Encoding Multi-layered Data into QR Codes for Increased Capacity and Security

Prepared by:
Carly Nesson

Faculty Advisors:
Dr. Grant Crawford
REU Site Director, Department of Materials and Metallurgical Engineering

Dr. Randy Hoover
Assistant Professor, Department of Electrical and Computer Engineering

Dr. Alfred Boysen
Professor, Department of Humanities

Program Information:
National Science Foundation
Grant NSF #EEC-1263343

Research Experience for Undergraduates
Summer 2013

South Dakota School of Mines and Technology
501 E Saint Joseph Street
Rapid City, SD 57701

Table of Contents

Abstract

Quick Response (QR) codes, currently used in marketing, warehouse management, product tracking, and other applications, is currently comprised of only black and white modules. The goal of this paper is to discuss the implementation of Colored QR codes in order to increase data capacity and security yet still maintain reasonably-sized codes. This objective will be completed by coloring various QR codes and using MATLAB to take advantage of the various color channels present. By layering colored QR codes, a layered code will present readers with enough information to obtain the original codes involved. This paper discusses methods for layering three base colors – Red, Green, and Blue – as well as six base colors – High Red, Low Red, High Green, Low Green, High Blue, and Low Blue.

It was found that layering colored QR codes effectively increased the data capacity three-fold and six-fold for three base colors and six base colors, respectively. The layering and un-layering process was fairly simple with the use of some basic MATLAB commands.

Introduction

Background

QR Codes. A Quick Response (QR) code is a two-dimensional barcode that was developed by Denso Wave Incorporated in 1994 for the automotive industry [5]. Today, it has a variety of uses in marketing, warehouse management, product tracking, etc.

Much like a traditional one-dimensional barcode, a QR code holds encrypted information in the form of one of four types of information: byte/binary, numeric, alphanumeric, and Kanji – a series of Japanese characters [3]. QR codes come in a variety of sizes, all of which are perfect squares. The smallest – known as version 1 – measures 21 x 21 elements, or modules, and each version expands by 4 elements to the right and 4 elements to the bottom. For example, version 2 measures 25 x 25 modules, version 3 is 29 x 29 elements, etc., until the highest-developed version, which is version 40. Version 40 measures 177 x 177 modules [2].

One of the unique characteristics of QR codes is the error-correction capability. Using a mathematical process known as Reed-Solomon error correction, users are able to scan QR codes effectively even if they are dirty or damaged. In each code, there is information about the error correction level – low, medium, quartile, or high [2].

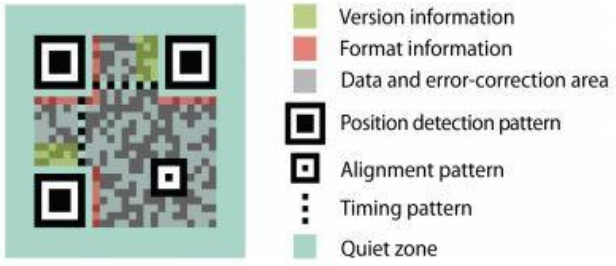

***Figure 1**. Standard characteristics present in every QR code [2]*

Another unique quality QR codes possess is the application of an encoding mask over the content. Its presence ensures readability by eliminating large chunks of darkened modules or

filling spaces with only white elements. Information regarding the type of mask used is available in the format information area of each code, as shown in Figure 1 [2].

As can be seen in Figure 1, every QR code has some similar basic characteristics, which are [2]:

- Three large blocks in the lower left, upper left, and upper right corners that serve as anchors and ensure quick readability
- One or more smaller blocks – depending on QR version – starting in the lower right corner, and in later versions are equally distributed throughout the code
- A row of alternating black and white modules between the upper left anchor to the upper right anchor, as well as a column of alternating black and white modules between the upper left anchor and the lower left anchor
- Version information located above the lower left anchor and to the left of the upper right anchor
- Formatting information located around the upper left anchor, to the right of the lower left anchor, and below the upper right anchor

The knowledge of these fundamental properties will help identify, classify, and standardize any QR codes that might be created during research.

RGB Colorspace. During this research, Mathworks' MATLAB is the primary platform being used. MATLAB reads in color images and recognizes them as arrays of pixels, each with three layers. Every pixel is represented by

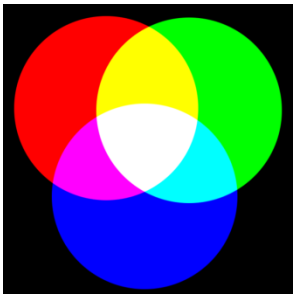

Figure 2. *A representation of RGB colorspace with base colors and their interactions. [6]*

an array itself, with that array being shown as [R,G,B], or some intensity in Red, some intensity in Green, and some intensity in Blue. If every layer has an intensity of 0 (the minimum), the color shown is black; an intensity of 255 (the maximum) on every layer displays a white color. Figure 2 provides a comprehensive representation of each of the colors and their combinations at minimum and maximum intensities.

Objectives

In this study, three goals will be achieved. They are as follows:

1. Singular QR codes will be layered in order to create a layered QR code
2. Layered QR codes will be un-layered in order to obtain singular QR codes
3. A system for reading layered QR codes efficiently and effectively will be established

Broader Impact

As mentioned in the Introduction, QR codes already have many uses in a variety of applications. With this research, QR codes will be expanded in their ability to store data without requiring a larger QR code to be used. The whole premise of QR codes is to contain a lot of information in a small, encoded space, and this project will optimize that capability and ensure varying usability.

Upon completion of this research, it is hoped that a layered QR code with three layers can hold up to 12,888 alphanumeric characters – an equivalent of about 10 pages of double-spaced 12-point font typing – and a layered QR code with six layers may hold up to 25,776 alphanumeric characters, or approximately 20 pages of typing. Ultimately, a layered QR code with six base layers should be able to hold all of the contents of this paper.

Procedure

Layering QR Codes

Three Base Colors. This study begins with three separate, herein called 'singular' QR codes. They are shown below in Figure 3.

QR1: en.wikipedia.org/wiki/Stickley

QR2: en.wikipedia.org/wiki/Numb3rs

QR3: en.wikipedia.org/wiki/Chopin

Figure 3. *Singular QR codes, obtained by using Kaywa QR Code Generator [4]*

The platform being used is Mathworks' MATLAB, which recognizes images in RGB layers, or three separate layers – red, green, and blue – in which the intensity of each pixel is a number between 0 and 255. Knowing this, it is best to transform the singular QR codes into one of the three primary colors recognized by MATLAB as shown in Figure 4. In this case, the QR codes were colorized using GNU Image Manipulation Program (GIMP), but any photo editing software may be used.

QR1R – QR1 in Red

QR2G – QR2 in Green

QR3B – QR3 in Blue

Figure 4. *Singular QR codes colorized in RGB (Author's Work)*

Another useful step is to colorize each corner box with one base color, so that, once layered, the base colors involved may be easily identified. An example of this can be seen in Figure 4.

After the manipulation of the singular QR codes, reading the images into MATLAB and adding them (yes, simply QR1R + QR2G + QR3B) produces a layered QR code, much like the one shown in Figure 5.

Figure 5. *IT – Layered QR code with three base colors (Author's Work)*

MATLAB reads this layered image, called IT, as a matrix with dimensions **{x, y, 3}**, where **x** and **y** are the horizontal and vertical dimensions, respectively, and there are 3 layers, as mentioned before – R, G, and B. Using imtool(IT), the user can identify the matrix value for each pixel, meaning simply: for that individual pixel, the values on levels [R,G,B] are [#,#,#]. For this example, each color is quantified as follows:

Black	=	[	0,	0,	0]
White	=	[	255,	255,	255]
Red	=	[	255,	0,	0]
Green	=	[	0,	255,	0]
Blue	=	[	0,	0,	255]
Light Blue	=	[	0,	255,	255]
Pink	=	[	255,	0,	255]
Yellow	=	[	255,	255,	0]

Once layered, this colorful QR code can provide three times as much storage space as a black and white QR code of the same size.

Six Base Colors. Until this point, the paper has discussed the layering of only three color channels, with eight possible combinations of color. With the addition of other colors, more options and combinations are available; therefore, more data can be stored in the same amount of space. One way of going about this is to change the intensities on each color channel – setting a 'high' color with an intensity of 170 and a 'low' color with an intensity of 85 – creating 6 base options for colors and 35 possible combinations. An example of the base colors is shown in Figure 6.

The numbers 170 and 85 were chosen because their sum is equal to 255 and they are distributed equally between 0 and 255, allowing for optimal distance for increased readability.

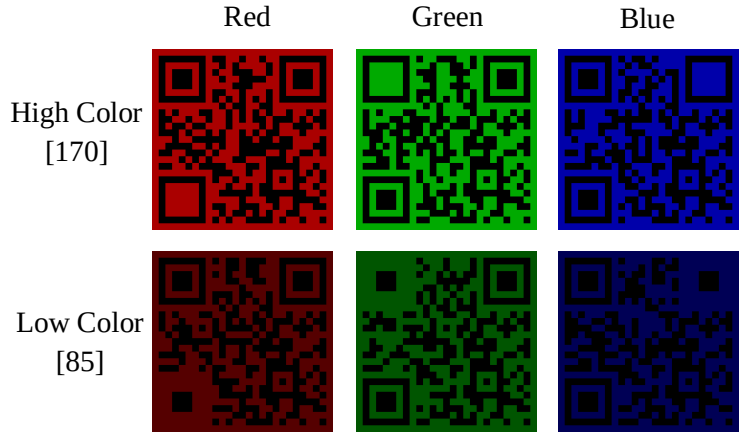

Figure 6. *Example of six base colors (Author's Work)*

Figure 7. *Layered QR code with six base colors. (Author's Work)*

Once again, the base colors can be easily identified once layered because the borders of the corner anchors have been colored accordingly.

The quantification of these new base colors and their combinations in matrix form can be found in Appendix A.

Un-layering QR Codes

Three Base Colors. Since MATLAB reads images in as a series of layers, it is fairly easy to – in a way – 'peel' those layers apart. As discussed before, each pixel of IT is recognized as a set of three intensities: [R, G, B]. By 'turning off' certain color channels, we can isolate the individual colors. For instance, setting the red and green colors equal to zero will isolate the blue color channel.

The method described above can be easily visualized when looking at the matrix form of the pixels. For example, any pink pixel in IT – the layered QR code – is simply a combination of Red and Blue, and looks like this:

Pink = [255, 0, 255]

An isolation of the red layer will set the green and blue layers to 0, resulting in this:

Red = [255, 0, 0]

Any pixel that is pink indicates a full intensity in red, meaning it is red in the red singular QR code. On the other hand, any pixel that is Light Blue:

Light Blue = [0, 255, 255]

isolated in the red channel:

Black = [0, 0, 0]

indicates zero intensity in the red singular QR code. This method allows for a simple decomposition of the layered QR code, as long as only three colors – red, green, and blue – are involved.

Once the color channels are isolated, the resulting layers are exactly identical to the original respectively colored QR codes. For further information, please reference Appendix B, which has the full MATLAB code for the process used.

Six Base Colors. Un-layering three base colors proved to be relatively simple because it was merely a process of isolating layers. However, un-layering a QR code with six base colors requires a few extra steps, as shown in the process outlined below. For further information, please reference Appendix B, which has the full MATLAB code for the process used.

1. **Isolating the Color Channels**

   As described in the section discussing three base colors, irrelevant layers need to be turned off. This can be accomplished by setting the irrelevant layers equal to 0. The result of isolating a color channel can be seen in Figure 8. For the sake of clarity, this isolated red layer, as shown, will be called ITR.

Figure 8. *ITR - Isolated Red channel with intensities at 0, 85, 170, and 255. (Author's Work)*

2. **Obtaining the High Color**

   Once the desired color is isolated, a variable called ITR1 can be created and set equal to ITR. After that, adding 85 to ITR1 then subtracting 170 will result in an identical copy of the original high color, only darker. This can be seen in Figure 9.

Figure 9. *ITR1 - Extracted High Color QR code using un-layering process (Author's Work)*

3. Obtaining the Low Color

To obtain the low color, the high color that was just obtained needs to be removed from the original layered, or mixed, color channel, or ITR. This can be done by simply subtracting ITR1 from ITR, and then subtracting ITR1 from the resulting array. Once completed, the original low color, called ITR2, will be present. This is shown in Figure 10.

Figure 10. *ITR2 - Extracted Low Color QR code using un-layering process (Author's Work)*

Results

Binary (black and white) QR codes can currently hold a maximum of 4296 alphanumeric characters. By layering QR codes, the capacity may be greatly increased. For instance, three base colors layered can triple the capacity of data, or hold 12,888 alphanumeric characters – an equivalent to approximately 10 pages of double-spaced 12-point font typing. Layering six base colors creates a code that is able to hold 25,776 alphanumeric characters, which is equivalent to about 20 pages of typing.

Discussion

This research has enabled a new step in optimizing available resources in regard to QR codes. As discussed in the Results section above, the layering processes effectively tripled and sextupled the current data capacity available in black and white QR codes of the same size. This advancement means greater usability at one's fingertips.

Many QR codes used by the general population are simply website addresses and redirect the user to something that requires an internet connection. With layered QR codes, however, more information can be stored within the QR code, which allows for users to read a QR code and not need a connection to the internet.

For example, one use of colored QR codes might be an instruction manual; if a new vacuum cleaner comes with a colored QR code, the user could scan that and have all of the information about that vacuum in front of him on his smartphone or tablet. This would save resources – paper and ink needed for the instruction manual – and require fewer things for the user to keep track of or store.

Another advantage of layered QR codes is limited accessibility. When a QR code redirects the user to a website, that website is almost always accessible by other internet users who do not scan that QR code. If the creator of a QR code wants to encrypt information but doesn't want to connect it to a domain on the internet, this is an efficient way of housing that information – up to 20 pages – and delivering it to the reader without any unintended implications or unwanted viewers.

These are only a few examples of the many ways that color QR codes might be implemented.

Conclusion

Summary

In this study, individual QR codes were colorized and layered to create a layered QR code. Firstly, three base colors were used, one base color for each of the three QR codes that

were used. Secondly, six base colors were used, one base color for each of the six QR codes that were used. These base colors were then added to create a colorful layered QR code.

Once the layered QR codes were achieved, they were un-layered to obtain the original QR codes in order to decode them. MATLAB proved to be an effective program for understanding and following the encoding, layering, un-layering, and decoding processes.

The completion of this research established an improvement in data management and storage within QR codes. Up to six times the original amount of data was effectively stored in a colored, layered QR code than a black and white QR code of the same size.

Future Work

First and foremost, before any of this research can truly be implemented, appropriate software needs to be constructed. The author does not have the skillset that allows for creation of said software at this time, but perhaps another researcher's work could improve the utility of this advancement. Ideally, there needs to be a patch created for the existing ZXing software that most QR code readers use to allow for the layering and un-layering process to occur [1].

Continuation of this work would also require researchers to delve into problem-solving aspects of the project; for example, currently this process works perfectly for images that are clean-cut and easily read by MATLAB because the colors are perfectly aligned along the RGB axes. If a user were to take a picture of the layered QR codes with, say, a cell phone, various lighting conditions would inhibit the perfected alignments and make it harder for MATLAB to recognize the discrepancies between the colors present. Additional code may need to be written

to either re-align the RGB layers (using anchor colors) and/or define the threshold values for all of the present colors in the code in order to keep them from being confused with one another.

References

1. Dean, T., & Dunn, C. (n.d.). *Quick layered response (qlr) codes*. Informally published manuscript, Electrical Engineering, Stanford University, Stanford, CA, Retrieved from http://www.stanford.edu/class/ee368/Project_12/Reports/Dean_Dunn_2D_Color_Barcoding_Using_Iterative_Error_Correcting_Codes.pdf
2. Denso ADC. (2011). *Qr code essentials*. Retrieved from http://www.nacs.org/LinkClick.aspx?fileticket=D1FpVAvvJuo=&tabid=1426&mid=4802
3. Eby, C. (2011, December). *Qr code tutorial*. Retrieved from http://www.thonky.com/qr-code-tutorial/
4. *Free qr code generator and qr management with tracking, analytics, and support*. (2013). Retrieved from http://qrcode.kaywa.com/dashboard/
5. Furht, B. (2011). *Handbook of augmented reality*. New York, NY: Springer Science Business Media, LLC.
6. RGB [Web Photo]. Retrieved from http://shiftbeep.s3.amazonaws.com/wp-content/uploads/2009/10/rgb.png

Acknowledgments

The author would like to thank the National Science Foundation for providing the funding for this research. Thanks are also due to Dr. Randy Hoover for providing much-needed project guidance and advice. More thanks are due to Dr. Grant Crawford for the organization and execution of this REU program. Lastly, the author would like to thank Dr. Alfred Boysen for his guidance in the writing of this paper and the rest of the REU staff for all the time and effort that they have put into this program.

Appendix A

This appendix displays intensity values for six base colors and all possible combinations, quantified in matrix form.

Black (B)	=	[	0,	0,	0	]
White (W)	=	[	255,	255,	255	]
High Red (FR)	=	[	170,	0,	0	]
High Green (FG)	=	[	0,	170,	0	]
High Blue (FB)	=	[	0,	0,	170	]
Low Red (HR)	=	[	85,	0,	0	]
Low Green (HG)	=	[	0,	85,	0	]
Low Blue (HB)	=	[	0,	0,	85	]
HR + LR = Mixed Red	=	[	255,	0,	0	]
HG + LG = Mixed Green	=	[	0,	255,	0	]
HB + LB = Mixed Blue	=	[	0,	0,	255	]
HR + HG = High Yellow	=	[	170,	170,	0	]
HR + HB = High Pink	=	[	170,	0,	170	]
HG + HB = High Light Blue	=	[	0,	170,	170	]
LR + LG = Low Yellow	=	[	85,	85,	0	]
LR + LB = Low Pink	=	[	85,	0,	85	]
LG + LB = Low Light Blue	=	[	0,	0,	85	]

HR + LG = Partial Yellow 1	=	[	170,	85,	0	]
HR + LB = Partial Pink 1	=	[	170,	0,	85	]
HG + LB = Partial Light Blue 1	=	[	0,	170,	85	]
LR + HG = Partial Yellow 2	=	[	85,	170,	0	]
LR + HB = Partial Pink 2	=	[	85,	0,	170	]
LG + HB = Partial Light Blue 2	=	[	0,	85,	170	]
MR + HG = Partial Yellow 3	=	[	255,	170,	0	]
MR + HB = Partial Pink 3	=	[	255,	0,	170	]
MG + HB = Partial Light Blue 3	=	[	0,	255,	170	]
MR + LG = Partial Yellow 4	=	[	255,	85,	0	]
MR + LB = Partial Pink 4	=	[	255,	0,	85	]
MG + LB = Partial Light Blue 4	=	[	0,	255,	85	]
HR + MG = Partial Yellow 5	=	[	170,	255,	0	]
HR + MB = Partial Pink 5	=	[	170,	0,	255	]
HG + MB = Partial Light Blue 5	=	[	0,	170,	255	]
LR + MG = Partial Yellow 6	=	[	85,	255,	0	]
LR + MB = Partial Pink 6	=	[	85,	0,	255	]
LG + MB = Partial Light Blue 6	=	[	0,	85,	255	]

Appendix B

This appendix provides the reader with the MATLAB code used for layering and un-layering QR codes with three and six base colors.

Three Base Colors

Note: In this section, the files being read in are:

QR1R – a colored QR code with a background at 255 red
QR2G – a colored QR code with a background at 255 green
QR3B – a colored QR code with a background at 255 blue

```
% Reading in images

% Red
im1 = imread('QR1R.png');

% Green
im2 = imread('QR2G.png');

% Blue
im3 = imread('QR3B.png');

% Creating Layered QR code
IT = im1 + im2 + im3;

ITR = IT;
ITG = IT;
ITB = IT;

% Isolating Red Channel
% Obtaining Red QR code
ITR(:,:,2) = 0;
ITR(:,:,3) = 0;

% Isolating Green Channel
% Obtaining Green QR code
ITG(:,:,1) = 0;
ITG(:,:,3) = 0;

% Isolating Blue Channel
% Obtaining Blue QR code
ITB(:,:,1) = 0;
ITB(:,:,2) = 0;
```

Six Base Colors

Note: In this section, the files being read in are:

QR1R1.png – a colored QR code with a background at 170 red
QR2G1.png – a colored QR code with a background at 170 green
QR3B1.png – a colored QR code with a background at 170 blue
QR4R2.png – a colored QR code with a background at 85 red
QR5G2.png – a colored QR code with a background at 85 green
QR6B2.png – a colored QR code with a background at 85 blue

```
% Reading in images

% High Red
im1 = imread('QR1R1.png');

% High Green
im2 = imread('QR2G1.png');

% High Blue
im3 = imread('QR3B1.png');

% Low Red
im4 = imread('QR4R2.png');

% Low Green
im5 = imread('QR5G2.png');

% Low Blue
im6 = imread('QR6B2.png');

% Creating Layered QR code
IT = im1 + im2 + im3 + im4 + im5 + im6;

ITR = IT;
ITG = IT;
ITB = IT;

% Isolating Red Channel
ITR(:,:,2) = 0;
ITR(:,:,3) = 0;

% Obtaining High Red
ITR1 = ITR + 85;
ITR1 = ITR1 - 170;

% Obtaining Low Red
ITR2 = ITR - ITR1;
ITR2 = ITR2 - ITR1;
```

```
% Isolating Green Channel
ITG(:,:,1) = 0;
ITG(:,:,3) = 0;

% Obtaining High Green
ITG1 = ITG + 85;
ITG1 = ITG1 - 170;

% Obtaining Low Green
ITG2 = ITG - ITG1;
ITG2 = ITG2 - ITG1;

% Isolating Blue Channel
ITB(:,:,1) = 0;
ITB(:,:,2) = 0;

% Obtaining High Blue
ITB1 = ITB + 85;
ITB1 = ITB1 - 170;

% Obtaining Low Blue
ITB2 = ITB - ITB1;
ITB2 = ITB2 - ITB1;
```