Embedding QR Codes onto B-spline Surfaces for 3D Printing

Ryosuke Kikuchi, Sora Yoshikawa

Department of Mechanical Engineering, Yokohama National University, Japan

Pradeep Kumar Jayaraman, Jianmin Zheng

School of Computer Science and Engineering, Nanyang Technological University, Singapore

Takashi Maekawa

Department of Mechanical Engineering, Yokohama National University, Japan

Abstract

Recent advance of Additive Manufacturing technologies allows us to manufacture various parts used in real-world products. Consequently, product tracking of such 3D printed parts is an important issue. Quick Response (QR) code which is a two-dimensional matrix barcode invented by Denso, a Japanese automotive industry, in 1994, can be used for this purpose. It can store more data than the 1D barcode in a smaller space, and using a smartphone as a scanner, one can directly visit a website where all the information of the parts is stored. However, QR codes require secondary procedures to add them to products and are also vulnerable to wear and tear. Moreover, QR codes cannot be added to freeform surfaces, but only to developable surfaces. In this paper we propose a novel technique to embed QR codes onto CAD models consisting of freeform surfaces represented by B-spline surfaces, which produces 3D QR codes. 3D QR codes work similar to 2D QR codes and can be read by existing QR scanners, but are designed by grooving the surface to obtain light and dark regions caused by ambient occlusion. Unlike conventional QR codes, 3D QR codes do not fall off from the part and can even be painted if necessary. Furthermore, we do not need to prepare dark-colored and light-colored materials for 3D printing as the dark color is provided by the grooving. We demonstrate the effectiveness of our technique with various examples.

Keywords: QR code, B-spline surface, 3D printing

1. Introduction

This paper considers the problem of embedding Quick Response (QR) codes onto B-spline surfaces for 3D printing. A QR code is a two-dimensional matrix barcode invented in 1994 by Denso, a Japanese automotive industry [1]. It consists of collections of unit black squares arranged in a white square matrix and can store more data than 1D barcodes in a smaller space. QR codes support numeric and alphabetic characters, Kanji, Kana, Hiragana, symbols, binary, and control codes [1]. They also have capability of restoring data even if they are partially damaged owing to Reed-Solomon error-correcting codes. If a smartphone is used as QR code scanner, one may directly connect to a website where all the required information is stored. B-splines are a common representation for freeform surfaces. Many CAD models consist of freeform surfaces represented by B-splines. Hence a good solution to the problem is very useful in practice.

In particular, additive manufacturing (AM) or 3D printing has achieved significant progress. Various AM technologies such as fused deposition modeling (FDM), selective laser sintering (SLS), stereolithography (SLA), indirect inkjet printing (binder 3DP) and laminated object manufacturing (LOM) have been developed [2, 3]. Different from conventional manufacturing processes such as lathing, milling, drilling, and grinding which are based on subtractive manufacturing, AM is a layer-wise manufacturing technology which enables us to fabricate objects with complex geometry and topology using a wide variety of materials [2]. The usage of 3D printed parts in automotive, aerospace, and medical industries, where safety issues are critical, is actually growing rapidly. As a result, tracking of such 3D printed parts and their past inspection records in the periodic inspection becomes highly important. It allows us to quickly track products and match replacement parts in case of failures. QR codes may provide a tool for this purpose.

Recently, Aprecia Pharmaceuticals Company announced that the U.S. Food and Drug Administration (FDA) has approved the first prescription drug product manufactured using 3D printing technology [4]. If we can embed medical prescriptions onto the drugs via QR codes, patients can easily access them by just scanning over the drugs by smartphones.

However, conventional QR codes usually require secondary procedures to add them to products and are also vulnerable to wear and tear. Moreover, QR codes cannot be pasted on a freeform surface like the wave-like shape shown in Figure 1, but only to developable surfaces. In this paper we propose a novel technique to embed QR codes onto CAD models having freeform surfaces represented by B-splines, which produces 3D QR codes, as illustrated in Figure 1. 3D QR codes are constructed by grooving the surface corresponding to the collection of black squares of the 2D QR code while leaving the surface corresponding to the white region as it is. The deeper the groove is, the darker it

*Corresponding author
Email address: maekawa@ynu.ac.jp (Takashi Maekawa)

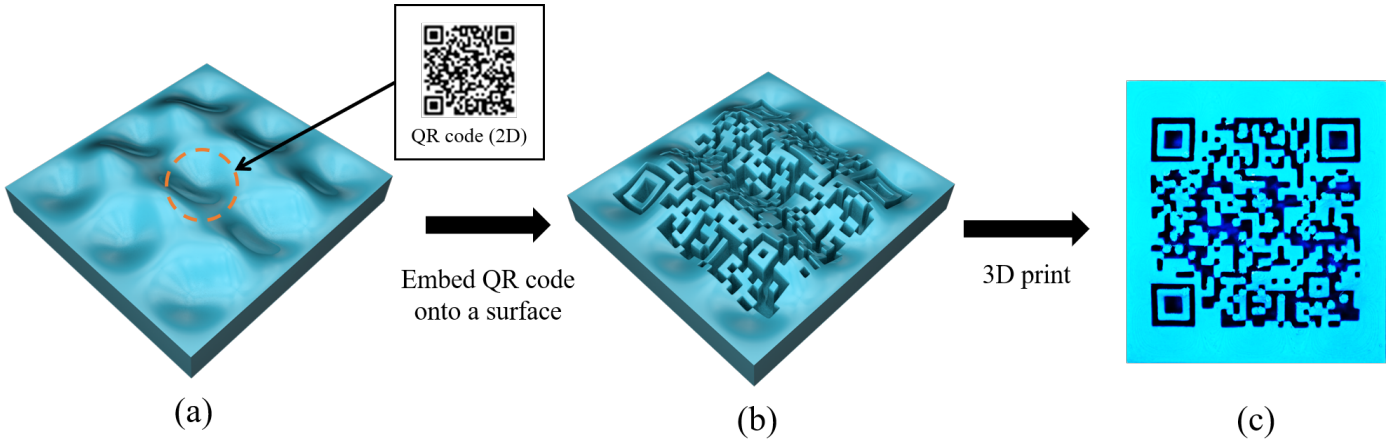


Figure 1: Embedding a QR code onto a freeform B-spline surface: Given the 2D QR code image and the target position on the B-spline surface (a), we embed the QR code directly onto the surface by grooving the surface (b) such that the parts of the surface corresponding to the black pixels of the QR code appear dark due to ambient occlusion. As a result, when smartphones or existing QR scanners scan the 3D printed surface from the top, they will see the image of the 3D QR code shown in (c).

appears. In other words, single-colored material is enough to represent the QR codes. The groove depth of each area, within the black region, is determined by evaluating the obscurance [5, 6] as a measure so that the area is dark enough to be recognized as the black area when seen from the top of the 3D QR code.

The grooving is realized by offsetting the surface inwards. In this way, the 3D QR codes are automatically included in the new generated B-spline surface. If an object is represented by NURBS/B-spline surfaces, which are the de facto industry standard for representing complex freeform geometry in CAD/CAM/CAE field, we can directly embed 3D QR code onto it without converting to a tessellated model. This advantage allows designers to change their design in NURBS/B-spline representation even if the 3D QR code is already embedded. Moreover, the output file is in the same format and can be used for other steps such as isogeometric analysis, and direct slicing [7, 8]. Hence, our method upholds the clear separations between design, analysis and fabrication phases.

The proposed 3D QR codes can also be embedded onto metal or glass parts by investment casting [9]. In investment casting a wax 3D printer is used to create parts with 3D QR code on it. Then the parts are dipped into a wet ceramic slurry repeatedly until a desired thickness is achieved and dried out completely to produce ceramic molds known as the investment. They are turned upside-down and placed in a furnace to melt out the wax. The metal is poured and cooled down. Finally the ceramic shell is hammered to release the casting.

The main contributions of the paper are as follows.

- We introduce a new concept called "3D QR codes", a 3D analogue of traditional 2D QR codes, which enables us to print directly on the freeform parts during the 3D printing process without any secondary procedures in adding it. Moreover it does not fall off from the part, and can even be painted.

- We propose an algorithm to amend the freeform B-spline surface to create the 3D QR code by grooving, which is realized by simply refining the knots and offsetting the surface inwards, i.e., along negative normal direction. The grooving depth (offset distance) is controlled by computing the obscurance [6], around each point within the black modules of the QR code by considering the surface around it. Accordingly, the parts can be printed with a single-colored material, whereas conventional QR code requires dark color for code and must be placed against a light-colored background.

We demonstrate the applicability of our method with several examples and evaluate 3D QR codes with various experiments.

2. Related work

QR codes are becoming ubiquitous in many applications including advertisement, marketing, document management, and manufacturing owing to the widespread use of smartphones. They provide a convenient way to communicate between mobile devices and their surrounding cyber-physical world. For example, QR codes have been used in a robot system for handling various objects in office or home environments [10]. Technical survey on QR codes is given in [1, 11] and readers are referred to the references therein. This section just briefly reviews some QR codes that are relevant to our work.

The design of early QR code generators mainly focused on achieving high decodability. As a result, QR codes typically appeared as random collections of black and white squares, which usually lack user friendly appearance [12, 13, 14]. Wang et al. proposed to embed a QR code in a color image to make the code colorful and more readable [15]. Chan [12] pointed out that multiple colors could be used and a color gradient could be applied without affecting the scanability as long as dark color was used for the code and placed against a light-colored background. Moreover, the corners and edges could be rounded to soften up the appearance to make it more friendly and approachable. Chan [12] also suggested to add an eye-popping logo in the center of the code by taking advantages of the fact that up to 30% of a QR code's data could be obstructed.

To further improve visual appearance or add visual semantics, Cox [13] developed an algorithm to encode image content as redundant numeric strings appended to the original data, which embedded a binary picture into a QR code. Chu et al.

[14] presented a method to combine halftone images with QR codes to make them visually pleasing without compromising their readability. The method converts a given object image into a halftone image and then estimates a saliency map to guide the QR code generation. Since then, various methods have been proposed, which on one hand make QR codes machine readable robustly and on the other hand make QR codes visually pleasant [16, 17, 18, 19, 20, 21, 22]. In particular, Lin et al. proposed a two-stage approach to generate QR codes, in which the first stage synthesizes a baseline QR code with reliable decodability but poor visual quality based on Gauss-Jordan elimination and the second stage designs a rendering mechasim to improve the visual quality while keeping the decodability [18]. The method can enhance the appearance of the QR code and reduce the processing complexity. Yang et al. proposed ARTcode: Adaptive Robust doT matrix barcode, which aims to improve both viewing experience and data communication accuracy in one visual pattern [20]. Duda et al. [21] proposed a solution as a generalization of the Kuznetsov-Tsybakov problem, which allows QR codes to resemble an arbitrary image or a logo.

AirCode [23] is an alternate approach which encodes information in air pockets placed beneath the object surface. The embedded information can be detected by separating the direct and global components of the light transport, where the latter conveys the influence of the subsurface air pockets corresponding to non-zero bits in the code. While this method is very useful if the user wants to conceal information or does not want to change the surface geometry for aesthetic reasons, it additionally requires a projector and special reading algorithm for decoding, which implies that existing smartphones cannot be trivially used as a reader as in the case of QR codes. Additionally, this method only works on material that allows subsurface scattering of light. On the other hand, it is quite common to fabricate objects using fully opaque materials in practical manufacturing applications.

Different from existing methods that generate 2D QR codes for planar faces, our work focuses on CAD models consisting of nonlinear B-spline surfaces. Our method can use the above-mentioned methods to generate visual pleasant QR codes, and then groove the 3D surface to embed the codes to create 3D QR codes by utilizing the fact that grooved regions of the surface appear dark compared with the non-grooved regions when they are viewed from the surface normal direction.

3. Overview of the Proposed Method

Our method requires three inputs to generate 3D QR codes:

(i) A *QR Code* that can be generated by any existing QR code generator. The output of the generator is a square binary image with a set of black and white pixels called modules. The image dimension typically varies from 21×21 to 177×177 in steps of 4 [1]. The length of the square matrix of the QR code is l. The size of the module is given by $l_m = \alpha P_r$ where P_r is 3D printer's resolution and α is a constant.

(ii) A *freeform B-spline surface* $\mathbf{r}(u, v)$ which is an order (K, L) tensor product surface defined by a topologically rectangular set of control points $\{\mathbf{P}_{ij} | 0 \le i \le n, 0 \le j \le m\}$ and two knot vectors $\mathbf{U} = (u_0, u_1, \ldots, u_{n+K})$ and $\mathbf{V} = (v_0, v_1, \ldots, v_{m+L})$ [24, 25]. In this paper, we assume that the surface is a bi-cubic B-spline surface. The method can be extended to any degree surfaces.

(iii) A *3D point* lying on the surface $\mathbf{r}(u_o, v_o)$ which is specified by the user to represent the target position where the center of the QR code image will be projected onto.

The output is a new 3D B-spline surface that conveys the 3D QR code. The method consists of the following five steps.

Step 1. Preprocess QR codes

To embed the QR code onto the surface, we define the QR code knot lines that will be used for knot insertion, as well as ensure that the QR code smoothly joins with the rest of the surface without causing artifacts (see Figure 2).

QR Code Knot Lines. We first attach a Cartesian coordinate system C_{QR} to the QR code image such that the origin is at the center of the image as shown in Figure 2(a). Next, we generate a 5×5 non-uniform grid on each module where the distance between the grid points are of ratio 1:4:4:1 yielding 16 non-uniform cells, as shown in the closeup view in Figure 2(b). This grid represents knot lines that will be inserted later during the construction of the 3D QR code. If a grid point is surrounded by four black cells, we flag the grid point as an offset point. The offset points are used to groove the surface inward so that the black modules appear darker on the surface due to ambient occlusion.

Buffer & Transient Zones. A buffer zone of length l_b is added to 4 sides of the QR code (see Figure 2(c)) so that the surface outside the black modules will not be dragged inward. We use $l_b = 0.01l$ for all results shown in this paper. As illustrated in Figure 2(c), insertion of the squared control net into general freeform surface generates sudden change in the isoparametric line directions. Therefore, we additionally add a transient zone outside of the buffer zone of QR code by inserting knots to relax the sudden change in the surface shape due to the embedding procedure.

Step 2. Build a 3D QR coordinate system

We build a 3D Cartesian coordinate system with the origin locating at the user-specified 3D point on the surface and the x, y axes aligning with the directions of iso-parameter lines as much as possible. The 2D QR code will be mapped to the xy-plane of the coordinate system.

Step 3. Generate knot lines for the 3D QR code

By projecting the grid of the 2D QR code onto the B-spline surface, we compute the knot values for the 3D QR code. We further adjust these knot values to ensure that they are on the respective isoparametric lines. The final set of grid points corresponding to the 3D QR code is shown in Figure 3. The $\mathbf{U}$ knot vector of the 3D QR code can be indexed as:

$$u_A = u_{\hat{i}}, \qquad u_B = u_{\hat{i}+N_{TZ}},$$
$$u_C = u_{\hat{i}+3+N_{TZ}}, \qquad u_D = u_{\hat{i}+3+N_{TZ}+N_{QR}},$$
$$u_E = u_{\hat{i}+6+N_{TZ}+N_{QR}} \qquad u_F = u_{\hat{i}+6+2N_{TZ}+N_{QR}},$$

where $\hat{i}$ is the knot index for knot u_A, N_{QR} and N_{TZ} are the number of knot intervals of the QR code and transient zone, respectively. The knot vector $\mathbf{V}$ of the 3D QR code can be indexed similarly, and is denoted by $v_G{=}v_{\hat{j}}$, v_H, v_I, v_J, v_K, v_L where $\hat{j}$ is the knot index for v_G.

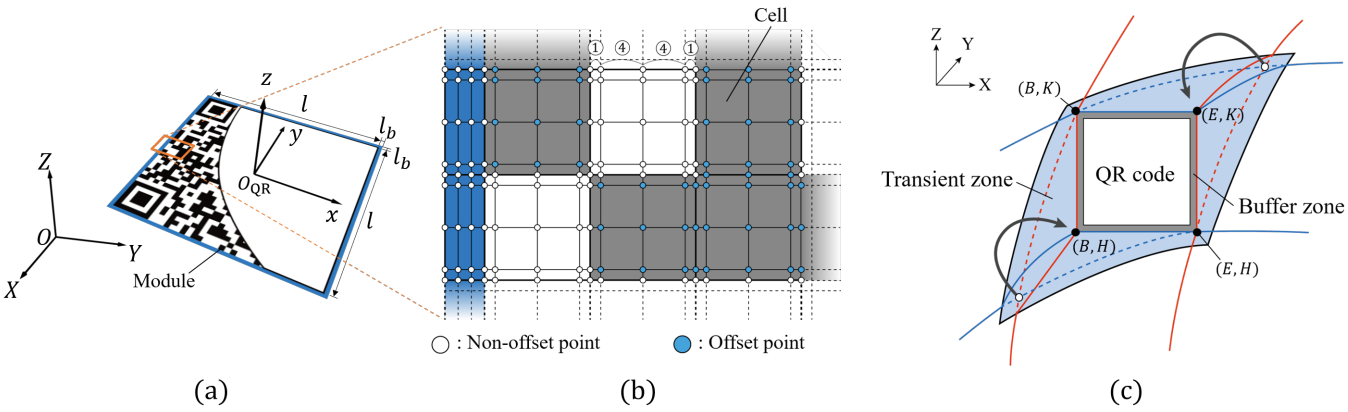


Figure 2: QR Code Knot Lines and Buffer & Transient Zones: (a) We first attach a coordinate system C_{QR} to the QR code image, and (b) generate a grid that will serve as knot lines for the 3D QR code. (c) Next, we add buffer and transient zones around the QR code region in the B-spline surface to avoid artifacts. Dotted lines are isoparametric lines passing through points corresponding to pre-images of (B,H) and (E,K) before 3D QR code insertion, while solid lines are isoparametric lines after the insertion. (B,H) and (E,K) are defined in Figure 3.

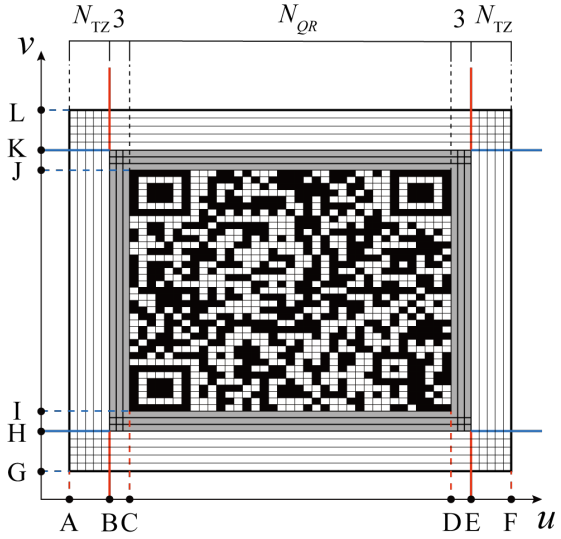


Figure 3: u-v parameter space of 3D QR code along with labeled knot lines.

Step 4. Refine the B-spline surface

Based on the knot lines for the 3D QR code, we refine the B-spline surface by knot insertion. This produces a required topological structure for 3D QR code. B-spline control points corresponding to the QR code zone, buffer zone and transient zone are also adjusted.

Step 5. Groove the B-spline surface

To create geometry representing the 3D QR code, we offset those B-spline control points corresponding to the black QR modules.

The details of steps 2-5 will be given in Section 4 and the computation of the offset distance in step 5 will be discussed in Section 5.

4. 3D QR Code Generation

We now discuss the procedure to generate the 3D QR code, which is illustrated in Figure 4.

4.1. 3D Cartesian coordinate system for the 3D QR code

The central position of the QR code to be embedded on B-spline surface at $\mathbf{r}(u_o, v_o)$ is specified by the user's mouse click on the surface in our graphical user-interface. Our goal now is to determine a suitable plane of projection from which the QR code can be projected onto the surface.

A set of $\bar{m} \times \bar{m}$ samples are then generated on the B-spline surface centering at $\mathbf{r}(u_0, v_0)$ as shown in Figure 4 (a). The sampling interval Δu along the u-direction is obtained by Taylor expansion:

$$|\mathbf{r}(u_o + \Delta u, v_o) - \mathbf{r}(u_o, v_o)| = \frac{l}{\bar{m} - 1}$$
$$\Longrightarrow \Delta u = \frac{l}{|\mathbf{r}_u(u_o, v_o)|(\bar{m} - 1)}$$

The interval Δv along the v-direction is obtained similarly.

To establish a local coordinate system centered on $\mathbf{r}(u_o, v_o)$, we apply least squares fitting to the samples, which amounts to applying principal component analysis (PCA) (see Figure 4(b)). The first principal component (eigenvector) associated with the largest eigenvalue, which represents the most variance, is the local x-axis. The second principal component associated with the second largest eigenvalue is the local y-axis, and the third principal component associated with the smallest eigenvalue is the local z-axis which coincides with direction of the normal vector of a plane which we denote as $\mathbf{\Gamma}$.

Tangent vectors $\mathbf{r}_u(u_i, v_i)$, $\mathbf{r}_v(u_i, v_i)$, where $i = [0, \ldots, \bar{m}^2 - 1]$ are normalized and averaged to obtain unit vectors $\hat{\mathbf{r}}_u(u_o, v_o)$ and $\hat{\mathbf{r}}_v(u_o, v_o)$ which lie on $\mathbf{\Gamma}$. We compute the final local x-axis and y-axis by rotating their bisector vector around the normal vector by $-45°$ and $45°$, respectively according to the right hand rule. If $\hat{\mathbf{r}}_u(u_o, v_o)$ and $\hat{\mathbf{r}}_v(u_o, v_o)$ are orthogonal, they just correspond to the local x-axis and y-axis. Finally we have the Cartesian coordinate system C_{3DQR} at $\mathbf{r}(u_o, v_o)$ as illustrated in Figure 4 (c).

We define the unit vectors along the positive x, y, and z axes of C_{3DQR} as $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$, respectively. All the grid points on all the modules defined on C_{QR} are transformed to C_{3DQR} and denoted as $\mathbf{\Gamma}_i$ (see Figure 4 (d)).

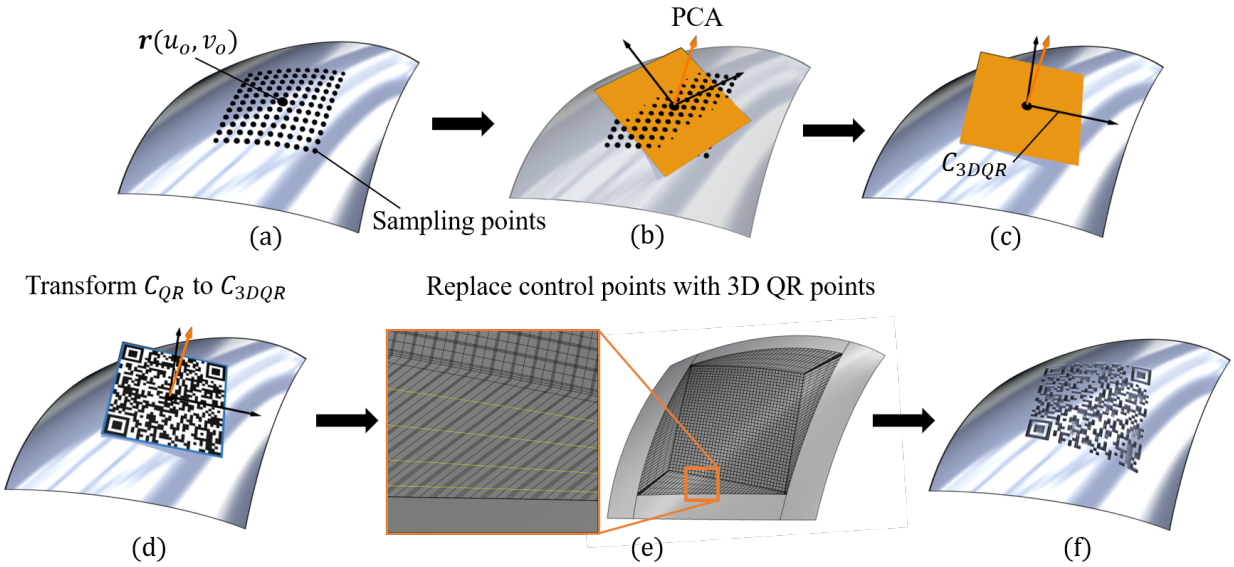


Figure 4: 3D QR code construction: (a) Central position of the QR code to be embedded is specified and points are sampled on the B-spline surface to estimate a suitable plane of projection for the QR code; (b) PCA is applied to the sampling points to determine the local x y, and z axes; (c) The final axes defining the C_{3DQR} coordinate system are obtained by rotating x and y axes obtained in (b) about the z axis; (d) All the grid points on all the modules defined on C_{QR} are transformed to C_{3DQR}; (e) Knot insertion is performed to generate control points corresponding to the 3D QR code points (closeup view shows isoparametric lines in the transient zone); (f) Finally the 3D QR code is embedded on the B-spline surface by grooving the surface inwards.

4.2. Knot Values for the 3D QR Code

We orthogonally project the points $\mathbf{\Gamma}_i$ onto the surface $\mathbf{r}(u, v)$ by solving the following vector equation:

$$\mathbf{r}(u_i, v_i) = \mathbf{\Gamma}_i - t_i \mathbf{k}, \tag{1}$$

to obtain $\{(t_i, u_i, v_i) \mid \forall i \in [1, \ldots, (N_{QR} + 7)^2]\}$. Here t_i is the parameter along the projected direction where $\mathbf{\Gamma}_i$ is projected onto, and (u_i, v_i) are the corresponding parameter values of the projected points on the surface. Let us denote these orthogonally projected points as $\mathbf{\Delta}_i$, $i = 1, \ldots, (N_{QR} + 7)^2$.

In general, these knot values (u_i, v_i) do not lie on the isoparametric lines. Hence, the u knot values for example, in Figure 3 along the columns of B to E, are not constant in general. Hence, we find the maximum and minimum of u knot values along each column, and set the knot value of u_B (refer to Figure 3 for notation) to the minimum value along the column B, and u_E to the maximum value along the column E. The rest of the knot values u_k corresponding to each column between B and E are obtained by linear interpolation:

$$u_k = \left(1 - \frac{k - (\hat{i} + N_{TZ})}{N_{QR} + 6}\right) u_{k_{min}} + \frac{k - (\hat{i} + N_{TZ})}{N_{QR} + 6} u_{k_{max}},$$

where $u_{k_{min}}$ and $u_{k_{max}}$ are the minimum and maximum u values along column u_k.

Similarly, we obtain for v knot values by setting v_H equal to the minimum value along the row H, v_K equal to the maximum value along the row K, and linearly interpolating the rest of the knot values for each row:

$$v_k = \left(1 - \frac{k - (\hat{j} + N_{TZ})}{N_{QR} + 6}\right) v_{k_{min}} + \frac{k - (\hat{j} + N_{TZ})}{N_{QR} + 6} v_{k_{max}}.$$

Knot values for the transient region u_A and u_F are set to $u_B - \delta$ and $u_E + \delta$, respectively. In this paper we empirically used $\delta = 0.004$.

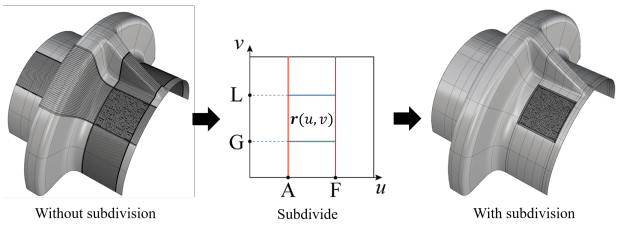


Figure 5: The original B-spline surface is subdivided into five patches to restrict knot lines from unnecessarily traversing the entire surface.

Similarly, knot values for v_G and v_L are set to $v_H - \delta$ and $v_K + \delta$, respectively.

4.3. B-spline Surface Refinement

There is a concentration of knots on the B-spline surface in the region of 3D QR code, see Figure 5. Näive knot insertion will result in a large number of control points which extend throughout the surface due to the B-spline's tensor product nature; hence, we split the surface into five B-spline patches as illustrated in Figure 5 using knot refinement [24] along the border of the transient zone, such that the patch in the middle represents the portion where the 3D QR code will be embedded. Hereafter, we will use $\mathbf{r}(u, v)$ to denote this middle patch.

We insert knots between B and E, and H and K to generate control points for 3D QR code. The resulting control points by this knot insertion operations are replaced by $\mathbf{\Delta}_i$, $i = 1, \ldots, (N_{QR}+7)^2$. Note that this replacement does not induce artifacts since we have a large number of control points concentrated in the small area of 3D QR code, and the control points are very close to the surface. We finally insert knots in the transient zone uniformly, namely between A and B, E and F, G and H, and K and L. Then the

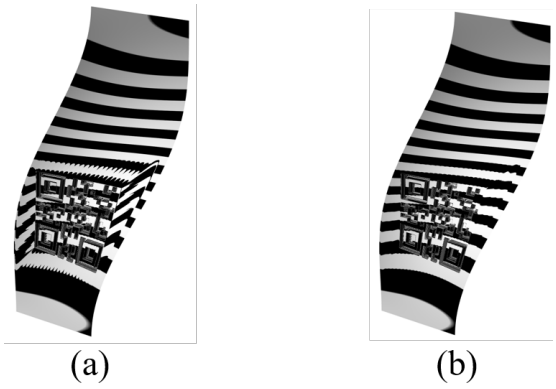


Figure 6: Importance of transient zone in maintaining surface fairness is illustrated using zebra mapping: (a) without transient zone, (b) with transient zone.

control points corresponding to the inserted knots in the transient zone are orthogonally projected onto $\mathbf{r}(u, v)$. The control points resulting from the knot insertion in the transient zone are replaced by these orthogonally projected points. Figure 4 (e) shows isoparametric lines focusing on the transient zone.

4.4. Grooving the B-spline Surface

To finally embed the QR code, we offset the flagged projected points corresponding to the black QR modules which currently lie on $\mathbf{r}(u, v)$ by the distance $d_{off} = l_m$ along $-\mathbf{k}$ in the C_{3DQR} coordinate system. This process grooves the surface inwards and thereby embeds the QR code. We denote the final embedded surface as $\bar{\mathbf{r}}(u, v)$ (see Figure 4 (f)).

To illustrate the importance of the transient zone, we compare the surface fairness with and without the transient zone by visualizing a zebra map which is depicted in Figure 6.

5. Obscurance

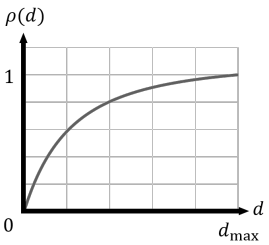
So far, we have embedded the QR code onto the surface by grooving the offset points inward by a fixed distance l_m. However, it is possible that this distance is insufficient to generate regions that are dark enough for the QR code scanner to recognize. Hence, we evaluate the obscurance at the grooved points to determine if the the area is dark enough to be recognized as black modules when seen from the top of the 3D QR code.

The obscurance of a point P is defined as the following integral [5, 6]:

$$W(P) = \frac{1}{\pi} \int_{\omega \in \Omega} \rho(d(P, \omega)) \cos\theta d\omega, \tag{2}$$

where $W(P)$ is integrated over the hemisphere Ω, $d(P, \omega)$ is the distance from P to the first intersection in direction ω (see Figure 7),

$\rho(\cdot)$ is an empirical monotonic increasing function which squashes d in the range $[0, 1]$ as shown in the inset figure, θ is the angle between direction ω and the normal at P, $1/\pi$ is the normalization factor such that if



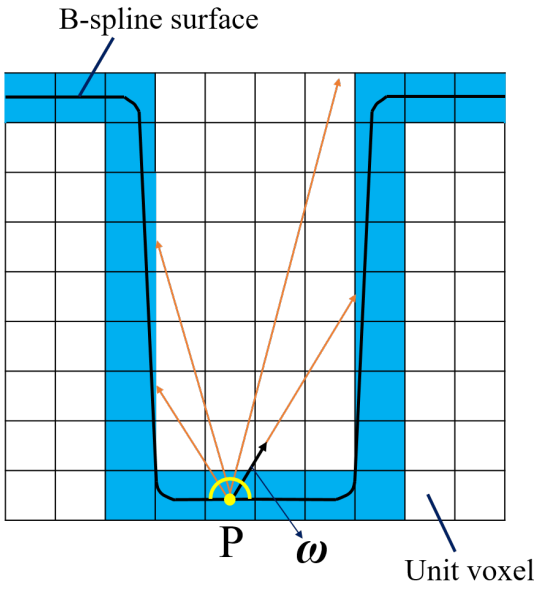


Figure 7: Obscurances, which are amount of occlusions around each point P, are evaluated by computing the distance of the rays emanating from P in direction ω to the B-spline surface approximated by voxels.

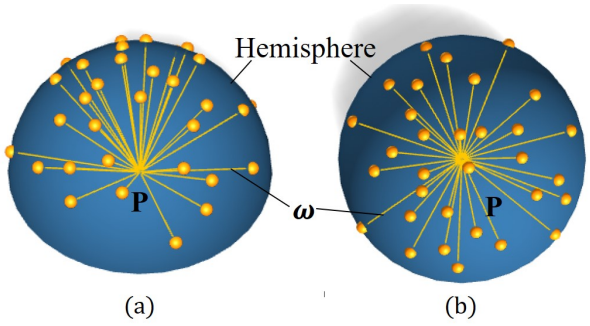


Figure 8: Halton sequences are used to determine the light directions over the hemisphere. (a) Oblique view, (b) top view.

$\rho(\cdot) = 1$ over the whole hemisphere then $W(P) = 1$. An obscurance value of 1 means the point is completely open. Considering the inverse-square law, we define $\rho(d)$ as:

$$\rho(d) = \begin{cases} 1 - \left(\frac{1+b}{(ad+1)^2} - b\right) & \text{for } d \le d_{max} \\ 1 & \text{otherwise,} \end{cases} \tag{3}$$

where,

$$a = \frac{\sqrt{\frac{1+b}{b}} - 1}{d_{max}},$$

and we set b to 0.1. The purpose of ρ is to consider intersections along ω within a limited neighborhood around P [26].

The obscurance integral (2) can be evaluated using Monte Carlo evaluation. Casting rays from P with uniform direction weighted by $\cos\theta$ can be considered using $\frac{1}{\pi}\cos\theta$ as a probability density function, and hence the obscurance integral (2) can be evaluated by sampling N rays as follows [6]:

$$W(P) = \frac{1}{N_{ray}} \sum_{i=1}^{N_{ray}} \rho(d_i(P, \phi_i, \theta_i)). \tag{4}$$

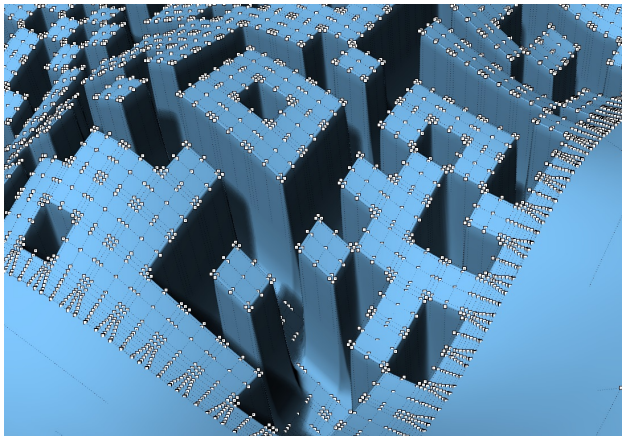


Figure 9: Closeup view of the lower right part of the wave-like surface with its control points.

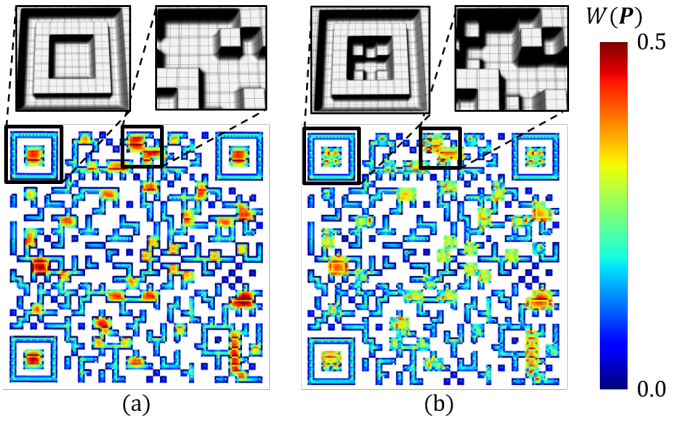


Figure 10: Color-coded obscurance map, (a) without additional offset, (b) with additional offset. Top row images show the closeup views of the 3D QR code squared in the color-coded map.

The direction of the ray $\boldsymbol{\omega}$ is determined by the azimuth angle ϕ_i and polar angle θ_i:

$$\phi_i = 2\pi\xi_i,$$
$$\theta_i = \sin^{-1}\sqrt{\zeta_i},$$

where ξ_i and ζ_i are obtained using quasi-random Halton sequences [27] (see Figure 8. Note that (4) varies between 0 and 1).

To speed up the distance computation of each ray from P to the intersection on the surface, we voxelize the region of the 3D QR code represented by B-spline surface. The union of voxels that enclose portions of the B-spline surface (see blue voxels in Figure 7) are used in the distance computation, which is performed by a fast and simple voxel traversal algorithm [28].

Local offset points. Once the obscurances $W(P)$ of all the offset (grooved) points are evaluated using Eqn. (4), we count the number of points N_{lim} which satisfies $W(P) < Ob_{lim}$ where Ob_{lim} is the prescribed limit for the obscurance. Although we can repeat the offsetting process until all the offset points satisfy $W(P) < Ob_{lim}$, some of the isolated pillars shown in Figure 9 may become very tall and become vulnerable to shear force on the surface due to high bending stress at the fixed end. Generally

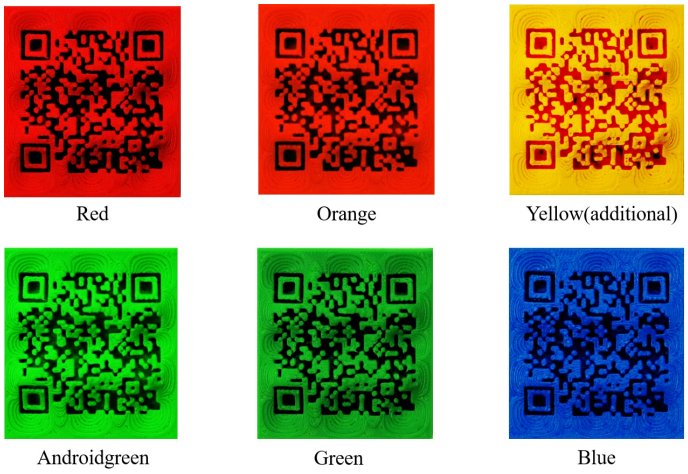


Figure 11: Scannability test for six different colors. Skyblue color is given in Figure 1. We note here that we have edited the photos of the 3D printed objects throughout this paper using the Levels tool in Adobe Photoshop to ensure the high contrast between the groove and light-colored surface so that they are scannable on the screen.

in the vicinity of such isolated pillars, the area is highly occluded and hence there is no need to groove deeply as shown in Figure 9. Accordingly, if N_{lim} is smaller than half the number of entire offset points, we further offset points with $W(P) > Ob_{lim}$ by l_m, until N_{lim} becomes larger than half the number of the entire offset points. Once N_{lim} is larger than half the number of entire offset points, we offset other points locally within the black modules where the condition $W(P) < Ob_{lim}$ is not satisfied by d_{off}=l_m for each iteration until the additional number of offsets is equal to the maximum number of offsets N_{offmax}. Figure 10 shows the color coded obscurance map before and after the local offsetting with N_{offmax}=2, where it is apparent that local offsetting leads to increased obscurance (lower $W(P)$).

6. Results

6.1. Potential Applications

We implement and evaluate our method on a PC with an Intel Core i7-6600U (2.60 GHz) processor with 16 GB of RAM. We demonstrate the applicability of our 3D QR code by applying it to four models (see Figure 12), namely:

(i) A FLANGE which is a mechanical part where the QR code can be used for part tracking. We 3D print this model using a Zortrax M200 FDM machine with a printing resolution of P_r=0.4[mm] with ABS plastics. The 3D QR code is placed on the cylindrical part with a module size l_m = 1.2[mm]. We began with Ob_{lim}=0.2 and resulting in offsetting twice without any local offsets resulted in an unreadable code. Two additional local offsets enabled the code to be read by both iOS and Android devices. We were also able to perform a successful scan with a lower value of Ob_{lim}=0.15 which resulted in four offsets without any local offsets.

(ii) A wax PERFUME bottle, which can be used for investment casting. We placed the 3D QR code on a doubly curved

region of the surface. As a wax 3D printer was not available, we fabricated the master perfume bottle using a Zortrax M200 FDM machine, and placed it inside a open box where the silicone was slowly poured, and cured. Finally, melted wax was poured into the silicone mold to obtain the wax perfume bottle. The module size was set to $l_m = 1.19$[mm] to make the demolding process easy. Four offsets with $Ob_{lim} = 0.15$ allowed us to read the code without any local offset operations.

(iii) A heart shaped CHOCOLATE model, which can be used as a mold for making pastries. We realized this model using the same procedure as that of the perfume model. The module size was set to a higher value of $l_m = 1.96$[mm] considering the fragility of the chocolate. Four global offsets with $Ob_{lim} = 0.13$ allowed us to read the code without any local offset operations.

(iv) A tiny pharmaceutical TABLET, where prescription information can be stored in the QR code. Unfortunately a high resolution 3D printer was not available, hence we printed the drug model by Zortrax M200 FDM machince yielding a 50 mm diameter tablet (which is not possible to swallow). Nevertheless, our emphasis here is that with a higher resolution printer, it is possible to embed medical prescriptions onto drugs.

Table 1 shows the computational statistics corresponding to all the models considered here. The last column is a measurement of illuminance in lux (lx) at the 3D QR code. Since recommended light levels in an office, classroom, library is typically 500∼700 lx, we can say that our 3D QR code can be read in an ambient lighting condition.

6.2. Readability Test for Various Material Colors

Chan [12] suggested that as long as dark color is used for the code and placed against a light-colored background, scannability is not affected. Based on this, we printed the WAVE-like surface with seven different colors, namely, red, orange, yellow, android green, green and blue as shown in Figure 11 (skyblue color shown in Figure 1) using a Zortrax M200 FDM machince. The prescribed limit for obscurance was set to $Ob_{lim} = 0.14$ which resulted in four offsets, i.e. $d_{off} = 4l_m$, and scanned by iOS and Android devices. All the seven colors were successfully read by both devices, except for yellow color which was not readable by Android devices. As yellow is a light color, we intentionally lowered the illuminance to 400 lx, but it was not readable by Android app. Therefore, the local offset operation was applied to the yellow color model with N_{offmax}=2 leading to a successful scanning.

7. Conclusion

We have introduced a new concept called "3D QR code" which can be printed directly on parts consisting of freeform surfaces represented in B-spline format. Its primary advantages are that it neither requires any secondary procedures in adding it during the printing process, nor multi-colored dark and light materials for printing, as the dark color is provided by ambient occlusion due to grooving. We have demonstrated our results on some interesting applications where such QR codes might prove to be useful.

Our method has certain limitations that require future research. First, 3D QR codes generated using our method do not work on materials that are too dark or too light in color. In such cases, a single material might not be sufficient. However, we note that QR codes by definition require a contrasting background to be picked up by existing scanners. Second, we observed that if the angle between the normal vector at the surface point and the viewing direction from the camera to the point is greater than 20°, the scanner usually fails to read the code. Third, we assume an ambient illumination model as in a typical office setup, and the scannability of 3D QR codes depends on ambient occlusion caused by grooving the surface. If the ambient light is too strong or weak, or if strong directional lights are present, then scanners may fail to read the code. Lastly, our work only considers embedding the QR code completely within a single B-spline surface. In practice, this is not a serious issue since the user can choose the region for placing the QR code. It is however good to be able to embed the QR to an arbitrary region even across multiple B-spline surfaces.

In future, we are interested in exploring a few extensions to our method. First, we would like to incorporate the presence of other lights in the scene or on the scanner itself (e.g., flashlight on a smartphone) directly in our algorithm to make 3D QR codes further robust to illumination changes. Second, we wish to experiment with materials that do not obey Lambertian reflectance. Third, we would like to identify more novel manufacturing applications where 3D QR codes can be applied. Lastly, we plan to consider implementing a custom QR code reader which can read 3D QR codes even with shallow grooving.

Acknowledgments

This research was partially supported by YNU Bilateral Collaborative Research Programs, and MOE Tier-2 Grant (2017-T2-1-076) of Singapore. The authors would like to thank Harutaka Imoto and Shoji Maruo for productive discussions, Taichi Kaneko and Taketoshi Suzuki for their assistance.

References

[1] DENSO WAVE, QR Code.com, `http://www.qrcode.com/en/` (2016).
[2] W. Gao, Y. Zhang, D. Ramanujan, K. Ramani, Y. Chen, C. B. Williams, C. C. Wang, Y. C. Shin, S. Zhang, P. D. Zavattieri, The status, challenges, and future of additive manufacturing in engineering, Computer-Aided Design 69 (2015) 65–89.
[3] X. Yan, P. Gu, A review of rapid prototyping technologies and systems, Computer-Aided Design 28 (4) (1996) 307–318.
[4] APRECIA PHARMACEUTICALS, FDA approves the first 3D printed drug product, `https://www.aprecia.com/pdf/ApreciaSPRITAMLaunchPressRelease__FINAL.PDF` (2015).
[5] À. Méndez-Feliu, M. Sbert, Comparing hemisphere sampling techniques for obscurance computation, in: Proceedings of the International Conference on Computer Graphics and Artificial Intelligence. Limoges, France, 2004.
[6] À. Méndez-Feliu, M. Sbert, From obscurances to ambient occlusion: A survey, The Visual Computer 25 (2) (2009) 181–196.
[7] B. Starly, A. Lau, W. Sun, W. Lau, T. Bradbury, Direct slicing of STEP based NURBS models for layered manufacturing, Computer-Aided Design 37 (4) (2005) 387–397.

Figure 12: Top-to-bottom: Flange, Perfume bottle, heart-shaped Chocolate models, and Pharmaceutical Tablet. Left-to-right: Rendering, 3D printed model, closeup view of 3D QR code of 3D printed model.

Table 1: Computational results of five models. ◯, △, × symbols denote "easy to read", "needs effort to read", "unable to read", respectively in the scannability of 3D QR codes.

Models	# of modules	Ob_{lim}	# of offsets	$N_{lim}(\%)$	Local offset		Time (sec)	Scannability		Illuminance (lx)
					N_{offmax}	$N_{lim}(\%)$		iOS	Android	
FLANGE	37×37	0.20	2	53.4	0	-	33.7	×	×	500 – 700
		0.20	2	53.4	2	64.7	64.8	◯	◯	500 – 700
		0.15	4	51.0	0	-	66.4	◯	◯	500 – 700
PERFUME	21×21	0.13	4	50.6	0	-	21.8	◯	◯	500 – 700
CHOCOLATE	25×25	0.13	4	50.1	0	-	31.3	◯	◯	500 – 700
TABLET	25×25	0.20	2	53.4	0	-	15.5	×	×	500 – 700
		0.20	2	53.4	2	71.8	30.0	◯	◯	500 – 700
		0.18	3	57.0	0	-	23.1	△	△	500 – 700
		0.15	4	54.0	0	-	30.4	◯	◯	500 – 700
WAVE-like (yellow)	33×33	0.14	4	51.9	0	-	53.8	△	×	400
		0.14	4	51.9	2	62.1	78.9	◯	◯	400 – 600

[8] Y. Sasaki, M. Takezawa, S. Kim, H. Kawaharada, T. Maekawa, Adaptive direct slicing of volumetric attribute data represented by trivariate B-spline functions, The International Journal of Advanced Manufacturing Technology 91 (5) (2017) 1791–1807.

[9] C. Cheah, C. Chua, C. Lee, C. Feng, K. Totong, Rapid prototyping and tooling techniques: a review of applications for rapid investment casting, The International Journal of Advanced Manufacturing Technology 25 (3) (2005) 308–320. doi:10.1007/s00170-003-1840-6.

[10] R. Katsuki, J. Ota, Y. Tamura, T. Mizuta, T. Kito, T. Arai, T. Ueyama, T. Nishiyama, Handling of objects with marks by a robot, in: Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453), Vol. 1, 2003, pp. 130–135 vol.1. doi:10.1109/IROS.2003.1250617.

[11] A.Singh, P.Singh, A review: QR codes and its image pre-processing method, International Journal of Science, Engineering and Technology Research 5 (6) (2016) 1955–1960.

[12] H.Chan, How To: Make Your QR Codes More Beautiful., http://mashable.com/2011/04/18/qr-code-design-tips/ (2011).

[13] R.Cox, QArt Codes., https://research.swtch.com/qart (2012).

[14] H.-K. Chu, C.-S. Chang, R.-R. Lee, N. J. Mitra, Halftone QR codes, ACM Transactions on Graphics 32 (6) (2013) 217.

[15] S. Wang, T. Yang, J. Li, B. Yao, Y. Zhang, Does a QR code must be black and white?, in: 2015 International Conference on Orange Technologies (ICOT), 2015, pp. 161–164. doi:10.1109/ICOT.2015.7498513.

[16] B. Liu, R. R. Martin, J.-W. Huang, S.-M. Hu, Structure Aware Visual Cryptography, Comput. Graph. Forum 33 (7) (2014) 141–150. doi:10.1111/cgf.12482.
URL http://dx.doi.org/10.1111/cgf.12482

[17] S. Qiao, X. Fang, B. Sheng, W. Wu, E. Wu, Structure-aware QR Code Abstraction, Vis. Comput. 31 (6-8) (2015) 1123–1133. doi:10.1007/s00371-015-1107-x.
URL http://dx.doi.org/10.1007/s00371-015-1107-x

[18] S. S. Lin, M. C. Hu, C. H. Lee, T. Y. Lee, Efficient QR Code Beautification With High Quality Visual Content, IEEE Transactions on Multimedia 17 (9) (2015) 1515–1524. doi:10.1109/TMM.2015.2437711.

[19] L. Li, J. Qiu, J. Lu, C.-C. Chang, An Aesthetic QR Code Solution Based on Error Correction Mechanism, J. Syst. Softw. 116 (C) (2016) 85–94. doi:10.1016/j.jss.2015.07.009.
URL https://doi.org/10.1016/j.jss.2015.07.009

[20] Z. Yang, Y. Bao, C. Luo, X. Zhao, S. Zhu, C. Peng, Y. Liu, X. Wang, ART-code: Preserve Art and Code in Any Image, in: Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp '16, ACM, New York, NY, USA, 2016, pp. 904–915. doi:10.1145/2971648.2971733.
URL http://doi.acm.org/10.1145/2971648.2971733

[21] J. Duda, P. Korus, N. J. Gadgil, K. Tahboub, E. J. Delp, Image-Like 2D Barcodes Using Generalizations of the Kuznetsov 2003: Tsybakov Problem, IEEE Transactions on Information Forensics and Security 11 (4) (2016) 691–703.

[22] W. Preston, S. Benford, E.-C. Thorn, B. Koleva, S. Rennick-Egglestone, R. Mortier, A. Quinn, J. Stell, M. Worboys, Enabling Hand-Crafted Visual Markers at Scale, in: Proceedings of the 2017 Conference on Designing Interactive Systems, DIS '17, ACM, New York, NY, USA, 2017, pp. 1227–1237. doi:10.1145/3064663.3064746.
URL http://doi.acm.org/10.1145/3064663.3064746

[23] D. Li, A. S. Nair, S. K. Nayar, C. Zheng, AirCode: Unobtrusive Physical Tags for Digital Fabrication, in: Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology, ACM, 2017, pp. 449–460.

[24] L. Piegl, W. Tiller, The NURBS book 2nd Ed., Springer-Verlag New York, Inc., 1997.

[25] N. M. Patrikalakis, T. Maekawa, Shape Interrogation for Computer Aided Design and Manufacturing, Springer-Verlag, Heidelberg, 2002.

[26] S. Zhukov, A. Iones, G. Kronin, An ambient light illumination model, in: G. Drettakis, N. Max (Eds.), Rendering Techniques '98, Springer Vienna, Vienna, 1998, pp. 45–55.

[27] J.H.Halton, Algorithm 247: Radical-inverse quasi-random point sequence, Communication of the Association for Computing Machinery 7 (1964) 701–702.

[28] J. Amanatides, A. Woo, et al., A fast voxel traversal algorithm for ray tracing, in: Eurographics, Vol. 87, 1987, pp. 3–10.