorChecker Passport Photo 2® kit. (b) The Pantone Color Match Card®.

QR Codes [14], present themselves as an alternative, already standardized, solution to solve the pattern recognition problem presented by color charts. QR Codes have been used before to encode color information in several fashions, i.e. to increase the color capacity of the QR Code [15], even using cryptography [16], or as an aesthetic feature [17, 18]. We propose to use color inside QR Codes to act as a color chart, thus enhancing color charts with the pattern features of QR Codes.

Recently, we have introduced different machine-readable patterns similar to the Pantone Color Match Card. In 2018, we presented a first implementation that integrated a color changing indicator (sensitive to gases related to bad odor) and a set of color references (to compare with and measure that color indication) [19]. In 2020, we reported a more refined solution allocating hundreds of colors inside a pseudo QR Code pattern [20]. Those patterns lacked the ability to contain digital data, like the former Pantone ones, based only in the ArUco codes [13]. These were, therefore, ad hoc solutions, without the advantages of combining a compact colorimetric readout and calibration pattern with the digital data available in a QR Code.

In fact, linking the colorimetric problem to a set of digital information opens the door to many potential uses related to automation. For example, the digital data could store a unique ID to identify the actual color calibration references used in the image, or other color-measurement properties e.g. by pointing at a data storage location. When used, for example, in smart packaging, this enables the identification of each package individually, gathering much more refined and granular information [21].

In this work, we propose a solution for that problem by placing together digital information and color references without breaking the QR Code standard [14] in a back-compatible Color QR Code implementation for colorimetric applications. This way, we contribute to create a more color-compact machine-readable pattern which preserves digital information, compared to previous state-of-the-art solutions.

2. Theoretical foundations

2.1. The QR Code life-cycle

QR Codes (Quick-Response Codes) are 2D barcodes introduced in 1994 by Denso Wave [14], which aimed at replacing traditional 1D barcodes in the logistic processes of this company. However, the use of QR Codes has escalated in many ways and are now present in manifold industries: from manufacturing to marketing and publicity, becoming a part of the mainstream culture. In all these applications, QR Codes are either printed or displayed and later acquired by a reading device, which normally includes a digital camera or barcode scanner.

The process of encoding and decoding a QR Code could be considered as a form of communication through a visual channel: a certain message is created, then split into message blocks, these blocks are encoded alongside error correction blocks, and finally encoded in a 2D binary array. This 2D binary array is an image that is transmitted through a visual channel (printed, observed under different illuminations and environments, acquired as a digital image, located, resampled, etc.). On the decoder side, the binary data of the 2D binary array is retrieved, the binary stream is decoded, and finally the original message is obtained. Figure 2 shows the block diagram for this process and also shows our back-compability proposal explained in the following subsections.

2.2. The QR Code encoding

When QR Codes are encoded, a huge amount of space is reserved for the error correction blocks (EC), much more than that for data blocks (D). This error correction feature, based on the Reed-Solomon error correction codes [14], is indirectly responsible for the popularity of QR Codes, since it makes them extremely ro-

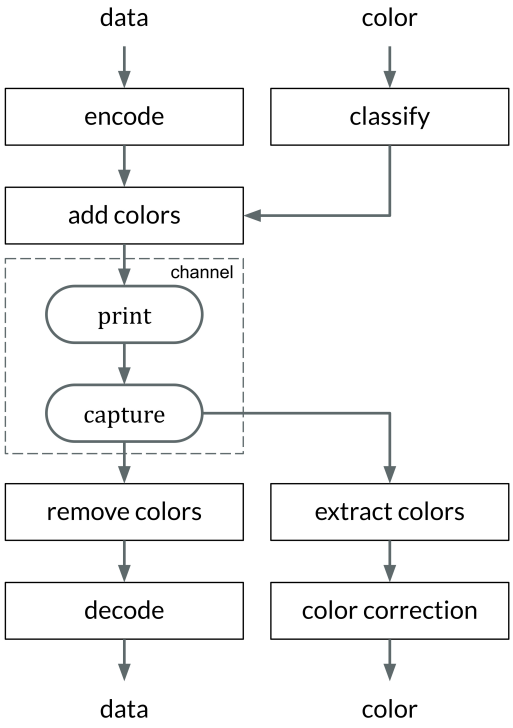

Figure 2: Block diagram for a back-compatible encoding-decoding process of a QR Code which features the embedding of a color layer for colorimetric applications. The process can be seen as a global encoding process (digital encode and color encode), followed by a channel (print and capture) and a global decoding process (extract colors and decode digital information). This process is back-compatible with state of the art scanners which remove colors and achieve the decoding of the data and compatible with new decoders which can benefit from color interrogation. The back-compatibility is achieved by following certain rules in the color encoding process (i.e. use the same threshold when placing the colors than when removing them).

bust while allowing for a large amount of pixel tampering to accommodate aesthetic features like logos. During the generation of a QR Code, the level of error correction can be selected, from high to low capabilities: H (30 %), Q (25%), M (15%) and L (7%). This should be understood as the maximum number of error bits that a certain barcode can support (maximum Bit Error Ratio, see below).

To compensate for the loss of data space, especially at higher error correction levels, the QR Code standard provides a way to increase the total data space by increasing the amount of pixels in the QR Code. This is the so-called version of a QR Code, that ranges from v1 (21×21 pixels) to v40 (177×177 pixels), by adding 4 rows and 4 columns each time the version increases [14].

2.3. *Color as a source of noise*

Error correction blocks can handle distortions in the QR Code image that can be regarded as sources of noise over the visual data communication channel (printing problems, light conditions, capture filters, etc.). In the same way, deliberate image modifications, like the insertion of a logo, or the inclusion of a color reference chart like we do here, can be regarded as additional noise. As such, the noise related to this tampering of pixels can be characterized with well-known metrics like the signal-to-noise ratio (SNR) and the bit error ratio (BER).

Let's exemplify this with a QR Code that encodes a website URL (see Figure 3.a.). First, this barcode is generated and resized (Figure 3.b.) to fit a logo inside (Figure 3.c.). The scanning process (Figure 2) follows a sequence of sampling –to detect the where QR Code is– (Figure 3.d.), desaturation –turning the color image into a grayscale image– (Figure 3.e.) and colorization –turning a grayscale image into a black and white image– (Figure 3.f.). The original binary barcode (Figure 3.a.) and the captured one (Figure 3.f.) will be clearly different, and here is where the error correction plays a key role to retrieve the correct encoded message -the URL in this example-.

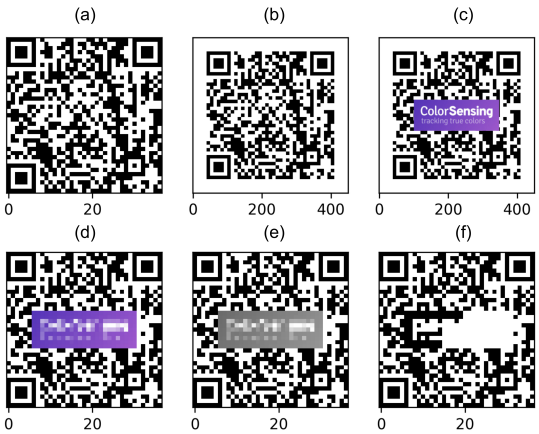

Figure 3: A QR Code is overlaid with a logo, which accumulates error due to the presence of the logo. (a) The QR Code is encoded. (b) The code is resized to accommodate the logo. (c) The logo is placed on top of the QR Code. (d) The code is "captured" and down-sampled again (A_{rgb}). (e) The sampled image is passed to grayscale (A_{gray}). (f) The image is binarized (A_{bin}), the apparent QR Code differs from the original QR Code (a).

SNR is computed following Equation 1 and Equation 2:

$$\mathrm{SNR} = \frac{\sum_0^n \sum_0^m (A_{gray}(i,j))^2}{\sum_0^n \sum_0^m (A_{gray}(i,j) - C_{gray}(i,j))^2} \quad (1)$$

$$\mathrm{SNR_{dB}} = 10 \log_{10}(\mathrm{SNR}). \quad (2)$$

where $A_{gray} \in [0,1]^{n \times m}$ are the pixels of an original QR Code image, which act as a '*signal image*' (see Figure 3.a), $C_{gray} \in [0,1]^{n \times m}$ are the pixels of the QR Code with the logo in a normalized grayscale (see Figure 3.e), the difference between both images acts as the '*noise image*', and the ratio between their variances is the SNR.

BER is computed following Equation 3:

$$\mathrm{BER} = \frac{\sum_0^n \sum_0^m |A_{bin}(i,j) - C_{bin}(i,j)|}{N} \quad (3)$$

where $A_{bin} \in \{0,1\}^{n \times m}$ is the binarized version of $A_{gray} \in [0,1]^{n \times m}$ pixels, $C_{bin} \in \{0,1\}^{n \times m}$ (see Figure 3.f) is the binarized version of $C_{gray} \in [0,1]^{n \times m}$ and $N = n \cdot m$ are the total pixels in the image (see online Supplementary Materials sections A and B).

As an example of these calculations, Table 1 shows the results for the computation of the SNR and BER figures for Figure 3 images. As we can see, adding a logo to the pattern represents a noise source that reduces the SNR to 10.53 dB, further noise sources (printing, capture, etc.) will add more noise thus reducing the SNR more. BER metric shows us the impact of the logo when recovering the digital bits. As we have mentioned before this quantity is directly related to the error correction level needed to encode the QR Code. In this example, with a BER of 8.54%, the poorest error correction level (L, 7%) would not suffice to ensure safe readout of the barcode.

Measure	Acronym	Value
Signal-to-Noise ratio	SNR	10.53 dB
Bit error ratio	BER	8.54 %

Table 1: The values for the SNR and BER are computed for the QR Code with a logo from Figure 3. The SNR is computed using grayscale images. The BER is computed using binary images (see Figure 3).

2.4. Back-compatibility proposal

We want to achieve back-compatibility with the QR Code standard. This means that we must still be able to recover the encoded data message from the colored QR Code using a standard readout process (see Figure 2).

To make it possible we must place these colors inside the barcode avoiding the protected key areas that ensure its readability. In the rest of the available positions, the substitution of black and white pixels with colors can be regarded as a source of noise added to the digital data pattern. We propose in this section a method to reduce the total amount of noise and miss-classifications introduced in the QR Code when encoding colors, that is based on the affinity of those colors to black and white (i.e. to which color it resembles the most). To that end, we classify the colors of the palette to be embedded in two groups: *pseudo-black* and *pseudo-white* colors.

To create these two groups, our proposed implementation uses a procedure that resembles the QR Code readout process. First, a *normalization* of the colors (from integer to float data) is done, then the standard *desaturation* (from color to grayscale data) is applied, later a *binarization* (from grayscale to binary data). Finally, we use these values to classify the colors and assign them to a certain areas of the QR Code, in a process we call *colorization* (see online Supplementary Materials section B). These steps are represented by arbitrary functions, and can be implemented in different ways depending on the application requirements. Our implementations of these steps are as follows:

Normalization, $f_{normalize}$ will be a function that transforms a 24-bit color image (RGB) to a normalized color representation. We used a linear rescaling factor for this:

$$G_{rgb}(k,c) = f_{normalize}(G'_{rgb}) = \frac{1}{255} G'_{rgb}(k,c) \quad (4)$$

where $G'_{rgb} \in [0,255]^{l \times 3}$ is a list of colors with a 24-bit RGB color depth and $G_{rgb} \in [0,1]^{l \times 3}$ is the normalized RGB version of these colors.

Desaturation, $f_{desaturate}$ will be a function that transforms the color channels (RGB) to a monochromatic grayscale channel. We used an arithmetic average of the RGB pixel channels:

$$G_{gray}(k) = f_{desaturate}(G_{rgb}) = \frac{1}{3} \sum_{c=0}^{3} G_{rgb}(k,c) \quad (5)$$

where $G_{rgb} \in [0,1]^{l \times 3}$ is the normalized RGB color palette and $G_{gray} \in [0,1]^l$ is the grayscale version of this color palette.

Binarization, $f_{binarize}$ will be a function that converts the monochromatic grayscale channel (L) to a binary

channel (B). We used a simple threshold function with a thresholding value of 0.5:

$$G_{bin}(k) = f_{binarize}(G_{gray}) = \begin{cases} 0 & G_{gray}(k) \leq 0.5 \\ 1 & G_{gray}(k) > 0.5 \end{cases} \quad (6)$$

where $G_{gray} \in [0,1]^l$ is the grayscale version of the color palette and $G_{bin} \in \{0,1\}^l$ is its binary version, which describes the affinity to black (0) and white (1) colors.

Colorization, f_{color} will be a function that will render a RGB image from the QR Code binary image, the palette colors and a certain mask to select which regions of the QR Code we want to preserve. It has the following general form:

$$C_{rgb} = f_{color}(A_{bin}, G_{rgb}, Z) \quad (7)$$

where $A_{bin} \in \{0,1\}^{n \times m}$ is the original QR Code binary image, $G_{rgb} \in [0,255]^{l \times 3}$ is a color palette to be embedded in the image, and $Z \in \{0,1\}^{n \times m}$ is a mask that protects the QR Code key patterns to be overwritten by the palette.

Our main proposal is one possible implementation of f_{color} (Equation 7), denoted as $f_{color,gray}$ and shown in algorithm 1, where the colors of the palette are mapped to positions of the QR Code based on their affinity to black and white. For each one of these two classes, the particular assignment of a color to one of the many possible pixels of the class (either black or white) is fully arbitrary and allows for further design decisions. In this implementation of the colorize function, we choose to assign the colors in random positions within the class.

The algorithm 1 uses the previous defined functions ($f_{binarize}$, $f_{desaturate}$ and $f_{normalize}$) to simulate the readout process and classify the colors in the palette. In other applications, interested e.g. in preserving a certain color order, additional mapping criteria can be used as shown below. Anyhow, preserving the assignment to the black or white classes based on the color affinity is key for back-compatibility.

Moreover, to illustrate how different colorize functions affect the readout process, we will consider 4 different situations, where f_{color} plays different roles, and we will compute their SNR and BER metrics:

Logo. When a logo-like pattern is encoded, G_{rgb} will be the colors of the logo, the mask Z_{logo} protects all the QR Code except the region where the logo is inserted, and the colorize function $f_{color,logo}$ returns the logo image overlaid on top of the original QR Code image (Figure 4.a.).

Algorithm 1: Colorization function $f_{color,gray}$

Input: $A_{bin} \in \{0,1\}^{n \times m}$, $G'_{rgb} \in [0,255]^{l \times 3}$, and $Z \in \{0,1\}^{n \times m}$
Output: $[0,1]^{n \times m \times 3}$

$W_{pos} \leftarrow []$, $B_{pos} \leftarrow []$
for $i = 0, \ldots, n$ *and* $j = 0, \ldots, m$ **do**
  if $Z(i,j) == 1$ **then**
    if $A_{bin}(i,j) == 1$ **then**
      Append (i,j) to W_{pos}
    else
      Append (i,j) to B_{pos}
$W_{color} \leftarrow []$, $B_{color} \leftarrow []$
$G_{bin} \leftarrow f_{binarize}(f_{desaturate}(f_{normalize}(G'_{rgb})))$
for $k = 0, \ldots, l$ **do**
  if $G_{bin}(k) == 1$ **then**
    Append k to W_{color}
  else
    Append k to B_{color}
$p \leftarrow lenght(W_{color})$, $q \leftarrow lenght(B_{color})$
$W'_{pos} \leftarrow$ Select p random values of W_{pos}
$B'_{pos} \leftarrow$ Select q random values of B_{pos}
$M \leftarrow \{0\}_{i,j} \;\; \forall i \in \{0, \ldots, n\}$ and $j \in \{0, \ldots, m\}$
for $k = 0, \ldots, p$ **do**
  $M(W'_{pos}(k)) \leftarrow W_{color}(k) + 1$
for $k = 0, \ldots, q$ **do**
  $M(B'_{pos}(k)) \leftarrow B_{color}(k) + 1$
$C_{rgb} \leftarrow \{0\}_{i,j,k} \;\; \forall i \in \{0, \ldots, n\}$, $j \in \{0, \ldots, m\}$ and $k \in \{0,1,2\}$
for $i = 0, \ldots, n,\; j = 0, \ldots, m$ *and* $k = 0,1,2$ **do**
  if $M(i,j) == 0$ **then**
    $C_{rgb}(i,j,k) \leftarrow A_{bin}(i,j)$
  else
    $C_{rgb}(i,j,k) \leftarrow G_{rgb}(M(i,j) - 1, k)$
return C_{rgb}

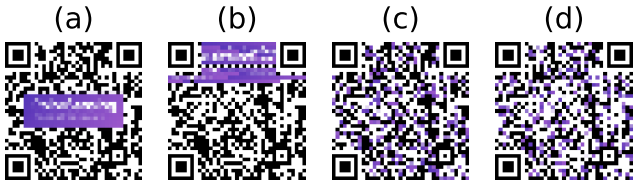

Figure 4: The color information from the ColorSensing logo is distributed using different criteria, each one of these distributions compute different measures of SNR and BER, although the total amount of colors is the same, the way they are distributed affects the signal quality. (a) The original QR Code with the logo. (b) The logo colors are sorted at the top of the QR Code. (c) The logo colors are randomly distributed among the QR Code. (d) The logo colors are distributed by using a threshold criterion among blacks and white colors.

Sorted. We are going to use the colors of the logo (thus G_{rgb} will be the same as before), but we are going to place them on top of the QR Code, sorting them as they appear in the color list. In this case, the mask $Z_{EC\&D}$ excludes only the key protected areas and allow covering with colors all the error correction and data regions. Finally, the colorize function $f_{color,\,sorted}$ establishes that the first color goes to the first available position inside A_{gray} pixels, etc. (Figure 4.b.).

Random. Again we use the same colors of the logo (G_{rgb} remains the same) and the same mask $Z_{EC\&D}$, but now the colorize function $f_{color,\,random}$ defines a random placement of the palette into the available positions of A_{bin} (Figure 4.c.).

Grayscale. Our proposed method, with colors and mask same as before, but the now the colorize function $f_{color,\,gray}$ makes a random assignment of that respects the rule that pseudo-white colors are only assigned to white pixels of A_{bin}, and pseudo-black only to the black ones, as described in algorithm 1 (Figure 4.d.).

Measure	Logo	Sorted	Random	Grayscale
SNR	10.53 dB	10.27 dB	10.35 dB	12.23 dB
BER	8.55 %	8.33 %	8.62 %	0.00 %

Table 2: Values of SNR and BER computed for each criteria in Figure 4. Using the logo as it is, the sorted criteria and random criteria yield to similar results. However, the use of a simple grayscale threshold criteria slightly increases the SNR and hugely depletes the BER, showing a good result for encoding colors in a back-compatible way.

Finally, Table 2 shows the SNR and BER figures for the four mappings (examplified in the images of Table 2). Using the grayscale approach to encode colors by their resemblance to black and white colors leads to much lower noise levels. Since the original data of the QR Code can be seen as a random distribution of white and black pixels, $f_{color,\,sorted}$ and $f_{color,\,random}$ yield similar results to $f_{color,\,logo}$, encoding the logo itself. Meanwhile, $f_{color,\,gray}$ shows us a 0% BER, and an almost 2dB SNR increase. This suggests that our proposal is an effective way to embed colors into QR Code in a back-compatible manner (see Figure 2), as it is demonstrated in the following sections.

3. Experimental design

Experiments were designed to test our proposed method, we carried out 3 different experiments where QR Codes were filled with colors and then transmitted through different channels. QR Codes were filled with a palette of random colors G_{rgb}. The color substitution factor ranged from only 1% of the available pixel positions in a QR Code replaced with colors up to 100%. Table 3 contains a summary of each experiment designed. Also, we masked the zone where color was embedded allowing to cover three different the error correction and data regions ($Z_{EC\&D}$), only the error correction region (Z_{EC}) and another, only the data region (Z_D).

QR Codes were created with a Python library named `python-qrcode` [22]. In all experiments, we calculated the SNR and BER as a measure of the signal quality of each QR Code once transmitted through different channels. Also, we checked the direct readability by using a QR Code scanner before and after going through the channels. Scans were performed with the popular ZBar scanner and its `pyzbar` wrapper for Python [23].

The use of QR Code in real world conditions imply additional sources of error, like differences in printing, different placements, ambient light effects, effects of the camera and data processing, etc. All these factors can be regarded as sources of noise in a transmission channel.

We considered 3 different channels for the experiments:

Empty. A channel where there is no color alteration due to the channel. It was used as a reference, to measure the noise level induced by the colorization process (see Figure 5.a).

All experiments	Values	Size
Color substitution (%)	1, 5, 10, 15, 20, 30, 40, 50, 60, 70, 80, 100	12
Colorized zone	EC, D, EC&D	3
Colorizing method	Random, Grayscale	2
Experiment 1	Values	Size
Digital IDs	from 000 to 999	1000
QR version	5, 6, 7, 8, 9	5
Channels	Empty, Image augmentation	1 + 1
Experiment 2	Values	Size
Digital IDs	000	1
QR version	5, 6, 7, 8, 9	5
Channels	Empty, Image augmentation	1 + 1000
Experiment 3	Values	Size
Digital IDs	000	1
QR version	5	1
Channels	Empty, Colorimetry setup	1 + 25

Table 3: Summary of parameter values for each experiment designed. All experiments share common parameters, at least each experiment has 72 different QR Codes that were generated using as reference the multiplication of the shared parameters. Experiment 1 generates 360,000 different QR Codes.

Image augmentation. With a data augmentation library [24] we generated images that mimic different printing processes and exposure to different light conditions. With this tool we also applied gaussian blur distortions, crosstalk interferences between the RGB channels and changed contrast conditions. (see Figure 5.b).

Colorimetry setup. We actually printed the QR Codes and captured them with a fixed camera (Raspberry Pi 3 with a Raspberry Pi Camera v2) [25] under different illumination-controlled conditions (Phillips Hue Light strip) [26]. The camera was configured to take consistent images. The light strip was configured to change its illumination conditions with two subsets of illumination conditions: white light (9 color temperatures from 2500K to 6500K) and colored light (15 different colors sampling evenly the CIExyY space) (see Figure 5.c).

Figure 5: The same QR Code with data and the same amount of colors (80% of the data area) is exposed to different channels. (a) The image passed-through an empty channel. (b) The image passed-through an augmentation channel which resembles a warm light scene. (c) The image passed-through a real environment channel, actually printed and captured in a scene with a lamp at 2500K (warm light).

4. Results

4.1. Embedding colors in QRs codes: empty channel

Let us start with the results of Experiment 1, where 360.000 different color QR Codes were encoded (see Table 3). Then, SNR and BER were computed against an empty channel (only the color placement was taken into account as a source of noise). Results show only data from those QR Codes where colors were placed using the $Z_{EC\&D}$ mask, reducing our dataset to 120.000 QR Codes. Figure 6 shows aggregated results of the SNR and BER as a function of the color substitution ratio, for $f_{color,\,random}$ and $f_{color,\,gray}$ methods data is averaged for all QR Code versions (5, 6, 7, 8 and 9) and for all 1000 different digital IDs. These results indicate that SNR and BER are independent of the QR Code versions and the QR Code digital data, since the standard deviation of these figures (shadow area in Figure 6) that

average different versions and digital IDs are very small. Only the BER for $f_{color,random}$ shows a narrow deviation. Of course, all these deviations increased when noise was added (see further results).

Regarding the SNR, it decreases for both $f_{color,random}$ and $f_{color,gray}$ when the total amount of colors increases. We found that our $f_{color,gray}$ proposal (affinity towards black and white) is 6 dB better than $f_{color,random}$, regardless of the quantity of colors embedded, the data, or the version of the QR Code. This means that our proposal to place colors based on their grayscale value is 4 times less noisy than a random method.

Concerning the BER, results show that, before including the effects of a real noisy channel, our placement method leads to a perfect BER score (0%). Instead, with a random substitution, and even in an ideal channel, BER increases linearly and reaches up to a 40% of BER. Taking into account the QR Code resemblance to a pseudo-random pattern, the maximum BER in this scenario is 50%. This slightly better result can be attributed to the fact that we are not tampering with the key protected areas of the QR Code (finder, alignment, timing patterns, ...).

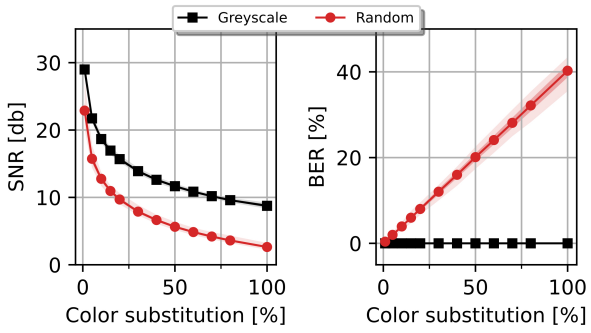

Figure 6: SNR and BER results for Experiment 1 before sending the QR Codes to any channel, only taking into account the QR Codes where all the area has been used ($Z_{EC\&D}$), for both methods $f_{color,gray}$ (squares, black) and $f_{color,random}$ (dots, red). Lines and points show average data, light shadows show the min and max values, and heavy shadows show the standard deviation.

4.2. Image augmentation channel

Results from Experiment 1 showed that the SNR and BER results are independent of the data encoded in the QR Code. Based on this finding, we reduced the amount of different IDs encoded to only one per QR Code version and increased the number of image augmentation channels to 1000. This was the key idea of Experiment 2, and by doing this we achieved the same statistics of a total 360.000 results, from 3.600 QR Codes sent though 1.000 different channels. Focusing again only on the QR Codes that are color embedded using the whole zone ($Z_{EC\&D}$) we ended up with 120.000 results to calculate the corresponding SNR and BER (see Figure 7).

Regarding the SNR, it worsened in comparison with Experiment 1, because now the image augmentation channel is adding noise (see details in Figure 5). The 6 dB difference between $f_{color,random}$ and $f_{color,gray}$ remains for higher color substitution. This can be explained because the noise generated by the color placement is larger than the noise generated by the channel when increasing the amount of colors.

Concerning the BER, it increased up to an average value of about 7% for $f_{color,gray}$ method due to the influence of a noisy channel. In the most extreme cases (channel with the lowest SNR and for the maximum color substitution ratio), BER values do not exceed 20%. Instead, the augmentation channel does not seem to increase the BER for $f_{color,random}$; essentially because it is already close to the theoretical maximum.

We have also observed (see Figure 8) the impact of the channel noise on the SNR and BER figures of $f_{color,random}$ and $f_{color,gray}$ are mostly independent of the QR Code version. Therefore, we can expect that the level of resilience to noise offered by one or another colorize function will remain, independently of the data to encode or the QR Code version needed. That is the reason why we removed the QR Code version from the set of variables to explore in the Experiment 3.

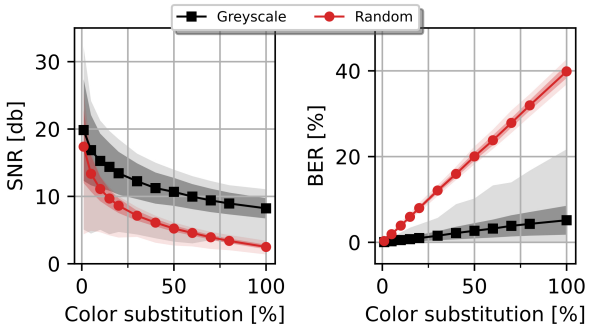

Figure 7: SNR (left) and BER (right) results for Experiment 2 after sending the QR Codes to an image augmentation channel, only taking into account the QR Codes where all the area has been used ($Z_{EC\&D}$), for both methods $f_{color,gray}$ (squares, black) and $f_{color,random}$ (dots, red). Lines and points show average data, light shadows show the min and max values, and heavy shadows show the standard deviation.

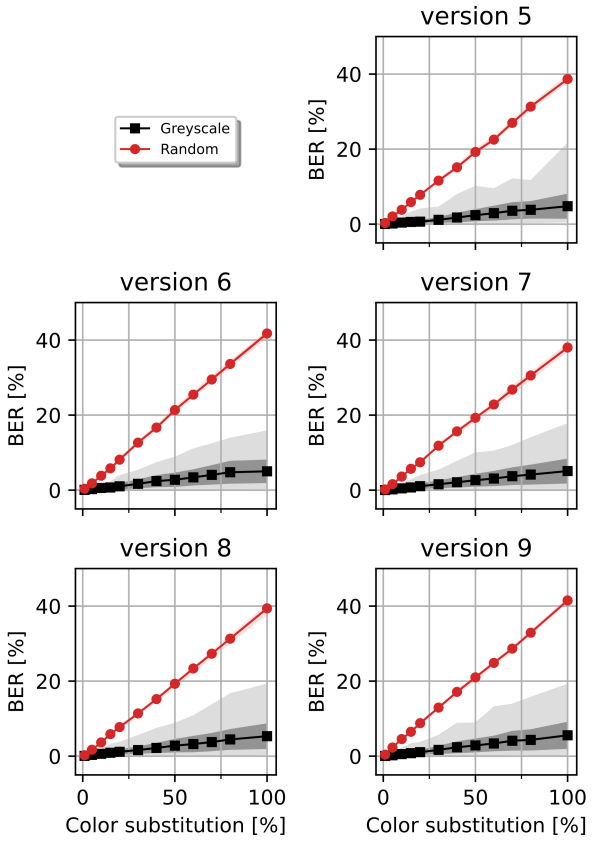

Figure 8: SNR results for Experiment 2, split by QR Code version, after sending the QR Codes to an image augmentation channel, only taking into account the QR Codes where all the area has been used ($Z_{EC\&D}$), for both methods $f_{color,\,gray}$ (squares, black) and $f_{color,\,random}$ (dots, red). Lines and points show average data, light shadows show the min and max values, and heavy shadows show the standard deviation.

4.3. Colorimetry setup as channel

Experiment 3 consisted of only one QR Code v5 (1 ID, 1 version) being colored in 72 different ways (12 color insertion ratios, 2 colorize functions –$f_{color,\,random}$ and $f_{color,\,gray}$– and 3 different zones to embed colors – $Z_{EC\&D}$, Z_{EC} and Z_D–), then printed and exposed to a colorimetry setup with a total of 25 different color illumination conditions captured with a digital camera. We performed this experiment as a way to check if the proposed method and the results obtained with the image augmentation channel held in more severe, and real, capturing conditions. This experiment led to a dataset of 1.800 images acquired from the real world. The calculations of the SNR and the BER were based on those images with colors placed with the $Z_{EC\&D}$ mask, reducing our dataset to 600 results (see Figure 9).

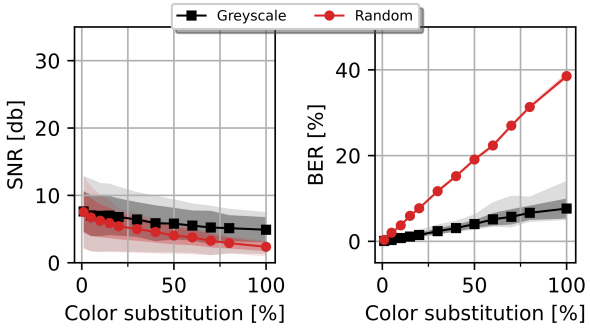

Figure 9: SNR (left) and BER (right) results for Experiment 3 after sending the QR Codes to a real channel (printing and capturing the QR Code in a colorimetry setup), only taking into account the QR Codes where all the area has been used ($Z_{EC\&D}$), for both methods $f_{color,\,gray}$ (squares, black) and $f_{color,\,random}$ (dots, red). Lines and points show average data, light shadows show the min and max values, and heavy shadows show the standard deviation.

Regarding the SNR, as our real channel was quite noisy, averaged values sank more than 10 dB, for all the color substitution ratio and for $f_{color,\,random}$ and $f_{color,\,gray}$. Here, the huge advantage of 6dB observed before for $f_{color,\,gray}$ was not so evident, since the channel was now the main source of noise. This should serve to illustrate that our proposed method starts with an initial advantage in ideal conditions with respect to the random colorize function, which can diminish due to the channel noise but will always perform better.

Regarding the BER, for $f_{color,\,gray}$, the BER values did not increase relative to the image augmentation channel, both distributions overlap in the range of 7-10% of BER. For $f_{color,\,random}$, the linear maximum behaviour up to a BER of 40% is also shown in this situation. As shown in further sections, although noise levels from both methods are similar in practical applications, the difference in how they are translated into BER determines the grayscale colorize function better performing.

4.4. Readability

Up to this point, results show how embedding colors in the QR Codes might increase the probability of encountering bit errors when decoding those QR Codes. Results also indicate that our back-compatible method can reduce the average probability of encountering a bit error from 40% to 7-10%, enabling proper back-compatible QR Code scan using the error correction levels included in the standard that can tolerate this amount of error (levels Q and H). This is a necessary but not sufficient demonstration of back-compatibility.

We must also be sure that the new method offers QR Codes fully readable with conventional decoders. To assess this readability, we checked the integrity of the data of all the QRs in our experiments using ZBar, a well-established barcode scanner [23]. We calculated the success ratio at each color substitution ratio as the amount of successfully decoded QR Codes by ZBar divided by the total amount of QR Codes processed. Also, we analyzed separately the results obtained when embedding colors in the 3 different zones ($Z_{EC\&D}$, Z_{EC} and Z_D masks), in order to identify further relevant behaviours.

On the one hand, readability results of the QR Codes of Experiment 1 (channel without noise) are shown in Figure 10. $f_{color,gray}$, the proposed method, scores a perfect readability, no matter the insertion zone. This is because $f_{color,gray}$ does not actually add BER when colors are inserted. Instead, $f_{color,random}$, the random method, is extremely sensitive to color insertion, and the readability success rate decays rapidly as the number of inserted colors increases. As seen in Experiment 1, the Data zone (Z_D) seems the most promising to embed the largest fraction of colors.

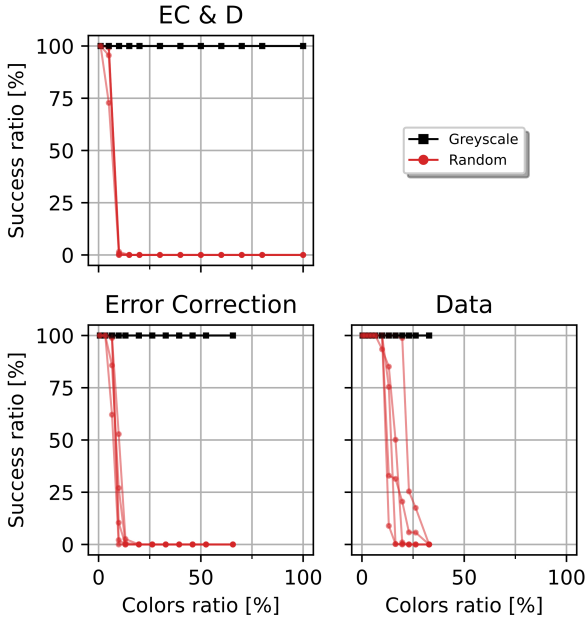

Figure 10: Success ratio of decoded QR Codes before passing through a channel among different embedding zones ($Z_{EC\&D}$, Z_{EC} and Z_D). Each curve represents a QR Code version, there are up to 5 curves for each method, $f_{color,gray}$ (squares, black) and $f_{color,random}$ (dots, red).

On the other hand, results after passing through the noisy channels of Experiment 2 and Experiment 3 are shown in Figure 11 and Figure 12, respectively. Clearly, the noise of the channel also affects the readability of $f_{color,gray}$, but the codes built this way are much more resilient and can allocate a much larger fraction of colors without failing. Even more: if colors are only placed in the Data (Z_D) encoding zone, grayscale mapped color QR Codes remain fully readable until all the available pixels of Z_D are occupied.

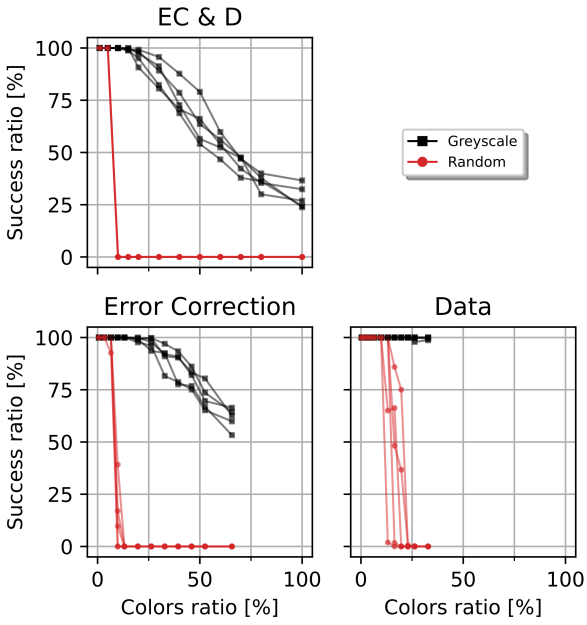

Figure 11: Success ratio of decoded QR Codes after passing through an image augmentation channel among different embedding zones ($Z_{EC\&D}$, Z_{EC} and Z_D). Each curve represents a QR Code version, there are up to 5 curves for each method, $f_{color,gray}$ (squares, black) and $f_{color,random}$ (dots, red).

To get a practical outcome of these results, one should translate the color substitution ratios into the actual amount of colors that these ratios mean when using different encoding zones in QR Codes with different versions. Table 4 summarizes these numbers grouped by encoding zone, QR Code version. Results compare the maximum number of colors that can be allocated using each one of the colorize functions (grayscale vs. random) with at least a 95% of readability. The 95% mark is often used to assess user experience when scanning barcodes with a smartphone device, examples with this metric can be found elsewhere [27].

Clearly, the $f_{color,gray}$ function allows for allocating between 2x to 4x times more colors than a naive $f_{color,random}$ approach. Interestingly, restricting the

		$Z_{EC\&D}$		Z_{EC}		Z_D	
version	size	Greyscale	Random	Greyscale	Random	Greyscale	Random
5	1072	**322**	**54**	**282**	**70**	**352**	**141**
6	1376	**206**	**69**	**448**	**90**	**464**	**139**
7	1568	**314**	**78**	**520**	**104**	**512**	**205**
8	1936	**387**	**97**	**499**	**125**	**672**	**269**
9	2336	**467**	**117**	**461**	**77**	**784**	**235**

Table 4: Number of different colors that can be embedded inside a QR Code with a 95% success ratio during the decoding process for each insertion mask ($Z_{EC\&D}$, Z_{EC} and Z_D), for both colorize function ($f_{color,gray}$ and $f_{color,random}$). In absolute terms, the mask corresponding with only the Z_D zone beats the other two, as expected the $f_{color,gray}$ method performs better than the $f_{color,random}$ one.

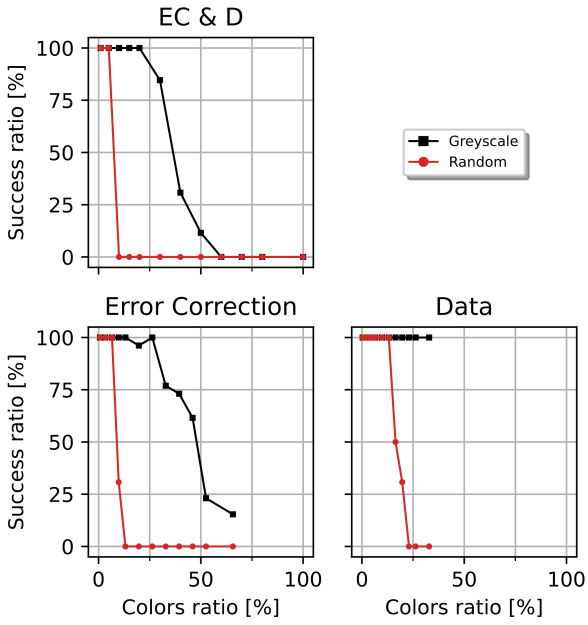

Figure 12: Success ratio of decoded QR Codes after passing through areal-life channel among different embedding zones ($Z_{EC\&D}$, Z_{EC} and Z_D). Each curve represents a QR Code version, there are up to 5 curves for each method, $f_{color,gray}$ (squares, black) and $f_{color,random}$ (dots, red).

placement of colors to the data zone (Z_D) leads to a much larger number of colors, in spite of having less pixels available; being the error correction (Z_{EC}) the less convenient to tamper with. In the best possible combination ($f_{color,gray}$ function, Z_D zone, v9 –the largest version studied–) this proposal reaches an unprecedented number of colors that could be embedded close to 800. As a matter of fact, this could mean sampling a 3-dimensional color space of 24 bits resolution (i.e. sRGB) with 9^3 colors evenly distributed along each axis.

Needless to say, that such figures can be systematically increased with QR Codes of higher versions. To get a specific answer to the question of how many colors can be embedded as a function of the QR Code version in the best possible conditions -data zone (Z_D) with our grayscale function ($f_{color,gray}$)-, we generated a specific dataset of QR Codes with versions running from v3 to v40, and checked their 95% readability in the conditions of Experiment 2 through 50 image augmentation channels (see Figure 13).

Results indicate that thousands of colors are easily to reach, with a theoretical maximum of almost 10,000 colors with QR Codes v40. In real life, however, making these high version QR Codes readable with conventional cameras at reasonable distances means occupying quite a lot of space (about 5 inches for a QR Code v40). That size, though, is comparable to that of a ColorChecker pattern but giving access to thousands of colors instead of only tens.

4.5. Example of use case

Finally, we illustrate how this approach can be applied to carry out actual color correction problems with full QR Code back-compatibility, using the 24 colors from the original ColorChecker [2] to create a barcode that contains them (see Figure 14). We created a compact color QR Code version 5 with H error correction

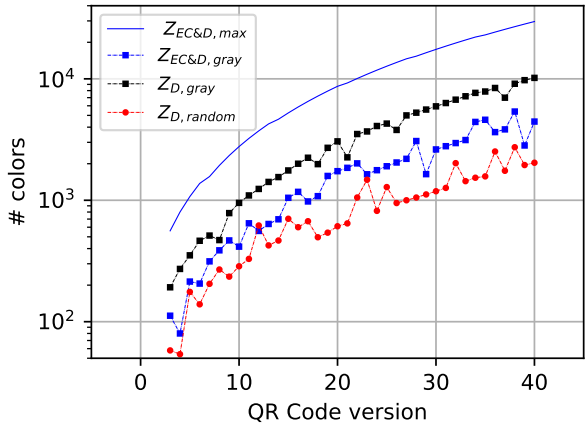

Figure 13: The measured number of colors that can be embedded in the Z_D zone using the $f_{color,\,gray}$ with a readability of a 95% in the conditions of Experiment 2 ($Z_{D,gray}$) as a function of the QR Code version (from v3 to v40) is compared to other configurations: the theoretical maximum number of colors ($Z_{EC\&D,max}$), the measured number of colors using the $Z_{EC\&D}$ zone and the $f_{color,\,gray}$ method ($Z_{EC\&D,gray}$) and number of colors using Z_D and the $f_{color,\,random}$ method ($Z_{D,random}$).

Figure 14: A color QR Code (version 5 with H error correction level) which contains 240 pixels that are coloured. This is implemented with our back-compatible method. These color pixels reproduce the 24 original ColorChecker colors with a redundancy of 10 pixels per color. Only 22% of the digital data pixels are used in this process, almost all the Z_D zone is used to allocate the colors.

level. According to our findings, this setup should let us embed 352 colors in the data zone (Z_D) zone without risking readability. In this example, this allowed us to embed up to 10 replicas of the 24 color references, offering plenty of redundancy to detect variation of the color calibrations across the image or to improve the correction itself. Table 5 shows the main quantitative results obtained with this colored QR Code, submitted to the conditions of Experiment 2, with one empty channel and 120,000 image augmentation channels.

Encoding		
Digital ID	000	
Version	5	
Error correction level	H	
Unique colors	24	colors
Total embedded colors	240	colors
Color substitution ratio	22	%
Empty channel	1	channel
SNR	12.68	dB
BER	0.0	%
Success ratio	100	%
Augmentation channels	120,000	channels
SNR	11 ± 2	dB
BER	2.7 ± 1.7	%
Success ratio	96	%

Table 5: Properties of the proposed QR Code with the ColorChecker colors embedded in it. Properties are related with different steps in the QR Code life-cycle, from encoding to decoding.

5. Conclusions

We have presented a method to pack a set of colors, useful for color calibration, in a QR Code in a fully back-compatible manner; this is, preserving its conventional ability to store digital data. By doing so, we enhanced the state-of-the-art color charts with three main features: (1) we did leverage the computer vision readout of the color references to the QR Code features; (2) we reduced *de facto* the size of the color charts to the usual size of a QR Code, one or two inches; and (3) we have demonstrated that the color capacity of the QR Codes constructed this way (up to a few thousand colors!) is orders of magnitude higher than that found in any other previous color chart, due to the image density and pattern recognition robustness of the QR Codes.

Also, compared to other colored QR Codes, our proposal, based on the grayscale affinity of the colors to white or black, leads to much lower signal alteration levels and thus much higher readability, than the one found in more naive approaches like, e.g. random assignment methods, which represent the aesthetic QR Codes (i.e. printing a logo). Plus, our method did not affect the computational performance of reading QR Codes with colors (see Supplementary Materials section C).

Moreover, tuning how we defined our criteria of color embedding upon the affinity of colors to black and white would lead to more efficient embedding methods one could develop more complex assignment criteria. As a

matter of fact, using other color affinity definitions, such as: a perceptual luminance value – $f_{desaturate}(r, g, b) = 0.2126 \cdot r + 0.7152 \cdot g + 0.0722 \cdot b$ –, a lightness definition – $f_{desaturate}(r, g, b) = \frac{1}{2} \max(r, g, b) + \frac{1}{2} \min(r, g, b)$ – [28], etc. Also, one could modify $f_{colorize}$ to treat differently colors with grayscale values in the middle range that accumulate a higher error impact: pairing the Z_{EC} with those colors far from the middle range and Z_D with those with in the middle range, as we demonstrated Z_D accommodates better defective bits.

Finally, we have assumed that our *back-compatible Color QR Codes* are meant to be used as machine-readable patterns to act as color charts. But, the above-presented results could be applied to encode data in the color of the QR Codes in a back-compatible manner. For example, the QR Codes proposed by Blasinski et al. [15] that contain 3 multiplexed QR Codes are not back-compatible. One could use our criteria to multiplex these patterns into a QR Code in a back-compatible manner.

Funding

This work has been funded in part by the European Research Council under the H2020 Framework Program ERC Grant Agreements no.727297 and no. 957527. J.D. Prades acknowledges the support from the DFG GrK NanoMet, BBVA Leonardo, Serra Húnter and ICREA Academia programs.

Authors

Ismael Benito-Altamirano is a Physicist, Electronics in Telecommunication Engineer and a M.Sc. in Photonics by the UB and the UPC, he is also a PhD in Engineering and Applied Science by the UB. As part of his PhD, he has worked on the implementation of machine-readable patterns to readout colorimetric sensors. He is also the former CTO of ColorSensing.

David Martínez-Carpena is a Mathematician and a Computer Science Engineer by the Universitat de Barcelona. He is also an M.Sc. in Advanced Mathematics by the UB. Also, he is a former employee of ColorSensing.

Olga Casals is an Optic and Optometrist by the UPC, a Physicist and a PhD in Electronic Engineering and Technology by the UB. After working in the R&D department of TECIL S.A, she returned to UB to hold different postdoc fellowships, including a TecnioSpring with a stay in Braunschweig University of Technology.

Cristian Fàbrega is a Physicist and PhD in Nanoscience and Nanotechnology by the University of Barcelona, where he is an Associate Professor since 2017. His research is focused on innovatives materials and devices for sensing and energy storage applications.

Andreas Waag is a Physicist by the Würzburg University. After a research stay at Purdue University, and Ulm University, he today is head of the Institute of Semiconductor Technology at Braunschweig University of Technology. Also, he is speaker of the Laboratory for Emerging Nanometrology (LENA) and head of the epitaxy competence center (ec2), devoted to GaN technology.

J. Daniel Prades is a Physicist, Electronic Engineer and PhD in Nanoscience by the Universitat de Barcelona, where he is Full Professor since 2019 and leads his group on gas and optical sensors. He is also a co-funder and CSO of ColorSensing, a spin-off company from UB.

References

[1] C. Ruppert, N. Phogat, S. Laufer, M. Kohl, H. P. Deigner, A smartphone readout system for gold nanoparticle-based lateral flow assays: application to monitoring of digoxigenin, Microchimica Acta 186 (2) (2019). doi:10.1007/s00604-018-3195-6.

[2] C. S. McCamy, H. Marcus, J. G. Davidson, COLOR-RENDITION CHART., J Appl Photogr Eng 2 (3) (1976) 95–99.

[3] P. H. Carvalho, I. Rocha, F. Azevedo, P. S. Peixoto, M. A. Segundo, H. P. Oliveira, Cost-efficient color correction approach on uncontrolled lighting conditions, in: International Conference on Computer Analysis of Images and Patterns, Springer, 2021, pp. 90–99.

[4] M. N. Sakinah, A. H. Saputro, Color correction technique using an artificial color board and root-polynomial color correction for smartphone-based urinalysis, in: 2021 17th International Conference on Quality in Research (QIR): International Symposium on Electrical and Computer Engineering, IEEE, 2021, pp. 21–26.

[5] N. M. Kinyanjui, T. Odonga, C. Cintas, N. C. F. Codella, R. Panda, P. Sattigeri, K. R. Varshney, Fairness of classifiers across skin tones in dermatology, in: Medical Image Computing and Computer Assisted Intervention – MICCAI 2020, Springer, Cham, 2020, pp. 320–329.

[6] Z. Li, X. Zhang, H. Müller, S. Zhang, Large-scale retrieval for medical image analytics: A comprehensive review, Medical Image Analysis 43 (2018) 66–84. doi:10.1016/j.media.2017.09.007.

[7] D. Wu, D.-W. Sun, Colour measurements by computer vision for food quality control – a review, Trends in Food Science & Technology 29 (1) (2013) 5–20. doi:10.1016/j.tifs.2012.08.004.

[8] G. M. Fernandes, W. R. Silva, D. N. Barreto, R. S. Lamarca, P. C. F. L. Gomes, J. F. da S Petruci, A. D. Batista, Novel approaches for colorimetric measurements in analytical chemistry – a review, Analytica Chimica Acta 1135 (2020) 187–203. doi:10.1016/j.aca.2020.07.030.

[9] H. S. Jung, P. Verwilst, W. Y. Kim, J. S. Kim, Fluorescent and colorimetric sensors for the detection of humidity or water content, Chem. Soc. Rev. 45 (5) (2016) 1242–1256. doi:10.1039/c5cs00494b.

[10] Y. Zhang, L.-T. Lim, Colorimetric array indicator for NH3 and CO2 detection, Sensors and Actuators B: Chemical 255 (2018) 3216–3226. doi:10.1016/j.snb.2017.09.148.

[11] P. Menesatti, C. Angelini, F. Pallottino, F. Antonucci, J. Aguzzi, C. Costa, RGB color calibration for quantitative image analysis: The "3D Thin-Plate Spline" warping approach, Sensors (Switzerland) 12 (6) (2012) 7063–7079. doi:10.3390/s120607063.

[12] G. D. Finlayson, M. MacKiewicz, A. Hurlbert, Color Correction Using Root-Polynomial Regression, IEEE Transactions on Image Processing 24 (5) (2015) 1460–1470. doi:10.1109/TIP.2015.2405336.

[13] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, R. Medina-Carnicer, Generation of fiducial marker dictionaries using Mixed Integer Linear Programming, Pattern Recognition 51 (2016) 481–491. doi:10.1016/j.patcog.2015.09.023.

[14] Information technology - automatic identification and data capture techniques - qr code bar code symbology specification, ISO ISO/IEC 18004:2015, International Organization for Standardization (2015).

[15] H. Blasinski, O. Bulan, G. Sharma, Per-colorant-channel color barcodes for mobile applications: An interference cancellation framework, IEEE Transactions on Image Processing 22 (4) (2013) 1498–1511. doi:10.1109/TIP.2012.2233483.

[16] Z. Fu, L. Fang, H. Huang, B. Yu, Distributed three-level qr codes based on visual cryptography scheme, Journal of Visual Communication and Image Representation (2022) 103567.

[17] H. Su, J. Niu, X. Liu, Q. Li, J. Wan, M. Xu, T. Ren, Artcoder: An end-to-end method for generating scanning-robust stylized qr codes, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 2277–2286.

[18] K. Pena-Pena, D. L. Lau, A. J. Arce, G. R. Arce, Qrnet: Fast learning-based qr code image embedding, Multimedia Tools and Applications 81 (8) (2022) 10653–10672.

[19] I. Benito-Altamirano, P. Pfeiffer, O. Cusola, J. Daniel Prades, Machine-Readable Pattern for Colorimetric Sensor Interrogation, Proceedings 2 (13) (2018) 906. doi:10.3390/proceedings2130906.

[20] L. Engel, I. Benito-Altamirano, K. R. Tarantik, C. Pannek, M. Dold, J. D. Prades, J. Wöllenstein, Printed sensor labels for colorimetric detection of ammonia, formaldehyde and hydrogen sulfide from the ambient air, Sensors and Actuators, B: Chemical 330 (2021). doi:10.1016/j.snb.2020.129281.

[21] J. Qian, B. Xing, B. Zhang, H. Yang, Optimizing qr code readability for curved agro-food packages using response surface methodology to improve mobile phone-based traceability, Food Packaging and Shelf Life 28 (2021) 100638.

[22] L. Loop, Pure python qr code generator, https://github.com/lincolnloop/python-qrcode (2010).

[23] L. N. H. Museum, pyzbar - python wrapper for zbar, https://github.com/NaturalHistoryMuseum/pyzbar (2016).

[24] A. B. Jung, K. Wada, et al., imgaug, https://github.com/aleju/imgaug (2020).

[25] M. Pagnutti, R. E. Ryan, G. Cazenavette, M. Gold, R. Harlan, E. Leggett, J. Pagnutti, Laying the foundation to use Raspberry Pi 3 V2 camera module imagery for scientific and engineering purposes, Journal of Electronic Imaging 26 (1) (2017) 013014. doi:10.1117/1.jei.26.1.013014.

[26] C. Cusano, P. Napoletano, R. Schettini, Evaluating color texture descriptors under large variations of controlled lighting conditions, Journal of the Optical Society of America A 33 (1) (2016) 17. doi:10.1364/josaa.33.000017.

[27] X. Tian, S. Qin, B. Jiang, Y. Gao, X. Wang, Fast batch reading densely deployed qr codes, IEEE Transactions on Mobile Computing (2021). doi:10.1109/TMC.2021.3106763.

[28] R. Hunt, The reproduction of colour, Color Research & Application 30 (6) (2005) 466–467. doi:10.1002/col.20163.