DOI: 10.1111/cgf.14394

COMPUTER GRAPHICS *forum*
Volume 00 (2021), number 0 pp. 1–17

Embedding QR Code onto Triangulated Meshes using Horizon Based Ambient Occlusion

György Papp,[1,2] Miklós Hoffmann[2,3] and Ildikó Papp[2]

[1]Doctoral School of Informatics, University of Debrecen, Debrecen, Hungary
papp.gyorgy@inf.unideb.hu
[2]Faculty of Informatics, University of Debrecen, Debrecen, Hungary
papp.ildiko@inf.unideb.hu
[3]Institute of Mathematics and Informatics, Eszterházy Károly University, Eger, Hungary

Abstract

QR code is a widely used format to encode information through images that can be easily decoded using a smartphone. These devices play a significant role in most people's everyday lives, making the encoded information widely accessible. However, decoding the QR code becomes challenging when significant deformations occur in the label. An easy and quick solution to keep the deformation on a minimum level is to affix the label that contains the QR code onto a developable surface patch of a 3D model. The perspective distortion that can appear is efficiently dealt with during the decoding process. In recent years an alternative method has emerged. In the work of Kikuchi et al., the QR code is embedded onto B-spline surfaces of CAD models to give more freedom in providing additional information. This method was further improved and extended by Peng et al. embed QR codes onto the surface of general meshes. This paper introduces a solution to embed QR codes onto the surface of general meshes without densely triangulating the selected area of the QR code. It uses the deferred shading technique to extract the surface normals and the depth values around the QR code's user-given centre. We propose two methods for automatically finding the projection direction even when highly curved areas are selected based on the retrieved information while rendering the model. Besides, we introduce two methods needing a projection direction and a QR code centre to determine a size for automatically embedding the QR code. We propose patterns for decreasing the carving depth of the embedded QR codes, and we use the Horizon-Based Ambient Occlusion to speed up the engraving process. We validate our method by comparing our results to the outcomes of Peng et al.

Keywords: rendering, CAD, modeling

ACM CCS: • Computing methodologies → Mesh models; rasterization;

1. Introduction

Providing information along with an item is essential for the user in terms of describing the proper usage of the product or how to assemble its parts. In most cases, this supplementary information is enclosed as a printed document along with the product. However, recently it often contains references to the manufacturer's website in the form of a QR code to register the product or get more information. Moreover, the QR code can even be printed on a label affixed to the item's surface.

Due to fundamental theoretical restrictions of differential geometry, smooth (i.e. isometric) label placement is limited only for developable surface patches of the product to avoid significant deformation of the QR code, which could negatively affect its readability. However, when an item does not have this kind of surface patch, or it has, but it is not suitable (e.g. too small) to place a large enough and readable QR code on it, the additive manufacturing techniques can help. The essence of this technique is that instead of applying labels of the generated QR code onto the surface, the QR code modules are projected onto the surface, and the dark ones are engraved into the surface to generate shadow and create the required contrast between the QR code's white and black parts for the decoding.

In recent years, this groundbreaking new method, the engraving of QR codes into the surface, has been introduced and improved by several authors. Kikuchi et al. [KYJ*18] embedded QR codes onto CAD models constructed by B-spline surfaces. Additionally, Peng

et al. [PLL*19] engraved QR code onto general meshes by optimizing its modules and carving depth. An embedding solution of the QR code that uses directed light to make the engraved QR code readable is presented by Peng et al. [PLL*20]. However, these solutions have some limitations and require preprocessing steps, typically modification and recomputation of the surface before the QR code can be embedded. In our previous work [PHP21], we presented an improved method to embed QR codes onto even problematic, highly curved free-form surfaces, but it works specifically for free-form surfaces.

In this paper, we propose a method that does not require any preprocessing step before the QR code is engraved onto a triangulated mesh. The previous works [PLL*19, PLL*20] that embed the QR code onto the surface meshes do not provide exact algorithms on how the surface normals are used to find a projection direction for engraving a QR code. Therefore, we present two methods in detail for finding the direction that can be used to project the QR code onto the surface. These methods are also based on using the surface normals to get the projection direction like in our previous work to [PHP21]. However, our second method additionally uses depth information along with the surface normals. Besides, our engraving solution can automatically determine the size of the QR code for a user-given centre and projection direction. The resulting size further depends on the user's decision about whether the QR code plane created with the current size of the search should be freely rotated inside the mesh boundaries defined by the used projection or not.

An alternative scenario is when the user has no specific preference on the position of the QR code, but a suitable (practically a minimum) size of the code is given. The task is to find the area(s) on the surface which the user can choose as a potential centre for that size of a QR code. This case will also be discussed.

The surfaces patches where a QR code can be embedded are limited by the carving depth required for successfully decoding the resulting engraved QR code. In order to embed QR codes onto even thinner parts of a mesh, we propose to apply patterns on the dark modules of the QR code. Our patterns change the module's carving shape and decrease the carving depth of the QR code.

A model's lighting can be calculated using one of the many ray-tracing techniques, and even real-time results can be achieved with the hardware-accelerated ray-tracing capabilities of the latest GPUs. However, our aim was to show that the SSAO techniques, which produce less precise lighting results, are good enough for measuring the amount of shadow in the engraved QR code areas. Therefore, in this work, we propose to use the Horizon Based Ambient Occlusion technique to speed up finding the carving depth of the QR code with the GPU's help.

The remaining of our paper is structured in the following way. Section 2 discusses the related work, while in Section 3, our projection direction search method is introduced in detail. Section 4 proposes a method to find the QR code's size or potential positions automatically, and the next section introduces how to embed the QR code. In Section 6, we present our patterns for reducing the carving depth of the QR code with a method to speed up finding this depth discussed in Section 7. Section 8 shows the measurements of our methods and comparisons to previous solutions. Conclusions close the paper.

2. Previous Work

The topic of how the QR code can be visually more appealing to the user is an actively researched area. The proposed works varies from changing the colour of the QR modules to modifying their position and shapes for enhancing the QR code with images and figures [CCLM13, LLC13, GALV14, GS15, WYL*15, LWLJ17, XSL*19].

However, having an attractive and informative QR code itself is not enough if the QR code is affixed on curved surfaces, and it suffers from deformation that makes it harder or impossible to decode. QR code readers are very sensitive in this manner. Reading a QR code from a cylindrical surface is a common problem [LSGH13, LWHL15, LWH16, LZ17, LMHW19]. A solution to tackle down the deformation in the QR code and successfully decode it by using perspective projection is proposed in the work of Lay et al. [LZ17]. As in other QR code engraving solutions [PLL*19, PLL*20], we also use the perspective projection during our QR code embedding process.

Besides affixing a QR code as a label on a surface, one can also embed it into the object. In the work of Wei et al. [WSHL18], the QR code is a safety feature, and it is embedded into metallic components to fight against illegal counterfeits. The selective laser melting (SLM) technology is used to embed the QR code, and the result is decoded with the help of X-ray imaging. Another reason for embedding a QR code inside the components can be to track them through the additive manufacturing chain. Chen et al. [CLT*19] proposed a solution to prevent 3D printing attacks by obfuscating the QR code before embedding it.

A 3D printed QR code can also be beautified based on the work of Yang et al. [YPLL19]. They propose a method that automatically generates perforated artistic QR codes and embeds them onto surfaces using a 3D printer. The generated perforated QR code's white components are connected and optimized to ensure the embedded QR code's readability.

Besides, objects can also be tagged with a QR code to describe and provide useful information. For example, AirCode method [LNNZ17] embeds the user-defined information by placing structured air pockets under the surface of the item. The resulting code of the method is only readable by computers, and the process takes minutes to provide results.

QR codes can also be embedded onto CAD models, as Kikuchi et al. [KYJ*18] proposed in their work. The models they worked with are constructed from B-spline surfaces, and the QR code is created by refining the surface with knot insertion. Then, the black modules of the QR code are grooved to reach the required contrast in the QR code for readability. Finally, the control points corresponding to the black modules are adjusted in an iterative process that automatically grooves the QR code.

The embedded QR code carving depth has been further optimized in the work of Peng et al. [PLL*19]. Before embedding the QR code, the number of connected black modules should be decreased by modifying the QR code's modules to reach higher contrast in the engraved QR code. Although the method of projecting the QR code onto the surface is only briefly sketched in their paper, they presented an iterative method to find a minimal non-uniform carving

depth for the embedding. The method determines the carving depths for each sample vertex point of the engraved modules separately. A method similar to our technique is used in [PLL*19] for embedding the QR code. However, the paper does not detail the embedding process; only the shared source code showed the implemented QR code embedding. The mentioned preprocessing of remeshing the selected QR code area to achieve a densely triangulated area is done by triangulating the 2D QR code image pixels forming a densely triangulated 2D QR code mesh, which vertices are projected onto the mesh. A densely triangulated zone is formed around the QR code after the projection. The selected region for the QR code embedding is removed, and the resulting densely triangulated surface is merged with the boundaries of the mesh's remaining part using a Delaunay triangulation.

A novel QR code 3D fabrication framework has been presented in [PLL*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 illuminated, while the directional light shades the black ones. The average of the surface normal vectors is used to determine the projection direction. However, the surface normal vectors' sampling process is insufficiently detailed in the paper. The method simulates the light to compute the black and white modules' carvings. The resulting average carving depth is approximately 50% less than in the work of [PLL*19]. The proposed solution has evident limitations in terms of embedding a QR code onto highly curved shapes because of the appearing self-occlusion and non-continuous projection phenomena.

The Screen Space Ambient Occlusion (SSAO) was first presented in the work of Mittring [Mit07] for CryEngine2. During the years, other techniques were created to generate more realistic lighting faster. Few of the many techniques are [BSD08, Tim13, NAM*17, ZXL*20]. Nowadays, most computer games use one of the many available SSAO techniques to quickly calculate lighting and give a natural feeling to the graphics. Moreover, the SSAO techniques can quickly produce a visually pleasant and convincing approximation of the shadow for a given camera view. One of the many available techniques is the Horizon-Based Ambient Occlusion [BSD08] (HBAO), which we have applied in our carving depth search method.

In our previous work [PHP21] we introduced a method for embedding QR codes, but it works only for free-form surfaces defined over a rectangular area of the $[x, y]$ plane in a $z = f(x, y)$ form. This previous method uses the surface normal to find a projection direction that eliminates deformations in the engraved QR code. Based on [PHP21], here we overview our previous method to help the comparison between our new and former methods. First, the user has to provide inputs for embedding the QR code: the QR code centre on the surface (point O) and a QR code size (s). Then, our iterative method creates an initial h plane with QR code size s centred at point O and orthogonal to surface normal at O. Next, plane h is projected onto the surface to form a rectangular area R in the surface parameter space. After, rectangle R is shifted until its R_{centre} is at point O_{uv}, which is the QR code's centre in the parameter space. We denote the shifted rectangle with R'. The resulting rectangles can be partially outside of the surface parameter boundaries. In order to avoid problems during the sampling, we need to modify the rectangles by clamping them to the surface parameter boundaries. The sampling of the surface normals is performed using the Halton sequence in the surface patch selected by rectangle R'.

The unit normal vector $\vec{n'}$ of the new plane h' is simply calculated by summarizing the sampled surface unit normals $\vec{n_i}$,

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

The resulting projection direction of the first or second iteration is used in the later steps, because in our practice it resulted an appropriate embedded QR code. However, it is not guaranteed that it will always provide deformation-free embedding.

The projected points of plane h' are searched in a similar way to the [PLL*19, KYJ*18] methods in a $P_{kl} = C' + t \cdot d(k, l)$ form, where C' is the new centre of the projection, and $d(k, l)$ gives the directions to project each point of the QR code. Then, we used the Newton method to calculate parameter t and get the projected point of the QR code plane.

In our previous method, we proposed patterns that defined surfaces with different normals vectors to increase the amount of shadow in the engraved areas. The presented surfaces were used at the bottom of the engraved black modules. Our first pattern consists of one surface applied uniformly for each black module, while our second pattern was created from two surfaces that are used alternately for the black modules to create a wavy pattern. The comparison of our introduced patterns showed that applying our patterns to the QR code increases the shadow in the engraved modules without carving the QR code further into the surface. However, the introduced patterns work only with sufficiently large QR code module size or high 3D printing quality for ensuring that the applied surfaces at the bottom of the engraved black modules remain visible after the 3D printing.

3. Finding the Direction of Projection of the QR Code

In the works of Peng et al. [PLL*19, PLL*20], the projection direction calculation is based on the assumption that QR codes are usually decoded perpendicularly from a top view. They use the normal direction to the surface in [PLL*19] and the surface normals' average in [PLL*20] for projecting a QR code. However, they do not define how to find the normal direction to the surface or sample the surface normals to average them. In the work of Kikuchi et al., the direction to project the QR code was found by using the Principal Component Analysis (PCA) with B-spline surfaces.

We propose two methods to sample the surface normals and depth values for determining the projection direction. These methods automatically find a normal vector to form a plane, which can be used to project the QR code's modules on the triangulated mesh's surface. Our methods' only required input parameter is the QR code centre, which the user needs to provide as a point on the mesh. The QR code size is an optional parameter of our method because we can find a size to embed the QR code around the previously given centre using the result of our projection direction search method (see Section 4).

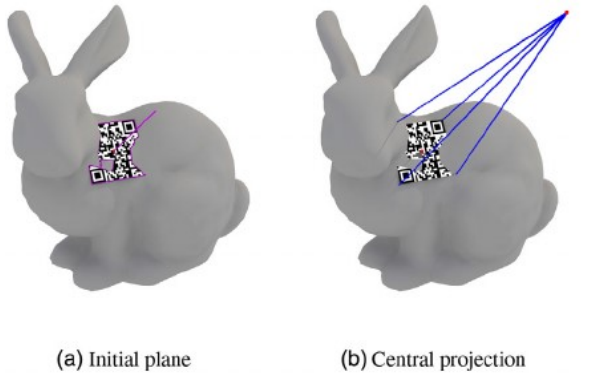

Figure 1: *The left image shows the initial plane, and the right one shows the central projection defined by the plane. Our method's central projection distance parameter was chosen to be 25 cm for a 0.12 mm QR code module size.*

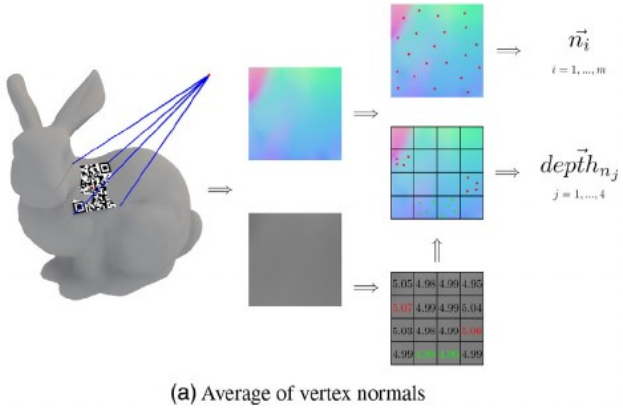

Figure 2: *The image shows how the surface is sampled in the second method using the normal and depth textures.*

The crucial difference between our new projection direction search method and the one proposed in [PHP21] is the way how we select the area on the mesh's surface and how we sample the surface normals in the selected area. Our previous method selected the sampling area in the surface parameter space and used the surface's equation to find its normal vectors at a given point. However, we always use the area selected by the current projection direction for sampling in our new methods. Also, we changed how our new method handles those cases when the selected sampling area is outside the mesh boundaries viewed from the projection direction.

The surface of the mesh is not directly sampled since we render the sampling area to create a map that contains surface normals for each pixel of the rendered area. Then, the mesh normals are interpolated using the smooth interpolation qualifier to sample the surface normals over the selected area. Besides, our second method's sampling process uses an additional sampling step, where the previously created map is resampled based on the distance between the mesh and the QR code plane. Our first method determines the new normal vector of the QR code plane similarly to our previous solution.

Both of our methods project an initial plane to the triangulated mesh to define the area where the mesh normals are sampled (see Figure 1). We define this initial plane through the selected centre, orthogonally to the surface normal at the user-selected point.

The QR code plane may be partially outside of the mesh silhouette viewed from the projection direction. As a result, there are areas in the texture containing the rendered normal vectors that do not contain any information about the mesh. Continuing the QR code embedding using this texture might result in a deformed embedded QR code. Therefore, we replace all of those normal vectors in the texture based on the given projection direction, which point is outside the mesh silhouette. For example, let B_{ij} be the points that are outside of the mesh silhouette. Its value is replaced with the $Bn_{ij} = B_{ij} - C - \vec{n}$ vector in the texture, where C is the selected centre of the QR code and $\vec{n}$ is the normal vector of the QR code plane. The resulting Bn_{ij} vector is used in our method for sampling the normal vectors. The replaced normal vectors can help modify the QR plane position to be inside the mesh's silhouette again. However, it is not guaranteed that the resulting plane of our iterative methods remains inside of the mesh's silhouette in every case.

Our first method samples the area defined by projecting the initial plane onto the mesh's surface and calculates the sampled normal vectors' average like in our previous work [PHP21]. However, instead of directly sampling the surface, our method renders the selected area's normals into a texture using the central projection defined by the initial plane and our central projection distance parameter, as seen in Figure 1. As the next step, our method takes m samples from the texture using the Halton sequence similarly to [PHP21] by substituting the surface parameters with the texture coordinates. The initial plane's normal vector is replaced with the average vector calculated from the samples. The updated plane is used later for embedding the QR codes.

The second method uses an iterative method to find a projection direction that minimizes the variance of the distance between the sampled area and the projection centre. The method starts with the same initial plane and renders the same area as the first method. However, the depth values are also stored in a texture besides the surface normals during the rendering process. An $n \times n$ grid is fitted on the resulting textures, and the average depth value is calculated for each grid. The new normal vector for the plane is calculated in two steps, where the first one calculates the normal textures' sampled average like in the first method. In the second step, only those grids of the normal texture are used in the sampling and average calculation ($\vec{depth}_{n_j}$) where the average depth value of the depth texture's grids was the two largest and lowest (see Figure 2).

An additional third step is required when the projected QR code plane is partially outside the QR code's silhouette using the actual projection direction. This step determines those areas in the normal texture that are far from the camera and are outside of the mesh's silhouette. We similarly sample the modified normal vectors in those areas which are entirely out of the silhouette. Let k denote the number of areas entirely out of the silhouette, then we take the average unit normal vector $\vec{out}_{n_k}$ of each those area. This step is similar to the previous one, but other areas of the normal vector texture are sampled.

A single vector is created from the average vectors of the selected areas by retaking their average, and it is performed for both the second and third steps. The final normal vector $\vec{n_f}$ for updating the plane is calculated as the average of each steps' resulting vector,

$$\vec{n_f} = \frac{\sum_{i=1}^{m} \vec{n_i} + \sum_{j=1}^{4} \vec{depth}_{n_j}}{2}.$$

Before the iteration starts again, it updates the normal vector of the plane used for selecting the sampling area with $\vec{n_f}$. The later iterations use the previously found plane for selecting the sampling area on the surface. Figure 2 shows the texture sampling and how the new normal vector of the QR code plane is calculated without the additional third step. The method stops when the distance's variance between the surface and the centre of projection is small enough or it reaches the maximum number of iterations. We used 0.05 as a threshold value and 5 as a limit for the number of iterations in our implementation. When the threshold value was not reached, the projection direction with the smallest variance is used. The applied values are determined based on working with 5×5 cm large QR codes.

4. Automatically Determine the Size/Position of the QR Code

When the user specifies the centre C of the future QR code on the surface, the size of the future code can only be roughly guessed. In this subsection, we propose two methods to find a size that fits the QR code into the model's silhouette determined using the given projection, either with or without a preferred orientation of the QR code. The user needs to provide a QR code centre along with a projection for finding the QR code's size, or the resulting projection direction of our previously proposed search method can be used with the QR code's centre. The first step of both methods is finding the model's silhouette viewed from the given projection direction. To do this, we select all the facets that are visible using the given projection. Then, the silhouette of these facets is determined by creating a half-edge structure from them.

In our first method, the centre of the QR code is given by the user, but no orientation preference is provided. Therefore, we search the closest point of the silhouette (point S) to the centre of the QR code (point C). Point S can be easily determined as the touching point of a circle, which is centred in point C (see Figure 4a). The size of the QR code's edge can be calculated from the $\overline{SC}$ distance, half of the QR code's diagonal. Every QR code with an equal or smaller size can definitely be embedded onto the surface, so this is a guaranteed size. Also, the user can choose any rotation for embedding the QR code since the QR code is always inside the mesh's silhouette. Figure 3 shows finding the mesh's silhouette, the closest point to the QR code centre, and the created QR code with the calculated size. However, our method's resulting guaranteed size is not necessarily the absolute upper limit for the QR code.

In our second method, the user defines the orientation (the direction of the side) of the QR code by providing a $\vec{v}$ vector besides the centre of the QR code. We decided to use a direction for setting the QR code rotation because the edges of the QR code are very frequently planned to be parallel to a given $\vec{v}$ direction. After our method similarly found the model's silhouette to the first one, it places a small square with edges parallel to the user-given $\vec{v}$ direction at point C, which is the centre of the QR code. Then the square is enlarged by a central similarity from C until one of its vertices reaches the silhouette (see Figure 4b). The size of the QR code can be calculated analogously to the first method from the $\overline{SC}$ distance.

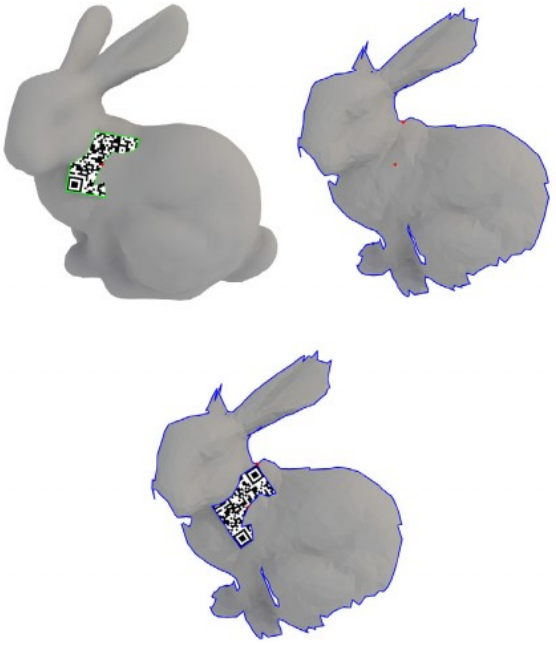

Figure 3: *It shows the mesh silhouette for a given QR code plane and the silhouette's closest point to the QR code centre. Every QR code smaller than this size can definitely be embedded to the surface using any rotation.*

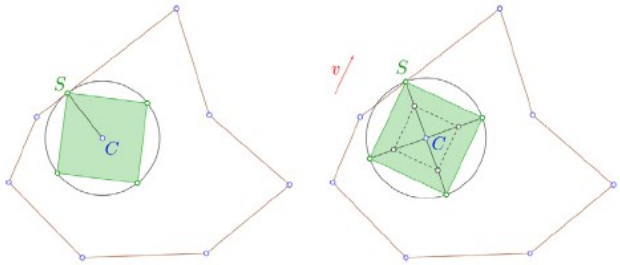

Figure 4: *Two examples of how the maximum size of the QR code is calculated using the centre C of the code and the silhouette's point S, In the first example, there is no preferred direction of the code (a), while a preferred direction was given by vector $\vec{v}$ (b) in the second one.*

Figure 5 shows that the plane of the QR code can be freely rotated around the user-selected C centre of the QR code when their guaranteed size is determined by our first method. For example, Figure 5a shows that the QR code is rotated in a way that one of its projected corners is on the silhouette. In the other case, we have selected a $\vec{v}$ vector to embed a QR code with sides parallel to this vector, which is rotated around the normal of the plane with 45° (see Figure 5b).

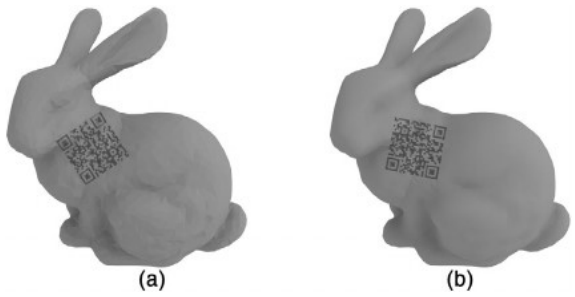

Figure 5: *Two rotated QR codes with the same centre. The left image shows when one of the QR code's plane vertices is on the silhouette of the shape (from the viewpoint of the projection centre).*

In the two scenarios discussed above, the user has defined the position of the centre of the QR code. An alternative scenario is when the user has no specific preference on the position of the centre of the QR code, but a suitable (practically a minimum) size of the code is given. In this case, the task is to find the area(s) on the entire surface where a potential centre for that size of a QR code can be chosen. Although this aspect is not at the forefront of our research and evidently needs further elaboration, here we provide a brief sketch of this scenario.

From a technical point of view, we have to determine those viewpoints from where the projected QR code of a given size is entirely inside the surface's silhouette. In the case of convex surfaces or surfaces with no sharp concave parts, a good guess could be the comparison of the projected area of the surface and the given area of the square of the code. If the area of the code is larger than the area of the surface from a particular viewpoint, then there is no chance to find a suitable position for the QR code. On the other hand, if the measure of the surface area from a given viewpoint is larger than the QR code, there is a good chance that we can place the code onto that part of the mesh. Recent studies provide a way of finding all viewpoints in the space around the given surface from where the actual visible area of the surface is of constant measure (here we suppose that the surface is given as a polyhedral mesh) [NKH18, CS16]. The set of these points are called an isoptic surface (of a given measure) of the mesh since this is somewhat similar to the well-known elementary planar case of isoptic circular arcs from where a given line segment can be seen under a certain angle. To find this surface, the authors apply heuristic spatial search methods (for the details, see [NKH18]).

Now we can determine the isoptic surface of the given mesh from where the silhouette measure is larger than the area of the square of the QR code. The next step, choosing any of the points of this isoptic surface, is to find the potential centres of the code from this specific viewpoint. To this step, we can compute the internal offset curve of the silhouette with half of the diagonal of the square of the code. The image of this offset curve on the mesh will roughly bound the suitable area of the centre from this viewpoint. Calculating these areas from the points of the isoptic surface, the union of these areas on the mesh provides all the potential positions of the centre of the QR code of that given size. Here we note again that this scenario evidently needs further study in terms of the details of the computation.

5. Embedding the QR Code

In both works of Peng et al., the selected area of the QR code is remeshed to achieve dense triangulation for the QR code and its surrounding area. Also, Kikuchi et al. use a preprocessing step to generate and insert the required knot lines and values for B-spline surfaces. These are all computationally expensive steps. We want to embed the QR code onto the triangulated mesh by significantly decreasing the preprocessing computational cost, remeshing only the facets on the borderline of the QR code to connect it with the rest of the mesh.

In our solution, we embed the QR code by projecting it onto the original mesh's surface. Our engraving method also uses the central projection defined by the QR code's plane for finding the projected points of the QR code's modules with ray-tracing. The modules are divided into submodules using an $n \times n$ grid. Similarly to [PLL*19], we applied a 5×5 grids for submodules. However, our proposed method differs from [PLL*19, PLL*20] in that we do not form a densely triangulated area for the QR code. Instead, we just project the submodules to the selected area on the surface for the next step, where the QR code is triangulated from the projected points. Therefore, there is no densely triangulated area around our embedded QR code, and we can directly connect the mesh with the triangulated code. The plane and its projection are used to find the model's facets affected by the embedding to achieve this direct connection. All those facets entirely inside the projection are discarded to replace with the triangulated QR code's modules. However, the projected points of the black modules and their submodules are shifted to create a small initial engraving before they are triangulated to form the embedded QR code. The borderline facets are modified to triangulate them with the projected points of the QR code's sides using constrained Delaunay triangulation. This way, our method can preserve most parts of the original mesh triangulation by connecting the border triangles directly to the QR code instead of the remeshed surroundings of the QR code. Figure 6 shows steps of the entire process. Our introduced method for merging the QR code with the rest of the mesh is similar to the one implemented by Peng et al. for connecting the remeshed QR code with the rest of the mesh. We presented the steps of our method since the QR code embedding was not described in detail in [PLL*19, PLL*20] for the readers.

6. Reducing Carving Depth: Patterns for the Black Modules

We propose replacing the rectangle shape of the black modules with different ones to decrease the carving depth of the embedded QR code. In the case of the 2D QR codes, the black modules are often modified to have a more artistic and pleasing look. Even the QR codes are beautified by using images. However, we want to shape our black modules to achieve a more shallow carving depth. Using different shapes for the module, we can control the number of connected black modules, and their areas.

Our module patterns were created using two different approaches. In the first one, each black module is replaced with a shape that stays

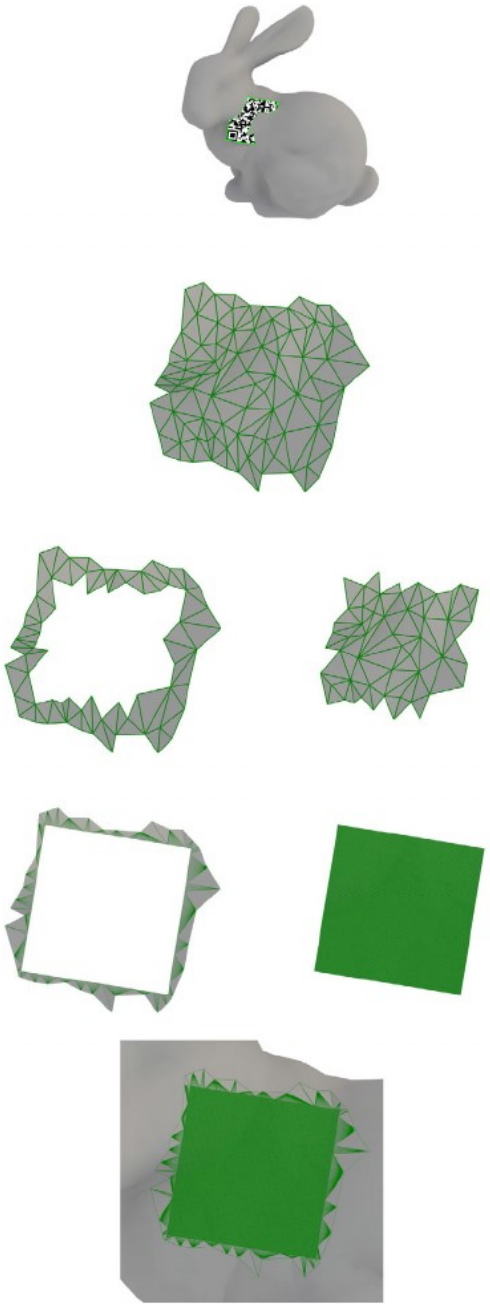

Figure 6: *First, the facets affected by the QR code are detected by embedding the QR code onto the surface. Next, the facets are separated into border and inner facets (inside the projected area). Then, the inner facets are removed and replaced with the triangulated QR code modules, while the border ones are modified and triangulated using the points from the projected sides of the QR code. Finally, combining the newly triangulated border, the embedded QR code, and the model's remaining facets results in the final mesh.*

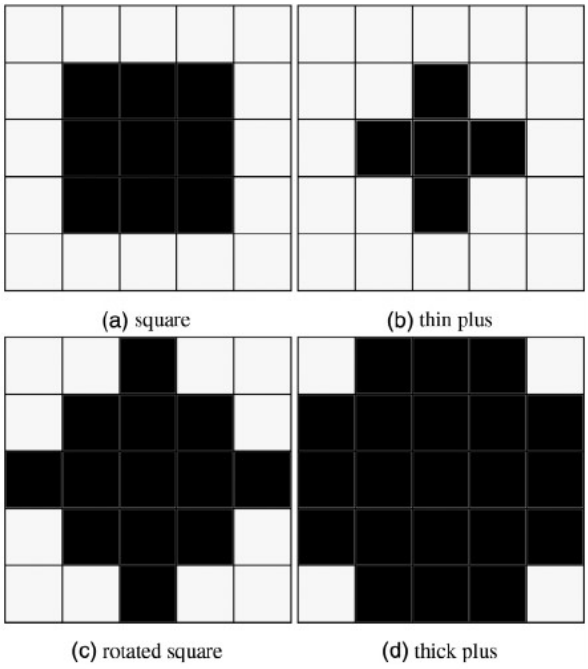

Figure 7: *The black squares show which submodule is enabled for engraving by the module patterns in the grids.*

inside the module's original rectangular shape. Therefore, it breaks the connected black areas in the QR code. In the second one, the shapes replacing the modules are designed to keep the connection between the neighbouring black modules but decrease their areas. Figure 7 shows all our proposed shapes for the modules. All of the embedded QR code from Figure 8 are created by carving only those submodules that are selected by the used patterns.

Our first two patterns separate the modules from each other and eliminate the large areas of the connected modules, which increase the carving depth (see Figure 7a and 7b). In the first one, only the inner submodules are enabled for embedding. Our second pattern reduces the number of the used submodules by removing the previous pattern's corners, leaving only a thin plus sign in the module's centre.

However, our next two patterns approach is the opposite because those keep the modules connected (see Figure 7c and d). The submodules in the corners are left out from the embedding to form obstacles in the large connected areas. Therefore, our third pattern is a more extensive 45° rotated square. The last one is similar to the second one because it is also a plus sign but larger. It uses the grid's inner horizontal and vertical lines.

The QR code's position detection markers play an important part in recognizing a QR code. Therefore, we propose different patterns for engraving them in our solution. Their engraving patterns combine two patterns used for the black modules. The position detection marker modules use the patterns from Figure 7c and 7d. The remaining four corner modules have a different, slightly shifted square pattern. Figure 9 shows the combination of these patterns in a position

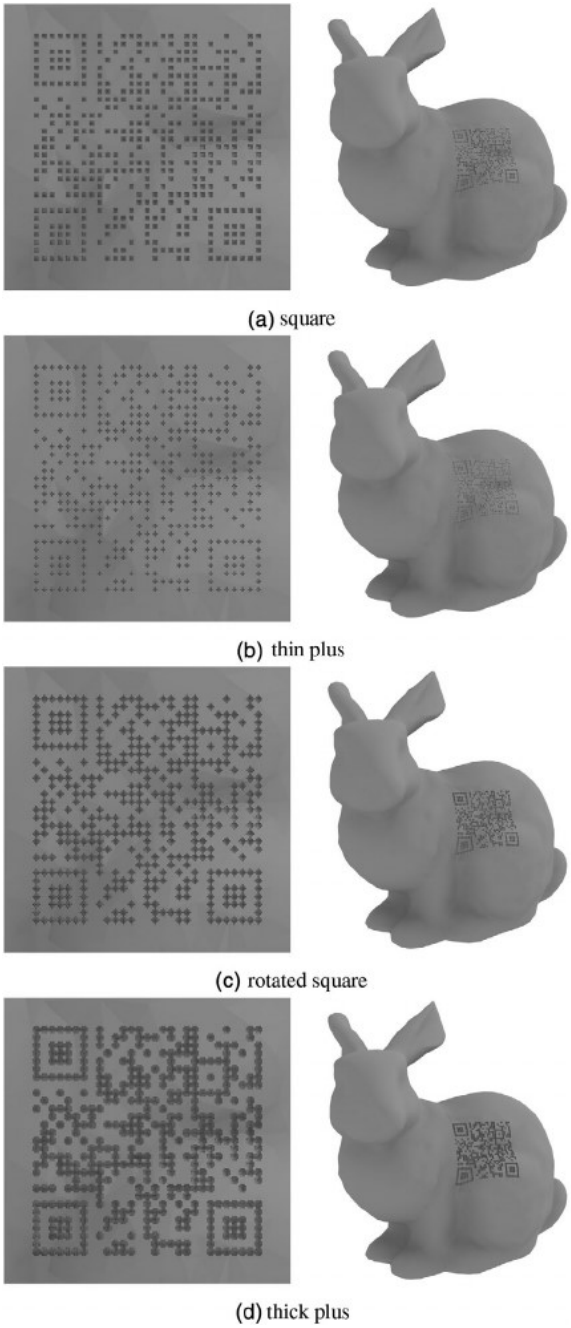

Figure 8: *Embedded QR codes using our proposed patterns. Only the enabled submodules by the applied pattern are engraved in the QR code.*

detection marker. Besides, it shows an example engraved QR code for each of them.

7. Carving Depth Search Method with Horizon-Based Ambient Occlusion

In the work of Kikuchi et al., the obscurance is evaluated at each grooving point to determine whether the engraved areas are dark

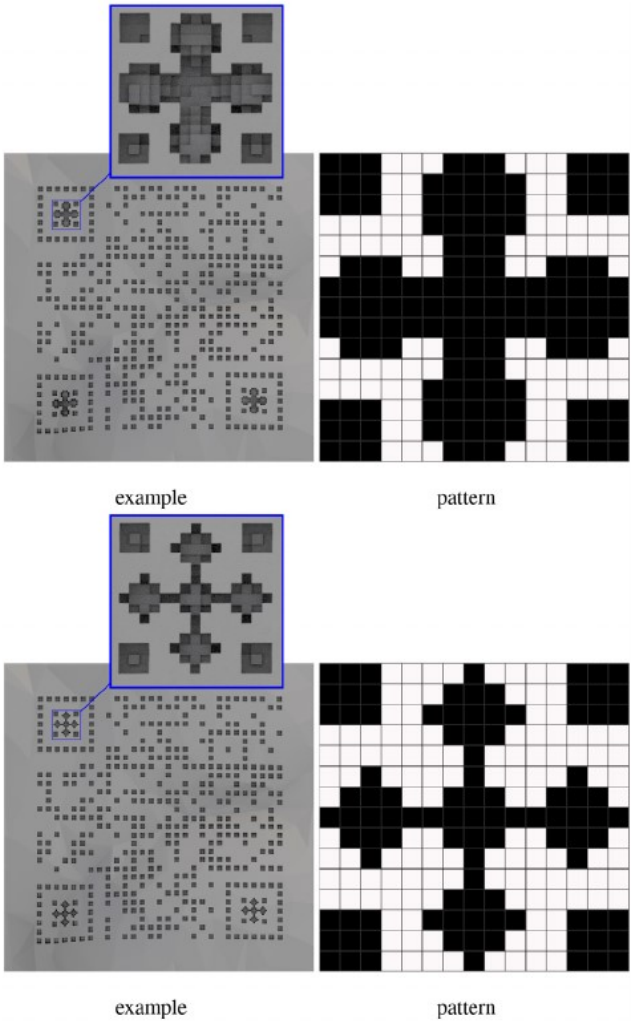

Figure 9: *The black squares in the grids indicate which of the submodules are enabled in an engraving pattern for the location detection marker.*

enough to identify them as the black modules of the QR code. On the other hand, Peng et al. used a different solution to find a carving depth that generates enough self shadow for the black modules. They calculate the amount of self shadow by using simulation, and the carving depth is decided by using the "symbol contrast (sc)" [fS15].

The methods of Peng et al. and Kikuchi et al. for finding the carving depth are time-consuming because it needs minutes to provides results. Comparing the running time of the carving depth search algorithms with 3D printing the embedded QR codes, we can see that finding the carving depth requires only a fraction of the time the 3D printer needs for making the model. Therefore it is essential to ensure a quick design process for embedding a QR code and viewing the resulting mesh, so the user can iteratively modify the embedded QR code until the result is satisfying. Our proposed algorithm aims to speed up the carving depth search phase of the design process while ensuring a good contrast between the mesh's surface and the

QR code's embedded modules. This way, the user can modify the embedded QR code more quickly to reach a satisfying result, a good candidate for 3D printing.

We created an iterative method like the ones from Kikuchi's and Peng's work for finding the carving depth. In our proposed method, we focused on improving the calculation of the self shadow inside the engraved modules. The QR code's projection direction and the direction to decode the QR code are usually the same or very similar. Therefore, we propose to use a Screen Space Ambient Occlusion (SSAO) technique to evaluate the self shadow instead of using simulation. The benefit of using an SSAO technique is that we can harness the GPU's computation capacity, along with the CPU. Our solution uses the Horizon-Base Ambient Occlusion (HBAO) technique to render the engraved QR code using the same projection used as the embedding. The rendered embedded QR code is stored in a texture and used to evaluate each engraved modules' illumination.

In our iterative depth search method, the QR's submodules are engraved in each iteration until the modules' darkness is not enough. The resulting texture of our rendering process, which stores the lighting information for each submodule of the QR code, is used to decide whether the darkness of the module is sufficient. The average luminance value of a submodule can be calculated using its corresponding pixels from this texture. In our implementation, the resulting pixels are from a 32-bit float texture with only a red channel. The texture is filled with the luminance value of the deferred shading's resulted RGBA colour. We use contrast criteria similar to [PLL*19].

We designed our method to handle those cases when the selected area for embedding a QR code is self-shadowed either because of self-occlusion or directed light sources. Therefore, our carving depth search method uses the average luminance value of a carved module's surrounding areas to decide when to stop. First, the submodules luminance value $LS_{k,l,i,j}$ is calculated using the resulting texture from the HBAO rendering pass. Then the modules' luminance value $L_{i,j}$ is defined by calculating the average of their submodules' value. These values are calculated at the beginning of each iteration to use these later in our objective function. In the next step, our method calculates the average luminance value $AL_{i,j}$ of a given module's surrounding areas by taking the average of the four darkest neighbour modules of the selected module, where the modules neighbourhood relation is defined with eight-connectivity. The resulting luminance value of the modules is used in our objective function to decide when to stop increasing the carving depth of a submodule. The objective function for a given module and its submodule is

$$G(k, l, i, j) = AL_{i,j} \cdot (-1 + c) + LS_{k,l,i,j},$$

where c is our threshold value parameter to have enough contrast in the resulting QR code and $LS_{k,l,i,j}$ is the luminance value of submodule $SM_{k,l}$ in module $M_{i,j}$. The carving depth of a given $M_{i,j}$ module $SM_{k,l}$ submodule is only increased during an iteration when $G(k, l, i, j) > 0$. We use the $G(k, l, i, j)$ function in the stopping criteria of our carving depth search method

$$\sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{n}\sum_{l=1}^{n} H(G(k, l, i, j)) \leq 0,$$

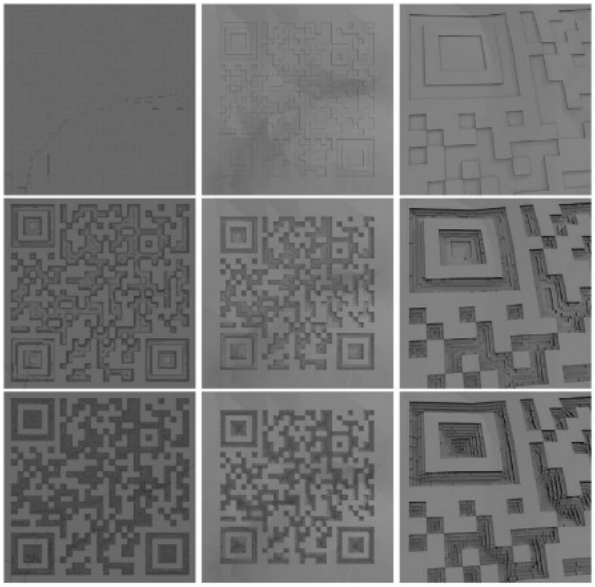

Figure 10: *The first column shows the output of our rendering process that is used during the carving depth search. In the following two columns, the engraved QR code can be seen in the actual state of the carving optimization.*

where m is the number of modules in the QR code's rows or columns, and n is the size of the $n \times n$ grid used for defining the submodules. Figure 10 shows the process of embedding QR codes using our method.

Instead of increasing the submodules carving depth with a constant value in each iteration, we propose a carving depth increase function. It is evaluated in each iteration and provides a dynamically changing value based on the number of iterations since the submodule's luminance value is not changed significantly. We use the tanh() function to manage the amount used to increase a submodule's carving depth in the actual iteration. Our function to manage the carving depth increase is

$$O(x) = c - c \cdot e\left(\tanh\left(\frac{1}{d} - f\right) + \tanh\left(f - \frac{x}{d}\right)\right),$$

where x is the number of iterations since the luminance value of the submodule is not changed significantly. We use constant values to control how aggressively the engraving process is performed, which needs to be defined before the carving depth search process. The c constant value is the default carving depth increase value, the d, e, and f values are used to transform the tanh() function. The used values are chosen to limit our function values around the minimum layer height available with the 0.4 nozzle's diameter value for 3D printing the QR-codes. Our parameters $c = 0.1$, $d = 2$, $e = 3$ and $f = 2.5$.

The HBAO technique only approximates the real shadow. However, we can estimate the modules' carving depth adequately with its result. Using the HBAO technique to calculate the amount of light in the embedded QR code, we can produce a result in 5–10 s

instead of minutes. However, the HBAO technique's parameters need to be chosen before the engraving so that a reasonable estimation of the realistic lighting can be achieved in the rendered result. The following section provides more results about how well our carving depth search method works with different lighting conditions and differently embedded QR codes.

8. Results

We implemented our proposed methods as a web application using Dart programming language with WebGL, and we tested it on a laptop with an Intel Core i7-8550U Processor (8M Cache, up to 4.00 GHz), 16GB memory, and an NVIDIA Geforce MX150 graphics card. The resulting models of our solution were printed out using a Prusa Mini 3D printer with a 0.4 and 0.25 mm nozzle diameter. Most of the models in the images were made using the 0.4 mm nozzle diameter, and we used a 1.2 mm QR code module size for embedding the code like in [PLL*19]. We provide the nozzle diameter and QR code module size for those created using different ones than the abovementioned models. The models were printed using PLA material since this is one of the most commonly used and cost-effective materials.

All of our embedded QR codes can be viewed on a web page[1] using the included obj viewer. Also, we placed a video there about decoding 3D printed embedded QR codes created using our proposed methods on the bunny and the knight model.

First, we tested how robust our proposed patterns are for changing the QR-code module size. We used the same module size in our test than Peng et al. in [PLL*19], and the results are shown in Figure 11. Replacing the shape of the black modules to decrease the carving depth affected how small module size can be chosen for the QR code that still results in a decodable 3D printed embedded QR code.

Figure 11 shows that the further we decrease the shape sizes in our pattern, the larger the QR code module is required to 3D print a readable QR code. This effect is also present in the case of our proposed location detection marker patterns because the smallest QR code module sizes were reached by not using our location detection marker pattern. Embedding a QR code using our thick plus module pattern has the smallest QR code module size (0.6 mm) in our test, while the largest module size was measured using our thin plus module pattern. Using our other two module patterns (square and rotated square) with our proposed location detection marker patterns for the embedding resulted in a 0.9 mm small QR code module size. However, the small module size for these patterns caused stringing in the embedded QR code, which we needed to remove for successful decoding. Figure 12 shows a few of our embedded QR codes with different patterns and QR code sizes.

Using the module patterns for the whole QR code resulted in the largest QR code module size. Changing the 0.4 mm diameter nozzle in the 3D printer to a nozzle with 0.25 mm diameter let us print the embedded QR code with more details and further decrease the QR code module size for some of our patterns as Figure 11 shows. Based on our test's results, we need to decide which one is more important,

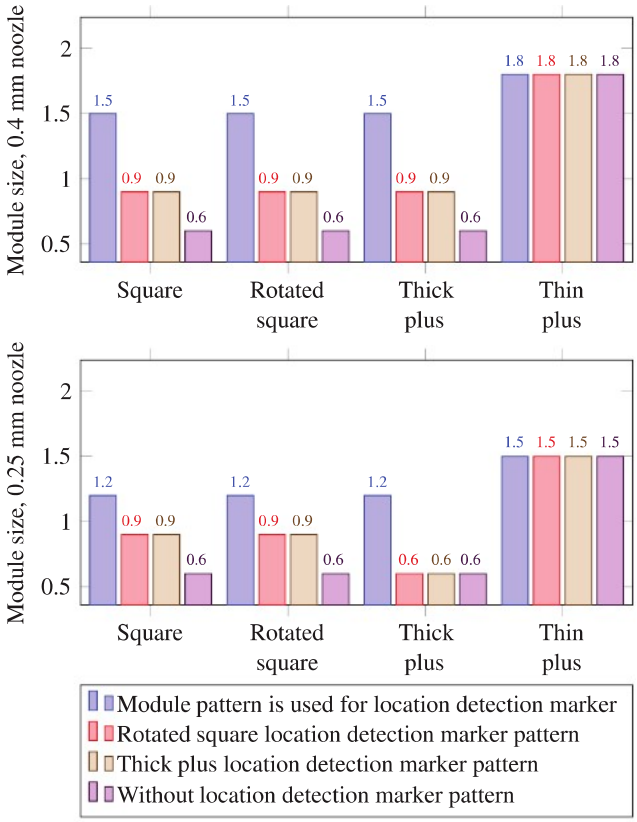

Figure 11: *The diagrams show the minimal QR code module size for each proposed pattern, which results in a decodable embedded QR code. We used a nozzle with 0.4 mm diameter for the first diagram, while we used a nozzle with a 0.25 mm diameter for the second one. The module sizes for testing our pattern are used from [PLL*19].*

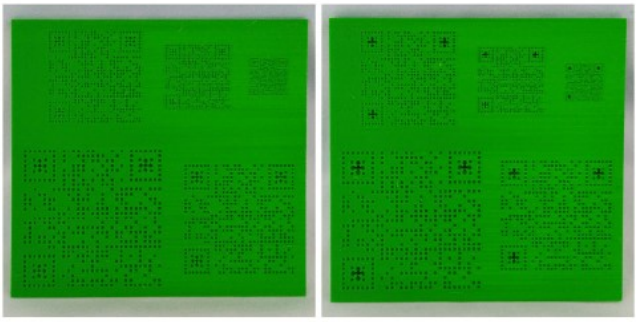

Figure 12: *QR codes embedded using our square module pattern with the rotated square and thick plus location detection marker patterns. The QR codes are 3D printed in different QR code module sizes (first row: 1.2 mm, 0.9 mm, 0.6 mm, and second row: 1.8 mm, 1.5 mm).*

[1] https://arato.inf.unideb.hu/papp.gyorgy/qr_in_mesh/

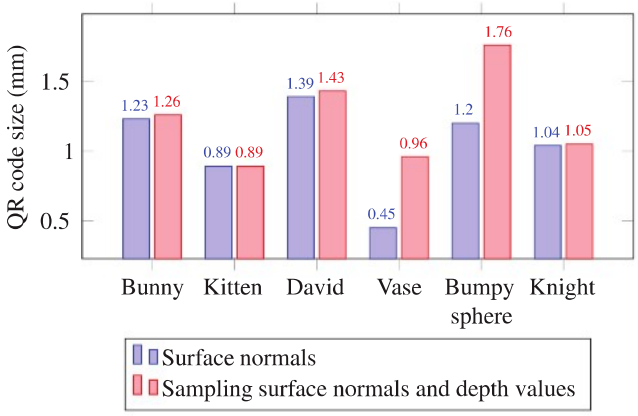

Figure 13: *It shows the guaranteed size of the QR code found using our two different methods for searching the projection direction.*

the size of the QR code or its carving depth. Then we can choose the most suitable pattern based on our decision for the embedding.

We proposed two projection direction search methods to automatically determine the direction for embedding a QR code at the given centre. We also presented a method that can automatically determine the QR code's size using a projection direction and a selected point on the mesh for the centre of the QR code. Our first method provides the largest size that does not exceed the mesh from the given projection direction and lets the user freely rotate the QR code before embedding it without any problem. Our second solution needs an additional $\vec{v}$ vector that defines the orientation of the QR code for searching a similarly large size than in our first method. We searched the QR code's guaranteed size in our next test using both our proposed projection direction search methods. The results are used to show the projection direction search methods' resulting QR code plane and compare which one provides a larger potential QR code size. Figure 13 shows the results of our test.

The embedded QR code and the QR code planes of the search methods are shown in Figure 14. Additional images for this comparison is added to the appendix in Figure A.1. Most of the shown models and QR code centres from Figure 14 and A.1 are used in our later tests for embedding QR codes.

The figures show that a larger guaranteed QR code size can be achieved when the surface is highly curved, and we include the depth values in the projection direction search. However, the difference between our two proposed methods becomes insignificant when the selected surface for the QR code is not highly curved or it is wavy. Therefore, we used our second projection direction search method to embed the QR code in the following test.

Our patterns for engraving a QR code were proposed to decrease the embedded QR code's carving depth. Therefore, we measured the carving depth of QR codes embedded with or without using our patterns (see Figure 15). Furthermore, we measured the carving depth of embedded QR codes engraved by combing the location detection marker patterns with the module patterns. These measurements are shown in Figure 16.

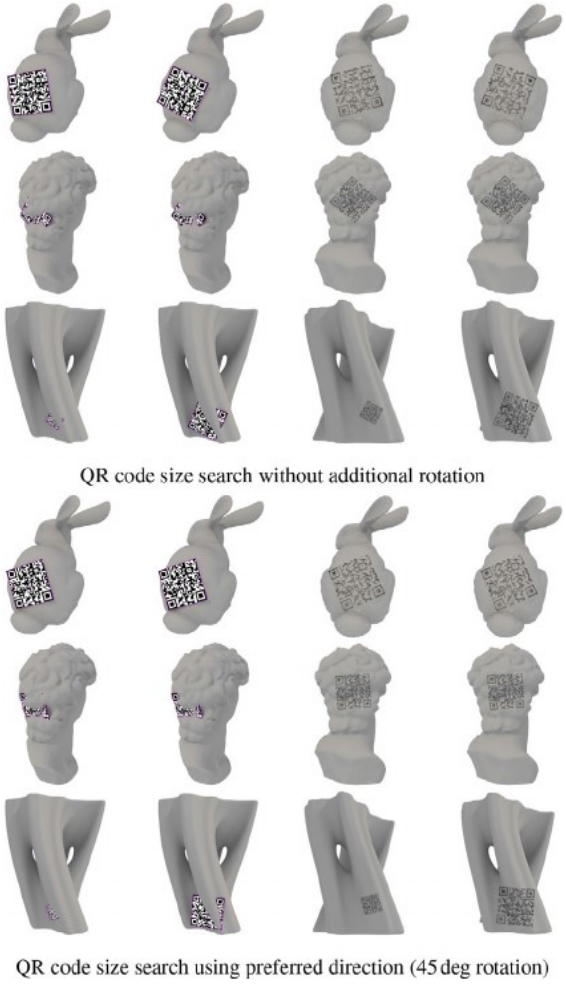

Figure 14: *Results are shown on the Bunny, David's head, and Vase model. The QR code size search was performed with two rotations. First, we compare the QR code plane found in each row, using only the surface normal for the projection direction search (first and third column), then additionally using depth values in the search (second and fourth column).*

Our results show that replacing the black modules' shape with our patterns decreases the needed carving depth to reach the required contrast between the QR code's black and white modules. The not connecting patterns (small thin, and square) produced the shallowest carving depth in all models. The best performing pattern was the thin plus, while the least performing was the thick plus pattern in our test. The square and rotated square patterns are close to each other, and the gap between their carving depth decreases as the selected area is a more curved surface.

Figure 16 shows the carving depth measurements of QR codes embedded using module and location detection marker patterns as well. The chart shows that not using location detection marker pattern results in a deeper carving depth for each module pattern. Also, it shows that our proposed patterns for the location detection marker

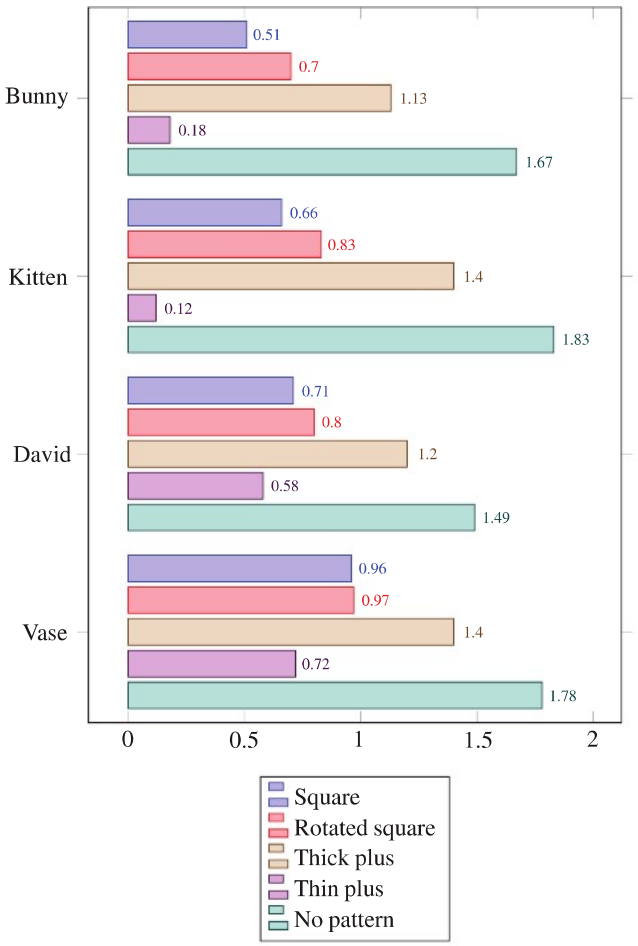

Figure 15: *Carving depth of embedding a QR code with different patterns.*

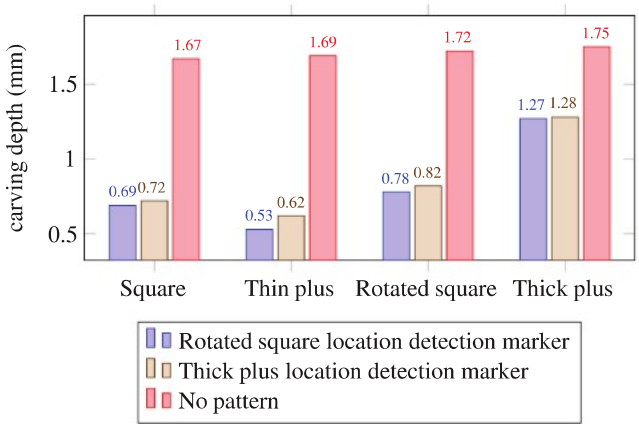

Figure 16: *It compares the resulting carving depth of embedding a QR code by combining the module patterns with the location detection marker patterns.*

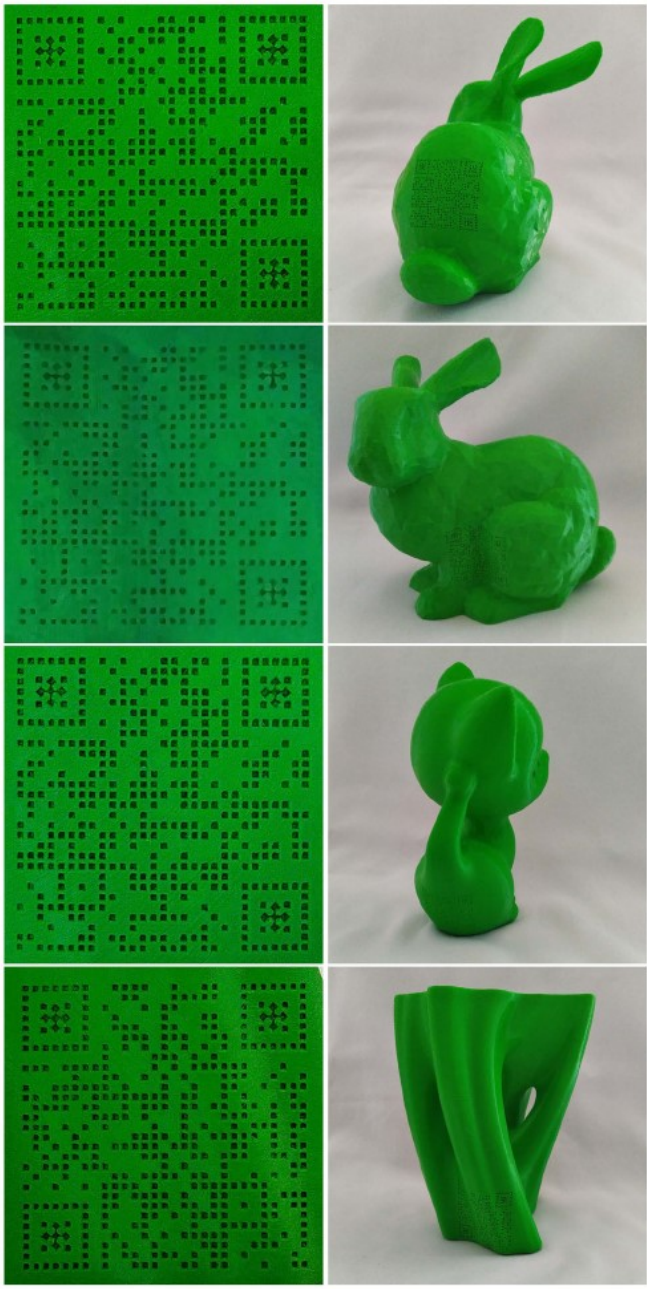

Figure 17: *3D printed embedded QR codes on different models using our module and location detection marker patterns together.*

decrease the carving depth, and the difference between our two patterns' carving depth is small.

We compared the readability of the embedded QR codes, which are created with our projection direction search method and patterns, using an Android smartphone with the QRbot application. Figure 17 shows 3D printed embedded QR codes combining the square module and rotated location detection marker patterns on different models. More images of embedded QR codes onto the surface

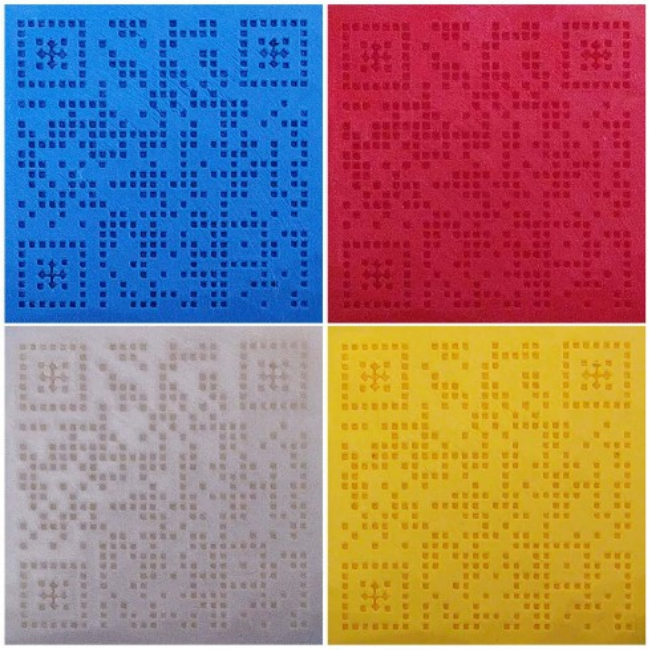

Figure 18: *3D printed embedded QR codes in different colours.*

of David's head model using our patterns are shown in the appendix (Figure A.2). We used the same models from the [PLL*19, PLL*20] works to compare our projection direction method to the previous works. Our test also shows how our pattern for position detection marker performs in QR code decoding compared to not using any pattern or using one of our previously mentioned patterns for modules in the whole QR code.

We tested our proposed projection direction method and patterns using different colours like the previous works. We show the 3D printed embedded QR codes in Figure 18. The images show that changing the colour does not affect the readability of our embedded QR code significantly.

We also tested our introduced method on new models. We choose the knight chess figure with a pyramid and a star mesh to embed a QR code onto their surface. The resulting 3D printed models and QR codes are shown in Figure 19. We created the last embedded QR code without using any pattern to show the robustness of our proposed methods.

Replacing the location detection markers with the module patterns results in an unreadable embedded QR code. However, increasing the module size solves the problem in exchange for having a more considerable carving depth. Our proposed location detection marker has better results than using the module patterns. It reduced the carving depth while the QR code remained readable.

Two of our module patterns, the square, and the rotated patterns, performed well during the readability test. The thick pattern produced a readable embedded QR code only when the selected surface was not highly curved or wavy. On the other hand, our thin plus module pattern resulted in too small shapes with the 1.2 mm module size, and the embedded QR code was not readable. The test

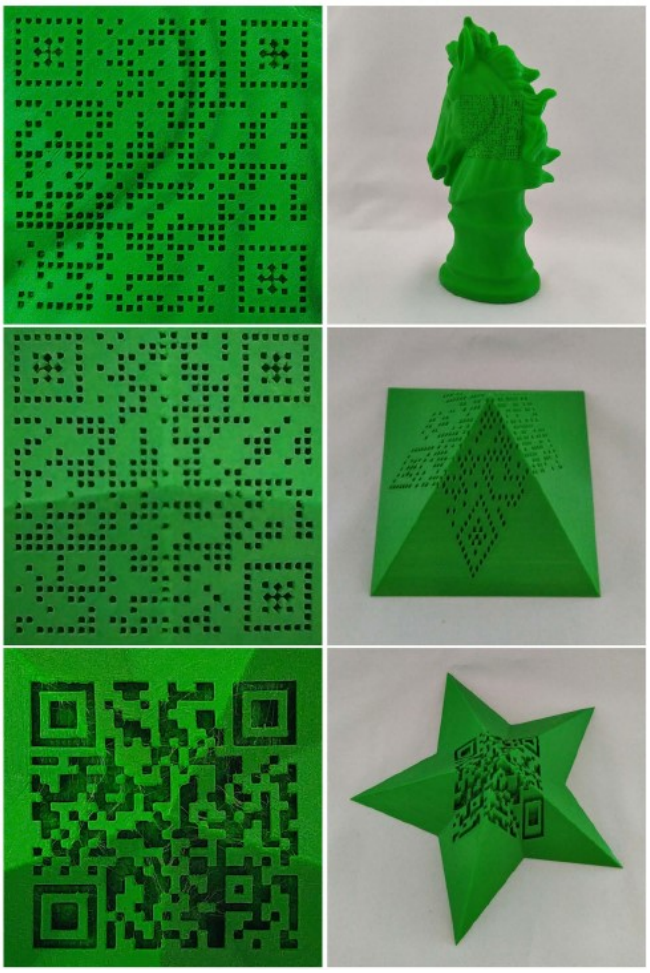

Figure 19: *3D printed embedded QR codes on different models. The first three QR codes were embedded using our proposed patterns, while the last one was created without a pattern to show our proposed method's robustness.*

also showed that the readability of QR codes embedded using our patterns are strongly depends on the quality of the resulted 3D print. Therefore, our patterns need to be printed with higher detail settings to have a readable QR code. Based on our test results, the best combination of our patterns considering their carving depth and readability are the square and rotated module pattern used with one of our proposed location detection marker patterns.

We also tested the robustness of decoding the embedded QR code created using our proposed patterns. We used a 1.2 mm QR code module size, which limited the test to the square, rotated square, thick plus module pattern with both our location detection marker patterns because only these can produce decodable results with the chosen QR code module size. The robustness of the embedded QR code decreases by selecting a pattern that contains fewer submodules enabled for engraving. However, this decrease in the robustness is not significant. The app can still successfully decode the embedded QR code, but it requires an additional 1 or 2 s.

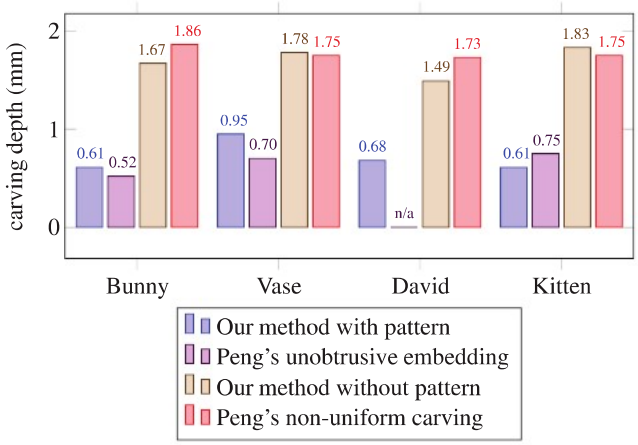

Figure 20: *Comparison of our methods with the two embedding methods of Peng et al.*

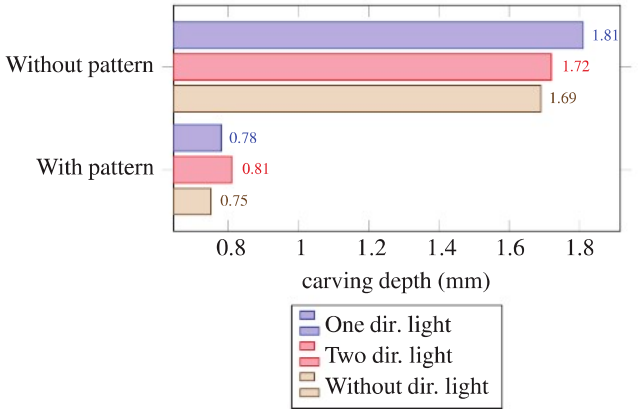

Figure 21: *It shows the comparison of the carving depth of embedding a QR code in different lighting environments with directional lights.*

Figure 20 shows the result of comparing embedded QR code's carving depth created using our method with [PLL*20] and [PLL*19]. In our method we used our best performing pattern for this comparison.

Engraving a QR code using our module patterns and our HBAO based carving depth search method produces an embedded QR code with a smaller carving depth than in [PLL*19]. The embedded QR code from [PLL*20] has the smallest carving depth in most of our test cases. Our HBAO carving depth search method finds similar carving depth for the engraving when the QR code is embedded without patterns as in [PLL*19]. However, it only needs a fraction of the time required to [PLL*20] and [PLL*19].

We tested how our carving depth search method works when directional light sources are used. The QR code was embedded using one or two directional lights, and their results were compared to the case where no directional light was used (see Figure 21).

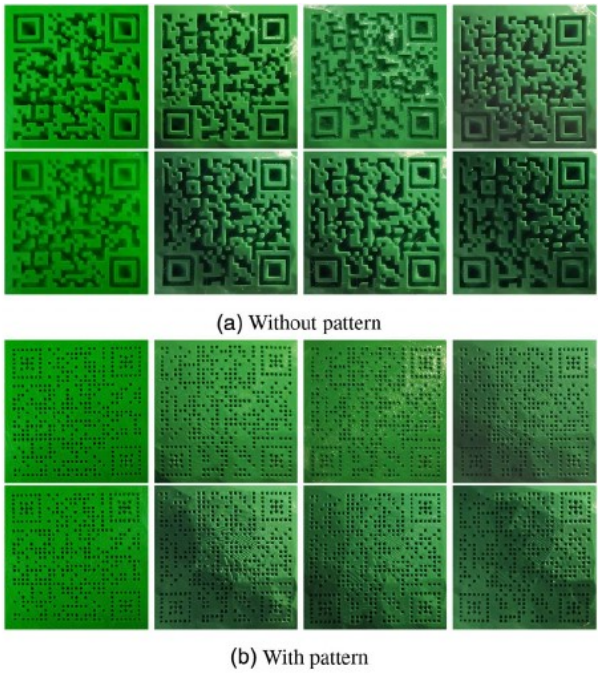

Figure 22: *The first column shows the embedded QR code without any directional light. The second one shows it with the directional light used during the engraving. For the last two columns, the direction's light is above and below the engraving light direction. The enclosing angle of the engraving light's direction and the QR code plane's normal is different for the embedded QR codes in the first and second row. First, this angle is small, then in the next row, it is set to large for both cases, either using our pattern or not.*

As our measurements show, engraving QR codes with additional directional light sources results in larger carving depth. Using additional directional light for the engraving increases the carving depth when patterns are used for the embedding. However, when the QR code is embedded without a pattern, the carving depth decreases by adding further directed light to the engraving.

We also tested how robust is our carving depth search method to the changes in the lights' direction. Figure 22 shows examples of using different light direction for decoding than the one used in the carving depth search method.

Figure 22 shows that the chosen light direction determines how robust the QR code is against changes in the lighting environment where the QR code is decoded. Using directional light and our patterns together resulted in unreadable QR codes. However, when no patterns were used, the larger the angle between the QR code plane's normal and the light's direction of the engraving process we choose, the more robust the embedded QR code becomes to changes in the light's direction during the decoding. Unfortunately, increasing this angle also increases the embedded QR code carving depth. Therefore, we need to sacrifice the small carving depth in exchange for having a more robust QR code in an environment with directional lights. In our test, the QR code engraved using a directional light with a larger angle (123°) tolerated increasing the engraving incom-

ing light angle for the decoding test with 30°. The other QR code engraved with a smaller direction light angle (98°) only tolerated 15° larger angle for the incoming light. Further increase of the incoming light enclosed angle with the QR plane normal resulted in an unreadable QR code. Decreasing the incoming light direction angle does not affect the readability of the QR codes engraved without a pattern.

9. Conclusion

This paper introduces improved methods for embedding a QR code onto triangulated meshes' surfaces. First, we proposed a solution that automatically finds the projection direction for engraving a QR code with a given centre and a guaranteed size. Then, we compared our two methods for finding a projection direction for the QR code embedding using models from previous works. We also used new models to show the robustness of our proposed methods. Our results show that decodable, deformation-free embedded QR codes can be achieved using our proposed projection direction search methods. It also shows that using the depth information of the model with its surface normals for finding a projection direction can result in a larger QR code size when the selected region is highly curved. In Our embedding solution, we also subdivide the modules as in [PLL*19] for the carving depth search process.

To further decrease the embedded QR code's carving depth, we propose using patterns for the black modules that define which sub-modules in a black module must be included in the carving depth search and embedding process. Using these patterns for the embedding, we can reach smaller carving depths than in [PLL*19], and we are close to the carving depth introduced in [PLL*20]. Our test showed that we reached the best result when our proposed module pattern and the location detection marker were used together. The best performing combinations of our proposed patterns are the square or rotated square module pattern with one of our location detection marker patterns because there is no significant difference between them.

We use a Screen-Space Ambient Occlusion technique in our carving depth search solution, the Horizontal Based Ambient Occlusion, to estimate the embedded QR code's self shadow. The luminance value is calculated from the resulting texture of rendering the QR code from the projection direction, and it is used for the objective function of our iterative search method to decide when to stop the iteration. We also defined a dynamic carving depth increase function to decrease our method's running time. As a result, our method can find an embedded QR code's carving depth only in seconds, while the previous methods needed minutes.

Furthermore, we can define directional light to include in the carving depth search process, so the resulting embedded QR code can be decoded even with directional light. However, our test result shows that the engraved QR code robustness depends on the chosen light direction. Also, using direction light in our carving depth search process increases the embedded QR code's carving depth. Besides these, our solution depends on the parameters of HBAO to adequately estimate the shadow in the modules.

In the future, we plan to use the ray-tracing support of the newest graphic cards to provide a more accurate and faster result for the QR code embedding. Also, our QR code size search algorithms can only find a guaranteed size, which is, in most of the case, not the largest possible size for embedding the QR code. A search algorithm to find a rotation for the QR code placement can be implemented to increase the QR code size as future work. Our iterative projection direction search methods can also be improved using a search method. For example, the simulated annealing can be used to reduce the difference between the depth values during the projection direction search. Besides, our normal vector sampling methods may be further improved by using more sophisticated algorithms to interpolate the normal vectors between the mesh vertices.

Acknowledgements

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.

Appendix A: Additional 3D printed embedded QR codes

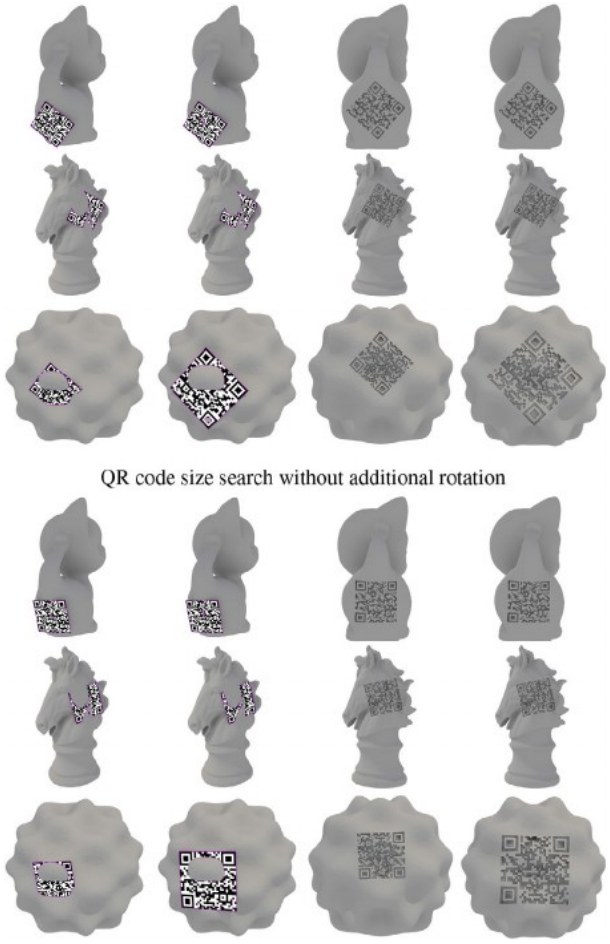

Figure A.1: *Additional models tested with our projection direction search method: the Kitten, Knight, and the Bumpy sphere. For further explanation see Figure 14.*

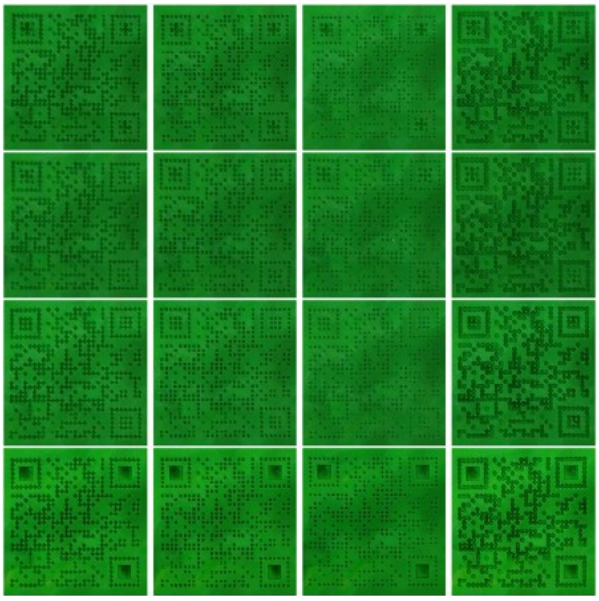

Figure A.2: *3D printed QR code embedded onto the surface of David's head model. Each column shows a different module pattern, while each row shows a different location detection marker pattern.*

References

[BSD08] Bavoil L., Sainz M., Dimitrov R.: Image-space horizon-based ambient occlusion. In *ACM SIGGRAPH 2008 Talks* (New York, NY, USA, 2008), SIGGRAPH '08, Association for Computing Machinery.

[CCLM13] Chu H.-K., Chang C.-S., Lee R.-R., Mitra N. J.: Halftone qr codes. *ACM Transactions on Graphics 32*, 6 (2013), 217:1–217:8.

[CLT*19] Chen F., Luo Y., Tsoutsos N. G., Maniatakos M., Shahin K., Gupta N.: Embedding tracking codes in additive manufactured parts for product authentication. *Advanced Engineering Materials 21*, 4 (2019), 1800495. https://doi.org/10.1002/adem.201800495.

[CS16] Csima G., Szirmai J.: Isoptic surfaces of polyhedra. *Computer Aided Geometric Design 47* (2016), 55–60. SI: New Developments Geometry.

[fS15] for Standardization I. O.: *Information technology – Automatic identification and data capture techniques – QR Code bar code symbology specification*. Standard ISO/IEC 18004:2015, International Organization for Standardization, 2015.

[GALV14] Garateguy G. J., Arce G. R., Lau D. L., Villarreal O. P.: Qr images: optimized image embedding in QR codes. *IEEE Transactions on Image Processing 23*, 7 (2014), 2842–2853.

[GS15] Gaikwad A. M., Singh K. R.: Embedding qr code in color images using halftoning technique. In *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)* (2015), IEEE, pp. 1–6.

[KYJ*18] Kikuchi R., Yoshikawa S., Jayaraman P. K., Zheng J., Maekawa T.: Embedding QR codes onto b-spline surfaces for 3d printing. *Computer-Aided Design 102* (2018), 215–223. ID: 271502.

[LLC13] Lin Y.-S., Luo S.-J., Chen B.-Y.: Artistic qr code embellishment. *Computer Graphics Forum 32*, 7 (2013), 137–146. https://doi.org/10.1111/cgf.12221.

[LMHW19] Li K., Meng F., Huang Z., Wang Q.: A correction algorithm of qr code on cylindrical surface. In *Journal of Physics: Conference Series* (Jun 2019), vol. 1237, IOP Publishing, p. 022006.

[LNNZ17] Li D., Nair A. S., Nayar S. K., 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, 2017), UIST '17, ACM, pp. 449–460.

[LSGH13] 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 (ASID)* (2013), IEEE, pp. 1–5.

[LWH16] 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 (ISNE)* (2016), IEEE, pp. 1–3.

[LWHL15] 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 (ICSIPA)* (2015), IEEE, pp. 389–393.

[LWLJ17] Lin L., Wu S., Liu S., Jiang B.: Interactive QR code beautification with full background image embedding. In *Second International Workshop on Pattern Recognition* (2017), Jiang X., Arai M., Chen G., (Eds.), vol. 10443, International Society for Optics and Photonics, SPIE, pp. 211–215.

[LZ17] 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 (ICSIPA)* (2017), IEEE, pp. 39–42.

[Mit07] Mittring M.: Finding next gen: Cryengine 2. In *ACM SIGGRAPH 2007 Courses* (New York, NY, USA, 2007), SIGGRAPH '07, Association for Computing Machinery, p. 97–121.

[NAM*17] Nalbach O., Arabadzhiyska E., Mehta D., Seidel H. P., Ritschel T.: Deep shading: convolutional neural networks for screen space shading. *Computer Graphics Forum 36*, 4 (2017), 65–78. https://doi.org/10.1111/cgf.13225; 20.

[NKH18] Nagy F., Kunkli R., Hoffmann M.: New algorithm to find isoptic surfaces of polyhedral meshes. *Computer Aided Geometric Design 64* (2018), 90–99.

[PHP21] Papp G., Hoffmann M., Papp I.: Improved embedding of QR codes onto surfaces to be 3d printed. *Computer-Aided Design 131* (2021), 102961.

[PLL*19] Peng H., Lu L., Liu L., Sharf A., Chen B.: Fabricating qr codes on 3d objects using self-shadows. *Computer-Aided Design 114* (2019), 91–100.

[PLL*20] Peng H., Liu P., Lu L., Sharf A., Liu L., Lischinski D., Chen B.: Fabricable unobtrusive 3d-qr-codes with directional light. *Computer Graphics Forum 39*, 5 (2020), 15–27.

[Tim13] Timonen V.: Line-sweep ambient obscurance. *Computer Graphics Forum 32*, 4 (2013), 97–105. https://doi.org/10.1111/cgf.12155.

[WSHL18] Wei C., Sun Z., Huang Y., Li L.: Embedding anti-counterfeiting features in metallic components via multiple material additive manufacturing. *Additive Manufacturing 24* (2018), 1–12. ID: 306190.

[WYL*15] 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 (ICOT)* (2015), IEEE, pp. 161–164.

[XSL*19] Xu M., Su H., Li Y., Li X., Liao J., Niu J., Lv P., Zhou B.: Stylized aesthetic qr code. *IEEE Transactions on Multimedia 21*, 8 (2019), 1960–1970.

[YPLL19] Yang J., Peng H., Liu L., Lu L.: 3d printed perforated qr codes. *Computers & Graphics 81* (2019), 117–124. ID: 271576.

[ZXL*20] Zhang D., Xian C., Luo G., Xiong Y., Han C.: Deepao: Efficient screen space ambient occlusion generation via deep network. *IEEE Access 8* (2020), 64434–64441.