Computer-Aided Design 131 (2021) 102961

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

# Improved Embedding of QR Codes onto Surfaces to be 3D Printed ☆

György Papp [a,b,*], Miklós Hoffmann [a,c], Ildikó Papp [a]

[a] University of Debrecen, Faculty of Informatics, Kassai 26, 4028 Debrecen, Hungary
[b] University of Debrecen, Doctoral School of Informatics, Kassai 26, 4028 Debrecen, Hungary
[c] Eszterházy Károly University, Institute of Mathematics and Informatics, Leányka 4, 3300 Eger, Hungary

ARTICLE INFO

*Article history:*
Received 25 March 2020
Received in revised form 15 September 2020
Accepted 19 October 2020

*Keywords:*
QR code
3D printing
Projection
Embedding
Free-form surface
Ambient occlusion

ABSTRACT

Providing additional information for parts or items usually means to enclose it next to the object or affix it on the component when that is possible. However, another solution is available by gaining the benefits of an additive manufacturing technology, 3D printing. This technology makes it possible to embed the additional information onto the surface of the items, for example, in the forms of QR codes. In the work of Kikuchi et al. (2018), the QR code is embedded into CAD models that consist of B-spline surfaces by grooving its dark regions to shadow them. The method proposed by Peng et al. (2019) optimized the modules of the QR code and the depth to carve its dark modules into any general mesh. However, embedding the QR code with these methods, in some cases, especially in case of highly curved surfaces, the QR code is deformed during the process of projection onto the surface. This deformation can highly restrict the readability of the QR code. In this paper, we propose an improved method to embed QR codes onto free-form surfaces by using a low-end consumer-level 3D printer. Our aim is to provide a robust method to project the QR code onto surfaces even with high curvature. We discuss the problematic cases for the works mentioned above, and we present a process to find an optimal position and direction of projection for the QR code to avoid deformations on highly curved surfaces. To validate our method, we compare our results with the outcomes of Kikuchi et al. (2018) and Peng et al. (2019).



## 1. Introduction

The demand to include additional information next to produced components, parts, or items has long been present, and many different solutions have been introduced to suffice this need. For example, one could affix labels on the surfaces of items. However, precisely affixing labels are feasible only on areas of developable surfaces. Therefore another solution is required for curved free-form surfaces, such as merely attaching paper documentation next to the product or parts.

Since the production of personalized items and early prototypes became an easy task for everyone with the arrival of affordable 3D printing solutions and free, user-friendly 3D modeling software, the question of how to provide additional information to the printed shapes came up again. The benefits of additive manufacturing technologies make it possible to develop new solutions that can solve this problem. A recently emerged method [1,2] is to encode the information into a QR code and embed the resulting image into the surface of the object. Since any smartphone can read QR codes, the end-user of the product can easily decode the information. In the method of Kikuchi et al. [1] and Peng et al. [2], the QR code is projected onto the surface, and the black modules are carved into it to produce shadowed areas. The appearing contrast between the surface and the engraved regions makes it possible to read the QR code. However, these solutions cannot be applied to any surface without difficulties. There are cases when the smartphones cannot read the embedded QR code due to deformed regions caused by the projection. In this paper, we propose a robust solution to project and engrave a QR code onto any area of a free-form surface without significant deformation, thus avoiding negative affects in the decoding process of the QR code. In this method, we present a new way of finding the optimal position of the QR code and the direction of parallel projection to project it onto the surface. We also discuss central projection as an alternative option to project the code onto the surface and evaluate its readability by comparing the results of the two projections. Further on, we propose a modification of the bottom of the engraved black areas of the QR code to quickly and effortlessly increase the contrast inside the embedded regions of the QR code.

☆ This paper has been recommended for acceptance by Takashi Maekawa.

* Corresponding author at: University of Debrecen, Faculty of Informatics, Kassai 26, 4028 Debrecen, Hungary.
*E-mail addresses:* papp.gyorgy@inf.unideb.hu (G. Papp), hoffmann.miklos@uni-eszterhazy.hu (M. Hoffmann), papp.ildiko@inf.unideb.hu (I. Papp).

https://doi.org/10.1016/j.cad.2020.102961

The rest of this paper is structured as follows. In Section 2, we discuss the different existing techniques related to QR codes to embed information onto the surface. In Section 3, we present the problematic cases where previous works mentioned above do not provide satisfying results. Our method is presented in Sections 4 and 5, and its outcomes are compared with the previous methods in Section 6. Conclusion and future work close the paper.

The main contributions of our paper are as follows.

- We introduce an improved method to project and embed a QR code onto a curved area on a free-form surface while preserving the readability of the resulting code.
- We propose and evaluate the application of central projection for embedding the QR code onto the surfaces as an alternative method to the usual parallel projection. We compare the outcomes of the two projections.
- We use patterns at the bottom of the embedded regions of the QR-code to increase the contrast between black and white areas and improve the readability of the code.

## 2. Related works

There are numerous ways to encode information for products, items, or components. One of the widely used methods for this task is to generate a QR code. This way, the information is encoded into an image that later can be decoded by any smartphone.

In the last decade, several techniques has been proposed to improve the QR codes visually and make them more appealing to the users. There are different techniques, which vary from changing the color of the white and black regions to modifying the modules in the QR code to enhance them with different shapes or images [3–9].

Although these methods ensure the readability of the resulting QR code positioned on a flat patch of the surface, there are cases when it must be placed on a curved surface, which makes the readability and decoding of the QR code challenging. Recognizing and decoding a QR code from a cylindrical surface is a common problem [10–14]. In the work of Lay et al. [13], perspective projection is used to decode the deformed QR code. In our method, we are also using central projection to project the modules of the QR code onto the given surface.

One can apply markers on meshes to estimate their pose and track their movements. In [15] a high-speed, occlusion-robust tracking method is proposed with markers, that consists of dots and are applied on 3D printed meshes. The method automatically generates these markers for various 3D shape models to track their movements with a Dynamic Projection Mapping monochrome camera. In the work of Asayama et al. [16], a method is proposed to estimate the position of a mesh and correctly map the projected image on the surface of a model. Their method uses 3D printed meshes with embedded visual markers designed so that it is detectable by infrared cameras but diminished by projections. A stereo camera system reads the markers that are created by using generic algorithms to maximize the number of viewpoints from where the position of the mesh can be correctly estimated.

The QR code can also be used in different 3D techniques to encode information about the product. In [17–19], one can see examples to use the additive manufacturing technique to embedding tags, tracking code, or safety features into 3D models as a QR code by using tagging materials.

In the work of Kikuchi et al. [1], a technique has been proposed to embed the QR code into a B-spline surface by engraving the black regions of the QR code. The appearing shadow in the embedded areas was used to reach the required amount of contrast between the black and white parts of the QR code to make it readable.

In the work of Peng et al. [2], one can find a solution to optimize the embedding process of the QR code onto the surface of a triangulated mesh by suppressing the black modules continuity in the QR code and minimizing their carving depth. In their work, the application of central projection is only implicitly given, and there is no information about how the center of the projection is chosen. Also, there is no comparison between using the central and parallel projection to embed a QR code.

A novel QR code 3D fabrication framework is presented in [20] to embed QR codes unobtrusively with minimal shape modification. The method also carves the model's surface, like the previous ones, but the created QR code pattern is only visible for decoding when it is lit from a specific direction. The white modules of the pattern are lit, while the black ones are shaded by the directional light. The method simulates the light to compute the carvings of the black and white modules. The resulting average carving depth is approximately 50% less than in the work of [2]. The limitation of the proposed solution in terms of appearing self-occlusion and non-continuous projection phenomena is present when the QR code is embedded onto a highly curved shape.

Analogously to the methods described in [1] and [2], we restrict ourselves to shapes defined over a rectangular area of the $[x, y]$ plane, similar to a graph of a function $z = f(x, y)$ (in the Vase model half of the shape is considered this way). To provide larger flexibility in interactive modification, each surface is given in a parametric form as $S(u, v)$ free-form parametric surface. In our work, we used B-splines to create the surfaces for the QR code embedding. However, our proposed methods work with any kind of free-form parametric surface. Throughout the paper we suppose that the sides of the QR code to embed are parallel to the sides of the rectangular area.

## 3. Problematic cases

The ideal surface to place a QR code is flat because it ensures to avoid deformation during attaching or embedding a QR code. Small distortions in the shape of the QR code can be experienced in cases when the given area on the surface to place the QR code is, for example, slightly curved, twisted, or a bit of wavy. Most QR code reader applications can handle these small deformations and read the decoded information without difficulty. Unfortunately, in some cases, the given 3D model represented by a free-form surface does not have flat or just slightly curved regions, which would fit for smooth embedding of a QR code. In these cases, when we try to project the QR code onto the more curved surface, deformation issues can appear, highly restricting the readability of the code. One potential solution would be to decrease the size of the QR code in order to fit it to a smaller smooth part of the surface, but in case of 3D printing the printer might not have the required capabilities to print readable QR code small enough to fit on these limited flat regions.

Let $S(u, v), (u, v \in [0, 1])$ denote the surface given in parametric form, onto which the QR code should be embedded. The deformation problem can occur when the Gaussian curvature of the surface is large somewhere in the potential position of the QR code, that is, the angle of the unit normal vectors of two close points $S(u_i, v_i)$, and $S(u_j, v_j)$ is significantly large. In our practice $10 \times 10$ point sampling is used on the surface, and $5^\circ$ difference in neighboring unit normal directions can already restrict the readability of a simply attached QR code.

After defining the center and size of the future QR code by the user, our method finds and optimal positioning of the QR code, projects and engraves it in a readable form even if the selected part of the surface contains highly curved areas. In Fig. 1

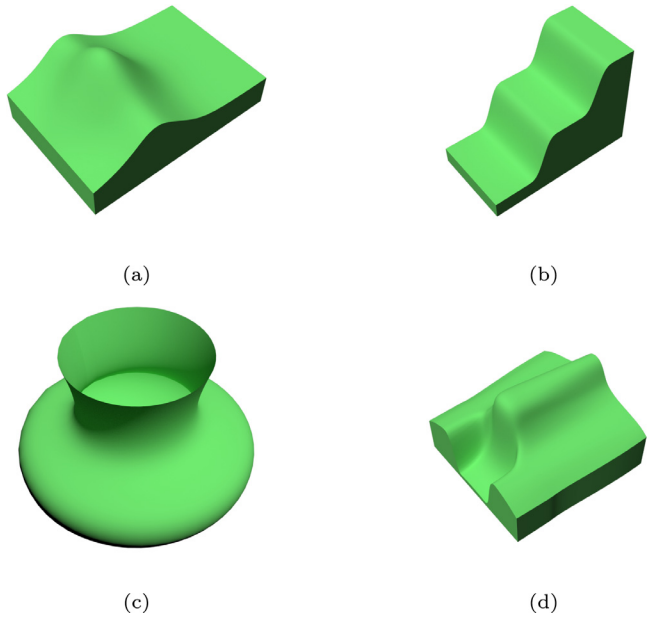



**Fig. 1.** Examples for surfaces with high curvature regions, where the QR code embedding method of Kikuchi et al. [1] and Peng et al. [2] can result in a less readable and deformed QR code. The surface (d) is created based on the car model available in [21].

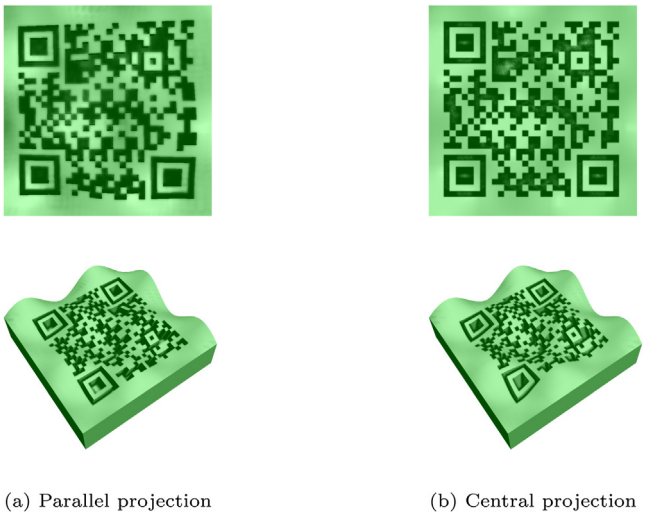



**Fig. 2.** The difference between parallel projection (left) and central projection (right) uses the same surface and position to engrave the QR code.

four examples can be seen to demonstrate the problem of highly curved surfaces where deformations can be expected to occur during embedding the QR code. Results can be seen in 3D printed form in Fig. 10.

It is important to note that we aim to create a method that works well with most low-end consumer-level 3D printers. These machines do not possess advanced capabilities, like the one used in [2]. To further improve the readability, we also provide a method to increase the shadow in case of using a lower-end consumer-level 3D printer.

## 4. Our method: improved embedding of the QR code

After positioning the center of the future code, the first question is the choice of projection method. Embedding the same QR code at the same position of the surface with central and parallel projection yields different results as can be seen in Fig. 2. In cases when the parallel projection is used to embed the QR code, the engraved modules show some deformation from the center of the QR code to its edges, making it hard or impossible to read. The level of this deformation evidently depends on the distance between the center of the QR code and the camera. The distortion is increasing when the camera is getting closer to the surface. The deformation can be reduced or eliminated by moving the camera far from the QR code. However, in this case, the QR code may become too small to read.

By engraving the QR code with central projection, a suitable position can be found for a QR code to read it easily without any deformation problem affected by the distance. In this paper, we discuss and present our solution by using central projection. However, our method is robust enough to be used with parallel projection as well, as one can see in the comparison part in Section 6. The algorithm is described in detail only for central projection because one can easily transform it into the parallel version.

The next step of our method is to define an initial plane of the QR code and project it onto the surface. Then we sample the surface normals in the points of the projected area. These surface points and normals are used to calculate a better position for the plane of the QR code and a center for the projection to embed the QR code onto the surface. It is an iterative method to gradually improve the position of the QR code's plane by sampling the normal vectors of the surface in each iteration. The normal vector of the resulting plane is used with the user-selected center of the QR code to calculate the center of the projection based on the central projection distance to embed the QR code. The iteration terminates when the angle between the normal vectors of two consecutive positions of the projected plane is less than our predefined error limit value (5°). However, in our practice, the first or second iteration has already provided an appropriate solution for position of the QR code plane. Further iterations yield higher computational costs with minor or no improvement in terms of readability.

Let us denote user-selected center position of the QR code on the surface by $O$. Also, the size of the code (the lengths of the square) is defined as $s$. In the beginning of the iteration, these given values are used to define an initial plane $h$, orthogonal to the surface normal in point $O$. The QR code is positioned onto this plane, with center $O$, and its vertices are projected onto the surface (see Fig. 3) to sample the surface. For the central projection, a point $C$ on the line defined by the surface unit normal vector $\vec{n}$ at point $O$, is defined as the projection center. A free parameter $g$ is used to control the distance of $C$ from $O$, and finally the position of point $C$ is defined as $C = O + g \cdot \vec{n}$. The projected square of the code forms a region on the surface, and the projected vertices yield a rectangular area $R$ in the surface parameter space (see Fig. 3(a)). In cases when the projected vertices are outside of the surface, we clamp their parameter values back to the boundaries of the surface parameter space.

Rectangle $R$ of the parameter space (see Fig. 3(a)) is then shifted to a new rectangle $R'$ in a way that its center $R_{center}$ moves to the point $O_{uv}$ assigned to center $O$ of the QR code in the parameter space (see Fig. 3(b)). In cases, when the shifted rectangle $R'$ is outside the surface parameter boundaries, $R'$ are clamped to the boundaries as shown in Fig. 3(b). The resulted area is used to sample normals of the surface $S(u, v)$ by using the Halton sequence [22]. Our algorithm works with $10 \times 10$ number of sample points.

For each sample point, we calculate a sample vector, which, in our case, is the unit normal vector of the surface in the sample

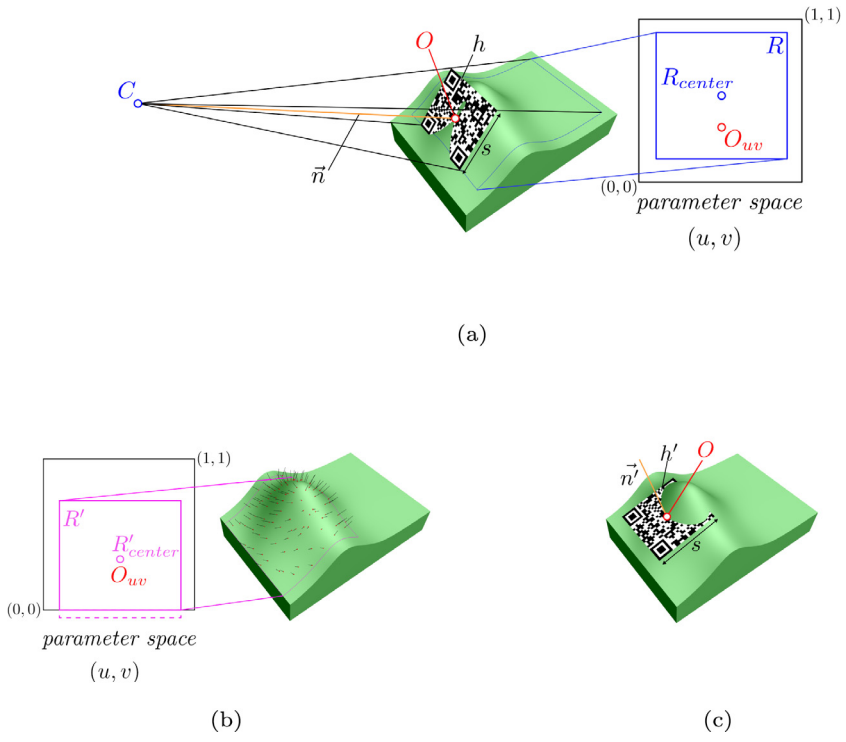



**Fig. 3.** The images show how our method finds the plane used for projecting the QR code points onto the surface. First, we define an initial plane $h$, which is centered at the given point $O$ with the normal vector $\vec{n}$ of the surface at this point (a). We use the projected area to sample the surface (b) and find the new position of the QR code and the direction to project it onto the surface (c). In (a) and (b), the parameter space of the surface is also visualized, with images $R$ and $R'$ of the code. The clamped area of $R'$ is shown with dashed line in (b).

point. These sample unit normal vectors $\vec{n}_i$ are summarized to define the unit normal vector

$$\vec{n'} = \frac{\sum_0^m \vec{n}_i}{\left\| \sum_0^m \vec{n}_i \right\|}$$

of the new plane $h'$ of the QR code, which is applied to project the modules of the QR code onto the surface (see Fig. 3(c)).

The algorithm calculates the points for embedding the QR code by projecting the modules of the QR code onto the surface from point $C'$, which is the new center of projection. Point $C'$ is defined analogously to point $C$: $C' = O + g \cdot \vec{n'}$ (see Fig. 4).

Each point of the QR code has its direction for the projection. Let the points of the QR code be $Q_{kl}$, where $1 \leq k \leq 10$, $1 \leq l \leq 10$. The direction of the projection for each point is the following $d(k, l) = Q_{kl} - C'$. Now we can define the projected points as $P_{kl} = C' + t \cdot d(k, l)$ where $C'$ is the new center of the projection, and $d(k, l)$ provides the directions to project each point. The usual Newton method is applied to calculate parameter $t$ to find the intersection point of this line and the surface, and to obtain the projected point of the QR code.

In the last step, vertices of each module of the QR code are projected to the surface, and those parts belong to a black region are grooved (Fig. 4). The final embedded QR code is created by triangulating the projected and engraved points of the QR code with the points of the surface. The carving depth for the black module is calculated using the proposed method of Kikuchi et al. [1]. However, in our application, we do not convert our embedded QR code into voxels to evaluate the amount of shadow in the grooved area. Instead, we use the triangulated embedded QR code and ray-tracing to sample the emitted rays for a given point in the engraved parts.

The engraving of the black modules is performed during the calculation of the carving depth. First, the QR code is triangulated with a minimal carving depth, then in each iteration, the amount of shadow is evaluated using the proposed function from [1]. Our stopping criteria are also the same, we increase the carving depth and modify the embedded QR code vertices according to it until more than half of the black module's calculated obscurance is under a given limit. We used the proposed obscurance limit of 0.2 from [1]. When the iteration stops, the black modules in our triangulated embedded QR code already have the required carving depth.

Images of an embedded QR code carved into the surface with our proposed method can be seen in Fig. 5. Models are printed out in different colors.

## 5. Increased shadow with patterns

In this section, we propose a method to further improve the readability of the code. The bottom of the embedded black regions of the QR code is modified in order to increase the amount of shadow and to increase the difference in intensity between the black and white modules. Our solution is to apply so-called patterns to the bottom of the carved regions of a QR code.

The idea behind using patterns is to obtain an engraved surface with diverse normal vectors at the bottom of the engraved area instead of a flat bottom surface with one normal vector. The obscurance value at point $P$ depends on the length of the emitted rays $r_i$ from it. The direction of the casted rays are calculated to be inside a hemisphere, as Fig. 6 shows. The hemisphere is defined by point $P$ and the surface unit normal at $P$. If the surrounding embedded QR code covers large parts of the hemisphere, then most of the emitted rays would hit them and have a smaller length, which helps us to reach a smaller obscurance value. Having a surface with diverse normal vectors results in hemispheres that have various positions during the obscurance calculation. Therefore, in order to have smaller obscurance values, we defined our proposed patterns' surfaces in a way that their normal vectors

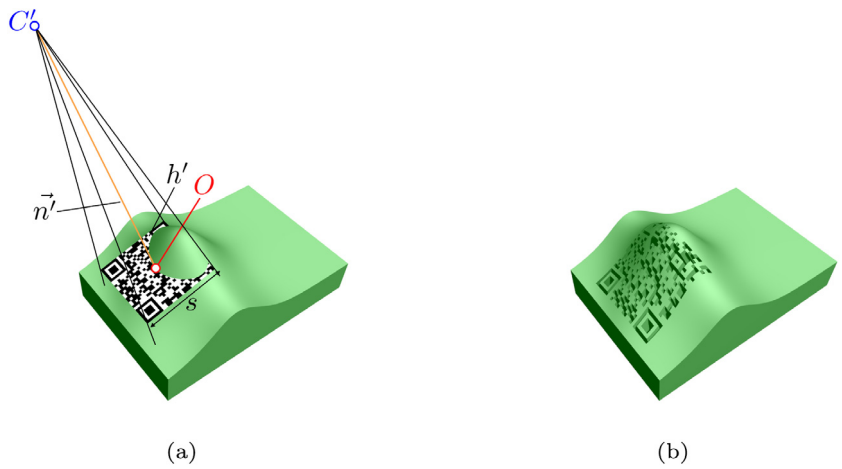



**Fig. 4.** Central projection of the planar QR code on the surface (left). The result of the projection (right). After calculating the projected points, black regions are carved into the surface.

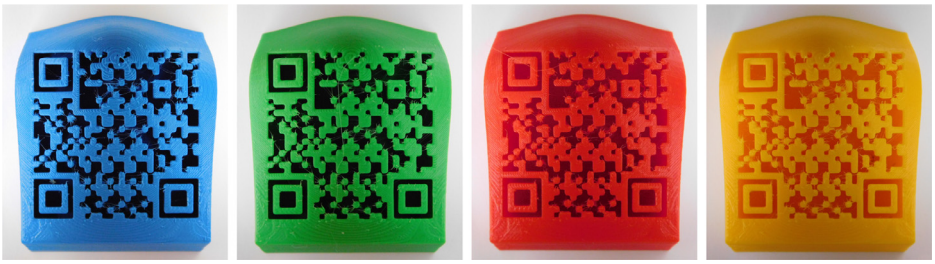


**Fig. 5.** A printed output of embedding QR codes with our methods in multiple colors.

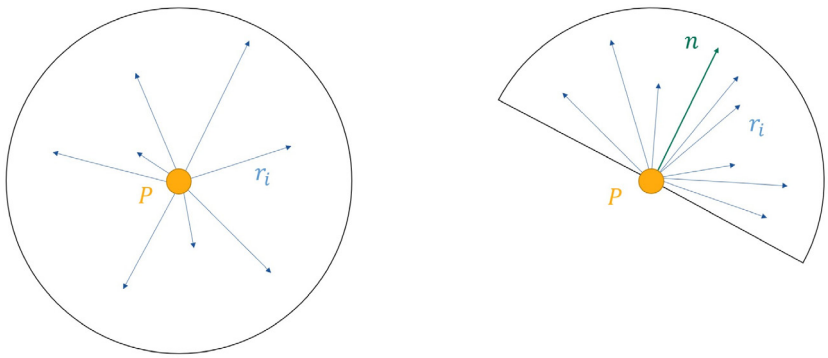



**Fig. 6.** The images illustrate how rays $r_i$ are emitted from point ($P$) inside the hemisphere to calculate the obscurance value. The left image shows the hemisphere from the top, while the right one shows it from the side.

would result in a hemisphere, which is mostly covered by the surrounded part of the embedded QR code.

We propose two different patterns, which are formed by modifying the engraved black modules of the QR code to provide a bumpy surface. These extruded shapes increase the shadow at the bottom of the QR codes. The two different patterns and their comparison are shown in Fig. 7.

The first pattern (see Figs. 7(a) and 8) is created to be the same for every engraved modules. We used eight additional vertices ($E, F, G, H, I, J, K, L$) in every engraved square module to create the pattern. The first four of these vertices are calculated by taking the mid-point of the module's sides. Point $I$ is the midpoint of the section connecting the center of the square and $H$. Points $J$, $K$, and $L$ can be calculated analogously. After we have all of the vertices at the bottom, we elevate orthogonally $E, F, G$, and $H$ up to the half of the current carving depth. The only remaining step is triangulation. The position of the new points and the triangulation of the vertices are shown in Fig. 8.

In the second pattern (see Figs. 7(b) and 9), we defined two similar surfaces for the bottom of the black modules. These are used alternately for the modules in each row to create our proposed pattern. Besides, our proposed surfaces are rotated 180 degrees for each module for every even row, as Fig. 7 shows. We insert two additional vertices $E$ and $F$ for this pattern (see Fig. 9). These are defined as the midpoints of segments $\overline{AD}$ and $\overline{BC}$, respectively. To form the first surface, the points $A$, $D$, and $F$ are elevated up to the half of the current carving depth. In the case of the other surface, the points $B$, $C$, and $E$ are moved in a similar way to create the shape. Then the points are triangulated, as shown in Fig. 9.

In Fig. 7, we show and compare how much these patterns can increase the amount of shadow and how they can help reduce the amount of incoming direct light from the sides to the inside

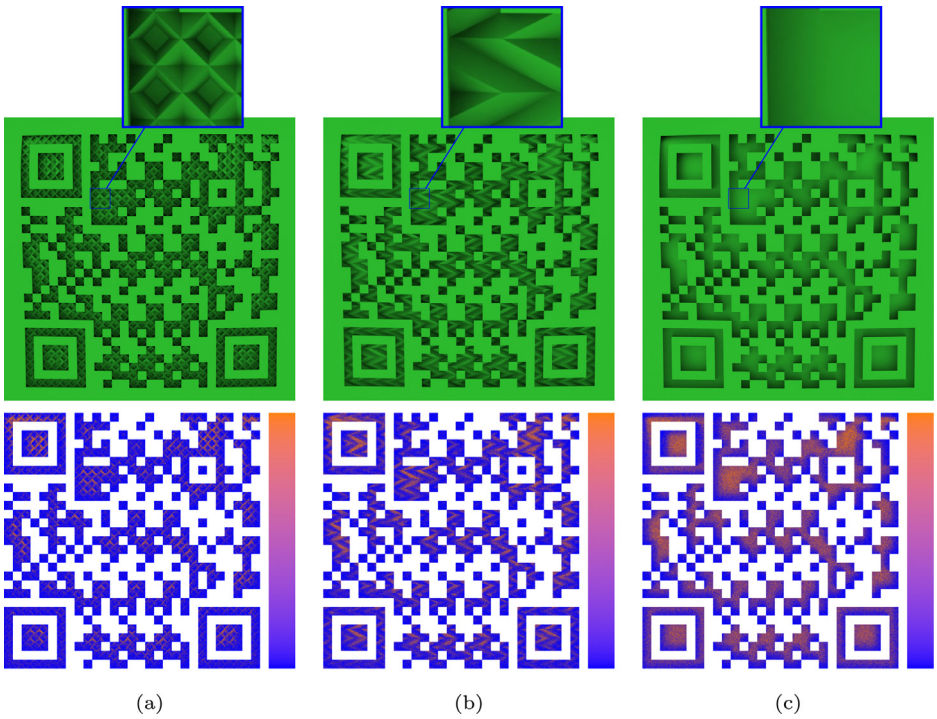



**Fig. 7.** Examples of the proposed pattern to increase the shadow intensity at the bottom of the engraved QR code. The first two images (a) and (b) show our proposed patterns, while the rightmost figure (c) with no pattern helps to compare the results. The amount of illumination in the embedded QR code for each pattern is presented in the bottom row, based on the calculations from [1] and [2]. Blue areas are not illuminated, while the orange ones are fully illuminated.

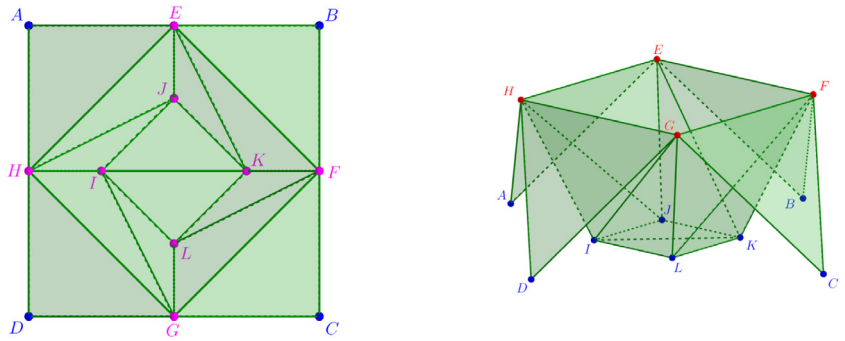



**Fig. 8.** The images show our first pattern to increase the shadow in the embedded modules. The left images show the surface of the pattern from the top with the additional vertices that are highlighted with pink and used to create the surface. The right image highlights the vertices that are moved to form our proposed pattern with red and shows the triangulated surface from the side.

the embedded QR code. The global illumination was computed using the method described in [1] to compare and find the carving depth of the QR codes.

## 6. Results

Our method is implemented in Dart programming language as a web application on a system with an Intel Core i7-8550U Processor (8M Cache, up to 4.00 GHz) with 16GB memory.

We tested our method on the example surfaces of the problematic cases and two other general surfaces to demonstrate its robustness. As Fig. 10 shows, our method works well with less curved surfaces or wavy surfaces, not only highly curved ones. In the case of surfaces with high Gaussian curvature, the center of the QR code was placed near to highly curved areas, and for the rest of the surface, it was placed at the center of the surface.

We used the Prusa i3 MK2.5 3D printer to test the results of our solution physically. The nozzle diameter of our printer is 0.4 mm, which defines the smallest size for the QR code that we are able to print. In the case of printing the embedded QR code with patterns, the size of the QR code's module should be chosen so that the details of the patterns remain visible after slicing and printing the surface.

Besides, during printing the surfaces, we noticed that it is worth rotating the models to a position where the embedded QR code walls are almost perpendicular to the table of the 3D printer. In this case, the printer can build the inner parts of the modules with higher precision, and the process does not require to use support material for the engraved QR code. Because of the rotation of the model, the printer might need to use support material for other parts, but it is probably easier to remove than the ones inside the embedded QR code. The problem of removing the support material from the engraved QR code can be solved using dissolvable support material, in which case there is no need for rotation of the model.

The readability of the embedded QR codes is compared by decoding them with different applications on Android and iOS smartphones. Our test showed that the iPhone built-in camera

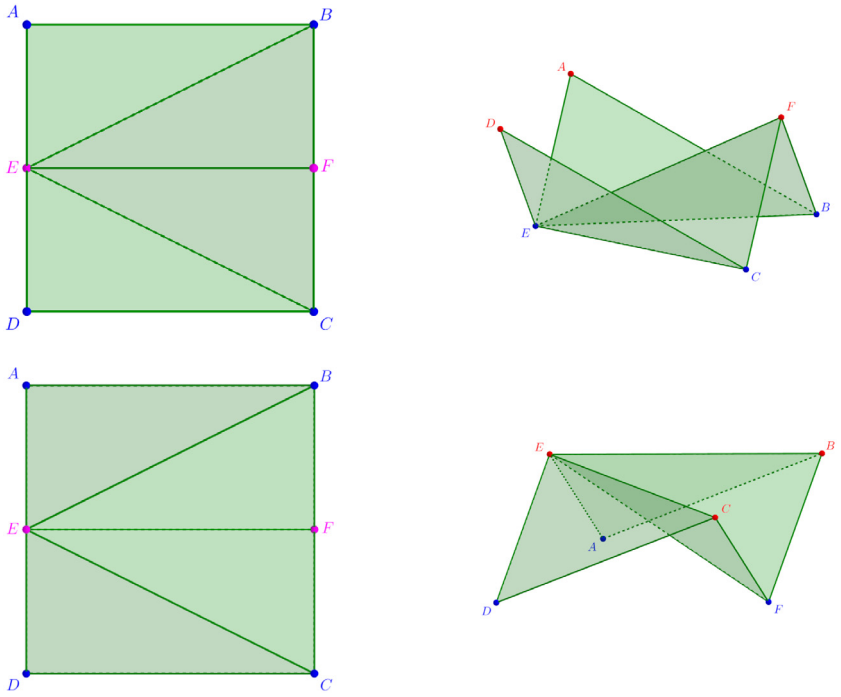



**Fig. 9.** The images show our second pattern for increasing the amount of shadow. This pattern has two main parts, which are shown on the left images. The additional vertices are again highlighted in pink. The right images show the triangulated surfaces from the side, and it highlights which vertices are moved to achieve our pattern with red.

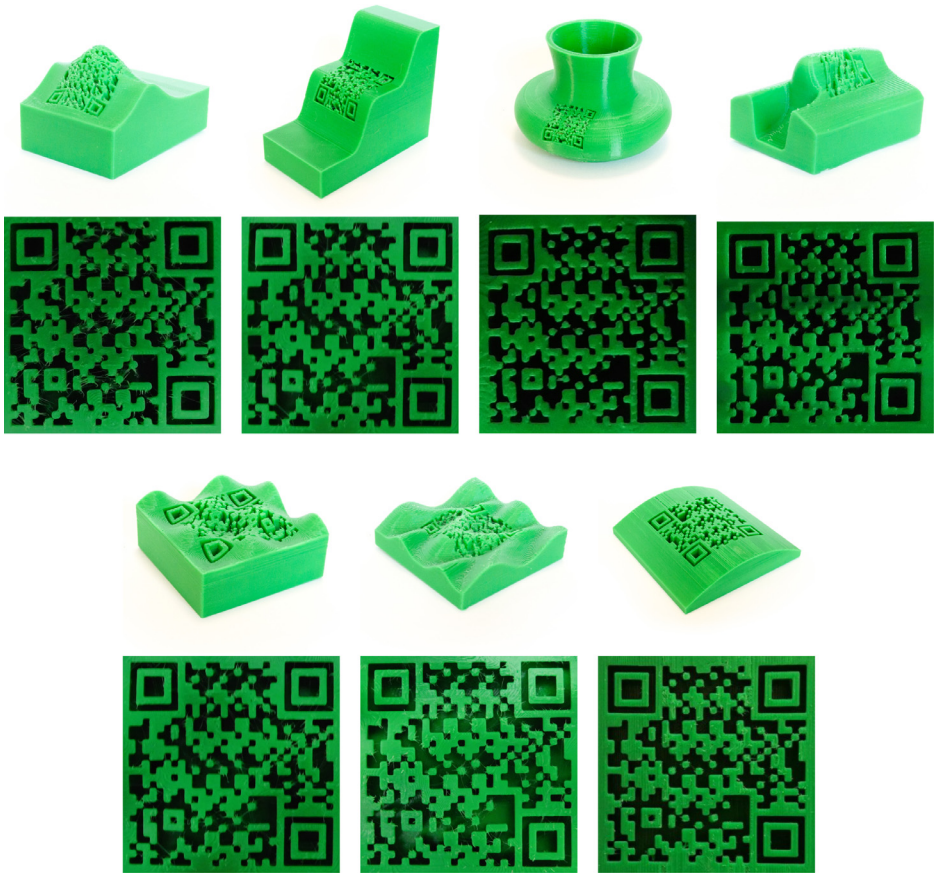


**Fig. 10.** The output of our method in 3D printed form. The upper figures show the surface that we used for embedding the QR codes. Below each surface, an image shows the embedded QR code from the direction of the projection. The first four pairs of images show that our method can embed QR codes onto highly curved surfaces without problems, while the remaining three pairs show that our method is also robust and can engrave the QR code onto wavy or less curved surfaces.

**Table 1**
This table presents the readability of the embedded QR codes created by Kikuchi's method, Peng's method, and our method. We used parallel projection for Kikuchi's method and central projection with a 10 cm center distance for creating Peng's and our models.

| Model | Phone OS | Method | | |
|---|---|---|---|---|
| | | Our | Kikuchi's | Peng's |
| Hill | iOS | Easy | Failed | Diff. |
| | Android | Easy | Failed | Diff. |
| Stair | iOS | Easy | Failed | Failed |
| | Android | Easy | Failed | Failed |
| Vase | iOS | Easy | Failed | Easy |
| | Android | Easy | Failed | Easy |
| Car | iOS | Easy | Failed | Easy |
| | Android | Easy | Failed | Easy |

**Table 2**
This table shows the readability measurements of our proposed method with different central projection distances.

| Model | Phone OS | Central projection | | | | | Orthogonal projection |
|---|---|---|---|---|---|---|---|
| | | Center distance | | | | | |
| | | 5 cm | 10 cm | 12 cm | 15 cm | 20 cm | |
| Hill | iOS | Diff. | Easy | Easy | Easy | Easy | Easy. |
| | Android | Failed | Easy | Easy | Easy | Easy | Easy. |
| Stairs | iOS | Diff. | Easy | Easy | Easy | Easy | Easy |
| | Android | Failed | Easy | Easy | Easy | Easy | Easy |
| Vase | iOS | Easy | Easy | Easy | Easy | Easy | Easy |
| | Android | Easy | Easy | Easy | Easy | Easy | Easy |
| Car | iOS | Diff. | Easy | Easy | Easy | Easy | Easy |
| | Android | Diff. | Easy | Easy | Easy | Easy | Easy |

app is better in reading the embedded QR codes than the Android applications we tested (QRbot, QR & Barcode Scanner). However, with both the iOS and the Android applications, we can read the embedded QR codes produced by our method in less than 5 s with one exception. In the cases when the center of the projection was placed close to the surface, only the iPhone was able to read the QR code, but the process required minutes of trying.

The result of the problematic surfaces (Hill, Stairs, Vase, and Car, see Fig. 1.) readability test can be seen in Table 1. and Table 2. First, we compared the readability of our proposed solution with the QR code engraving method of Kikuchi and Peng. After, in Table 2, we show the effect of the different central projection distances on the readability of the embedded QR codes. Three categories are used to measure the readability of the embedded QR code: abbreviation "Easy", "Diff.", and "Failed" are refer to "Easy to scan", "Difficult to scan" (means minutes of trying), and "Failed to scan". The 3D printed QR codes were tested both with iOS and Android-based smartphones.

To compare the parallel and central projection, we measured the range in which the embedded QR codes were steadily read without problems (see Table 3.). We used five different central projection distance in our test (5 cm, 10 cm, 12 cm, 15 cm, 20 cm). The center of the projection was calculated by multiplying the unit normal vector of the QR code with a distance and adding the resulting vector to the QR code center. As Table 3. shows, there are differences between the reading ranges of the four problematic surfaces because we used different QR code size for the embedding. We can also see that the reading ranges depend on how curved the selected surface was for the embedding.

Using the closest center of projection (5 cm distance), we could not efficiently read the QR code if its size was large (stairs and hill surface). However, with the remaining cases, we were able to quickly read the QR codes within a reassuringly broad range, as one can observe in Table 3.

**Table 3**
This table contains our measurements for the reading distance ranges between the code and the camera, where the embedded QR codes can be scanned easily in a stable manner. Measurements are given in cm. The larger QR code size with a small central projection distance results in a smaller reading range. However, the differences in the reading ranges between the QR codes that are embedded with different sizes becomes less significant if the central projection distance is increased.

| Projection | Center distance | Hill | | Stairs | | Vase | | Car | |
|---|---|---|---|---|---|---|---|---|---|
| | | Min | Max | Min | Max | Min | Max | Min | Max |
| Central | 5 cm | 7 | 7 | 5 | 5 | 10 | 82 | 10 | 14 |
| | 10 cm | 9 | 51 | 8 | 18 | 10 | 90 | 14 | 35 |
| | 12 cm | 12 | 60 | 9 | 24 | 11 | 100 | 17 | 45 |
| | 15 cm | 13 | 68 | 10 | 52 | 12 | 105 | 20 | 55 |
| | 20 cm | 14 | 80 | 11 | 75 | 12 | 110 | 22 | 65 |
| Parallel | – | 17 | 90 | 21 | 92 | 12 | 120 | 64 | 92 |

We also wanted to compare the reading ranges of our method to how close and far 2D printed QR codes can be read. The embedded QR code can be read from almost as close as the paper version if the central projection is used for engraving it. However, as we expected, the maximum distance for decoding is much smaller than in reading a 2D version. Therefore, the center of the projection or the projection method should be chosen based on the requirements of how close or far the resulting embedding QR code needed to be read stably.

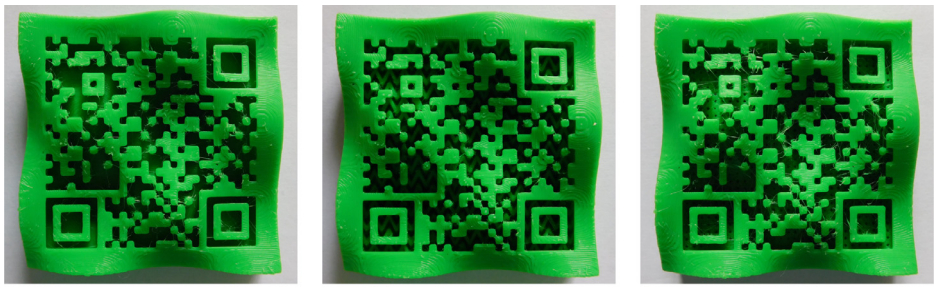
The reading ranges of the embedded QR code can be tweaked by changing the central projection distance. Therefore, our method can produce embedded QR codes that are readable only from a close distance and ones that can either be decoded from a close and a further distance. Also, similar reading ranges can be achieved using the central projection with a large central projection distance than using the parallel projection for embedding a QR code. Another advantage of providing similar reading ranges is that it eliminates or greatly decreases the effects of deformation when the QR code is read from the position of the center of the original central projection.

Our proposed patterns from Fig. 7 increases the amount of shadow at the bottom of the engraved modules of the QR code without the need for additional calculations (see Fig. 11). Also, they can be used with both the central and the parallel projection.

The wavy pattern (see Fig. 11(b)) remains visible enough after printing it with a larger nozzle diameter (0.4 mm) and considerable layer height (0.2 mm) to significantly increase the shadow at the bottom of the embedded QR code. The square pattern (see Fig. 11(c)) cannot produce as much shadow as the wavy one if a larger height and diameter nozzle is used. However, it can produce more shadow if we print a more detailed embedded QR code using a smaller layer height and nozzle diameter.

The printing time of embedding a QR code with a wave pattern is around 20 percent more than printing the same code without a pattern. Using the square pattern for the embedded QR code the required time of printing is further increased by around 8% comparing to the time required for the printing with the wave pattern. The printing time is further affected negatively when a smaller layer height is selected for the printing to achieve more shadow with the more detailed shapes of the pattern. The printing times calculations are based on our 3D printer, the Prusa i3 MK2.5. They can evidently alter if a different printer is used for creating the embedding QR code.

In Fig. 12 one can compare our results to the outputs of [1] (Kikuchi's method) and [2] (Peng's method) on the four surfaces mentioned in Section 3. In the case of Peng's method, their top view direction is implemented as a projection along the negative direction of the $z$ axis to project the QR code onto the surface.



(a) Without pattern (b) Wave pattern (c) Square pattern

**Fig. 11.** Using the pattern of Fig. 7 at the bottom of the carved modules increases the amount of shadow and the readability of the embedded QR code. However, as the square pattern shows, the QR code needs to be large enough for the pattern to be fully printable.

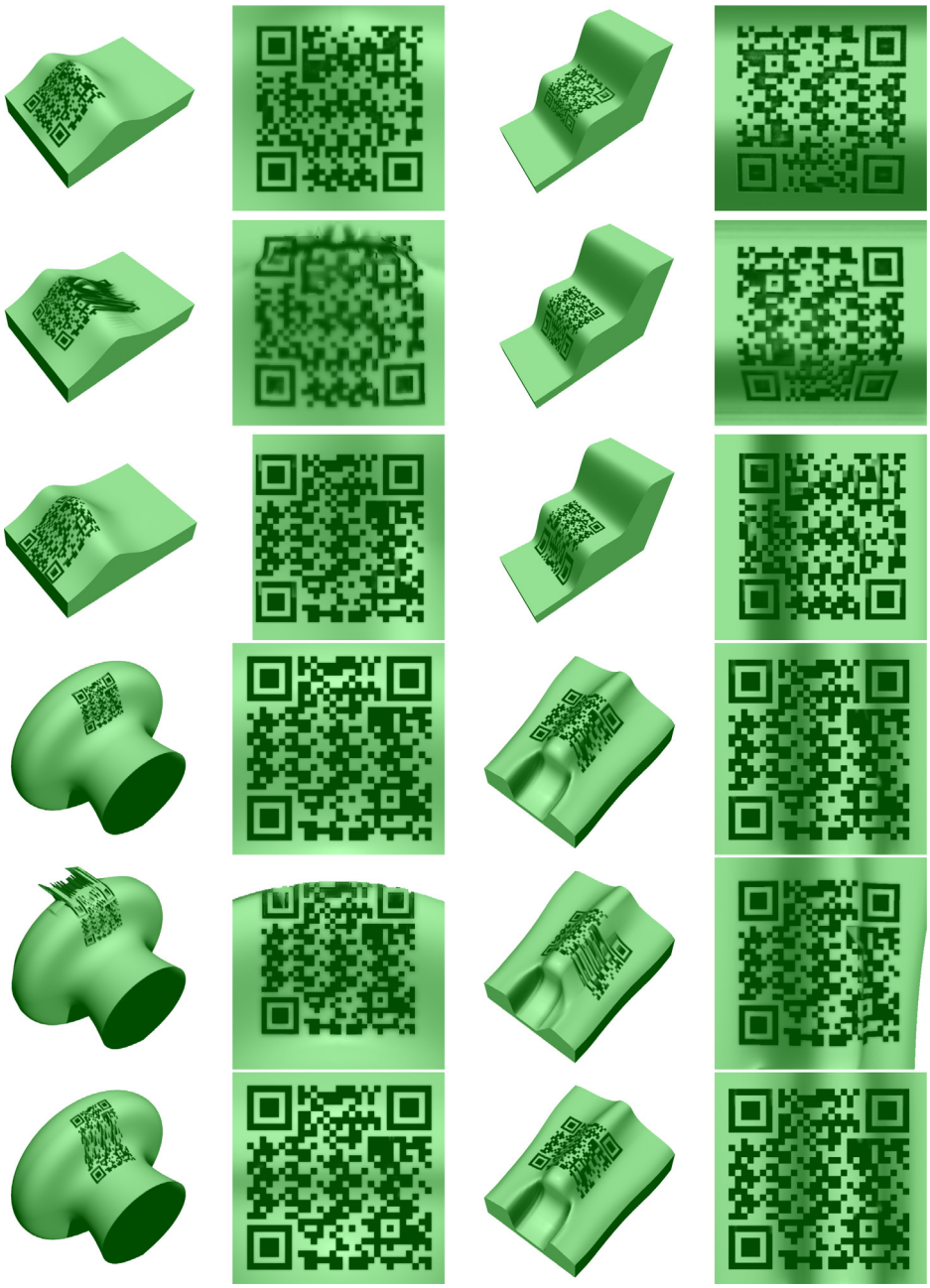


**Fig. 12.** Comparison of the outputs of our method (1st and 4th rows) to the ones given by [1] (Kikuchi's method, 2nd and 5th rows) and [2] (Peng's method, 3rd and 6th rows). The surfaces from Fig. 1 are used for generating the images.

## 7. Conclusion and future work

Free-form surfaces are widely used in the field of design and manufacturing for creating parts of machines or other items. Embedding QR codes on the surfaces of these items makes it easier to identify or use them. However, these parts or items' surfaces can contain highly curved areas (patches with large Gaussian curvature), which makes the use regular QR code labels impossible, and can also cause deformations and problems in readability in the recently developed embedded QR codes. We proposed an improved solution to engrave QR codes into highly curved areas of surfaces while preserving its readability in our work. Our method is robust and provides a readable embedded QR code for smoother of flat surfaces as well (see the last object in Fig. 10).

We used central projection throughout this paper. However, our method can compute and carve a QR code onto a surface with parallel projection as well. The advantage of using central projection is that one can read the QR code from a closer distance without significant deformation, and in a larger size. Besides, the range of reading the QR code can be broad enough based on the selected distance for the projection center from the surface patch.

The amount of shadow at the bottom of the embedded QR codes is vital for scanning the code successfully. Therefore, in our solution, we proposed to use patterns for generating more shadow in the engraved regions, increasing the readability of the generated QR code.

We compared our solution to the methods of [1] and [2]. The impact of using our pattern or leaving blank the bottom of the embedded QR code has also been compared and measured.

Our future plan is to improve the way of sampling the surface and determine the largest possible size of a QR code, which can be embedded around the user-specified point. Also, we plan to extend our method for general meshes.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This work was supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was supported by the European Union, co-financed by the European Social Fund.

## References

[1] Kikuchi R, Yoshikawa S, Jayaraman PK, Zheng J, Maekawa T. Embedding QR codes onto B-spline surfaces for 3D printing. Comput Aided Des 2018;102:215–23. http://dx.doi.org/10.1016/j.cad.2018.04.025, http://www.sciencedirect.com/science/article/pii/S0010448518302537. ID: 271502.

[2] Peng H, Lu L, Liu L, Sharf A, Chen B. Fabricating QR codes on 3D objects using self-shadows. Comput Aided Des 2019;114:91–100. http://dx.doi.org/10.1016/j.cad.2019.05.029, https://www.sciencedirect.com/science/article/pii/S001044851930212X.

[3] Chu H-K, Chang C-S, Lee R-R, Mitra NJ. Halftone QR codes. ACM Trans Graph 2013;32(6):217:1–8. http://dx.doi.org/10.1145/2508363.2508408, 09.

[4] Lin Y-S, Luo S-J, Chen B-Y. Artistic QR code embellishment. Comput Graph Forum 2013;32(7):137–46. http://dx.doi.org/10.1111/cgf.12221.

[5] Garateguy GJ, Arce GR, Lau DL, Villarreal OP. QR Images: Optimized image embedding in QR codes. IEEE Trans Image Process 2014;23(7):2842–53. http://dx.doi.org/10.1109/TIP.2014.2321501, https://ieeexplore.ieee.org/document/6810015.

[6] Gaikwad AM, Singh KR. Embedding QR code in color images using halftoning technique. In: 2015 international conference on innovations in information, embedded and communication systems. IEEE; 2015, p. 1–6. http://dx.doi.org/10.1109/ICIIECS.2015.7193016, https://ieeexplore.ieee.org/document/7193016.

[7] Wang S, Yang T, Li J, Yao B, Zhang Y. Does a QR code must be black and white? In: 2015 international conference on orange technologies. IEEE; 2015, p. 161–4. http://dx.doi.org/10.1109/ICOT.2015.7498513, https://ieeexplore.ieee.org/document/7498513.

[8] Lin L, Wu S, Liu S, Jiang B. Interactive QR code beautification with full background image embedding. In: Jiang X, Arai M, Chen G, editors. Second international workshop on pattern recognition. 10443, International Society for Optics and Photonics, SPIE; 2017, p. 211–5. http://dx.doi.org/10.1117/12.2280282.

[9] Xu M, Su H, Li Y, Li X, Liao J, Niu J, Lv P, Zhou B. Stylized aesthetic QR code. IEEE Trans Multimed 2019;21(8):1960–70. http://dx.doi.org/10.1109/TMM.2019.2891420, https://ieeexplore.ieee.org/document/8604076.

[10] Li X, Shi Z, Guo D, He S. Reconstruct argorithm of 2D barcode for reading the QR code on cylindrical surface. In: 2013 international conference on anti-counterfeiting, security and identification. IEEE; 2013, p. 1–5. http://dx.doi.org/10.1109/ICASID.2013.6825309, https://ieeexplore.ieee.org/document/6825309.

[11] Lay K-T, Wang L-J, Han P-L, Lin Y-S. Rectification of images of QR codes posted on cylinders by conic segmentation. In: 2015 IEEE international conference on signal and image processing applications. IEEE; 2015, p. 389–93. http://dx.doi.org/10.1109/ICSIPA.2015.7412222, https://ieeexplore.ieee.org/document/7412222.

[12] Lay K-T, Wang L-J, Han P-L. Decoding of QR codes printed on spheres. In: 2016 5th international symposium on next-generation electronics. IEEE; 2016, p. 1–3. http://dx.doi.org/10.1109/ISNE.2016.7543336, https://ieeexplore.ieee.org/document/7543336.

[13] Lay K-T, Zhou M-H. Perspective projection for decoding of QR codes posted on cylinders. In: 2017 IEEE international conference on signal and image processing applications. IEEE; 2017, p. 39–42. http://dx.doi.org/10.1109/ICSIPA.2017.8120576, https://ieeexplore.ieee.org/document/8120576.

[14] Li K, Meng F, Huang Z, Wang Q. A correction algorithm of QR code on cylindrical surface. J Phys Conf Ser 2019;1237:022006. http://dx.doi.org/10.1088/1742-6596/1237/2/022006.

[15] Watanabe Y, Kato T, Ishikawa M. Extended dot cluster marker for high-speed 3D tracking in dynamic projection mapping. In: 2017 IEEE international symposium on mixed and augmented reality; 2017. p. 52–61. http://dx.doi.org/10.1109/ISMAR.2017.22. https://ieeexplore.ieee.org/document/8115404.

[16] Asayama H, Iwai D, Sato K. Fabricating diminishable visual markers for geometric registration in projection mapping. IEEE Trans Vis Comput Graphics 2018;24(2):1091–102. http://dx.doi.org/10.1109/TVCG.2017.2657634, https://ieeexplore.ieee.org/document/7831400.

[17] Li D, Nair AS, Nayar SK, Zheng C. AirCode: Unobtrusive physical tags for digital fabrication. In: Proceedings of the 30th annual ACM symposium on user interface software and technology. New York, NY, USA: ACM; 2017, p. 449–60. http://dx.doi.org/10.1145/3126594.3126635.

[18] Wei C, Sun Z, Huang Y, Li L. Embedding anti-counterfeiting features in metallic components via multiple material additive manufacturing. Addit Manuf 2018;24:1–12. http://dx.doi.org/10.1016/j.addma.2018.09.003, http://www.sciencedirect.com/science/article/pii/S2214860418305189. ID: 306190.

[19] Chen F, Luo Y, Tsoutsos NG, Maniatakos M, Shahin K, Gupta N. Embedding tracking codes in additive manufactured parts for product authentication. Adv Energy Mater 2019;21(4):1800495. http://dx.doi.org/10.1002/adem.201800495.

[20] Peng H, Liu P, Lu L, Sharf A, Liu L, Lischinski D, Chen B. Fabricable unobtrusive 3D-QR-codes with directional light. Comput Graph Forum 2020;39(5):15–27. http://dx.doi.org/10.1111/cgf.14065, https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14065.

[21] Palmiga. Turbosquid - free car 3d model. https://www.turbosquid.com/3d-models/free-car-3d-model/166348. [visited on 25/06/2020].

[22] Kocis L, Whiten WJ. Computational investigations of low-discrepancy sequences. ACM Trans Math Software 1997;23(2):266–94. http://dx.doi.org/10.1145/264029.264064.