VISUALLY SIGNICANT QR CODES

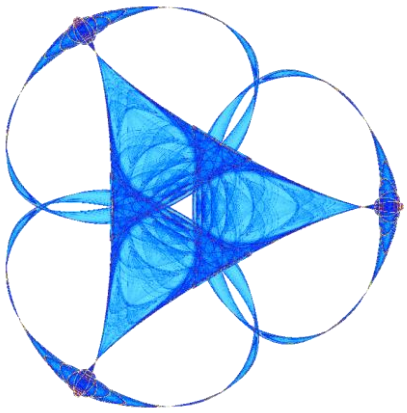
By

M. Bayeh, E. Compaan, T. Lindsey, S. Melczer, N. Orlow, and Z. Voller

Mentor: Izhak (Zachi) Baharav

IMA Preprint Series #2431

(August 2014)



INSTITUTE FOR MATHEMATICS AND ITS APPLICATIONS
UNIVERSITY OF MINNESOTA
400 Lind Hall
207 Church Street S.E.
Minneapolis, Minnesota 55455-0436
Phone: 612-624-6066 Fax: 612-626-7370
URL: http://www.ima.umn.edu

Visually Significant QR Codes

M. Bayeh [*] E. Compaan [†] T. Lindsey [‡]
S. Melczer [§] N. Orlow [¶] Z. Voller [‖]
Mentor: Izhak (Zachi) Baharav

August 16, 2013

Abstract

While QR codes have recently become popular as a method of rapidly bringing information to consumer, they have the drawback that they contain no human readable content. This project concerns combining image data with QR codes while maintaining the machine readability of the code. We break each module down into pixels and halftone it to correspond to the image data while preserving the overall shade necessary for QR readability. One method dynamically creates a library of common module types and then finds the library module which best approximates the image while respecting the QR constraint. These templates provide a good starting point for a modified DBS algorithm, which further improves the image quality. Another approach uses convex optimization on an input grayscale image, followed by a halftoning algorithm, to produce a black and white result which combines the image and QR code. A third method involved intelligantly halftoning the modules using thresholds based on the qualities of the input data and QR code. Finally, we consider data loss and use entropy calculations to investigate how the algorithms affect the amount of data encoded.

[*]Department of Mathematics and Statistics, University of Regina, Regina, Saskatchewan. bayeh20m@uregina.ca

[†]Department of Mathematics, University of Illinois, Urbana-Champaign, Illinois. compaan2@illinois.edu

[‡]Department of Mathematics, University of Kansas, Lawrence, Kansas. theodore.lindsey@gmail.com

[§]Department of Mathematics, Simon Fraser University, Vancouver, British Columbia. srm9@sfu.ca

[¶]Department of Mathematics, University of Illinois, Urbana-Champaign, Illinois. nathan.orlow@gmail.com

[‖]Department of Mathematics, Iowa State University, Ames, Iowa. zvoller@iastate.edu

1 Introduction

This project considers the QR (short for Quick Response) codes invented by a Toyota subsidiary as a inventory-tracking device. Recently, they have become particularly popular in advertising as viewers can use a smartphone to scan codes and be immediately navigated to product websites or other information. However, since the information is encoded as a sequence of black and white squares, there is no human readable content in current QR codes. Our objective is to integrate images into the codes to make them more visually appealing and significant to human eyes while maintaining readibility for scanning devices.

Although the codes come in many versions, each with its own number of black and white modules and data capacity, this work is concerned with Version 6 QR codes. The codes of this type have several features of particular importance to their scanability. The squares in the upper two and lower left corner, as well as the strips of alternating pixels which connect them, enable the reader software to adjust for skew and determine the size of the QR code when scanning. Accordingly, these features were not tampered with as any changes greatly decrease the probability that the code will be readable. Each square of the QR code is called a ''module', and to embed image data, we divide each module into a number of smaller pixels.

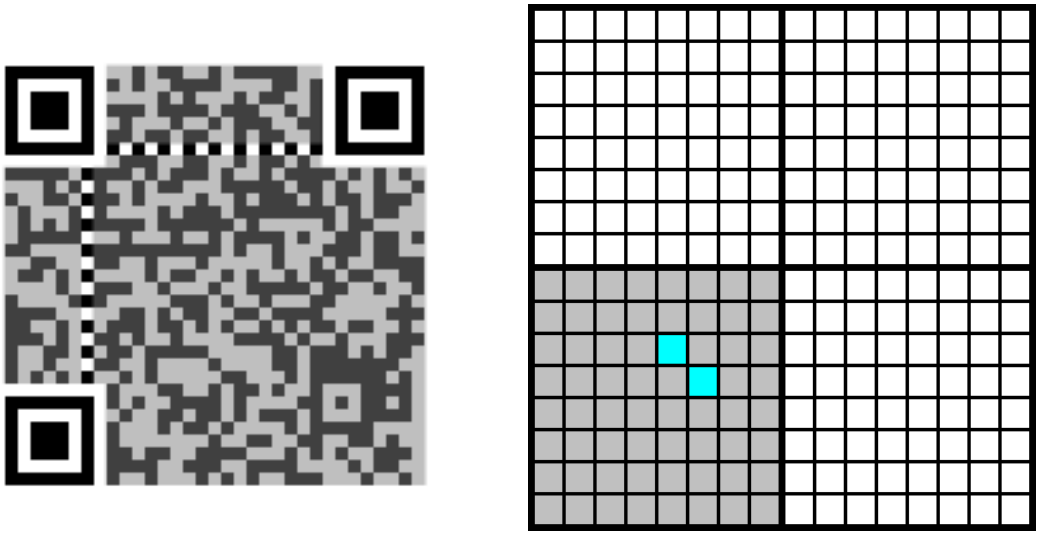


Figure 1: A QR code, and an illustration of modules and pixels

1.1 The Human Visual System

To assess the quality of our images, we use a filter which approximates the blurring effect of the human visual system. Each pixel has a perceived shade which is obtained by averaging the pixel's shade with that of its neighbors according the weighting given in the table shown. To quantify the difference between two pixels, we compare the difference between the perceived tones of the pixels (see the below table). To obtain a measure of the difference between two images, we sum the

squares of the error over all pixels. This error measure will be used throughout our work.

Table 1: The Human Visual System Filter

0.1	0.25	0.1
0.25	1	0.25
0.1	0.25	0.1

1.2 Previous Attempts

While some software is presently available to combine images with QR-codes, the most common type relies on the redundancy built into the code; the methods take an image and simply superimposes it on the center of the code. If the image is small enough, the error correction built into the code will ensure that it is still readable. Less commonly, images are embedded by shading in the background of the code – this is often very difficult to perceive – and by arrangements of black and white modules in the code. In contrast, we hope to achieve the combination without heavy relience on the error correction properties by breaking each module down into multiple pixels.

Halftoning is the process of converting a greyscale or colored image into a binary representation. The subject is well-studied, and various algorithms exist. Our problem is to implement a halftoning so that each module of image data scans to a desired color based on the QR-code data.

2 Module Rounding Approach

The main idea in this method is to combine QR code with a grayscale image by changing the brightness of modules in the image after comparing with a given QR code. After dividing an image into modules it compares the the averages of pixels in each module from QR code and image, respectively. Next, the method constructs the output by deciding whether keep the image modules or change the pixels to match the QR code more closely.

For any image, all the pixels in the image can be divided to two parts: a black part which includes all the pixels with grayscale value less than or equal to 0.5, and a white part which includes all the pixels with grayscale value above 0.5.

Let M_QR and M_{img} stand for the average grayscale of the pixels in a QR module and its corresponding image region, respectively, and T denote the determined threshold for deciding when a pixel is light or dark. Then there are cases to consider:

1. $M_{QR} \leq T$ and $M_{img} \leq T$: Pixels of module in the image will be kept in output.
2. $M_{QR} \leq T$ and $M_{img} > T$: Pixels of module in the image will become darker in the output.
3. $M_{QR} > T$ and $M_{img} \leq T$: Pixels of module in the image will become lighter in the output.
4. $M_{QR} > T$ and $M_{img} > T$: Pixels of module in the image will be kept in output.

In our code, and in the above discussion, we have taken $T = 0.5$. Furthermore, in our implementation we make a pixel darker by mapping the grayscale interval $[0, 1]$ onto the grayscale interval $[0.5, 1]$ and make a pixel lighter by mapping the grayscale interval $[0, 1]$ onto the grayscale interval $[0, 0.5]$. After the image is converted to a grayscale comptable with the QR code, we perform the error diffusion of Jarvis, Judice, and Ninke [7]. Error diffusion works by considering whether each pixel is above or below a threshold, and then spreading the error from this change to neighboring pixels which haven't yet been processed. The portion of the error which is transferred to a pixels neighbours is given by the matrix

$$M = \begin{pmatrix} 0 & 0 & X & 7/48 & 5/48 \\ 3/48 & 5/48 & 7/48 & 5/48 & 3/48 \\ 1/48 & 3/48 & 5/48 & 3/48 & 1/48 \end{pmatrix},$$

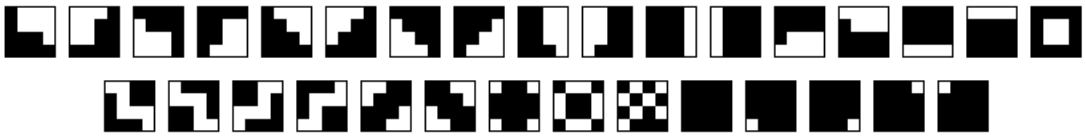
where the X represents the current pixel which is being considered. By construction if the previous pixel was black, the next pixel is more likely to be white.

3 Template Approach

3.1 Preselected Templates

One method of combining a QR-code with image data relies upon a collection of pre-configured templates. A relatively small list of templates – in our case 30 out of the 2^n possible – is created, based on human estimates of which shape arrangements are most useful in displaying an image. These templates default to a majority of black pixels so that when used unmodified, a QR code scanner would consider them to form a black module. Once this list is created, we use it to embed an image in a QR-code. Given an image, we consider each pixel block corresponding to a module in the QR-code; if the module in the code is black we use a sum squared measurement to compare the grayscale image data with each template pattern successively until we find a template which minimizes the error. The template is then used to represent the module in the final result and we move on to the next module. If the QR module needs to scan as white rather than black, the each module is inverted, then, as with a black module, each template is checked to determine which provides the minimum error and that template is used to represent the module.

Figure 2: A Selection of Templates for 4×4 Modules



3.2 Dynamic Templates

A disadvantage of the pre-configured template approach described above is that it does not adapt to a particular image which may not have specific shapes provided by the template definitions. In order to customize the collection of templates for a specific image, we divide the image into module-sized blocks. Because we only consider "black" module templates, we convert every block with more than half pixels white to a black modules with at least half of the pixels as black ones ($block = 1 - block$). We create a list of blocks in the image, throwing out any that are too similar to blocks already present in the list. We then take the n most common blocks and use them as module templates. From here, we proceed as before to assign a template with minimizes error to each module.

This method yields visually pleasing results; however, they are often not readable by scanning devices. We believe this to be due to the fact that scanning utilities put greater weight on information near the center of a modules, perhaps to account for inaccuracies in determining where the module edges lie. Thus, for instance, a module with only a few center pixels shaded will scan as black even though only a small portion of its area is darkened. In order to overcome this issue, the result can either be passed through some filter which spreads the pixels more evenly in the

module (such as DBS), or, in the case of the pre-configured templates, care can be used in ensuring that no center-biased templates are present.

4 Optimization Approach

A more analytic method involves formulating an optimization problem for each module using the grayscale data from the image. We wish to minimize the error between the input data and the halftoned image, subject to a threshold constraint imposed by the QR data which ensures a certain proportion of light or dark pixels for the module.

More formally, in any module which is black in the QR code we solve the optimization problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \|h * I - h * x\|_2^2 \\ \text{subject to} \quad & \mathbf{1} \cdot x \leq TOL \\ & 0 \leq x_i \leq 1, \end{aligned}$$

while in any module which is white in the QR code we solve the optimization problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \|h * I - h * x\|_2^2 \\ \text{subject to} \quad & \mathbf{1} \cdot x \geq n^2 - TOL \\ & 0 \leq x_i \leq 1, \end{aligned}$$

where h is the human visual filter described above, n is the size of each module in pixels, and TOL is a tolerance, taken to be $0.3n^2$ in the following results unless otherwise specified. As we have a quadratic objective function and linear constraints, this is a convex optimization problem and can be solved quickly for each module using readily available algorithms in Matlab (see [4, 6, 5] for details).

After finding the minimizing vector, we are left only with the task of converting the grayscale vector into something representing a black and white image. This is accomplished in three ways.

- **(Error Diffusion)** Here, we run the error diffusion method proposed by Jarvis, Judice, and Ninke [7] on each module. As mentioned above, this gives a good Black-and-White approximation to our grayscale image/QR combination.
- **(Ordered Dithering)** Here, we round each value of a module from its value in $[0, 1]$ to either 0 or 1 depending on whether or not it is less than the corresponding entry in a threshold matrix. We use the matrix $\frac{1}{m}B_m$, where

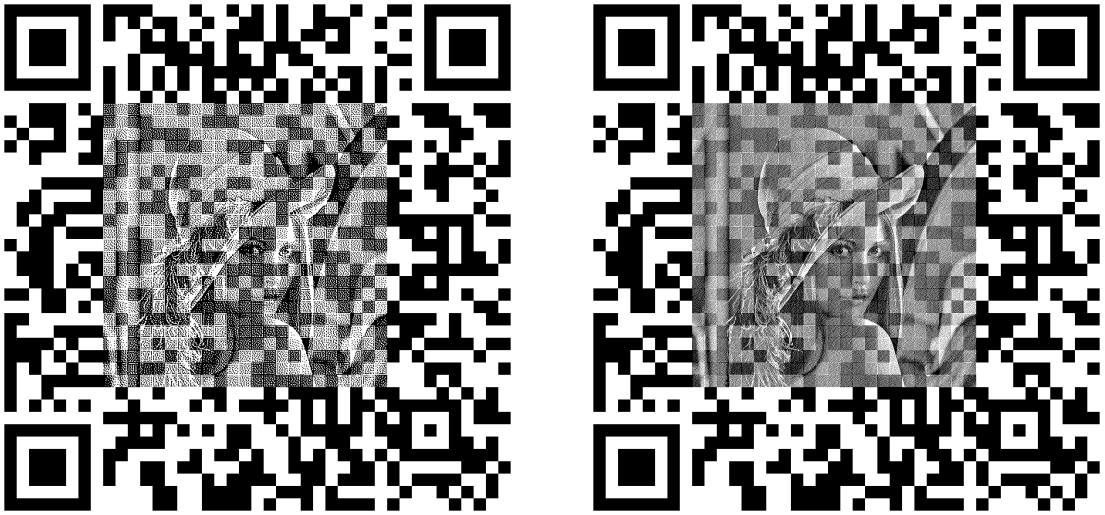


Figure 3: QR codes created with the error diffusion method containing 16 and 32 pixels per module, respectively.

the B_m are the class of matrices given by Bayer [3], and are defined for powers of two by

$$B_{2^n} = \begin{pmatrix} 4B_{2^{n-1}} + 1 & 4B_{2^{n-1}} + 2 \\ 4B_{2^{n-1}} + 3 & 4B_{2^{n-1}} + 0 \end{pmatrix}, \qquad B_2 = \begin{pmatrix} 1 & 2 \\ 3 & 0 \end{pmatrix},$$

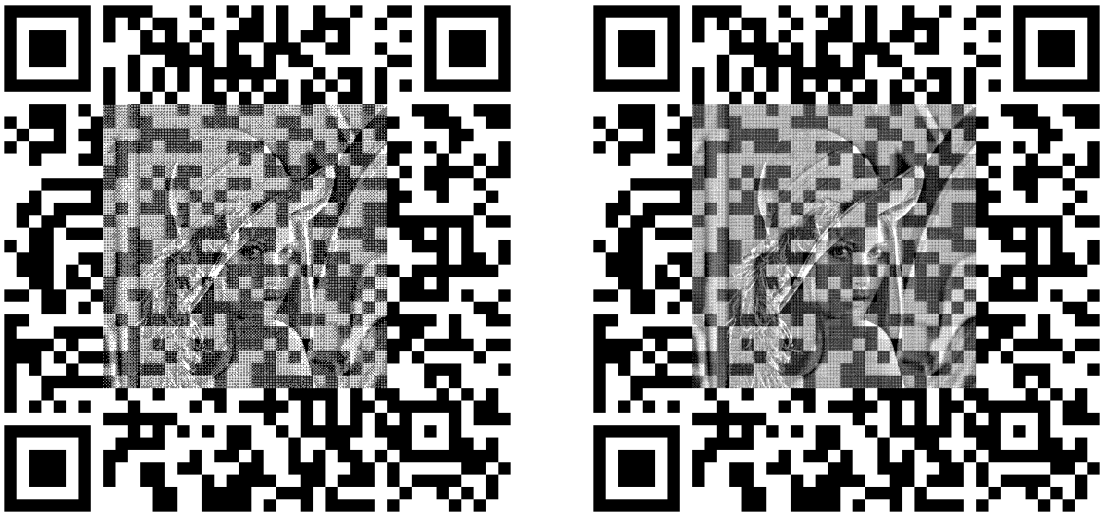


Figure 4: QR codes created with the ordered dithering method containing 16 and 32 pixels per module, respectively.

- **(Sub-pixel Method)** Here we split each pixel into a $k \times k$ region of subpixels and apply the error diffusion of Jarvis, Judice, and Ninke to make the subpixels appear close to the original grayscale pixel. This produced the best looking images of the three methods listed here, but results in a resolution k times greater (in both width and height) than the other methods, rendering

it impractical for situations where QR codes are to be printed as a small scale image.

Figure 5: QR code created with the sub-pixel method using 32 pixel modules and a scale factor of $k = 6$

5 DBS Approach

5.1 The General Direct Binary Search (DBS) Algorithm

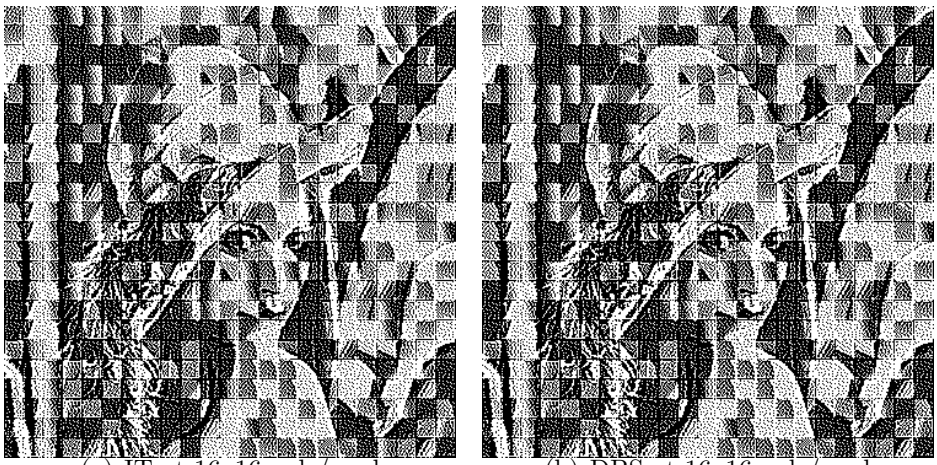
One method of refining these approaches employs a variant of the direct binary search halftoning introduced by Analoui and Allebach [1]. General DBS is a brute force iterative scheme that utilizes an initial halftoning, for example a 50% thresholding, to generate a halftoned version of a greyscale image. Each iteration of the DBS algorithm calculates the change in error that results from either toggling each pixel (from black to white or vice versa) or swapping it with one of its eight neighbors. The algorithm then implements the desired toggle or swap if any of these changes decrease the overall error, as measured by the sum squared error between filtered versions of the actual and halftoned images. Otherwise, the pixel is left

unchanged, and the next iteration commences on the adjacent pixel. This process continues until there no longer exists a toggle or swap that reduces the overall error.

The general DBS is only guaranteed to find a local minimum of the error function. Thus it is heavily dependant upon the quality of the incoming image. Higher quality initial guesses therefore yield both better results, in terms of the error function, and in the computational time required to obtain the optimal solution. The template approach, discussed in Section 3, serves as the initial guess in our implementation of a version of the DBS algorithm.

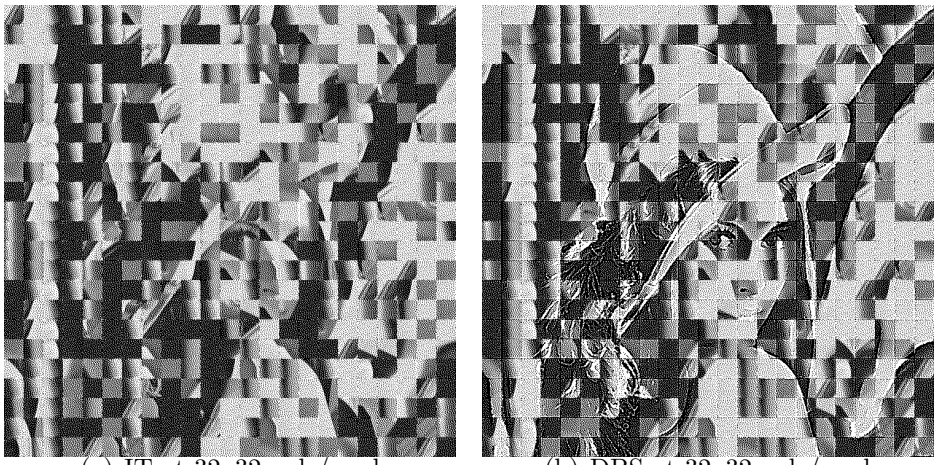
5.2 A Module Based DBS Implementation

Our implementation of the DBS algorithm considers the image data over each module individually. In order to preserve the the correct overall shading of the QR module under consideration, a constraint ensures that a toggle or swap is performed if it reduces the overall error and respects the threshold that facilitates a camera's ability to quickly scanthe image. This constraint, calculated by taking pictures of QR-codes over varying threshold values, required that 65% of the pixels within a module matched the color of the corresponding module in the QR-code, and was incorporated as a maximum or minumum possible number of black or white pixels. The above construction yielded pleasing results, seen in the figures 5.2, 5.2, and 5.2, but the computational complexity of our implementation did not the reach the theoretical best case of $\mathcal{O}(N^2)$[1].



(a) IT at 16x16 pxls/mod (b) DBS at 16x16 pxls/mod

Figure 6: Templates vs. DBS on 368 pixels



(a) IT at 32x32 pxls/mod (b) DBS at 32x32 pxls/mod

Figure 7: Templates vs. DBS on 736 pixels

Figure 8: DBS with QR code on 736 pixels

6 Information Loss

One concern which arises when combining QR codes with images is information loss. We wish to understand how the amount of data encoded is a pixel is changed

by imposing a constraint on the overall luminescence of the module. We do this by computing the entropy of the pixel to measure the amount of information encoded.

The entropy of a random variable X, in bits, is defined to be

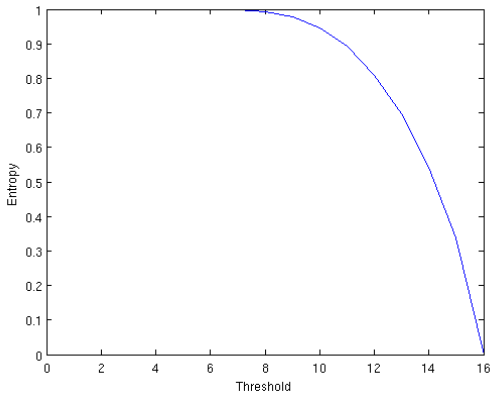
$$H(X) = -\sum_{\omega} P(\omega)\log_2(P(\omega)),$$

where the sum runs over possible values ω of the random variable.

Consider the entropy of a single pixel in a module of N pixels with the constraint that this module must meet a threshold of at least k black pixels. With some assumptions on the nature of the pixel color distribution and through enforcement of this threshold constraint, we can obtain a model for the entropy of a pixel.

First, suppose that we receive an input module of N pixels. Each pixel is modeled by a uniformly distributed random variable on $[0, 1]$ (i.e. grayscale). If we obtain a halftoned module by taking all pixels in $[0, \frac{1}{2}]$ to be white and the remainder black, the result is a collection of pixels which are either black or white with equal probability. If the resultant module has fewer than k black pixels, we enforce the threshold constraint by randomly choosing white pixels to make black in order to meet the threshold.

Figure 9: Entropy for Simple Threshold Grayscaling



For a pixel p, we have

$$\begin{aligned}
&P(p \text{ is black}) \\
&= \sum_{j=k}^{N} P(p \text{ is black} \mid j \text{ of } N \text{ pixels are black}) P(j \text{ of } N \text{ pixels are black}) \\
&= \frac{k}{N} \sum_{j=0}^{k} \binom{N}{j} / 2^N + \sum_{j=k+1}^{N} \frac{j}{N} \binom{N}{j} / 2^N.
\end{aligned}$$

The result of running a Matlab simulation of this situation with $N = 16$ is shown in the figure. The simulation yields entropy results which match those predicted by

the model above. As one expects, as the threshold increases, the entropy decreases to zero.

A more complex computation arises when the incoming data is halftoning according to a dither matrix with entries $\frac{1}{2N}, \frac{3}{2N}, \ldots, \frac{2N-1}{2N}$. Then a pixel in the halftoned module is black exactly when its input module value is greater than the respective value in the dither matrix. (For example, the first pixel is black when its value is greater than $1/2N$.) In this case, the probability of the pixel in the ℓ-place (corresponding to matrix entry $\frac{2\ell-1}{N}$) being black is given by

$$\frac{2N-2\ell+1}{2N} + \left(\frac{\prod_{m=1}^{N}(2m-1)}{2^N N^N} \right) \times$$

$$\sum_{j=0}^{k-1} \left(\frac{k-j}{N_j} \sum_{\substack{\{x_1,\ldots,x_j\} \\ x_n \in \{1,\ldots,\ell-1,\ell+1,\ldots,N\}}} \prod_{i=1}^{j} \frac{2N-2x_i+1}{2x_i-1} \right).$$

Note if we wish all pixels to be equally likely to be black as each other, we can simply randomly permute the dither matrix before applying it. In this case, it turns out that the resulting entropy for each pixel is the same for both dither matrices. For a given value of N, we can plot the resulting entropy for specific threshold values. The plots for N=16 and N=256 are below:

To dither an actual image, it is common to randomly permute the entries of the dither matrix to prevent visual artifacts. This results in each pixel having an identical distribution which is the average over ℓ for the distributions described above. This too was simulated in Matlab, and yields results nearly identical to those of the simpler thresholding described above. The distributions agreed with those predicted by the model.

Alternatively, we may assume that each pixel of the N pixels of incoming data is at the same gray level, with N possible gray levels. Here we consider the entropy of the entire collection of pixels, rather than of single modules separately. In this case, there is no data lost by performing a dither halftoning. Once the constraint on the number of black pixels is enforced, though, there is data loss. We have

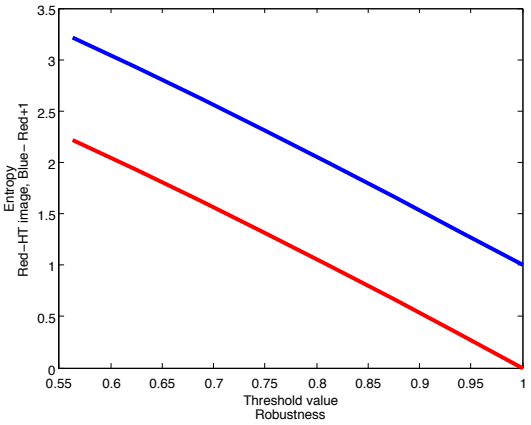
$$P(j \text{ of } N \text{ pixels black}) = \begin{cases} \frac{k}{N} & j \leq k \\ \frac{1}{N} & k < j \leq N \end{cases}.$$

The resulting entropy for $N = 16$ is shown in the figure.

7 Conclusion

We have exhibited several methods for combining an image with a QR code, in order to provide human readable content on the QR code. These methods improve

Figure 10: Entropy for Uniform Grayscale Data



on previous methods by changing the QR code on the pixel level, providing allowing much more precision with images rendered on the QR code. Slight hand improvements can be still be done for each of the algorithms. We also only investigated the sum-squares metric for calculating error; more work could be done investigating and refining these methods with the L^1 norm, or another metric.

More work needs to be done to optimize modules containing both black and white for QR readers. Specifically, determining which features of pixel configurations in a module make it easier or harder to read the module. By utilizing more readable combinations, better error correction can be built into the code. Due to the wide availability of QR readers, additional research can also be done to investigate compatibility with different readers. In addition, accumulating a population of logos and images for input would help develop features for testing and benchmarking current and future methods.

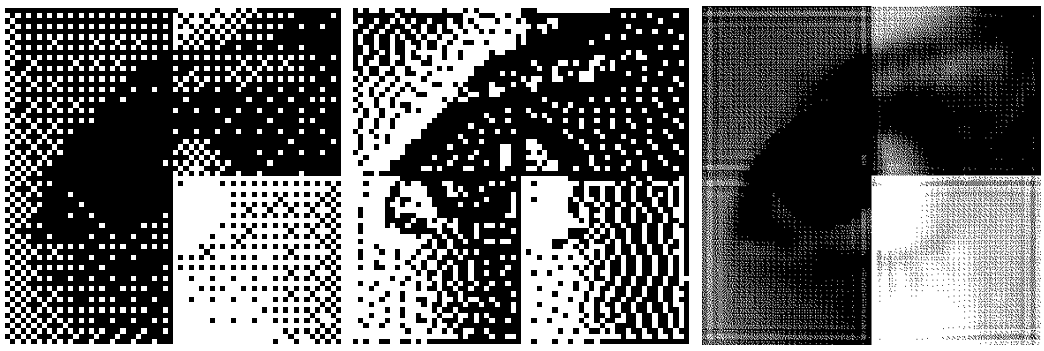


Figure 11: Closeups of the Lena image's right eye under three of the above methods (from the left: Optimization with Bayer ordered dither, module by module DBS, and sub-pixel error diffusion with a scale factor of 6)

Finally, the dominant color of each module in the QR code is the same as in the image. However, due to the error correction built into QR-codes, one can change a few of these modules to be a different color. Future work could compute which of these modules should be switched at the cost of a less robust code, and more

precisely document the balance between robustness and image quality.

Acknowledgements

Thank you to Zachi Baharav for his guidance. Thank you to the Institute for Mathematics and Its Applications and the Pacific Institute for the Mathematical Sciences for their generous support.

References

[1] M. Analoui and J. P. Allebach, *Model based halftoning using direct binary search*, Proc. SPIE **1666** (1992), 96-108.

[2] D. J. Lieberman and J. P. Alleback, *A dual interpretation for direct binary search and its implications for tone reproduction and texture quality*, IEEE Transactions on Image Processing, **9(11)** (2000), 1950-1963.

[3] B. E. Bayer, *An optimum method for two-level rendition of continuous-tone pictures*, IEEE Internation Conference on Communications **1** (1973), 11-15.

[4] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[5] T. F. Coleman and Y. Li, *A reflective Newton method for minimizing a quadratic function subject to bounds on some of the variables*, SIAM J. Optim. **6(4)** (1996), 1040-1058.

[6] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, 1981.

[7] J. F. Jarvis, C. N. Judice, and W. H. Ninke, *A survey of techniques for the display of continuous tone pictures on bilevel displays*, Computer Graphics and Image Processing **5(1)** (1976), 13-40.

[8] D. L. Lau and G. Arce, *Modern Digital Halftoning*, CRC Press, 2008.

[9] *QR code 2005 bar code symbology specification*, International Organization for Standardization, 2006.

[10] R. Ulichney, *Digital halftoning*, MIT Press, 1987.