| | |
|---|---|
| Title | 3D barcodes : theoretical aspects and practical implementation |
| Author(s) | Gladstein, David; Kakarala, Ramakrishna; Baharav, Zachi |
| Citation | Gladstein, D., Kakarala, R., & Baharav, Z. (2015). 3D barcodes: theoretical aspects and practical implementation. Proceedings of SPIE - Image Processing: Machine Vision Applications VIII, 9405, 94050N-. doi:10.1117/12.2082864 |
| Date | 2015 |
| URL | http://hdl.handle.net/10220/46899 |
| Rights | © 2015 Society of Photo-optical Instrumentation Engineers (SPIE). This paper was published in Proceedings of SPIE - Image Processing: Machine Vision Applications VIII and is made available as an electronic reprint (preprint) with permission of Society of Photo-optical Instrumentation Engineers (SPIE). The published version is available at: [http://dx.doi.org/10.1117/12.2082864]. One print or electronic copy may be made for personal use only. Systematic or multiple reproduction, distribution to multiple locations via electronic or other means, duplication of any material in this paper for a fee or for commercial purposes, or modification of the content of the paper is prohibited and is subject to penalties under law. |

# 3D Barcodes: Theoretical Aspects and Practical Implementation

David Gladstein[a], Ramakrishna Kakarala[b], and Zachi Baharav[a]

[a]Cogswell Polytechnical College, Sunnyvale, California, USA;
[b]Nanyang Technological University, Singapore

## ABSTRACT

This paper introduces the concept of three dimensional (3D) barcodes. A 3D barcode is composed of an array of 3D cells, called modules, and each can be either filled or empty, corresponding to two possible values of a bit. These barcodes have great theoretical promise thanks to their very large information capacity, which grows as the cube of the linear size of the barcode, and in addition are becoming practically manufacturable thanks to the ubiquitous use of 3D printers.

In order to make these 3D barcodes practical for consumers, it is important to keep the decoding simple using commonly available means like smartphones. We therefore limit ourselves to decoding mechanisms based only on three projections of the barcode, which imply specific constraints on the barcode itself. The three projections produce the marginal sums of the 3D cube, which are the counts of filled-in modules along each Cartesian axis.

In this paper we present some of the theoretical aspects of the 2D and 3D cases, and describe the resulting complexity of the 3D case. We then describe a method to reduce these complexities into a practical application. The method features an asymmetric coding scheme, where the decoder is much simpler than the encoder. We close by demonstrating 3D barcodes we created and their usability.

**Keywords:** Barcode, 3D coding, marginal sums, binary matrices

## 1. INTRODUCTION

Barcodes have served for many years as a very robust and efficient way to connect the printed physical world with computers. From the one-dimensional barcodes printed on most products for easy price and inventory management, to the various types of two-dimensional (2D) barcodes, including quick-response (QR) codes, that are widely used to link objects to websites, we see a progression of codes that bear increasing amounts of information while maintaining reliability and ease of use. In particular, 2D barcodes (including QR codes) have found such uses as the following: conveying passenger information on airline boarding passes, providing web links to magazine advertisements, as well as tracking parts in the automotive industry. Furthermore, research has also improved on the appearance of QR codes, which traditionally use random-looking black-and-white patterns. Recent papers have shown methods to make QR codes visually significant by using color blending of corporate logos[1] as well as through halftoning techniques.[2]

In this paper we introduce the idea of 3D barcodes. The barcode is composed of an array of 3D cells, called modules, and each can be either filled or empty, corresponding to two possible values of a bit. These barcodes have very large information capacity, which grows as the cube of the linear size of the barcode. In addition these are becoming practically manufacturable thanks to the ubiquitous use of 3D printers, which allows personal printing of 3D objects. In order to make these 3D barcodes practical for consumers, it is important to keep the decoding simple using commonly available means like smartphones. Most smartphones have an embedded camera, which allows capturing an external view of the 3D barcode. Multiple vantages can be used to capture images from various angles. We therefore limit ourselves to decoding mechanisms based only on three projections of the

Further author information: (Send correspondence to Zachi Baharav)
Zachi Baharav: zbaharav@cogswell.edu, zachi@ieee.org
Ramakrishna Kakarala: ramakrishna@ntu.edu.sg, r.kakarala@ieee.org
David Gladstein: dgladstein@cogswell.edu

Image Processing: Machine Vision Applications VIII, edited by Edmund Y. Lam, Kurt S. Niel,
Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 9405, 94050N · © 2015 SPIE-IS&T
CCC code: 0277-786X/15/$18 · doi: 10.1117/12.000000

barcode, which implies specific constraints on the barcode itself. The three projections produce the marginal sums of the 3D cube, which are the counts of filled-in modules along each Cartesian axis.

The flow of the work is as follows. We start by introducing the novel concept of 3D barcodes, and relating it to previous work on barcodes. We then move to the theoretical side, as we analyze and show the applicability of the work done on the reconstruction of 2D binary matrices based on row and column projections. We discuss both some of the original papers[3,4] as well as recent ones[5–8] . The latter works suggest ways to determine the uniqueness of the matrix given the marginal sums, the cardinality of the non-unique cases, and the cases where marginal sums do not correspond to any feasible matrix. We build upon those works and describe the hurdles in applying their techniques to the 3D case. We then present a novel algorithm for encoding information in 3D cubes, and using the marginal sums for decoding. Our approach combines both computer vision and image processing techniques, which we describe.

We close with the demonstration of various printed 3D barcodes, and describe practical aspects of their manufacturing.

## 2. 3D BARCODE

Two dimensional (2D) barcodes are in common use, and the most common form are the QR codes.[9] The 2D barcodes can be considered as a two-dimensional matrix, where each element (called module in the QR literature) is either Zero or One (Black or White). The decoding process involves capturing an image of the whole code, and analyzing the picture. In this paper we consider the possibility of expanding this idea to a 3D realm. Thus, the code can be viewed as a 3D cube (rather than a 2D matrix), and each module is a smaller unit-cube. Again, each module can be Zero or One. However, rather than blocking all light passing through it (the equivalent of Black in the 2D case), in the 3D case each "dark" module simply attenuates the light by certain amount. This will allow us, as we will see shortly, to infer the number of "dark" cells in each row and column.

This 3D construction allows for much more data-storage capacity within the code. However, it raises the issue that some internal features cannot be seen (without an x-ray vision). To overcome this, we rely on light transmission measurements. By taking an image of the side of the cube, we can see in each row/column how much light is transmitted *through* the cube. This will be indicative of the number of "dark" cells that attenuated the light-ray through the cube.

Thus, the general idea is to design a code that is described as the 3D cube of semi-transparent modules. In the decoding process, three images are taken of the 3 different sides of the cube. From those images, the light attenuation in each point is calculated, which in turn indicates the number of "dark" modules in this row/col.

## 3. 2D CASE: RECONSTRUCTION FROM MARGINALS

We start by reviewing the 2D case. The problem of reconstruction from row-sum projections (aka marginals) is well known and investigated since the late 1950's, but some results are still in terms of bounds and limits.[7,10] We first set up some definitions and the conditions for existence, and then follow with a description of the algorithm for constructing one such matrix.

### 3.1 Existence of solution given marginals

Gale[3] and Ryser[4] derived the classical results giving necessary and sufficient conditions on the existence and uniqeness of a solution to the reconstruction of $(0,1)$-matrices from marginals. Rather than repeat the formal results, we will use an example to describe a constructive solution method.

Assume $A = [a_{ij}]$ is an $m \times n$ matrix (with $m$-rows and $n$-cols) such that:

$$\text{Binary matrix:} \quad a_{ij} = 0 \quad \text{or} \quad 1 \qquad \text{for} \quad i = 1, \ldots, m \quad \text{and} \quad j = 1, \ldots, n; \tag{1}$$

$$\text{Row sum:} \quad \sum_{j=1}^{n} a_{ij} = r_i \qquad \text{for} \quad i = 1, \ldots, m; \tag{2}$$

$$\text{Column sum:} \quad \sum_{i=1}^{m} a_{ij} = s_j \qquad \text{for} \quad j = 1, \ldots, n; \tag{3}$$

and let $R = (r_1, r_2, \ldots, r_m)$ and $S = (s_1, s_2, \ldots, s_n)$ be the two vectors denoting, respectively, the row and column marginal sums. For the sequel, let us assume that the vector $R$ is monotone:

$$R \qquad \text{is monotone iff} \qquad r_1 \geq r_2 \geq \ldots \geq r_m \tag{4}$$

This is not a strong assumption, as any matrix with a specified row-sum vector can be transformed into one with a monotone row-sum vector by re-arranging the rows of the matrix. Similarly, we will assume the column-sum vector is also monotone. A condition which will be implicit in most of the following analysis (but we will need to use it explicitly in the 3D discussion) is the Weight Conservation condition, which merely states that given that the vectors describe the same matrix, their total sums should be the same:

$$\text{'Weight-of-A'} = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij} = \sum_{i=1}^{m} r_i = \sum_{j=1}^{n} s_j \tag{5}$$

One more definition we need before heading into the construction is that of a vector $U$ *majorized* by another vector $V$, denoted by $U \prec V$ (assume both vectors of length $n$):

$$U \prec V \qquad \text{iff} \qquad u_1 + u_2 \ldots + u_i \leq v_1 + v_2 \ldots + v_i \qquad \forall i = 1 \ldots n. \tag{6}$$

In the case where $U, V$ have different lengths, we pad the shorter vector with zeros and apply the same condition to determine whether majorization exists. In what follows, we use specific numerical values for ease in following the algorithm. Assume for this example that the following two vectors are given:

$$\text{Row sum:} \qquad R = [5, 4, 4, 3, 2, 1] \tag{7}$$

$$\text{Column sum:} \qquad S = [4, 3, 3, 3, 3, 3] \tag{8}$$

This implies that for this specific example, the matrix we are searching for is an $m \times n = 6 \times 6$ matrix.

In order to determine whether there is a solution (at least one), given the above two marginal distributions, we construct a new matrix of size $m \times n$ which has the row sums as required by the given marginal, and where all the 1's are located at the top left corner of the matrix (again, rigorous descriptions can be found elsewhere[5]). We then calculate the column-sums of this new matrix, and denote the column-sum as $R^*$, which is an $n$-length vector. This is shown for our specific example in Figure 1.

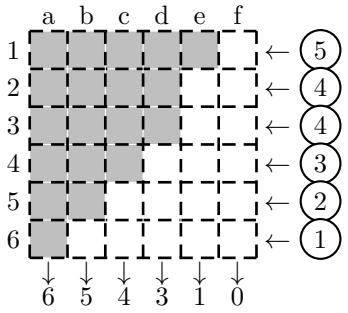



Figure 1: Constructing the matrix to determine solution-existence. The matrix is built to preserve the given row-sum (numbers on right hand side), and then the column-sum for this matrix is calculated, denoted as $R^*$.

The theorem (by Ryser and Gale) now declares: *There exists a matrix $A_{m\times n}$ having row-sum and column-sum as described by vectors $R$ and $S$ respectively, iff*

$$S \prec R^* \quad , \tag{9}$$

and

$$r_k \leq n \qquad \forall \qquad k = 1, \ldots, m \tag{10}$$

First, let's see that this holds in our particular case:

$$S = [4,3,3,3,3,3] \tag{11}$$

$$R^* = [6,5,4,3,1,0] \tag{12}$$

$$\text{and indeed} : S \prec R^* \quad \text{, as} \quad 4 \leq 6 \quad ; \quad 4+3 \leq 6+5 \quad ; \quad \text{and so on...} \tag{13}$$

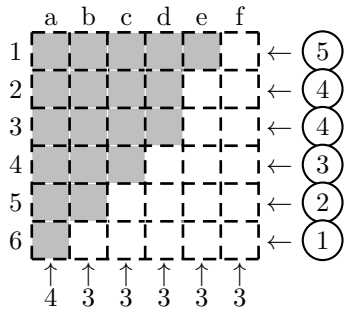
$$\text{and also} : r_k \leq 6 \qquad \forall k \tag{14}$$

This assures that there is indeed a solution (at least one)! If, for example, $S$ would have been a vector "heavy" to the left, such as $S = [6,6,3,2,2,0]$, then the above wouldn't hold, and therefore there is no matrix that satisfies these marginals.

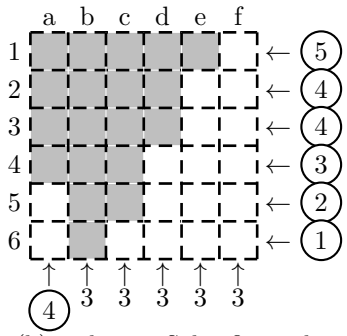
Taking a closer look at the conditions for solution existence reveals that the condition $r_k \leq n$ merely requires that no one row has sum greater than the number of columns, which is trivial to satisfy. The other requirement (of $S \prec R^*$) will become more evident as we now describe a constructive process for solution.

### 3.2 Construction of solution given marginals

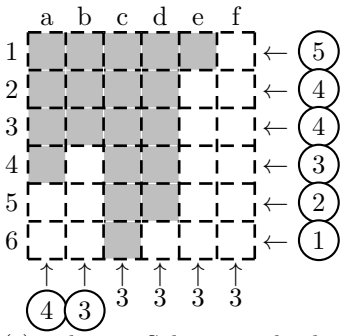
We continue the previous example, now that we know at least one solution exists, and describe the algorithm to build one such solution. The process is described in Fig. 2. At the first step, the elements are set in a matrix according to the row-sum given. At the bottom of the matrix we wrote the column-sum requirements, which are **not** satisfied at this stage. We then proceed from left to right, one column at a time, and move elements to the right in order to satisfy the needed column-sum. In each step, we adjust one column, and the number is then encircled in the picture. Thus, after the first step we have indeed column-sum equals 4 for the leftmost column. At the end of the process, all the column-sums are correct, and since we started with the right row-sums, the matrix indeed satisfies the marginals requirement.



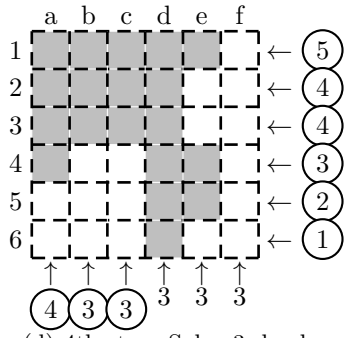

(a) 1st step: Fill-in properly



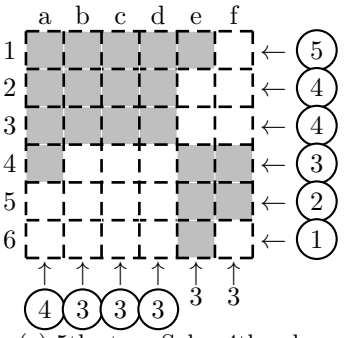

(b) 2nd step: Solve first column



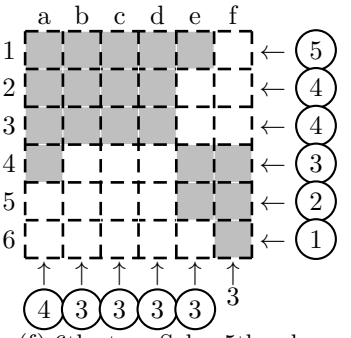

(c) 3rd step: Solve second column



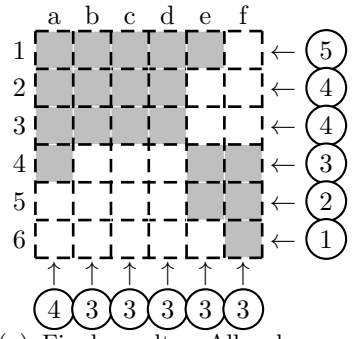

(d) 4th step: Solve 3rd column




(e) 5th step: Solve 4th column




(f) 6th step: Solve 5th column




(g) Final results: All columns are solved for.

Figure 2: Demonstrating the solution for the 2D case. Given marginals, finding a matrix that satisfies the conditions. In each of the sub-drawings, the numbers to the right and bottom represent the expected row-sums and column-sums (accordingly). Each one of those which is circled is already guarnteed in the present step of the solution. As can be seen, all the rows are taken care off from the setup, and the columns are solved-for one by one, from left to right.

A careful observer will notice that the requirement that $S \prec R^*$ guarantees that we only need to push elements to the right, and always have enough to satisfy the condition for the column-sum.

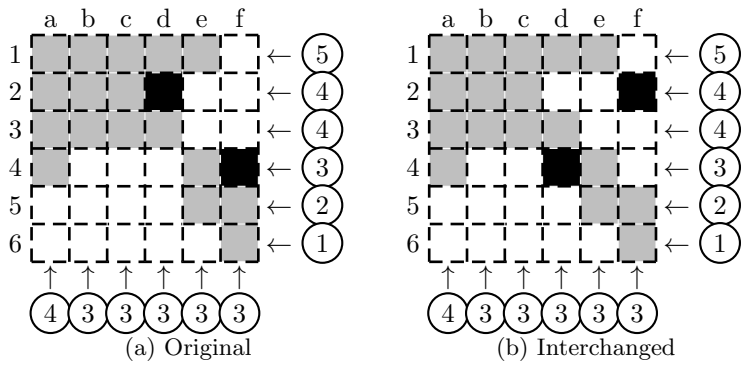
### 3.3 Uniqueness and Interchanges

Given a binary matrix, an interchange operation is demonstrated using the below two submatrices:

$$A_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad A_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Interchange operation creates a new matrix, yet keeps the marginals unchanged. Of course, interchange operation can be done on any relevant submatrix of a larger binary one. Ryser[4] proved that *any two* $(0,1)$*-matrices with the same marginal-distributions are transformable into each other by a finite sequence of such interchanges.*

For example, in our specific case, Fig. 3 demonstrates two possible solutions derived by interchange.



(a) Original (b) Interchanged

Figure 3: A 2D intercahnge is an operation that changes values in a matrix in such a way that the marginals remain the same.

To conclude this discussion, it is interesting to note that no general results are known for the cardinality of the solution space. There are results about bounds on the number of possible matrices, and even ways to enumerate these, but no direct calculation of the cardinality.

## 4. 3D DISCUSSIONS AND RESULTS

Unlike the 2D case which was investigated carefully on the theoretical side, the 3D case is mostly left to the practical approximate algorithms.[6] One of the ways to understand the reasons for that is by noting that in the 2D case we "pushed" extra 1's to the right in each step of the construction algorithm; In the 3D case, it is not clear which direction should the extra 1's go to. Another helpful way to look at it is through the numbers involved and the related computational complexities.

Assume we have a cube of size $w \times h \times d$ (Width $\times$ Height $\times$ Depth). The number of bits of information available in the cube is $w * h * d$, as each one can be either 0 or 1. When considering the marginals, we have three projections that yield three matrices:

- $w \times h$ matrix of integer values in range $[0, d]$. This amounts to $w * h * \log_2(d+1)$ [bits].
- $w \times d$ matrix of integer values in range $[0, h]$, leading to $w * d * \log_2(h+1)$ [bits].
- $h \times d$ matrix of integer values in range $[0, w]$, leading to $h * d * \log_2(w+1)$ [bits].

Again, to gain specific insight, we will follow with specific numerical values of $3 \times 4 \times 5$. For this case, the below follows:

$$\begin{aligned}
\text{Total number of bits in the cube} &: 3*4*5 = 60[\text{bits}] \\
\text{Number of bits in projection 1} &: 3*4*\log_2(5+1) = 31[\text{bits}] \\
\text{projection 2} &: 3*5*\log_2(4+1) = 35[\text{bits}] \\
\text{projection 3} &: 4*5*\log_2(3+1) = 40[\text{bits}] \\
\text{Total number of bits in projections} &: = 31+35+40 = 106[\text{bits}]
\end{aligned}$$

But this of course cannot hold, since we cannot get more information from the projections than from the original cube! The reason for the discrepancy is that the projections are not independent; in other words, not all combinations are allowed. For example, according to the "weight-conservation" principle, all the marginal matrices should have the same number of 1's in them.

One thing that should be apparent is that the complexity of the problem (enumeration for example) grows exponentially with the side of the cube. This fact, together with the fact there is no closed solution for the 3D case (as far as the authors are aware), may lead one to look for a way to limit the search space. Such approaches are common in the tomographic reconstruction community[6] , where assumptions about the connectivity and the content of the matrix are deployed.

In our case, since we are using the cube for coding information, and we have the freedom of creating the cube itself, we were able to devise a different novel solution to the problem of reconstruction from marginals.

## 5. PRACTICAL 2D AND 3D CODING ALGORITHM

Assume a 2D matrix of size $n \times n$. The number of bits of information available is $n^2$ [bits]. The following definition and observation are needed for the new algorithm.

DEFINITION 5.1. *A marginal sum-vector for a row (or column) of length n is called an* admissible vector *iff all its elements are integers in the range* $[0, n]$.

If a vector is admissible, it means it can be a row (or column) sum vector for a binary matrix of size $n$. This leads to the following observation

OBSERVATION 1. *Given an admissible vector, one can* always *create a binary matrix such that this vector is the row-sum of the matrix.*

Note that in the above observation, we did *not* specify the column-sum vector. Only one vector is predetermined.

This observation follows directly from the reconstruction algorithm we showed in the previous section. One method to finding the matrix is creating the set up construction matrix, where each row has enough 1 elements to satisfy the row-sum, starting from the very left column. This will create a valid matrix, with the required row-sum. Note that this matrix might not be unique, and all its elements are pushed to the top-left, but it *is* a valid solution.

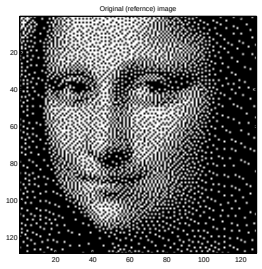
Therefore, our approach is to encode all the information in the row-sum vector, and since we produce it in a manner that guarantees the row-sum vector to be admissible, we are guaranteed there is a matrix to support it. Moreover, as we will show shortly, we choose a matrix which is aesthetically pleasing and that also satisfies the marginal row-sum condition.

Before we describe that phase of the algorithm, let us consider how many bits we can encode. Since each row-sum element can have $n+1$ values, the number of bits afforded is $n*\log_2(n+1)$. Admittedly, it is less than the $n^2$ in the original matrix, and is also less than the number of all possible marginals combinations, since we are focusing on only setting one-marginal for decoding.

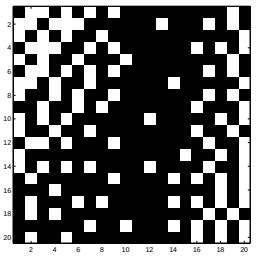
### 5.1 Ink-constrained image rendering for encoding

In the last section we showed that if we define one admissible marginal-sum vector, we can produce a corresponding matrix. In this section we show how we can choose a matrix which has certain features, and yet complies to the required row-sum requirement.

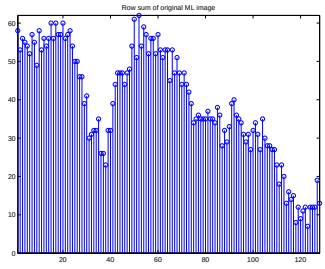
Assume we start from a black-and-white image as in Figure 4a. This image is $128 \times 128$ pixels of values $\{0, 1\}$.




(a) Original Mona Lisa, Black and White.



(b) Zooming on top-left corner.




(c) Plotting the row-sum vector. Note that Black pixels are taken as 0 in this case.

Figure 4: Image for encoding and its properties.

We then build an encoding table that encodes any input character into a number between 0 and 128 (since the maximum value of row-sum we have is 128). The sentence we chose to encode is (an example)

*"mona lisa drawn by leonardo da vinci. is she smiling? we will make this sentence about the right length... ... and added spaces!"*

Each of the characters corresponds to a value according to our coding table, and if we were to construct a matrix based solely on this number, and follow the simple construction of pushing all the 1's in each row to the left, we would get the image as in Figure 5a. This is a *valid* solution, albeit not visually pleasing.

An alternative method is to look at the number of ones needed to be encoded in each row, and at the number of ones present in the original image. If these two numbers happen to be the same, then we simply copy the row from the original image into our matrix. If the numbers differ (which will usually be the case), we need to either subtract, or add, 1's pixels to the original image in this row. There are various ways to do it, and in this paper we used a pure random selection to either add or subtract ones from the original image.

The resulting image is shown in Figure 5b, and can be contrasted with the brute force method, and with the original image.

Note: All the results were produced using Matlab, and the code and files are available from the authors.

As can be seen, the encoded image suffers degradation compared to the original, which is an expected result. Since we impose row-sum, often we will have to subtract (or add) 1's. However, one can clearly see the original image within the resulting noise. The difference as compared to the brute force method is obvious.

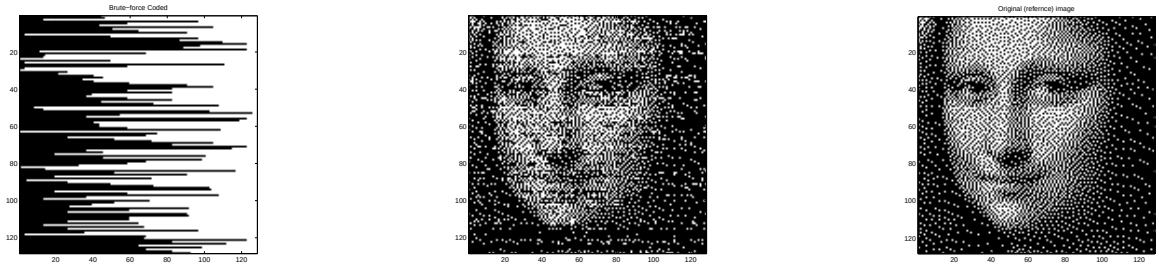
in Figure 6 we encode the *same* sentence in the image of David. Again, the degradation, and the quality remaining, can be easily compared.

### 5.2 3D case

The 3D case follows directly the 2D description above. The number of bits available in a cube of size $w \times h \times d$ is $w * h * d$ [bits].

If we determine the sum *ONLY* on one marginal, we are guaranteed a solution (as long as the marginal is admissible). Thus, assume we determine the marginal on the $w \times h$ side, which gives us $w * h * \log_2(d + 1)$ [bits].

And, similar to the 2D case, we can now use a given 3D shape or structure as a base for encoding, and modify the arrangement of modules there.

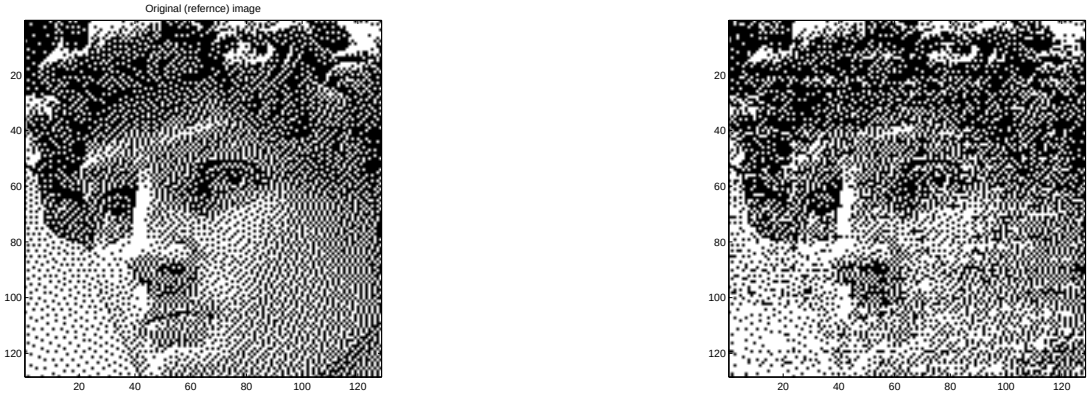



(a) Brute force matrix based on row-sum.

(b) Encoded image, where the original Mona Lisa's row-sums were modified to meet the desired sums.

(c) Original Mona-Lisa, Black and White.

Figure 5: Encoding process using randomization.




(a) Original Black-and-White image of David.

(b) Encoded image, where the original David's row-sums were modified to meet the desired row sums.

Figure 6: Encoding the same sentence in a differnet image: David.

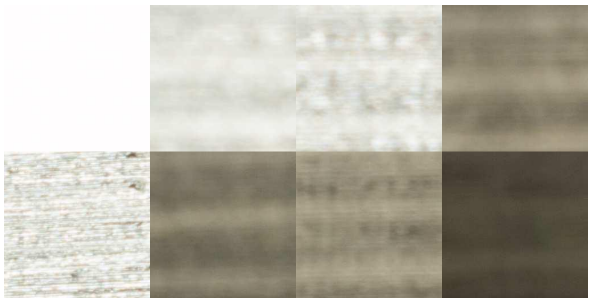
## 6. PRACTICAL IMPLEMENTATION

To validate the concept, we printed cube elements using 3D printer, and captured images of the marginals. Sample images and results are shown in Figure 7. For validation, rather than create a solid module, we created walls with known transmission of light. We then positioned these cells in different configurations, to simulate different number of modules in the cube. We projected light from one side of the cube, and measured the resulting image on the other side. As can be seen in the figure, the resulting gray level correlates linearly to the number of cells.

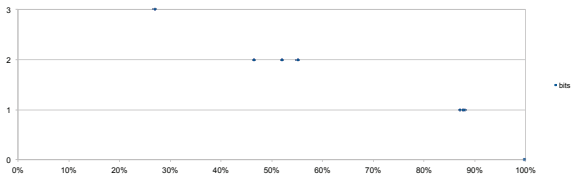
Though we used professional tools in this experiment (camera and lighting), it became apparent that neither collimated light nor special lensing (to avoid perspective distortions) are needed, and thus the mechanism is suitable for capturing by regular smartphone camera.

(a) 3D barcode measuring setup.

(b) Closer look at individual cells.



(c) 3D barcode measured images.




(d) Gray level decoding of images.

Figure 7: Measurement setup and results.

## 7. SUMMARY AND FUTURE DIRECTIONS

In this paper we introduced the idea of 3D barcodes. The decoding mechanism relies only on the marginal sums of the cube in 3 orthogonal projections. The equivalent concept in the 2D case is a binary matrix with known row and column sums.

We discussed the theoretical results and algorithms as pertained to the 2D case, and explained that the 3D counterparts are either missing or too computationally involved. We then proceeded to suggest a new encoding algorithm, which relies on the notion of admissible row-sum vector. Furthermore, we showed this algorithm can be combined to produce aesthetically pleasing results (matrices that resemble pre-defined images). This encoding scheme does not attain the maximum bit density possible, but falls close to it. We discussed the ink-constrained halftoning[11] , and demonstrated it's application.

Moving forward, there are a few interesting directions we wish to pursue. The most promising one is the notion of ink-constrained halftoning, especially as applied to the 3D case. Known methods like error-diffusion and dither-halftoning can be incorporated[12] as well as 3D additional constraints (for manufacturability for example).

Another direction of interest is the application to memory architectures, where the interrogation of information is done only through the edges. This might have applications in dense 3D memory architectures.

## ACKNOWLEDGMENTS

The support of the Singapore Ministry of Education through the grant MOE2013-T2-1-010 is gratefully acknowledged.

## REFERENCES

[1] Baharav, Z. and Kakarala, R., "Visually significant qr codes: Image blending and statistical analysis.," in [*ICME*], 1–6 (2013).

[2] Chu, H., Chang, C., Lee, R., and Mitra, N. J., "Halftone QR codes," *ACM Trans. Graph.* **32**(6), 217 (2013).

[3] Gale, D., "A theorem on flows in networks.," *Pacific J. Math.* **7**(2), 1073–1082 (1957).

[4] Ryser, H. J., "Combinatorial properties of matrices of zeros and ones," *Canadian Journal of Math* **9**, 371 – 377 (1957).

[5] Brualdi, R. A., "Matrices of zeros and ones with fixed row and column sum vectors," *Linear Algebra and its Applications* **33**(0), 159 – 231 (1980).

[6] Herman, G. and Kuba, A., [*Discrete Tomography: Foundations, Algorithms, and Applications*], Applied and Numerical Harmonic Analysis, Birkhäuser Boston (1999).

[7] da Fonseca, C. and Mamede, R., "On (0, 1)-matrices with prescribed row and column sum vectors," *Discrete Mathematics* **309**(8), 2519 – 2527 (2009).

[8] Snijders, T., "Enumeration and simulation methods for 0-1 matrices with given marginals," *Psychometrika* **56**(3), 397–417 (1991).

[9] Denso Wave Inc., "Qr code specification." `http://www.qrcode.com/en/about/standards.html` (2014).

[10] Barvinok, A., "Matrices with prescribed row and column sums," *Linear Algebra and its Applications* **436**(4), 820–844 (2012).

[11] Bayeh, M., Compaan, E., Lindsey, T., Orlow, N., Melczer, S., and Voller, Z., "Ink-constrained halftoning with application to qr codes," *Proc. SPIE* **9015**, 90150U–90150U–8 (2014).

[12] Ulichney, R., [*Digital Halftoning*], MIT Press, Cambridge, MA (1987).

[13] Dietze, S., Riedrich, T., and Schmidt, K. D., [*On the solution of marginal sum equations*], DVSM, Dresden (2006).

[14] Barvinok, A., "Matrices with prescribed row and column sums.," *Linear Algebra Appl.* **436**(4), 820–844 (2012).