An evaluation of the effect of JPEG, JPEG2000 and H.264/AVC on CQR Codes decoding process

Max E. Vizcarra Melgar[a], Mylène C. Q. Farias[a] and Alexandre Zaghetto[b]

[a]Department of Electrical Engineering - University of Brasilia, Brasília DF, Brazil;
[b]Department of Computer Science - University of Brasilia, Brasília DF, Brazil

ABSTRACT

This paper presents a binary matrix code based on QR Code (Quick Response Code), denoted as CQR Code (Colored Quick Response Code), and evaluates the effect of JPEG, JPEG2000 and H.264/AVC compression on the decoding process. The proposed CQR Code has three additional colors (red, green and blue), what enables twice as much storage capacity when compared to the traditional black and white QR Code. Using the Reed-Solomon error-correcting code, the CQR Code model has a theoretical correction capability of 38.41%. The goal of this paper is to evaluate the effect that degradations inserted by common image compression algorithms have on the decoding process. Results show that a successful decoding process can be achieved for compression rates up to 0.3877 bits/pixel, 0.1093 bits/pixel and 0.3808 bits/pixel for JPEG, JPEG2000 and H.264/AVC formats, respectively. The algorithm that presents the best performance is the H.264/AVC, followed by the JPEG2000, and JPEG.

Keywords: CQR Code, QR Code, JPEG, JPEG2000, H.264/AVC

1. INTRODUCTION

QR Codes are two-dimensional structures used to transmit information through a print-scan channel. They were proposed in 1994 by the Japanese company Denso Wave Incorporated. The idea behind this technology is similar to the one used on linear or matrix barcodes,[1,2] but it has a superior data density capacity and a high speed reading algorithm. These properties have made the QR Codes very popular along the past few years. The ISO/IEC 18004 standard is one of the documents that describes the encoding of data of QR Codes.[3] An example of a QR Code is shown in Figure 1 (a).

In a previous publication, we proposed a technique to generate colored QR Codes.[4] Up to our knowledge, this was the first technique used to generate Colored QR (CQR) Codes which had the purpose of increasing the stored data capacity without using black modules in the Encoding Region. Figure 1 (b) shows an example of the proposed CQR Code.

(a) (b)

Figure 1. Example of (a) regular QR Code[3] and (b) Colored QR (CQR) Code.[4]

Given that the capture, storage and transmission of a QR Code may involve its compression, it is necessary to evaluate the effect of lossy compression on the decoding process. In this paper, our goal is to evaluate the effect that degradation

Further author information: (Send correspondence to)
Max E. Vizcarra Melgar: E-mail: maxvizcarra@ieee.org, Telephone: +55 61 31075575
Mylène C. Q. Farias: E-mail: mylene@ieee.org, Telephone: +55 61 31075575
Alexandre Zaghetto: E-mail: zaghetto@image.unb.br, Telephone: +55 61 31075575

Digital Photography XI, edited by Nitin Sampat, Radka Tezaur, Dietmar Wüller, Proc. of SPIE-IS&T Electronic Imaging
SPIE Vol. 9404, 94040Q · © 2015 SPIE-IS&T · CCC code: 0277-786X/15/$18 · doi: 10.1117/12.2083263

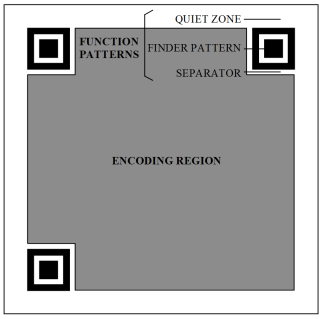

Figure 2. Distribution of CQR Code modules over Function Patterns and Encoding Regions.

inserted by common image compression algorithms has on the decoding proccess. At the same time, we want to verify what is the maximum compression rate that a CQR Code image can be compressed without affecting the decoding process. For this study, we have tested three popular compression algorithms: the JPEG,[5] JPEG2000,[6,7] and H.264/AVC.[8–10] JPEG is still the most popular compression standard, while JPEG2000 is its successor. H.264/AVC, on the other hand, is a video compression standard that has recently been used for image compression, surpassing the JPEG2000 performance.[11,12] All three standards are lossy algorithms[13–16] that can derail the correct information extraction from the CQR Code depending of the compression factor chosen by the user at the time of image capture.

This paper is divided as follows. In Section 2, we briefly introduce the algorithm to generate the CQR Code. In Section 3, we evaluate the effects of the above-mentioned image compression algorithms on the CQR decoding process. Finally, in Section 4 we present our conclusions.

2. COLORED QR CODES

The proposed Colored QR (CQR) Codes are made up of 49×49 modules.[4] A module is defined as a colored square area that represents data and redundancy bits. The colors red (255,0,0), green (0,255,0), blue (0,0,255) and white (255,255,255) are used to represent information and redundancy bits, while the colors black (0,0,0) and white (255,255,255) are used as Function Patterns. These set of colors were chosen because they are all maximally equidistant on the RGB color space, what makes them less prone to errors in the decoding stage.

Each module has a size of $n \times n$ pixels, where n depends on the target size of the CQR Code. Modules are distributed over two distinct regions: Function Patterns and Encoding Region.[4] The modules of the Function Patterns have the sole purpose of detecting the presence of a CQR Code in the image. Function Patterns are divided into: (a) quiet zones, (b) finder patterns, and (c) separators.

The modules of the Encoding Region contain information and redundancy bits. Figure 2 illustrates the CQR Code modules distributed over the Function Patterns and Encoding Regions. From the 2401 (49×49) modules of the CQR Code, 192 are inside the Function Patterns and 2209 are inside the Encoding Region. Since each module inside the Encoding Region represents 2 bits, there are 4418 bits available to carry data.

The print-scan process may be seen as part of a communication system.[4] So, the information transmitted through a CQR Code image may be susceptible to transmission errors introduced by a noisy channel. Error-correcting codes are used to prevent errors by adding extra information (redundancy) data that cross-checks the original data.[4] Similar to the QR Code, the CQR Code also uses the Reed-Solomon algorithm error-correction code to protect the transmitted information.[17,18] The Reed-Solomon algorithm takes as input k symbols of s bits and adds $n-k$ parity symbols, forming a code-word of n symbols.

In the case of the CQR Code, the Reed-Somolom parameters[17,18] are $k = 64$, $s = 16$ and $n = 276$. The resulting code-word RS(n,k) is given by the following equation:

$$RS(276, 64) = [D_1 \cdots D_{64}\ RS_1 \cdots RS_{212}], \tag{1}$$

where $D_i, i = 1 \cdots 64$ represents the k input symbols and $RS_j,\ j = 1 \cdots 212$, represents the $n-k$ parity symbols. The primitive polynomial used in the generation of redundancy bits is given by following equation:

$$p(D) = D^{16} + D^{12} + D^3 + 1. \tag{2}$$

The Reed-Solomon decoder can correct up to $t = (n - k)/2.$ symbols that contain errors in a codeword. For the CQR Code, t is 106, what means that even if 38.41% of the symbols are lost or degraded, it is possible to recover all original 64 data symbols. Therefore, of the 4418 bits available for data, 1024 are reserved for information bits and 3392 are redundancy bits added by the Reed-Solomon error correcting code. Two bits (1 module) are left unused. Hence, instead of storing only one bit in each module like in traditional QR Codes, CQR Codes store 2 bits per module. This duplicates the data capacity of the Encoding Region.

The modules are placed in the Encoding Regions in a bottom-up order, i.e. from the most right to the most left column. Figure 3 illustrates the order in which Encoding Region is filled with the data modules.

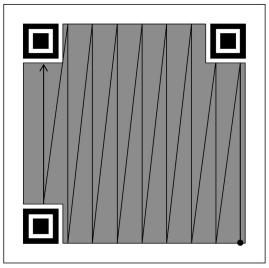

Figure 3. Order in which Encoding Region is filled with data modules: from the most right to the most left column.

The stages of decoding of the CQR Code are depicted in Figure 4, more details of the decode process can be found in.[4] After the stages of acquisition (using a digital camera) of a CQR Code image, it is common to compress the generated image for a more efficient transmission or storage. This scenario is particularly useful when the decoding process is performed in a remote receiver that supports several image formats. In the next section, we analyze the effects of JPEG, JPEG2000 and H.264/AVC compression on the decoding of CQR Codes.

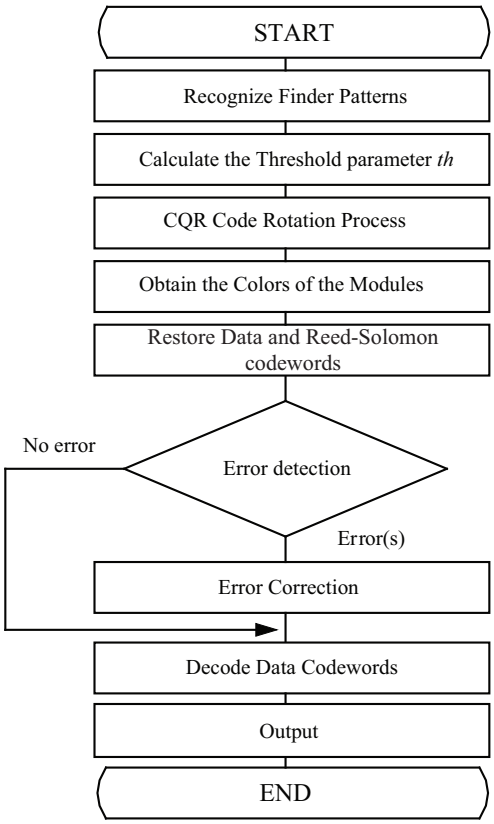

Figure 4. Fluxogram of CQR Codes decoding stage.

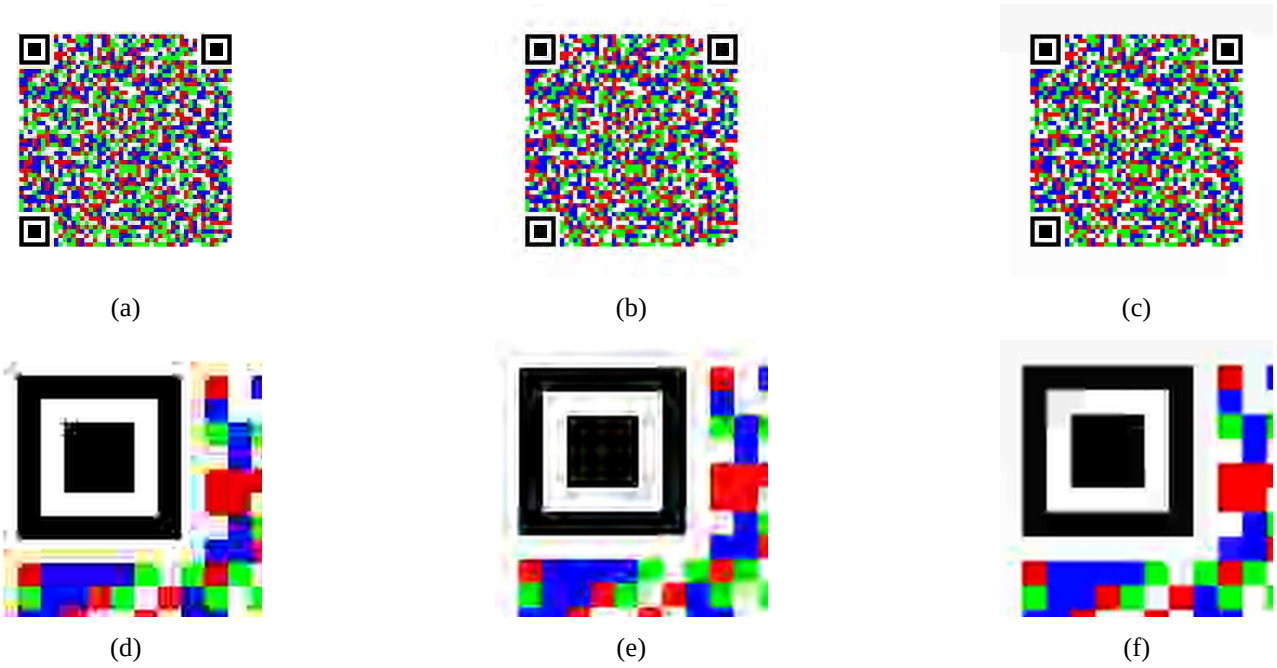

Figure 5. CQR Code Image from Figure 1 (b) compressed with: (a) JPEG@0.3841 bpp and 27.90% corrupted symbols, (b) JPEG2000@0.3908 bpp and 5.43% corrupted symbols, finally (c) H.264/AVC@0.3862 bpp and 0.36% corrupted symbols. Images shown in (d), (e), and (f) are enlarged versions of (a), (b), and (c), respectively.

3. RESULTS

Our tests were performed with CQR Codes with seven modules of Quiet Zone on each side. The original CQR Code images (1 module per pixel) were resized to make them ten times larger, resulting in a CQR Code of 630×630 modules, 490 pixels of Encoding Regions, and 140 pixels of Quiet Zone in the vertical or horizontal directions. The input images are in a 8 bits/pixel bitmap format (RGB). In this paper, we evaluate the distortions from the compression using the algorithms JPEG, JPEG2000, and H.264/AVC. Effects of the distortions caused by printing and acquisition processes have been analyzed in a previous article.[4]

The CQR Code shown in Figure 1 (b) is the first of ten CQR Codes used in our tests. The other nine images are very similar and are not shown here. To evaluate the effect that compression degradations have on the decoding of CQR Codes, we performed an analysis of the compression bitrate values (bits per pixel – bpp) versus the percentage of corrupted symbols for all three compression algorithms.

For a better understanding of the impact that compression algorithms have on the quality and, consequently, on the degradation of CQR Code images, Figures 5 (a), (b), and (c) depict examples of the image in Figure 1 (b) compressed using JPEG, JPEG200, and H.264/AVC, respectively. The compression bitrates used in this example are around 0.38 bpp (0.3841 bpp for JPEG, 0.3909 bpp for JPEG2000, and 0.3862 bpp for H.264/AVC). Figures 5 (d), (e), and (f) depict a zoom of the images in Figures 5 (a), (b), and (c), respectively. As can be noticed from these images, the difference between these three images is remarkable. As expected, the image compressed using JPEG has the worst quality (more visible degradations), while the image compressed with H.264/AVC has the best quality (less visible degradations).

As mentioned in the previous section, the error correction algorithm guarantees that if the percentage of corrupted symbols decoded (information and redundancy symbols) is less than 38.41%, the CQR Code is decoded successfully. In this case, all 1024 information bits are retrieved and the CQR Code decoding process is considered 'successful'. If, on the other hand, the percentage of corrupted symbols is greater than 38.41%, the CQR Code decoding process is considered 'unsuccessful'. The results obtained for JPEG, JPEG2000, and H.264/AVC are presented in Tables 1, 2, and 3, respectively.

Our tests were performed using 10 different CQR Codes images. In the case of JPEG compression (Table 1), CQR Codes can be decoded from rates greater than 0.3877 bpp (average value for set), which corresponds to the highest compression rate (level 100 in JPEG compression). At this rate, the CQR Code obtains (on average) 74 corrupted symbols,

Table 1. JPEG: Corrupted Symbols (%) versus compression bitrate (bpp).

Compression Bitrate	Corrupted Symbols (%)	Status
5.0512bpp	0.00%	Successful
1.9710bpp	0.00%	Successful
1.4772bpp	0.04%	Successful
1.2006bpp	0.33%	Successful
1.0480bpp	2.25%	Successful
0.9410bpp	4.86%	Successful
0.8243bpp	6.70%	Successful
0.7135bpp	8.41%	Successful
0.5417bpp	11.20%	Successful
0.4889bpp	15.87%	Successful
0.4259bpp	22.21%	Successful
0.3877bpp	26.81%	Successful

Table 2. JPEG2000: Corrupted Symbols (%) versus compression bitrate (bpp).

Compression Bitrate	Corrupted Symbols (%)	Status
0.7909bpp	0.00%	Successful
0.4697bpp	1.23%	Successful
0.3904bpp	2.28%	Successful
0.1871bpp	6.63%	Successful
0.1698bpp	6.99%	Successful
0.1552bpp	9.24%	Successful
0.1332bpp	23.80%	Successful
0.1242bpp	29.13%	Successful
0.1157bpp	31.07%	Successful
0.1093bpp	35.30%	Successful
0.1001bpp	43.99%	Unsuccessful
0.0810bpp	53.26%	Unsuccessful

which translates into an average percentage of 26.81% of symbols. This percentage of corrupted symbols can be corrected by the Reed-Solomon algorithm. For JPEG2000 compression (Table 2), CQR Codes can be decoded with compression rates greater than 0.1093 bpp (average value for set), which corresponds to level 212 in JPEG2000 compression. For this rate, an average of 97.4 corrupted symbols were found which corresponds to an average percentage of 35.30% of symbols. Rates like 0.1001 or 0.0810 cause the CQR Code to be decoded with a greater percentage of corrupted symbols than what is allowed. Finally, for H.264/AVC compression (Table 3), CQR Codes can be decoded with rates higher than 0.3808 bpp (average value for set). This rate is also the highest rate possible, which corresponds to level 51 in H.264/AVC. The average percentage of corrupted symbols for this rate is 0.18%.

To get a better idea of the variation of results, Table 4 shows the minimum, maximum, and average population variance of the percentage corrupted symbols for each compression algorithm. The minimum values for the population variance (0 for all three compression algorithms) correspond to the images with the lowest compression ratio and, therefore, no decoding errors. As expected, the maximum values for the population variance depend heavily on the compression algorithms and correspond to images with the highest compression ratios. For all test images corresponding to each compression algorithm, the average variance is not high variance between. H.264/AVC shows the smallest variation and JPEG2000 the larger variation.

Figure 6 summarizes the results for all three compression algorithms, presenting a graph of percentage of corrupted symbols in the decoded image versus the compression bitrate (bpp). The red line in the graph correspond to H.264/AVC results, the green to JPEG200, and the blue to JPEG. The graph shows that for all birates H.264/AVC has the lowest percentage of corrupted symbols. These results are consistent with the subjective quality of CQR Code compressed images,

Table 3. H.264/AVC: Corrupted Symbols (%) versus compression bitrate (bpp).

Compression Bitrate	Corrupted Symbols (%)	Status
1.7218bpp	0.00%	Successful
1.3066bpp	0.00%	Successful
1.0199bpp	0.00%	Successful
0.7737bpp	0.00%	Successful
0.6003bpp	0.00%	Successful
0.4576bpp	0.00%	Successful
0.4176bpp	0.00%	Successful
0.3808bpp	0.18%	Successful

Table 4. Population variance of corrupted symbols.

Compression Algorithm	Minimum Population Variance	Maximum Population Variance	Average Population Variance
JPEG	0.00	0.0006590	0.0002376
JPEG2000	0.00	0.0101007	0.0033516
H.264/AVC	0.00	0.0000178	0.0000029

presented in Figure 5.

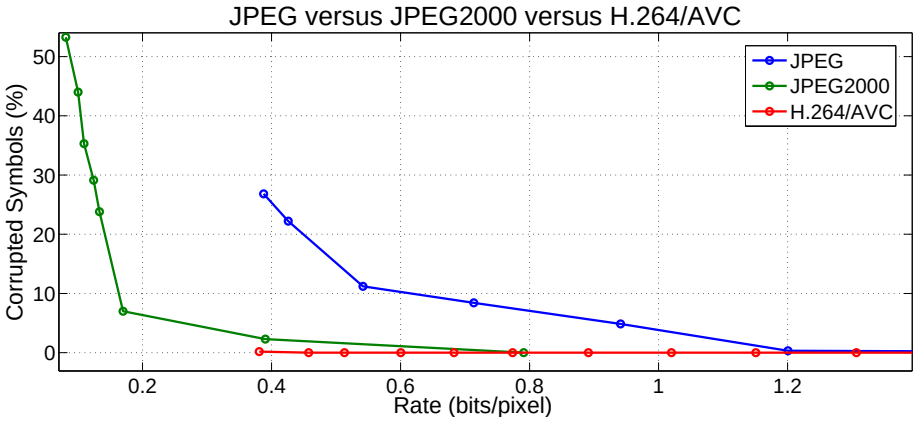

Figure 6. Percentage of corrupted symbols of CQR Code versus compression bitrate (0-1.5 bpp) using JPEG, JPEG2000 and H.264/AVC.

4. CONCLUSIONS

This paper evaluated the effect that compression degradations have on the decoding of Colored QR (CQR) Codes. We tested three compression algorithms: JPEG, JPEG2000 and H.264/AVC. These algorithms were used to compress CQR Code images. After compression, the degraded images were decoded using the CQR decoding algorithm. We then measured the percentage of corrupted pixels and compared the performance of the three compression algorithms. It was verified that the CQR Code is successfully decoded for compression rates higher than 0.3877 bpp, 0.1093 bpp and 0.3808 bpp for JPEG, JPEG2000 and H.264/AVC, respectively. The algorithm that presented the best performance was H.264/AVC, followed by JPEG2000, and, finally, by JPEG.

REFERENCES

1. H. Kato and K. T. Tan, "2D Barcodes for mobile phones," in [*Proc. of the 2nd Int. Conf. on Mobile Tech., Applications and Systems*], (November 2005).
2. I.C., Dita and M. Otesteanu and Q. Franz, "Data Matrix Code - A reliable optical identification of microelectronic components," in [*Proc. of the IEEE 17th Int. Sym. for Design and Tech. in Electronic Packaging*], (October 2011).

3. “ISO/IEC 18004:2005 - Information technology - Automatic identification and data capture techniques - QR Code 2005 bar code symbology specification,” (August 2005).
4. Melgar M. E. V. and Zaghetto A. and Macchiavello B. and Nascimento A. C. A, “CQR Codes: Colored Quick-Response Codes,” in [*Proc. of the 2ND IEEE International Conference on Consumer Electronics - Berlin (IEEE 2012 ICCE-Berlin)*], (September 2012).
5. “ISO/IEC 10918-1:1994 Information technology – Digital compression and coding of continuous-tone still images: Requirements and guidelines,” (1994).
6. “ISO/IEC 15444-1:2004 Information technology – JPEG 2000 image coding system: Core coding system,” (2004).
7. Taubman, D. S. and Marcellin, M. W., [*JPEG 2000: Image Compression Fundamentals, Standards and Practice*], Kluwer Academic (2002).
8. Queiroz R L, “Fringe Benefits of the H.264/AVC,” in [*Proc. of International Telecommunications Symposium*], 208–212 (September 2006).
9. Marpe D. and George V and Wiegand T., “Performance Comparison of Intra-only H.264/AVC and JPEG2000 for a set ofMonochrome ISO/IEC Test Images ,” in [*Contribution JVT ISO/IEC MPEG and ITU-T VCEG*], **Doc JVT M-014** (October 2004).
10. Marpe D, “Performance Evaluation of Motion-JPEG2000 in Comparison with H.264/AVC Operated in Pure Intra Coding Mode,” in [*Wavelet Applications in Industrial Processing*], **5266**, 129–137 (October 2004).
11. Alexandre Zaghetto and Ricardo L. de Queiroz., “Segmentation-Driven Compound Document Coding Based on H.264/AVC-INTRA,” in [*Image Processing, IEEE Transactions*], **16**, 1755–1760 (July 2007).
12. Alexandre Zaghetto and Ricardo L. de Queiroz., “Scanned Document Compression Using Block-Based Hybrid Video Codec,” in [*Image Processing, IEEE Transactions*], **22**, 2420–2428 (June 2013).
13. Gonzales, R. C. and Woods, R. E., [*Digital Image Processing*], Prentice Hall, 2nd ed. (2002).
14. Sayook, K., [*Introduction to Data Compression*], Morgan Kaufmanl, 3rd ed. (2006).
15. Poularikas, A. D., [*The Transform and Applications Handbook*], IEEE Press, 2nd ed. (2000).
16. Wu, H. R. and Rao, K. R., [*Digital Video Image Quality and Perceptual Coding*], Willey, 3rd ed. (2001).
17. Berlekamp E., “Nonbinary BCH decoding,” in [*Proc. of the Int. Symp. Inf. Theory (ISIT)*], **14 n.2**, 242 (1967).
18. BARRY, A. CIPRA, “The Ubiquitous Reed-Solomon Codes,” in [*Society for Industrial and Applied Mathematics*], (January 1993).