Accepted Manuscript
Version of record at: https://doi.org/10.1007/s11042-022-12357-6

QRnet: fast learning-based QR code image embedding

Karelia Pena-Pena[1], Daniel L. Lau[2],
Andrew J. Arce[3], and Gonzalo R. Arce[1]

[1] University of Delaware, Newark, DE 19716, USA
[2] University of Kentucky, Lexington, KY 40506, USA
[3] Newark, DE 19711, USA

Contact:
Karelia Pena-Pena kareliap@udel.edu
Daniel L. Lau dllau@uky.edu
Andrew J. Arce ajarce100@gmail.com
Gonzalo R. Arce arce@udel.edu

Abstract
Quick Response (QR) codes usage in e-commerce is on the rise due to their versatility and ability to connect offline and online content, taking over almost every aspect of a business from posters to payments. Thus, many efforts have aimed at improving the visual quality of QR codes to be easily included in publicity designs in billboards and magazines. The most successful approaches, however, are slow since optimization algorithms are required for the generation of each beautified QR code, hindering its online customization. The aim of this paper is the fast generation of visually pleasant and robust QR codes. The proposed framework leverages state-of-the-art deep-learning algorithms to embed a color image into a baseline QR code in seconds while keeping a maximum probability of error during the decoding procedure. Halftoning techniques that exploit the human visual system (HVS) are used to smooth the embedding of the QR code structure in the final QR code image while reinforcing the decoding robustness. Compared to optimization-based methods, our framework provides similar qualitative results but is 3 orders of magnitude faster.

Keywords QR codes · Machine learning · Optimization-free

1 Introduction

Quick Response (QR) codes are extensively used in e-commerce and publicity campaigns, engaging new customers by offering discounts, and measuring the impact of the campaign

Accepted Manuscript
Version of record at: https://doi.org/10.1007/s11042-022-12357-6

[1]. QR codes are decodable by any smartphone, and thus, constitute a powerful tool for mobile commerce applications, providing a link between offline and online marketing [25]. The black and white structure of standard QR codes, however, hinders their insertion into artistic designs. Furthermore, traditional QR codes do not provide any visual information about the encoded message. Currently, due to the pandemic crisis, the use of QR codes is on the rise since businesses are looking for ways to offer customers a touch-free experience [18, 25]. For all these reasons, a fast method for the generation of visually enhanced QR codes has emerged as an attractive tool for both big and small businesses.

There are several works related to improving the visual appearance of QR codes without losing decoding robustness. Initial approaches rely on the error correction capacity of QR codes, replacing some modules of the QR code with the desired image [22]; however, to retain the decodability of the code, the ratio between image and code area should be approximately proportional to the error correction capacity of the QR code [22]. Cox et al. [7] proposed a method that exploits the characteristics of the encoding procedure of QR codes to modify the padding modules within the QR code without affecting their decoding robustness by utilizing a Gauss-Jordan elimination procedure. Their approach takes advantage of the padding bits within the QR code bit-stream to include an image or logo, which restricts the area available for the image.

Other approaches take advantage of the decoding process of standard QR codes and modify the QR codes based on the desired luminance or color matching [3, 8, 26]. In [26], the image luminance corresponding to the central pixels of each QR code module is modified, preserving the contrast between the encoding image and the blended image but creating undesirable artifacts. In [3], the authors rely on image blending to alter the appearance of the code, modifying the luminance of the pixels based on the average luminance, blended QR code, and two threshold parameters that control the robustness and visual quality trade-off. Chu et al. [6], inspired by halftoning techniques, proposed a method that generates halftone QR code images. This is achieved by subdividing each QR code module in 3×3 pixels and keeping the center pixel to be equal to the QR code module while the rest of the pixels are chosen to minimize not only the visual quality but also the probability of error that is estimated experimentally using synthetic data. The computational complexity of this data-driven approach grows with changes in the resolution of the image. Garateguy et al. [8], on the other hand, modified the luminance of central and non-central pixels to embed a color image into the whole area of the QR code. In this method, a halftone mask [15, 16, 20] is used to distribute the modified non-central pixels, and a probabilistic model is proposed to predict the distortion generated by the embedding image. The concentration of modified pixels and their luminance were optimized to minimize a visual distortion metric subject to a constraint in the maximum probability of error.

Most recent approaches, not only take advantage of the decoding process but also the encoding process of the QR codes to further improve their visual quality [17, 31, 32]. In [17], a two-stage QR code generation method is proposed. In the first stage based on the work proposed by Cox et al. [7], a binary baseline QR code that is more similar to the color image but robust under traditional binarization methods is generated. In the second stage, the brightness of the image is modified based on the binary baseline QR code and the pixel-based thresholded color image. Even though significant improvement is achieved on the visual quality, the robustness of these codes is inferior compared to other approaches [30]. Using a very similar framework, Zang et al. [32] proposed the combination of a module-based blending that generates the binary baseline QR code followed by a pixel-based blending that generates the color QR code. Even though this method is fast, it lacks an

Accepted Manuscript
Version of record at: https://doi.org/10.1007/s11042-022-12357-6

error correction mechanism to ensure decodability. Xu et al. [31] introduced a style-transfer-based method to generate Stylized Aesthetic QR codes. Although this is a learning-based method, a feed-forward network needs to be trained for each specific target style [12] and the final Stylized Aesthetic QR code is the output of a robustness-optimization algorithm that iteratively checks and repairs each module's robustness.

Although some of the previous methods successfully embed a color image into a QR code while keeping a low probability of error, these methods are either slow due to their reliance on optimization procedures or fast but without decoding robustness guarantees. In [8] for instance, the embedding of QR code images is formulated as an optimization problem that minimizes the visual distortion of the color image subject to a constraint in the probability of error. Even though QR code images generated by this method are visually pleasant and robust during decoding, its computational complexity prohibits its commercial applications. In this work, this problem is addressed by leveraging the power of deep learning, designing, and training a convolutional neural network (CNN) that embeds the color image into a black-and-white QR code. While the training of the CNN architecture is resource and time demanding, the training only happens once. After training, the embedding is simple and fast.

As shown in Fig. 1, the proposed framework consists of two stages. In the first stage, a binary baseline QR code $\mathbf{Q}_b$ that is more similar to the embedding image is generated, taking advantage of the method proposed by Cox et al. [7]. Different from other approaches, the local binarization process performed by popular decoders is taken into consideration, further improving on the decoding robustness as it is shown in our experiments. The second stage corresponding to the image embedding, traditionally based on optimization algorithms [8], is substituted by a CNN architecture, dubbed as QRnet, as depicted in Fig. 1. After experimenting with many network architectures, QRnet resulted in being the best suited for

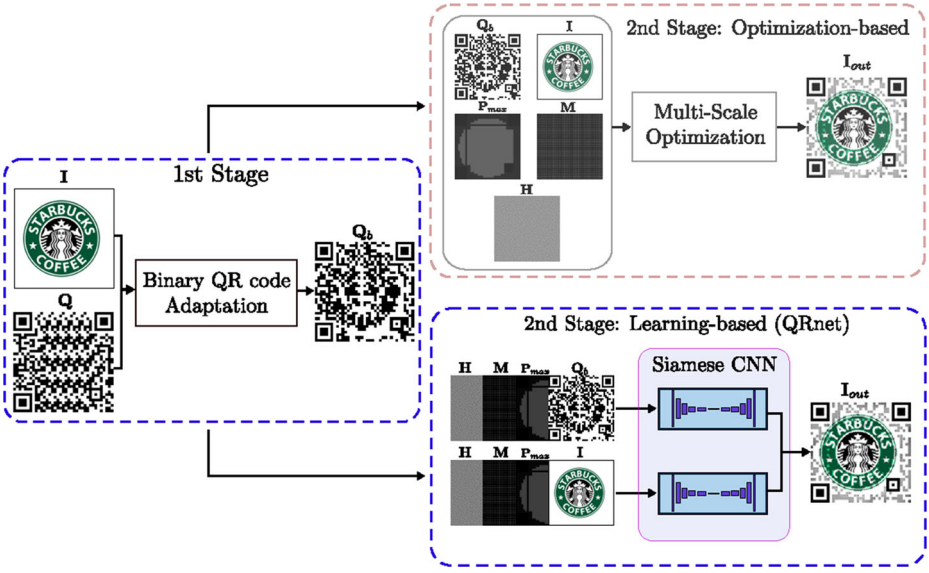

Fig. 1 Two-stage framework for fast QR code image generation (blocks in blue-dashed lines). The 1st stage aims at generating the binary adapted QR code $\mathbf{Q}_b$. The 2nd stage embeds the color image $\mathbf{I}$ into $\mathbf{Q}_b$. Traditionally, the 2nd stage is performed by optimization-based methods (top red-dashed block), which are slow. Instead, we use the proposed QRnet (bottom blue-dashed block), significantly speeding up the process

Accepted Manuscript
Version of record at: https://doi.org/10.1007/s11042-022-12357-6

this application. This architecture is composed of two identical branches, each inspired by U-net with dense layers for feature reuse, blending the image at multiple scales and still preserving the resolution of the embedded image [10, 14, 21]. To the best of our knowledge, it is the first time that a CNN architecture is used for end-to-end embedding of an image into a QR code, significantly speeding up the process.

The contribution of this manuscript is thus three-fold. First, a CNN architecture is proposed and successfully used to perform the embedding of an image into a QR code. Second, the selection of the QR code modules to be modified considers not only the contrast but also the saliency map and the edges; simultaneously reducing the visual contrast and giving priority to the region of interest in the image. Third, each element within the framework is carefully designed to consider the local binarization process performed by the decoder, generating codes that are more robust when compared to the prior art.

The remainder of this paper is organized as follows. Section 2 presents the multi-scale optimization-based QR code embedding procedure, adapted from [8], and used for the generation of the training data; together with the description of the proposed CNN architecture corresponding to the second stage of the framework. Section 3 introduces the first stage of the proposed framework, devoted to generating a binary adapted QR code. Section 4 presents the evaluation and comparison of the results. Section 5 introduce the conclusions.

2 QR code image embedding

In this section, we first introduce the optimization-based method for QR code image embedding used for the generation of a baseline dataset, followed by the description of the proposed network architecture for the fast learning-based and optimization-free QR code image embedding.

2.1 Multi-scale optimized QR code image embedding (MS-Opt)

For the generation of the dataset, we used a modified version of the method proposed by Garateguy et al. [8] in which the luminance of the input color image $\mathbf{I}$ is modified to embed the QR code information, taking advantage of the decoding procedure of standard QR codes (see Appendix B) to achieve a trade-off between the visual quality and the decoding robustness of the QR code. This section focuses on the proposed modifications to the embedding algorithm, a more detailed explanation can be found in [8].

For the embedding, given the fact that during decoding only the central pixel of each module within the QR code is sampled, central pixels in each module are selected for luminance modification by the mask $\mathbf{M}$, and their luminance parameters $\boldsymbol{\alpha}_c$, $\boldsymbol{\beta}_c$ are optimized to encode the QR code information. Non-central pixels could be sampled due to errors during the sampling grid estimation; hence, a halftone mask $\mathbf{H}$ with concentration parameter $\mathbf{p_c}$ is used to select non-central pixels for modification with luminance parameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$. The use of the halftone mask to select non-central pixels improves the decoding robustness as well as reduces the coarse square structure of the QR code. Pixels that are not selected for modification preserve the luminance of the input image $\mathbf{I}$ [8]. As shown in Fig. 2, the inputs to the multi-scale optimization are the original color image $\mathbf{I}$, the QR code $\mathbf{Q}_b$, the mask for the maximum probability of error $\mathbf{P}_{\max}$, and the masks $\mathbf{M}$ and $\mathbf{H}$ for the selection of central and non-central pixels, respectively. As in [8], the concentration of modified pixels $\mathbf{p_c}$ and

Accepted Manuscript
Version of record at: https://doi.org/10.1007/s11042-022-12357-6

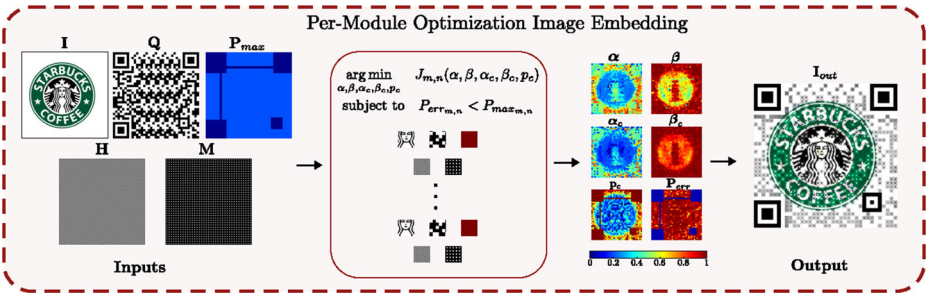

Fig. 2 Optimization-based QR code embedding procedure. The inputs are the original color image $\mathbf{I}$, the standard QR code $\mathbf{Q}$, the maximum probability of error $\mathbf{P}_{\max}$, and the masks $\mathbf{M}$ and $\mathbf{H}$ for the selection of central and non-central pixels, respectively. All the input images are subdivided in local windows of size $5w \times 5w$ centered at each QR code module of size $w \times w$ to be optimized independently. The concentration of modified pixels and their luminance are finally combined to generate the final embedding $\mathbf{I}_{out}$

the luminance parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\alpha_c}, \boldsymbol{\beta_c}$ are locally optimized to achieve a trade-off between visual quality and decoding robustness.

To estimate the probability of decoding error, we use a normalized version of the probability of error model proposed in [8] (explained in Appendix C) in which two independent probability models are developed and combined based on the sampling accuracy of the central pixels of the QR code modules. In this paper, for the local optimization, we propose the use of a module-based overlapping window, motivated by the thresholding method utilized by the popular ZXing library [19] (explained in Appendix B). In this way, the result is more robust against the difference between the size of the image at the generation stage and the size of the image at the scanning stage of the QR code. To this end, given a QR code with $q \times q$ modules, the $(m, n)-$th module of size $w \times w$ is optimized independently using an overlapping window of 5×5 modules centered at the $(m, n)-$th module in the QR code. A total of q^2 independent local optimizations are then required. The formulation of the local optimization problem for each overlapping window consists of minimizing a visual quality metric subject to a constraint over the corresponding probability of error as

$$\begin{aligned} &\underset{\alpha,\beta,\alpha_c,\beta_c,p_c}{\arg\min} \quad J_{m,n}(\alpha, \beta, \alpha_c, \beta_c, p_c) \\ &\text{subject to} \quad P_{err_{m,n}} < P_{\max_{m,n}}, \end{aligned} \tag{1}$$

where $P_{err_{m,n}}$ is the probability of error of the (m, n)-th module; $P_{\max_{m,n}}$ is a parameter selected by the user, which determines the maximum probability of error of each module; and $J_{m,n}(\alpha, \beta, \alpha_c, \beta_c, p_c)$ is the distortion quality metric defined in [8] computed on the $(m, n)-$th overlapping window. Since the optimization process is performed independently for each window, it can be parallelized and a different bound in the probability of error $P_{\max_{m,n}}$ can be given to each module. For instance, a higher bound in the probability of error can be given to those modules that are modified during the first stage.

Finally, since color differences are measured in the HSL color space which is a perceptually uniform color space [8], to obtain the optimal RGB values of a modified pixel given the target luminance Y, the input image $\mathbf{I}$ is first transformed to the HSL color space $(H, S, L) = T(R, G, B)$. Then, the H and S components are kept fixed and the L component is changed depending on the luminance Y. An optimization problem is formulated in [8] to find the value of L, which is computationally expensive due to the required

Accepted Manuscript
Version of record at: https://doi.org/10.1007/s11042-022-12357-6

number of evaluations of the objective function. We, on the other hand, determine L from the piecewise, linear, and monotone function $Y = f(L)$ as follows

$$L^* = \begin{cases} \frac{Y}{2\,Y_{0.5}} & \text{if } L \leq 0.5, \\ Y - 2\,Y_{0.5} + \frac{1}{2-2\,Y_{0.5}} & \text{if } L > 0.5, \end{cases} \tag{2}$$

where $Y_{0.5}$ is the luminance value when $L = 0.5$, i.e. for $(R, G, B) = T^{-1}(H, S, 0.5)$, and $Y = 0.2989R + 0.5870G + 0.114B$.

2.2 Fast deep learning-based QR code image embedding

Even though the optimization-based method presented in Section 2.1 successfully embeds the QR code into the color image with a constraint in the maximum probability of error, this is slow and computationally expensive. Inspired by the developments in learning-based image operators [4], we propose a learning-based QR code image embedding operation by means of a convolutional neural network architecture (CNN). Different architectures have been explored to learn an image operator f that takes a color image $\mathbf{I}$ as input and transforms its content without changing its dimensions [4] e.g. image processing operators like multiple variational models, non-local dehazing, photographic style transfer, and non-photo-realistic stylization. In our case, we aim at learning an operator that transforms multiple input images to the desired output image $\mathbf{I}_{out}$; all with the same size. Additionally, the network should adjust to different image resolutions, generating as output an RGB image of the same size as the inputs.

A naive approach would be to concatenate all input images and use a fully-convolutional network proposed either for fast image processing [4] or for pixel-wise classification [21] with regression loss to produce continuous colors. After experimenting with many already proposed network architectures, we designed the proposed architecture QRnet, shown in Fig. 3, achieving the best performance at embedding an image into a QR code. In the optimization-based QR code embedding approach (Section 2.1), color differences are measured in the HSL color space which is a perceptually uniform color space [8]. Thus, the QR code is embedded directly into the lightness (L) of the image, while the hue (H) and the saturation (S) are kept the same. Taking this into account, QRnet has a siamese architecture in which the baseline QR code $\mathbf{Q}_b$ and the lightness of the color image $\mathbf{L}$, each concatenated with the masks $\mathbf{P}_{\max}$, $\mathbf{M}$, and $\mathbf{H}$, are fed to identical branches, as depicted in Fig. 3, blending the features from the different images at different scales to generate the lightness of the embedded image $\mathbf{L}_{out}$. Siamese architectures have been previously used in stenography and image recognition settings [14, 29]. Each network branch is inspired on U-net for multi-scale analysis [21]. Even though U-net was initially proposed for segmentation, it has been used successfully in other applications [11, 23]. In this case, the U-net shape loosens the image structure by extracting its main features yet reconstructing the embedded image by using not only these features but reusing each multilevel feature mapping such that the embedding is achieved but the quality and main structure of the image is preserved. Additionally, dense blocks for feature reuse [10] are included at each level. Each dense block is composed of dense layers in which new extracted features are not combined but stacked onto the next layers instead. The image information from all different scales is then carried out through the whole network, exploiting the high correlation between the inputs and the output. The output is finally obtained by the blending of the multi-scale reconstructed outputs of each branch.

Accepted Manuscript
Version of record at: https://doi.org/10.1007/s11042-022-12357-6

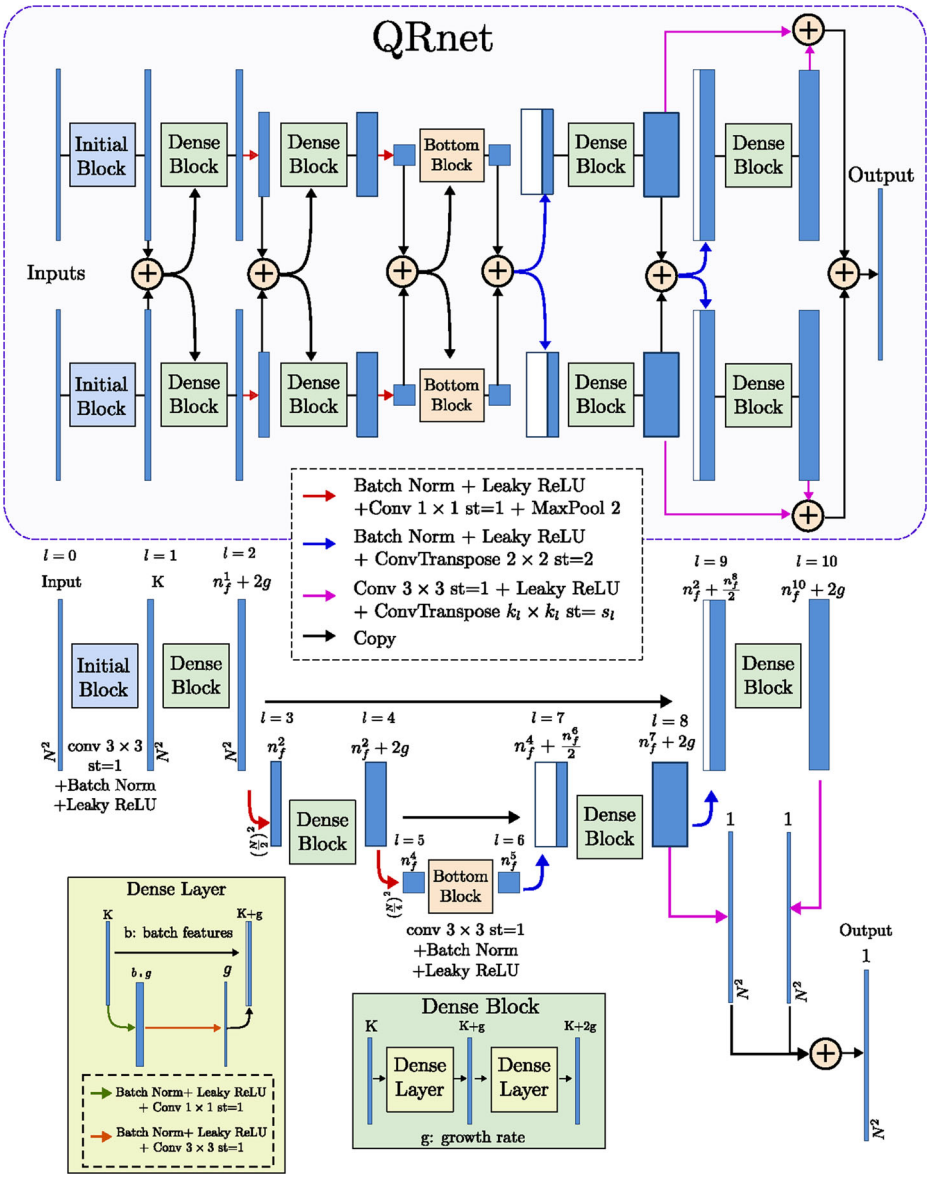

Fig. 3 Siamese CNN Architecture for QR code image embedding conformed of two identical branches (top). The architecture of each branch (bottom) is based on U-Net [21] and dense blocks for feature reuse [10] (bottom left). The inputs for the first branch are the QR code $\mathbf{Q}$ and the masks $\mathbf{P}_{max}$, $\mathbf{M}$, and $\mathbf{H}$, and for the second branch are the lightness of the color image $\mathbf{L}$ and the masks $\mathbf{P}_{\max}$, $\mathbf{M}$, and $\mathbf{H}$. n_f^l is the number of features maps in layer l

To introduce the network, let the QRnet be denoted by $E()$ and $\mathcal{K}$ be the set of parameters across the whole network. Then $E_{\mathcal{K}}()$ is trained on a set of inputs-output pairs $\mathcal{D} = \{\mathcal{I}_i, f(\mathcal{I}_i)\}$ where $(\mathbf{L}_{out})_i = f(\mathcal{I}_i)$ and $\mathcal{I}_i = \{\mathbf{L}_i, \mathbf{M}_i, \mathbf{H}_i, \mathbf{Q}_i, \mathbf{P}_{\max i}\}$ correspond to the i-th element in the dataset. To map the inputs $\mathcal{I}$ to the desired output $\mathbf{L}^{out}$, the network

Accepted Manuscript
Version of record at: https://doi.org/10.1007/s11042-022-12357-6

parameters $\mathcal{K}$ are optimized, by minimizing the following regression loss across all the images in the dataset:

$$E^*_{\mathcal{K}} = \arg\min_{E_{\mathcal{K}}} \quad \frac{1}{N} \|\mathbf{L}_{out} - E_{\mathcal{K}}(\mathbf{L}, \mathbf{Q}, \mathbf{M}, \mathbf{H}, \mathbf{P}_{\max})\|_2^2 \tag{3}$$

where N is the number of pixels in each of the input images. In the RGB color space, the MSE is known to have a limited correlation with perceptual image fidelity. The HSL color space, on the other hand, is a more perceptually uniform color space i.e. euclidean metrics represent perceptual differences. In our experiments, other loss functions such as the generative loss led to poor performance.

The optimization-based method for image embedding introduced in Section 2.1 was used for the generation of the inputs-output dataset $\mathcal{D}$. The color images were selected at random from the DIV2K dataset [2] that contains 2K resolution high-quality images. These images were randomly resized to train the network at different resolutions; hence the resolution of the images in our data set range from 256×256 to 2000×2000 pixels. For the standard QR codes $\mathbf{Q}$, versions 3, 4, and 5 with random alphanumeric messages were considered. The maximum probability of error mask $\mathbf{P}_{max}$ was designed to be constant with value $P_{\max}$ within the QR code and zero on the finder patterns. The constant value $P_{\max}$ was randomly varied from 0.01 to 1. With respect to the mask $\mathbf{M}$ selecting the central pixels, the ratio $R = d_a/w$ was randomly varied from 0 to 1. The final data set contains $|\mathcal{D}| = 830$ samples for training and 300 for validation. Additionally, data augmentation was applied to the dataset, further increasing the diversity of the samples. To this end, each image in the dataset was randomly cropped to 256×256 and randomly flipped horizontally [11]. The proposed architecture $E_{\mathcal{K}}()$ was then trained by minimizing (3) with the Adam optimizer [13]. A learning rate of 0.001 and betas equal to (0,0.999) were used. The stopping criterion was based on the stabilization of the value of the loss function on the training set and the validation set. The total training time was 2 days. The batch size was set to 4. The network was trained using TensorFlow 2.1 and an NVIDIA GeForce GTX 1080 Ti graphic card.

3 Binary QR code adaptation

In this section, a detailed explanation of the first stage of the proposed framework is included. The first stage exploits the QR code's encoding characteristics (explained in Appendix A) to improve the visual quality of the final QR image. To this end, Cox et al. in [7] showed that a standard QR code $\mathbf{Q}$ can be modified without affecting its decoding rate by utilizing a Gauss-Jordan elimination procedure. Given that the padding bits of the original bitstream, $\mathbf{s_0}$, in the standard QR code, $\mathbf{Q}$, do not carry relevant information, they can be modified without altering the message; however, if any padding bit is modified then the error correction bits should be updated accordingly. Otherwise, the error correction algorithm would fail and, consequently, the decoding would be unsuccessful. The padding bits can be modified and the error correction bits updated through the exclusive-or operation (XOR). Recently, Xu et al. [31] proposed a method to schedule the changeable modules based on the minimization of the contrast between the blended image $\mathbf{I}$ and the QR code black-and-white modules. Although this strategy improves the visual performance, it can generate noise patterns on important parts of the image as shown in Fig. 4. For this reason in this paper, changeable modules are scheduled by considering not only the minimization of the contrast but also prioritizing the edges and the saliency region within the image. Hence, modules in the QR code $\mathbf{Q}$ that correspond to pixels in $\mathbf{I}$ with highest contrast (darkest/lightest) and that

Accepted Manuscript
Version of record at: https://doi.org/10.1007/s11042-022-12357-6

Fig. 4 Binary standard aesthetic QR code generated to reduced only contrast as proposed in [31] (left) and binary baseline QR code $\mathbf{Q}_b$ generated using the proposed binary QR code adaptation method (right). QRnet with equal parameters ($P_{max} = 0.3$ and $R = 0.5$) was used in both cases for QR code image embedding

belong to either an edge or the saliency region are schedule for modification. The image priority mask, $\mathbf{M_I}$, is then computed as $\mathbf{M_I} = \lambda_S \mathbf{S} + \lambda_E \mathbf{E} + \lambda_C \mathbf{C}$ where $\mathbf{S}$ is a saliency map computed using a pyramid feature attention network [33], $\mathbf{E}$ identifies with '1' the edges of $\mathbf{I}$ using a Canny edge detector, and $\mathbf{C}$ is the contrast attention mask [31]. The values of λ_S, λ_E, and λ_C are equal in our experiments. A general scheme of this process is depicted in Fig. 5. The per-module weight $\bar{\mathbf{M}}_\mathbf{I}$ is computed from the priority mask $\mathbf{M_I}$ as $(\bar{\mathbf{M}}_\mathbf{I})_{m,n} = \frac{1}{w \times w} \sum_{i,j \in \mathbf{B}_{m,n}} (\mathbf{M_I})_{i,j}$ where $\mathbf{B}_{m,n}$ represents the (m, n)-th QR code module of size $w \times w$. Modules with higher weight, which are considered to be more important within the image, are then schedule for modification.

Traditionally, the target QR code, $\mathbf{Q}_t$, is obtained from the color image $\mathbf{I}$ through a module-based binarization method as proposed in [30, 31]. This binarization method, however, does not account for the local thresholding procedure performed by popularly used standard QR code decoders as it is in the case of the ZXing [19] library, causing the decoding to fail under unexpected circumstances. As it can be observed in Fig. 6(a), the module-based binarization (top) and the local binarization (bottom) outcome can be quite different

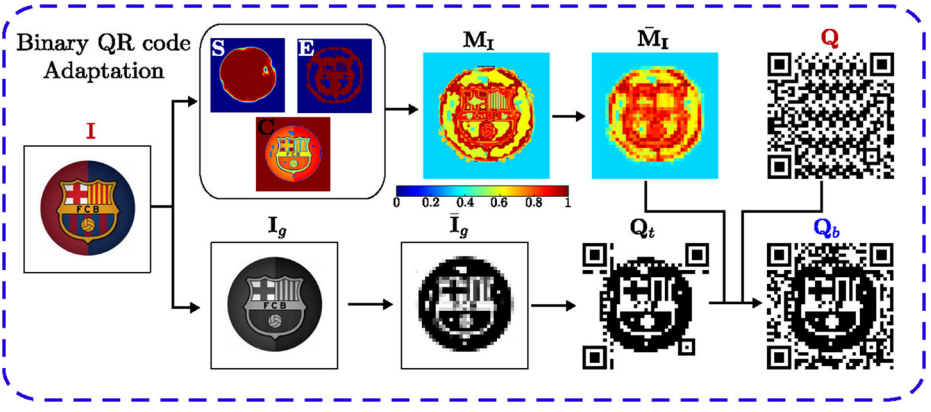

Fig. 5 First stage corresponding to binary QR code adaptation. The inputs are the original color image $\mathbf{I}$ and the traditional black and white QR code $\mathbf{Q}$ and the output is the binary adapted QR code $\mathbf{Q}_b$

Accepted Manuscript
Version of record at: https://doi.org/10.1007/s11042-022-12357-6

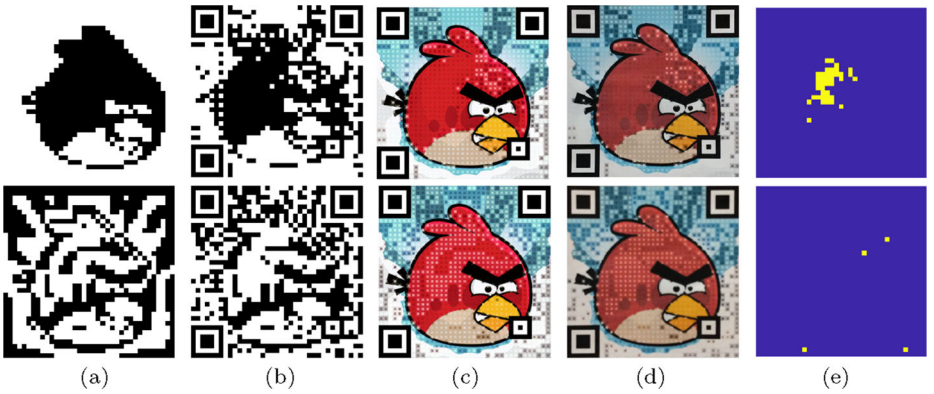

Fig. 6 Comparison with the binary QR code adaptation proposed in [30] (top) and the proposed method (bottom). Each column represents a step in the QR code generation and detection process: (a) $\bar{\mathbf{I}}_g$ obtained after binarization. (b) Binary baseline QR code $\mathbf{Q}_b$. (c) $\mathbf{I}_{out}$ obtained as output of QRnet. (d) $\mathbf{I}_{out}$ captured and rectified using a ZXing-library-based decoder [19]. (e) Miss-classified modules in captured $\mathbf{I}_{out}$ colored in yellow. QRnet with equal parameters ($P_{max} = 0.3$ and $R = 0.5$) was used in both cases for the QR code image embedding

on even regions of the image. While the decoding robustness is not notoriously affected on black and white QR codes (Fig. 6(b)), this effect can detriment the decoding robustness when an image is embedded into a QR code (Fig. 6(c)). In this paper, a per-module local binarization method that is more robust against this problem is proposed. To this end, we first compute a local threshold for each module $\mathbf{B}_{m,n}$ as the mean value of the image intensity $\mathbf{I}_g$ in an overlapping window $\mathbf{W}_{m,n}$ of 5×5 modules centered at the (m, n)-th module in the QR code: $t_{m,n} = \frac{1}{5w \times 5w} \sum_{(i,j) \in \mathbf{W}_{m,n}} (\mathbf{I}_g)_{i,j}$, where $(\mathbf{I}_g)_{i,j}$ is the luminance value of the (i, j)-th pixel. Each pixel in the image $\mathbf{I}_g$ is then binarized based on the corresponding local threshold $t_{m,n}$ to generate $\mathbf{I}_b$. Next, a per-module grayscale value is obtained as: $(\bar{\mathbf{I}}_g)_{m,n} = \sum_{i,j \in \mathbf{B}_{m,n}} (\mathbf{I}_b)_{i,j} (\mathbf{G}_\mathbf{W})_{i,j}$, where $\mathbf{G}_\mathbf{W}$ is a scaled Gaussian distribution. The target QR code image $\mathbf{Q}_t$ is determined from the per-module grayscale image $\bar{\mathbf{I}}_g$ as:

$$(\bar{\mathbf{I}}_b)_{m,n} = \begin{cases} 1 \text{ if } (\bar{\mathbf{I}}_g)_{m,n} \geq 0.5, \\ 0 \text{ if } (\bar{\mathbf{I}}_g)_{m,n} < 0.5. \end{cases} \tag{4}$$

It can be clearly observed in Fig. 6(e)(bottom) that the proposed binarization method decreases the errors when compared to traditional binarization in Fig. 6(e)(top).

4 Experiments and results

In this section, we present a set of experiments that not only compares the speed, visual quality, and decoding robustness of the proposed framework with the baseline method but also with the state-of-the-art.

4.1 Comparison of QRnet with respect the baseline

A series of experiments to compare the proposed QRnet architecture to the baseline method that is the optimization-based method introduced in Section 2.1 are described next.

Accepted Manuscript
Version of record at: https://doi.org/10.1007/s11042-022-12357-6

4.1.1 Using different CNN architectures

As mentioned before, a naive approach to embedding an image into a QR code would be to concatenate all input images and use a fully-convolutional network proposed either for fast image processing [4] or for pixel-wise classification as in [21] with regression loss to produce continuous colors. In Fig. 7, we visually compared the QR image generated by the optimization-based method introduced in Section 2.1 with the embedding generated by QRnet-Pix2pix, QRnet-CAN24, and QRnet-Siamese. For QRnet-Pix2pix, we used the publicly available TensorFlow implementation of Pix2Pix which encoder architecture is based on U-net [11, 21]. The second architecture is based on a multi-scale context aggregation network (CAN) proposed for fast image processing [4] with parameters $d = 9$ and $w = 24$. More details about this method can be found in [4]. For these two architectures, all input images ($\mathbf{I}$, $\mathbf{Q}$, $\mathbf{P}_{\max}$, $\mathbf{M}$, and $\mathbf{H}$) were concatenated and fed to the network to generate the color QR embedding $\mathbf{I}_{out}$. QRnet-Siamese is described in detail in Section 2.2 of this manuscript. A regression loss was used in all the cases since it always provided better performance in our experiments. The same data and training criteria were used for the three architectures. As it can be observed in Fig. 7, QRnet-Siamese better preserves the colors and details of the original image as well as the halftone mask. Further experiments were only considered for QRnet-Siamese since it provided the best image fidelity.

4.1.2 Speed

In Table 1, the run-time of the proposed method is compared with the run-time of the optimization-based method introduced in Section 2.1. The first stage, which can be incorporated into both methods, takes less than 2 seconds in our experiments. Regarding the second stage, the optimization-based method was implemented in MATLAB taking advantage of MATLAB's parallel processing. As observed in Table 1, even though the time increases with the image resolution, the proposed learning-based framework running either on the CPU or the GPU is still much faster. The experiment was conducted on an i7-8700 CPU with 6 Core(s) and an NVIDIA GTX 1080 Ti GPU.

4.1.3 Parameter variation

Two parameters, $P_{\max}$ and R, are left to be controlled by the user. As it can be seen in Fig. 8, a small value of $P_{\max}$ chooses more robustness over visual quality and vice-versa. Bigger values of R lead to a larger center region in each module, improving on robustness but

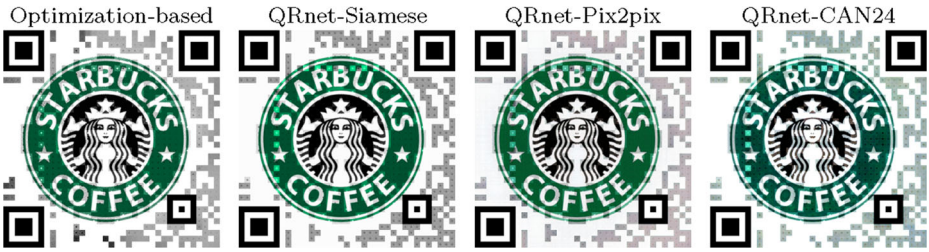

Fig. 7 QRnet using different CNN architectures. The baseline QR code $\mathbf{Q}_b$ is generated using the binary QR code adaptation method proposed in Section 3 of the manuscript. Parameters $P_{max} = 0.25$ and $R = 0.25$ were used for all the cases

Accepted Manuscript
Version of record at: https://doi.org/10.1007/s11042-022-12357-6

Table 1 Image embedding speed for the proposed fast deep learning-based QR codes vs the optimization-based QR codes for varying image resolutions

Image size	Optimization-based	QRnet	
	MATLAB - CPU	CPU	GPU
256×256	1.69×10^3 s	0.41 s	0.14 s
512×512	5.29×10^3 s	0.95 s	0.16 s
1024×1024	1.70×10^4 s	3.6 s	0.21 s

Our method provides results with similar robustness and visual quality but it is orders of magnitude faster

increasing the presence of the structure of traditional QR codes. A good trade-off between these two parameter allows users to adapt the QR Image to more or less challenging scanning circumstances. The fast generation of QR Images allows users to tune these parameters in almost real-time.

4.1.4 Visual quality

In Table 2, the proposed fast learning-based embedding method is compared to the optimization-based method explained in Section 2.1. The peak signal to noise ratio (PSNR), the multiscale structural similarity index (MSSIM) [28], and J (the visual distortion metric introduced in [8]) are used for image quality assessment on a testing set of 60 QR images with varying robustness parameter $P_{\max}$. It can be observed that the visual quality metrics are very similar for both methods, being slightly better for QR images generated by the proposed QRnet. Figure 9 includes a visual comparison of some of the images used in this experiment. In Fig. 9, QRnet high-resolution images are also compared to their low-resolution counterparts, demonstrating the adaptability of the framework to different image sizes.

4.1.5 Decoding robustness

The robustness of the QR images generated by the proposed framework and the optimization-based method was evaluated experimentally. The experiment consisted on manually scanning QR Images with varying robustness parameter $P_{\max}$. Each printed QR image of size 3.15 in $\times$ 3.15 in was scanned 5 times and the scan was considered successful if the

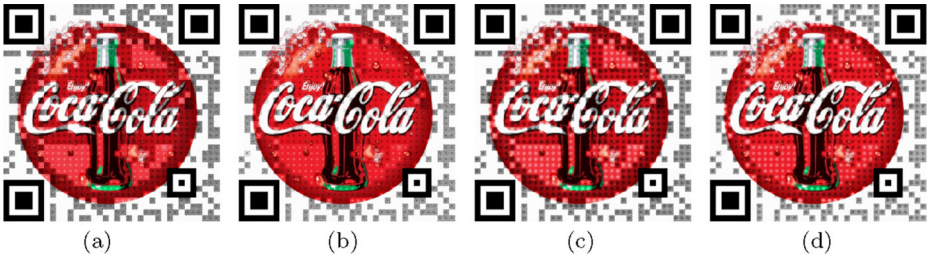

Fig. 8 Visual comparison for the variation of parameters R and P_{max} in QRnet. (a) and (b) have a fixed value of $P_{max} = 0.10$ and $P_{max} = 0.25$, respectively, and a parameter $R = 0.25$. (c) and (d) have a fixed value of $P_{max} = 0.10$ and $P_{max} = 0.25$, respectively, and a parameter $R = 0.5$

Accepted Manuscript
Version of record at: https://doi.org/10.1007/s11042-022-12357-6

Table 2 Average visual quality comparison of QR images generated by QRnet and the optimization-based algorithm

Method	PSNR	MSSIM	J
QRnet	10.6978	0.5498	0.3434
Optimization-based	10.2839	0.5241	0.3523

decoder returned the message within 5 seconds. A total of 54 QR images from the testing set were used in this experiment. As depicted in Fig. 10, the decodability rate decreases as the robustness parameter $P_{\max}$ decreases. Additionally, the robustness of the QRnet images are comparable to the QR Images generated by the optimization-based method explained in Section 2.1.

4.2 Comparison with state-of-the-art

In this section, we compare the visual quality, speed, and robustness of the proposed framework with the recent method proposed in [30]. To this end, from a set of 11 color images,

Fig. 9 The first row represents the ground truth i.e. QR Images generated using the optimization-based method introduced in Section 2.1 of the manuscript. The second row shows the equivalent QR Image obtained by QRnet. The third row includes high resolution images (∼1K) obtained by QRnet. The baseline QR code $\mathbf{Q}_b$ is generated using the binary QR code adaptation method proposed in Section 3 of the manuscript. Parameters $P_{max} = 0.25$ and $R = 0.25$ were used for all the cases

Accepted Manuscript
Version of record at: https://doi.org/10.1007/s11042-022-12357-6

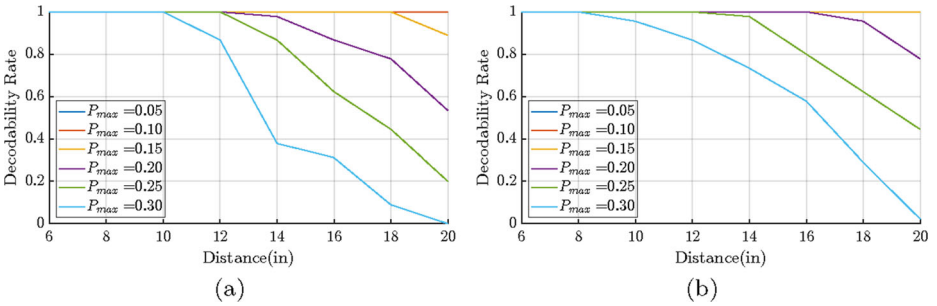

Fig. 10 Decodability rate vs scanning distance for QR images generated by (a) QRnet and (b) the optimization-based method explained in Section 2.1, using different robustness parameters $P_{max} = \{0.05, 0.10, 0.15, 0.20, 0.25, 0.30\}$

we generated different QR images of size 518 × 518 using both methods (QRnet and ART-UP [30]). We tuned the robustness parameters of each method to have similar visual quality and robustness performance. In Table 3, the PSNR, MSSIM, J, SSIM [27], and NCC [5] were used to measure the visual quality of the QR Images generated by each method with respect to the original color image. Additionally, the images generated by both methods are visually compared side-by-side in Fig. 11(a)-(b). To measure the decoding robustness, we printed each QR image with size 3 in × 3 in and manually scanned them 5 times each. Then, we measure the bit error rate (BER) for each scanned QR Image. The average BER for each method is reported in Table 3. With respect to the speed, the run-time of the first stage is very similar for both methods, being less than 2 seconds. For the second stage, as shown in Table 3, QRnet is significantly faster than ART-UP [30]. All the experiments were conducted on an i7-8700 CPU with 6 Core(s) and an NVIDIA GTX 1080 Ti GPU. Even though the method proposed by Zang et al. [32] is very fast, it is not included here for comparison since it lacks of decodability guarantees. These experiments validate that the proposed approach is not only faster but slightly more robust than prior art when similar visual quality metrics are achieved.

Moreover, to show the versatility of our method at generating stylized QR codes as in [31], we used the online tool GoArt [9] to stylize the input image and fed this image into our framework. The output is shown in Fig. 11(c). In order to compare our result with previously proposed stylized aesthetic QR code, we implemented a slightly modified version of the method proposed in [31]. The only difference introduced in our implementation is in the style transfer technique used for stylizing the baseline aesthetic QR code in which we used the online tool GoArt [9] instead of the neural network architecture used in [31]. Additionally for the sake of comparison, we used here the same image used in the experiments in [31]. It can be observed in Fig. 11(d) as well as in the results in [31], that some stroke artifacts can be produced in the style transfer output when the baseline aesthetic QR code

Table 3 Average visual quality, run-time, and bit error rate (BER) comparison of QR images generated by QRnet and ART-UP [30]

Method	PSNR	MSSIM	J	SSIM	NCC	Time(s)	BER(%)
QRnet	12.4161	0.6309	0.2623	0.5565	0.9006	0.2113	1.0282
ART-UP [30]	11.3732	0.5536	0.2662	0.5491	0.9446	372.9558	1.1674

Accepted Manuscript
Version of record at: https://doi.org/10.1007/s11042-022-12357-6

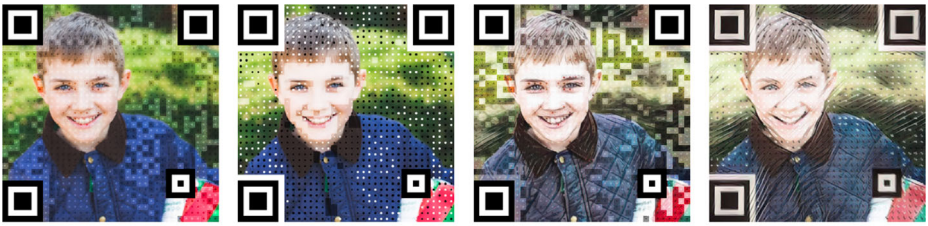

Fig. 11 Comparison of different methods for QR code visual quality enhancement. QR image generated by (a) QRnet, (b) ART-UP [30], (c) GoArt+QRnet, and (d) modified implementation of Stylized Aesthetic QR code [31]

is used as input. These strokes are not generated when the image is first stylized and then feed to the QR code embedding framework as shown in Fig. 11(c). On the other hand, it is worth mentioning that both methods [30, 31] rely on iterative algorithms.

5 Conclusion

In this paper, novel deep learning techniques are leveraged to significantly reduce the time in the generation of visually pleasant QR codes with robustness guarantees. The proposed method is four orders of magnitude faster than the optimization-based method introduced in Section 2.1. The first stage in the proposed framework modifies the binary QR code without affecting the decoding rate and the second stage instantly embeds a QR code into a color image, being decodable by QR code standard applications. The proposed local binarization for the generation of the adapted binary QR code further improves the robustness of the QR images. The method proposed was tested and compared with the baseline method introduced in Section 2.1 and with the state of the art. The decoding robustness and the visual quality of the generated embedded images were verified. The proposed adaptive window for per-module QR code optimization reinforces the decoding robustness of the QR code, taking into account more real scanning conditions. Furthermore, using the analytical relationship between the luminance Y and the component L from the HSL color space in the color optimization considerably reduces the computational cost for the generation of the data. As part of our future work, we will evaluate the robustness of the proposed QR code images under different types of attacks such as the compressive sensing attack.

Appendix A: Encoding procedure

QR codes have a well-defined structure for correcting geometric transformations and achieving quick machine detection and decoding. Some encoders analyze the input message to find the most suitable encoding mode (numeric, alphanumeric, byte or Kanji), version and, error correction capability, while in others, these are chosen by the user. Although the method used to convert the input text into bits varies according to the encoding mode, a general procedure is followed to complete the bit stream $\mathbf{s_0}$. That is, the encoded message is always located in the first m-bits of the bit stream which is padded with zeros to be later broken up into 8-bit codewords.

Accepted Manuscript
Version of record at: https://doi.org/10.1007/s11042-022-12357-6

Due to QR codes structure, the QR code standard requires to add specific padding bits p after the message to completely fill the total capacity of the QR code. Finally, u error correction bits are generated for the message m and padding bits p, generating an n-bits Reed Solomon (RS) code. This bit stream is distributed through the QR code, starting from the bottom right as it is shown in Fig. 12.

Appendix B: Decoding procedure

As illustrated in Fig. 13, after an image of a QR code is captured by the decoder, it is binarized to a black and white image. The QR code standard [24] does not define a specific binarization method, thus, there is not an unique manner to perform this step. However, many widely used QR code decoders apply local thresholding strategies to overcome the effects of uneven lighting conditions. For instance, the widely used open source ZXing [19] library for QR code generation and reading computes local thresholds for non-overlapping blocks $B_{m,n}$ as the mean value of the image intensity in a local window $\mathbf{W}$ as

$$t_{m,n} = \frac{1}{N} \sum_{(i,j)\in \mathbf{W}} Y_{i,j}, \tag{5}$$

where $Y_{i,j}$ is the image luminance value at a pixel location (i, j), $t_{m,n}$ is the local threshold for the block $B_{m,n}$, and N is the number of pixels in the local window $\mathbf{W}$ with size of $W_a \times W_a$ pixels. This binarization method has been used before to design a probability of error model on beautified QR codes [8, 30] and to evaluate their robustness [31].

The decoder then searches for the finder patterns and the alignment pattern regions, which possess a black-white-black-white-black pattern with a ratio of 1:1:3:1:1, and black-white-black pattern with ratio 1:1:1, respectively. Based on the three finder patterns and the closest-to-corner alignment pattern, a perspective transform is performed to compensate for geometric distortions from the scanning process. Once the detected barcode region is converted into a square region, a sampling grid is estimated, using as reference the detected coordinates of the finder and alignment pattern, as well as, the timing patterns. As shown in Fig. 13, the grid samples the central pixel from each binarized module and demodulate these data as '1' or '0' for white and black, respectively. After the error correction algorithm is applied, the message is extracted from the QR code.

Appendix C: Probability of error models

The luminance modification in the original binary structure of the QR code distorts the binarization threshold, increasing the probability of error during the decoding process. To

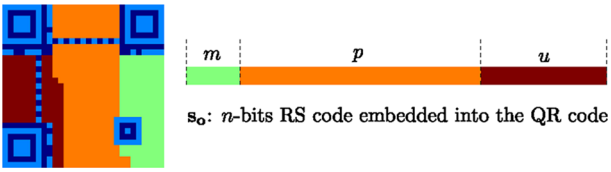

Fig. 12 QR Code Encoding. Each part of the bit stream is shown and highlighted into the QR code

Accepted Manuscript
Version of record at: https://doi.org/10.1007/s11042-022-12357-6

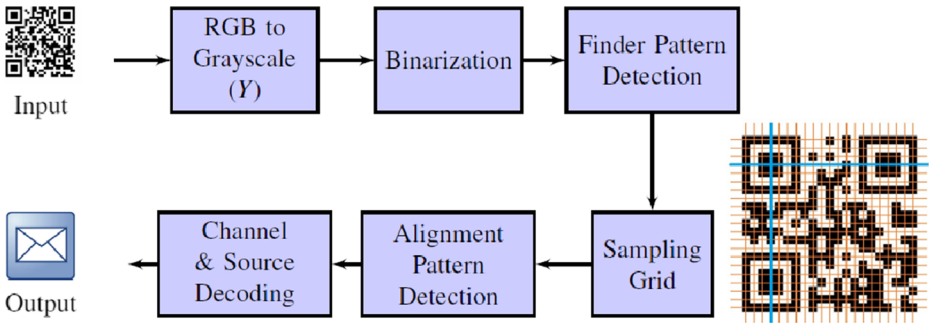

Fig. 13 QR Code Decoding Process. Luminance Y from the scanned image is first determined and then binarize for the detection of the QR code using the finder patterns. Next, only the center of each module is sample to decode the message

consider this error, a normalized version of the probability of error model proposed in [8] is used. In this model, two independent probability models are developed based on the sampling accuracy of the central pixels of the QR code modules. Then, these two models are combined through a probability of sampling error model.

C.1 Probability of binarization error model

This first model considers the probability that any given pixel inside a QR code module is binarized in the wrong direction during the decoding procedure. Then, the probability of error at the detected binary pixel (i,j) is given by

$$\begin{aligned} P_{Berr} &= P(\hat{Q}_{i,j} = 1 | Q_{i,j} = 0) p_0 \\ &\quad + P(\hat{Q}_{i,j} = 0 | Q_{i,j} = 1) p_1, \end{aligned} \tag{6}$$

where $p_0 = P(Q_{i,j} = 0)$ and $p_1 = P(Q_{i,j} = 1)$ are the probabilities of having a black or a white pixel in the QR code $\mathbf{Q}$, respectively. As mentioned before, the calculation of the threshold $t_{m,n}$ depends on the decoder. In this case, a decoder built on ZXing [19] library open source code is considered. Consequently, the threshold depends on the local distribution of luminance values in the image, the distribution of pixels in the QR code, and the values of the parameters for the luminance transformation. Since $\hat{Q}_{i,j}$ represents the binary decision made by the decoder depending on the threshold, the conditional probabilities above can be written as

$$\begin{aligned} P(\hat{Q}_{i,j} = 1 | Q_{i,j} = 0) &= P(Y^{out}_{i,j} > t_{m,n} | Q_{i,j} = 0), \\ P(\hat{Q}_{i,j} = 0 | Q_{i,j} = 1) &= P(Y^{out}_{i,j} < t_{m,n} | Q_{i,j} = 1). \end{aligned} \tag{7}$$

The exact solution to (7) requires the knowledge of the joint distribution of the image and threshold values. The problem can be simplified assuming that the components of the embedding luminance are independent and can be represented in function of the luminance transformation parameters as [8]

$$Y^{out}_{i,j} = [\beta Q_{i,j} + \alpha(1 - Q_{i,j})] I_{p_c(i,j)} + Y_{i,j}(1 - I_{p_c(i,j)}). \tag{8}$$

Accepted Manuscript
Version of record at: https://doi.org/10.1007/s11042-022-12357-6

Using (8) and (5) together with certain reasonable assumptions stated in [8], a model for the probability distribution of the threshold is given by

$$f_t(x) = \sum_{k=0}^{n} f(x - t_k) \binom{n}{k} p_1^k p_0^{n-k}, \tag{9}$$

where $n = \lfloor p_c N \rfloor$ is the total number of pixels selected for modification in the window $\mathbf{W}$, $t_k = \lfloor \frac{k\beta}{N} + \frac{(n-k)}{N}\alpha \rfloor$ are all the possible average contributions of n modified pixels to the total threshold, and f is the convolution of $f_Y(x)$ and $f_\eta(x)$. $f_Y(x)$ represents the probability distribution of the unmodified pixels in the image, modeled as independent Gaussian random variables that depend on the window $\mathbf{W}$, and $f_\eta(x)$ accounts for the noise at the detector, modeled also as a Gaussian distribution. Thus, conditioning on the set of modified pixels, the conditional probabilities in (7) can be determined as follows

$$\begin{aligned} P(Y_{i,j}^{out} > t_{m,n} | Q_{i,j} = 0) &= P(\alpha > t_{m,n}) p_c \\ &\quad + P(Y_{i,j} > t_{m,n})(1 - p_c), \end{aligned} \tag{10}$$

$$\begin{aligned} P(Y_{i,j}^{out} < t_{m,n} | Q_{i,j} = 1) &= P(\beta < t_{m,n}) p_c \\ &\quad + P(Y_{i,j} < t_{m,n})(1 - p_c), \end{aligned} \tag{11}$$

where

$$P(x > t_{m,n}) = \frac{\int_0^x f_t(t)dt}{\int_0^1 f_t(t)dt}, \tag{12}$$

$$P(x < t_{m,n}) = \frac{\int_x^1 f_t(t)dt}{\int_0^1 f_t(t)dt}. \tag{13}$$

Substituting (10) and (11) in (6), we finally estimate the probability of binarization error P_{Berr}.

C.2 Probability of detection error

As in [8], assuming the central pixels in the QR code modules are precisely sampled, this model considers the probability of a given central pixel to be detected in the wrong direction during the decoding procedure. The probability of detection error per each QR code module is defined as

$$P_{Derr} = P(\alpha_c > t) p_0 + P(\beta_c < t) p_1. \tag{14}$$

In this case, the binarization threshold at the decoder can be decomposed as $t = \mu + b + \eta$, considering not only the average contribution to the threshold from the modified pixels μ computed as

$$\mu = \frac{\alpha n_\alpha + \beta n_\beta + \alpha_c n_{\alpha_c} + \beta_c n_{\beta_c}}{N}, \tag{15}$$

but also from the non-modified pixels b in the window, as well as the noise at the decoder η. In (15), n_α, n_β, n_{α_c} y n_{β_c} are the number of pixels per each modified luminance level and N is the total number of pixels in the window $\mathbf{W}$.

Non-modified pixels are modeled as Gaussian random variables as before, with mean $\mu_b = E[Y_{i,j}](1 - p_c)$ and variance $\sigma_b^2 = \frac{Var[Y_{i,j}]}{N}(1 - p_c)$. As a result, the probability distribution of the threshold t is Gaussian with mean $\mu_t = \mu + \mu_b$ and variance $\sigma_t^2 = \sigma_b^2 + \sigma_\eta^2$ where σ_η^2 is the variance of the Gaussian noise at the decoder. Substituting this probability distribution of the threshold in the normalized probabilities defined in (12), we can estimate the probability of detection error P_{Derr} in (14).

Accepted Manuscript
Version of record at: https://doi.org/10.1007/s11042-022-12357-6

C.3 Global probability of error

A probability of error sampling model is considered to combine the two independent stages in a global probability of error [8]. According to the decoding procedure, ideally, central pixels are more likely to be sampled. Thus, it is reasonable to model the sampling process by means of a Gaussian distribution around the center regions with $\sigma = W_a/6$. The probability of sampling out of the center region p_s can be determined by integrating outside the central region $d_a \times d_a$. In this paper, this probability is also normalized. Once the probability of sampling error p_s is known, the global probability of error is determined by

$$P_{err} = P_{Berr}\, p_s + P_{Derr}(1 - p_s). \tag{16}$$

Acknowledgements We would like to thank Gonzalo Garateguy and Jose Luis Paredes for their collaboration in this project.

Funding This work was supported in part by Graphiclead LLC.

Availability of Data and Material The datasets used for the experiments in this manuscript are available from the corresponding author on reasonable request. Additionally, the method used for the generation of the dataset is described in the manuscript.

Code Availability The manuscript focuses on the development of the underlying method. However, the codes used for the experiments in this manuscript are available from the corresponding author on reasonable request.

Declarations The logo images used in this paper were not endorsed by the trademark owners and they were used here only to evaluate the capabilities of our method. Coca Cola, Starbucks, FCB, Golden State Warriors, and Juan Valdez Logos are a trademark of the Coca Cola Company, Starbucks U.S. Brands, LLC, Futbol Club Barcelona, Golden State Warriors, LLC, and National Federation of Coffee Growers of Colombia, respectively. They do not sponsor, authorize or endorse the images in this paper. All Rights Reserved.

Conflict of Interests Karelia Pena-Pena and Andrew Arce declare that they have no competing interests. Gonzalo Arce and Daniel Lau are part of Graphiclead LLC. Gonzalo Arce received support from Graphiclead LLC.

References

1. 7 Proven ways ecommerce uses QR codes to bring users back. beaconstac.com/2019/10/how-ecommerce-is-using-qr-codes-in-interesting-ways-to-bring-back-users/. Accessed 20 March 2021
2. Agustsson E, Timofte R (2017) Ntire 2017 challenge on single image super-resolution: dataset and study. In: IEEE conf comput vis pattern recognit (CVPR) workshops
3. Baharav Z, Kakarala R (2013) Visually significant QR codes: image blending and statistical analysis. In: 2013 IEEE int. conf. on multimedia and expo (ICME). IEEE, pp 1–6
4. Chen O, Xu J, Koltun V (2017) Fast image processing with fully-convolutional networks. In: Proceedings of the IEEE int. conf. on computer vision, pp 2497–2506
5. Chidambaram N, Raj P, Thenmozhi K, Amirtharajan R (2021) A new method for producing 320-bit modified hash towards tamper detection and restoration in colour images. Multimed Tools Appl 80(15):23359–23375
6. Chu HK, Chang CS, Lee RR, Mitra NJ (2013) Halftone QR codes. ACM Trans Graph (TOG) 32(6):1–8
7. Cox R (2012) Qart codes. research.swtch.com/qart. Accessed 30 June 2018
8. Garateguy GJ, Arce GR, Lau DL, Villarreal OP (2014) Qr images: optimized image embedding in qr codes. IEEE Trans Image Process 23(7):2842–2853
9. GoArt - AI photo effects. fotor.com/features/goart.ht ml. Accessed 15 March 2021

10. Huang G, Liu Z, Maaten LVD, Weinberger KQ (2017) Densely connected convolutional networks. In: Proc. IEEE comput. soc. conf. comput. vis. pattern recognit, pp 4700–4708
11. Isola P, Zhu J, Zhou T, Efros AA (2017) Image-to-image translation with conditional adversarial networks. In: Proc. IEEE comput. soc. conf. comput. vis. pattern recognit, pp 1125–1134
12. Johnson J, Alahi A, Fei-Fei L (2016) Perceptual losses for real-time style transfer and super-resolution. In: European conference on computer vision. Springer, pp 694–711
13. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv:1412.6980
14. Koch G, Zemel R, Salakhutdinov R (2015) Siamese neural networks for one-shot image recognition. In: ICML deep learning workshop, vol 2. Lille
15. Lau DL, Arce GR (2018) Modern digital halftoning, vol 1. CRC Press
16. Lau DL, Ulichney R, Arce GR (2003) Blue and green noise halftoning models. IEEE Signal Proc Mag 20(4):28–38
17. Lin SS, Hu MC, Lee CH, Lee TY (2015) Efficient QR code beautification with high quality visual content. IEEE Trans Multimed 17(9):1515–1524
18. Mullen C (2020) The pandemic has given QR codes a shot in the arm. Will it last post-Covid? bizjournals.com/bizwom en/news/latest-news/2020. Accessed 15 Nov 2020
19. Owen S (2016) Multi-format 1d/2d barcode image processing library with clients for android, java and c++ (zxing). opensource.google.com/projects/zxing. Accessed 30 May 2018
20. Rodriguez JB, Arce GR, Lau DL (2008) Blue-noise multitone dithering. IEEE Trans Image Process 17(8):1368–1382
21. Ronneberger O, Fischer P, Brox T (2015) U-net: convolutional networks for biomedical image segmentation. In: Int. conf. on med image comput comput assist interv. Springer, pp 234–241
22. Samretwit D, Wakahara T (2011) Measurement of reading characteristics of multiplexed image in QR code. In: 2011 third int. conf. on intelligent networking and collaborative systems. IEEE, pp 552–557
23. Schonfeld E, Schiele B, Khoreva A (2020) A u-net based discriminator for generative adversarial networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 8207–8216
24. Technology I (2006) Automatic identification and data capture techniques - bar code symbology QR code
25. The new marketing opportunity for the e-commerce industry is QR codes. qr-code-generator.com/blog/e-commerce-industry-qr-codes/. Accessed 20 March 2021
26. Visualead company free visual QR code generator. visualead.com/. Accessed 30 May 2018
27. Wang Z, Bovik AC, Sheikh HR, Simoncelli EP (2004) Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process 13(4):600–612
28. Wang Z, Simoncelli EP, Bovik AC (2003) Multiscale structural similarity for image quality assessment. In: The thrity-seventh asilomar conference on signals, systems & computers, vol 2, pp 1398–1402
29. Wengrowski E, Dana K (2019) Light field messaging with deep photographic steganography. In: Proc. IEEE comput. soc. conf. comput. vis. pattern recognit, pp 1515–1524
30. Xu M, Li Q, Niu J, Liu X, Xu W, Lv P, Zhou B (2018) ART-UP: a novel method for generating scanning-robust aesthetic QR codes. arXiv:1803.02280 [cs.MM]
31. Xu M, Su H, Li Y, Li X, Liao J, Niu J, Lv P, Zhou B (2019) Stylized aesthetic QR code. IEEE Trans Multimed 21(8):1960–1970
32. Zhang Y, Deng S, Liu Z, Wang Y (2015) Aesthetic QR codes based on two-stage image blending. In: Int. conf. on multimedia modeling. Springer, pp 183–194
33. Zhao T, Wu X (2019) Pyramid feature attention network for saliency detection. In: IEEE conf comput vis pattern recognit (CVPR