Constructing Printable Surfaces with View-Dependent Appearance

MAXINE PERRONI-SCHARF, Princeton University, USA
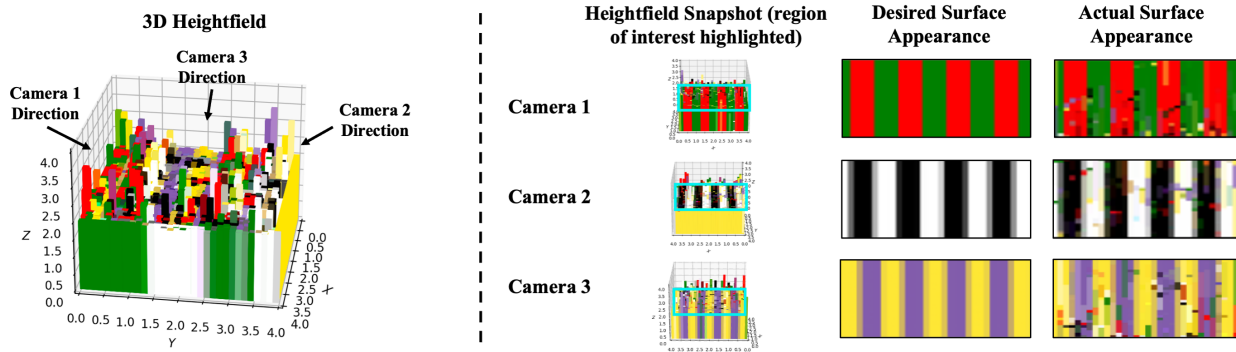SZYMON RUSINKIEWICZ, Princeton University, USA

Fig. 1. Left: A heightfield in which heights and colors have been optimized by our method in order to produce desired appearances from 3 different directions as indicated. Right: Snapshots of this heightfield from the 3 viewing directions together with the desired and rendered appearances of the regions of interest.

We present a method for the digital fabrication of surfaces whose appearance varies based on viewing direction. The surfaces are constructed from a mesh of bars arranged in a self-occluding colored heightfield that creates the desired view-dependent effects. At the heart of our method is a novel and simple differentiable rendering algorithm specifically designed to render colored 3D heightfields and enable efficient calculation of the gradient of appearance with respect to heights and colors. This algorithm forms the basis of a coarse-to-fine ML-based optimization process that adjusts the heights and colors of the strips to minimize the loss between the desired and real surface appearance from each viewpoint, deriving meshes that can then be fabricated using a 3D printer. Using our method, we demonstrate both synthetic and real-world fabricated results with view-dependent appearance.

CCS Concepts: • **Computing Methodologies → Image Manipulation**.

Additional Key Words and Phrases: Digital Fabrication, Ultraviolet Printer

1 INTRODUCTION

Recent advances in 3D printing technology, including the ability to print the entire color spectrum at very high resolutions, have enabled many new applications in digital fabrication. One such application of these capabilities is making directly printable 3D objects whose surface appearance changes dynamically with viewing angle to create multi-image displays. Such tangible displays have applications in various communication tasks and art. For example, one could use such a surface to display co-located AprilTags on one small square [Olson 2011], or to make color changing artwork [Lee et al. 2022; Weissman 2015] and interactive objects.

The current state of the art solution for this task, "Lenticular Objects" [Zeng et al. 2021], employs a UV printer to print 3D objects covered in tiny lenses. Underneath each lens lies an array of colored dots. The lenses then cause different colors to appear based on viewing direction, allowing multiple images to be displayed on a single object's surface. However, this method requires the use of clear material, and the printed objects appear to have dark a hexagonal overlay around the edges of each lens. These objects also require a polish coating to be painted on them after printing, and the visual effects of the surfaces are very sensitive to the type of coating used. Furthermore, lens-based approaches can only a achieve low resolutions: with lenticular objects, the resolution of the images is limited by the requirement that each lens be large enough to cover a given number of dots (equal to the number of viewing directions).

Our work aims to achieve higher printable resolutions than prior approaches by using a self-occluding colored heightfield that can be directly fabricated with UV 3D printers and require no additional fabrication steps such as lenses or polish. In the heightfield, certain strips will obstruct each other from different vantage points, creating view-dependent effects such as those shown in Fig. 1. We propose a machine-learning based method for automatically creating these heightfields based on the desired surface appearance. At the heart of our method lies a novel differentiable rendering algorithm tailored specifically to 3D heightfields.

We present a suite of various optimization techniques for self-occluding heightfields. To better search for a global optimum of

our system, we use coarse-to-fine stochastic-gradient-descent (SGD) interspersed with simulated annealing. Furthermore, we employ alternating block coordinate descent to improve the accuracy of our results. We regularize our surfaces to make them suitable for 3D printing with extra barrier loss and neighbor loss terms in our objective function. We also conduct a series of experiments to validate the effectiveness of these various techniques and the limitations of our approach in terms of number and range of viewing directions, accuracy and resolution.

In this work we contribute:

(1) A novel special-purpose differentiable renderer designed for self-occluding heightfields.
(2) A tailored SGD-based optimization algorithm that includes coarse-to-fine surface subdivision, alternating block coordinate descent, steps of simulated annealing, and surface regularization, which is effective for optimizing self-occluding heightfields.
(3) A demonstration of the effectiveness of our algorithm on synthetic and real-world 3D printed results.

2 RELATED WORK

View Dependent Appearance and Fabrication. Previous research employs a variety of methods to achieve surfaces capable of displaying multiple images. Pjanic and Hersch [2015] propose a two metal-printing method that uses superposition of horizontal and vertical colored lines to create surfaces which appear to change color with view direction. Their results, however, have a grainy quality. Also, the approach is specific to metal media and a maximum of two distinct images can be embedded into a single surface.

Self-occlusion is a popular technique for displaying multiple images or changing images on a single surface. Snelgrove et al. [2013] employs parallax walls to cause certain colors to appear depending on the direction of incident light on the surface. Unlike our method, the heights of these walls are regular and fixed regardless of the desired input images, making the method constrained to changes in appearance across only one axis. In one of the closest existing works to ours, Sakurai et al. [2018] uses a UV printer to create tiny heightfields that induce self-occlusion by creating several subcells for each desired color at each point in the image and designing walls that block the colors of certain subcells from certain viewing directions. This method, however, compromises image quality, as the black walls and fragments of the other colors are visible from each view angle, creating a grainy texture. This method is also explicitly limited to a low number of views and cannot exploit the benefits of shared pixel color across viewpoints, whereas our method can.

Lenticular lens surfaces can also be used to create surface appearance that changes with view angles. Most recently, Zeng et al. [2021] demonstrate printing lenses directly onto 3D objects to cause changing surface appearance. Due to the nature of the lenses, these surfaces have the appearance of a thin dark hexagonal lattice overlay due to the shadows between the lenticular lenses, whereas our method produces no dark regular artefacts or shadows.

Klehm et al. [2014] and Nindel et al. [2021] both propose approaches for respectively optimizing synthetic and 3D printable surfaces by changing the material color and opacity of the surface per voxel to achieve a desired appearance. However, unlike our approach, neither focus on updating surface geometry to create view-dependent appearance.

Self-Occlusion and self-shadows. The technique of fabricating complex surfaces that have self-occluding or self-shadowing properties has also been previously used in a variety of applications outside of fabricating multi-image displays. Alexa and Matusik [2011] uses self-shadowing heightfields to produce images that vary based on the direction of incident light. Additionally, Alexa and Matusik [2012] exploits shadowing to dither images by placing irregular pits on surfaces where the deeper the pit, the darker the appearance at that point on the surface. Along these same lines, Peng et al. [2019] fabricates indented 3D surfaces to create QR codes on 3D objects. Our current algorithm does not consider the effects of self-shadowing, but as a future extension of our work it could be augmented to do so using a similar approach to that in [Alexa and Matusik 2011].

Applications of Digital Fabrication. The process of applying graphics to the physical world has been the subject of a longstanding and growing area of research [Salisbury Jr 1999; Séquin 2013], with a number of studies focusing in particular on creating surfaces with unique and unusual optical effects. Papas et al. [2011] uses the technique of indenting surfaces in order to manipulate incident light so that when light shines through the surface, a desired image appears on a projection plane. Regg et al. [2010] fabricates 3D holograms by indenting parabolic and hyperbolic grooves into specular materials. Pereira et al. [2017] uses magnetic flakes embedded in resin to print surfaces with anisotropic appearance. Weyrich et al. [2009] optimizes microfacet heightfields to replicate desired reflected highlight shapes, and manufactures these surfaces with a milling machine.

Differentiable Rendering. There is a large body of previous work that designs and employs differentiable rendering on meshes for a variety of applications [Kato et al. 2020]. To tackle discrete discontinuities that occur from occlusion and object boundaries, one can facilitate gradient computation by approximating the rendering forward pass with a smooth function. For example, Rhodin et al. [2015] fades the density of objects at boundaries, and Liu et al. [2019] proposes the *Soft Rasterizer*, which utilizes spatial blurring and a probabilistic pixel color aggregation method. Recently, Petersen et al. [2022] presents a generalized family of differentiable renderers that employ a large variety of sigmoid functions to approximate the Heaviside stepwise function. In our work we also utilize a smooth Heaviside stepwise function approximation for differentiable rendering. However, unlike previous methods, our differentiable renderer is optimized for 3D heightfields. This is a computationally inexpensive approach that allows us to directly compute gradients in terms of the heights and colors of the bars in the heightfield.

3D Reconstruction. 3D reconstruction is a popular computer vision task, with a variety of deep-learning based approaches being used for this purpose [Maxim and Nedevschi 2021]. Typically these aim for the appearance of the object to be consistent and coherent across views. In contrast, our work aims for the surface of an object to change its appearance when viewed from different vantage points. For example, NeRF achieves very accurate and consistent results by using a neural radiance field representation for scenes
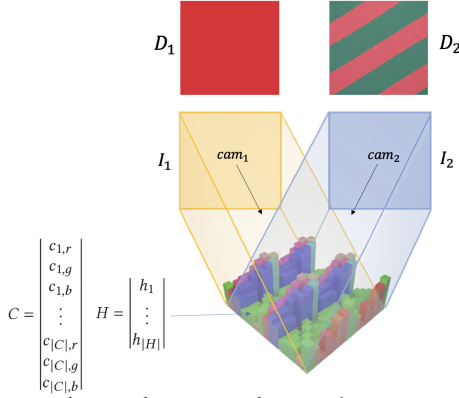
Fig. 2. Diagram of setup of our system for two arbitrary views. D_1 and D_2 are desired appearance images. A heightfield with heights H and colors C are viewed by cameras cam_1 and cam_2 to produce actual appearance images I_1 and I_2.

[Mildenhall et al. 2021] but is not necessarily well-suited for our task, as it does not output an explicit geometry suitable for 3D printing. Algorithms have been proposed to reconstruct 3D meshes from volumetric representations [Lorensen and Cline 1987; Uy et al. 2022], but often suffer from artifacts and introduce an additional phase in the reconstruction process. This is why we instead propose an explicit heightfield optimization approach tailored to creating surfaces with view-dependent appearances.

3 HEIGHTFIELD RENDERING ALGORITHM

We propose a special-purpose differentiable rendering method for 3D heightfields, which we describe below.

3.1 Heightfield and Camera Setup

A heightfield is represented by a height matrix H and color matrix C. The heightfield is viewed by an series of orthographic cameras CAM pointed at the heightfield, as shown in Fig. 2.

$$
H = \begin{vmatrix} h_1 \\ \vdots \\ h_{|H|} \end{vmatrix}, \quad
C = \begin{vmatrix} c_{1,r} \\ c_{1,g} \\ c_{1,b} \\ \vdots \\ c_{|C|,r} \\ c_{|C|,g} \\ c_{|C|,b} \end{vmatrix}, \quad
CAM = \begin{vmatrix} cam_{1,x} \\ cam_{1,y} \\ cam_{1,z} \\ \vdots \\ cam_{|CAM|,x} \\ cam_{|CAM|,y} \\ cam_{|CAM|,z} \end{vmatrix}.
$$

We use weak perspective projections due to the relatively shallow nature of our heightfields. Each camera cam_i is simply parameterized by a 3D viewing direction vector. For each of these viewing directions, there is a given desired image D. Below, we describe the rendering algorithm to find the actual projected image I for each camera.

For each camera, we obtain a matrix of camera rays by tracing from evenly distributed points on the xy plane backwards along the direction of the camera. Each camera ray has a direction vector (d_i, d_j, d_k) and origin (o_i, o_j, o_k).

3.2 Slicing the Heightfield

To render the resulting image for each camera, we slice the heightfield to obtain a cross-section for each camera ray. We do this by projecting the original 3D ray $\mathbf{r}$ onto the xy plane to obtain a 2D projected ray $\mathbf{r}_{xy}$. On the xy plane, we determine which heightfield strips this 3D projected vector intersects with and the corresponding coordinates of each intersection.

We then calculate the distance between each adjacent pair of intersection points. We use these distances to construct a slice represented by a 1D array of strip heights H', a cumulative sum of widths W' and colors C'. We then re-project the original ray onto this slice to retrieve a final projected ray $\mathbf{p}$. The process of projecting a 3D ray to obtain a 2D slice of the heightfield and 2D ray is illustrated in Fig. 3.

3.3 Rendering a Single Pixel

We can determine the color of a single ray by determining which strip of the heightfield slice is hit by the projected ray $\mathbf{p}$ with direction (d_x^p, d_y^p). We first set the origin of $\mathbf{p}$ at $x = 0$ and solve for the corresponding y value at this point, to get a projected ray origin (o_x^p, o_y^p).

We then back-trace along the direction of p from each strip to find strip boundary parameters $Y = (y_0, \ldots, y_{n+1})$ for each strip, as visualized in Fig. 4. We can calculate $y_0, \ldots, y_{n+1}$ from the sliced strip heights $H_{1D} = h_0, \ldots, h_{n+1}$ and cumulative widths $W_{1D} = (w_0, \ldots, w_{n+1})$ by using the invertible function f defined as

$$
y_i = f(i) = h_i - \frac{d_x^p}{d_y^p} \cdot \big((i+1) \cdot w_i\big). \tag{1}
$$

We can determine which strip the ray p hits by looking for an i such that, for the pair of boundary parameters (y_i, y_{i+1}), we have $y_i < o_y^p < y_{i+1}$ as shown in Fig. 4. Note that we first need to make sure that the set of boundary parameters is monotonically increasing by using a cumulative maximum, to deal with the case where a strip is entirely obstructed and should be "skipped" accordingly, as shown in Fig. 5.

We devise a formula for directly obtaining the strip that ray p hits based on the reasoning above. Let $g(y)$ be the Heaviside step function where $g(y) = 0$ if $y < 0$ and $g(y) = 1$ if $y \geq 0$. Then,

$$
y_i \leq o_y^p < y_{i+1} \iff g(o_y^p - y_{i+1}) == 0, g(o_y^p - y_i) == 1 \tag{2}
$$

Thus:

$$
y_i \leq o_y^p < y_{i+1} \iff g(o_y^p - y_i) - g(o_y^p - y_{i+1}) == 1 \tag{3}
$$

Now, as the y boundary values are monotonically increasing, for the ray p with origin height o_y^p we are guaranteed that we will have $y_i \leq o_y^p < y_{i+1}$ exactly once across all pairs (y_i, y_{i+1}); in all other cases, either $o_y^p < y_i, y_{i+1}$ or $o_y^p \geq y_i, y_{i+1}$, in which case the value $g(o_y^p - y_i) - g(o_y^p - y_{i+1}) = 0$ (as the values in this subtraction are 0 or both are 1). Thus we can calculate the color seen by ray p as

$$
\begin{aligned}
color(p) = \ & c_0\big(g(o_y^p - y_0) - g(o_y^p - y_1)\big) \\
& + c_1\big(g(o_y^p - y_1) - g(o_y^p - y_2)\big) \\
& + \ldots \\
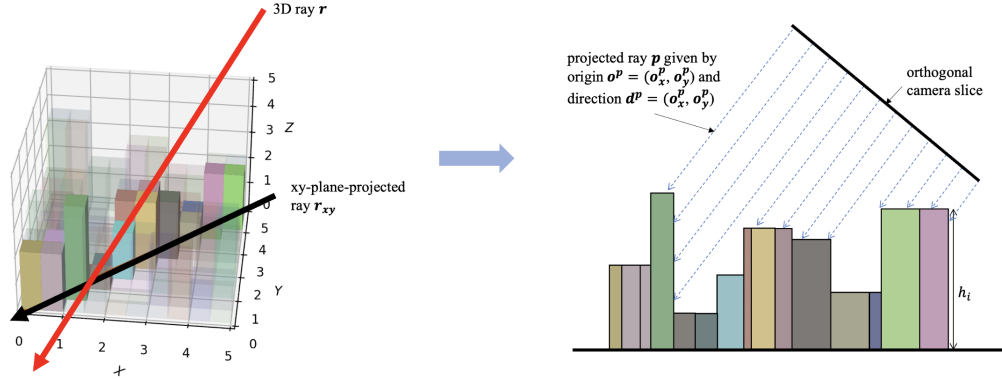& + c_n\big(g(o_y^p - y_n) - g(o_y^p - y_{n+1})\big).
\end{aligned} \tag{4}
$$

Fig. 3. Left: a 3D ray r (red) hitting 3D heightfield, with its projected ray (black) and the strips the projected ray intersects highlighted. Right: the corresponding 2D slice of the heightfield with multiple projected camera rays hitting it.
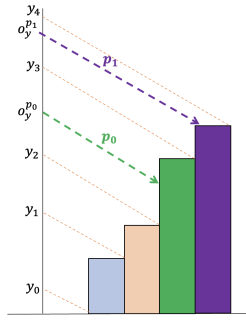

Fig. 4. An illustration of two camera rays hitting the surface of the heightfield. Ray p_0 will be green, as $y_2 < o_y^{p0} < y_3$. Ray p_1 will be purple, as $y_3 < o_y^{p0} < y_4$.

This formula can be rearranged as

$$
\begin{aligned}
color(p) = \quad & g(o_y^p - y_0) \cdot c_0 \\
+ & g(o_y^p - y_1) \cdot (c_1 - c_0) \\
+ & \ldots \\
+ & g(o_y^p - y_n) \cdot (c_n - c_{n-1}) \\
- & g(o_y^p - y_{n+1}) \cdot c_n.
\end{aligned} \tag{5}
$$

We thus have a simple formula in terms of backtraced strip heights $y_0, \ldots, y_n$, strip colors $c_0, \ldots, c_n$, and ray height o_y^p for the view of ray p on a given slice of the heightfield:

$$
\begin{aligned}
color(p) = \quad & g(o_y^p - y_0) \cdot c_0 - H(o_y^p - y_{n+1}) \cdot c_n \\
& + \sum_{i=0}^{n} g(o_y^p - y_i) \cdot (c_i - c_{i-1}).
\end{aligned} \tag{6}
$$

To enable differentiation of the above formula, we use a smooth approximation of the Heaviside step function g. More details of this are provided in Section 4.1.

3.4 Rendering all the pixels

For each camera, we render all of the rays coming from the camera with the process above. Note that we can simplify the process somewhat, as for each camera every column of rays will have the same corresponding heightfield slice, so we do not need to re-slice the heightfield for every camera pixel. These rendered rays form an output image I_i for camera cam_i.

3.5 Optimization Process

Based on the above, the complete process to get from a ray r_i to a view color consists of the following steps:

(1) Project the ray onto the xy axis and obtain the parameters H', W', C', of the corresponding slice of the heightfield and the new "2D" ray $\mathbf{p}$.
(2) Apply the differentiable, invertible function f to all of the strips heights $h'_0, \ldots, h'_n$, to obtain backtraced boundary heights $y_0, \ldots y_1$.
(3) Make the sequence of y_i monotonically increasing, via the function $monotonic(y_i) = \max\{y_0, \ldots, y_i\}$.
(4) Calculate the color of the ray with $color(p)$.

This establishes a completely differentiable forward rendering process to derive the image seen by a camera pointed at the heightfield. Let this mapping be denoted as

$$
I_i = forward(cam_i, H, C). \tag{7}
$$

Repeating Units. There are two general application scenarios of our method: reproducing single images or reproducing repeated patterns. If we wish to reproduce repeated patterns, we optimize for a smaller unit of the heightfield and repeat this unit while considering the occlusion that occurs across units.

4 OBJECTIVE FUNCTION

Our objective is to minimize the pixel-wise MSE loss between the actual appearance of the heightfield obtained from our forward rendering algorithm and the given m by m desired appearance images $D_1, \ldots, D_{|CAM|}$:
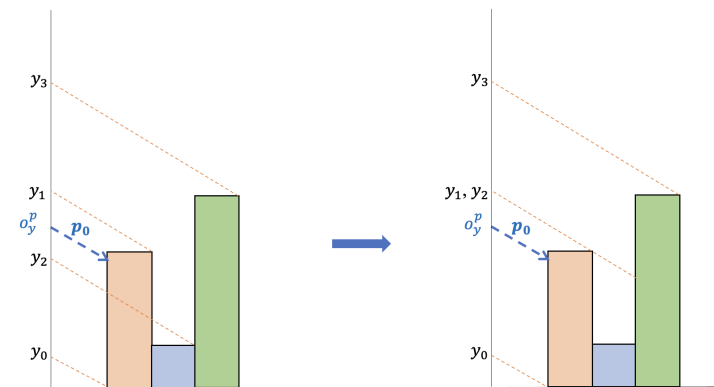
Fig. 5. Left: An example of non-monotonic back-traced strip heights. Right: The same strips back-traced converted into monotonically increasing heights.

Table 1. MSE loss after 100 steps on two-view optimization

camera 1	camera 2	none	barrier	smoothing	barrier smoothing
black	white	**0.066**	0.092	0.087	0.136
black	random	**0.086**	0.105	0.127	0.133
black	stripes	**0.156**	0.162	0.168	0.169
random	stripes	**0.198**	0.249	0.209	0.203
random	random	**0.116**	0.131	0.113	0.124

$$\min_{H,C} \frac{1}{m^2|CAM|} \sum_{i=1}^{|CAM|} \sum_{x=1}^{m} \sum_{y=1}^{m} \left(Di_{x,y} - forward(cam_i, H, C)_{x,y} \right)^2. \tag{8}$$

We also consider several variants on this objective function obtained by adding extra loss terms for surface regularization and different choices for the smooth Heaviside approximation function. We perform an ablation across the regularization variants, which is presented in Table 1. In this ablation, we find that additional loss terms for surface regularization decrease the resulting accuracy of the images on the heightfield's surface. However, this trade-off is necessary to ensure that our surfaces remain within height bounds and do not contain spikes that can easily break or deep troughs that may have extreme shadowing.

4.1 Smooth Heaviside Approximations

The Heaviside step function is not differentiable. We solve this problem by choosing a smooth approximation of this function $\lambda(x, k)$ from a variety of options (outlined in Table 2) with an additional parameter k which increases the amount of smoothing. We perform an ablation study across various estimation functions with fixed $k = 0.1$, comparing the change in loss over time for a surface optimization over two cameras from opposite sides of the surface with 45 degree elevation. The cameras have desired surface appearance of solid white and solid black respectively. The results of our ablation are presented in Fig. 6. Based on this study, we identify the hyperbolic tan (tanh) approximation as the most effective approximation for our algorithm.

Table 2. Heaviside function approximations

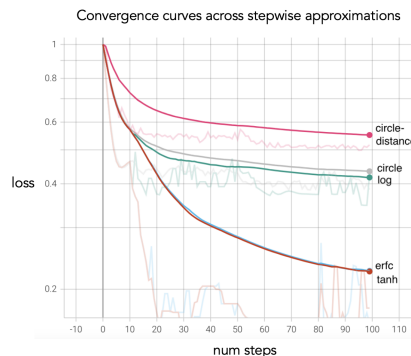| Heaviside approximation	Formula
circle	$\lambda(x, k) = \frac{1}{2} + \frac{1}{2} \frac{x}{(x^2+k^2)^{1/2}}$
circle distance	$\lambda(x, k) \sim \text{Bern}\left(\frac{1}{2} + \frac{1}{2} \frac{x}{(x^2+k^2)^{1/2}} \right)$
erfc	$\lambda(x, k) = \frac{1}{2} + \frac{1}{2} \left(1 - \frac{2}{\sqrt{\pi}} \int_0^{kx} e^{-(kx)^2} dx \right)$
tanh	$\lambda(x, k) = \frac{1}{2} + \frac{1}{2} \tanh(kx)$
log	$\lambda(x, k) \sim \text{Bern}\left(\frac{1}{2} + \frac{1}{2} \tanh(kx) \right)$

Fig. 6. Convergence curves for various smooth approximations for the Heaviside stepwise function. erfc and tanh approximations outperform circle distance, circle and log approximations.

4.2 Regularization

Barrier Loss. We regularize our surfaces by enforcing a minimum height h_{min} and maximum height h_{max} via a barrier loss term:

$$barrier\_loss = -\sum_{i=1}^{|H|} \log(h_{max} - h_i) - \log(h_{min} - h_i). \tag{9}$$

Smoothing. We also add a neighbor loss term for the difference in height between adjacent strips, so that there are no "spikes" in the heightfield:

$$neighbor\_loss = -\sum_{i=1}^{|H|} |h_i - h_{i-1}|. \qquad (10)$$

Table 3. Loss after 100 steps on surfaces with uniform initial configurations, for different combinations of optimizations.

cam 1	cam 2	none	coarse to fine	coord. descent	coarse to fine coord. descent sim. annealing
black	white	0.136	0.067	0.107	**0.011**
random	random	0.124	0.099	0.108	**0.093**
black	random	0.133	0.108	0.094	**0.051**
black	stripes	0.169	0.124	0.141	**0.037**
random	stripes	0.203	0.205	0.180	**0.057**

5 OPTIMIZATION METHODS

Our basic optimization algorithm uses the Adam Optimizer [Kingma and Ba 2015] to minimize the MSE loss between the desired views and actual appearance of the surface for each camera. To design our surfaces, we introduce a suite of various optimization techniques. We perform an ablation across these methods, which is presented in Table 3. We find that a combination of coarse-to-fine optimization, alternating block coordinate descent, and simulated annealing significantly improves the performance of our algorithm in all test cases.

5.1 Coarse-To-Fine Optimization

We use coarse-to-fine optimization, whereby every 50 steps of the algorithm, each of the heightfield strips is subdivided into a 2×2 block of strips with the same height and color as the original strips. We begin the optimization process with heightfields of 8×8 strips, and end the process with heightfields of 32×32 strips. This provides some additional surface regularization and also improves the overall performance of the algorithm.

5.2 Alternating Block Coordinate Descent

During optimization, we alternately update heights of the strips for 10 steps and the colors of the strips for 20 steps, until convergence. This greatly improves the performance of our algorithm relative to optimizing both heights and colors simultaneously at each iteration.

5.3 Simulated Annealing

To improve our results for more complicated cases and better escape local optima, we optionally perform steps of simulated annealing [Bertsimas and Tsitsiklis 1993], as outlined in algorithm 1, on both the strip heights and colors at the start of the algorithm and subsequently every 100 steps. We set $T_{max} = 3$, $T_{min} = 0.5$ and cool T using exponential decay with a factor of 0.99. If a more uniform final pattern is desired, then simulated annealing can be omitted from the optimization process.

Algorithm 1 Simulated Annealing [Bertsimas and Tsitsiklis 1993]

$T \leftarrow T_{max}$
$C, H \leftarrow \text{init\_C, init\_H}$
while $T > T_{min}$ **do**
    $C_{rand}, H_{rand} \leftarrow \textbf{RANDOM\_NEIGHBOUR}(T, C, H)$
    $\Delta L \leftarrow \textbf{LOSS}(C_{rand}, H_{rand}) - \textbf{ENERGY}(C_{rand}, H_{rand})$
    if $\Delta L < 0$ **then**
        $COL, H \leftarrow C_{rand}, H_{rand}$
    else if RANDOM() $< e^{-\Delta L/T}$ **then**
        $C, H \leftarrow C_{rand}, H_{rand}$
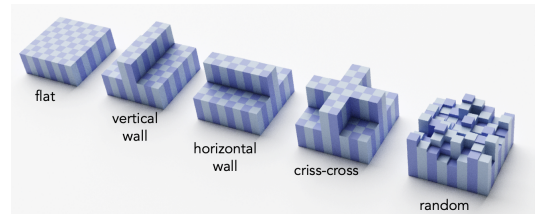 $T \leftarrow \textbf{COOL}(T, C, H)$
return C, H

Fig. 7. Renders of 8×8 initial surface configurations (left to right: flat, vertical wall, horizontal wall, cross, random).
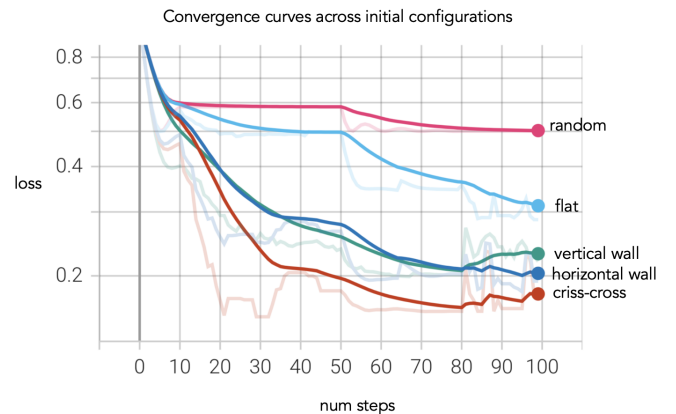

Fig. 8. Comparison of convergence curves across different initial surface configurations.

5.4 Initial Configurations

We experiment with several different initial configurations for our optimization, including uniform, random, and preset configurations, as illustrated by Fig. 7. We compare the convergence curves when using these initial configurations for the four-view optimization problem (with elevation 30 and desired appearances of solid cyan, magenta, yellow and black). The results of this experiment are shown in Fig. 8. We find that using a criss-cross initial configuration pattern

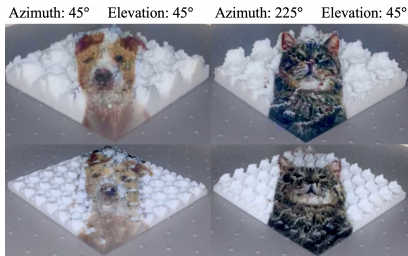Azimuth: 45°   Elevation: 45°   Azimuth: 225°   Elevation: 45°

Fig. 9. The same surface as viewed from two directions displaying an image of a dog (left) and cat (right), printed with higher bar width (top) and lower bar width (bottom). The images on the bottom row suffer from more color mixing due to translucency.

gives the best performance, closely followed by vertical and horizontal wall configurations (which, aside from randomness, should theoretically all have the same effect for this four-view problem).

5.5 Post-optimization projection

As an additional step after optimizing the heights and colors of the field, we create multiple vertical segments for each heightfield bar, and then re-project the desired images onto the heightfield. This allows us to achieve higher resolution results without additional computation time, which is particularly important for detailed images.

6 OPTIMIZATION RESULTS

We implemented the optimization algorithm using *PyTorch*, and the fabrication mesh conversion script in Blender with the Python *bpy* module. Optimization times range from 30 minutes to 2 hours.

We perform optimizations for various test cases, with varying numbers of cameras, viewing angles and desired appearance. In Fig. 10, we show surfaces optimized for five views (top) and four views (bottom). The results closely match the desired appearance, but we can see some quality degradation occurring when introducing the fifth view. We show additional rendered heightfields in our video.

We also carry out a study to determine which pairs of viewing angles are compatible with each other. We examine the loss after 100 steps on a two-view (solid black vs. solid white) optimization. In Fig. 13a both views have the same elevation and different azimuth, and in Fig. 13b both views have the same azimuth and different elevation. We find that as difference in elevation and azimuth increases beyond 40 degrees and 60 degrees respectively, we are able to achieve a final MSE loss of < 0.1.

Additionally, we compare our approach to a lenticular-based approach [Zeng et al. 2021], and present renders of resulting surfaces in Fig. 11 that reflect the highest possible resolution for each approach with the same 3D printer. We find that our approach is qualitatively better at capturing sharp lines and high-resolution details from the desired appearance images, although it does suffer slightly more from color contamination across views. We also note that our method displays a smaller crop of the desired appearance image than the lenticular surface, due to the need for cropping during image projection onto the heightfield surfaces.

7 FABRICATION

To fabricate the surfaces, we first convert the heightfields into meshes and material template libraries using Blender. We then fabricate the surfaces using a Stratasys J55 Polyjet UV printer. Results are shown in Fig. 9 and Fig. 12. An example of a rendered and fabricated version of the same surface can be seen in Fig. 14a, which also shows a close match between rendered and real results. We identified the best workable resolution for the printer by performing several strip tests, and found that 300dpi was possible without aliasing. We used this resolution as the minimum strip width for our fabrication. We are able to achieve view-dependent effects at a very high resolution as can be seen in Fig. 14b, which is less than 1cm wide. In order to prevent strip breakage for such high-resolution prints, we added a layer of ultra-clear printed material on top of this surface. This layer, however, reduces the actual elevation required for the desired surface appearance due to refraction.

Some results also suffer from color mixing across views due to undesired printing material translucency, such as in Fig. 9. Our current approach to solve this is to reduce the printing resolution for a given target. Future work may also regularize the heightfield by minimizing high color variance between neighboring bars to further mitigate the negative effects of material translucency without having to compromise on printing resolution.

8 DISCUSSION AND LIMITATIONS

Our algorithm explores the use of self-occluding heightfields in fabricating multi-image displays. We demonstrate generating surfaces with view-dependent appearance at up to five distinct viewing angles and fabricate surfaces that closely match the rendered results at high resolution and with up to four viewing angles. The most important advantages of our method as opposed to prior works are the high working resolution and ability to share colors across views. We are also able to use bright colors on our surfaces, as we do not rely on additional walls that darken the overall surface appearance as in Sakurai et al. [2018]. However, we observe that the quality of results from our procedure is highly dependent on the relative azimuth and elevation of the cameras, unlike with lenticular-based methods. Additionally, we do not take the self-shadowing of environment lighting into account. Future work could, however, augment our algorithm to account for self-shadows by darkening or lightening the colors in the heightfield to offset these effects.

9 CONCLUSION

We devise a novel approach for fabricating multi-image displays that does not rely on building fixed-color walls or using lenses and polish. We present a suite of techniques that comprise a new optimization algorithm specifically tailored to our task, including a simple differentiable ray-casting inspired rendering algorithm designed to render colored heightfields to achieve our task. Our approach allows us to use a UV printer to successfully fabricate colorful 3D objects whose surface-appearance changes depending on viewing angle.

REFERENCES

Marc Alexa and Wojciech Matusik. 2011. Images from self-occlusion. In *Proceedings of the International Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging.* 17–24.

Marc Alexa and Wojciech Matusik. 2012. Irregular pit placement for dithering images by self-occlusion. *Computers & Graphics* 36, 6 (2012), 635–641.

Dimitris Bertsimas and John Tsitsiklis. 1993. Simulated annealing. *Statistical science* 8, 1 (1993), 10–15.

Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. 2020. Differentiable rendering: A survey. *arXiv preprint arXiv:2006.12057* (2020).

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proc. ICLR.*

Oliver Klehm, Ivo Ihrke, Hans-Peter Seidel, and Elmar Eisemann. 2014. Property and lighting manipulations for static volume stylization using a painting metaphor. *IEEE Transactions on Visualization and Computer Graphics* 20, 7 (2014), 983–995.

Eunsil Lee, Jae-Hwan Kang, Jun-Dong Cho, and Junsuk Kim. 2022. Touch the color change: Representation of color change using tactile grating patterns. *Psychology of Aesthetics, Creativity, and the Arts* (2022).

Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. 2019. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 7708–7717.

William E Lorensen and Harvey E Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. *ACM siggraph computer graphics* 21, 4 (1987), 163–169.

Bogdan Maxim and Sergiu Nedevschi. 2021. A survey on the current state of the art on deep learning 3D reconstruction. In *2021 IEEE 17th International Conference on Intelligent Computer Communication and Processing (ICCP).* IEEE, 283–290.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

Thomas Klaus Nindel, Tomáš Iser, Tobias Rittig, Alexander Wilkie, and Jaroslav Křivánek. 2021. A gradient-based framework for 3D print appearance optimization. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–15.

Edwin Olson. 2011. AprilTag: A robust and flexible visual fiducial system. In *2011 IEEE international conference on robotics and automation.* IEEE, 3400–3407.

Marios Papas, Wojciech Jarosz, Wenzel Jakob, Szymon Rusinkiewicz, Wojciech Matusik, and Tim Weyrich. 2011. Goal-based caustics. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 503–511.

Hao Peng, Lin Lu, Lin Liu, Andrei Sharf, and Baoquan Chen. 2019. Fabricating QR codes on 3D objects using self-shadows. *Computer-Aided Design* 114 (2019), 91–100.

Thiago Pereira, Carolina LA Paes Leme, Steve Marschner, and Szymon Rusinkiewicz. 2017. Printing anisotropic appearance with magnetic flakes. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–10.

Felix Petersen, Bastian Goldluecke, Christian Borgelt, and Oliver Deussen. 2022. GenDR: A Generalized Differentiable Renderer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 4002–4011.

Petar Pjanic and Roger D Hersch. 2015. Color changing effects with anisotropic halftone prints on metal. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 1–12.

Christian Regg, Szymon Rusinkiewicz, Wojciech Matusik, and Markus Gross. 2010. Computational highlight holography. *ACM Transactions on Graphics (TOG)* 29, 6 (2010), 1–12.

Helge Rhodin, Nadia Robertini, Christian Richardt, Hans-Peter Seidel, and Christian Theobalt. 2015. A versatile scene model with differentiable visibility applied to generative pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision.* 765–773.

Kaisei Sakurai, Yoshinori Dobashi, Kei Iwasaki, and Tomoyuki Nishita. 2018. Fabricating reflectors for displaying multiple images. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–10.

J Kenneth Salisbury Jr. 1999. Making graphics physically tangible. *Commun. ACM* 42, 8 (1999), 74–81.

Carlo H Séquin. 2013. Making graphics tangible. *Computers & graphics* 37, 3 (2013), 148–164.

Xavier Snelgrove, Thiago Pereira, Wojciech Matusik, and Marc Alexa. 2013. Parallax Walls: Light fields from occlusion on height fields. *Computers & graphics* 37, 8 (2013), 974–982.

Mikaela Angelina Uy, Yen-Yu Chang, Minhyuk Sung, Purvi Goel, Joseph G Lambourne, Tolga Birdal, and Leonidas J Guibas. 2022. Point2Cyl: Reverse Engineering 3D Objects From Point Clouds to Extrusion Cylinders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 11850–11860.

Yitzhak Weissman. 2015. 3D Lenticular Imaging for Art. In *Proceedings of Bridges 2015: Mathematics, Music, Art, Architecture, Culture.* 507–510.

Tim Weyrich, Pieter Peers, Wojciech Matusik, and Szymon Rusinkiewicz. 2009. Fabricating microgeometry for custom surface reflectance. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 1–6.

Jiani Zeng, Honghao Deng, Yunyi Zhu, Michael Wessely, Axel Kilian, and Stefanie Mueller. 2021. Lenticular objects: 3D printed objects with lenticular lens surfaces that can change their appearance depending on the viewpoint. In *The 34th Annual ACM Symposium on User Interface Software and Technology.* 1184–1196.
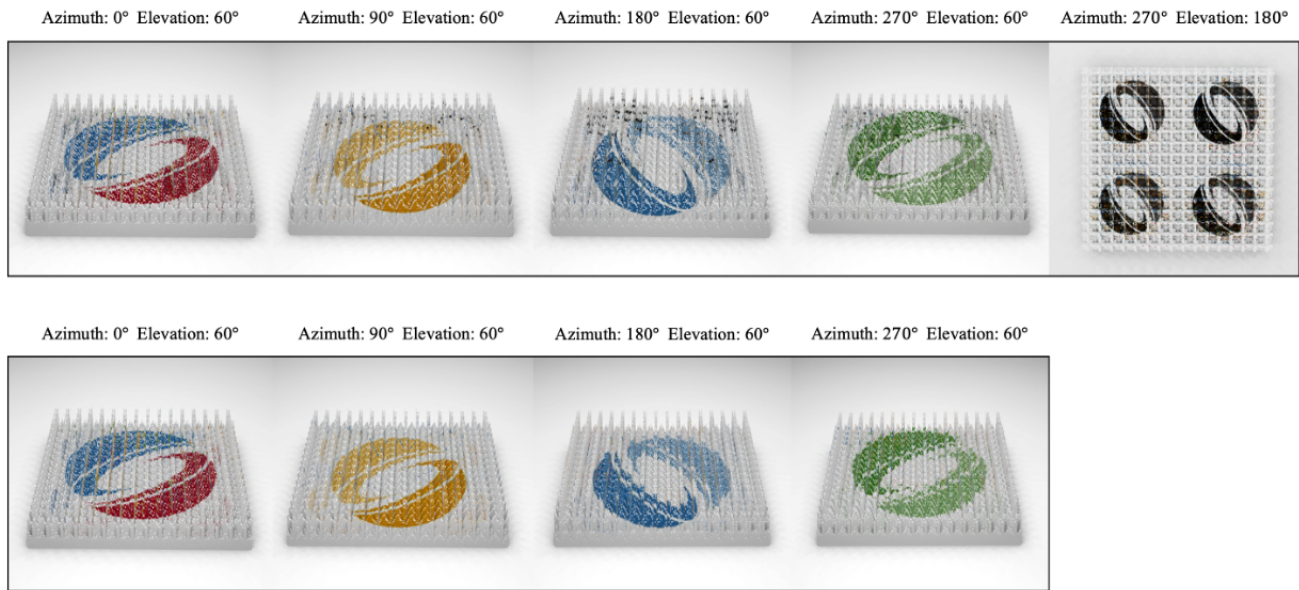
Fig. 10. Top: Optimized surface with five distinct desired appearances. Bottom: Optimized surface with four distinct desired views.
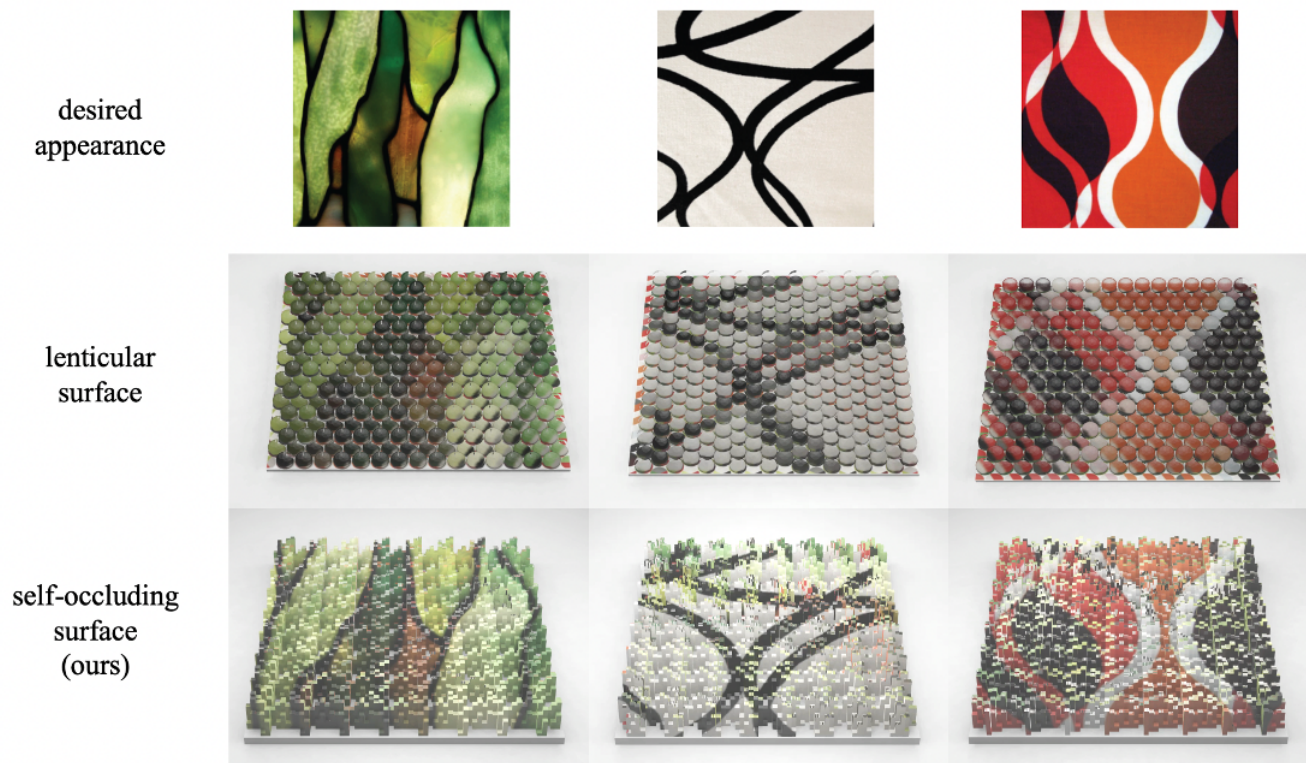

Fig. 11. Top: desired appearances. Middle: Renders of lenticular surface from three different viewing directions. Bottom: Render of self-occluding surface generated via our method from three different viewing directions.
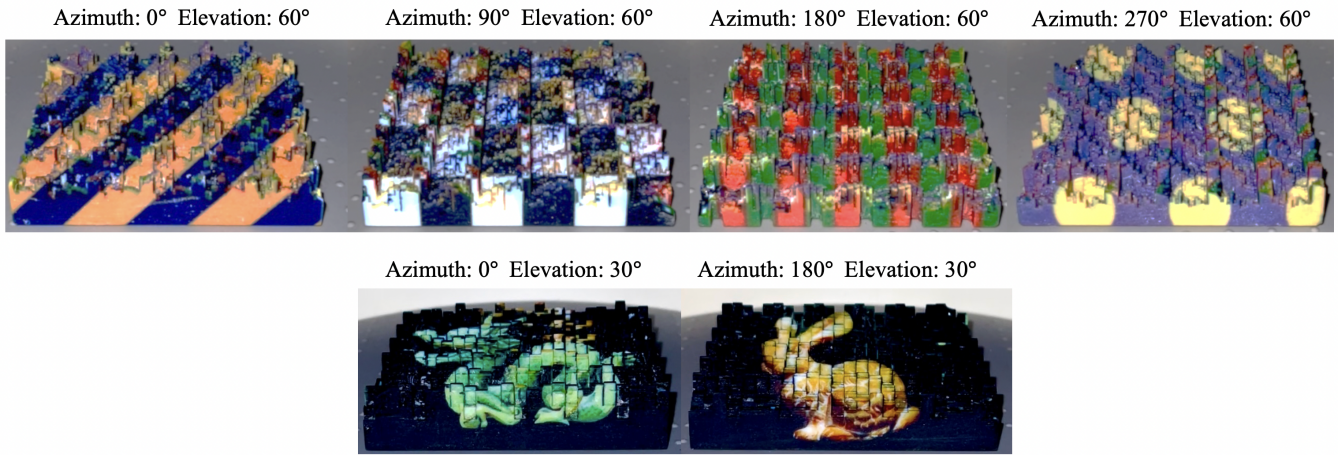
Fig. 12. Photographs of various fabricated surfaces. Top row: The same surface as viewed from four directions, displaying four unrelated colorful patterns. Bottom row: The same surface as viewed from two directions, displaying the Stanford Dragon and the Stanford Bunny.
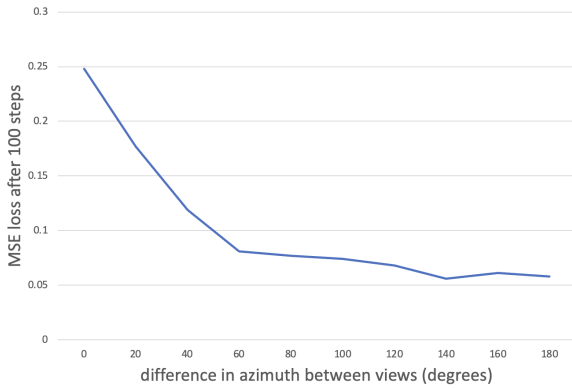

Fig. 13a. MSE loss after 100 steps with two viewing directions. The first viewing direction has a desired appearance of solid black, with elevation 60 and azimuth 0. The second viewing direction has a desired appearance of solid white, elevation 60 degrees and variable azimuth.
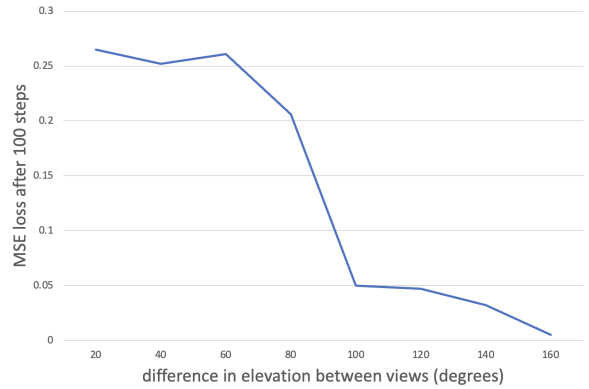
Fig. 13b. MSE loss after 100 steps with two viewing directions. The first viewing direction has a desired appearance of solid black, with elevation 20 and azimuth 0. The second viewing direction has a desired appearance of solid white, variable elevation and azimuth 0.
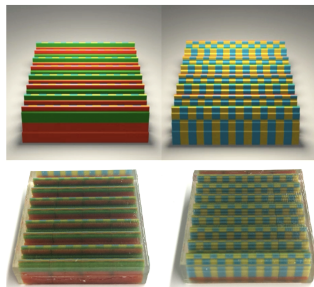

Fig. 14a. Top: Render of synthetic surface as viewed from two directions. Bottom: Fabricated version of this same surface.
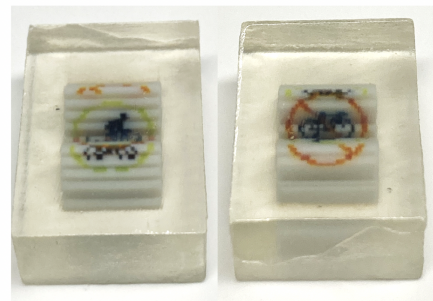
Fig. 14b. High-resolution fabrication of a biking/no biking sign as viewed from two high-elevation directions (60 degrees). The surface is less than 1cm wide.