Available online at www.isr-publications.com/jmcs
J. Math. Computer Sci., 18 (2018), 306–313
Research Article

Online: ISSN 2008-949x

Journal of Mathematics and Computer Science

Journal Homepage: www.tjmcs.com - www.isr-publications.com/jmcs

Construction of multi-layered QR codes utilizing partitions of positive integers

Passawan Noppakaew*, Sukanya Khomkuth, Sureepat Sriwilas

Department of Mathematics, Faculty of Science, Silpakorn University, Nakhon Pathom, 73000, Thailand.

Abstract

Multi-layered QR (MLQR) codes are created by superimposing many black and white QR codes, all of which are assigned their white areas with the colors in RGB color space. These colors must be different enough to enable distinguishing each layer of a MLQR code. This makes a MLQR code be able to hold more data than a common QR code. In this work, the procedures for generating and un-layering MLQR codes were proposed and the according graphical user interfaces (GUIs) were created with MATLAB. They use the property of a partition of the number 255, constructed by the geometric sequence $\{2^{n-1}\}_{n\in\mathbb{N}'}$ to compute the collection of suitable colors for assigning to black and white QR codes in the generating process. Our developed procedures can promote better intercommunication between human and computers, therefore ensure easier computer programming and being more flexible in the number of layers of created MLQR codes. We found that the developed GUIs could work accurately up to 15 layers because more QR code layers require the use of more colors, which diminish an ability to clearly distinguish the color of each QR code layer.

Keywords: Quick response codes, 3-dimensional codes, geometric sequences, partitions of positive integers.

2010 MSC: 90C90, 68U10.

1. Introduction

A universal product code (UPC) or also known as a barcode, developed by IBM in 1970, is widely used by the industry because of its benefits in cutting costs, saving time and eliminating human errors. A usual barcode, referred to as a 1-dimensional (1D) code, represents the data relating to the object to which it is attached by the series of parallel bars and spaces with different widths. A barcode can only store upto 30 numbers of information. As the economic grows highly, the limitation of a barcode becomes apparent.

In 1994, Denso Wave Incorporated developed a 2-dimensional (2D) code, called a Quick Response (QR) code, which can stores upto 7,089 numbers. Beyond numbers, a QR code can hold a variety types of data such as texts, hyperlinks, telephone numbers, emails, etc.. However, the capacity of a QR code is

*Corresponding author
Email addresses: noppakaew_p@silpakorn.edu (Passawan Noppakaew), goi_sukanya@hotmail.com (Sukanya Khomkuth), sureepat22@gmail.com (Sureepat Sriwilas)

doi: 10.22436/jmcs.018.03.06

Received: 2016-08-19 *Revised:* 2017-12-02 *Accepted:* 2017-12-03

still not enough in the high economic growth period. In recent years, a variety of efforts have been made in creating higher-dimensional codes to increase the capacity of the codes such as SpectraCode, Mobile Multi-Colored Composite (MMCC) [3], Microsoft's High Capacity Color Barcode (HCCB) [2], and Quick Layered Response code (QLR) [1].

This work proposes a new method to create a colored code, called a Multi-Layered QR (MLQR) codes, which is QR codes superimposed in different color channels. A resulting MLQR code thus has higher capacity compared with the black and white codes of the same size. The color system we use in this work is the RGB color space. Using MATLAB to take advantages of the image processing, we design two graphical user interfaces (GUIs). One is for layering usual QR codes to obtain a MLQR code and the another is for un-layering a MLQR code to obtain all black and white QR codes that it is composed of, so that all data can be decoded by any QR code reader later.

Each pixel of a picture being processed in MATLAB is represented by an array shown as $[R, G, B]$ with the first, second, and third component representing the intensity in Red, Green, and Blue, respectively; the intensity varies from 0 (the minimum) to 255 (the maximum). Hence one may consider that MATLAB recognizes images in three separate layers which are red, green, and blue layers. In 2013, Nessen [4] gave an idea to encode three and six QR codes into a new code by using photo editing software to color the white areas of the QR codes and superimpose them by using MATLAB. The key point in Nessen's work is choosing a collection of suitable colors for assigning to the white areas of all three and six QR codes. By choosing an unsuitable collection of colors, the un-layering precess cannot be done. To generalize Nessen's work, this motivates us to propose an algorithm to compute a collection of suitable colors we need in order to construct an n-layered QR code, where n is a positive integer.

2. Generating MLQR codes procedure

Generating MLQR codes is done by layering usual QR codes of equal size whose the white areas are assigned by different colors in RGB color system. This can be done as follows.

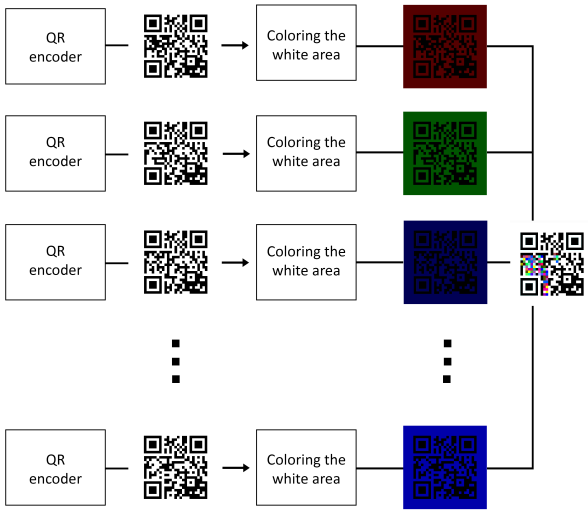

Figure 1: The diagram illustrating the procedure for encoding a MLQR code.

2.1. *Loading black and white QR codes*

The data to encode in a MLQR code is separated to be stored in black and white QR codes with equal size. Then all these codes are loaded in MATLAB by using the function "imread". As the codes are in black and white, when they are imported in MATLAB, each of them is recognized as a binary image and represented by a square matrix consisting of the numbers 1 (white) and 0 (black).

2.2. Computing a collection of suitable colors

A collection of suitable colors for assigning to the white areas of QR codes must have the following properties:

1. there is no color in the collection which is the combination of two or more colors in the collection;
2. any two colors in the collection must be different enough so that they can be distinguished correctly.

This gives us an idea of a partition of a positive integer which is defined as follows.

Definition 2.1. A set $\mathcal{P}$ of positive integers is a *partition* of a positive integer m if it has two following properties

1. $\sum_{x\in\mathcal{P}} x = m$;
2. for any subset $\mathcal{S}$ of $\mathcal{P}$ such that $|\mathcal{S}| \geqslant 2$, $\sum_{x\in\mathcal{S}} x \notin \mathcal{P}$.

The cardinality of $\mathcal{P}$ is called the *length* of the partition $\mathcal{P}$.

The algorithm, to construct a partition with length l of a number m, can be done by using a geometric sequence $\left\{a^{n-1}\right\}_{n\in\mathbb{N}}$, where a is a positive integer greater than or equal to 2 and $\frac{m(a-1)}{a^l-1}$ is greater than or equal to 1, with the following procedure

1. compute $s = \left\lfloor \frac{m(a-1)}{a^l-1} \right\rfloor$ which is the greatest integer less than of equal to $\frac{m(a-1)}{a^l-1}$;
2. the set $\mathcal{P} = \left\{ s, sa, sa^2, sa^3, \ldots, sa^{l-2}, m - \frac{s\left(a^{l-1}-1\right)}{a-1} \right\}$ is a partition with length l of m as required.

Theorem 2.2. *The proposed algorithm gives a partition of* m *with length* l.

Proof. By using above notions, it is obvious that $\sum_{x\in\mathcal{P}} x = m$, whence the first property of a partition is satisfied. Next, we will show that $\mathcal{P}$ also satisfies the second property of a partition. As

$$1 + a + a^2 + \cdots + a^{k-1} = \frac{a^k - 1}{a-1} < a^k$$

for any positive integer k, and $m - \frac{s\left(a^{l-1}-1\right)}{a-1} \geqslant sa^{l-1}$, one can conclude that, for any subset $\mathcal{S}$ of $\mathcal{P}$ such that $|\mathcal{S}| \geqslant 2$, $\sum_{x\in\mathcal{S}} x \notin \mathcal{P}$. Therefore $\mathcal{P}$ is a partition of m with length l as required. □

In order to compute a collection of suitable colors, the proposed algorithm is used to construct a partition of the number 255, which is the highest intensity for R, G, and B channels in the RGB color space and the geometric sequence used in this work is $\left\{2^{n-1}\right\}_{n\in\mathbb{N}}$ as it gives the most difference between elements in the partition. Therefore, a collection of suitable colors can be archived by the following procedure.

1. Let k be the number of all black and white QR codes brought for layering. Then the length of the partition of 255 constructed by the proposed algorithm is $\left\lceil \frac{k}{3} \right\rceil$ which is the least integer greater than or equal to $\frac{k}{3}$.

2. Compute the partition $\mathcal{P}$ with length $\left\lceil \frac{k}{3} \right\rceil$ of 255 induced by the geometric sequence $\left\{2^{n-1}\right\}_{n\in\mathbb{N}}$.

3. Any k-element subset of the set

$$\{(x,0,0),(0,x,0),(0,0,x)\,|x \in \mathcal{P}\}$$

forms a collection of suitable colors in the RGB color space.

Example 2.3. When the number of black and white QR code brought for layering is 7, the set of the first 7 largest elements in $\{(x,0,0),(0,x,0),(0,0,x)\,|x \in \{36,72,147\}\}$ when equipped with the lexicographic order is a collection of suitable colors for assigning to the white areas of the QR codes.

2.3. Layering the QR codes

Any RGB image can be realized by a 3D-matrix M of dimension $a \times b \times 3$ in MATLAB where $a \times b$ is the size of the image. The first layer $(M(:,:,1))$, the second layer $(M(:,:,2))$, and the third layer $(M(:,:,3))$ in of M correspond to the red, green, and blue channel of the image, respectively. Thus all imported black and white QR codes are separated into 3 groups according to the position of non-zero components of the colors that will be assigned to the codes.

Example 2.4. As in Example 2.3, the codes will be separated into 3 groups. One consists of the codes that will be colored by the colors in the set $\{(36,0,0),(72,0,0),(147,0,0)\}$, one consists of the codes that will be colored by the colors in the set $\{(0,36,0),(0,72,0),(0,147,0)\}$, and one consists of the code that will be colored by the color $(36,0,0)$.

These 3 groups correspond to the red, green, and blue channels. By using MATLAB, the matrix representing a black and white QR code, which consists of 0 and 1 (as explained in Section 2.1), will be multiplied by the number in the non-zero component of the assigned color. To superimposed all QR codes, after all matrices are multiplied by the assigned scalar, the summation is taken over the matrices in each group; this yields 3 matrices one of which corresponds to one channel of the primary colors. Finally, these resulting 3 matrices are concatenated to get a 3-D matrix representing the constructed MLQR code by using the command $\text{cat}(3,\ldots,\ldots,\ldots)$ in MATLAB.

Example 2.5. As in Examples 2.3 and 2.4, a seven-layered QR code constructed by this procedure is shown in Figure 2.

Figure 2: A seven-layered QR code.

However, the background's color of a MLQR code with the number of layers not a multiple of 3 is not white. If a MLQR code is printed out, this will affect ink consumption. To save the resource, the background of a MLQR code may be necessary to be white. This will happen when the number of black and white QR codes brought for layering is a multiple of 3. Therefore, the procedure to layer black and white QR codes may need dummy codes. The picture of the dummy code is show in Figure 3. The dummy code contains no data, and so it is almost blank. But in a MLQR code, we still want to reserve

the color of the position detection pattern because it will help us detecting the position of a MLQR code. Therefore the position detection pattern still appears on the dummy code. A number of the dummy code will be added to the set of codes brought for layering so that the total number is the least integer which is a multiple of 3.

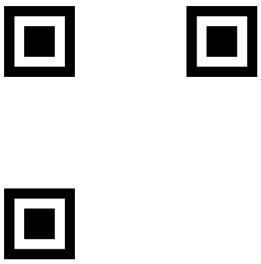

Figure 3: The dummy code.

Example 2.6. The nine-layered QR code, constructed from the seven black and white QR codes as in Example 2.5 and two dummy codes, is shown in Figure 4.

Figure 4: The nine-layered QR code modified from the MLQR code in Figure 2 by adding two dummy codes.

2.4. The reference bar

The reference bar is a bar appearing on the top left of a generated MLQR code. It consists of square blocks of colors $(0, 255, 0)$, $(0, 0, 255)$, and $(255, 0, 0)$, alternately. The number of the square blocks informs the number of layers of the MLQR code that the reference bar adheres to.

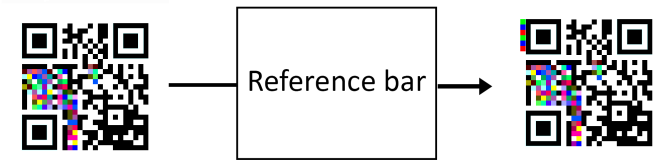

Figure 5: The reference bar is added after a MLQR code is generated. (This figure is an example of a six-layered QR code.)

The reference bar not only informs the user about the number of layers of a MLQR code but also can be used as a reference when the color correction is needed. The correction is needed when a MLQR code is discolored; this may be caused by printing or taking pictures of MLQR codes from multiple devices. The color appearing in the reference bar may be used to help correcting the color of a MLQR code image more accurately.

Figure 6 is a sample MATLAB GUI developed for generating MLQR codes.

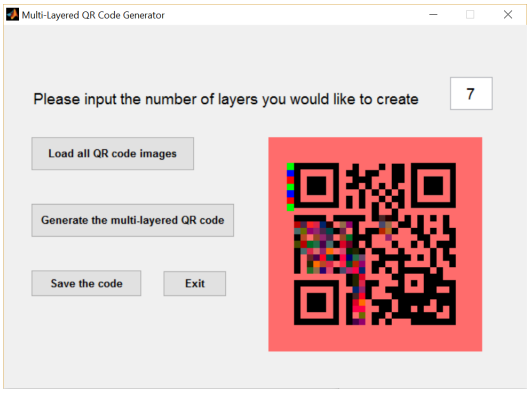

Figure 6: A sample frame of a seven-layer QR code generation.

3. Un-layering MLQR codes procedure

As MLQR codes are generated by superimposing black and white QR codes, in order to make them readable by usual QR code reader, their layers will be separated apart. This can be done as follows.

3.1. *Detecting MLQR codes*

An image of a MLQR code may contain the background surrounding the code. This background is unwanted in the un-layering procedure. In order to remove it, the position of a MLQR code in the image must be detected. This can be done by using the advantages of MATLAB in image processing to detect its position detection pattern. The position detection pattern are specially designed as the sequences of black (b) and white (w) square modules in vertical (v), horizontal (h), and diagonal (d) lines having the same ratio which is w:b:w:bbb:w:b:w as in Figure 7.

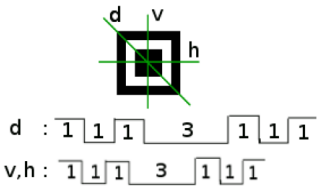

Figure 7: Black and white module proportions in the position detection pattern for v, h, and d lines.

Once the position detection pattern is known, the image will be cropped so that there is only a MLQR code appearing in the cropped image. Figure 8 shows an example of detecting a MLQR code in an image.

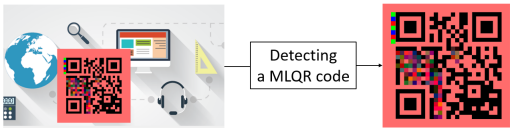

Figure 8: A sample of detecting a seven-layer QR code in an image.

3.2. *Deleting the reference bars*

As mentioned in Section 2.4, the reference bar informs the number of layers of the MLQR code. This number will be read and input by the user. After the number is recorded, the reference bar is deleted from the MLQR code. The position of the reference bar can be detected by detecting the top left of the

position detection pattern first, and then the reference bar is on the left of that position with the width equal to the length of sides of the square modules in the MLQR code.

The number of layers of the MLQR code will be used to calculate as in Section 2.2 to see what is the collection of colors used in generating procedure of the MLQR code.

3.3. Un-layering MLQR codes

As the colors used in generating a MLQR code are of the forms $(x,0,0)$ or $(0,x,0)$ or $(0,0,x)$ where x is a positive integer less than or equal to 255, by considered each color channel of the MLQR code, one can un-layering it by comparing with the value in each channel with the elements in the partition of 255 that we obtain from Section 3.2.

To recover all black and white QR codes which are colored by red tone colors, i.e., the colors represented by $(x,0,0)$, where x is a positive integer less than or equal to 255, in generating procedure, consider each value in the matrix representing the red channel of the image of the MLQR code and see that it is the combination of what numbers in the partition of 255, then one can decide that the pixel corresponding to that position of that value is black or white in those QR codes. One can do the same process to recover all black and white QR codes which are colored by green and blue tone colors. Finally, all black and white QR codes brought to generate the MLQR code are recovered.

Example 3.1. Once the RGB image of a seven-layered QR code is read by MATLAB, it will be represented by a 3D-matrix, lets call it M. The red, green, and blue channel of the MLQR code can be obtained by the commands $M(:,:,1)$, $M(:,:,2)$, and $M(:,:,3)$, respectively, all of which are 1D-matrix. The collection of colors used for constructing the MLQR code can be known by computing from the algorithm in Section 2.2. The white areas of the QR codes brought to construct the MLQR code are colored by $(36,0,0)$, $(72,0,0)$, $(147,0,0)$, $(0,36,0)$, $(0,72,0)$, $(0,147,0)$, and $(0,0,36)$ in generating process.

The red channel of the image of the MLQR code may be represented by the matrix

$$\begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \iddots \\ \cdots & 0 & 36 & 108 & \cdots \\ \cdots & 219 & 147 & 72 & \cdots \\ \cdots & 183 & 255 & 36 & \cdots \\ \iddots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

Consider the position of the matrix with the value 0, it is not a combination of anything in the partition $\{36,72,147\}$ of 255. Thus the pixels corresponding to that position of the usual QR codes that are colored by $(36,0,0)$, $(72,0,0)$, and $(147,0,0)$ in the generating procedure must be black.

Consider the position of the matrix with the value 36, one can see that 36 is just the number 36 in the partition $\{36,72,147\}$ of 255. Thus the pixels corresponding to that position of the usual QR codes that are colored by $(36,0,0)$ in the generating procedure must be white and of the rest (the ones which are colored by $(72,0,0)$ and $(147,0,0)$) must be black.

Consider the position of the matrix with the value 108, one can see that 108 is the combination of 36 and 72 in the partition $\{36,72,147\}$ of 255. Thus the pixels corresponding to that position of the usual QR codes that are colored by $(36,0,0)$ and $(72,0,0)$ in the generating procedure must be white and of the rest (the one which is colored by $(147,0,0)$) must be black.

By doing this to all the position of the red channel matrix, all black and white QR codes which are colored by red tone colors in generating process are recovered. And by also doing this to blue and green channel matrix, all black and white QR codes used in generating the MLQR code are recover and all data encoded in them can be decoded by usual QR code reader.

Figure 9 is a sample of MATLAB GUI developed for un-layering MLQR codes. All black and white QR codes obtained from the un-layering procedure appear in the list box. When each QR code in the list box is clicked, its image will be shown and can be saved to used later on.

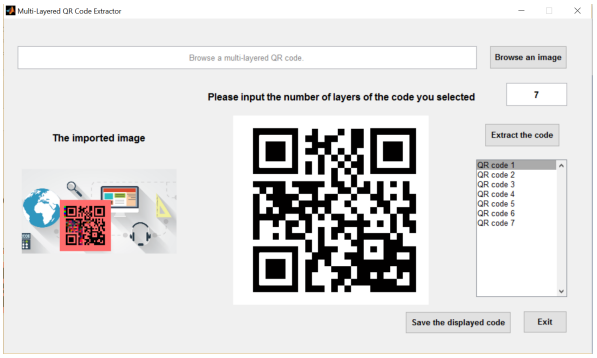

Figure 9: A sample frame of un-layering a seven-layer QR code.

4. Conclusion

By layering QR codes, the capacity of a MLQR codes are greatly increased when compared with the capacity of a black and white QR code. Moreover, a MLQR code can hold many images, pdf files, or websites depending on its number of layers, while a black and white QR code can hold just only on image, one pdf file or one website. Some advantages of MLQR codes have already mentioned in [4].

The restriction of MLQR codes are their discoloration caused by printing and photographic conditions such as camera, shadow, light. Color correction may be done by using the colors in the reference bar as reference colors. To do so, the reference bar may be adjusted to help correcting the color of a MLQR code image being more accurate. However, MLQR codes work well in online systems which do not cause a lot of discoloration in images such as Email, Facebook or Line applications because the intensity of each color channel will be varied less than 3 A.U. when the image information is transferred via these application.

This work introduces procedures that can generate and un-layer MLQR codes. The proposed procedures use the property of a partition of a positive integer (which is 255 in this work). This enables us not only to easily communicate with computer so that it can be programmed to flexibly generate a MLQR code with any layers but also to know the maximum number of layers that still makes the GUIs being effective. Generating more layers of a MLQR code requires use of smaller interval of intensity for each color channel that is assigned to each layer. This effects the accuracy of the un-layering process of the image of a MLQR code. From computing, we found that the GUIs work accurately up to 15 layers. The intensities of each color channel chosen for layering are differ by more than 3.

Future work

This research can be implemented to be able to used in Android devices as a generating and reading platform. This can be built on top of the existing ZXing (”Zebra Crossing”) library that most QR code readers use.

Moreover, the number of layers of a MLQR code that still makes the layering and un-layering procedures being effective may be increased by using different color space.

References

[1] T. Dean, C. Dunn, *Quick Layered Response (QLR) Codes*, Informally published manuscript, Electrical Engineering, Stanford University, Stanford, (2012). 1

[2] G. Jancke, *High Capacity Color Barcodes (HCCB)*, Available Online, (2007). 1

[3] H. Kato, *Mobile Multi-Color Composite: A Novel Color 2D-Barcode for True Ubiquitous Computing*, Doctoral thesis, School of Computer and Information Science, Edith Cowan University, (2009). 1

[4] C. Nessen, *Encoding Multi-layered Data into QR Codes for Increased Capacity and Security*, Research Experience for Undergraduates Summer, South Dakota School of Mines and Technology, (2013). 1, 4